# *MeterGoat*: A Low Cost Hardware Platform for Teaching Smart Meter Security

Jefferson Capovilla, Nelson Uto
GTSIC – Information Security Department
CPqD
Campinas, São Paulo, Brazil
{jrodrigo, uto}@cpqd.com.br

Danilo Suiama, Jose Resende
Management of Metering, Losses and Technology
ELEKTRO
Campinas, São Paulo, Brazil
{danilo.suiama, jose.resende}@elektro.com.br

***Abstract*—Smart meters play an important role in smart grid architectures by enabling best operational efficiency, enhanced usage monitoring and variable pricing structure, among other advantages. On the other hand, meters have often been deployed with several security vulnerabilities that can compromise the mentioned benefits and result in cyber attacks. Therefore, teams involved in the development of smart meters should be trained in security aspects related to the area. In this context, this paper describes the work in progress project called *MeterGoat*, whose main objective is to develop an open source and low cost hardware platform for teaching smart meter security. In order to achieve this purpose, it deliberately implements meter's functionalities in a vulnerable way, providing a flexible framework for smart meter pentesting including hardware and firmware weaknesses.**

***Keywords- Smart meter pentesting; trainning platform; cyber attacks; vulnerability.***

## I. INTRODUCTION

Smart grids have the objective of transforming today's power grid by providing an extra level of grid status control, energy consumption profiles, powerful control mechanisms, and flexible billing processes. However, these advantages increase the overall complexity of the system, since it is necessary to use more advanced components, in order to maintain and transmit the information required by the utility companies. A major drawback of this new scenario is the high number of vulnerabilities reported on smart meters and the underlying infrastructure, as can be seen in the papers of Skopik et al. [1] and Carpenter et al. [2].

Clearly, there is a gap between the energy sector and information technology field with respect to information security. The main objective of this work in progress paper is then to fill this gap by introducing a low cost and open source hardware platform for teaching security aspects that must be considered in the implementation of a smart meter.

We named it *MeterGoat*, after the Open Web Application Security Project (OWASP) *WebGoat* project [3], which has the same purpose, but in the scope of web applications. *MeterGoat* will then implement the most relevant smart meter's functionalities in a vulnerable way, allowing the students to perform real attacks without affecting live environments. In this way, they can fully understand why the exploits are possible and avoid making the same mistakes in real products they develop.

The remaining part of this paper is structured as follows: Section II presents a few works related to security training platforms and smart meter penetration testing. Section III describes the *MeterGoat* project, outlining the vulnerabilities that are being implemented and the hardware components we are using. In Section IV, we list a few tools to compose the toolkit for testing smart meter security. Section V gives an overview of training scenarios, and, finally, Section VI concludes the paper and discusses future work.

## II. RELATED WORK

The recent demand of countries in substituting standard energy meters by the smart version contributes to foster research in Advanced Metering Infrastructure (AMI) security, such as the AMI Penetration Test Plan [4], by the National Electric Sector Cybersecurity Organization Resource (NESCOR). That document proposes that the security evaluation of smart meters should cover four main areas: embedded devices, network communications, server Operating System (OS), and server applications.

The work of Grand [5] complements the aforementioned guideline, since, besides testing, it also covers the concept of designing secure hardware for embedded systems, introducing, in this way, security in the earlier phases of development lifecycle. According to the author, in order to have a more secure product, one needs, at least, to avoid vulnerabilities in the enclosure, the circuit board, and the firmware. In our project, we are using several examples of insecure design given by Grand.

Regarding security training, one can find several open source projects, which implement insecure web and mobile applications, such as *WebGoat*, *iGoat* [6], and *GoatDroid* [7]. However, to the best of the author’s knowledge, there is no testbed related to smart meters, making it hard for those interested in the area to learn how to test this type of device. Thus, we believe that *MeterGoat* can help to fill this gap.

As we already mentioned, *Webgoat* is a deliberately insecure web application, designed to teach penetration testing. It contains more than 30 lessons that emulate vulnerabilities commonly found in real applications. The *iGoat* and the *GoatDroid* are quite similar projects, presenting themselves as fully functional and self-contained security training environments, for iOS and Android, respectively. All three projects are maintained by OWASP.

## III. THE *METERGOAT* PROJECT

This section presents the functional requirements of the *MeterGoat* Project, as well as preliminary project decisions.

### A. *Hardware Architecture of Commercial Smart Meters*

The hardware architecture of *MeterGoat* is based on commercial smart meters that, in general, are composed of the elements showed in Fig. 1: current and voltage sensors, Analog-to-Digital Converters (ADC), Central Processing Unit (CPU) for metrology calculations, Random-Access Memory (RAM) for volatile data and firmware execution, non-volatile storage, e.g. flash and Electrically-Erasable Programmable Read-Only Memory (EEPROM), for firmware and data storage, and communication peripherals for maintenance and update procedures.

The majority of the microcontrollers contain integrated memory in the System on Chip (SoC) itself. However, if system functionalities require more memory than the default amount, external memory chips can be used to increase the overall capacity. Architectures that use microcontrollers generally comply with low cost, low power, and compact design requirements. On the other hand, systems that demand high performance employ dedicated processors and components, resulting in a more expensive device. Since smart meters are produced in high volumes, the first, cheaper, architecture is commonly adopted.

Anti-tampering mechanisms comprise a layer of defense against a possible integrity violation of an equipment. In this way, they must cope with any attempt, physical or electronic, of adulteration, such as opening the device's case, accessing internal memory, and replacing components. These controls, explained in detail in [5], are usually categorized into four groups: resistance, evidence, detection, and response.

The Joint Test Action Group (JTAG) Debug, illustrated in Fig. 1, provides a direct connection to most of the components of a SoC. One can use it, through the JTAG interface, to perform memory programming, boundary scanning, and on-chip debugging. Although the security best practice is to disable this interface, by physically damaging the Printed Circuit Board (PCB) connection or by blowing the JTAG fuse, several meters do not implement this recommendation.

It is important to note that the architectural diversity, presented in this section, determines the types of hardware vulnerabilities that one may encounter on these devices, since it provides different levels of access to metrological information and code.

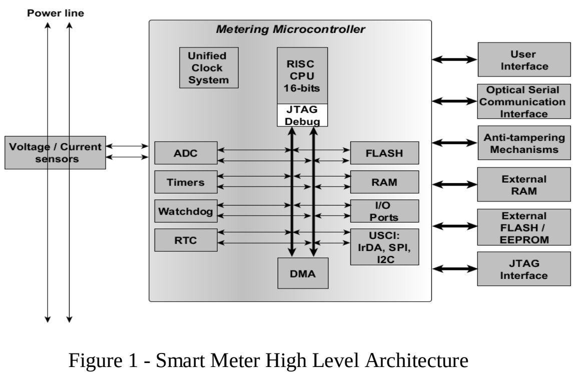

Figure 1 - Smart Meter High Level Architecture

### B. *Hardware Architecture of MeterGoat*

The *MeterGoat* architecture, depicted in Fig. 2, derives from the models discussed in the previous section. We made some simplifications in order to reduce the project final cost and to obtain a modularization of the training platform, while still implementing the main hardware and software security weaknesses commonly found on real devices [8]. We briefly explain the main components of our project below:

#### 1) *Microcontroller MSP430F5438 and Development Kit*

Based on the architecture illustrated in Fig. 2, we selected the Texas Instruments (TI) MSP-EXP430F5438 [9] development kit and the corresponding microcontroller, MSP430F5438 [10], for building *MeterGoat*. The later has, as main features, a 256KB internal flash memory, 16KB internal Static RAM (SRAM) memory, two Serial Peripheral Interface (SPI) interfaces, and 87 Input/Ouput (I/O) pins.

The microcontroller above contains a hardware multiplier that performs signed and unsigned operations with 8, 16, 24, and 32-bit operands. This feature allows the optimization of some metrologic operations, and, for this reason together with the aforementioned characteristics, several smart meters are built over this specific model or similar members from the same family of microcontrollers.

The development kit also comes with a 138x110 grayscale Liquid Crystal Display (LCD) for measurement display, a JTAG interface, a 5-position joystick, and two push buttons for user interface control. Finally, it can be powered in three different ways: Universal Serial Bus (USB), Flash Emulation Tool (FET), and two AA batteries.

#### 2) *External Components*

Since the objective of the *MeterGoat* project is not to build a real smart meter, we are not using a high-priced energy metering Integrated Circuit (IC) such as the ADE7758 [11]. Instead, we are employing a programmable pulse generator, in order to emulate the signal from the voltage/current sensor, responsible for measuring energy consumption, and sample it using a digital I/O pin. One of the ways to generate such signal consists in assembling

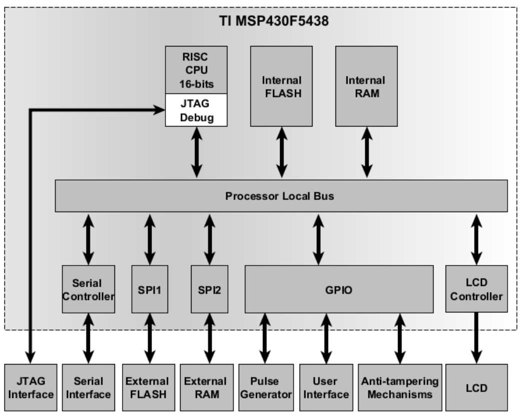

Figure 2 - *MeterGoat* High Level Architecture

discrete components and a LM555 timer [12] in a breadboard or using an Arduino Uno microcontroller board [13].

We also use a breadboard to assemble and power the external Flash and RAM memories. These components communicate with the microcontroller using an SPI interface.

*3) Anti-tampering Mechanisms*

We are implementing detection and response anti-tampering mechanisms in *MeterGoat*. The former includes: (i) anti-tampering switches, which one normally places on the meter's case so they trigger when someone attempts to open the chassis; (ii) temperature gradient measurement, against attacks that freeze components in order to take advantage of data remanence. This countermeasure will use the internal temperature sensor included in MSP430F5438.

With regard to response mechanisms against tampering, *MeterGoat* will implement passive zeroization, which involves disconnecting power from volatile memory so that content is lost, and active zeroization, which securely erase sensitive data used by the system.

## C. Exposed Interfaces

In order to perform an attack, it is fundamental to initially identify the target's surface attack. In the case of smart meters, one must consider external configuration ports, network interfaces, buses, and electronic component pins, as explained in the present section.

*1) JTAG Interface*

JTAG is a standard interface composed at minimum by four pins, Test Mode Select (TMS), Test Data In (TDI), Test Data Out (TDO), and Test Clock (TCK), that is commonly used for interfacing with circuit boards, microprocessors, and several other peripherals for debugging purposes. In microcontrollers, the JTAG is also used to upload and store the firmware in internal flash memory. As showed in Fig. 2, by having access to an enabled JTAG interface, it is possible to perform attacks such as memory dumps and firmware extraction. For these reasons, the JTAG interface must always be disabled in final products, otherwise an attacker can dump and modify the component's contents.

*2) External Flash/EEPROM Memory*

These are discrete chips with the exclusive function of non-volatile storage, usually providing no protection whatsoever. When a smart meter architecture includes this type of element in the project, it is possible to dump or modify the entire memory content, that may contain measurement data, configuration parameters, or the firmware. This task can be performed with the chip soldered on the PCB, if it uses the Inter-Integrated Circuit ($I^2C$) communication protocol (accepting multi-master on the bus). Otherwise, the chip needs to be desoldered for direct pin manipulation using the tools presented in Section IV.

*3) External RAM*

Consists in discrete chips with the exclusive function of volatile storage, usually providing no native protection against an attacker with physical access to them. When a smart meter architecture includes this kind of element in the project, it is possible to dump the memory addresses accessed by the microcontroller through bus snooping. This task can be done using the tools presented in Section IV.

*4) Optical Interface*

The optical port is often used by a smart meter for configuration purposes. It is based on short range communication, making vendors wrongly assume it is less prone to sniffing and adulteration. A weak access control system may allow an attacker to send commands to the meter by this type of interface.

*5) Network Interface*

A smart meter can use a wired or a wireless network to communicate with the central system or aggregators. If a secure communication protocol is not employed, it is possible to capture and alter information in transit as well as inject packets.

*6) Communication Interface*

A smart meter can have several communication interfaces such as USB, $I^2C$, Universal Asynchronous Receiver/Transmitter (UART), Controller Area Network (CAN), SPI, etc., which do not provide any tamper protection. Therefore, an attacker can use them to analyze information being transmitted, dump accessible data, and disturb the communication by injecting invalid packets or random data.

## D. Vulnerabilities on MeterGoat

In the first version of *MeterGoat*, we are implementing all the vulnerabilities explained below:

- **Unprotected interfaces –** the exposed interfaces described in Section III.C allows an attacker to dump information from the meter as well as to send commands to be performed by the device.
- **Broken cryptographic algorithms –** this class of vulnerability includes the use of home-made algorithms and classical cryptosystems, such as the shift cipher and Vigenère's, for which cryptanalysis is possible.
- **Incorrect use of cryptograph –** includes Electronic CodeBook (ECB) block cipher mode of operation for large messages, binary additive stream cipher with key reuse, and use of a cryptographic mechanism for a purpose different than the originally intended [14].
- **Insecure cryptographic key management** – consists in using predictable keys or known weak keys such as those for DES [14], embedding cryptographic keys in code, storing cryptographic keys in cleartext files, the lack of use of a secure static memory for key protection, and the absence of a functionality for key substitution.
- **Unprotected data at rest –** consists in the storage of sensitive information in cleartext form and the use of an encoding mechanism such as Base64 instead of a cipher.
- **Insecure communication –** encompasses cleartext communication between client software and the meter or through a protocol with known vulnerabilities such as the one described in [15].

- **Broken user authentication mechanism –** includes weak or absent password policy, flawed authentication protocol, and insecure storage of passwords.
- **Authorization flaws –** this type of problem arises when the concept of a reference monitor is not properly implemented, resulting, for instance, in privilege escalation attacks and direct accesses to resources.
- **Lack of integrity mechanism –** the absence of such a security mechanism allows an attacker to replace the firmware, altering or inserting functionalities, and to tamper with metrology data.
- **Firmware implementation flaws –** involve classical attacks such as buffer overflow, format string attack, and integer overflow.
- **Flawed anti-tampering mechanism** – although *MeterGoat* will not have a case at all, the idea of this exercise is to show how a bad casing could be explored to bypass anti-tampering controls.

## IV. TRAINING TOOLKIT

The training toolkit comprises a selection of tools that can be used in the evaluation of smart meter security. One should not assume, however, ours is the only possible list, since there are similar tools available for choosing. If one decides to build its own set, one should select those that cover as many features as possible.

We divided our list in three groups: *interface identifier*, *interface manipulator*, and *software*. The former aids in identifying pins and test points, verifying pin voltage, and monitoring signal transitions. In this set, we recommend the tools below:

- **Multimeter** - is an electronic instrument that combines several measurement functions in a single unit. Since it can measure a wide range of voltages without being damaged, it is a robust equipment for measuring pins and test points for the first time. It is also used to find the connections between components by using the continuity test feature. When two points are electrically connected, a tone is emitted.
- **Oscilloscope** - is used to observe the change of an electrical signal over time. For reverse engineer, the purpose is to verify signal transitions to check the communication between the microcontroller and other components for later bus sniffing.
- **JTAGgulator** - is an open source hardware tool that assists in identifying JTAG pin (TMS, TDI, TDO, TCK, TRST) connections from test points, vias, or component pads on a target device [16].

The interface manipulator tools aid in obtaining information stored in memory or transmitted through interfaces and buses. In this category, we suggest the following tools:

- **Bus Pirate** - performs serial bus manipulation. Supports many serial protocols at 0-5.5 volts such as 1-Wire, $I^2C$, SPI, and asynchronous serial [17].
- **Logic Analyzer** - performs bus sniffing by recording, viewing, and measuring digital signals in transit between components. It can understand different protocols including serial, $I^2C$, SPI, and CAN.
- **GoodFET** - is an open source JTAG adapter, used for TI MSP430 as a debugger and flash emulation tool [18].
- **MSP-FET430UIF** - is the official equipment provided by TI to MSP430 for JTAG debugging and flash emulation tool [19].

Finally, in order to perform firmware analysis, one needs a disassembler and a debugger, which support the platforms employed by the smart meter. One of the best tools in this category is the powerful IDA, but it has the disadvantage of being relatively expensive. Unfortunately, no open source counterpart works with the very specific processors used by meters. Moreover, there is no disassembler at all for some architectures, and, then, one would have to be built.

## V. TRAINING SCENARIOS

The training scenarios we propose in this section are based on references [4] and [5] and on security weaknesses found in tests we performed against commercial smart meters. Since security is a dynamic area, with new vulnerabilities being discovered every day, the list of lessons may be updated in future versions.

In order to make the most of the training, the student must have basic knowledge of the following topics: electrical circuits, communication protocols (e.g., $I^2C$, SPI, and serial), embedded systems, assembly and Python languages, software reverse engineering, and cryptography.

Each lesson of the course covers the analysis of a different part of a smart meter, resulting in four groups [4]: (1) electronic components; (2) field technician interface; (3) binary firmware; and (4) cryptographic mechanisms.

In the first part of the training, the students learn how to identify the main components and to read the corresponding datasheets, in order to map pins and understand their functionalities. After that, we proceed to the reverse engineer of the PCB, by identifying, with the help of a multimeter and an oscilloscope, connections among components, operating voltages, and signal transitions. Techniques to identify and bypass a simple tamper detection mechanism are taught using a multimeter. The idea is to show how to detect a logic ‘1’ pin and to employ an extra wire, with the purpose of keeping the signal high for the port responsible for tamper monitoring.

The lecture ends with dumping of data from the non-volatile memories and snooping of the bus connecting the memory modules to the microcontroller, respectively, by using Bus Pirate and Logic Analyzer. *MeterGoat* will store and transmit information protected by a multitude of mechanisms, so the students can practice string analysis, entropy analysis, data decoding, and systematic key search.

For the second part of the course, *MeterGoat* provides a serial port as a field technician interface. The instructor will teach how it can be used to interact with the equipment, using standard protocols, and how to implement these with the Python language. This will be the base for teaching protocol fuzzing and vulnerability exploitation.

The third group of lessons comprises binary firmware extraction and analysis, through the manipulation of the JTAG interface. The students will use GoodFET or MSP-FET430UIF, in order to dump the simple firmware stored in microcontroller's internal memory of *MeterGoat*. Since reverse engineering requires very advanced knowledge, we will cover only the disassembly and analysis of a short piece of authentication code.

Although advanced cryptanalysis is beyond the scope of a course about smart meter security, one needs to know how to identify basic mistakes in the implementation of cryptographic mechanisms. For this reason, the lessons of the fourth group are related to the most common vulnerabilities in this area, mainly those resulting from improper key management.

## VI. Conclusions and Future Work

This paper presented the work in progress project called *MeterGoat*, whose main objective is to provide a low cost platform for smart meter security training. The platform will provide most of the functionalities and interfaces of real smart meters, all implemented in a vulnerable way. It is important to mention that we do not intend to build a real meter. Thus, some of the functions will just be emulated, but in a way that allows the student to learn about a given security vulnerability we want to stress. The framework we defined is very flexible and one can easily extend it to include new types of vulnerabilities or variations of old weaknesses.

Currently, we have already specified the full platform, selected the list of components, and started the hardware assembly and coding of the chosen firmware vulnerabilities. We expect to spend six more months in this task, and, once finished, we intend to provide the schematics and firmware as an open source project. In this way, engineers and security analysts will be able to build and use *MeterGoat*, totally free of charge, for personal use, provided the user license be respected.

The estimated cost for building the platform lies under US$200, which is reasonable for this type of equipment. On the other hand, the training toolkit is more expensive, and one can expect to spend about US$1700, without a disassembler/debugger.

## Acknowledgments

This research project has been carried out in partnership between CPqD (independent institution focused on R&D and innovation) and ELEKTRO (the eighth largest power supply organization in Brazil) and relies on R&D financial resources provided and managed by ANEEL.

We would like to give special thanks to Rafael de Simone Cividanes, CPqD's consultant, for coordinating the project and reviewing this paper.

## References

[1] F. Skopik, Z. Ma, T. Bleier, and H. Gruneis, "A survey on threats and vulnerabilities in smart metering infrastructures", International Journal of Smart Grid and Clean Energy, vol. 1, no. 1, Sep. 2012, pp. 22-28.

[2] M. Carpenter, T. Goodspeed, B. Singletary, J. Searle, E. Skoudis, and J. Wright, "Advanced metering infrastructure attack methodology", Mar. 2011, InGuardians, Inc. 2.0.

[3] WebGoat. Available from: https://www.owasp.org/index.php/ Webgoat_WebGoat_Project. Accessed: Feb. 26th, 2014.

[4] J. Searle, G. Rasche, A. Wright, and S. Dinnage, "AMI Penetration Test Plan". Available from: http://www.smartgrid.epri.com/doc/AMI-Penetration-Test Plan-1-0-RC3.pdf. Accessed: Apr. 11th, 2014.

[5] J. Grand, "Practical Secure Hardware Design for Embedded Systems", Proc. 2004 Embedded Systems Conference (ESC 04), Mar. 2004, pp. 1-25.

[6] iGoat. Available from: https://www.owasp.org/index.php/ OWASP_iGoat_Project. Accessed: Apr. 11th, 2014.

[7] GoatDroid. Available from: https://github.com/jackMannino/ OWASP-GoatDroid-Project/wiki. Accessed: Apr. 11th, 2014.

[8] InGuardians, Inc., "Advanced metering infrastructure attack methodology", Mar. 2011.

[9] MSP430F5438 Experimenter board. Available from: http://www.ti.com/tool/msp-exp430f5438. Accessed: Feb. 26th, 2014.

[10] MSP430F5438 Features. Available from: http://in.embeddeddeveloper.com/processors/3252/Texas-Instruments/MSP430F5438.htm. Accessed: Feb. 26th, 2014.

[11] Analog Devices, "Poly phase multifunction energy metering IC with per phase information", ADE7758 datasheet, Oct. 2011.

[12] Texas Instruments, "LM555 Timer", Mar.2013.

[13] Arduino Uno. Available from: http://arduino.cc/en/Main/ arduinoBoardUno. Accessed: Feb. 26th, 2014.

[14] A. Menezes, P. C. van Oorschot, and S. A. Vanstone, "Handbook of applied cryptography". CRC Press. Aug. 2001.

[15] N. AlFardam, D. Bernstein, K. Paterson, and J. Schuldt, "On the security of RC4 in TLS", Proc. of 22nd USENIX Security Symposium. Aug. 2013, pp. 305-320.

[16] JTAGulator - open source hardware for OCD identification. Available from: http://www.grandideastudio.com/portfolio/ jtagulator/. Accessed: Feb. 26th, 2014.

[17] Bus Pirate - open source hacker multi-tool. Available from: http://dangerousprototypes.com/bus-pirate-manual/. Accessed: Feb. 26th, 2014.

[18] GoodFET - JTAG debugger for TI MSP430. Available from: http://goodfet.sourceforge.net/. Accessed: Feb. 26th, 2014.

[19] MSP-FET430UIF - Official JTAG debugger for TI MSP430. Available from: http://www.ti.com/tool/msp-fet430uif. Accessed: Feb. 26th, 2014.