Implementation Aspects of MeterGoat, a Smart Meter Security Training Platform

[Tool Demo]

Parth Bhatt
GTSIC, CPqD
Campinas, Sao Paulo, Brazil
parth@cpqd.com.br

Johny de Souza
GTSIC, CPqD
Campinas, Sao Paulo, Brazil
johny@cpqd.com.br

Nelson Uto
GTSIC, CPqD
Campinas, Sao Paulo, Brazil
uto@cpqd.com.br

Jefferson Capovilla
GTSIC, CPqD
Campinas, Sao Paulo, Brazil
jrodrigo@cpqd.com.br

Jose Resende
ELEKTRO
Campinas, Sao Paulo, Brazil
jose.resende
@elektro.com.br

Danilo Suiama
ELEKTRO
Campinas,Sao Paulo, Brazil
danilo.suiama
@elektro.com.br

ABSTRACT

With the increase in popularity of smart devices for metering purposes, the concerns and the scope of the threats also gets bigger. In order to raise awareness and provide a simulated environment for information security training to smart meter manufacturers, we developed MeterGoat, which is a flexible framework for smart meter penetration testing. In this research paper, we describe the implementation of MeterGoat and present some sample lessons that explore vulnerabilities based on weak anti-tampering techniques, flawed authentication, and improper use of cryptography.

Categories and Subject Descriptors

C.3 [**Special Purpose and Application based system**]: Real-time and embedded systems; K.6.5 [**Security and Protection**]: Unauthorized access

General Terms

Security, Experimentation

Keywords

smart meter vulnerabilities, penetration testing, tool demo, security training, cyber attacks

1. INTRODUCTION

Smart Grid technologies were successful, initially, by improving operational efficiency, flexible billing, cost reduction in energy distribution, and then in providing smart grid status control capabilities using remote commands, such as connecting and disconnecting the users remotely. Although introduction of all these advanced technologies is promising, the presence of large number of security vulnerabilities in smart meters and the underlying infrastructure [12, 2, 9] has created problems that never existed in the energy infrastructure. According to the US Federal Energy Regulatory Commission, the estimated penetration rates were as high as 30.2% in 2013 [11]. These vulnerabilities can certainly be misused for small profits on local levels while can even lead to epidemic and targeted cyber attacks. Therefore, we consider smart meter security training for the development teams of smart meter manufacturers as an important factor that can lead to significant reduction of inbuilt security vulnerabilities and generate awareness.

In an introductory paper from our team about MeterGoat [1], a low cost hardware platform for smart meter security training, we published about the research and implementation plans. In this research paper we aim to describe the actual implementation and some sample lessons that can be taught to the trainees with practical hands-on experience. MeterGoat was named after the Open Web Application Security Project (OWASP) WebGoat project [13], which has similar purpose, but for the scope of web applications. MeterGoat implements the most relevant smart meter functionalities in a vulnerable way, allowing the trainees to perform real attacks without affecting live environments. In this way, they can learn about the methods an adversary can use to exploit the real products they develop in their industry. In this paper we then selected vulnerabilities related to anti-tampering techniques, flawed authentication, and improper cryptography.

The remaining part of this paper is structured as follows: Section 2 presents implementation aspects related to MeterGoat. Section 3 provides some sample lessons for the vulnerabilities that are implemented and the ways they can be exploited. In Section 4, we list the future works for this research. Section 5 provides final remarks.

2. IMPLEMENTATION ASPECTS

We based the hardware used to implement the MeterGoat on Texas Instruments MSP430F5438A experimenter board

SIN '14, September 09 - 11 2014, Glasgow, Scotland Uk

http://dx.doi.org/10.1145/2659651.2659744

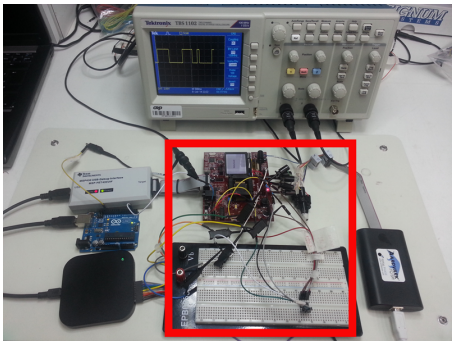

Figure 1: The MeterGoat Lab.

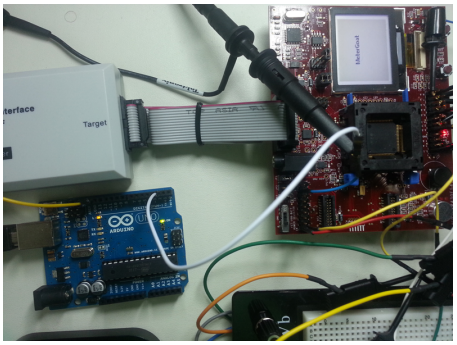

Figure 2: Arduino board used for breaking anti-tampering mechanisms.

[5], featuring the MSP430F5438 microcontroller. Several smart meters use this model because it is low power and has on-chip memories, embedded controllers for SPI, LCD, UART and their corresponding libraries. It also comes with on-chip hardware multipliers that can handle signed and unsigned multiplications of 8, 16, 24, and 32-bit numbers [5].

As seen on several smart meters, the JTAG, used for performing memory programming, boundary scanning and on-chip debugging, is enabled in MeterGoat. Other relevant features include the SPI interface, the LCD display and the external memory that is used to backup old information.

We use an external pin to emulate a switch for protecting against tampering. If someone opens the chassis, the pin will go low and the firmware can protect itself, using active and passive zeroization. The permanent data such as consumption history and user database are encrypted and stored in the internal flash memory.

3. SAMPLE LESSONS

This section gives samples lessons related to physical tampering, flawed authentication, and improper cryptography.

3.1 Physical Tampering

In order to ensure the integrity of the device, commercial meters usually have a method for detecting physical tampering as well as a strategy in their code to protect critical information in case the device is violated. One common anti-tampering mechanism is a switch that is kept high if the case is closed. This poses a vulnerability because it is easily recognizable and fooled by an attacker that wants to open the case to explore the internals of the device.

In this sample lesson, we identify an anti-tampering device in the MeterGoat implementation as an active low switch. To prevent the firmware from making any changes to itself or even detect the opening of the case, we hook a wire and feed the base of the switch with a logical '1' digital signal generated with an Arduino Uno board [10], as illustrated in Figure 2. With this setup, we can trick the firmware and ensure that it behaves in the same way it would if the security of the case was not violated.

3.2 Flawed Authentication

Generally, smart meters are equipped with an optical port for the field operators to directly access individual meters in order to retrieve the meter readings, and for maintenance purposes [12]. All the legitimate interactions with a smart meter are generally performed either via network access or the optical port. In either case, such interactions first require authentication credentials, which are verified against a database present somewhere in the memory of the smart meter. In order to facilitate and avoid unnecessary problems for the field operators, many manufacturers store the credential information in capital letters. Thus, any input for the authentication is first converted to upper case before verification.

In MeterGoat, we implemented similar access authentication for the field operators. The user credentials are stored in the MSP430 flash memory at a particular segment of INFO memory, whereas the passwords are in the form of unsalted MD5 hashes of their corresponding capital letter form. This type of protection mechanism is flawed and can be easily bypassed, because of the simplicity of performing memory dumps of embedded systems [12, 2] and of cracking hashes of passwords chosen from sets of low cardinality.

Next, we explain in detail how one can easily exploit the aforementioned flawed mechanism. For the scope of teaching security of smart meters and not making it much complicated, we just consider dumps from the internal flash of MSP430, taken with the software tool MSP430Flasher [4] provided by Texas Instruments. Now as we know that MSP430 is a very commonly found processor on industrial and consumer devices, such as smart meters, information about its internal flash memory can easily be found and studied. A basic overview about its organization, as shown in Figure 3, would be enough to extract valuable information from the memory dumps.

After dumping the memory, one can load it into a disassembler, such as IDA [3]. Since cryptographic hashes are employed to protect passwords at rest, one can check if a common algorithm, such as MD5 or SHA-1, is implemented. To test for the former, one can look for the initialization values of the MD buffers [8] in the disassembled code. This can be done using the search immediate functionality in IDA. Once the location is found, as shown in Figure 4, we know, based on the majority of MD5 implementations, that it must be inside an initialization function. Based on this, by looking for xrefs to it, one can find the primary MD5 function. Thus, the code calling the MD5 function must be writing or verifying the returned values to some location in memory. Similarly, the memory location is traced and user ID and password database is found. Now, we can easily crack those MD5 hashes using online rainbow tables, hence obtaining the original passwords. Figure 5 illustrates a (key, value)

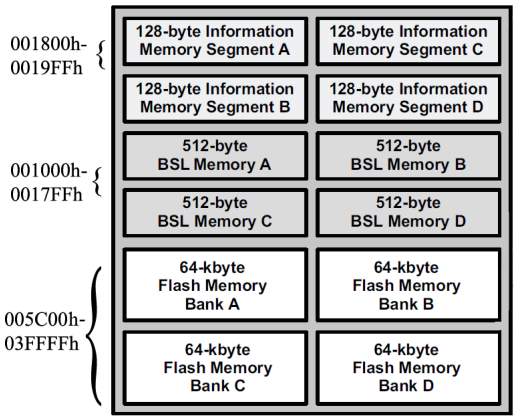

Figure 3: MSP430's memory organization [7, 6].

```
ROM:AAE2
ROM:AAE2                decd.w  SP
ROM:AAE4                mov.w   R12, 2+var_2(SP)
ROM:AAE8                mov.w   #2301h, 0(R12)
ROM:AAEE                mov.w   #6745h, 2(R12)
ROM:AAF4                mov.w   @SP, R15
ROM:AAF6                mov.w   #0AB89h, 4(R15)
ROM:AAFC                mov.w   #0EFCDh, 6(R15)
ROM:AB02                mov.w   @SP, R15
ROM:AB04                mov.w   #0DCFEh, 8(R15)
ROM:AB0A                mov.w   #98BAh, 0Ah(R15)
ROM:AB10                mov.w   @SP, R15
ROM:AB12                mov.w   #5476h, 0Ch(R15)
ROM:AB18                mov.w   #1032h, 0Eh(R15)
ROM:AB1E                mov.w   @SP, R15
ROM:AB20                clrx.a  10h(R15)
ROM:AB26                mov.w   @SP, R15
ROM:AB28                clrx.a  14h(R15)
ROM:AB2E                incd.w  SP
ROM:AB30                reta
```

MD5 buffers

Figure 4: Initialization values for the MD buffers.

```
ROM:1800                .byte  65h ; e
ROM:1801                .byte 0FFh
ROM:1802                .byte  59h ; Y
ROM:1803                .byte  75h ; u
ROM:1804                .byte 0C8h ;
ROM:1805                .byte  5Eh ; ^
ROM:1806                .byte  8Dh ;
ROM:1807                .byte  27h ; '
ROM:1808                .byte  56h ; V
ROM:1809                .byte  81h ;
ROM:180A                .byte    4
ROM:180B                .byte  91h ;
ROM:180C                .byte 0F9h ;
ROM:180D                .byte  9Eh ;
ROM:180E                .byte 0DFh ;
ROM:180F                .byte 0A1h ;
ROM:1810                .byte 0CFh ;
ROM:1811                .byte 0EDh ;
ROM:1812                .byte 0FFh
```

User id

MD5 Password

Figure 5: The user IDs and MD5 passwords stored as (key, value) pairs.

```
ROM:187E                .byte 0FFh
ROM:187F                .byte 0FFh
ROM:1880                .byte    1
ROM:1881                .byte    0
ROM:1882                .byte    0
ROM:1883                .byte    0
ROM:1884                .byte 0D8h ;
ROM:1885                .byte    0
ROM:1886                .byte    0
ROM:1887                .byte    0
ROM:1888                .byte    2
ROM:1889                .byte    0
ROM:188A                .byte    0
ROM:188B                .byte    0
ROM:188C                .byte  74h ; t
ROM:188D                .byte    3
ROM:188E                .byte    0
ROM:188F                .byte    0
ROM:1890                .byte    3
ROM:1891                .byte    0
ROM:1892                .byte    0
```

Month

Value

Figure 6: The month and reading values found in form of (key, value) pairs.

pair, from the credentials database, found using IDA, and, according to the loading address, we know that this is the beginning address of INFO memory in MSP430.

3.3 Improper Cryptography

The lessons of this section cover weak encryption techniques and key discovery using visual entropy analysis.

3.3.1 Weak Encryption Techniques

The use of heavy cryptographic algorithms is generally avoided on embedded systems, such as smart meters, due to lack of resources on them. Therefore, to still meet confidentiality needs, they are equipped with simpler encryption techniques that might be vulnerable.

In order to teach vulnerabilities posed by weak encryption techniques, we consider a scenario of AMI (Advanced Metering Infrastructure) in rural areas without proper network access. The smart meter stores monthly energy consumption readings in its memory and the field operators on their monthly trips fetch readings for last month into their devices using the optical port. The monthly meter readings are encrypted by XORing the readings with a secret key and stored with corresponding month in its local flash memory segment. When readings are to be fetch to field operator's device, they are retrieved by XORing them again with the key.

For this lesson, memory dump is loaded into IDA and, as we know from our previous lesson's experiment about flawed authentication, the persistent data values are stored in the INFO memory. One can then look into this region for the location that is used for writing the energy consumption values. Similarly, at the location 0x1880, it can be understood by simple observation that some sort of (key, value) data is stored, as illustrated by Figure 6. Supposing that these are meter reading values, on initial observation, the key seems to be incrementing and looks like the month for the reading. Now, upon comparing the stored monthly reading to that on the consumption bill of the corresponding month, we see they differ. Hence, it can be concluded that some sort of encryption is being used.

The simplest check one can do is to XOR a value from a given month with the meter reading value that came on that month's consumption bill. This gives us a candidate key, which can be XORed with the remaining readings, for the purpose of verification. If the obtained values are the same as those contained in the corresponding bills, one can infer the correct key was found. Thus, on the practical scenario, the trainees are able to find the key for the weak encryption technique, which allows them to overwrite the readings with lower values.

In the case when one is unable to figure out which encryption technique is used, the easiest thing one can do is to place the encrypted reading value of the last month in next month's value location. This would be interpreted as consumption of zero units for the present month.

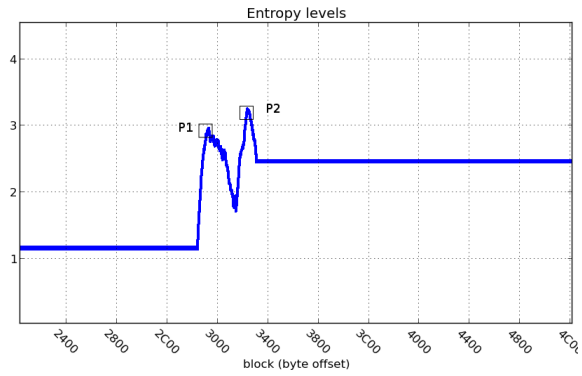

Figure 7: Entropy levels of RAM memory dump.

3.3.2 *Key Discovery using Visual Entropy Analysis*

There is no doubt about the presence of one or more cryptographic keys in a smart metering device. Because of inherent property of cryptographic keys, they are supposed to be more random than other data, thus this can be easily leveraged by hackers. A memory dump of RAM from MeterGoat was taken for further analysis. Block wise entropy is calculated and entropy levels are plotted against memory blocks on a graph shown in Figure 7. Presence of cryptographic keys can be easily guessed near the peaks in the graph. Therefore, in Figure 7, we can observe that P2 is the peak that might contain a cryptographic key.

The memory dump should then be searched for the block of data where P2 was found. There might be more than one peak in the entropy graph but hit and trial on the code combined with the brute force for the parts of the key can help hackers to discover the actual key. Although this method is not perfect and may sometimes generate a large sample space for brute force, the idea is just to teach the trainees about this technique.

4. FUTURE WORK

Currently, MeterGoat implements some basic vulnerabilities and can be used to teach their exploitation as described in the previous sections. We have been working to implement many other vulnerabilities on MeterGoat together with simple lessons that can be taught to the trainees such as sniffing the SPI of external memory, direct dump of external memory, buffer overflow on MSP430, format string attack, and integer overflow. In addition, we intend to make other lessons related to fuzzing on the external port and discovery of information related to cryptography based on power consumption of the system.

5. CONCLUSION

In this paper we discussed the implementation of MeterGoat, a low cost and open source platform, and we provided practical sample lessons based on it. The central idea is to teach developers of smart meter manufacturers using simple lessons with hands-on experience about security vulnerabilities of smart meters. This will bring awareness about the subject, contributing to more secure energy infrastructures.

6. ACKNOWLEDGEMENTS

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 (Brazilian Electricity Regulatory Agency).

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

7. REFERENCES

[1] J. Capovilla, N. Uto, D. Suiama, and J. Resende. Metergoat: A low cost hardware platform for teaching smart meter security. In *The 9th International Multi-Conference on Computing in the Global Information Technology*, 2014.

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

[3] The IDA disassembler and debugger. `https://www.hex-rays.com/products/ida/`. Accessed: July 6th, 2014.

[4] Msp430 flasher - command line programmer. `http://processors.wiki.ti.com/index.php/MSP430_Flasher_-_Command_Line_Programmer`. Accessed: July 6th, 2014.

[5] Msp430f5438 experimenter board. `http://www.ti.com/tool/msp-exp430f5438`. Accessed: Feb. 26th, 2014.

[6] Msp430f543xa, msp430f541xa mixed signal microcontroller (rev. d). `http://www.ti.com/lit/ds/symlink/msp430f5438a.pdf`. Accessed: July 6th, 2014.

[7] Msp430x5xx and msp430x6xx family userguide. `http://www.ti.com/lit/ug/slau208n/slau208n.pdf`. Accessed: July 6th, 2014.

[8] R. Rivest. The MD5 message-digest algorithm. `http://tools.ietf.org/html/rfc1321`. Accessed: July 6th, 2014.

[9] 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*, 1(1):22–28, September 2012.

[10] Arduino uno. `http://arduino.cc/en/Main/ArduinoBoardUno`. Accessed: July 6th, 2014.

[11] Assessment of demand response and advanced metering. Technical report, US Federal Energy Regulatory Commission, 2013.

[12] D. C. Weber. Optiguard: A smart meter assessment toolkit. Technical report, InGuardians, Inc., July 2012.

[13] Webgoat. `https://www.owasp.org/index.php/Webgoat\_WebGoat\_Project`. Accessed: July 6th, 2014.