Structured Security Testing in the Smartgrid

Patrick McDaniel and Stephan McLaughlin
Department of Computer Science and Engineering
Pennsylvania State University
University Park, PA 16802
Email: {mcdaniel,smclaugh}@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 often outpace security efforts: new devices from a dizzying array of vendors are being introduced into grids with limited 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. 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. We provide a detailed example of one such attack as tested using our developed methodology.**

I. 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 [1]. 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.

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 limited understanding of the security problems they represent. Prudence demands 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 [2] are a step in the right direction, but only identify affirmative steps for secure systems. Beginning in the next section, we develop a systematic penetration testing methodology for neighborhood-level AMI systems and demonstrate is application on an example attack.[1]

[1]A thorough treatment of this work can be found in in [3]. The reference includes a number of examples on diverse systems under test and further details on attack tree methodology.

II. Approach

An attack tree is a structure for enumerating the kinds of attacks that achieve a particular adversarial goal [4]. 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 [5] 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. 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. This method similar to that originally used for attack patterns [6], [7] which itself was developed from fault analysis techniques in aviation and nuclear power systems [8], [9].

Attack trees by themselves are useful as a guide for penetra-

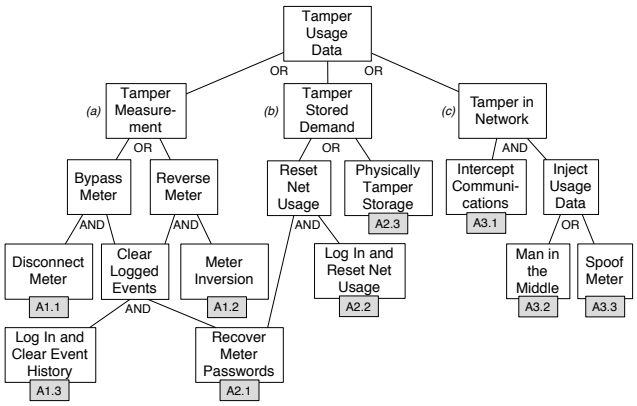

Fig. 1. Example energy fraud attack tree. The three subgoals beneath the root are labeled as (a), (b), and (c) for reference purposes.

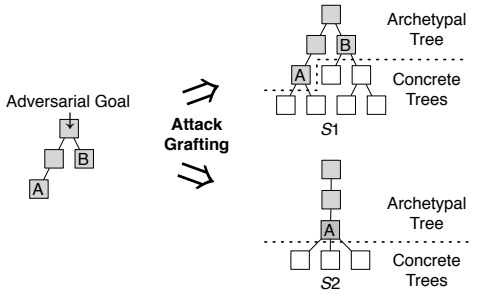

Fig. 2. Grafting concrete trees for two different systems ($S1$ and $S2$) onto an archetypal attack tree for a specific adversarial goal.

tion 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 [10], fuzz testing [11], and the construction of custom attack tools. The testing effort therefore includes:

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

We substantiate this process by describing the methodology and applying it to the testing of a specific adversarial goal (denial of service) on a real world SUT.

A. AMI Security Concerns

Since smart meters have first come under scrutiny, concerns have been raised regarding their accuracy, reliability, security and privacy [12]. Academic and industrial pen-testing efforts have found flaws in smart meter hardware [13], firmware [14] and network protocols [5]. 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 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 [16], [17]. 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 [18]. Hart posited NILM's use as a means of surveillance over activities that are normally considered within the sanctity of the home [19]. More recently, Lisovich et al. showed that the appliance information extracted by NILM is useful to recover some information about occupant behavior [20]. 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 [21], along with the necessary conditions for such attacks to lead to large scale cascading failures [22].

III. Archetypal Attack Trees

We now construct archetypal trees that describe attacks in a way that is applicable to any system. 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.

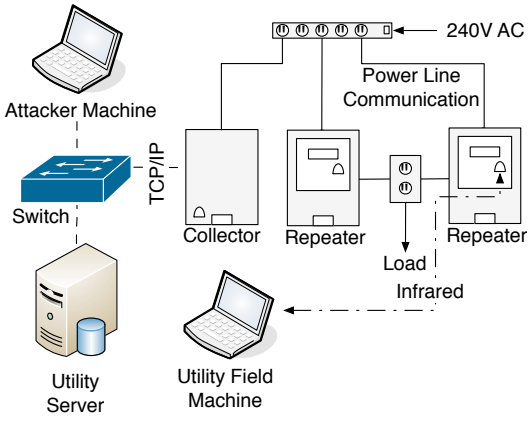

Fig. 3. The concrete trees for targeted disconnect $S1$.

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.

If a subgoal does not meet these conditions, it is broken down further as elements of the archetypal tree.

IV. 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.

V. Example Attack Testing

The following provides a brief description of a pentration testing effort on a specific attack vector: *denial of service* on the meter. We begin by describing the specific system under test that is used to test against.

A. Example System Under Test (SUT)

Figure 3 dipicts our laboratory testbed system under test. 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 SUT environment consists of several repeaters and a single collector. In the test environment, the collector does not function as a meter but is a stand-alone device. We constructed sockets to allow the meters in our lab to function using wall socket power. The meters in the SUT 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. The the backhaul and meters use PLC-based LAN protocols. Upon initial inspection, one notices that the SUT is 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.

B. Pentration Testing Trees

This section considers *denial of service* (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

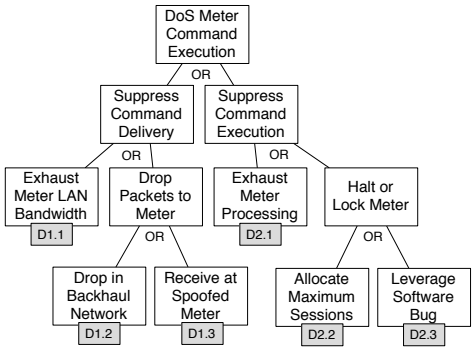

Fig. 4. Archetypal tree for Denial of Service.

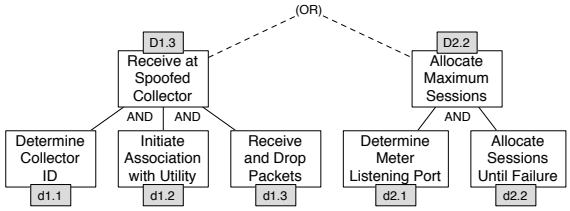

Fig. 5. The concrete trees for DOS in $S'2$.

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).

The concrete tree is combined by disjunction in the archetypal tree. Thus, fulfilling the requirements of either tree is sufficient for achieving denial of command execution. Recall that there are two options because communication in the SUT 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).

This example demonstrates how one can leverage the high-leveling modeling activities of the archetypal tress to perform *vertical penetration testing*. In this way, one can instantiate each tree for different vendors and reuse apparatus and effort across a multitude of vendors. In this way, the community will be able to scale of testing efforts to match aggressive and diverse deployments.

REFERENCES

[1] C. S. King, "The Economics of Real-Time and Time-of-Use Pricing For Residential Consumers," American Energy Institute, Tech. Rep., 2001.

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

[3] S. McLaughlin, D. Podkuiko, A. Delozier, S. Miadzvezhanka, and P. McDaniel, "Multi-vendor Penetration Testing in the Advanced Metering Infrastructure," in *Proceedings of the 26th Annual Computer Security Applications Conference (ACSAC)*, Austin, TX, December 2010.

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

[5] 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.

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

[7] 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*. New York, NY, USA: ACM, 2005, pp. 1–7.

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

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

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

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

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

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

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

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

[16] 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)*. ACM Press, 2005, pp. 393–404.

[17] 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)*. New York, NY, USA: ACM, November 2009, pp. 223–234.

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

[19] ——, "Residential Energy Monitoring and Computerized Surveillance via Utility Power Flows," *IEEE Technology and Society Magazine*, June 1989.

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

[21] 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.

[22] 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*, vol. 46, no. 1, pp. 101–107, July 2005. [Online]. Available: http://dx.doi.org/10.1140/epjb/e2005-00237-9