Submitted for publication. Author Copy - do not redistribute.

Smart Grid Protocol Testing Through Cyber-Physical Testbeds

Tim Yardley, Robin Berthier, David Nicol, and William H. Sanders
Information Trust Institute, Coordinated Science Laboratory, and Electrical and Computer Engineering Department
University of Illinois at Urbana-Champaign
{yardley,rgb,dmnicol,whs}@illinois.edu

***Abstract*—Sound cyber security testing is a critical challenge, in particular for large and complex systems such as the smart grid. In this paper, we explore the need for, and specific issues involved in, security testing for smart grid components and standards and how testbeds play a critical role in that environment. We present three main problems; the need for a methodology to define the appropriate tests, the need for a means of comparing and measuring the security of the system under scrutiny, and the current lack of tools and instrumentation with which to carry out those tests. We present work addressing those problem areas through approaches in methodology, quantification of security, formal methods, and tool creation. We illustrate our approach through a case study showing applications of these techniques to the Advanced Metering Infrastructure (AMI) protocol space and discuss advances in cyber-physical testbed experimentation that ease this testing at scale.**

I. Introduction

As evidenced by recent news, cyber-based threats to critical infrastructure are real and increasing in frequency. From major game-changers like the Stuxnet, Flame, and Duqu worms, to less dramatic items like default derivable passwords and other factory default weaknesses in commonly deployed critical infrastructure hardware, it is obvious that security problems exist. Important efforts like the NIST IR 7628 report [14] focus on providing recommendations for security and laying the groundwork that helps pave the way towards interoperability and a more secure smart grid. This groundwork is focused on standards, guidelines, and recommendations.

The NIST CSWG design principles subgroup spent a considerable amount of time reviewing specifications for many of the critical smart grid protocols, identifying security and interoperability issues to be addressed in future revisions. That review process will have significant value, as long as the recommendations for corrective actions are carried out by the standards bodies receiving those results. However, implementations of these security-reviewed specifications mostly have not yet been implemented or tested. The testing effort alone is challenging due to the general lack of a defined methodology and a prescribed way to quantify security combined with the constantly evolving threat landscape. To close that gap, applied research and methods are needed to improve security quantification and rigorous security assessment, not only for single components but also for complex systems in which heterogeneous components constantly interact.

Further, while review of system components is critical, one must also review the system as a whole in its deployment environment. Evaluating a system without consideration of the deployment environment, or the potential changes in that environment, will likely miss entire classes of configuration and integration vulnerabilities or deficiencies. Thus, those considerations need to be taken into account as one approaches the design, audit, and review of a system and its components. This problem is further complicated with closed or newly developed protocols that do not have test harnesses or that may not have a proven track record. With the rapid evolution of protocols in the smart grid, and the emerging security controls on those protocols, this deficiency may often be present.

With the domain of critical infrastructure, cyber-physical security issues also can have a different type of impact than traditional cyber security issues. As such, one must look at the interplay of the cyber-physical domain and how that interplay is affected in a bidirectional manner. This is key to the trustworthiness of the system under test and the resiliency of the critical infrastructure as a whole.

II. Facilities

To explore the security of critical infrastructure, one must be able to replicate the deployment environment with sufficient fidelity. This is where testbeds have a key role. The University of Illinois has established an extensive cyber-physical testbed facility focused on creating a high fidelity representation of the smart grid. The cyber-physical testbed facility was designed to be a realistic, flexible, configurable, and easily customizable environment that is heavily outfitted with power system communications and control hardware, software, and simulation systems and intended to enable a pipeline from fundamental research through transition to industry.

The facility uses a mixture of commercially available products and tools combined with developed research to scientifically experiment with next-generation technologies that span communications from generation through consumption. More specifically, these resources are uniquely combined in a dynamic cyber-physical framework allowing multiple experiments to utilize the resources at hand and configure them in varying topologies with minimal assistance. Through instrumentation and test harnesses, this provides a flexible framework for security testing and assessment in critical infrastructure (even beyond smart grid technology).

III. Approach

Leveraging testbeds is only part of the equation and more is required than simply having facilities in which to test. In order to conduct cyber security testing on leading-edge critical infrastructure systems, traditional approaches to testing must be adapted and applied with new constraints. The following subsections lay out the approach by defining the methodology and approaches to security measurement and describing tools that are needed to help satisfy the needs of cyber security testing. The methodology presented in the following subsections is generally applicable, and also lays the groundwork for application in the case study.

It is important to draw a distinction between security testing and the testing of security features. In security testing, one deliberately tests the ability of a system to withstand attacks, whereas testing of a security feature is a conformance check to make sure that the defined behavior happens. In this paper, we discuss security testing rather than feature testing.

A. Methodology

To accurately identify the appropriate places to test security, one must go through a series of steps and assess each component. For that purpose, each system can be broken down into 1) interfaces, 2) logic, 3) protocols, and 4) environment.

To assess those components, one needs to address them both individually and in composition. Following is a description of a generic approach to security assessment that is also applicable to smart grid assessment. The assessment needs to begin with a review that results in a hypothesis, followed by testing that either proves or disproves that hypothesis. Some example steps of the methodology for security review include the following:

1) Gather system designs and documentation.
2) Identify the components of the system, the protocols used, and the environment in which it will operate.
3) Gather protocol specifications for protocols used in the system under test.
4) Assess the system for potential inputs and outputs.
5) Analyze system boundaries in which data are transferred, or data flow diagrams.
6) Identify the threats to the system and its protocols, and the potential vectors by which those threats may enter.
7) Analyze any unintended consequences that can affect the state machine logic.
8) Assess the use of security controls.
9) Assess the use of cryptography, including keys and key management.

At each phase, observations are made and potential security problems noted. It is useful to analyze the system from both the attacker's and defender's point of view, as taking both perspectives can help reveal potential issues. Once those potential issues have been noted and the initial review completed, the reviewer can begin to create a specific test plan reflecting the observations. Wherever a potential security issue is noted, the reviewer should use quantitative or qualitative measures to make sure that the test cases have adequate coverage of the system. Coverage metrics will vary depending on the system, but one example may be based on the number of input and output interfaces checked and the number of tests checked at each of those interface points. It is worth noting that a comprehensive test plan will be derived from the problem areas and also from the areas that do not directly represent problems. By taking the approach of classifying inputs, outputs, transitions, boundaries, and composites of the components, the reviewer will end up with a complete view of the system including aspects that may not initially be security concerns.

Reviewing a protocol is often similar to reviewing a system in terms of methodology. Evaluating prudent engineering practices [2] for protocols is also often helpful, as it shows common pitfalls that help frame the test cases and a thorough review will eventually result in basic templates of common pitfalls or test cases with this coverage. One such example is a review of the Secure Authentication Extensions [8] to the DNP3 specification that was conducted by the University of Illinois. This review followed these same type of procedures as documented above and derived generic principles from that work [10] to provide guidance to others writing similar authentication extensions in resource constrained environments.

B. Overcoming Hurdles

Due to the varied conditions that security assessments face, the test environment must be based on the desired test plan. For example, will the system under test be evaluated from the perspective of software, hardware, or both? Will the interfaces to the system be tested from the supplied tools or via crafted third-party tools that may not implement the same data verification checks? The answers to those questions shape the scope of the testing effort and allow the testing plan and testing environment to be honed further.

One goal of hardware assessment is to discover undocumented or unintentional entry points into the system. Complex product engineering often unintentionally leaves open paths that can be used to penetrate a system. Another goal is to gain additional insight into the workings of the system or to intercept data or material from the system that can help with software assessment. In both cases, the approach is similar, and below we outline some of the important steps of hardware assessments:

1) Open the hardware; examine and document internal components.
2) Identify any ports, storage media, chips, communication buses, headers, or other relevant components.
3) Identify any certifications or listings where one may look up the engineering diagrams or other public information regarding the hardware. They may include FCC repositories, patent filings, UL test results, Google searches for model numbers, and product data sheets, among others.
4) Download and analyze firmware. Firmware analyses go from simple plain test extraction to full decompilation, depending on the type of firmware and the way the firmware is stored.

Hardware is often one of the more complex portions of an assessment, and this often holds true for smart grid assessments as well. With a move towards interoperability, this creates some standardization and eases the path for assessments in some cases but the platforms and implementations on which these systems are based still varies greatly. The hardware may also be hardened, making it more resistant to attack and more difficult to assess beyond the protection mechanisms.

C. Measuring Security

An important step of sound security assessment is identification of metrics that enable evaluation and comparison. If a metric offers only pass or fail conclusions, it must have had precise criteria in order to be reproducible and unambiguous. Definition of criteria and qualitative or quantitative ways to compare results is a difficult research challenge that has not yet been fully addressed by the security testing community. We next present some of the latest approaches.

A simple approach is to use guidance [13] such as that provided by NIST as the bar against which security is measured. For instance, lets suppose the protocol specification states that an implementer must use AES family encryption schemes with key lengths of no less than 256 bits. Therefore, the criterion by which to compare the protocol specification and its implementation against is straightforward. Namely, when comparing the implementation to the specification, anything other than AES-family encryption and less than 256-bit key length would not be adequate. Since the NIST guidance allows 128, 192, or 256-bit keys the use of 256bit keys in this example would pass that test as well. Unfortunately, many security tests cannot be directly evaluated with that type of clear criteria.

Metrics based on *system* behavior are likewise context sensitive, and must be interpreted with respect to assumptions about the system, its vulnerabilities, and attackers. For example, one may have sensitive information in a data historian and want some measure of how well protected the historian is from unauthorized access. This will be a function of protection mechanisms such as firewalls and access control mechanisms. Measures that focus on connectivity while under attack make sense. For example, assuming that unauthorized access to the historian is prohibited by the standard protection mechanisms, a measure of security is the minimum number of penetration/compromises needed by an external attacker to reach the historian. To quantify the metric for any given system, one needs a fairly detailed model of the system and its protection mechanisms.

As another example, one might wish to quantify the resiliency of a network facing smart grid device to an attack. A number of metrics may play a role in this assessment. How intense an attack (in terms of the bandwidth to the device for instance) can be sustained before the device simply fails due to overload, if it so fails? Or, what is the probability of a non-conforming packet affecting the device? System metrics such as these lend themselves well to being evaluated experimentally in a testbed.

Other approaches include using formal methods as a way to verify the coverage of specific security properties. While formal methods are extensively used to design and check critical hardware implementations [9], [11], their use in security quantification has been fairly limited, mainly because the state space to explore is often too large (e.g., unbounded protocol security is undecidable in theory [19]). However, recent efforts in the specific domain of intrusion detection have led to interesting strategies for measuring security [17]. We successfully applied those strategies to verify that checkers built for a specification-based intrusion detection system for AMI [4] were sufficient to provide the necessary coverage for the security policy driving them. In other words, under well-defined assumptions, it is not possible for an attacker to violate the security policy without being detected by the checkers.

We performed the successful verification by building formal models of the checkers and the security policy and developing a theorem and proof to show that all possible network traces that respect the checkers will also respect the security policy. Those models were implemented as functions and data structures in a formal framework. We used ACL2, a software tool that combines a programming language based on Common Lisp, a logic, and a theorem prover. ACL2 automates most of the proof effort using techniques such as rewriting and mathematical induction. The advantage of formal methods is that they force the precise definition of all the assumptions and offer strong mathematical guarantees about the results.

On the other hand, to apply formal methods one must put extensive effort into learning the formal specification language and the proof system and understanding the correct abstraction level needed for a particular security evaluation. Moreover, use of formal methods does not prevent inaccuracies between the formal model and the implementation from producing security issues. As a result, one should combine formal methods with other methods like experimental testing, instead of relying solely on a pure mathematical approach.

D. Leveraging Testbeds to Build Tools

What has been described so far could be characterized as a "whiteboard" style of analysis that consists mostly of a thinking and information-gathering exercise. Indeed, the methodology described earlier has focused on hypothesis definition, quantitative, and qualitative analysis. Here, we focus on the instrumentation and supporting environment that is needed to realize a realistic testbed for critical infrastructure testing.

The deployment of critical infrastructure testbeds (e.g., [1]) has been key to understanding the needs for sound instrumentation and the impact of cyber attacks on that infrastructure. The University of Illinois has built such a testbed [3], [15] with unique capabilities that allow for the union of simulation, emulation, and real equipment to mix varying degrees of fidelity, scalability, and flexibility.

Recent testbed expansion has been focused on automated configuration and instrumentation to support and facilitate secure testing both locally and through federated resources that are geographically distributed. By extending the DETER

[18] framework, the Illinois testbed has brought cyber-physical instrumentation capabilities into the DETER framework and leveraged that new capability with other testbeds around the nation. This allows an experimenter to utilize power specific equipment in the same way that cyber equipment is leveraged. Further, Illinois has added a high fidelity power simulation resource, the Real-Time Digital Simulator, as a resource to projects internal to the Illinois testbed. These cyber-physical resources are key to providing the high-fidelity and realistic environment by which systems are tested for security concerns and the *impact* of those issues on the physical systems to which they are connected.

One common roadblock to building realistic representations of the smart grid is that in most situations, these critical systems are designed to be deployed in hardened environments and connected solely in the way they were designed. However, in general they are not designed to be scientifically observable, instrumented for research, or connected in the non-standard ways that researchers may need. This poses an issue as researchers need to compose the system differently as they test, derive new protection schemes or more advanced reliability strategies.

Testbeds therefore need to reach deeper into systems than is generally done in production and provide connections that are generally not present in those systems. Naturally, deep understanding of the systems must be developed to provide that; in some cases, special relationships with the vendors may be required. A testbed creates an ideal platform for custom tools, test harnesses, instrumentation, and frameworks that aim to provide interoperability between systems. One such example is providing a bridge between power simulation software and network simulation software, allowing for cyber-physical coupled simulation experimentation.

Another major issue is that of providing the necessary fidelity at scale. To address grid scale problems on critical infrastructure, one needs an environment that is not only high-fidelity and realistic enough to validate research of appropriate scale. An expensive and unrealistic solution would be to build a duplicate smart grid to test everything without impacting the production system. Another solution would be to look at the problem in pieces, replicating only what is needed for the problem under consideration, but this piecemeal approach is cumbersome and eventually ends up in the same problem area as the first. Yet another option is to build microgrids that represent the case under consideration, but at a smaller scale.

While that approach has value, there are unsolved problems such as addressing scale or build out issues associated with supply-chain and engineering costs of deploying real hardware. Simulation can help in some cases, but it suffers from the problem that models are not always available, accurate, or capable of running in real time at the necessary scale. Finally, there are the approaches of emulation and coupling of hardware and simulation together, which round out and address some of the previously mentioned faults. Emulation brings its own issues in software complexity and coupled hardware-software simulation often requires deep understanding and custom interfacing beyond original design to work.

With each type of research, the balance between scale and fidelity has to be weighed against the need for speed. The problem is challenging even in a single experimentation domain. In cyber-physical systems like the smart grid, where two domains are operating simultaneously, the interactions between them make the inter-relations even more complicated and even more difficult to handle. The testbed at Illinois is advancing the state of the art in this area. This is done by leveraging real equipment in both representative configurations and individual components, virtualization technology for emulation and simulation, increasing the accuracy of model behavior used in simulation, and developing interconnections between systems that previously could not be connected.

IV. Case Study: Advanced Metering Infrastructure (AMI)

Having discussed general methodology, security measurements, and the applicability of testbeds, we now present a case study on application of those techniques to Advanced Metering Infrastructures (AMI) of the kind being deployed around the world. We illustrate the efficacy of the techniques through the description of tools and instruments that have been used to conduct AMI security research. The methodologies discussed above have been applied by other groups as well, as evidenced by an AMI Penetration Test Plan published recently by NESCOR [16] and a multi-vendor attack methodology paper from Pennsylvania State University [12]. Further, several studies [5], [6] have been published showing the threat landscape faced by AMI deployments.

First, it is important to understand the complexities of the AMI space. At the highest level, an AMI architecture consists of a smart meter, an aggregator, a head-end, the software running on each of the components, and the communications links among those components. While this decomposition appears to be simple, challenges arise from the heterogeneous mix of software, hardware, firmware, protocols, and functionalities that are deployed in the architectures in order to provide observability and controllability under a variety of abstraction levels. In some cases, there are even provisions for access to and integration of third parties, such as the third-party model in the AMI-SEC specification. Moreover, protective solutions including cryptography, access control, and key management are used with different modalities and configurations throughout the system.

Communication links are another example of complexity. Meters often have home-area-network interface (e.g., ZigBee) for interfacing with consumers, combined with medium-length radio interface for communication with other meters via mesh communication or with aggregation nodes (e.g., via ANSI C12.22 protocol [7]). Additionally, a meter will often have an optical port for physical interfacing, an LCD panel for human interfacing, and computer interfaces for debugging purposes. Following the communication path towards the utility network, we find that the aggregators have a medium-length radio interface to communicate with meters, and a

provision for long-haul communications that can use a variety of technologies (e.g., Ethernet, fiber, GSM, CDMA, PLC, or others). The goal of the aggregators are to take the localized data collection and bring it to the head-end, and to forward control commands from the head-end to the meters. The head-end is the main software package handling command and control for interaction with the meters, including aggregation of metering information for billing purposes.

A. Problems

Several issues become apparent in the analysis of new architectures like AMI. Such systems are designed for production use and tend not to be instrumented in a way that exposes controls that allow for deep testing. Further, these systems are designed to be secure, and security itself often makes testing more difficult. Since the AMI architecture has only recently been adopted, the associated knowledge base is limited and often not available to people who are not involved in creating, operating or installing the systems. Most of the usable tools that exist are the ones that were utilized during development of the system or developed as part of the deployment or operational processes. Access to those tools is sometimes limited, and often they are available only in executable format, which makes it difficult to adapt them for testing purposes.

Standards are another concern in emerging architectures. As standards are formed, systems built around them can become interoperable from a standards point of view but still have incompatible configurations or different maturity levels, or include nonstandardized functions. Even in areas that are standardized, there are sometimes implementation decisions that can result in different security behaviors.

Likewise, the implementation of systems is often affected by individual manufacturers' choices. For example, the communication technologies, routing methods, and authentication mechanisms chosen by a particular manufacturer are all potential entry points in a security assessment of that company's AMI. They also present security testing problems, as communication with one AMI may be different from communication with another.

B. Tools

After we defined our objectives and methods, our next step towards analysis and testing of AMI device security was to develop a set of tools that could empower researchers and testbed operators to rapidly instrument systems and start collecting metrics. The unavailability of a toolset for analyzing the specific protocols and components of AMI pushed us to develop custom applications. For example, when this work started, the well-known wireshark analyzer could not be used, as it did not include a dissector for the ANSI C12.22 communication protocol. There was a user contributed patch to provide that support however our environment was looking for a more nimble solution. Note, the most recent version of wireshark now includes a dissector for AMI.

Our efforts were focused on creation of two primary tools: 1) a multi-platform protocol dissector library that can collect and analyze AMI traffic in various locations in the network, and 2) an AMI visualization framework that can be used to gain situational awareness, quickly grasp the behavior of components, and present results to non-experts. Note that while a protocol dissector and application specific visualization tool are not novel approaches in networking, no such tools that met our needs were readily available for the c12.22 protocol.

The main two requirements in our case for both tools were rapid prototyping and easy maintainability. Those requirements are particularly important in a testbed environment in which research needs are very dynamic and the personnel involved have a wide range of backgrounds. As a result, we chose to develop the traffic dissector in Python, and the visualization solution in Processing. Both of those languages have a proven record of rapid development speed, ergonomy, and flexibility. Further, as these tools were developed, special attention was put on making the tools not only leveragable for research but also applicable for use by Industry for increased field awareness and security engagement.

The architecture of the dissector is straightforward, parsing the TCP/IP and C12.22 structures and outputting the data payload as a parsed data feed. This is then used as input into the state machine which provides the flow structure to track node behavior over time outputting its results via syslog. This state machine uses an API to enable other applications or modules to extract network-level and application-level information. For instance, a module that we implemented to build a specification-based intrusion detection sensor includes a C12.22 state machine that can track meter state transitions based on C12.22 requests and replies recorded from network traces. Constraints on those transitions allow the system to automatically trigger alerts when the behavior of a node starts to deviate from the normal behavior profile. The entire Python codebase of the tool, including the intrusion detection system, represents less than 2,000 lines of code.

The visualizer receives syslog output directly from one or more C12.22 dissectors and interprets the logs to extract the network node and flow information. As shown in Figure 1, nodes and flows are represented with a simple connected particle system of dots and lines. The particles float on the visualization canvas and a gravitational force simulation enables the particles to adopt a clear layout automatically. A color-coding scheme and a legend help to identify the types of network nodes so that one can immediately differentiate meters from relays and the collection engine.

C. Application

As mentioned above, formal methods were used to verify the specification-based intrusion detection system for AMI. The same methods could be applied in security testing as well. Each test case would be viewed as an attack, and each goal of resiliency, reliability, or prevention would be viewed as the policy. Strong mathematical guarantees about the results should add value to the testing. Each checker from the IDS is a potential testing point that could be used to inform a test plan for the protocol. The definition of the security policy for

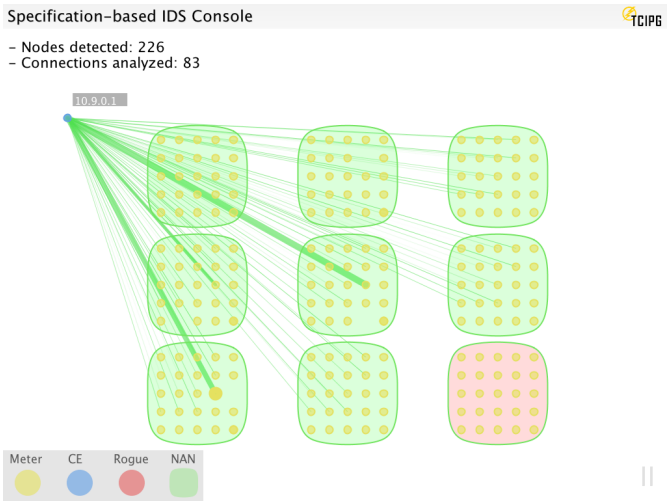

Fig. 1. Interface of the AMI visualization tool designed to analyze interactions in the mesh network

the IDS is also a potential guideline for the criteria by which the test plan is evaluated.

Experimental verification was leveraged in the testing and creation of the protocol dissector, visualization framework, and specification-based intrusion detection system to verify adherence to the specification and the ability to parse real data feeds from a variety of sources. Further, these tools are also used as a core piece involving wireless spectrum analysis to reverse engineer proprietary mesh communication and investigate attack surfaces on AMI systems, continuing the investigation of AMI security assessment. As intended, these tools have allowed the testbed researchers to rapidly augment the functionality to adapt to new research and to prove out new theories as they go forward with research and assessment of systems. This work continues to be a catalyst for further studies into attack trees and response mechanisms for AMI.

In our work with the AMISEC and NIST CSWG working groups, as well as other projects, we have been exposed to numerous proposed AMI architectures and conducted evaluations at both theoretical and practical levels. Application of the methodology laid out above for the protocol, system, architecture, and hardware has proven to be useful both in this context and in many other smart grid domains.

V. Conclusion

Security testing involving complex cyber-physical systems like the smart grid has required a combination of methodology, quantification, and testbed environments to drive tool creation to assist in the evaluation of the systems under test. This paper presents an approach to security testing methodology and illustrates the use of testbeds in developing tools for cutting-edge systems. The testbed at the University of Illinois offers a realistic environment providing state-of-the-art cyber security testing capabilities for current systems as we demonstrated through a case study in AMI. In addition to that, this paper also addresses problems in security testing and testbed creation for critical infrastructure and demonstrates progress in tackling those challenges through tool creation.

VI. Acknowledgment

This material is based upon work supported by the Department of Energy under Award Number DE-OE0000097. The authors also thank Jenny Applequist for her editorial assistance.

References

[1] National SCADA Test Bed (fact sheet). Available at http://www.inl.gov/scada/factsheets/d/nstb.pdf.

[2] M. Abadi and R. Needham. Prudent engineering practice for cryptographic protocols. *IEEE Transactions on Software Engineering*, 22(1):6–15, 1996.

[3] D. C. Bergman, D. Jin, D. M. Nicol, and T. Yardley. The virtual power system testbed and inter-testbed integration. In *Proceedings of the 2nd Conference on Cyber Security Experimentation and Test*, CSET'09, page 5, Berkeley, CA, USA, 2009. USENIX Association.

[4] R. Berthier and W. Sanders. Specification-based intrusion detection for advanced metering infrastructures. In *Proceedings of the 17th IEEE Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 184–193. IEEE, 2011.

[5] D. Grochocki, J. H. Huh, R. Berthier, R. Bobba, W. H. Sanders, A. A. Crdenas, and J. G. Jetcheva. AMI Threats, Intrusion Detection Requirements and Deployment Recommendations. In *Proceedings of the 3rd IEEE International Conference on Smart Grid Communications (SmartGridComm), Tainan City, Taiwan, Nov. 5-8, 2012*, to appear.

[6] Florian Skopika, Zhendong Maa, Thomas Bleiera, and Helmut Grneisb A Survey on Threats and Vulnerabilities in Smart Metering Infrastructures In *International Journal of Smart Grid and Clean Energy, vol. 1, no. 1, September 2012*, pp. 2228 ISSN: 2315-4462

[7] *ANSI C12.22: Protocol specification for interfacing to data communication networks*. National Electrical Manufacturers Association, 2008.

[8] DNP3 Users Group Technical Committee. DNP3 secure authentication specification version 2.0, DNP users group documentation as a supplement to volume 2 of DNP3. Technical report, DNP Users Group, 2008.

[9] D. Hardin, E. Smith, and W. Young. A robust machine code proof framework for highly secure applications. In *Proceedings of the Sixth International Workshop on the ACL2 Theorem Prover and its Applications*, pages 11–20. ACM, 2006.

[10] H. Khurana, R. Bobba, T. Yardley, P. Agarwal, and E. Heine. Design principles for power grid cyber-infrastructure authentication protocols. In *Proceedings of the 2010 43rd Hawaii International Conference on System Sciences*, HICSS '10, Washington, DC, USA, 2010. IEEE Computer Society.

[11] H. Liu. *Formal Specification and Verification of a JVM and its Bytecode Verifier*. PhD thesis, The University of Texas at Austin, 2006.

[12] S. McLaughlin, D. Podkuiko, S. Miadzvezhanka, A. Delozier, and P. McDaniel. Multi-vendor penetration testing in the advanced metering infrastructure. In *Proceedings of the 26th Annual Computer Security Applications Conference (ACSAC)*, pages 107–116, New York, NY, USA, 2010. ACM.

[13] National Institute of Standards and Technology. NIST 800-57: Recommendation for key management - part 1: General. Available at http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57-Part1-revised2_Mar08-2007.pdf.

[14] National Institute of Standards and Technology. NIST IR 7628: Guidelines for smart grid cyber security. Available at http://csrc.nist.gov/publications/PubsNISTIRs.html#NIST-IR-7628.

[15] D. M. Nicol, C. M. Davis, and T. Overbye. A testbed for power system security evaluation. *Int. J. Inf. Comput. Secur.*, 3(2):114–131, Oct. 2009.

[16] J. Searle, G. Rasche, A. Wright, and S. Dinnage. Available at http://www.smartgrid.epri.com/doc/AMI-Penetration-Test-Plan-1-0-RC3.pdf.

[17] T. Song, C. Ko, C. Tseng, P. Balasubramanyam, A. Chaudhary, and K. Levitt. Formal reasoning about a specification-based intrusion detection for dynamic auto-configuration protocols in ad hoc networks. *In Proceedings of the Fourth International Workshop on Formal Aspects in Security and Trust*, pages 16–33, 2006.

[18] USC ISI. DETER Project. http://deter-project.org.

[19] S. Whalen, M. Bishop, and S. Engle. Protocol vulnerability analysis. Technical report, 2005.