ADVANCED METERING INFRASTRUCTURE ATTACK METHODOLOGY

INGUARDIANS, INC.

Version 2.0
Mar. 1, 2011

Change History

Author	*Date*	*Version*	*Description*
Matthew Carpenter	11/7/2008	0.1	Outline
Travis Goodspeed	11/12/2008	0.2	Initial Draft
Matthew Carpenter, Travis Goodspeed, Joshua Wright	12/20/2008	0.3	Updated Draft
Travis Goodspeed	12/23/2008	0.8	Reorganization Draft
Joshua Wright, Matthew Carpenter, Brad Singletary, Ed Skoudis	12/31/2008	0.9	Final Draft
Matthew Carpenter, Joshua Wright	1/5/2009	1.0	First Release
Justin Searle	3/1/2011	2.0	Second Release

Contributing Authors: Matthew Carpenter, Travis Goodspeed, Bradley Singletary, Justin Searle, Ed Skoudis, Joshua Wright

Table of Contents

1 Introduction

1.1 Purpose and Scope

This document describes InGuardians' approach to the security testing of different AMI architectures. This document specifically focuses on field-deployed devices employing embedded computer architectures, not physically protected by utility-premises security measures. This equipment includes the following resources:

- The AMI meter used for the collection of customer electricity consumption data;
- Aggregators, access points, and related up-stream network gear, up to but not including the head-end AMI system controller;
- Supporting data networks between the AMI meter and head-end, including:
    - Broadband Power Line systems;
    - Power Line Carrier systems;
    - Public Switched Telephone Network (PSTN) systems;
    - Radio Frequency communication systems including IEEE 802.15.4, ZigBee, 6LoWPAN, IEEE 802.11, IEEE 802.16, and proprietary systems.

An illustrative representation of this architecture is shown in Figure 1. While Home-Area-Network (HAN) devices are not specifically within scope of the initial attack team security testing, the approach and attack-methodology described in this document may be applied to HAN devices as well.

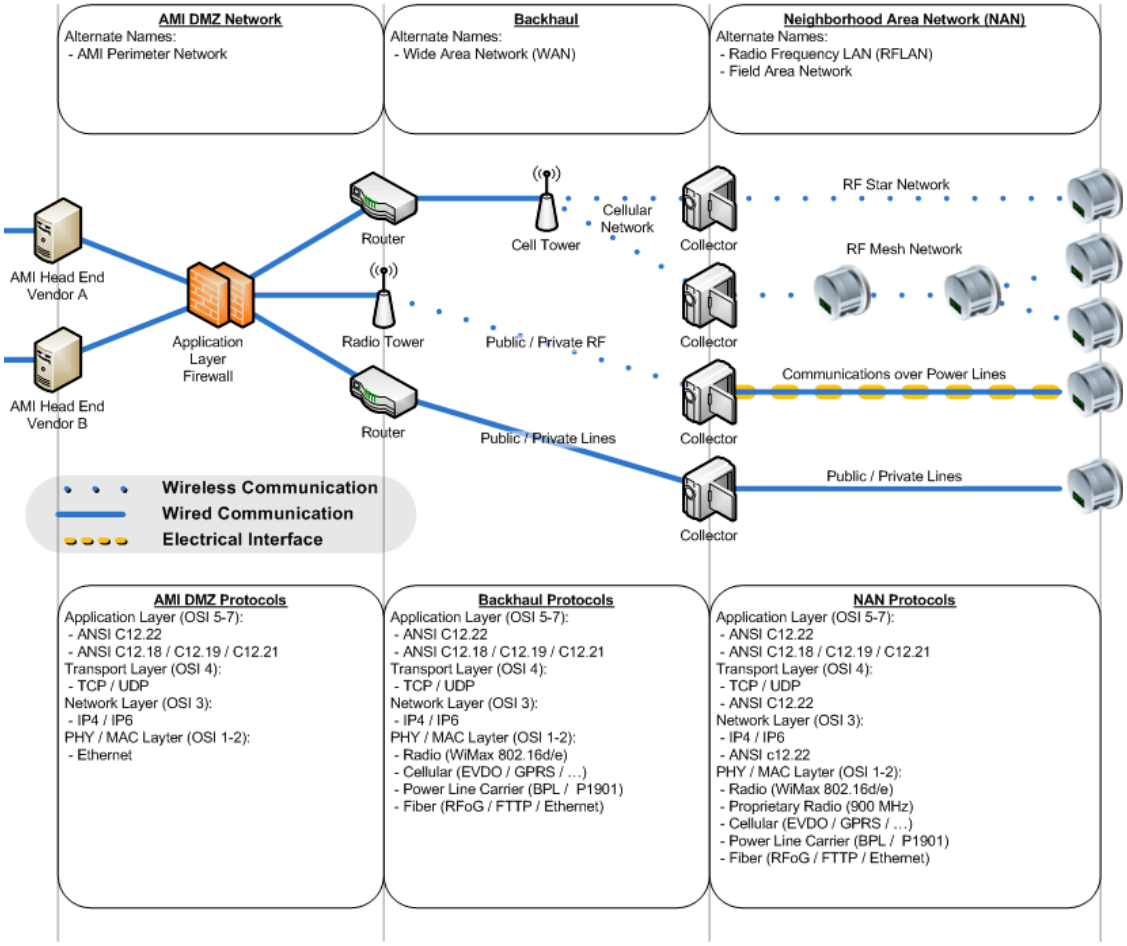

Figure 1. AMI Architecture Overview

The purpose of this document is to provide guidelines for utilities and vendors to test their own equipment or to outsource such testing with a better understanding of what to expect from their attack team. It includes a description of the recommended lab equipment, a description of the testing principles, a break down of vulnerability classes that can be tested for, and a detailed attack methodology. While this document's authors have attempted to communicate a significant level of depth clearly, it is important to note that these are simply guidelines for testing. A deep technical understanding of the involved equipment, protocols, electronics, and vulnerability research must be coupled with creativity and time for valuable testing results to be achieved.

This document is comprised of four major sections: Constructing a Lab, Principles of Testing, Common Vulnerability Types, and Attack Methodologies. Wherever appropriate, we have attempted to provide a step-by-step walk-through with a hands-on feel to our descriptions.

The scope of this document does not include utility-premises components such as head-ends, meter data management system (MDMS), customer relationship management

(CRM) systems, or other related systems that utilize data from the AMI system. It should be noted that the principles of testing these systems are similar to the techniques described in this document. However, as they have different access constraints and tend to be implemented on larger-scale computers with complex multiprocessing operating-systems, the toolset, methodology, vulnerabilities and impact are quite different.

1.2 Executive Summary

Vulnerabilities exist in any complex system. Identifying weaknesses and evaluating their associated risk allow for these vulnerabilities to be addressed, protecting customers, utilities and vendors alike. The stakes are very high, impacting the ability to generate, distribute, and consume energy in a given region.

This document is best read by vendor/utility security teams. It has been prepared to provide vendors the ability to understand the vulnerabilities and attacks in order to protect their equipment properly; to provide utilities the knowledge required to enlist appropriate skill sets for testing their own implementations; and to ensure test consistency among attack teams between different vendor architectures. Throughout the document, the authors aim to bring value both to the attack-team and the vendor/utility employing them.

InGuardians recommends vendors and utilities create full-time security teams if none currently exist, and additionally employ internal and third-party attack teams to search for and illustrate vulnerabilities in the utility/vendor AMI architectures and the impact of exploitation. Security must be designed into and tested at every stage in the AMI rollout process, including AMI system design, manufacturing, delivery, storage, and implementation. Weaknesses in any stage of this process can lead to catastrophic failure of the power grid. Regular penetration-tests executed by qualified attack teams are a necessary proof of such security measures and can provide valuable insight to improve existing architectures.

InGuardians recommends utilities and vendors perform regular penetration testing of the following systems (see Conclusions for a more descriptive list):

- Physical Security Perimeters
- Utility Premises Backend Systems
- Field Deployed Devices and Systems

Properly securing the backend AMI and SCADA systems that reside on the utility premises is even more important than field deployed equipment, because they can often affect more damage if compromised. Many security devices and processes exist to attack, analyze and secure general-purpose operating systems, upon which the utility premises backend systems are most often built. Nearly every hacker conference discloses new and old attack methodologies for attacking them, while training organizations such as the SANS Institute offer classes on defending them. However this document focuses on the last category of Embedded Devices deployed to in the field such as meters, aggregators,

collectors, access points, relays, and routers because less is publicly known about attacking these types of systems.

1.3 Glossary and Acronyms

AMI	Advanced Metering Infrastructure, a collection of systems used to evaluate data associated with the use of utility resources
ARM	Advanced RISC Machine, a processor used in embedded computer systems
Attack Team	Authorized team attacking AMI technology in a lab and other simulated environments
BPL	Broadband over Power Lines, a MAC layer networking transport system utilizing Power Line Communications as a physical layer medium
Chi-square Test	A data evaluation function used to measure the randomness of data, particularly in random number generators
Compiler	Software used to convert computer source code into machine language
CRM	Customer Relationship Management, a process used to track and store information about customers and prospects
Disassembler	Software used to convert machine language code into assembly-level instructions
Utility Premises Backend Systems	Computing systems supporting the analysis of data from customer data collection systems located in a head-end or supporting facility, typically consisting of IA32/IA64 architecture platforms
Field Aggregation Systems	Computing systems supporting the analysis of data from customer data collection systems located in a dispersed manner to support regional neighborhood area networks, typically consisting of a wide variety of architecture platforms. These are often called data aggregators, collection points, or access points.
Field Deployed	Data collection devices installed by field personnel at a customer premises for data collection, reporting to field aggregation systems

Equipment	and ultimately to backend systems like headends
FFT	Fast Fournier Transform, an algorithm to represent the frequency domain of a signal
Firmware	Software used to control embedded electronic devices
HAN	Home Area Network, representing customer-owned devices interfacing with a utility meter or associated device
I^2C	Inter-Integrated Circuit, a multi-master serial bus protocol
JDE	JD Edwards Enterprise One, a suite of Oracle-based applications for enterprise resource planning
JTAG	Joint Test Action Group, a standard for interfacing circuit boards, microprocessors and other peripherals for debugging purposes
MDMS	Meter Data Management System, a repository for the storage and processing of AMI-related data
NAN	Neighborhood Area Network, a collection of utility devices communicating within the same approximate geographic location
PLC	Power Line Communications, a physical layer transport system supporting protocols such as Broadband over Power Lines
RF Channel	A range of radio frequencies allocated for intended use with a specific application
RF Spectrum	A range of radio frequencies allocated without a specific use
RNG	Random Number Generator
SAP	System Analysis and Program Development (translation), a suite of applications for enterprise resource planning
SCADA	Supervisory Control And Data Acquisition, an industrial control system
SMD	Surface Mount Device, a method of mounting components to a circuit board
SPI	Serial Peripheral Interface, a master/slave serial bus protocol

USRP	Universal Software Radio Peripheral, a software-defined radio hardware platform

1.4 Works Cited

[1] Aircrack-ng. December 31, 2008. http://www.aircrack-ng.org/doku.php

[2] AMI-SEC Task Force. "AMI System Security Requirements and Guidelines." December 17, 2008.

[3] ANSI. "C12.18 Protocol Specification for ANSI Type 2 Optical Port." NEMA, 1996.

[4] ANSI. "C12.22 Protocol Specification for Interfacing to Data Communication Networks, Draft." NEMA, December 28, 2007.

[5] Beck, Martin and Tews, Erik. "Practical Attacks Against WEP and WPA", December 31, 2008. http://dl.aircrack-ng.org/breakingwepandwpa.pdf

[6] Fluhrer, Scott; Mantin, Itsik and Shamir, Adi. "Weaknesses in the Key Scheduling Algorithm of RC4", December 31, 2008. http://www.drizzle.com/~aboba/IEEE/rc4_ksaproc.pdf

[7] EMBER EM250. "Single-Chip ZigBee/802.15.4 Solution", pp. 106. January 5, 2009. http://www.ember.com/pdf/EM250_Datasheet.pdf

[8] FIPS-PUB 74. "Guidelines for Implementing And Using The NBS Data Encryption Standard." January 5, 2009. http://www.itl.nist.gov/fipspubs/fip74.htm

[9] Goodspeed, Travis. "A Side-Channel Timing Attack of the MSP430 BSL." Blackhat USA. 2008.

[10] Goodspeed, Travis. "Practical attacks against the MSP430 BSL." 25th Chaos Communication Congress. 2008.

[11] Kaplan, Dave. "Report: TJX Breach Began in Minnesota Marshalls Parking Lot", SC Magazine, May 4 2007. December 31, 2008, http://www.scmagazineus.com/Report-TJX-breach-began-in-Minnesota-Marshalls-parking-lot/article/34954/

[12] Kelly, Scott. "Security Implications of Using the Data Encryption Standard (DES)", IETF RFC 4772, December 2006. http://www.ietf.org/rfc/rfc4772.txt

[13] Kershaw, Mike. "How to Neuter Cryptography for Thousands of Users in Two Lines", December 31, 2008. http://kismetwireless.net/articles/2600-08-summer-ssl.txt

[14] SecurityFocus. "FreeBSD IPsec Replay Vulnerability", BUGTRAQ ID 17191, December 31, 2008. http://www.securityfocus.com/bid/17191/info

[15] Security Focus. "Windows NT Syskey Reused Keystream Vulnerability", BUGTRAQ ID 873, December 31, 2008. http://www.securityfocus.com/bid/873

[16] Walker, John. "Ent", December 30, 2008. http://www.fourmilab.ch/random/

[17] Warner, JS; Johnston, RG, "A Simple Demonstration that the Global Positioning System (GPS) is Vulnerable to Spoofing", The Journal of Security Administration 25, 19 (2002).

[18] Wright, Joshua. "Pcaphistogram", December 30, 2008. http://802.11ninja.net/~jwright/code/pcaphistogram.pl

[19] Xilman, Paul. "2_1574L.c156", December 31, 2008. http://www.mersenneforum.org/showpost.php?p=124079&postcount=97

1.5 References

Catsoulis, John. "Designing Embedded Hardware, Second Edition". New York, O'Reilly Press, 2006.

Erickson, Jon. "Hacking, the Art of Exploitation." San Francisco, No Starch Press, 2008.

Francillon, Aurélien, and Claude Castelluccia. "Code Injection Attacks on Harvard-Architecture Devices." 15th ACM Conference on Computer and Communications Security, 2008. December 29, 2009. http://www.ist-ubisecsens.org/publications/code_inject_attack_ccs08.pdf.

Fortify Software. "Fortify Taxonomy: Software Security Errors", 2008. December 20, 2009. http://www.fortify.com/vulncat/en/vulncat.

Goodspeed, Travis. "Tracing with MSP430simu, LaTeX, and PowerPoint". January 2, 2008. http://travisgoodspeed.blogspot.com/2008/01/tracing-with-msp430simu-latex-and.html

Koizol, Jack, et al. "The Shellcoder's Handbook." Indianapolis, Wiley, 2004.

1.6 Contact Information

Please direct questions, comments and suggestions on this document to:

InGuardians, Inc.
http://www.inguardians.com
+1 202-448-8958
security@inguardians.com

2 Principles of Operational AMI Vulnerability Assessment

To maximize the effectiveness and applicability of the security evaluation of AMI-related technology, the attack team should follow a series of testing principles supporting consistent technology analysis. Through these techniques, a team of analysts can produce a cohesive assessment of one or more target devices or AMI deployments. The resulting documentation will provide necessary information to remediate the threats and vulnerabilities that would otherwise hinder the successful deployment of AMI-based technology.

2.1 Impact to AMI Value Streams

The AMI System Security Architectural Description describes five categories of value streams for the adoption of AMI technology [1]:

- Billing
- Customer
- Distribution System
- Installation
- System

Each value stream represents tangible benefits for AMI adopters. When exposed to system vulnerabilities and risks, these values are threatened, effectively reducing the overall benefit of adopting AMI technology.

As a key component of all security evaluation performed by the attack team, each attack or vulnerability should be evaluated in the scope of the affected value streams that threaten the overall AMI technology benefit for adopters. Through this analysis, each attack or vulnerability can be described such that utilities and vendors are able to evaluate the affected system impact with respect to the key utility value streams.

2.2 Practical and Pertinent Vulnerability Analysis

As a critical component of each security evaluation exercise, the attack team should work with vendors, the AMI security team, utilities and applicable third-parties to identify an appropriate scope for the analysis. The scope must take into account the resources of an adversary that may attempt to compromise the security and integrity of AMI-related technology. The resources accessible to a potential adversary will significantly influence the threats that require defense and mitigation strategies. For example, an attacker who has up to $10,000/USD in accessible resources for the purpose of exploiting AMI technology represents a different threat than a well-funded adversary whose goals exceed that of common mischief or simple service theft. The scope for the analysis should address the highest-level of attacker funding feasible; while the team must keep in mind all of the different levels of attack. For instance, the scope may include the resources available to a foreign government, but the attack team should be cautious not to ignore the attack vectors likely to be targeted by self-funded individuals.

Based on the scope, the attack team should apply scientific analysis methods to enumerate and evaluate potential threats, collecting supporting data through observation and experimentation then designing further analysis test cases to evaluate through experimentation. Threats that are deemed irrelevant or impractical based on the identified adversary resources can be disregarded, focusing on the practical and pertinent threats immediately affecting customers, utilities and vendors. Noting these discarded threats, their likelihood and impact of exploitation should be considered additional value to a final report. Vendors and utilities must understand the contextual constraints the analysis project scope places on any engagement and subsequent report.

Through this method, the attack team will focus on the areas of direct value for affected parties, providing the greatest possible benefit in the analysis and findings reporting.

2.3 Testing Team Expertise

In order to be effective in the analysis of AMI technology, the attack team will require expertise in multiple areas of information security, electrical and radio engineering. Due to the varied expertise requirements, it is anticipated that the attack team will be

composed of several individuals with diverse backgrounds, collaborating on the analysis of AMI technology.

Fields of expertise required for effective analysis include:

- Knowledge of the assembly language for processors used by meters and associated hardware
- Software vulnerability identification and analysis skills
- Experience writing software-based exploits
- RF modulation and coding analysis experience
- Experience analyzing standards-based protocols including infrared, IEEE 802.15.4/ZigBee, IEEE 802.16/WiMAX, BPL/PLC, ANSI C12.18/C12.19/C12.21/C12.22 and proprietary protocols
- Basic and advanced understanding of electronics and principles of electricity
- Safety training and experience working with high-voltage electronics
- Experience developing embedded technology including embedded software development programming and hardware engineering
- Analysis skills for evaluating cryptographic algorithms and associated functions
- Ability to creatively evaluate technology for the goal of subversion

2.4 Reproducible Findings

The attack team recognizes that any findings identifying security threats or vulnerabilities in AMI-related technology will be subject to significant scrutiny and analysis. Throughout the project, all security-related findings will be documented such that other analysts will have sufficient information to reproduce the findings for an independent findings evaluation, where desired. Methodologies and results should both be carefully documented to make results repeatable by other researchers, the associated vendor(s), and utility companies with access to similar resources to the testing team.

2.5 Risk Evaluation

The identification of risks to customers, utilities and vendors of AMI-related technology should first be determined. Where vendors and utilities have a finite amount of resources to apply to the mitigation of risks, it is necessary to provide an overall prioritization of identified threats such that threats of the greatest urgency are resolved before threats with little to no overall risk.

To support the prioritization and evaluation of risk, the attack team should evaluate all threats and vulnerabilities with supporting documentation as follows:

- Risk Description – Information describing the risk, citing references where applicable.
- Risk Probability – The probability of the risk happening to a vendor based on the understanding of potential adversaries that require defensive measures. This probability should be recorded on a scale of 1 to 10 with 1 being least probable and 10 being the most probable.
- Risk Impact – The potential impact of the risk to the utility or vendor should the threat be realized. The impact should be recorded on a scale of 1 to 10 with 1 being least impact and 10 being the greatest impact.
- Risk Data Quality Evaluation – Risk evaluation requires unbiased and accurate data for credibility. The risk documentation should include the following data points to describe the quality of the risk data:
    - Risk Understanding – How well is the risk understood?
    - Data Availability – How complete is the data pertaining to the risk?
    - Data Quality – Is the data available relevant to the risk being described? Is the data current?
    - Data Reliability – How objective is the data that has been supplied to describe the threat? The reliability of data will increase with multiple data points from different sources coming to similar conclusions, or decrease if the data is highly biased or widely disparate across multiple sources.
- Risk Urgency Assessment – What is the urgency of the risk?

The completed risk data will be used to populate a quantitative risk evaluation model, providing the necessary information to the utility or vendor for prioritizing threat remediation, transfer or acceptance. An example of the quantitative risk evaluation model for evaluating the priority of risk remediation is shown in Figure 2.

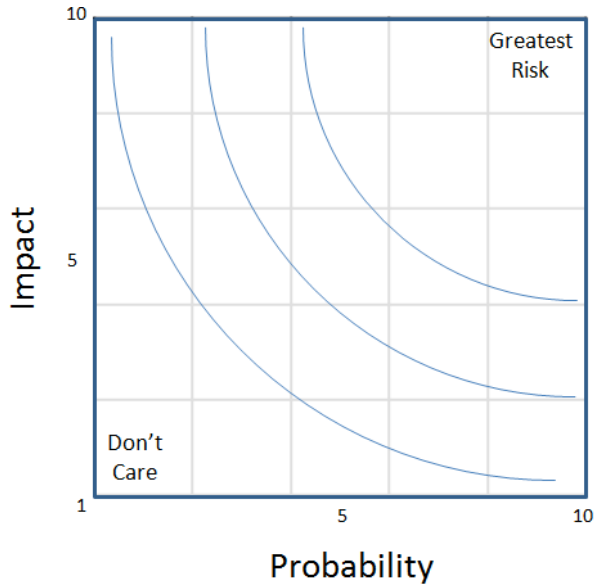

Figure 2. Quantitative Risk Measurement Model

2.6 Opportunities for Defensive Measures

While the focus of the attack team is to identify vulnerabilities threatening AMI technology, such in-depth analysis will also reveal defensive measures for mitigating the impact or probability of threats. While not a definitive measure of the only strategy for defending against identified vulnerabilities, the attack team should identify strategies for the mitigation of vulnerabilities that may be adopted by the utility or the vendor.

3 Lab Construction

For the analysis of AMI devices, InGuardians recommends utilizing several classes of devices and technology. These resources provide insight into the workings of various components included in AMI system implementations with a focus on security analysis.

3.1 Connecting to Component Buses

The ability to capture and inject data onto a serial or serial-like data bus can be exceptionally useful for an attacker. Using commercial or open-source tools, an attacker not only can monitor vital information between components on the bus, such as configuration information or cryptographic keys, but also can interact quickly with components such as radios and EEPROMs without desoldering them from a circuit-board.

For the I^2C and SPI serial protocols, Total Phase provides excellent tools such as the Beagle I^2C/SPI Protocol Analyzer and the Aardvark I^2C/SPI Host Adapter devices. Total Phase provides these tools for between $250 and $350. The PC software for interacting with them is provided on Windows, Linux and Mac OS X. This software includes user applications which allow a technician to program memory chips easily and sniff bits from the bus. The low-level attacker may use the provided code libraries to write custom applications in various programming languages including C, C++, Python and Visual Basic. For example, a custom program might capture specific data on the bus to piece together a key which is stored in different areas of an EEPROM. This key information may then be sent to a wireless packet-sniffer which can decrypt traffic protected by the observed key.

The Bus Pirate, shown in Figure 3, is an open-source protocol adapter. While slower than the Total Phase Aardvark, its open-source hardware and firmware allow it to be modified for use with other protocols, including proprietary and vendor-specific protocols.

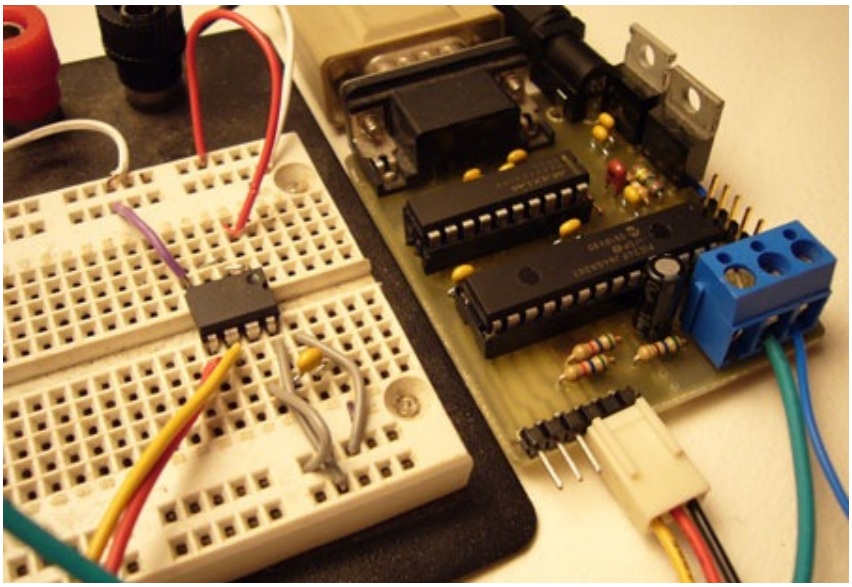

Figure 3. Bus Pirate

3.2 Development Kits

In addition to analyzing existing hardware, it will often be necessary for the attack team to quickly fabricate new hardware from microcontrollers, reconfigurable logic boards, radios, and similar devices. With development kits for these components, tools can be rapidly assembled without the overhead traditionally associated with ground-up design and implementation.

For example, Michael Poppitz of Germany has published a logic analyzer core for Digilent's Spartan 3 FPGA development board. This core can be loaded into the memory of the development board, creating a logic analyzer without the delay that would be expected of stand alone logic analyzers and would negate the need to purchase one.

Microcontroller kits, particularly those with radios such as the Crossbow TelosB and the Atmel Raven, allow for the rapid prototyping of less timing-critical applications such as radio packet sniffers and protocol fuzzers, which cannot be purchased commercially. Development kits can also be acquired for the standard ZigBee HAN stack and for non-802.15.4 radios that will compose Neighborhood Area Networks (NANs). Such a kit, the Texas Instruments EZ430T2500, is shown in Figure 4.

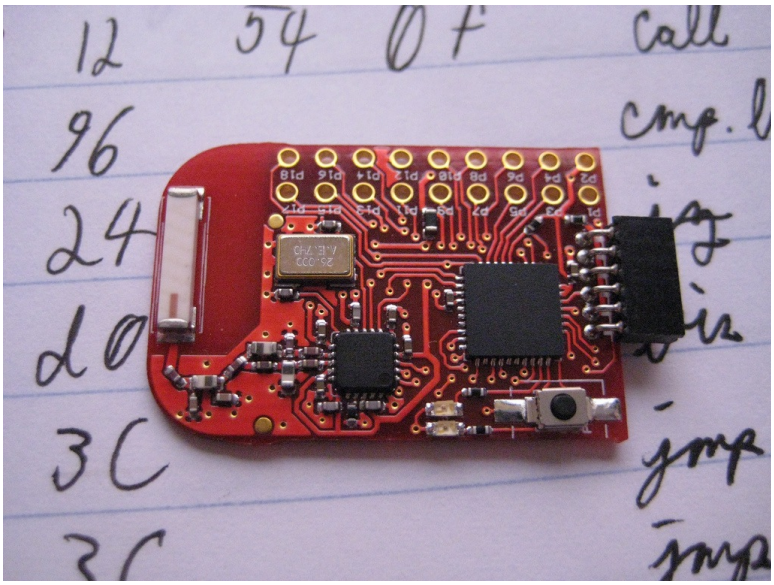

Figure 4. EZ430T2500

3.3 Logic Analyzers and Oscilloscopes

In addition to high-level protocol analyzers, such as those described in Section 3.1, the team must be equipped with logic analyzers and oscilloscopes. A digital logic analyzer, by recording digital values, allows an engineer to determine the type of traffic that might not otherwise be readable. By reading logic recordings, it is also possible to recognize timing attacks and to decode bus protocols which are partially unknown, but not encrypted. For best results in low-power embedded systems, a 32-channel analyzer with a sample rate of late least 100 MHz should be used such as the Intronix LogicPort logic analyzer.

An oscilloscope allows for a similar view of the analog world. Lacking the many channels of a digital logic analyzer, a scope is more appropriate for determining clock rates, decoding simple serial protocols of unknown voltage, and quickly locating signals for use with the digital analyzer. For use in AMI security analysis, we recommend a minimum sample rate of 100 MHz such as the Agilent Technologies U2701A modular oscilloscope. A screenshot of such a scope can be seen in Figure 5.

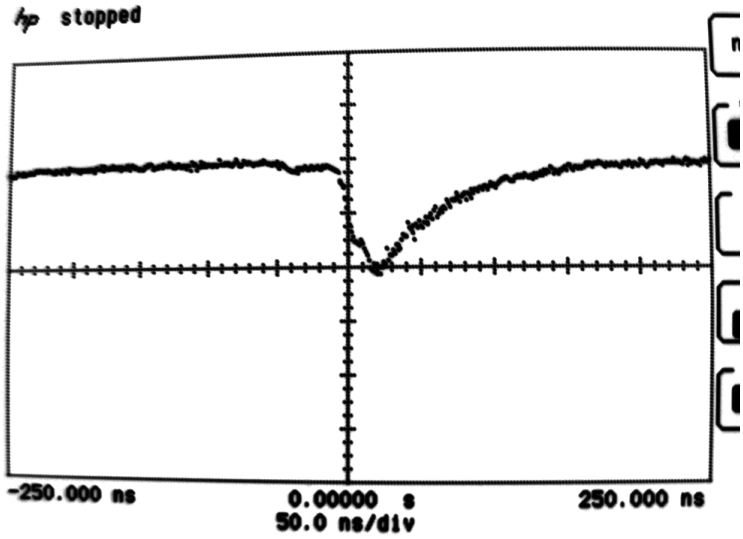

Figure 5. An oscilloscope's view of a voltage-glitching attack.

3.4 JTAG ICD

Although I^2C and SPI protocol adapters will suffice for programming most serial memory chips, JTAG In Circuit Debugger (ICD) tools are indispensable for reprogramming and debugging microcontrollers. Many ARM-derived microcontrollers can be debugged with the standard Olimex JTAG USB OCD, but device-specific debuggers are required for working with the TI MSP430 and Atmel AVR chips.

3.5 Firmware Analysis Software

Given sufficient time and experience, an attacker with physical possession of an AMI device will successfully retrieve the device firmware. As such, the firmware must be evaluated for security threats and vulnerabilities through static analysis. If source code is not available, this process requires the attack team to disassemble the binary processor instructions from the firmware. Software to perform this disassembly ranges from primitive to advanced. Alternatively, the attack team could write its own disassembly software customized to meet its particular test methodology and needs.

Most compilers ship with a primitive disassembler such as Objdump from the freely available GNU Compiler Collection (GCC). More advanced firmware analysis requires an advanced disassembler, one which is capable of constructing function-call and control-flow diagrams, drawing memory maps, and performing similar advanced functions. IDA Pro, a popular tool among software reverse engineers, supports many but not all embedded architectures, such as the Freescale MC1322X series of microprocessors. More obscure architectures are supported by various open-source utilities, such as MSP430Static for the TI MSP430.

3.6 Probes

In order to observe the electronic behavior of a meter, it is necessary to be able to probe the device electronically. Although oscilloscopes and logic analyzers often include simple probes, other types of probes such as sharp needles and grabber clips make it easy to monitor a signal without losing electric contact with a board. An example of a medical syringe converted for use as an electrical probe with LED polarity indicators is shown in Figure 6.

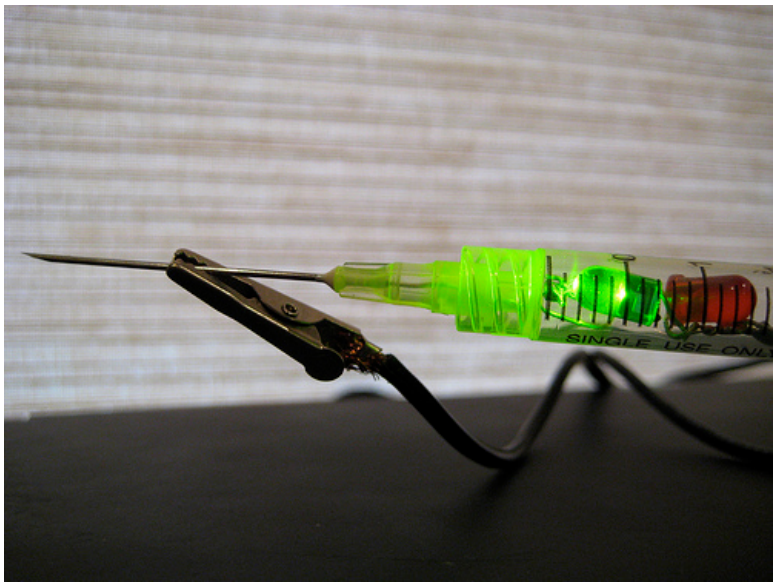

Figure 6. Medical syringe repurposed as an electrical probe

3.7 Spectrum Visualization and Analysis

In the analysis of HAN or NAN environments, spectrum visualization tools may be leveraged to evaluate the use of the RF spectrum to identify channels in use, channel bandwidth, utilization approximation, and interference sources. For HAN and NAN devices operating in the 2.4 GHz band, inexpensive visualization tools such as Wi-Spy are useful for identifying spectrum utilization behavior, as shown in Figure 7. Other frequencies such as the 868/915 MHz bands can be assessed with software defined radio technology such as the Universal Software Radio Peripheral (USRP), shown in Figure 8.

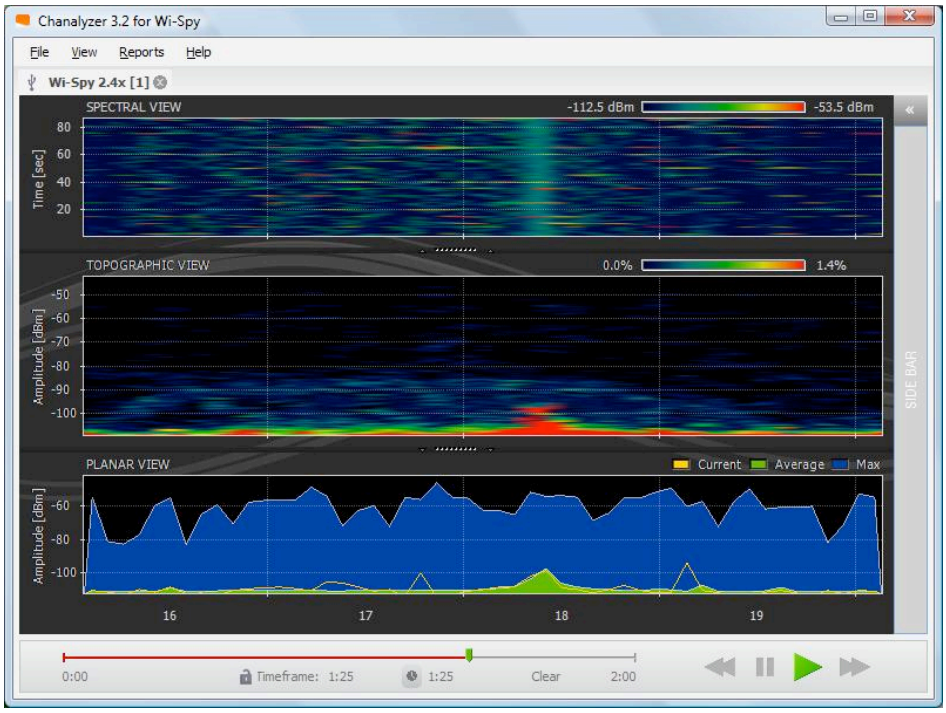

Figure 7. Wi-Spy representation of RF activity at 2.4 GHz

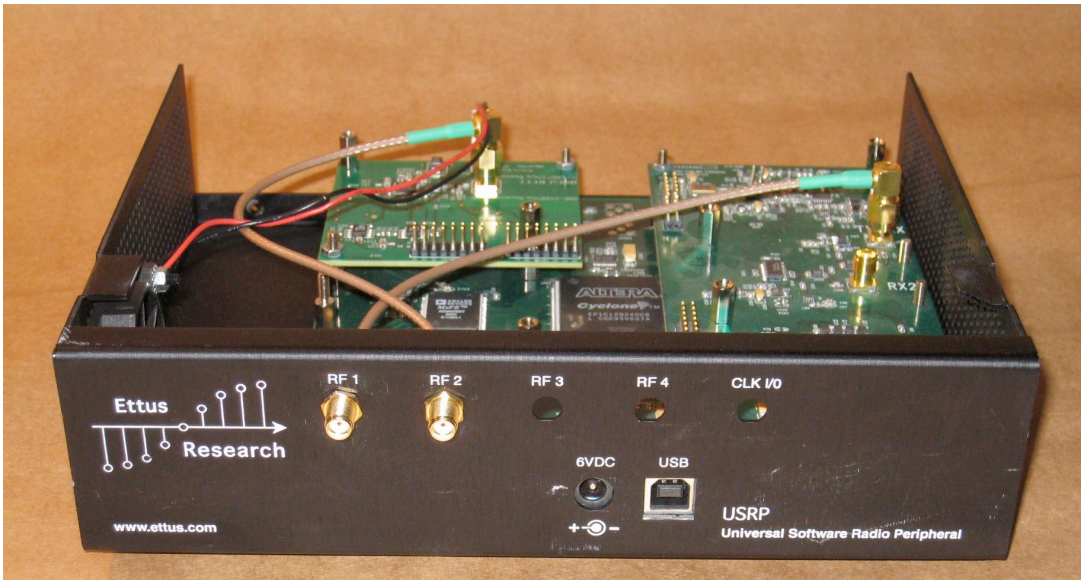

Figure 8. Universal Software Radio Peripheral

As an implementation of a software-based radio, the USRP also performs as a digital spectrum analyzer, measuring the Fast Fourier Transform (FFT) of incoming signals, as shown in Figure 9. FFT analysis tools are also accessible on some digital oscilloscopes.

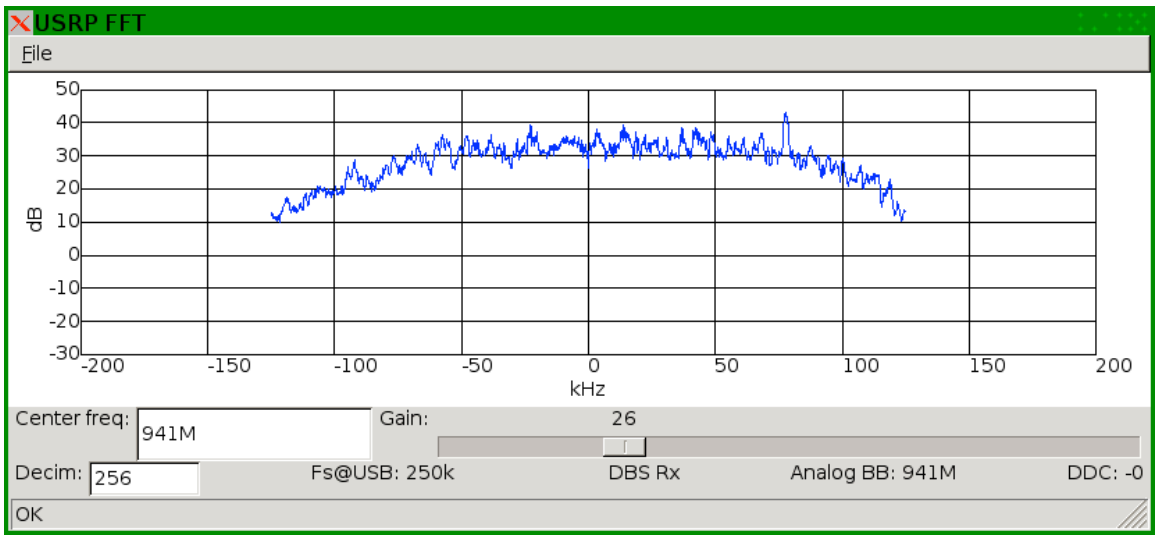

Figure 9. USRP FFT Display

3.8 Soldering and Rework Stations

In addition to probing a chip or bus in-place, it is often convenient or necessary to remove the chip, replacing it later without damaging either the chip or the board. While in the past it was common to find chips mounted in sockets allowing for easy removal or replacement, this is no longer the case. An SMD rework station, consisting of both a traditional soldering iron and a hot-air wand, is the perfect tool for removing or replacing soldered-on chips. A technician can heat a board to the point at which solder reflows, then lift a chip off of that board, using care to avoid damaging other components under the heat of the soldering iron. Resoldering is performed similarly, with perhaps the use of solder paste or an iron to correct the absence or surplus of solder as needed.

3.9 Common Hardware Tools

In the process of opening meters and extracting circuitry for analysis, the attack team will require common hardware tools, including but not limited to:

- Common screwdrivers (torx, Phillips, straight-edge)
- Various security-bit screwdrivers
- Hex wrenches, common wrenches, socket sets
- Wire cutters, dental picks, tweezers
- Magnifying glasses, loupes, stereo zoom microscopes
- SMD chip removal kits

- Security lock bits
- Lock-pick sets

3.10 Cryptographic Accelerators

Cryptographic acceleration hardware will aid the attack team in the analysis of cryptographic properties including selected algorithms, cryptographic operating modes and key length. While costly cryptographic accelerators such as the Cavium OCTEON represent an opportunity to rapidly evaluate the properties of a selected cryptographic suite, other common hardware such as the Graphics Processing Unit (GPU) in high-end video cards used in desktop and laptop systems may also be used. Devices such as the NVIDIA GeForce GTX 280 video card shown in Figure 10 provide a significant cryptographic processing advantage over traditional host processors.

Figure 10. NVIDIA GeForce GTX Video Card

3.11 Payload Statistical Distribution Tools

In the analysis of data collected through bus analysis, packet sniffing or other means, ciphertext payload statistical distribution software algorithms may be applied to evaluate the quality of cryptographic algorithms and data. While some payload histogram tools are available as open-source components such as Joshua Wright's Pcaphistogram tool [18] and John Walker's Ent [14], other software must be constructed to suit the specific algorithm and data collection needs for vendor-specific implementations.

3.12 Networking Supplies

For the purposes of establishing and emulating network connectivity to a meter, or for the purpose of interacting with multiple systems in the process of analyzing AMI technology, the attack team will require the use of common networking hardware and associated cabling, as well as specialty cables and interfaces such as C12.22 optical cables. A precise list of networking supplies will be influenced by the interfaces and technology leveraged by the AMI product being evaluated.

3.13 Contracted Lab Work

During the reverse engineering process of a meter under analysis, testers generally lack the design documents for the device. They might find it prudent to make use of various laboratory services, such as microscopic or X-ray photography, to assess the device's nature and overall organization. Such services are relatively time-consuming and can be costly, so timing and budget should be adjusted accordingly within the scope.

3.14 Custom Equipment

Most but not all of the relevant equipment needed for testing will be available for purchase commercially. However, it is expected that custom electronic prototypes will be useful in testing and analysis. For example, packet sniffers are unavailable for many of the proprietary radio protocols which might be present on the NAN. Further, equipment for the quantitative measurement of leaked infrared radiation could answer the question of exactly how much data will be leaked by a given infrared communication framework. To deal with these specialized needs, the testing team must be pragmatic, creating hardware equipment and the related software to operate it as necessary for testing.

4 Vulnerabilities

Throughout the analysis of AMI system components, the attack team must apply a common set of testing principles to confirm or negate the presence of vulnerabilities in the target systems. The nature of vulnerabilities to be evaluated can be broadly differentiated as being the fault of either design or implementation.

Design flaws are those vulnerabilities that stem from the underlying architectural concepts of a system. Design security flaws are made at multiple levels, including chip-design, firmware, protocol, usage etc. Such flaws are often the fault of a standard or protocol, rather than any particular implementation; therefore expected to be found in many implementations regardless of processor, language, or programming competency. For example, a communications protocol that allows access to key elements of the meter without authentication would contain a design flaw. Frequently, such issues are caused by ease-of-use and other considerations being given priority over security. Traditional examples include the Windows Messaging subsystem neglecting to authenticate senders of system messages leading to so-called “Windows shattering attacks”, the Address Resolution Protocol's vulnerability to cache poisoning, and the use of clear-text password authentication in telnet sessions.

Implementation flaws, by contrast, are vulnerabilities that are caused by programming mistakes. These vulnerabilities are most commonly the results of programmers failing to consider or to understand the full impact of their code. For example, suppose that a network protocol mandates that a given field be only 150 bytes in length and NULL-terminated. Many programmers will assume that the field is only 150 bytes long; therefore, they might use the standard C function strcpy() to copy data from one location to another. By crafting special network frames, an attacker who includes 160 bytes in that

field before NULL-termination could then overwrite the ten bytes immediately following the memory buffer created for that field. This is known as a buffer overflow. There are many causes of buffer overflows, most of which are made possible by this abuse of API calls. Implementation-flaws lend themselves to automated analysis better than design-flaws, as the same types of programming flaws occur in many types of programming, from decryption-routines to authentication to message-parsing.

The delineation between design and implementation flaws is important, because the thinking and analysis process for identifying these faults differs greatly. While implementation flaws may be discovered through fuzzing, emulation and other programmatic analysis, identifying design flaws often requires a greater understanding of system functionality, design, and implementation. Throughout the evaluation of AMI device security, analysts must consider both design and implementation flaws.

4.1 Plaintext NAN Traffic

Due to the rapid development and changing implementation specifications for the AMI product space, vendors are responsible for a number of implementation decisions that strongly influence the security of the system. Among these decisions, vendors must choose how they implement privacy and integrity controls to protect confidential data between a meter and backend systems. In most modern AMI systems, vendors encrypt most, if not all traffic transmitted over the NAN, however, some vendors give the utility the option to disable encryption in their configurations or may charge addition expenses to enable encryption. Also, in many first generation AMI systems did not use encryption.

The attack team will leverage various traffic capture tools to collect NAN traffic for a given target, collecting information over wireless interfaces or other data sources. Once data is collected, cryptographic evaluation functions such as data entropy analysis and payload histogram analysis can be applied to evaluate the collected data, identifying the presence of plaintext or encrypted traffic. If plaintext NAN traffic is observed, the attack team will further evaluate the content of data to identify the nature and information disclosure impact of observed transmissions.

4.2 Bus Snooping

Embedded systems commonly use peripheral devices such as radios or EEPROM chips, interfacing with microcontrollers through SPI, I2C or other types of serial bus interfaces. Although this is a convenient and industry standard method for interfacing peripheral devices with each other, it represents a security risk for a device with little or no physical protection. For example, Figure 12 demonstrates the use of two electrical probes constructed from medical syringes connected to a protocol adapter to extract the firmware from an EEPROM device over an I2C bus. The extracted EEPROM data could contain executable code, configuration information or cryptographic keys, each of which could be stolen or modified.

From a security perspective, it is helpful for developers to evaluate the potential gains for an attacker through bus snooping. While many hardware engineers would recognize the risk of using external memory to boot a secure device, the same engineers use an insecure serial bus to connect a radio chip to a microcontroller. Many radio chip manufacturers, in effort to reduce development costs and accelerate platform adoption have implemented cryptographic algorithms internally in hardware. Implementing cryptography in hardware, which would be claimed as a security feature on many datasheets, has introduced a vulnerability: traffic between the microcontroller and the radio is left unencrypted on the bus. Using a bus sniffer, such as the device described in Section 3.3, an attacker is free to passively read information from the bus in an attempt to capture sensitive information.

In order to sniff packets on the network, an attacker must simply connect an appropriate bus sniffer to the serial interface between the microcontroller and the radio. By capturing the data transmitted over this interface, an attacker is able to observe all communications between the two peripherals, capturing radio configuration information, cryptographic keys, network authentication credentials and other sensitive data. This collected data can then be levered on third-party devices to extend the attacker's access into the target network.

Alternatively, an attacker could manipulate the target network by injecting new packets onto the bus. This provides them with a reliable communications mechanism to actively participate in the network, where any data originates from a legitimate node on the network. Through this mechanism, the attacker may choose to exploit any trust relationships established with the victim device, to deliver manipulated frames intended to exploit other systems, or for the manipulation of other services supporting the network and services it provides.

As a defense against this attack, several unified radio chips manufacturers have produced technology that includes both the microcontroller and radio within the same physical package. These chips, which include the TI CC2480A1, Freescale MC1322X, and Ember EM250, limit the effectiveness of this style of attack by protecting the bus connecting the radio to the microcontroller. However, many radios include hardware-sniffing or similar functionality into the chip, and these unified chips may still communicate with other devices over exposed data buses. The Ember chip documentation recommends enabling sniffer modes and pins by default for rapid debugging and analysis, providing detailed register settings required to do so [7].

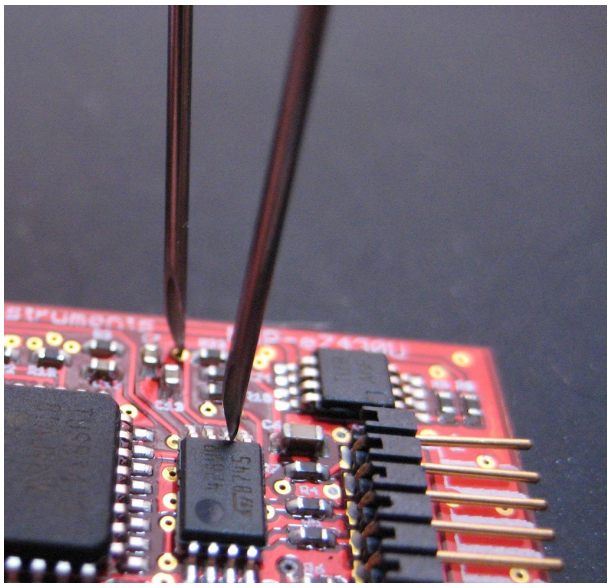

Figure 11. Snooping an I^2C bus.

4.3 Improper Cryptography

While it is often trivial to identify the absence of cryptography as described in Section 4.1, it is significantly more difficult to detect cryptography which is present but improperly used.

Consider, for example, the Debian/OpenSSL vulnerability in which the OpenSSL RNG was mistakenly crippled. This critical vulnerability limited the number of possible keys of a given type to fewer than forty thousand, a number sufficiently small for an attacker to generate and store all possible keys in advance. Further, this issue went unnoticed for two years before being repaired, allowing many organizations to deploy systems with vulnerable keys unaware of their risk and exposure. Despite later patches that resolved the RNG flaw, users who have not replaced all keys generated with the flawed software remain vulnerable [12].

Throughout the analysis of AMI-related technology, the attack team should evaluate multiple cryptographic mistakes that would otherwise threaten the integrity of the system.

4.3.1 Weak Key Derivation

Many cryptographic algorithms require unpredictable data as an input to the key derivation functions responsible for creating symmetric or asymmetric keys. Without sufficiently random content as an input, all keys used by the algorithm are suspect, allowing an attacker who can reproduce the input data to reproduce keys and decrypt data or otherwise impersonate trusted devices. Weak key derivation has been observed in cryptographic implementation-flaws such as the OpenSSL/Debian flaw described earlier, as well as algorithmic flaws in the DES, Blowfish and RC4 ciphers [6].

The attack team should evaluate the input values used for deriving keys at device initialization time, measuring the entropy of input data and evaluating the Chi-square test results to evaluate the randomness of data.

4.3.2 Improper Re-Use of Keystream Data

In stream ciphers, key stream data cannot be re-used without threatening the integrity of the cryptosystem. For performance reasons and in through implementation mistakes, past cryptographic protocol implementations have re-used key stream data, thereby allowing an attacker who observes a plaintext/ciphertext pair to recover the plaintext of an unknown ciphertext value. This flaw was identified in late version of the Windows NT operating system, allowing an attacker to decrypt locally stored passwords [14].

This vulnerability can also extend to block ciphers in Cipher Block Chaining (CBC) mode where the initial initialization vector is re-used, although this is generally less common.

The attack team should evaluate the use of key stream data to identify inappropriate re-use, identifying areas of protocols and system components that are threatened by this flaw.

4.3.3 Lack of Replay Protection

Some cryptographic primitives accept received data as valid after checking that the data decrypts properly. Without additional verification functions such as sequence enforcement, the algorithm is vulnerable to replay attacks, where an attacker can capture a valid encrypted data stream and retransmit the content. This vulnerability affects both stream ciphers and some modes of block ciphers, and has been observed in many cases including an implementation flaw in the FreeBSD IPsec stack where an attacker who observes encrypted data may retransmit the data repeatedly, potentially manipulating the source and destination systems [14].

The analyst can identify systems vulnerable to replay attacks by identifying the lack of unique identifiers in each cryptographic frame, or by observing repeated ciphertext content transmitted by one or more sources. Active analysis can also be used to evaluate replay attack vulnerabilities by replaying valid cryptographic data and observing the system state and operation after receiving the replayed data.

4.3.4 Insecure Cipher Modes

While some encryption algorithms are considered secure, such as the Advanced Encryption System (AES) cipher, the use of a cipher in an insecure mode can threaten the integrity of the cryptosystem. For example, the AES Electronic Cookbook (ECB) mode is considered weak, allowing an attacker to deduce repetitions plaintext content from repeated ciphertext blocks.

Through the analysis of firmware and cryptographic chips, the attack team should identify ciphersuite modes used in AMI technology, identifying insecure modes that represent a threat to the system.

4.3.5 Weak Integrity Protection

Many encryption algorithms, particularly stream ciphers, do not validate the content of decrypted content without a separate integrity check function. By transmitting an integrity check value (ICV) associated with the plaintext content, the receiving station can decrypt data and validate the resulting plaintext against the observed ICV.

The use of weak integrity check functions allow an attacker to manipulate ciphertext data, allowing them to selectively modify data while preserving a valid ICV (so-called "bit flipping" attacks), and may allow an attacker to decrypt ciphertext without knowledge of the encryption key. These vulnerabilities have been observed in the Temporal Key Integrity Protocol (TKIP) used by the IEEE 802.11i protocol for wireless networks, allowing an attacker to decrypt arbitrary frames [5].

4.3.6 Insufficient Key Length

Encryption ciphers may prove to provide inadequate protection against an attacker if an insufficient key length is used. This is most notable in symmetric ciphers such as the Data Encryption Standard (DES) where an attacker with approximately $15,000/USD in available computing resources can recover a DES key in approximately 12 hours [12]. Asymmetric ciphers are also vulnerable to insufficient key length attacks, where, at the time of this writing, it is possible to factor the prime numbers used for RSA 512-bit encryption within one month [19].

The attack team should identify the key lengths used for both symmetric and asymmetric protocols. The appropriateness of the key lengths will be evaluated based on the determined resources of a potential adversary, discussed in Section 2.2.

4.3.7 Cryptographically Weak Initialization Vectors

Stream ciphers including the RC4 algorithm have a key stream generation weakness when cryptographically weak initialization vectors (IVs) are used. If an attacker is able to identify consistent known plaintext in frames (such as protocol header information) and can collect a group of cryptographically weak IVs, they may be able to recover the encryption key used to protect data with fewer operations than the entire keyspace bounds.

The RC4 weakness in the selection of IVs was first publicized by Scott Fluhrer, Itsik Mantin and Adi Shamir [6], describing a vulnerability in the Wired Equivalence Protocol (WEP), once part of the IEEE 802.11 specification. This flaw was later widely exploited by tools such as the Aircrack-ng suite [1], a contributing factor to an attack against a US-based retailer which revealed payment card data for 45.7 million customers [11].

The attack team should evaluate AMI technology for the use of stream ciphers, investigating the length and section of IV values.

4.4 Direct Tampering

Tamper-protection mechanisms are intended to protect against malicious modification of a meter device, and should be part of the attack team analysis. As part of a defense-in-depth component, the tamperproof mechanisms represent an opportunity for a vendor to protect the consumer and utility from undesirable system modification.

With enough experimentation, tamper-protection mechanisms will inevitably fail. As will most defenses, the goal must be to afford significant response-time and detection ability. Appropriate tamper-protections will delay an attacker from fully compromising the integrity of meter-data for alerts to be responded to, while possibly forcing attackers to harvest multiple meters. Utilities must implement effective alerting abilities for tracking missing meters, including a means to identify and annotate the reason for missing meters. This head-end capability will allow for better reporting at the utility, and increase the likelihood of detecting stolen meters or tampering.

Tamper-protection mechanisms which should be evaluated include:

- Local tamper detection systems, including physical indicators that someone has tampered with a meter;
- Remote tamper detection systems, where a meter can remotely notify a head-end office that someone is tampering with a meter;
- System integrity protection systems, where a meter can protect the integrity of a system, including self-erasure of keys and firmware;
- Intended repair modes, used for authorized repair personnel from the utility company or vendor;
- Security of physical locks.

Vendors and utilities must assume that an attacker has physical possession of a meter, either through sale from the vendor being targeted, or as the result of theft from a customer premises. Based on the analysis of the first meter, the attacker will be better prepared to compromise a second device.

Unlike real-world adversaries, a contracted analysis team must maintain a delicate balance of time and monetary resources. If an analysis team is unable to devote the required time to successfully compromise the tamper-protections, it is in the vendor or utility's best interest to provide devices without tamper-protections. The analysis should report the effectiveness of the tamper-protections, and move on to deeper layers of analysis. Assuming an attacker with enough time and stolen meters can defeat the tamper-protection methods, the rest of the meter still requires analysis.

4.5 Stored Keys and Passwords

Due to the requirement for authentication, encryption and integrity protection on meter devices, it is reasonable to expect that authentication credentials such as passwords or derived keys and related encryption key material will be stored locally on the meter device. The presence of stored keys and password content represents an opportunity for attackers who wish to compromise the meter or extend their access into the NAN.

Depending on the authentication methods and cryptography utilized on the meter, various keys and authentication credentials may be recovered, including:

- *Asymmetric keys:* Asymmetric keys used in public key cryptography implementations are likely to be used for authentication and dynamic key derivation functions when a meter joins the NAN and potentially for upper-layer protocols. If attackers are able to recover the public/private key pair from a meter, they will be able to authenticate to the NAN while impersonating the victim meter, but will lack the ability to impersonate the NAN infrastructure.
- *Symmetric keys:* Symmetric keys are often used for shared-authentication methods, potentially present on the NAN but more likely to be found for protecting local interfaces for out-of-band management methods. Symmetric keys are often deployed in a distributed manner, where the compromise of a symmetric key may also provide access to a number of other devices sharing the same key.

With access to stored configuration and firmware content, the attack team can evaluate data stored on a meter to extract key content. Recovered keys will be an important component of the project exploitation phase as described in section 5.4.

Depending on the cryptographic algorithms in use and the mechanisms used for rekeying a device, an attacker may insert itself in the middle of the rekeying mechanism to obtain new keys. For example, some PKI trust mechanisms may be vulnerable to MD5 and SHA1 collision vulnerabilities, allowing an attacker to generate a trusted root certificate authority. With their own CA signing certificate, an attacker can generate a new certificate for any entity in the PKI architecture, effectively allowing them to sniff and generate network traffic to/from any system.

4.6 Cryptographic Key Distribution

Enabling cryptography is not terribly difficult, as most radios support it in hardware and software libraries with these capabilities are readily available. Key management, however, is a much more difficult problem, one which is likely to be an implementation and management challenge for meter devices and associated technology.

In the simplest case, symmetric keys might be preloaded into the firmware of every meter of a region. An attacker who accesses the memory of a single device, such as one which is stolen from a neighbor, will have gained the keys with which to access the entire network. With this key content, the attacker will have the ability to access the NAN as an authorized meter, impersonating any other device using shared keys within the region.

Some deployments will use certificates, asymmetric keys and some fashion of Public Key Infrastructure (PKI) to validate the authenticity of another device's certificate. This arrangement is similar in design to SSL as used in the HTTPS protocol to secure web-based transactions. An attack threatening these systems includes the ability to spoof system update mechanisms for the insertion of an unauthorized certificate (or certificate authority certificate), allowing an attacker to decrypt and inject encrypted traffic to or from any system.

4.7 Insecure Primary Interfaces

Because infrared is very short-ranged communications, some vendors assume that its communications are less likely to be sniffed or altered by an attacker. Based on this faulty assumption, some vendors pay considerably less attention to the network security of a meter's infrared port than to its radios. As such, the infrared port is likely to be an effective attack vector.

For example, the ANSI C12.18 Protocol Specification for ANSI Type 2 Optical Port specifies that a password be used to protect each meter from unauthorized access [3]. Given that such passwords are likely to be shared among meters, an attacker with access to the firmware of one meter would then gain the ability to communicate with other meters over infrared. As ANSI C12.22 is intended for use over more easily sniffed networks, it uses cryptography to obscure the channel during this authentication [4]. Unfortunately, the more important of the two vulnerabilities remains: every node contains a table of symmetric keys which can be used to access other meters.

Had asymmetric cryptography been used in the standard, a meter with a hand-held reader's public key would have enough information to verify an authorized user, but not so much information as to be able to impersonate one when compromised.

An attacker may also target other well-documented interfaces, such as those defined by ANSI C12.22 between the meter and the communications module.

4.8 Meter Authentication Weaknesses

The process of validating authentication credentials between a meter and NAN device will require many steps, all likely to take place before the use of cryptography in the network. Often, authentication processes appear architecturally sound, but can be manipulated by an adversary to impersonate legitimate devices, expose systems to denial of service conditions or to gain information which can later be used to undermine cryptographic protocols.

As part of the attack team analysis, the process of meter authentication will be scrutinized, identifying areas where insufficient techniques are utilized to protect against authentication-related attacks targeting current or future authentication exchanges. Several common weaknesses in authentication exchanges will be evaluated, including but not limited to:

- Poor nonce selection;
- Replay attack conditions;
- Memory exhaustion denial of service vulnerabilities;
- Protocol version negotiation manipulation;
- Authentication credential precomputation vulnerabilities;
- Predictable authentication credential deficiencies;
- Failed authentication account lockout and logging weaknesses;
- Certificate trust and validation weaknesses;
- Certificate revocation list checking practices;
- Plaintext authentication credential disclosure;
- Tunneled authentication protocol binding vulnerabilities.

4.9 NAN Authentication Weaknesses

While it is important to authenticate the meter device to the NAN before granting network access, it is also vital to perform mutual-authentication where the NAN is authenticated to the meter device. Failure to authenticate the NAN to the meter may result in an attacker's ability to manipulate one or more meter devices by impersonating the NAN and supporting back-end infrastructure components. Further, deficiencies in the order of NAN and meter authentication may reveal sensitive authentication credentials which could be leveraged by an attacker to bypass authentication restrictions.

Further, the varying architectures of NAN implementations will complicate the techniques and protocols used for authentication. In some cases, meter devices will authenticate to a NAN collector device, or will leverage wide-area networking (WAN) protocols such as WiMAX or 2G/3G connectivity to communicate directly with the headend. Other deployments leveraging mesh technology delegate routing, traffic forwarding and node authentication functionality to each meter in the NAN. In the latter case, secure authentication of a new node represents a significant challenge since any new node could be an attacker, wishing to manipulate communications within the NAN.

During the analysis of the NAN authentication process, the attack team will examine both the authentication properties and potential vulnerabilities described in Section 4.8, as well as the properties of mutual authentication for NAN. In the case of mesh networking, the functions for authenticating new mesh routing nodes will be examined, as well as the traffic routing and topology discovery behavior that represents manipulation opportunities for an attacker.

4.10 Firmware Implementation Flaws

Implementation flaws in software and related firmware are a common source of security vulnerabilities, with an impact ranging from denial of service conditions to full system compromise where an attacker can execute code of their own choosing on a target device. During the attack team analysis of meters and related AMI technology, several software-related vulnerability categories will be evaluated to identify a variety of flaws, including but not limited to, buffer overflow, off-by-one overwrite, format string, and integer overflow vulnerabilities.

Each of the described implementation flaws represents an opportunity for an attacker to exploit the target system. While the opportunities to an attacker will be influenced by the exact conditions of the system at the time when the flaws are exploited, it is reasonable to achieve one or more of the following conditions:

- Denial of service;
- Unexpected functioning of the device;
- Manipulation of system variables;
- Arbitrary code execution of the attacker's choosing;
- Manipulation of local system function calls or functionality.

4.10.1 Buffer Overflow Conditions

A common source of vulnerabilities resulting from the under-allocation of memory buffer space or through the inappropriate use of system functions calls, a buffer overflow condition is exemplified when an attacker can influence a quantity of data to be written to a buffer that exceeds the intended buffer length. Buffer overflow conditions are common in the C programming language string handling functions, but are also found in functions with iterative loops that do not properly check the bounds of a destination buffer.

4.10.2 Off-By-One Overwrite Conditions

Related to a buffer overflow vulnerability, an off-by-one overwrite is a condition in which one iteration of a loop is executed, often allowing an attacker to write data beyond the end of the intended buffer. This condition is most frequently caused when the developer fails to evaluate the behavior of their code at the boundaries of an allocated buffer of memory. An off-by-one overwrite may simply cause a denial-of-service, or allow an attacker to overwrite vital data, sometimes a function pointer which can be used to divert execution to an attacker-provided buffer of malicious code.

4.10.3 Format String Vulnerabilities

Format string vulnerabilities are found when a developer does not provide a format-string for any of the various format-related C library functions (e.g. printf) which write attacker-provided data to a screen, printing it into another variable, or storing it in a log file.

Using a format string vulnerability, an attacker may provide a format-string with several variations of the "%x" format directive to discover data higher in memory than the format string. However, because the C libraries provide the ability to write arbitrary data using the "%n" format directive, an attacker may be able to overwrite vital data, including function pointers which could allow an attacker to execute arbitrary code.

4.10.4 Integer Overflow

An integer overflow vulnerability is a condition where a numeric value is manipulated such that it is larger than can be naturally represented in the given number of bits in its associated variable. Depending on the platform, an integer value that exceeds its maximum value may wrap, returning to 0 or another value that is the inverse of the maximum value, or it may saturate, retaining the maximum result despite additional arithmetic operations. Integer overflows may be used to cause buffer overflows, or otherwise confuse the processor, leading to arbitrary code execution or denial of service.

4.11 Name Resolution Deficiencies

In order to make the identification of hosts simpler for humans and to accommodate extensibility and flexibility in contacting computing systems, many types of name-to-address resolution protocols have been developed. DNS, the most common of such protocols used on the Internet, was not designed to accommodate any sophisticated level of security, and is well-known as a significant vulnerability source in many Internet systems.

The attack team will evaluate standards-based and proprietary name resolution methods implemented for local and global resolution used by meter devices and supporting infrastructure. Common vulnerabilities in name-resolution protocols include:

- Meter name resolution cache poisoning;
- Response manipulation through race condition exploits;
- Denial of service attacks against name servers;
- Gratuitous name resolution injection;
- Name server hosting services and other accessibility methods;
- Name server configuration and implementation;
- Name server entry manipulation.

4.12 Weak Default Configuration Properties

The implementation of strong security practices is often a challenging process, requiring detailed working knowledge of the environment, organizational tolerance for risk and an understanding of the impact of security features for other systems in use. As a result, few

products are developed with a strong security configuration by default, requiring the end-user or system administrators to enable or apply the desired security options. When deployed using the vendor-specified default settings, these systems are often exposed to multiple vulnerabilities.

In the analysis of both deployed AMI systems and the settings used by vendors for default configuration properties, the attack team will identify system settings that expose products in their default configuration. These settings may include:

- Default configuration authentication credentials including passwords and encryption keys;
- Insecure remote management and communication protocols;
- Inadequate logging settings;
- Default or other untrustworthy certificates and key pairs;
- Weak authentication for locally accessible interfaces.

4.13 Traffic Routing Deficiencies

Many modern networked systems rely on intelligent devices with dynamic route selection for the forwarding of traffic through the network. The security of the routing and forwarding decisions influences the integrity and availability of traffic throughout the network. If adversaries are able to manipulate traffic-forwarding decisions, they may be able to insert themselves into the network to enable man-in-the-middle attacks.

The attack team will evaluate the behavior of traffic routing, path discovery, topology enumeration and path availability introduction to examine the integrity of routing behavior in the NAN and greater WAN environment. Careful evaluation will be applied to any opportunities where an adversary could influence traffic routing to manipulate large quantities of meter devices.

4.14 Denial of Service Threats

Denial of service threats are a common source of vulnerabilities in which an attacker is able to manipulate a system, preventing or prohibiting access to intended functionality. Many conditions are accessible to trigger denial of service conditions, which will be examined during the course of the attack team evaluation. While it is not believed that an exhaustive taxonomy of denial of service conditions will be documented (nor is it likely desirable), the most prevalent conditions will be identified, prioritized by the risk evaluation criteria specified in Section 2.2.

4.15 Information Disclosure Threats

While the use of cryptographic primitives serves to protect the confidentiality of data transmitted over a network medium, seldom does the confidentiality extend to the entire information exchange between nodes. Data including protocol headers and trailers as well as authentication and key negotiation exchanges are often transmitted in plaintext. This data potentially represents valuable information for an adversary, even without knowledge of the encryption keys in use. Not only can this information be useful by itself, it can also assist an attacker in exploiting other components of a system.

In addition to technical information, an attacker may also derive power usage information from the behavior of a meter. The ability to identify power usage presents a customer privacy threat, as this information may be leveraged to identify when customers are home. Furthermore, an attacker may possibly determine what types of appliances customers are using, and extrapolate additional sensitive information about the customer's usage patterns and living habits.

The attack team will evaluate data transmitted by AMI meters and related infrastructure over networks to identify information disclosure threats delivered without privacy controls.

4.16 Static Authentication Credentials

In addition to the default passwords evaluated in Section 4.12, the analysis team will also evaluate firmware and configuration data for the presence of fixed or static authentication credentials that cannot be changed by the end-user. The presence of such credentials has been identified in numerous vendor products, introduced and potentially forgotten for debugging and testing purposes, or simply as a result of a design flaw. The use of static credentials represents a threat to the technology adopter, potentially allowing an attacker to access the system through the use of these credentials.

4.17 Deficient Random Number Generators

The generation of random numbers is a vitally important operation for ensuring the integrity and confidentiality of cryptographic protocols. Unfortunately, many implementations fail to implement an adequate random number generation (RNG) function, threatening the security of all functions relying on random numbers.

During the assessment of meters and related AMI technology, the attack team will evaluate multiple facets of RNG implementations:

RNG Seed Initialization: A common flaw in the implementation of RNGs is the failure to adequately seed the process with unique and unpredictable content. When a RNG is seeded with a value that is predictable or otherwise limited in entropy, it may be possible for an attacker to predict future random values. The attack team will examine the

frequency of RNG seeding, as well as the data used to initialize the RNG to evaluate potential weaknesses that threaten the integrity of the system.

RNG Predictability Analysis: Using attractor analysis and information gleaned from the analysis of applicable protocols using random numbers, the attack team will evaluate a sampling of random numbers to determine the predictability of future random values. Using a metric of the data sampling space (the number of random values observed) and the number of potential guesses for an adversary, it is possible to evaluate the RNG to determine if an attacker can determine and potentially manipulate future random number selections.

RNG Value Chart: Using a variety of available tools, the attack team will chart the distribution of a sampling of random numbers to visualize attractors indicating deficiencies in the RNG selection. Weak RNG implementations with imperfections in the randomness of data will be characterized as having geometric patterns in the RNG value chart, indicating that the implementation does not select random numbers from the entire available value pool. By comparison, an ideal RNG will appear as a well-dispersed cloud of values, without significant patterns or attractors.

An example of a weak RNG implementation with distinct attractors is shown in Figure 12, contrasted with a well-distributed RNG implementation shown in Figure 13.

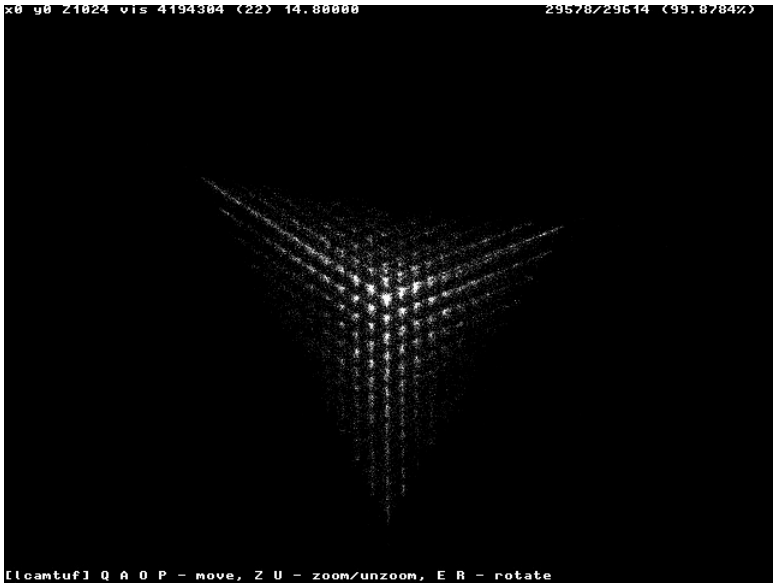

Figure 12. Random number charting indicating a weak RNG

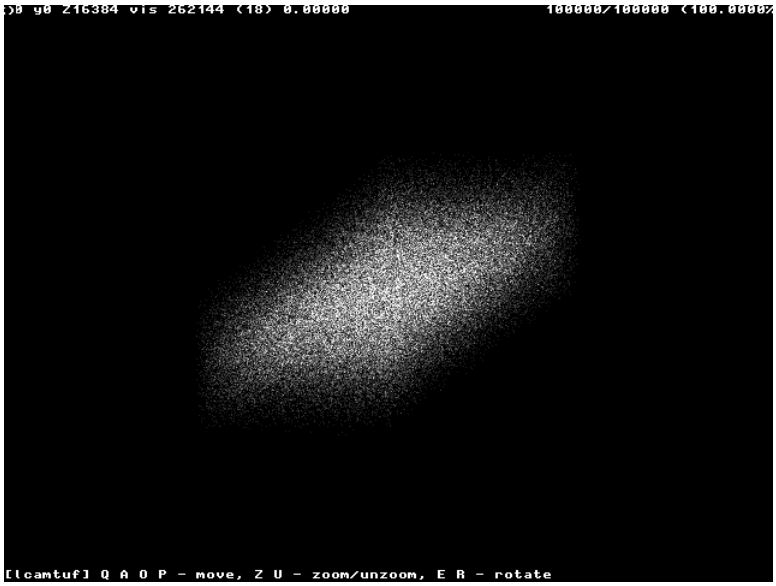

Figure 13. Random number chart indicating a well-distributed RNG

4.18 Network Time Services

A commonly overlooked aspect of security is a device's concept of the current time. Synchronizing time between many different systems has been the focus of significant research, yielding excellent technologies such as the Network Time Protocol (NTP), Global Positioning System technology (GPS). However, these services include their own weaknesses, representing opportunities for an attacker to manipulate their representation of time.

The NTP protocol is widely deployed for time synchronization throughout the Internet, but is vulnerable to a number of threats including time-server impersonation attacks, man-in-the-middle attacks, race conditions and key recovery attacks. Impersonating a GPS satellite using GPS satellite simulators allows an attacker to send false information to a target, manipulating the time on the target device [17].

By manipulating the time of a target device, an attacker may be able to incur several types of mischief and havoc, including metrology data overwriting and forcing polls for events such as firmware updates. These two types of attacks could be leveraged to damage the integrity of rate and billing data, and potentially cause various forms of denial of service including network, firmware delivery servers, and meters themselves.

5 Attack Methodology

With an appropriate toolset, a firm understanding of the attack surface, and knowledge of the categories of vulnerabilities commonly found in computing and network devices, the attack team is ready to apply the attack principles on specific AMI system implementations. The attack methodology can be categorized into four phases:

- *Reconnaissance:* In the reconnaissance phase, the attack team begins collecting pertinent information for the systems to be evaluated. Resources can be collected from publicly available sources as well as through analysis of accessible AMI hardware devices;
- Initial Analysis: Once the reconnaissance phase is complete, information about the devices including hardware configuration, accessible interfaces, adopted standards and more will be considered and identified for further analysis and vulnerability pursuit. In this phase, the security attack team examines the areas that represent potential risk areas on devices while searching through data collected during reconnaissance for valuable information including cryptographic key content, wireless configuration features and basic firmware and code analysis. At the end of the initial analysis phase, the analysts will have identified areas that require additional analysis that represent potentially fruitful attack avenues while potentially discounting other areas that may not be practically exploited or otherwise require resources to exploit that are outside of the scope of the investigation;
- Deep Analysis: The deep analysis phase is the most time-intensive part of the attack methodology, exercising an in-depth analysis of a target device. In this phase, the security attack team leverages the tools available to perform debugging

and tracing of devices including analysis of system firmware for the identification of potential vulnerabilities. Exploit development, when applicable, will be one component of the deep analysis phase to turn firmware vulnerabilities into privileged access in the HAN or NAN environment.

- Exploitation: Following the deep analysis phase, the attack team enters the exploitation phase of the analysis where developed exploits, attack tools and other techniques are used to attack the system.

Each phase of the attack methodology remains iterative; at any point, new discoveries or attack opportunities may require a return to analysis phases leading up to potential system exploitation. In order to most effectively target system components of interest, the reconnaissance phase is completed with as much detail as possible so as not to overlook potential attacker opportunities. Documentation and reporting are vital components, and should include the output of each attack methodology phase, as shown in Figure 14.

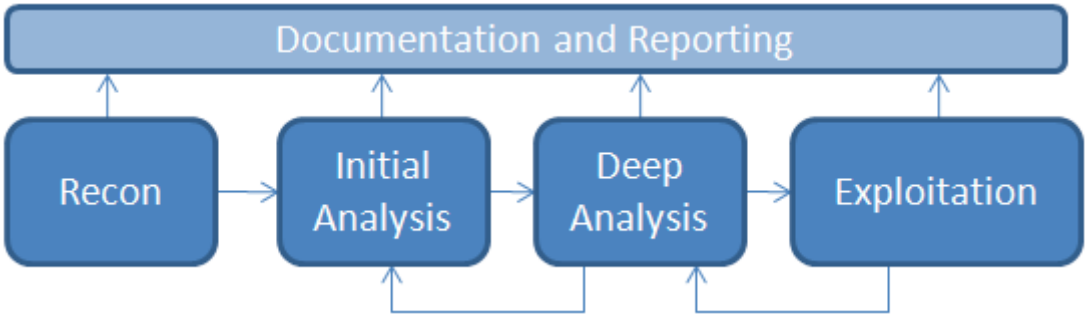

Figure 14. InGuardians' Attack Methodology

5.1 Reconnaissance

The reconnaissance phase establishes a baseline of data regarding the technology being evaluated, providing the security attack team with the resources to further assess meters and associated technology. In this phase of the analysis, the security team will leverage documentation from multiple sources as part of the initial technology enumeration, as well as various analysis tools, such as packet sniffers, which allow for a detailed observation of live equipment operation within a lab or production deployment.

5.1.1 Document Enumeration

Public documentation and marketing material provide a wealth of information pertinent to attack. New features in a product revision announcement often indicate code that is newer and less thoroughly reviewed for vulnerabilities. The team's understanding of an AMI component's features and intended behavior will provide insight into security weaknesses as well as context for binary code analysis.

Sources of public information pertinent to analysis include, but are not limited to:

- Marketing literature,
- Component datasheets,
- Component application notes,
- Operating manuals,
- Radio block diagrams,
- External and internal photographs,
- Operational descriptions,
- Schematic diagrams,
- FCC test reports, and
- Patent filings.

Available documentation will vary greatly between manufacturers. In some cases, an adversary may resort to attacking publicly accessible systems or collecting resources from trash receptacles for a target manufacturer to obtain access to otherwise private information, including internal support base access, firmware updates, unpublished press releases, and similar sensitive information. Evaluation of the security of enterprise computing resources for a particular vendor is beyond the scope of this document, but no vendor should neglect to secure their enterprise environments properly or to consider this threat to their products.

Throughout the document enumeration phase, the attack team should identify the sources of information that is collected. These sources may include information retrieved from publicly accessible-sources, from documentation released under NDA, through unauthorized information access or from word-of-mouth and casual conversations with informed personnel. By evaluating the sources of information leveraged for the analysis of target devices, vendors and utilities can evaluate their data prevention systems.

5.1.2 Hardware Overview

The visual inspection of a meter will reveal additional information about the configuration and use of the device. Items of interest include out-of-band management interfaces, tamper-protection measures, physical layout and connections, and antennas.

The following (incomplete) list provides a starting-point for analysis during this phase:

- Analyze the antenna size and shape, which reveals the intended band of operation
- Document which ICs (particularly microcontrollers and EEPROMs) are connected to the board, what they are connected to, and which pins are used to connect them

- Identify and document any interfaces to microcontrollers including JTAG, ICP/ISP, IR, RS232, Ethernet, SPI, I^2C, etc
- Identify physical organization of circuit-boards; related components are most often grouped together, providing context for determining their purpose
- Identify groupings of long traces on the circuit-boards; these are likely a bus of some sort, interesting for logic-analyzers and providing context to the relationship between sections of the circuit.

5.1.3 Communications RF Characterization

In order to perform network analysis (sniffing) and injection attacks, the attack-team must first identify key radio information, including frequency spectrum, modulation, channel selection, frequency hopping patterns, and higher-level protocols.

An example set of this information follows for a ZigBee implementation:

- Frequency range: 2.4GHz / 868KHz / 915KHz (802.15.4)
- Modulation: 2.4GHz: MSK, 868/915KHz: BPSK
- Channel Selection: Manual: 16/10/1 channels for 2.4GHz/915KHz /868KHz
- Channel Access: CSMA-CA
- Hopping Pattern: N/A
- Higher-level protocol: ZigBee (Security-Enhanced Profile)

The attack-team can find information available from the communication chip's datasheet and the FCC test reports.

5.2 Initial Analysis

During the second phase in the attack sequence, the team will begin to interact with the target devices. Using the information gathered in the reconnaissance phase, the team will leverage a variety of tools and techniques to identify vulnerabilities or technical areas worthy of further analysis (see Deep Analysis phase). While much of the work will be influenced by information gathered in the reconnaissance phase, several techniques will be useful for assaying the security of most target devices.

5.2.1 Case Tamper-Protection

Although it is not strictly necessary for the attacker to bypass tamper-protection, it is very valuable once infrared, radio, and key-management vulnerabilities have been corrected. For this reason, the team will investigate methods of preventing the detection and reporting of tampering.

5.2.2 Radio Packet Analysis

By use of either a compatible radio receiver or a bus protocol analyzer, the team will capture a number of digital radio packets, both from the NAN and the HAN. The presence or absence of cryptography should be immediately apparent by comparing just a few packets visually. The bits of an encrypted packet appear random; therefore, non-random bits indicate unencrypted traffic. Further, if bits are random within each packet, yet identical packets are seen to repeat, an analyst can deduce that the cryptography is likely vulnerable to a ciphertext replay attack.

During this phase of the attack, the team will most likely develop an attack tool based on the radio chip which will provide many different attack-vectors. This tool may be developed by simply co-opting the radio chip on another target-device, or by purchasing a development kit for the appropriate radio chip external to the target device. The attacks available once this tool is complete include packet-capture, injection, fuzzing and jamming.

5.2.3 Schematic Capture

Once in possession of a meter, the team should device thorough documentation of all integrated circuits, interfaces, buses, and peripherals. The intended functionality and purpose of each device will be further ascertained through hardware reverse-engineering aided by public integrated circuit databases and manufacturer documentation.

5.2.4 EEPROM Dumping

Serial flash and EEPROM memory devices on a circuit board provide no means of protection, and they are thus immediately available to any attacker who cares to dump them with the equipment described in Section 3.1. The team will review the use of such devices, manipulating them by a variety of method to extract interesting data.

The simplest of these attacks to perform, photographed in Figure 11 on page 29, involves the use of two electric probes and a shared ground to dump an I^2C EEPROM's contents. The multi-master features of I^2C allow this to be performed while the device is still active. The two probes, modified hypodermic syringes, are tapping the Serial Data (SDA) and Serial Clock (SCL) lines.

SPI is more difficult to tap, yet not insurmountable. Because it lacks the multi-master mode of I^2C, any attempt to read or write the memory chip will result in interference from the master microcontroller. Three methods exist by which the memory chip may be accessed without the microcontroller's interference.

First, the EEPROM may be desoldered by use of the SMD rework station described in Section 3.8. Once removed, it can be soldered to a fresh board or connected to a ZIF socket and its contents extracted to a computer, as shown in Figure 15. While this method is certainly among the easiest, two alternative methods are available, each with a discrete advantage.

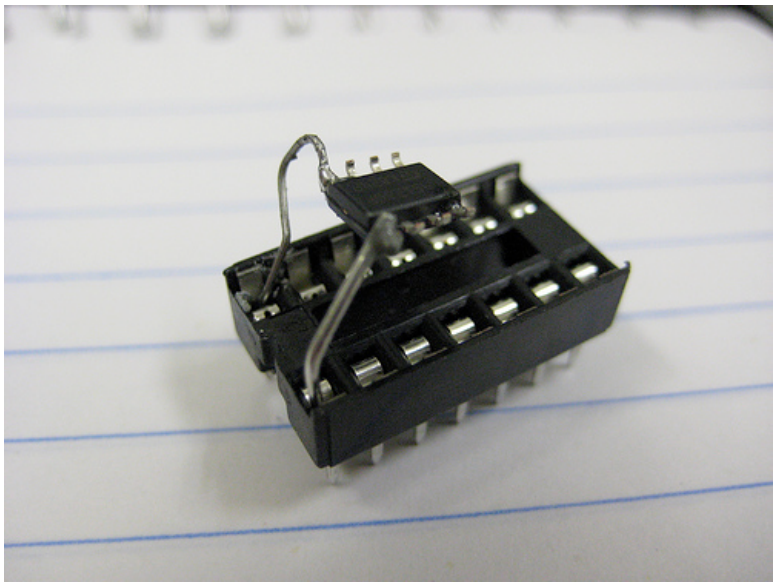

Figure 15. Extracted EEPROM Connected to ZIF Socket

The second method is a passive tap, in which the MISO and MOSI lines are observed but not modified. In this configuration, memory requests are recorded. The analyst has the advantage of knowing the sequence of memory fetches and writes, but it is inconvenient for the tester to change a request.

A third method involves lifting the CE (Chip Enable) pin of either the microcontroller or the memory chip as shown in Figure 16. This allows the chip to be disabled temporarily, causing I/O pins to switch to a high impedance mode. By lifting the CE pin of a memory chip and soldering another into the bus, the attacker can temporarily replace a surface-mount chip with a ZIF socket holding a DIP chip that can be quickly replaced for experimentation.

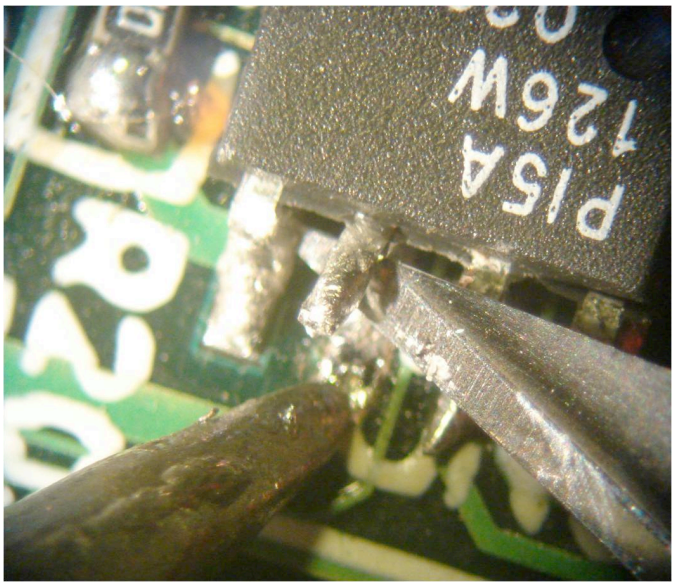

Figure 16. Lifting the Chip Enable Pin

5.2.5 Microcontroller Dumping

In addition to the valuable information found in external, serial EEPROM chips, many microcontrollers and system-on-chip (SOC) devices hold very useful information. In some ways, this data is more valuable in that often it is the only source of executable code. Due to the increased complexity of these devices and less standardization between vendors, many different interfaces exist for programming and debugging these chips. The following are some of the most common.

JTAG, also known as the Joint Test Action Group or IEEE 1149.1, is a protocol for accessing test-points within an ASIC chip, debugging a microcontroller, and programming the memory of a device, such as a microcontroller or an FPGA. While most microcontrollers have a JTAG fuse which can be blown to disable debugging access, many manufacturers leave the fuse intact for debugging purposes. The fuse itself might be physical, in which case it can be bypassed with an invasive micro-electronic probe station, such as the one shown in Figure 17. It might also be an EEPROM cell, in which case semi-invasive optical attacks can often reset the chip to its unprotected state. With access to the JTAG interface, the attack team can utilize the appropriate Flash Emulation Tool (FET) to retrieve the contents of volatile and non-volatile memory for further analysis. An example of the MSP Flash Emulation Toolkit used for retrieving the firmware of a MSP430 device is shown in Figure 18.

Spy-Bi-Wire and similar protocols are variants of JTAG which require fewer wires. This is made possible by the use of bidirectional I/O. Spy-Bi-Wire is popular in newer TI chipsets.

Serial Bootstrap Loaders are a software method by which microcontrollers may be programmed. Rather than requiring custom testing hardware, such as that used by JTAG,

a bootloader is placed within the permanent masked ROM of each chip. The chip can then be programmed with little more than a level-converter and a serial port. Bootloaders are often vulnerable to timing attacks which allow code to be forcibly extracted [6]. Further, voltage-glitching attacks can be used to skip individual instructions of the bootloader code, allowing software protection measures to be bypassed [10]. Serial Bootstrap Loaders are common in many microcontrollers, and have even been added to some architectures which do not include them from the factory (e.g. AVR ATmega processors when used in Arduino toolsets).

I^2C, SPI and other protocols are sometimes used for the programming of microcontrollers. Such chips rarely offer a code-protection feature. InCircuit Serial Programmers (ICSP), common in AVR and PIC microcontrollers, often use the SPI protocol and wiring.

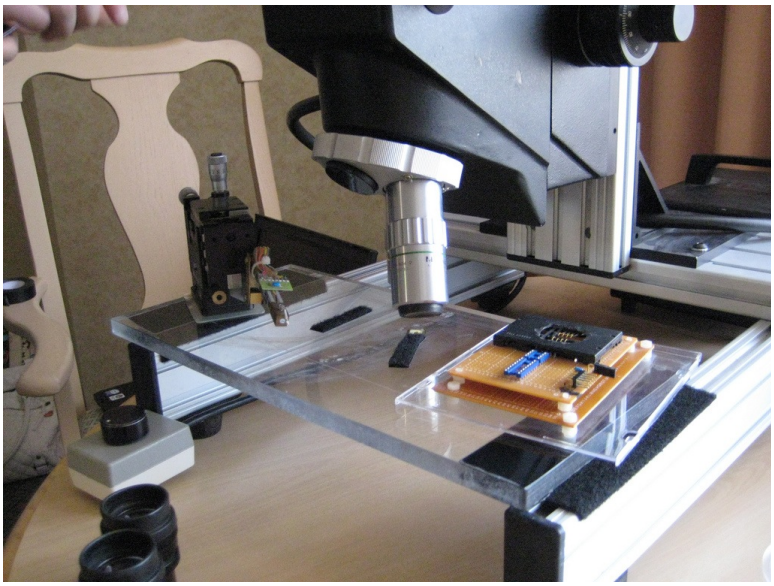

Figure 17. Flylogic Engineering's portable probe station

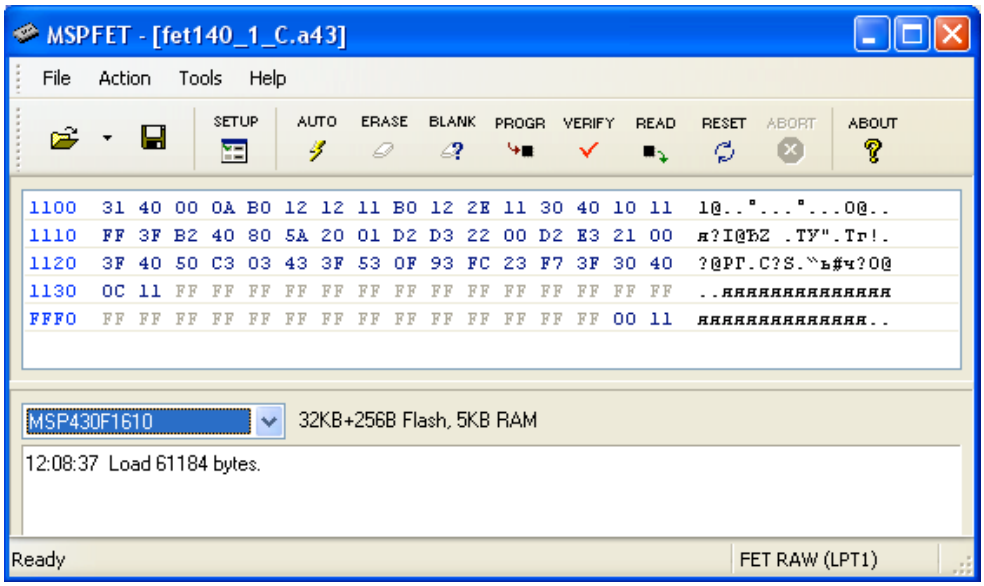

Figure 18. MSP430 Flash Emulation Toolkit

5.2.6 *Bus Snooping*

Bus snooping, described in depth in section 4.2, is performed at this stage. While the potential discoveries from bus-snooping could include numerous categories of flaws, the attack-team at this point will focus on looking for configuration and security information such as radio configuration, crypto-keys, and other protocols which give context for how the device operates and allow control of the various components on the circuit-board.

5.2.7 *Network Scanning*

Interacting with a meter network, such as by C12.22, requires the possession of a network adapter capable of communicating using the supported network protocols. For versatility, the team might find it expedient to build such an adapter that may be used with compromised authentication credentials to access NAN resources. Such a device would be configured with the ability to enumerate and identify other devices on the NAN, representing privilege escalation opportunities for the attacker.

5.3 Deep Analysis

Using the information gathered in the initial analysis phase, the attack team can then identify system functions and potential vulnerability areas that demonstrated merit with a reasonable probability of further success in exploiting the system. The actionable tasks for the deep analysis phase are strongly influenced by the initial analysis phase, but will reasonably consist of several techniques believed to be likely attack vectors.

5.3.1 Fuzzing

Fuzzing is the art of interacting with an application in unorthodox and sometimes random ways often causing a vulnerable application to crash or otherwise behave incorrectly. Fuzzers often inject increasingly large data into buffers attempting to cause an overflow, manipulate numbers in protocol fields to cause an overflow or underflow condition, and/or insert special characters into various fields hoping to cause some unexpected control sequence. Fuzzing should be performed wherever interaction with the target system is allowed, including:

- Network interaction over radio interfaces;
- Direct system bus interaction;
- Local infrared management ports;
- Local serial management ports.

While vulnerabilities which can be remotely exploited are the highest value due to the scale of attack potential, vulnerabilities which require touching an individual meter are also valuable.

5.3.2 Key Extraction

Since it is known that the C12.18 and C12.22 leverage a shared secret for authentication, the attack team will carefully review the firmware and EEPROM data contents for that secret. A combination of ASCII data analysis for plaintext C12.18 password storage and code block reference analysis (such as identifying references made by a function known to perform authentication) will be used to recover authentication credentials.

The locations of cryptographic keys will be identified through the use of data entropy analysis. Since cryptographic keys have a high entropy level compared to that of executable code or other non-random memory contents, graphing the distribution of entropy throughout firmware or EEPROM data will reveal the approximate location of key content with a peak of high entropy, as shown in Figure 19. In this example, an unusually high entropy level is detected at approximately 44,000 blocks offset, corresponding to the location of an asymmetric RSA key stored in memory.

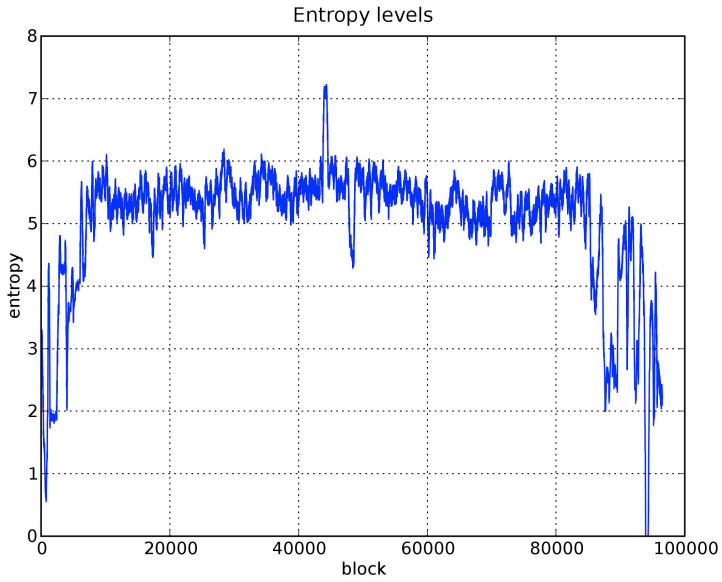

Figure 19. Entropy visualization for an example binary

5.3.3 *Fault Tracing and Enumeration*

Microcontroller software can be subjected to fuzzing techniques, just as workstation software can. The difficulty, however, lies in identifying when a fault has occurred. The team will accomplish this by two methods. First, many chips reset pins to high-impedance input states on a reset. Second, software may be extracted from a microcontroller and run within a simulator, where the analyst can observe other deviations from proper execution. Once a fault condition has been identified which will crash the victim machine, the exact behavior of the test case packet is analyzed to determine the vulnerable region of code. Once this is understood, a new packet can be crafted to inject executable code or perform a similarly malicious function.

5.3.4 *Power-Glitching Attacks*

Due to the operating characteristics of microcontrollers, manipulating the power input to a microcontroller may cause it to behave inappropriately. Carefully executed, an attacker can manipulate this behavior in a predictable fashion. For instance, providing slightly less than the required power for a given microcontroller to operate may allow the microcontroller's instruction pointer to increment but not perform the instruction.

An attacker may leverage a power-glitching attack to manipulate the system microcontroller to skip authentication failure processing routines or other undesirable instructions, granting access to meter and NAN resources without adequate authentication credentials.

5.3.5 *Clock-Glitching Attacks*

If a microcontroller is configured for use with an external clock, an attacker may be able to manipulate the system behavior to their benefit. For example, if an attacker forces two

clock signals at a rate that is slightly faster than the target microcontroller can accommodate, they may be able to manipulate the system to skip the execution of a particular instruction. In this technique, an attacker may use a digital I/O pin from an "attack" microcontroller to provide the accelerated clock for the target microcontroller.

Similar to power-glitching, this technique may provide the ability to skip failed-authentication routines or other undesirable instructions, granting the attacker greater control over the system.

5.3.6 Firmware Binary Code Analysis

An attacker can only understand a limited amount about an application without analyzing the code that makes it work. Disassembling firmware is the act of converting machine-language processor instructions back to human-readable assembly language. Once the firmware is obtained, running it through disassembly routines will allow for much deeper analysis, providing both data and context to target-device operation.

Once an appropriate disassembler is obtained, the firmware will be analyzed for vulnerabilities, and used to provide context to other analysis and attacks. For instance, the attack team may use disassembled firmware first to identify the code which implements a particular feature, and then use that code to extract pertinent cryptographic key information from a specific location in an external EEPROM. The attack team will also look for memory copy loops which may be manipulated to cause a buffer overflow. Analyzing firmware will also help to identify where appropriately timed power-glitching and clock-glitching attacks may be most beneficial.

5.3.7 Simulation

Through the use of available system emulators, the attack team may simulate the hardware of a meter with extracted firmware to aid in the analysis of firmware functionality and response to malformed stimuli through fuzzing. By using an emulated environment, the attack team will have access to more sophisticated monitoring and debugging tools which may prove useful in the development and testing of exploits in a controlled environment.

5.3.8 Firmware Source Code Analsys

Static analysis of the firmware's source code also provides an in-depth understanding of the firmware that may not be obtained through code generated during the disassembly of the firmware. This type of evaluation can identify exploitable functionality that may not be accessible during selected configurations of the device during assessments and penetration testing. Static source code analysis typically identifies exploitable vulnerabilities such as, but not limited to, buffer overflows, race conditions, authorization weaknesses, and unnecessary functionality. A good example of firmware functionality that benefits from source code analysis is the device's encryption technologies. The following list demonstrates the focal points the attack team would use to initiate to evaluation the firmware's encryption capabilities:

- Random number generation and usage,
- Key and certificate generation and distribution,
- Initialization Vector generation and incrementing,
- Key and certificate revocation and rotation, and
- Key and certificate data storage.

5.3.9 Exploit Development

Based on the vulnerabilities discovered through firmware code analysis, tracing and enumeration, runtime evaluation and fuzzing, the attack team will develop and evaluate exploits to manipulate an unmodified meter mirroring a customer premises deployment environment. The purpose and intent of payloads will vary, but may include meter denial-of-service conditions requiring a field service visit to repair, power service termination or instantiation, service utilization reporting manipulation or other desirable conditions for an attacker.

5.4 Exploitation

Exploitation will often consist of piecing together multiple attack-vectors discovered during earlier phases to leverage false-assumptions, design flaws and poor programming. As such, this necessarily is the least descriptive section of the document.

The exploitation phase is the culmination of all the other phases of attack, with the express goals of disrupting value-streams as defined by [1]. While business value-streams are the ultimate goal, gaining control over a meter, collector/bridge, a meter's data, or a network are the basic building blocks of most attacks.

When evaluating potential directions, the attack-team should begin with knowledge they've gained about the target architecture or device; knowledge about how it works, how it was intended to work, and how outside influence can be applied to modify the behavior of the system. Successful exploitation comes from creatively piecing together different weaknesses to form one or more substantial attacks.

While this phase will differ with each engagement, several potential attacks are worth mentioning, as their value at this stage is easily understood.

Note: Because engagements have set time-limitations, the attack-team will likely not be able to develop exploits for every promising attack. These undeveloped attacks should be documented with indications of vulnerability and potential mitigations.

5.4.1 Unauthorized Device Authentication

Authentication allows varying levels of access to any given system. The ability to circumvent or otherwise manipulate the authentication process to gain access to a device is a primary goal. Potential methods of unauthorized authentication include buffer overflows in networking or serial communication, power-glitching to skip out of a failed authentication loop or validation routine, or identifying an outlier condition which either skips validation or the authentication process completely.

5.4.2 Malicious Patching of Firmware

If an attacker successfully updates the firmware of a device, they can completely control the device. Authentication routines are no longer an issue and other vulnerabilities are not required. The discussion immediately revolves around the methods an attack has to update the firmware, and how much work and time is involved to update 100,000 devices.

5.4.3 Manipulation of Reported Data

Power is stolen every day. While new and exciting ways of stealing energy is interesting, manipulating the reported data, either in-meter or in-transit, opens many other questions. Perhaps the bigger problem is charging customers 100 times more than they should be. Perhaps the problem is reporting negative usage causing a utility to pay the customer. Many other interesting possibilities exist.

Such manipulation may take place using wireless man-in-the-middle (MITM) style attacks as well as by attacking the bus on the circuit board.

5.4.4 Collector/Network Gear Impersonation

By convincing meters that you are a collector or other control system entity, an attacker may be able to force control messages to perform such tasks as turning off the power latch, updating firmware, or attacking HAN devices. Potential ways of successfully executing such an attack may include stealing symmetric keys from a meter, private/public keys from a legitimate collector, or any other such authentication information.

6 Conclusion

Vulnerabilities exist in any complex system. Identifying and evaluating weaknesses allows for these vulnerabilities to be addressed, protecting customers, utilities and vendors alike. The stakes are very high, impacting the ability to generate, distribute, and consume energy in a given region.

Security analysis is inherently a snapshot in time. Because systems change over time, and people have access to changed systems with and without authorization, vendors and utilities should perform regular reviews of their architectures, using both internal and third-party attack teams. Because of the creative and complex nature of this type of security analysis, regular reviews will likely find more vulnerabilities than only a one-off analysis, even if nothing is changed in the system itself. Changing the scope or approach between engagements can further serve to improve findings. For example, including source-code analysis to a previously "black-box" only analysis can change the attack-team's perspective and improve findings. Alternately, performing a "black-box" analysis even after analysis with source code will likely yield new findings, because the perspective has changed, removing inevitable assumptions about the target system.

InGuardians recommends utilities and vendors perform regular penetration tests on the following infrastructure:

- Physical Security Perimeters
    - Utility Data Centers
    - Utility Control Centers
    - Substations

- Utility Premises Backend Systems
  - Head-Ends
  - Meter Data Management Systems (MDMS)
  - Customer Resource Management (CRM) Systems (SAP, JDE, etc…)
  - Customer Portals
  - Corporate Network and Systems
  - Internet-facing systems
- Field Deployed Devices and Systems
  - Neighborhood Collectors
  - Bridges
  - Supporting NAN and WAN Networking Equipment
  - Utility Meters Data Collection Devices
  - Supporting HAN Technologies

To achieve the best results, an attack-team requires appropriate funding, project scope, time, toolset and skill set. Management must understand the impact of scope on the results of an analysis. While third-party analysis can often yield more professional and fruitful results, internal attack teams should be built and challenged. Their internal knowledge can help discover more obscure weaknesses and the experience gained from attacking a system will positively impact the rest of their work. Internal security evangelism is born from such knowledgeable employees.

InGuardians hopes this document aids you in your efforts to secure your portion of the Smart Grid. If you have any questions or comments, please don't hesitate to reach out to us. Also be sure to check out website for updated versions of this document and other such resources.

Please direct questions, comments and suggestions on this document to:

InGuardians, Inc.
http://www.inguardians.com
+1 202-448-8958
security@inguardians.com