Kalpa Publications in Engineering
Volume 2, 2018, Pages 30–36
Proceedings on International Conference on Emerging Trends in Expert Applications & Security (2018)

Malware Detection for Cyber Security Enhancement in Smart Grid

Li Congmiao[1,2], Dipti Srinivasan[1]*, and Thomas Reindl[2]
[1] Department of Electrical & Computer Engineering, National University of Singapore,
4 Engineering Drive 3, Singapore 117583
[2] Solar Energy Research Institute of Singapore, National University of Singapore,
7 Engineering Drive 1, Singapore 117574
licongmiao@nus.edu.sg, dipti@nus.edu.sg

Abstract

Many embedded systems in a Smart Grid have special constraints in terms of timing, cost and power consumption to ensure security. This paper addresses the Smart Grid security problem with a focus on improving the security of crucial components, and reducing the risks from cyber attacks. A hardware architecture to enhance the security of important embedded devices in the smart grid has been proposed and implemented. This hardware based malware detection system runs on a dedicated hardware implemented with FPGA logic, and allows detection in near real-time. The system architecture and results are presented in the paper.

1 Introduction

The traditional electricity grid typically has one—way power flow from large-scale power generation sources to the distribution substations through extensive networks of transmission to supply electricity to a large number of customers. In recent years, these systems are undergoing rapid transformation with inclusion of smart grid features, and the increasing penetration of distributed and renewable energy sources due to environmental concerns. This has necessitated the need for improved grid reliability, higher operational efficiency, and superior communication infrastructure to allow for two-way transfer of electrical power and data. The smart grid features are being incorporated into the grid in stages, and cover various operational aspects of the power distribution system. These distribution networks are being transformed as a result, to accommodate a large number of renewable power generation technologies, hybrid electric vehicles, virtual power plants, and demand response options. The future grid will incorporate extensive automation and advanced distributed algorithms to allow for bidirectional flow of electricity and information, so that it can efficiently respond to demand and price signals for improved efficiency and reduced cost. Additionally, it will be able to autonomously respond to events that occur anywhere in the grid such as transformer failure and demand profile shaping.

Research, development and deployment of efficient algorithms and technologies for optimal management of such smart electricity grids are gaining considerable popularity in the industry and academia. The building blocks of future home energy management systems will be smart meters. These devices, which will be installed in all residential and industrial premises, will collect real time data for various applications.

In addition to enhancing the capabilities of these meters which are connected to the end points of electricity consumption chain [Charavi, 2011], many more issues need to be considered from operational reliability and security points of view.

The research and development in the field of smart grid spans three domains: (1) the infrastructure for information, communication and delivery of energy, (2) smart management system for implementation of distributed control and energy management applications, and (3) smart protection system for incident detection, fault restoration, advanced grid reliability analysis, and security & privacy protection.

1.1 Demand Side Management (DSM)

The smart infrastructure includes advanced tools for metering which generate information from sensors, smart meters, and phasor measurement units (PMUs) which are deployed across the network, and information management as well as communication subsystems which transmit data among different systems, devices and applications in the smart grid. It supports two-way information flow between various entities and provides the foundation for realizing smart management and control system. Cyber security is one of the biggest challenges in smart grid. The system has several points which make it possible for an attacker to gain access to the control software, user data, and load profiles. Such vulnerabilities can potentially allow the attacker to change load conditions which can destabilize the grid in unpredictable ways [Metke, 2010]. The smart meters can bring security issues because they are easy target for malicious hackers. The attackers can manipulate the information on electricity price signals or fabricate energy meter readings of a compromised meter. In the scenario that a few central controllers control millions of smart meters, a compromised controller can cause disastrous results. In general, many of the privacy and security issues that exist in the modern internet and wireless communication networks are also present in a smart grid.

Malicious attacks on information transmission in smart grid include three major types based on their goals [Lu, 2010]. They are network availability (DoS) attacks; data integrity attacks (modify or corrupt information), and information privacy attacks (eavesdrop on communications). The authors in [Khurana, 2010] discussed a set of design principles and engineering practices that can help ensure the correctness and effectiveness of standards for authentication in smart grid protocols.

This is a very complex system-related problem with multiple levels of interdependencies between the variable distributed generation and user demand, potential interaction between the grid and the PV systems and possible storage devices. This paper addresses the Smart Grid security problem with a focus on improving the security of crucial components, and reducing the risks from cyber attacks. A hardware architecture to enhance the security of important embedded devices in the smart grid was also proposed and implemented. Demand Side Management (DSM) requires sophisticated real-time embedded control systems to manage distributed energy resources. Many embedded systems in smart grid have special constraints in term of timing, cost and power consumption to ensure security.

In the context of a smart grid, researchers have proposed various techniques to prevent and detect malware. One promising approach is called attestation, which belongs to anomaly-based technique. It verifies the signature of executing code remotely to detect any alteration by malware. LeMay et al. proposed architectures to provide remote attestation for smart meters [LeMay, 2009, Seshadri, 2004]. They check the integrity of cryptographic hash of the firmware revisions in the audit data using flash microcontroller units and provide attestation through virtual machines implemented on commodity desktops. Compared with their work, our approach was implemented on FPGA with much lesser hardware overheads and was able to achieve finer granularity and faster detection speed. Other software-based attestations do not rely on specialized hardware, but make assumptions that the verifier can uniquely communicate with the device after verification [Seshadri, 2004, Shah, 2008]. Shah et al. implemented this concept on the devices in SCADA system.

In this paper, we propose a hardware based malware detection system to allow detection in near real-time. The system runs on a dedicated hardware that is implemented with FPGA logic. It is directly integrated with an embedded processor to monitor program execution and send immediate feedback to the processor. The main idea is to generate the function call graphs for the program to be monitored through static analysis of its source code before execution. During runtime, the processor sends execution progress to the detection system. If the execution violates the flow defined in the function call graph, then a malicious attack is reported. The following sections describe the proposed malware detection system, its hardware implementation, and experimental results on benchmark sets.

2 Proposed Malware Detection System

The detailed hardware architecture of our malware detection system is shown in Figure 1. It includes two main subsystems, which are general purpose embedded system and real-time detection system. The two systems run in parallel separately with feedback from each other.

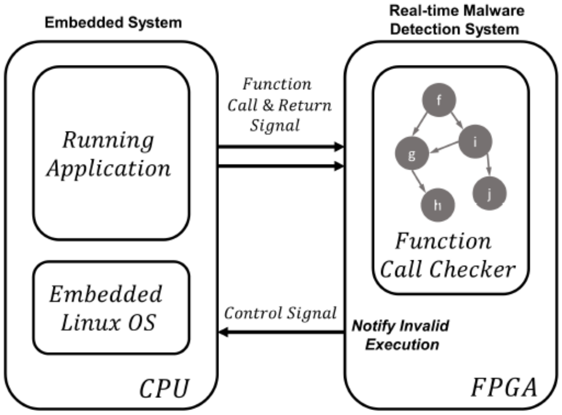


Figure 1. System Architecture of the proposed malware detection system

The general purpose embedded system is the same as the conventional one, which includes an embedded CPU, necessary I/O pins, memory and other elements to support application execution. The CPU runs embedded Linux operating system. User applications execute on top of the operating system. The embedded system sends real-time function calls and returns signals of the target user application to be monitored to the detection system in real-time. This updates the detection system with current progress in terms of function calls or returns of the running application. For example, the smart meter can be implemented on the embedded system, and monitored in real-time. Additionally, there could be multiple applications running in parallel to provide different functionalities such as controlling all the smart home appliances and communicating with the utility company.

The real-time detection system is implemented in FPGA. With the re-configurable feature of FPGA, the detection logic can be changed easily according to different applications running in the CPU. For smart meter, the real-time detection logic will be configured dynamically according to the current running application on the meter. The detection system consists of a function call checker for the running application. It checks the function calls, and returns with the given information stream from the embedded system and pre-generated function call graphs. The function call graph represents the sequence of permissible control flow. If the program flow deviates from the graph, a control signal will be triggered to stop or notify invalid execution in the embedded processor. Any such invalid execution will be assumed as a malicious attack.

The function call graph is generated offline at compile time of the application. This process is considered as a static analysis of the source code without running the code. It firstly parses the source code to generate the syntax tree, and then finds names of the defined functions and the functions they are calling to generate the call graph. The graph is then used to design the logic of FPGA for detection system. We assume that this process is carried out in a secured environment where hackers are not able to alter the source code or call graph. Before runtime, the application binary is loaded to CPU and the bit stream of detection system is loaded to the FPGA fabric.

3 System Implementation

The key component for the malware detection system is the function call checker. To implement it, we need to firstly generate the function call graph offline at a secure environment as mentioned previously. Each node in the graph represents a function, and an edge indicates a function call or return from one function to another. The function call graph (FCG) is then converted to a finite state machine (FSM) for hardware implementation. We use the Mealy type state machine to implement the FSM, where the outputs of the machine depend on both the inputs and its current state. In our design as shown in Figure 2, the finite state machine with N states takes the input, which is set to 1 if it is a function call or 0, which represents a function return, and another set of input bits, which are the binary representation of function index, as well as the value of current state index to decide the value of output. The FSM only has one bit output v, which is set to 1 for valid transition and 0 for an invalid one.

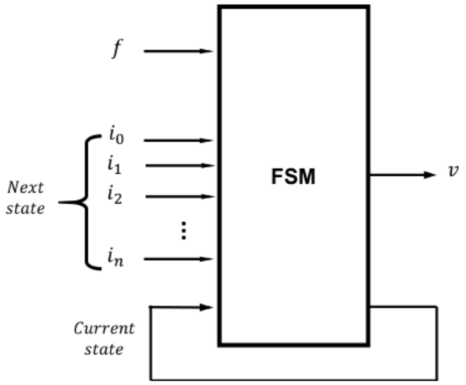


Figure 2. Hardware implementation block diagram for finite state machine (FSM)

A FCG with *N* functions is translated into a FSM with *N* states. Function with index *i* is mapped to state *i*. For each edge *eij* in FCG, add a transition from *statei* to *statej* with input *f=1* and and output *v=1* for function call, and another transition in the reverse direction from *statej* to *statei* with input *f=0* and and output *v=1* for function return. Therefore, transitions with input values defined above will output 1 to indicate a valid execution. Other input values or transitions not defined will output 0 to report an invalid transition to the running application. The generated FSM is stored as DOT (graph description language) format for VHDL code generation in the next step.

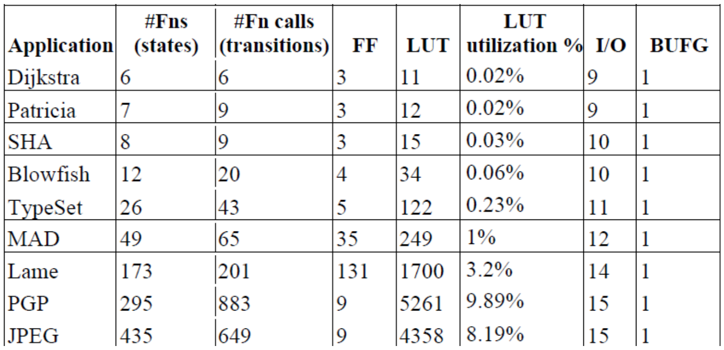
After generating the finite state machine in DOT format, an automated tool has been developed to interpret the DOT graph and translate it into VHDL file for hardware synthesis and implementation. We use the Vivado Design Suite from Xilinx to perform hardware synthesis and implementation for the Zynq™-7000 All Programmable SoC. This hardware was selected because the Zynq-7000 consists of a complete ARM®-based Processing System, and a tightly integrated FPGA fabric which resemble the architecture of our proposed hardware-assisted malware detection system. It also allows partial reconfiguration, which could be useful to extend its application to other devices and functions in the smart grid.

4 Experimental Results

As our proposed system is application-specific, and it scales with different application sizes, we chose the applications from MiBench benchmark suite [Guthaus, 2001] to generate realistic embedded system workloads. The suite is a commercially representative embedded benchmark suite, which contains applications from different domains including automotive/industrial, consumer, office, network, security and telecom sectors.

We chose representative applications within different size ranges, and ignored the trivial ones with less than five function calls to study the area overheads. We synthesized and implemented the FSM for function call checker on the Zynq-7000 from Xilinx, which includes embedded arm processor as well as field programmable gate arrays (FPGAs). The area overheads for different benchmark applications are listed in Table 1.

Table 1: Area overheads for FSM



Application	#Fns (states)	#Fn calls (transitions)	FF	LUT	LUT utilization %	I/O	BUFG
Dijkstra	6	6	3	11	0.02%	9	1
Patricia	7	9	3	12	0.02%	9	1
SHA	8	9	3	15	0.03%	10	1
Blowfish	12	20	4	34	0.06%	10	1
TypeSet	26	43	5	122	0.23%	11	1
MAD	49	65	35	249	1%	12	1
Lame	173	201	131	1700	3.2%	14	1
PGP	295	883	9	5261	9.89%	15	1
JPEG	435	649	9	4358	8.19%	15	1

In general, the results show that the lookup table LUT count grows with increasing number of functions and function calls in applications. For the sample experiment, the LUT counts show a near linear increase with rising complexity of FSM. For the largest application PGP, it utilized 9.89% of the LUT resources, which is still within a reasonable range. For applications with average size, the utilization rate is around 1%, which means with a relative small size of extra hardware logic, the embedded system can enjoy much better security assurance. This is especially important for Internet connected critical control systems in the smart grid environment.

With rapid technological developments in the FPGA industry, FPGAs are able to provide significantly reduced power, increased speed and lower cost solutions. The relative overheads compared with the total available hardware resources will be greatly reduced when implementing hardware based malware detection system. This will also enable more sophisticated detection techniques to be implemented in hardware with evolving malware detection techniques.

5 Conclusions

In this paper, the Smart Grid security problem has been addressed, with a focus on improving the security of crucial components and reducing the risks from cyber attacks. A hardware architecture to enhance the security of important embedded devices in the smart grid has been proposed and implemented. This hardware based malware detection system runs on a dedicated hardware implemented with FPGA logic. The hardware architecture can be implemented to protect all the data and control signals from cyber attacks. The results show that the proposed architecture is capable of allowing malware detection in near real-time.

The proposed architecture can be implemented for various data analysis and control systems in smart grid environment. In addition, we can utilize the existing TrustZone technology in ARM processors, which provides a secure enclave to isolate trusted applications in a secure execution environment, to implement the critical control systems in the trust domain compared with normal applications.

References

A. H. Gharavi and R. Ghafurian. “Smart grid: The electric energy system of the future,” Proc. IEEE, 99(6): 917 – 921, 2011.
R. Metke and R. L. Ekl. “Security technology for smart grid networks,” IEEE Trans. Smart Grid, 1(1):99–107, 2010.
Z. Lu, X. Lu, W. Wang, and C. Wang. “Review and evaluation of security threats on the communication networks in the smart grid,” Military Communications Conference’2010, pages 1830–1835, 2010.
H. Khurana, R. Bobba, T. Yardley, P. Agarwal, and E. Heine. “Design principles for power grid cyber-infrastructure authentication protocols,” Hawaii International Conference on System Sciences, pages 1–10, 2010.
M. LeMay, G. Gross, C. Gunter, and S. Garg, "Unified architecture for largescale attested metering," in Proc. Annu. Hawaii Int. Conf. Syst. Sci., Jan. 2007.
M. LeMay and C. A. Gunter, "Cumulative attestation kernels for embedded systems," in Proc. Eur. Symp. Res. Comput. Security, Sep. 2009, pp. 655–670.
Seshadri, A. Perrig, L. van Doorn, and P. Khosla, "SWATT: softwarebased attestation for embedded devices," in Proc. IEEE Symp. Security Privacy, May 2004, pp. 272–282.
Shah, A. Perrig, and B. Sinopoli, "Mechanisms to provide integrity in SCADA and PCS devices," in Proc. Int. Workshop Cyber-Physical Syst. Challenges Appl., Jun. 2008.
M.R. Guthaus, J.S. Ringenberg, D. Ernst, T.M. Austin, T. Mudge, and R.B. Brown, "MiBench: a free, commercially representative embedded benchmark suite,” in Proc. IEEE Fourth Ann. Workshop Workload Characterization, Dec. 2001.