Detecting Malicious Software Execution in Programmable Logic Controllers Using Power Fingerprinting

Carlos Aguayo Gonzalez, Alan Hinton

Chapter 2

DETECTING MALICIOUS SOFTWARE EXECUTION IN PROGRAMMABLE LOGIC CONTROLLERS USING POWER FINGERPRINTING

Carlos Aguayo Gonzalez and Alan Hinton

Abstract Traditional cyber security mechanisms, such as network-based intrusion detection systems and signature-based antivirus software, have limited effectiveness in industrial control settings, rendering critical infrastructure assets vulnerable to cyber attacks. Even four years after the discovery of Stuxnet, security solutions that can directly monitor the execution of constrained platforms, such as programmable logic controllers, are not yet available. Power fingerprinting, which uses physical measurements from a side channel such as power consumption or electromagnetic emissions, is a promising new technique for detecting malicious software execution in critical systems. The technique can be used to directly monitor the execution of systems with constrained resources without the need to load third-party software artifacts on the platforms.

This paper demonstrates the feasibility of using power fingerprinting to directly monitor programmable logic controllers and detect malicious software execution. Experiments with a Siemens S7 programmable logic controller show that power fingerprinting can successfully monitor programmable logic controller execution and detect malware similar to Stuxnet. Indeed, power fingerprinting has the potential to dramatically transform industrial control system security by providing a unified intrusion detection solution for critical systems.

Keywords: Industrial control systems, malware detection, power fingerprinting

1. Introduction

Industrial control systems are computer-based systems that monitor and control process systems in critical infrastructure assets such as water treatment and distribution facilities, transportation systems, oil and gas pipelines,

electrical power transmission and distribution systems, and large telecommunications systems. Attacks against industrial control systems by a well-funded adversary can have devastating consequences to modern society.

Current industrial control system defenses involve updating and patching, strengthening the periphery and implementing other traditional information technology solutions. Unfortunately, these approaches have limited success in industrial control system environments [11], which render critical systems highly vulnerable to cyber attacks – as Stuxnet famously demonstrated [13]. Consider, for example, intrusion detection systems that rely on traffic analysis. Such systems are notoriously ineffective against advanced persistent threats, which leverage attacks that are immune to signature detection, minimize network utilization and mimic legitimate network traffic [6, 12, 19]. Furthermore, the systems are incapable of detecting malicious software whose execution does not generate traditional network traffic. For example, the malware could communicate using alternative channels (e.g., Bluetooth [8]) or simply remain dormant for extended periods of time. Signature-based solutions also have severe shortcomings in industrial control system environments, including the inability to detect zero-day attacks [7, 9, 14, 15], the consumption of valuable host resources that CPU-constrained platforms simply do not have [11, 16], and the lack of support for embedded systems [10].

Power fingerprinting (PFP) is a promising new technique that detects malicious software execution using physical side channel measurements. The technique involves the direct monitoring of systems with constrained resources and does not require the loading of third-party software artifacts on target platforms. As such, power fingerprinting is ideal for detecting malicious software execution in industrial control system environments and can provide an extra layer of protection that is not afforded by traditional intrusion detection approaches.

This paper demonstrates the feasibility of using power fingerprinting to directly monitor programmable logic controllers and detect malicious software execution. The experimental results demonstrate that power fingerprinting can successfully detect the execution of malware similar to Stuxnet in a Siemens S7 programmable logic controller.

2. Power Fingerprinting

Power fingerprinting analyzes a processor side channel, such as power consumption or electromagnetic emissions, to determine whether or not it deviates from expected operation. A power fingerprinting monitor, shown in Figure 1, uses a physical sensor to capture electromagnetic signals containing small patterns that emerge during the transition from one instruction to another. In power fingerprinting, captured power traces are processed by an external device that implements signal detection and classification techniques. The observed traces are compared against baseline references to assess whether or not execution has deviated from its expected behavior, such as when malware alters normal operation.

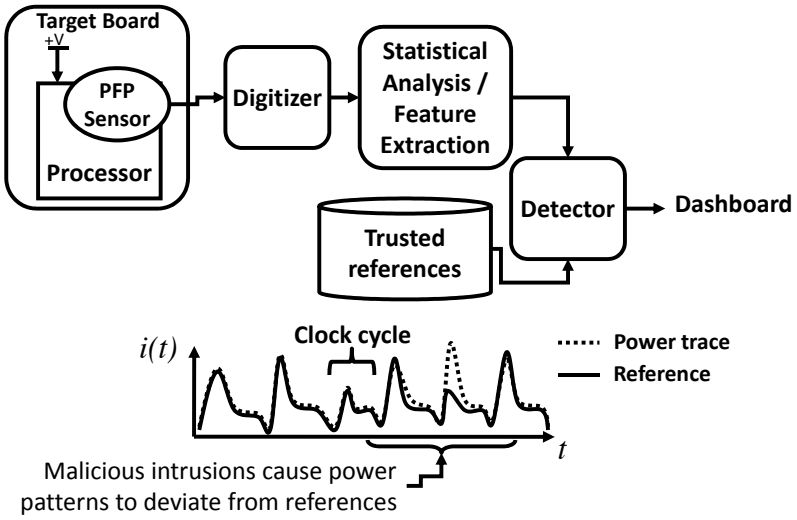

Figure 1. Power fingerprinting monitor.

Because monitoring is performed on an external device, memory and processing overhead on the target device are eliminated. Additionally, a power fingerprinting monitor can be built using commercial off-the-shelf components.

2.1 Basic Concepts

The concept behind power fingerprinting is relatively straightforward. It involves three main elements that are common to pattern recognition systems: (i) sensing; (ii) feature extraction; and (iii) classification. Sensing involves direct or indirect measurements of the instantaneous current drain. The measurements may be made using a variety of techniques, including current or electromagnetic probes.

During a runtime assessment, power fingerprinting compares the captured traces against baseline references and looks for deviations beyond what are characterized as normal execution. The baseline references uniquely identify the execution of software routines that are extracted in a controlled environment before the system is deployed. The power fingerprinting monitor uses the stored references to detect anomalous execution deviations at runtime.

The level of expected deviation during normal operation is identified during a characterization process that determines the threshold between normal and anomalous execution. An intrusion is deemed to have occurred when the observed traces do not match the baseline references within a defined tolerance.

2.2 Characterization

The baseline references contain the expected side channel signals and indicate the acceptable tolerance variation. Power fingerprinting baselines are

determined by exercising a good sample in a controlled environment while capturing side channel signals. Note that this process is similar to automated software testing; thus, power fingerprinting can leverage existing tools to facilitate the baseline extraction process. Indeed, while references are unique to a target system, the process for extracting them is general and can be applied across platforms and applications.

Ideally, a reference is extracted for every execution path in the target. Programmable logic controllers are ideal candidates for complete characterization because their execution is limited in functionality. In cases where extracting a reference for every execution path is not feasible due to complexity, the characterization may focus on critical system modules (e.g., kernel and bootloader).

2.3 Advantages and Limitations

Power fingerprinting enables the continuous, real-time and direct monitoring of industrial control devices that currently lack commercial solutions for detecting malicious software execution. Power fingerprinting can detect malware that induces the slightest anomalies in execution, even when the malware remains dormant or mimics legitimate network traffic. This enhanced detection capability enables the implementation of immediate responses to neutralize a threat. Furthermore, power fingerprinting does not interfere with the operation of critical industrial control systems, allowing the monitoring of the most sensitive components.

While power fingerprinting is a powerful mechanism for detecting malicious software execution, it provides limited support for forensic analysis and attack attribution. Specifically, power fingerprinting can identify the modules that have been tampered with, but not the modifications made to the system or the attacker's intentions. Power fingerprinting is intended to be applied in a defense-in-depth approach as part of a comprehensive security solution.

2.4 Related Work

Power fingerprinting has been demonstrated in a number of experiments on a variety of target platforms [1–5]. Aguayo Gonzalez and Reed [4] have detected unauthorized software modifications in a basic commercial radio platform (PICDEM Z Evaluation Board with a PIC18 processor). The unauthorized modifications had a physical impact on the behavior of the system that could trigger regulatory certification violations. In a different experiment using the same platform, Aguayo Gonzalez and Reed [3] used a power fingerprinting monitor to detect execution deviations that affect the encryption process of radio transmissions.

Other researchers (e.g., [17, 18]) have used techniques similar to power fingerprinting for industrial control system security. In particular, they have used electromagnetic emissions to detect anomalies in Allen Bradley SLC-500 programmable logic controllers using a correlation-based approach.

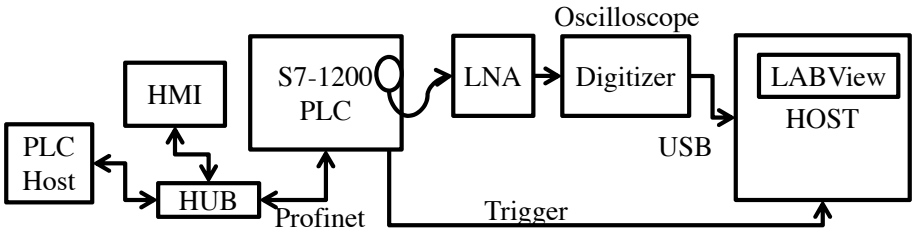

Figure 2. Power fingerprinting monitor setup.

3. Experimental Setup and Methodology

This section demonstrates the ability of power fingerprinting to monitor industrial control systems and identify malicious software execution. A reference system was implemented using a Siemens SIMATIC S7-1200 micro programmable logic controller to extract its power fingerprinting baseline references. A malicious modification, similar in structure and operation to Stuxnet, was introduced in the programmable logic controller and the baseline references were used to detect the resulting anomalous execution. The following sections describe the experimental setup and methodology.

3.1 Target Platform

The Siemens SIMATIC S7-1200 micro programmable logic controller used in the experiments had a 1212C CPU; a scalable and flexible design for compact solutions; an integrated Industrial Ethernet/PROFINET interface for programming, I/O and HMI connections, and CPU-to-CPU communications; and integrated technology functions for counting, measurement, closed-loop control and motion control.

3.2 Measurement Setup

The power fingerprinting monitor was implemented using commercial off-the-shelf components. The target programmable logic controller was first instrumented with a near-field sensor for electromagnetic compatibility testing to capture the side channel signal. The near-field sensor employed was a commercial probe from Beehive Electronics with fine spatial resolution that reduced interference from other subsystems on the board. The increased spatial resolution resulted in reduced sensitivity, which was compensated for by a wide-band amplifier with 30 dB gain. The power fingerprinting monitor setup is presented in Figure 2.

The signal captured by the sensor was digitized using a Tektronix oscilloscope. The oscilloscope was configured with a sampling rate of 2.5 GSPS; a total of 100K samples were collected in each trace. A triggering signal was provided by an I/O pin in the programmable logic controller for synchronization

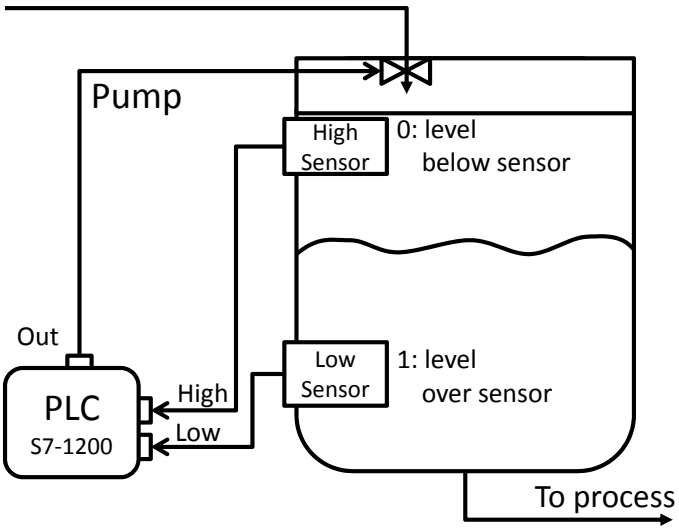

Figure 3. Tank level control system.

purposes. The captured signals were transferred via a USB drive and processed by the power fingerprinting host using custom software tools and scripts.

3.3 Control System Logic

The experiment involved a simple tank level control system shown in Figure 3. In the experiment, the S7-1200 programmable logic controller was used to control the tank level using two sensors to determine when to turn the pump on and off.

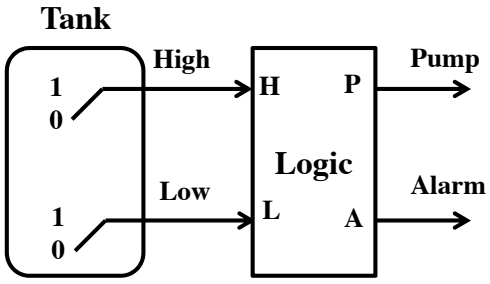

Figure 4. Control system operation.

Figure 4 shows a simplified model of the control logic. The sensors were configured to provide a logical one when the tank water level was at or above the sensor level and a logical zero when the water level was below the sensor level.

Table 1. Control system logic table.

High Sensor	Low Sensor	Pump	Alarm
0	0	1 (On)	0 (Off)
0	1	*	0 (Off)
1	0	0 (Off)	1 (On)
1	1	0 (Off)	0 (Off)

According to the control system logic shown in Table 1, the programmable logic controller turns the pump on when the tank level drops below the low sensor and turns the pump off when the level reaches the high sensor. When the level is between both sensors (low sensor = 1 and high sensor = 0), there is no change in the pump state. The remaining combination of input values (low sensor = 0 and high sensor = 1) is a faulty condition and raises an alarm.

The control system logic was implemented in the S7-1200 programmable logic controller as a SCL program in block OB1. The following pseudocode specifies the control system logic:

```
// Power Fingerprinting Trigger
if L = 0 && H = 0 then
    pump = On
    alarm = Off
else if L = 1 && H = 1 then
    pump = Off
    alarm = Off
else if L = 0 && H = 1 then
    alarm = On
    pump = Off
    increase alarm counter
else
    outputs unchanged
end
// Power Fingerprinting Trigger
```

The control system logic has four execution paths. An execution path is selected based on the combination of input values at the beginning of the logic cycle. To facilitate synchronization, the logic incorporates a physical trigger, an electric signal sent to the digitizer via the output port of the programmable logic controller to indicate when the logic cycle is started.

3.4 Modified Control System Logic

In order to test the ability of power fingerprinting to detect malicious software execution, the control system logic was modified to incorporate a malicious

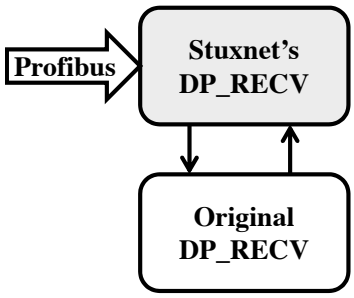

Figure 5. Functional representation of the attack.

attack. The alteration resembles the Stuxnet modification to Siemens S7-315 programmable logic controllers that hooked DP_RECV to collect information about normal uranium hexafluoride centrifuge operations.

The attack, which is shown in Figure 5, moves the original DP_RECV routine to a different logic block and replaces it with an infected block that monitors inputs and forwards requests to the original DP_RECV routine. The attack causes the pump to be turned on regardless of the sensor inputs while disabling the alarm.

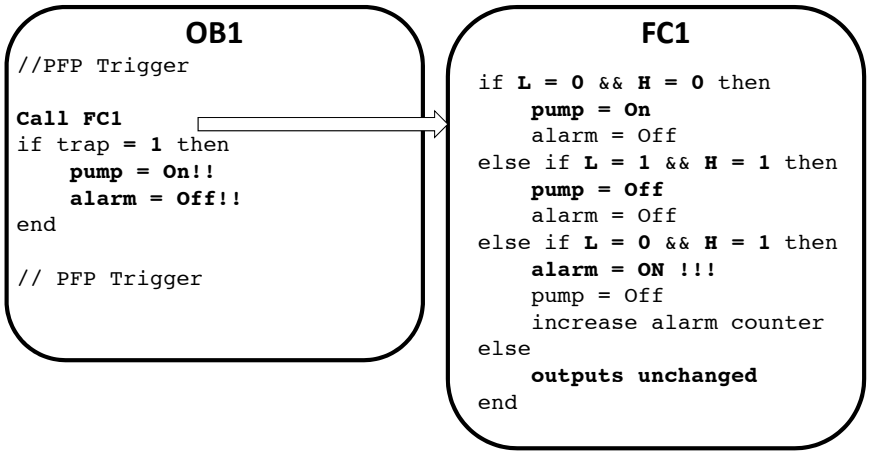

Figure 6. Modified control system logic SCL OB1.

Figure 6 shows how the original logic block is moved in the tampered version. After the original logic is executed, the tampered block post-processes the results to change the system behavior. The most important element of the tampering, however, is the fact that behavioral modifications only take place under specific conditions. Similar to Stuxnet, the attack remains dormant and the system exhibits normal behavior until the triggering condition is encountered.

The triggering condition is induced by another digital input pin that controls the sabotage routine. Note that the triggering mechanism is arbitrary; selecting

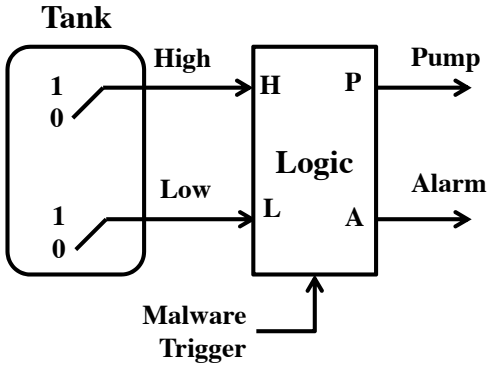

Figure 7. Tampered control system operation.

a different triggering mechanism would have no impact on power fingerprinting. Figure 7 shows a simplified model of the tampered control system logic.

Table 2. Tampered control system logic table.

High Sensor	Low Sensor	Malware Trigger	Pump	Alarm
x	x	1	1 (On)	0 (Off)
0	0	0	1 (On)	0 (Off)
0	1	0	*	0 (Off)
1	0	0	0 (Off)	1 (On)
1	1	0	0 (Off)	0 (Off)

Table 2 shows the tampered control system logic. When the triggering condition is induced, the programmable logic controller turns the pump on regardless of the sensor inputs, causing the water in the tank to overflow. When the triggering condition is absent, the observed behavior matches the original logic.

4. Experimental Results

After characterizing the original control logic and extracting the power fingerprinting references for all the execution paths, the power fingerprinting monitor was able to effectively monitor the integrity of the Siemens S7-1200 programmable logic controller. Furthermore, power fingerprinting successfully detected malicious software execution even when the triggering condition was absent.

4.1 Baseline Reference Extraction

In order to perform the runtime assessment of the original programmable logic controller, it was necessary to extract the baseline references for all the

execution paths during the characterization process. Training traces were captured in a controlled environment in which input vectors were provided to exhaustively exercise all the possible execution paths.

A total of 100 training traces were captured for each execution path and processed using a spectral periodogram (spectrogram) to extract the frequency components of each training trace at different time segments. The spectrogram, which corresponds to the squared magnitude of the discrete-time short-time Fourier transform ($X(\tau, \omega)$), is given by:

$$\texttt{spectrogram}\{x\,(t)\}\,(\tau,\omega) = |X\,(\tau,\omega)\,|^2$$

where

$$X\,(m,\omega) = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-j\omega n}.$$

Note that $x[n]$ is the captured power fingerprinting trace and $w[n]$ is a Gaussian window. The power fingerprinting references were constructed by averaging the spectrograms of the 100 training traces for each execution path. For Path 0, the power fingerprinting reference is denoted by S_0; for Path 1, the power fingerprinting reference is denoted by S_1; and so on.

After the references for each execution path were computed, the power fingerprinting monitor captured a new runtime test trace $r[n]$, and compared it against the references to determine if $r[n]$ corresponded to an authorized execution path or if it should be flagged as an anomaly. In order to match $r[n]$ to a specific path reference S_i, the spectrogram of $r[n]$ was computed and subtracted from each baseline reference over selected time segments and frequency bands. The difference was then smoothed and summed across the selected time segments and frequency bands to determine the final distance for each path reference y_i.

The reference that produced the minimum distance from the test trace, $y_f = \texttt{min}_i\{y_i\}$, was selected as the likely execution path that generated the test trace $r[n]$. If y_f is within the normal range as determined during the characterization, the power fingerprinting monitor classifies the trace as belonging to the corresponding execution path. If the test trace does not match any reference within the predefined tolerance, then the power fingerprinting monitor determines that an anomaly exists and raises an alarm.

4.2 Detection Performance

The ability of power fingerprinting to detect malicious software execution was tested by capturing 100 traces from the tampered programmable logic controller with the malware in a dormant state (i.e., the triggering condition was absent and the tampered version displayed the same observable behavior as the original logic).

Figure 8 shows the sample distribution (histogram) of the differences (y_f) between the original execution traces and the traces during the execution of

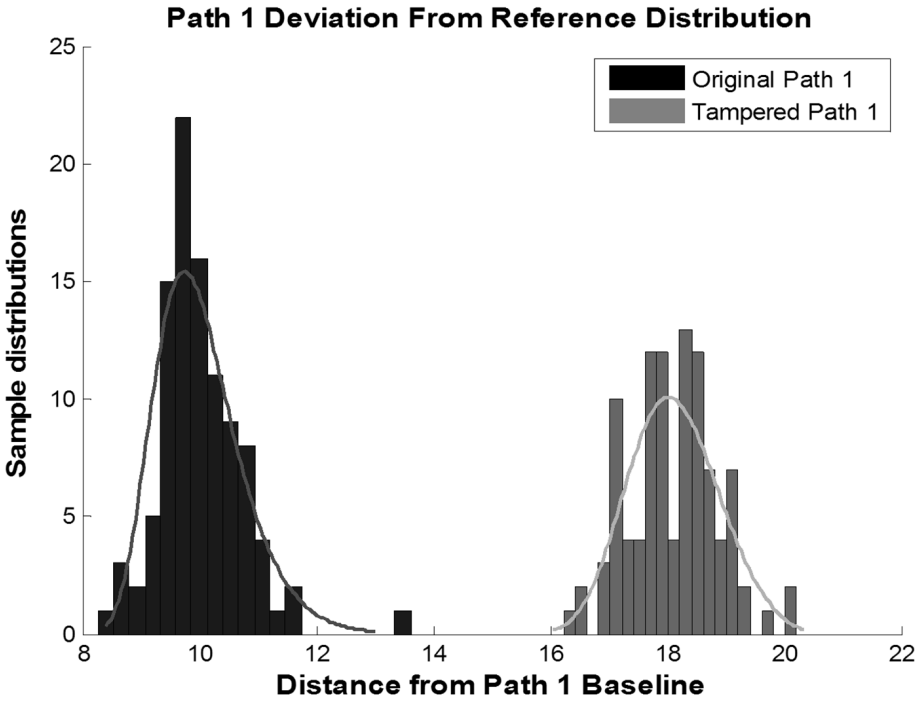

Figure 8. Deviation of Path 1 from the baseline sample distribution.

the tampered control system logic. Note that the closer y_f is to zero, the more similar the tampered execution trace is to the baseline reference trace. A clear separation can be seen between the distributions, which demonstrates the ability of power fingerprinting to detect malicious software execution.

Similar results were obtained for the other execution paths. Figure 9 presents a boxplot of an aggregated view of the execution paths. The boxplot shows that the separation between the original and tampered distributions is maintained for all possible execution paths. The results demonstrate the ability of power fingerprinting to detect malicious software in an industrial control system by directly monitoring programmable logic controller execution.

5. Conclusions

Power fingerprinting is a novel technique for directly monitoring the execution of systems with constrained resources. The technique, which has been successfully demonstrated on a variety of platforms, does not require software artifacts to be loaded on the target platforms.

The experimental results demonstrate that power fingerprinting can directly monitor programmable logic controller execution and detect the presence of malware. Because of its zero-day detection capability and negligible overhead,

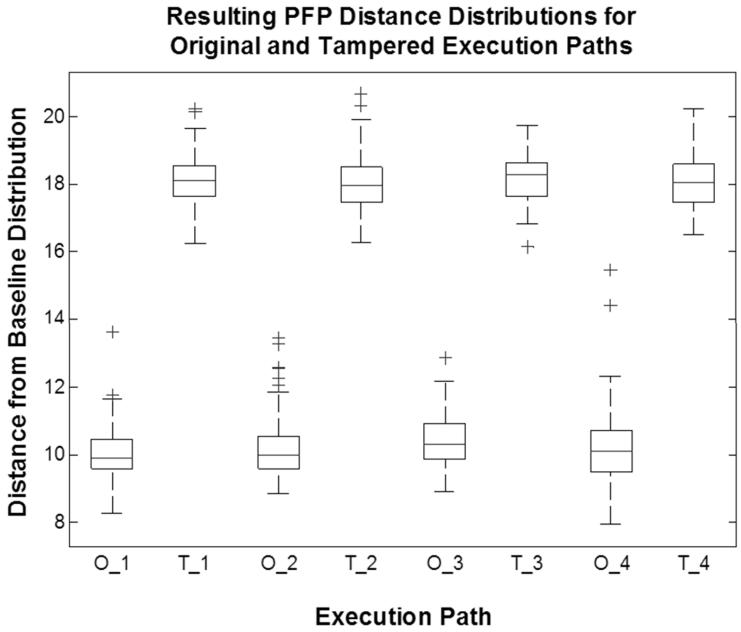

Figure 9. Anomaly detection performance for execution paths in the original logic.

power fingerprinting can potentially transform cyber security by enabling malware detection in industrial control systems as well as in other critical systems.

References

[1] C. Aguayo Gonzalez and J. Reed, Dynamic power consumption monitoring in SDR and CR regulatory compliance, *Proceedings of the Software Defined Radio Technical Conference and Product Exposition*, 2009.

[2] C. Aguayo Gonzalez and J. Reed, Power fingerprinting in SDR and CR integrity assessment, *Proceedings of the IEEE Military Communications Conference*, 2009.

[3] C. Aguayo Gonzalez and J. Reed, Detecting unauthorized software execution in SDR using power fingerprinting, *Proceedings of the IEEE Military Communications Conference*, pp. 2211–2216, 2010.

[4] C. Aguayo Gonzalez and J. Reed, Power fingerprinting in unauthorized software execution detection for SDR regulatory compliance, *Proceedings of the Software Defined Radio Technical Conference and Product Exposition*, pp. 689–694, 2010.

[5] C. Aguayo Gonzalez and J. Reed, Power fingerprinting in SDR integrity assessment for security and regulatory compliance, *Analog Integrated Circuits and Signal Processing*, vol. 69(2-3), pp. 307–327, 2011.

[6] S. Axelsson, The base-rate fallacy and the difficulty of intrusion detection, *ACM Transactions on Information and System Security*, vol. 3(3), pp. 186–205, 2000.

[7] A. Bose, X. Hu, K. Shin and T. Park, Behavioral detection of malware on mobile handsets, *Proceedings of the Sixth International Conference on Mobile Systems, Applications and Services*, pp. 225–238, 2008.

[8] A. Bose and K. Shin, On mobile viruses exploiting messaging and Bluetooth services, *Proceedings of the Second International Conference on Security and Privacy in Communications Networks and the Securecomm Workshops*, 2006.

[9] M. Christodorescu, S. Jha, S. Seshia, D. Song and R. Bryant, Semantics-aware malware detection, *Proceedings of the IEEE Symposium on Security and Privacy*, pp. 32–46, 2005.

[10] A. Cui and S. Stolfo, Defending embedded systems with software symbiotes, *Proceedings of the Fourteenth International Symposium on Recent Advances in Intrusion Detection*, pp. 358–377, 2011.

[11] S. Das, K. Kant and N. Zhang, *Handbook on Securing Cyber-Physical Critical Infrastructure: Foundations and Challenges*, Morgan Kaufmann, Waltham, Massachusetts, 2012.

[12] H. Erbacher and S. Hutchinson, Distributed sensor objects for intrusion detection systems, *Proceedings of the Ninth International Conference on Information Technology*, pp. 417–424, 2012.

[13] N. Falliere, L. O'Murchu and E. Chien, W32.Stuxnet Dossier, Version 1.4, Symantec, Mountain View, California, 2011.

[14] J. Oberheide, E. Cooke and F. Jahanian, CloudAV: N-version antivirus in the network cloud, *Proceedings of the Seventeenth USENIX Security Symposium*, pp. 91–106, 2008.

[15] M. Rajab, L. Ballard, N. Jagpal, P. Mavrommatis, D. Nojiri, N. Provos and L. Schmidt, Trends in Circumventing Web-Malware Detection, Google Technical Report rajab-2011a, Google, Mountain View, California, 2011.

[16] J. Reeves, A. Ramaswamy, M. Locasto, S. Bratus and S. Smith, Lightweight intrusion detection for resource-constrained embedded control systems, in *Critical Infrastructure Protection V*, J. Butts and S. Shenoi (Eds.), Springer, Heidelberg, Germany, pp. 31–46, 2011.

[17] S. Stone, Radio-Frequency-Based Programmable Logic Controller Anomaly Detection, Ph.D. Dissertation, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, 2013.

[18] S. Stone and M. Temple, Radio-frequency-based anomaly detection for programmable logic controllers in the critical infrastructure, *International Journal of Critical Infrastructure Protection*, vol. 5(2), pp. 66–73, 2012.

[19] C. Tankard, Advanced persistent threats and how to monitor and deter them, *Network Security*, vol. 2011(8), pp. 16–19, 2011.