# Multi-vendor Penetration Testing in the Advanced Metering Infrastructure

Stephen McLaughlin, Dmitry Podkuiko, Sergei Miadzvezhanka,
Adam Delozier, and Patrick McDaniel
Department of Computer Science and Engineering
Pennsylvania State University
University Park, PA 16802
{smclaugh,podkuiko,swm5344,delozier,mcdaniel}@cse.psu.edu

**Abstract -** The advanced metering infrastructure (AMI) is revolutionizing electrical grids. Intelligent AMI "smart meters" report real time usage data that enables efficient energy generation and use. However, aggressive deployments are outpacing security efforts: new devices from a dizzying array of vendors are being introduced into grids with little or no understanding of the security problems they represent. In this paper we develop an *archetypal attack tree* approach to guide penetration testing across multiple-vendor implementations of a technology class. In this, we graft archetypal attack trees modeling broad adversary goals and attack vectors to vendor-specific concrete attack trees. Evaluators then use the grafted trees as a roadmap to penetration testing. We apply this approach within AMI to model attacker goals such as energy fraud and denial of service. Our experiments with multiple vendors generate real attack scenarios using vulnerabilities identified during directed penetration testing, e.g., manipulation of energy usage data, spoofing meters, and extracting sensitive data from internal registers. More broadly, we show how we can reuse efforts in penetration testing to efficiently evaluate the increasingly large body of AMI technologies being deployed in the field.

## 1. INTRODUCTION

The Advanced Metering Infrastructure (AMI) is changing the way electric energy is produced, priced, and consumed. The introduction of digital sensors–*smart meters*–in homes and enterprises has allowed regional and national producers to more efficiently produce and deliver energy [18]. In short, the vast yet antiquated analog control system that has served electricity consumers for decades is entering the information age. Here AMI is evolving and being deployed quickly. In the US, the recent stimulus package allocates US $4.5 billion for smart grid technology development [25], with the energy sector making substantial additional investments. Similar efforts are under way internationally, with the EU, Canada, and China launching broad initiatives in recent years. Such expenditures are driving the dizzying array of new products that reach the market almost every day.

The transition of electric meters to digital systems is not without risks. New technologies offer new opportunities for adversaries to manipulate the grid to further their malicious ends. Moreover, deployments are outpacing security efforts: new devices and technologies are being introduced into grids with little or no real understanding of the security problems they represent. Current penetration testing efforts are piecemeal, ad hoc and often superficial. Not surprisingly, new vulnerabilities are being found almost as quickly as AMI products are being deployed [33, 20, 9].

Prudence critically demands better analyses of AMI system security: manufacturers and utilities must leverage modeling and analysis efforts for the large body of systems towards a global understanding of the security problems they represent. Efforts like the NIST smart grid guidelines [30] are a step in the right direction, but only identify affirmative steps for secure systems. They do not posit the causes and effects of critical vulnerabilities, nor identify a roadmap for offensive testing of smart meter technology. In the absence of guidance on these key issues, current industrial pen-testing strategies focus on specific vendor lines and are agnostic to critical security concerns–such as utilities' concerns with revenue protection from fraud and cost of operations.

In this paper, we design and execute a systematic penetration testing process for AMI systems and uncover a number of real attacks on commercially available systems. Our contributions in this effort include:

- We develop a new approach to guiding penetration testing. This approach uses vendor independent *archetypal attack trees* to model broad adversary goals and attack vectors, and *concrete attack trees* to instantiate specific attack subgoals on vendor systems.
- We develop archetypal and concrete attack trees for three important classes of attacks, ($a$) energy fraud, ($b$) denial of service, and ($c$) targeted disconnect. These trees represent practical (and in some cases trivial) attacks that can be carried out in widely deployed AMI systems.
- We identify from our penentration testing results of one and one half years a broad range of security vulnerabilities for two popular AMI vendors, and use them to instantiate real attack scenarios in fielded systems.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
*ACSAC* '10 Dec. 6-10, 2010, Austin, Texas USA
Copyright 2010 ACM 978-1-4503-0133-6/10/12 ...$10.00.

Representative attacks include the manipulation of energy usage data and signaling as it traverses public networks, spoofing meter identity, and physically extracting sensitive data from meters.

In this work, we focus solely on AMI: neighborhood-level smart grids including smart meters, utility management services, and the communications between them. However, there is nothing specific to AMI in the archetypal attack tree approach. The explored techniques are applicable to a broad range of products such as SCADA, medical devices, or automotive systems. We begin the exploration of this approach and its use in the next section.

# 2. METHODOLOGY

An attack tree is a structure for enumerating the kinds of attacks that achieve a particular adversarial goal [27]. It does this by recursively breaking down a goal into finer- and finer-grained subgoals and finally to a set of attacks that achieve the original goal. An example attack tree that formed the genesis of this work [24] is shown in Figure 1. The root specifies the end goal, committing energy fraud by forging the energy usage information reported to the utility. The internal nodes (those with parents and children) describe the different combinations of conditions that must be met to commit fraud. Finally, the leaves of the tree are the attacks necessary for energy fraud. The final attribute of the tree is the conjunctions (AND/OR) between each layer of child nodes. These specify whether all or just one of the child branches must be followed to reach the goal in the parent node.

What we notice about this example is that the attacks at the leaves of the tree are fairly general, and seem applicable to most smart metering systems. This suggests that this type of tree is a widely applicable tool. However, because it lacks details about any specific system, its usefulness is limited in finding concrete vulnerabilities. Because we are pen-testing multiple commercially available metering systems, we will want to further specify the details of each attack in this generic tree. Thus, as we learn about the individual systems, we extend this generic tree with vendor-specific attack strategies. These ideas can be refined into two types of attack trees: *archetypal* and *concrete*.

The process of grafting a concrete tree to an archetypal tree is shown in Figure 2. For a given adversarial goal, one may define an *archetypal tree* that enumerates strategies for reaching the goal against any system of a given architecture. In the case of the example above, the goal is forged energy demand and the architecture is smart metering. Each leaf of an archetypal tree is an *archetypal attack*. A concrete tree then refines an archetypal attack with respect to a specific vendor's system. The subgoals in the concrete tree sensitive to the security mechanisms present in the system, and thus define the exact conditions under which the root goal can be achieved. The leaves of the concrete tree are the *concrete attacks* which ultimately allow an adversarial goal to be achieved. For the purposes of our study, we use penetration testing to determine the feasibility of each concrete attack.

Our method is similar to that originally used for attack patterns [14, 10]. An attack pattern is a parameterized description of an attack, e.g. an injection attack, that is generic until its parameters are instantiated. Attack patterns may be described in terms of attack trees. When considering a particular attack against a particular *instance* of a system, e.g. a company's network, its parameters are instantiated with the specific details of that system. The concept of attack trees is based on that of fault trees, which were originally use to model the dependencies between potential faults in aviation and nuclear power systems [8, 32].

Attack trees by themselves are useful as a guide for penetration testing. However, once the knowledge of system interfaces has been exhausted and the concrete attacks are developed, we resort to standard pen-testing techniques such as reverse engineering [6], fuzz testing [28], and the construction of custom attack tools. For example, we later examine an energy fraud attack based on a meter spoof program written in Python.

Documented throughout, our methodology for directing penetration testing includes:

1. **Capture architectural description:** Elicit the features of a general architecture for target domain (see Section 3).
2. **Construct archetypal tree:** Given the architectural description, design a generic and comprehensive archetypal tree for each adversarial goal (see Section 4).
3. **Capture vendor-specific description:** Identify the structures and security mechanisms present the Systems Under Test (SUTs) that may thwart a given archetypal attack (see Section 5).
4. **Construct concrete trees:** Graft the vendor-specific goals to an archetypal goal to form concrete trees (see Section 6).
5. **Perform Penetration Testing:** Attempt to achieve the concrete goals by performing penetration testing on the SUT (see Section 7).

# 3. THE ADVANCED METERING INFRASTRUCTURE

AMI may be divided into utility-side management, smart electric meter deployments, and the networks that connect these two. This section describes these three along with AMI security concerns. At the edge of the AMI resides its main component: smart electric meters. A smart meter is a digital equivalent of a stand-alone electromechanical meter. Their most distinguishing characteristic is the use of two-way network communication with utilities. Smart meters have evolved from early Automated Meter Reading (AMR) systems [5] to allow for automatic updates of dynamic pricing information [26] and curtailment of individual loads when the grid is under stress [11]. Internal storage is used to keep time of day measurements for Time of Use (TOU) pricing schemes [18] and logs for both power outages [17] and potential intrusions, the latter of which is further explored in section 5.1.

## 3.1 Smart Meter Architectures

A smart meter is a networked embedded system equipped with a special apparatus for sensing electrical currents flowing through wires. In this section, we tease out the details of this definition, starting with the individual computing platform and finishing with the network. Unless otherwise specified, the features described in this section are present in the vast majority of commercial smart meters.

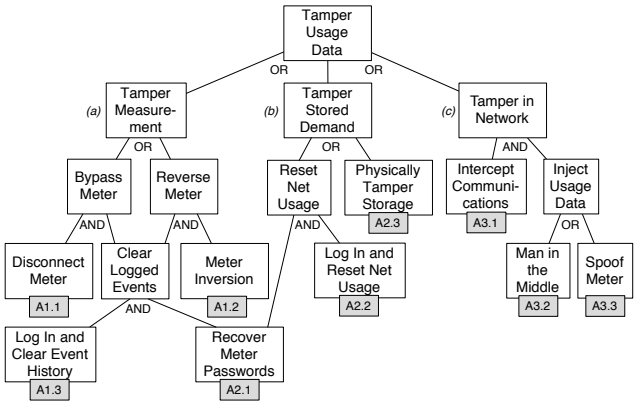


**Figure 1: Example energy fraud attack tree. The three subgoals beneath the root are labeled as $(a)$, $(b)$, and $(c)$ for reference purposes.**

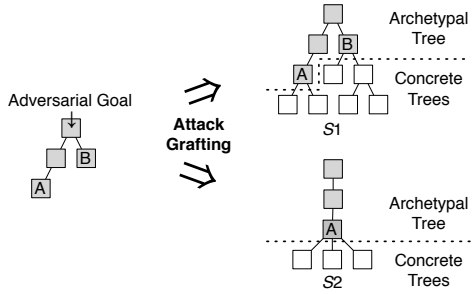


**Figure 2: Grafting concrete trees for two different systems ($S1$ and $S2$) onto an archetypal attack tree for a specific adversarial goal.**

Meters that are kept outside, such as those in the US, reside in protective socket enclosures, while those kept inside, which is common in the EU, often do not require a socket. The meter's internals are further protected by its cylindrical housing which consists of a base and a removable cover. To detect tampering by removal of the cover, a "flag" style aluminum tamper seal connects the cover to the base. This inexpensive seal consists of a stem which must be broken to remove the cover and a flag with a stamped identifier for the seal. As one might expect, there are no restrictions preventing the purchase of the seal with whatever flag marking is desired, making the removal of a seal for the purposes of physical tampering inconsequential.

The activities of a smart meter are coordinated by its Microcontroller Unit (MCU). The majority of work done by the MCU involves retrieving energy measurements from the low-level *meter engine* and storing them in flash memory for later transmission to the utility. Smart meter storage, however, is not used for electrical measurements alone. Like any general-purpose system, smart meters maintain logs of event histories and operating conditions. While the set of logged events varies between meter vendors, we cover the logs relevant to our security analysis later.

For flexibility of installation, smart meters within the same deployment can communicate over a number of different network mediums and topologies. Thus, meter firmware is designed to support a generic communication interface, leaving the specifics of a given network to a pluggable Network Interface Card (NIC). The meter exports a generic serial interface to communicate with the NIC, leaving the processing of specific network communication to the NIC.

If a meter is out of network communication with the utility and configurations or repairs are needed, it can be controlled locally through a standard infrared optical port located on its front panel. These ports are accessed via a small optical probe consisting of an LED and a photo-sensor at the range of less than one inch. While most meter vendors follow the physical layer standard for this port [4], the application layer is often proprietary. Typically, the optical ports transmit all data in the clear including passwords for user authentication. This includes the meter's administrator password.

One final component that deserves attention is the remote disconnect switch. If a utility wishes to disconnect a customer's power, it may do so remotely by transmitting a request to the meter to open the switch. The request is received by the digital portion of the meter, which issues the signal to the switch to break the circuit for the power flowing through the meter.

## 3.2 Meter Networks and Utility-Side Management

Given the sheer size of a utility's customer base, achieving networking connectivity with a meter at each individual home is a serious logistical challenge. Given the near impossibility of placing each individual meter on a public network, smart meters are designed to form their own LANs, each of which relies on a gateway device for communication between the LAN and public network. Some common choices of LAN and public network configurations are shown in Figure 3.

In the most common meter LANs, meters are connected in an adaptive wireless mesh network. Each meter in the mesh is a *repeater* that propagates data through the LAN to a *collector*. In some cases, the collector may itself be a meter. Power Line Communication (PLC) networks piggyback signalling over power distribution lines to form a star network topology that directly connects each meter in the LAN with the collector. The collector connects to the utility via a *backhaul* network such as the cellular or landline phone network, or the Internet.

On the utility end of the meter network resides a PC or server machine responsible for performing all regularly scheduled interactions with the meter. This machine runs a commodity OS, e.g. Microsoft Windows, a database server and the proprietary meter server software. If the utility server is compromised, the entire meter deployment is compromised.

## 3.3 AMI Security Concerns

Since smart meters have first come under scrutiny, concerns have been raised regarding their accuracy, reliability, security and privacy [23]. Academic and industrial pentesting efforts have found flaws in meter hardware [20], firmware [9] and network protocols [24]. Recently, Pacific Gas and Electric (PG&E) has experienced problems with measurement accuracy and meter network connectivity in their 5 million meter deployment, one of the largest in the US [15]. The

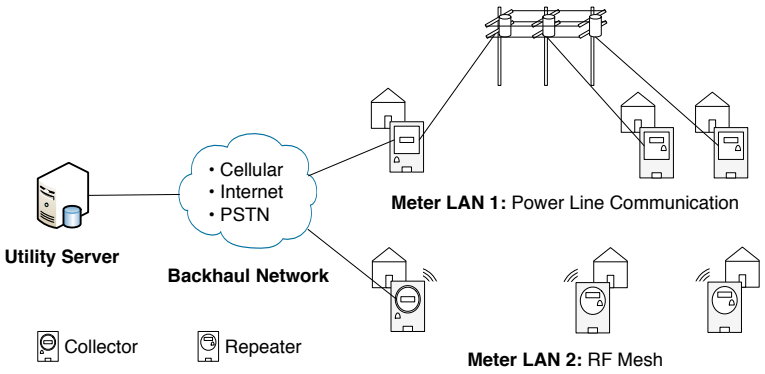


**Figure 3: Connectivity of meters to utilities given two configurations of meter LANs.**

addition of networks of such large numbers of devices to the uncontrolled Internet has been known to leave systems vulnerable to Denial of Service (DoS) attacks stemming from incompatibilities between their rigid proprietary designs and the Internet's open architecture [7, 31]. It will later be shown that this is the case for one of our pen-tested systems.

In addition to basic cyber security concerns, the advanced measurement capabilities of smart meters makes them a potential threat to privacy if used in an unrestricted manner. This is due to their ability to implement Non-Intrusive Load Monitoring (NILM), which can disaggregate the loads exerted by the individual appliances in a house from the net load recorded at the electric meter [13]. Hart posited NILM's use as a means of surveillance over activities that are normally considered within the sanctity of the home [12]. More recently, Lisovich et al. showed that the appliance information extracted by NILM is useful to recover some information about occupant behavior [21]. While this paper is limited to AMI related concerns, we mention that attacks on sensors in the grid's core distribution network have also been considered [22], along with the necessary conditions for such attacks to lead to large scale cascading failures [19].

# 4. ARCHETYPAL ATTACK TREES

Having reviewed the general architecture of smart metering systems, we may now construct archetypal trees that describe attacks in a broad sense that is applicable to any system within the architecture. An archetypal tree is an attack tree that is general enough to be applicable to all systems of a given architecture. As with a regular attack tree, the root of an archetypal tree is a single adversarial goal. This goal is repeatedly broken down into subgoals that describe the individual conditions that must exist to reach the root goal. Unlike a regular attack tree, the leaf nodes of the archetypal tree are not targeted at a specific system. Instead, the leaves constitute the points to which concrete trees are grafted. It is thus critical that they be selected to clearly define the boundary between broad architectural goals and vendor-specific goals. While this is somewhat of an art rather than a science, we have devised a set of criteria to aid us in differentiating between archetypal and concrete goals. If any of the following are true of a goal during the construction of an archetypal tree, then it becomes a leaf node, to which a concrete tree can be grafted.

1. *The goal targets a component whose implementation is vendor-specific.* An example of such a component is the meter LAN. While an archetypal tree can prescribe an attack on a meter LAN, the attack can not be specific to any particular LAN media.
2. *The goal may be hindered by the presence of a vendor-specific protection mechanism.* The addition of any subgoals for circumventing vendor-specific protection mechanisms is by definition not archetypal. Such details must be described in the concrete tree. An example of this can be seen in the following section on energy fraud (Section 4.1), where nothing general is known about the protection mechanisms present at the collector's link to the backhaul network.

If a subgoal does not meet these conditions, it is broken down. In the following sections, we provide justification for extending or terminating a given subgoal where instructive.

## 4.1 Energy Fraud

For our initial pen-testing efforts [24], we constructed an archetypal tree for energy fraud (shown in Figure 1). It is described here so that it may be instantiated later. We define energy fraud as any tampering with the metering infrastructure that leads to a customer not being billed for some energy consumed. (Note that in this particular archetypal tree, we do not consider using energy fraud to artificially inflate a victim's bill.) In AMI, fraud may be committed in the field by modifying the recorded energy usage before it is read by the utility. Known methods for fraud in electromechanical meters include interfering with the meter's sensors using magnets and rewinding usage gauges by inverting the meter in the socket (thereby reversing current flow through the meter).

Smart meters, present new opportunities for tampering with usage data. As shown in the first level of subgoals in the example tree, this can be done in three places ($a$) in the meter's low-level components, ($b$) the meter's long-term storage, and ($c$) in transmission to the utility. The archetypal attacks in this tree, as in the others, are labeled as $TX.Y$, where $T$ is a letter specific to the tree, $X$ is the index of the subtree below the root to which the attack belongs, and $Y$ is the index of the attack within that subtree. Starting with the physical attacks in subtree $a$, there are two means to interrupt a smart meter's physical measurement of usage. A1.1 simply requires that the meter is removed from the path of current flow, and A1.2 that it be reversed in

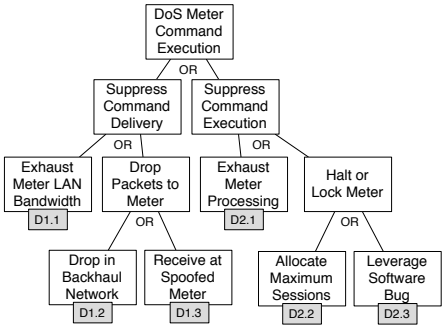


**Figure 4: Archetypal tree for Denial of Service.**

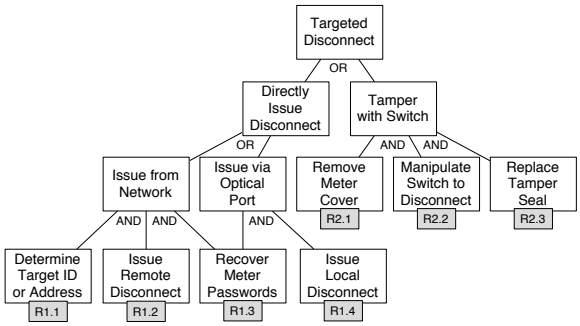


**Figure 5: Archetypal tree for targeted disconnect.**

its socket. As described in section 3.1, virtually all smart meters will log and report both of these events (power cycle and reverse energy flow respectively). Thus, in the archetypal level, we already recognize that the log messages will need to be cleared of these events. As a final note on physical attacks, because obtaining physical access to the meter is specific to a *particular installation*, we do not consider this prerequisite in either the archetypal or concrete trees. This does not matter for the case of fraud because it is assumed that the adversary already has access to her own meter.

Modifying logs and usage in meter storage is the goal of subtree *b*. This can be achieved in one of two ways. Either the meter's administrator password can be obtained and used to clear the log files: A2.1 AND A2.2, or the physical storage device may be tampered without interfering with the meter. As this is an archetypal tree, the implementation of the storage is left unmentioned.

The strategies for forging usage data on the wire are shown in subtree *c*. The interception of network communications is assumed to be necessary both for the purposes of understanding the meter's protocol stack, assuming it is nonstandard, and for interposing one's self in the communication path with the utility. In the archetypal tree, we ignore over which network (meter LAN or backhaul) the interception occurs, as well as any potential protection mechanisms. Along with A3.1, the adversary must either hijack a session between the meter and utility (A3.2) or impersonate a meter for the entire session (A3.3).

## 4.2 Denial of Service

This section considers DoS attacks that prevent meters from acting on commands such as usage queries, firmware upgrades, and remote disconnects. This is a realistic adversary goal. For example, if the retrieval of meter log files can be prevented for a sufficient period of time, a suspicious event such as a meter power cycle can be erased when the logs roll over with benign events.

The archetypal tree for meter DoS against meter command execution is shown in Figure 4. The adversary has two choices for a general strategy, either prevent the command from reaching the meter, or prevent its execution on the meter. The former can be achieved either through network resource exhaustion, or by tampering with the routing of packets away from the meter. As the LAN media is system specific, we do not break this subgoal down any further in the archetypal tree. A potentially more practical strategy is to drop traffic destined for the meter. This may either be done at a link or routing layer (D1.2) or at the transmission layer (D1.3). The latter seems like the more reasonable method, as dropping a packet at an intermediate hop will result in a retransmission by a higher layer.

The second strategy for command DoS prevents the meter from executing a command once it is received. An extremely simplistic method for doing this is to exhaust the meter's input processing capability (D2.1). This could be done either from the backhaul network or meter LAN. While effective, this type of attack is not covert, and cannot guarantee the command will fail. A more failsafe approach would be to put the meter into an unresponsive state. This may be done through interactions that exhaust a particular system resource, e.g. allocating and maintaining the maximum allowed number of open connections (D2.2), or by leveraging a firmware bug causing a system hang (D2.3).

## 4.3 Targeted Disconnect of Electrical Service

Most meter vendors include remote disconnect functionality in their meters. The ability to disconnect a target's power can cause at best, inconvenience and in worse scenarios, financial or physical harm depending on the setting. As described earlier, remote disconnect systems consist of a physical switch that breaks the current flowing to the house, and a set of remote commands to operate this switch. The archetypal tree for this attack is shown in Figure 5.

The ideal case for an adversary would be to issue the disconnect command remotely. Doing this requires at least that the ID be known for the target device (R1.1), and that its administrator password has been recovered (R1.3). Notice that this is the second archetypal tree with a leaf node requiring meter passwords to be recovered. This illustrates a secondary usefulness of attack trees: they act as a reference for quickly mapping security flaws to the adversarial goals they enable.

We reason that the disconnect functionality will be accessible through the optical ports on most systems because optical port functionality needs to contain at least the network functionality to allow the meter to function in the event that it is not network accessible, e.g. the meter's network card is malfunctioning. This is the basis of archetypal attacks R1.3 and R1.4.

Finally, physical access to a meter may also be useful for manipulating the disconnect switch, be it by mechanical or electrical means (R2.2). From experience, we have found that virtually all smart meters use the same tamper seal [1]. We have contacted the manufacturer of these seals and con-

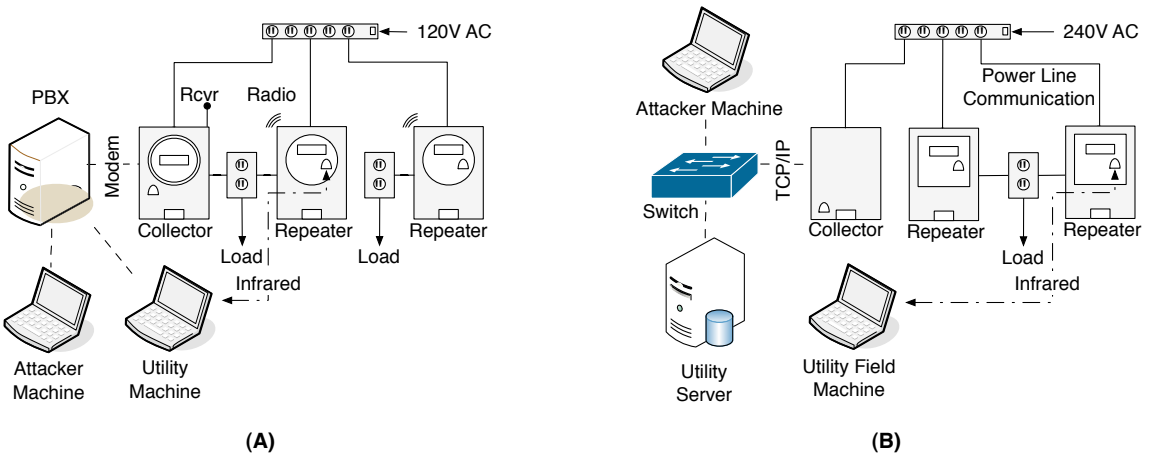


**Figure 6: The two SUTs used in our experiments. In $S1$ (A), the collector also functions as a meter, and relays data from a wireless mesh LAN to a telephone network backhaul. $S2$ (B) uses a dedicated device as a collector to relay data between a PLC network and an Internet connection to the utility.**

firmed that there are no limitations on the text which we could have embossed on the flag.

## 5. SYSTEMS UNDER TEST

This section details the two Systems Under Test (SUTs) that have been the subject of our penetration testing[1]. We will denote the two systems as $S1$ and $S2$. Besides the meters themselves, this section covers the additional components needed to run utility-end software and to network meters with the utilities. In describing the two systems, we will refer to the *utility machine* or *utility server* to mean a Microsoft Windows-based PC or laptop computer running software for meter management. We found that Windows by far the most common choice of utility-end operating system across vendors. The *attacker machine* is used to represent our machine used for various pen-testing purposes. In practice, this could be any machine within network reachability of a meter that is controlled by an adversary.

The general environment for both systems is identical. Both SUTs consist of several repeaters and a single collector, the main difference being that in $S2$, the collector does not function as a meter itself. We constructed sockets to allow the meters in our lab to function using wall socket power. The meters in $S1$ are able to run on 120V AC at 60 Hz, while the meters in $S2$ require a 240V step up transformer. A simple load was exerted by a small synchronous motor and measured to check the proper installation of each meter.

### 5.1 S1 Specifics

An overview of $S1$ is given in Figure 6.A. In $S1$, utilities communicate with meters via Public Switched Telephone Service. For obvious security reasons, we were unable to directly connect our collector to the telephone network. Instead, an Asterisk [29] based private branch exchange (PBX) on an x86 Linux machine provided call routing between the collector and utility machine. The PBX routes calls according to a table called the *dial plan*. The attacker machine sits on the PBX along with the meter and utility machine. Calls to the meter can be routed to the attacker machine by modifying the PBX dial plan. The ability to perform such rerouting using a commodity system was instrumental in our instantiation of the energy fraud attack for $S1$.

For all communication, the utility machine initiates communication with collector meters, with the exception of alarm conditions such as outage management or potential intrusions, in which case the meter preemptively contacts the utility. We augmented the utility machine with a modem monitor for analyzing the telephone protocol. What we quickly found is that it largely conforms to the ANSI C12.21 standard for telephone modem communication with meters. After this, the monitor was only needed to understand the occasional deviations from the standard.

The PSTN backhaul link at the collector is guarded by an "intrusion detection" mechanism. The purpose of this mechanism is to prevent both active and passive attacks from telephony devices connected on the same link as the collector, i.e. via a line splitter. The intrusion detection mechanism will immediately terminate a call from the utility if another device on the line goes off the hook. When a device goes off the hook, it receives a dial tone and voltage via an onboard component called the Foreign Exchange Office (FXO). All endpoint devices in a telephone network use an FXO. The dial tone and voltage are supplied from the other end of the line by the Foreign Exchange Service (FXS), usually implemented by the phone company. Because the meter can detect when another device is receiving a voltage and dial tone, it can terminate its current call.

The main operation of concern in $S1$ is the diagnostic protocol between the meter and utility. This protocol can perform many functions from a simple meter reading, to a full check of every parameter set in the meter. For the purposes of energy fraud we are mainly concerned with how this protocol performs energy usage readings. The utility initiates a diagnostic by calling a collector, resulting in the collector responding with an identification message. An authentication round is then carried out according to the default scheme specified by ANSI C12.21 (ANSI X3.92-198) [3]. If authentication is successful, the utility will probe the meter for

[1]We do not reveal vendor identities here, as we are already in contact with them, and both SUTs are already deployed in the US and Europe.

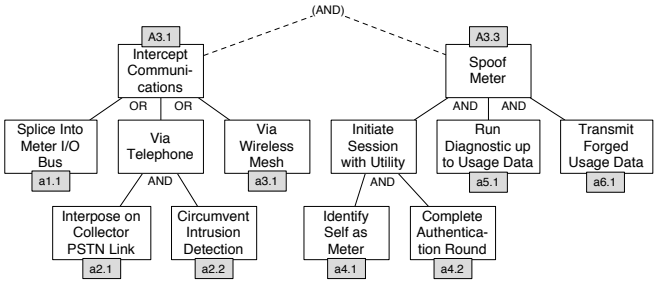


**Figure 7: The concrete trees for energy fraud in $S1$.**

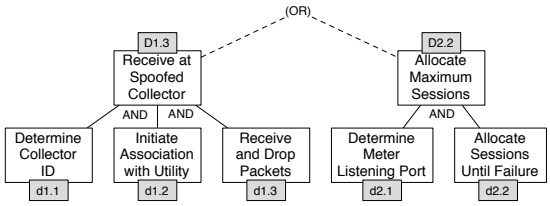


**Figure 8: The concrete trees for DOS in $S2$.**

some variable number of parameters, after which the current net usages are read. This is the point in the protocol where a usage forgery must occur. The remainder of the protocol consists of potentially more parameter queries, and finally a goodbye message. The meter LAN, a wireless mesh operating in the 900 MHz band, is currently under evaluation.

## 5.2 S2 Specifics

Our testbed for $S2$ is shown in Figure 6.B. The main differences from $S1$ are the backhaul and meter LAN protocols, and the collector, which does not function as a meter in $S2$. Upon initial inspection, one notices that $S2$ is more accessible to remote attacks due to the use of an Internet-based backhaul. This fact becomes useful when instantiating a concrete tree for DoS against meter command execution. The meter LAN uses a proprietary protocol that requires special equipment to analyze.

Though the application layer protocol between the utility and collector is proprietary, two thing are clear from initial inspection. First, an initial association between the two is started by the collector, and each subsequent command execution is started by the utility. This suggests that both directions should be considered when designing a concrete DoS attack. Second, in the initial association, the collector transmits its unique ID number and associated network address in the clear to the utility. Thus, knowing this ID for a target collector may be useful in a DoS attack.

# 6. CONCRETE ATTACK TREES

Concrete attack trees function as a guide for penetration testing a specific system. As with the archetypal trees, we use basic guidelines to determine when a concrete tree is specific enough. Any details not elaborated in the concrete tree must either already be known about the system, or must be discovered during pen-testing. In constructing the concrete trees for fraud, DoS, and targeted disconnect, we use the following two rules:

1. *A goal should be a leaf if it is achievable completely by known means in the system.* This is the simplest case as no additional pen-testing is required. Several leaves in the concrete DoS tree are of this type.
2. *A goal should be a leaf if no vulnerability is yet known that would allow it to be executed.* At this point, determining the existence of a vulnerability enabling the goal becomes the job of penetration testing.

We instantiate concrete trees for the three adversarial goals for $S1$ and $S2$ below. The root of each concrete tree shares a reference number with a leaf in one or more archetypal attack trees to which it may be grafted. We instantiate fraud and targeted disconnect for $S1$, and DoS for $S2$[2].

## 6.1 Energy Fraud in S1

The archetypal attack tree for energy fraud presented three broad strategies: tampering with the measurement process, tampering with the recorded usage in meter storage, and tampering with the usage data in transmission. For our first attempt to implement a fraud attack in $S1$, we chose the third strategy because of its relatively low invasiveness and our understanding of the backhaul network operation. This strategy terminated in three archetypal attacks: a mandatory requirement of being interposed on the backhaul link (A3.1), and the option of either performing a man in the middle attack (A3.2) or meter spoofing (A3.3). After evaluating the ANSI C12.21 specification via trace of $S1$'s telephony-based diagnostic protocol, we determined that meter spoofing was more straightforward. Thus, to complete the goal of fraud in $S1$, we must instantiate and execute concrete trees for archetypal attacks A1.1 and A1.3. Both concrete trees are shown in Figure 7.

Archetypal attack A3.1 requires that the adversary be interposed somewhere on the path between the meter's networking interface card (NIC) and the utility. In one extreme end, this may be achieved by directly tampering with the communications bus on which the NIC resides (a1.1). Two more likely places are the mesh network (a3.1), and the telephone backhaul (a2.1). For the latter, the additional prerequisite of bypassing the "intrusion detection" mechanism is necessary (a2.2).

The second archetypal attack for energy fraud requires meter spoofing. This calls for three steps to successfully deliver forged usage data as part of $S1$'s diagnostic protocol. First, the spoofing device must initiate a new diagnostic session with the utility. This will require first identifying itself as the expected meter (a4.1), and second, completing the authentication round (a4.2). Once the session is established, the spoofing device must answer all diagnostic queries up to the forged demand (a5.1), and finally, insert the forged demand value (a6.1). The remainder of work to realize these attacks is achieved by pen-testing as described in section 7.

## 6.2 Denial of Service in S2

Unlike the two concrete trees for energy fraud, the root nodes of the two for DoS are combined by disjunction in the archetypal tree. Thus, fulfilling the requirements of either

[2]While there are a large number of attempted attacks, we find the ones described here to be the most instructive.

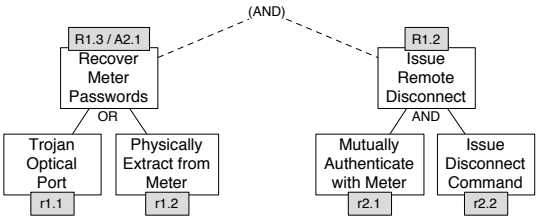


**Figure 9: The concrete trees for targeted disconnect $S1$.**

tree is sufficient for achieving denial of command execution. Recall that there are two options because communication in $S2$ may be initiated by both the collector and the utility at different points in time. The first tree (D1.3) requires another device to spoof the collector node in order to receive any commands destined for meters and drop them en route. This requires first the necessary reconnaissance to determine the collectors network ID (d1.1), and to establish a new session with the utility using that ID (d1.2). Finally, the spoofed collector can receive and drop commands from the utility (d1.3). All three of these are leaves in the concrete tree because they are achievable using known actions within the system.

The other option for DoS against utility command execution is to allocate a maximum number of sessions in the meter (D2.2). First, it must be determined on which port the meter listens for commands (d2.1). If this is possible, an attempt may be made to open multiple sessions on this port in an attempt to exhaust either memory or OS resources in the meter (d2.2). Both concrete attacks are leaf nodes because pen-testing of $S2$ is needed to determine how they may be executed in practice.

## 6.3 Targeted Disconnect in S1

The final concrete attack tree analyzed here is for the disruption of electrical service. As an adversary would ideally want to execute this attack remotely, we chose archetypal attacks R1.1 - R1.3 for instantiation. In $S1$, the meter ID is printed on the front of each meter, making R1.1 achievable by visual inspection. The concrete trees for R1.2 and R1.3 are shown in Figure 9.

Two strategies are feasible for meter password recovery in $S1$ (R1.3). If the optical port can be physically monitored, then the password can be obtained upon the next visit by the utility (r1.1). Alternatively, if the contents of meter storage can be extracted, the password may be recoverable, though potentially only in a hashed format (r1.3). As both of these are physical attacks, they may only be used to recover a password from a single meter. This would normally be a limiting factor in the impact of an attack against $S1$, but we observe that its architecture encourages utilities to use the same password for a large number of meters. In the administrative utility-end software, a single password set (consisting of a read-only and administrative user) is chosen for a template program that is pushed to the meters at configuration time. This makes it very tedious to create a different program template for each meter. A brute force guessing attack is not considered, as the maximum length of a password in $S1$ is well over ten bytes. The final archetypal attack needed is the issuance of the command to the target meter. This requires that the known password be used in the mutual authentication round (the same as that used in $S1$'s diagnostic protocol) (r2.1). Once authenticated, the command can be issued (r2.2).

# 7. RESULTS

We now turn to the results of the penetration testing to achieve each goal as summarized in Table 1.

## 7.1 Energy Fraud by Forged Usage Data

The energy fraud attack in $S1$ works as follows. First, an adversarial device is interposed on the PSTN link from a collector (a2.1) so as not to trigger the intrusion detection mechanism (a2.2). This was achieved by interposing our PBX on the line. Recall that the purpose of the intrusion detection feature is to protect meter communication in situations where the link to the PSTN is shared with that already present in a house. The PBX is used to route incoming calls to the meter to a laptop computer that impersonates the meter using a Python program we wrote. This is sufficient for circumventing the intrusion detection mechanism for two reasons. First, routing a call to the laptop need not involve the meter at all. Second, if the PBX is used for the purposes of eavesdropping on communication between the meter and utility, it cannot be detected by the intrusion detection mechanism that can only sense other FXOs on the line (as described in section 5.1). Thus two requirements of A3.1 are satisfied.

Once the adversarial laptop has been contacted by the utility, it must identify itself as the target meter (a4.1) and complete the authentication round (a4.2). This was possible without knowing the meter's password, which is used to derive the key for the authentication protocol. Spoofing meter identification only required using the ID which was printed on the meter's nameplate. Completing the authentication round without knowing the password required one observation about the protocol: the meter generates the nonce used for mutual authentication, but nonces are not tracked by the utility's server. Thus, a replayed nonce is sufficient for replaying the remainder of the authentication protocol. What was not initially obvious was that the meter places the nonce in a special field as part of the identification round. Thus, replaying both the identification and authentication rounds of ANSI C12.21 is sufficient for spoofing the meter during a diagnostic.

The remaining protocol up to forged demand insertion may also be replayed in this manner, satisfying (a5.1). The final task towards energy fraud is inserting a forged net usage value into the diagnostic. This requires adding two additional pieces of information along with the numerical usage value. First, a one byte checksum of the value is placed in the application-layer header, and second a CRC is placed in the MAC layer header, again as specified in ANSI C12.21.

**Table 1: Summary of concrete attacks and discovered vulnerabilities for each adversarial goal.**

| Ref. | Description | Enabling Feature or Vulnerability |
|---|---|---|
| | *Energy Fraud in $S1$* | |
| a2.1 | Interpose between utility and collector | Telephone line may be accessible. |
| a2.2 | Defeat modem intrusion detection | The mechanism cannot detect an FXS. |
| a4.1 | Identify self as meter | A meter's ID is printed on its faceplate. |
| a4.2 | Complete authentication round | Lack of nonce-tracking allows replayed authentication. |
| a5.1 | Run diagnostic up to usage data | Protocol is standardized. |
| a6.1 | Transmit forged usage data | Usage data is not integrity protected. |
| | *Denial of Service in $S2$* | |
| d1.1 | Determine collector ID | The ID is transmitted in the clear. |
| d1.2 | Initiate association with utility | Initialization uses a simple `init` message. |
| d1.3 | Receive and drop packets | The utility uses the IP address of the initiator of the most recent association. |
| d2.1 | Determine meter listening port | The collector is responsive to port scanning. |
| d2.2 | Allocate sessions until failure | The collector does not handle many sessions robustly. |
| | *Targeted Disconnect in $S1$* | |
| r1.2 | Physically extract passwords | Passwords are stored in the clear in EEPROM storage. |
| r2.1 | Mutually authenticate with meter | The encryption key is derived from passwords. |
| r2.2 | Issue disconnect command | Administrative software is commercially available. |

## 7.2 Denial of Service Against Command Execution

Two concrete attack trees were previously introduced for Denial of Service against the execution of utility commands by meters in $S2$. The first assumed that an association could be formed between the utility and a device impersonating a collector (d1.1,d1.2). At this point, the fake collector could simply drop all commands issued by the utility (d1.3). The initial association with the utility is initiated by an `init` sent by the collector. This message, which is transmitted in the clear, contains the unique serial number used to identify the collector. The utility assumes that the source IP address of the `init` message is the collector. Any device may submit an `init` message to the utility, but will not be able to establish a secure channel without knowing the collector's symmetric key. This does not prevent the DoS attack however, as receiving the `init` message causes the utility to drop its previous association with the real collector. After this, the collector will only attempt to create a new association if it is rebooted or if some alarm condition occurs such as a power outage or potential physical tampering. A subsequent second forged `init` would suffice to immediately break this association.

The other concrete attack tree in this category is based on the idea that the collector has a maximum number of sessions which can be reached (D2.2). In practice, finding the port on which a collector listens for utility requests (d2.1) is done using the `nmap` [2] utility to perform a port scan of the collector. What we found was that while attempting to open many concurrent TCP connections on the collector's listening port, the collector would become unresponsive after fewer than ten such connections. If continual attempts at establishing new connections were made at the rate of once per ten seconds, the collector remains unresponsive, and the utility-end server is unable to complete any commands on that collector, thus satisfying d2.2.

Under the category of DoS, we do have one result that was found completely independently of the methodology presented in this paper. The use of a software fuzz tester [16] found that the collector was vulnerable to crashing while processing malformed packets. While we had not planned on systematically exploring methods for leveraging software bugs (D2.3), the use of fuzz testing would make a viable addition to the archetypal tree.

## 7.3 Targeted Disconnect

The final result we explore is the application of concrete trees R1.2 and R1.3 to disrupting electric service at a target meter by subverting its remote disconnect feature. We were unable to verify the efficacy of this attack due to fact that our $S1$ meters do not include the optional physical disconnect switch. However, we reason by inspection that the attack is possible. The first step needed to issue the remote disconnect command is password recovery (R1.3). After some experimentation, we found that concrete attack (r1.2) is possible. By desoldering a small SPI-based EEPROM memory chip from the $S1$ collector's radio card, we were able to extract the plaintext password. While this is a potentially dangerous operation, given that the same password may be used throughout a deployment, the payoff is high. Upon discovering that the meter passwords may be extracted from memory, a check of the archetypal trees reveals that A2.1 from Figure 1 is also satisfied, enabling several alternate strategies for energy fraud. This demonstrates the usefulness of archetypal attack trees in mapping newly discovered vulnerabilities to adversarial goals.

Once the password is recovered, it must be used to perform the default C12.21 authentication function with the target meter (r2.1). This authentication is a keyed hash based on the DES cipher, and thus requires a DES key. An internet search revealed that one distributor of $S1$ had placed the manual for the utility-end software in a publicly accessible directory. The manual revealed the fact that the first eight bytes of the password are used to derive a DES key. This is done using an unknown obfuscation method. The easiest procedure to use the recovered passwords for authenticating to the target meter would be to obtain the utility-end software, which can be purchased from third party distributors, and provide it the password to issue the discon-

nect command (r2.2). Otherwise, a degree of reconnaissance and reverse engineering will be necessary to determine the obfuscation method.

# 8. CONCLUSIONS

In this paper, we have investigated a technique for evaluating the security of the myriad of devices being deployed into the AMI. We have shown that we can leverage focused penetration efforts in one vendor to others, and explored where such evaluations must focus solely on the unique artifacts of a system under test. In so doing, this work has sought not only to streamline security analysis, but also to ensure greater and more consistent coverage of potential attacker goals and methods.

Yet there is much work left to be done. Government agencies such as the NIST and the Federal Energy Regulation Commission (FERC) continue to provide the essential guidelines for the design and maintenance of AMI *security infrastructure*. Complementary efforts at codifying penetration testing of AMI such as the one documented in this paper are essential to the future reliability of electric power grids. In the future, we will expand the base of attacker goals and associated trees, as well as extend this work to other vendor devices. It is through these collected efforts that we hope to garner a broad view of the security issues in AMI, and ultimate positively influence the safety of smart grid.

# 9. REFERENCES


[1] B.T. Aluminum Tamper Seal. `http://www.brooksutility.com/catalog/product-detail.asp?ID=302`.

[2] Nmap Reference Guide. `http://nmap.org/book/man.html`.

[3] American National Standards Institute. ANSIX3.92-198 Data Encryption Algorithm, 1981.

[4] American National Standards Institute. C12.18 Protocol Specification for ANSI Type 2 Optical Port, 2006.

[5] A. Brothman, R. D. Reiser, N. L. Kahn, F. S. Ritenhouse, and R. A. Wells. Automatic Remote Reading of Residential Meters. *IEEE Transactions on Communication Technology*, 13(2):219 – 232, 1965.

[6] E. Eilam. *Reversing: Secrets of Reverse Engineering*. Wiley, 2005.

[7] W. Enck, P. Traynor, P. Mcdaniel, and T. L. Porta. Exploiting Open Functionality in SMS-capable Cellular Networks. In *Proceedings of the 12th ACM Conference on Computer and Communication Security (CCS)*, pages 393–404. ACM Press, 2005.

[8] C. A. Ericson, II. Fault Tree Analysis — A History. In *Proceedings of the 17th International System Saftey Conference*, 1999.

[9] K. Fehrenbacher. Smart Meter Worm Could Spread Like A Virus. `http://earth2tech.com/2009/07/31/smart-meter-worm-could-spread-like-a-virus/`.

[10] M. Gegick and L. Williams. Matching attack patterns to security vulnerabilities in software-intensive system designs. In *SESS '05: Proceedings of the 2005 workshop on Software engineering for secure systems—building trustworthy applications*, pages 1–7, New York, NY, USA, 2005. ACM.

[11] M. Goldberg. Measure Twice, Cut Once. *IEEE Power and Energy Magazine*, May/June 2010.

[12] G. W. Hart. Residential Energy Monitoring and Computerized Surveillance via Utility Power Flows. *IEEE Technology and Society Magazine*, June 1989.

[13] G. W. Hart. Nonintrusive Appliance Load Monitoring. *Proceedings of the IEEE*, 2004.

[14] G. Hoglund and G. McGraw. *Exploiting Software: How to Break Code*. Addison Wesley, 2004.

[15] D. Hull. PG&E details technical problems with SmartMeters. `http://www.siliconvalley.com/news/ci_14963541`, April 2010.

[16] Infigo.hr. Infigo FTPStress Fuzzer. `http://www.infigo.hr/en/in_focus/tools`.

[17] R. Kelley and R. D. Pate. Mesh Networks and Outage Management. White Paper, September 2008.

[18] C. S. King. The Economics of Real-Time and Time-of-Use Pricing For Residential Consumers. Technical report, American Energy Institute, 2001.

[19] R. Kinney, P. Crucitti, R. Albert, and V. Latora. Modeling cascading failures in the North American power grid. *The European Physical Journal B - Condensed Matter and Complex Systems*, 46(1):101–107, July 2005.

[20] N. Lewson. Smart meter crypto flaw worse than thought. `http://rdist.root.org/2010/01/11/smart-meter-crypto-flaw-worse-than-thought`.

[21] M. A. Lisovich, D. K. Mulligan, and S. B. Wicker. Inferring Personal Information from Demand-Response Systems. *IEEE Security and Privacy*, 8:11–20, 2010.

[22] Y. Liu, P. Ning, and M. K. Reiter. False Data Injection Attacks against State Estimation in Electric Power Grids. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, November 2009.

[23] P. McDaniel and S. McLaughlin. Security and Privacy Challenges in the Smart Grid. *IEEE Security & Privacy Magazine*, May/June 2009.

[24] S. McLaughlin, D. Podkuiko, and P. McDaniel. Energy Theft in the Advanced Metering Infrastructure. In *Proceedings of the 4th International Workshop on Critical Information Infrastructure Security*, 2009.

[25] R. Meritt. Stimulus: DoE readies $4.3 billion for smart grid. *EE Times*, February 2009.

[26] A. H. Rosenfeld, D. A. Bulleit, and R. A. Peddie. Smart Meters and Spot Pricing: Experiments and Potential. *IEEE Technology and Society Magazine*, March 1986.

[27] B. Schneier. Attack Trees. *Dr. Dobb's Journal*, December 1999.

[28] A. Takanen, J. DeMott, and C. Miller. *Fuzzing for Software Security Testing and Quality Assurance*. Artech House Publishers, 2008.

[29] The Asterisk Project. Asterisk open source pbx. `http://www.asterisk.org`.

[30] The Smart Grid Interoperability Panel – Cyber Security Working Group. Smart grid cyber security strategy and requirements draft nistir 7628, February 2010.

[31] P. Traynor, M. Lin, M. Ongtang, V. Rao, T. Jaeger, P. McDaniel, and T. La Porta. On Cellular Botnets: Measuring the Impact of Malicious Devices on a Cellular Network Core. In *Proceedings of the 16th ACM Cnference on Computer and Communications Security (CCS)*, pages 223–234, New York, NY, USA, November 2009. ACM.

[32] W. Vesely, F. Goldberg, N. Roberts, and D. Haasl. *Fault Tree Handbook*. U.S. Nuclear Regulator Commission, 1981.

[33] K. Zetter. Security Pros Question Deployment of Smart Meters. *Threat Level: Privacy, Crime and Security Online*, March 2010.