On the Feasibility of Device Fingerprinting in Industrial Control Systems

Conference Paper · September 2013

DOI: 10.1007/978-3-319-03964-0_14

4 authors, including:

Marco Caselli
Siemens
20 PUBLICATIONS **303** CITATIONS

SEE PROFILE

On the Feasibility of Device Fingerprinting in Industrial Control Systems

Marco Caselli[1], Dina Hadžiosmanović[1],
Emmanuele Zambon[1], and Frank Kargl[1 2]

[1] Distributed and Embedded Security Group, University of Twente, the Netherlands,
{m.caselli, d.hadziosmanovic, e.zambon, f.kargl}@utwente.nl
[2] University of Ulm, Germany,
frank.kargl@uni-ulm.de

Abstract. As Industrial Control Systems (ICS) and standard IT networks are becoming one heterogeneous entity, there has been an increasing effort in adjusting common security tools and methodologies to fit the industrial environment. Fingerprinting of industrial devices is still an unexplored research field. In this paper we provide an overview of standard device fingerprinting techniques and an assessment on the application feasibility in ICS infrastructures. We identify challenges that fingerprinting has to face and mechanisms to be used to obtain reliable results. Finally, we provide guidelines for implementing reliable ICS fingerprinters.

Keywords: Fingerprinting, Critical Infrastructure, ICS, SCADA, PLC

1 Introduction

Power plants and industrial facilities have used industrial control systems for a long time and, until few decades ago, ICS has not significantly changed its architecture and protocols. In the last decade, ICS infrastructure increasingly opened up to standard IT networks. The advantage of this decision is twofold. First, a complex infrastructure previously running only on-site became easily accessible from remote premises by engineers. Second, the possibility to use the TCP/IP protocol suite and COTS (Commercial Off-the-Shelf) components reduces design time and costs [1]. However, this change has resulted in an increase of cyber-threats for ICS infrastructures. The situation endangers both the security of the ICS system and of the supervised physical process [1] [2]. For example, Stuxent and Flame malwares exploit Windows vulnerabilities to damage or steal information from industrial systems.

ICS and standard IT networks nowadays resemble more as they are using common protocols and devices. Researchers have been working on adapting security methodologies and tools previously used only in standard IT. Some of the applications include: firewalls, IDS (Intrusion Detection Systems), IPS (Intrusion Prevention Systems) as well as forensics, system discovery and vulnerability assessment applications.

In the IT field, *fingerprinting* is a set of activities for automatic system discovery. Fingerprinting tools exploit different information to identify devices, software and processes inside a computer network. Fingerprinting is an important building block of many security related activities and it is often used in unknown IT environments. For example, penetration testing methodologies state that system discovery, including device fingerprinting, is often the first step towards vulnerability identification [3]. Moreover, fingerprinting is used to support security systems in order to increase the accuracy of the assessments (i.e. together with intrusion prevention or detection systems).

For the ICS context, a reliable technique to recognize devices can be useful to improve industrial penetration testing techniques and security checks. Fingerprinting tools can be used to measure the level of knowledge about ICS components that can be obtained by the attackers from the network. Moreover, ICS operators can use fingerprinting to check their network verifying the absence of deviations from its standard configuration.

Problem Fingerprinting techniques are not commonly supported in ICS environments. Comprehensive studies on ICS fingerprinting still do not exist. This is mainly because ICS environments experience different operational conditions compared from traditional IT networks (e.g. operate on proprietary protocols, specific embedded devices, etc.).

Contribution In this paper we analyze challenges and opportunities for performing network-based device fingerprinting in ICS. In particular, our contributions are:

1. we perform a comprehensive analysis of traditional fingerprinting techniques to identify steps that are common to all approaches.
2. we analyze the applicability of common fingerprinting techniques in the ICS domain.
3. we build a reference model that highlights which ICS features can be used to build reliable network fingerprints.

2 Background

In this section we present concepts and terminology that will be used in the remaining of the paper.

2.1 ICS Overview

ICS is a term generally used to indicate several types of control systems used in industrial production and aimed at monitoring and controlling physical processes. ICSs include "Distributed Control Systems" (DCS) and "Supervisory Control And Data Acquisition" (SCADA). DCSs are locally deployed systems with the only purpose of gathering information and presenting it to control engineers. On the other hand, SCADA systems are usually extended to geographical

scale and collect data from different locations implementing complex mechanisms of process control.

ICS networks are often divided in two main sub-networks: the *Field Network* and the *Process Network*. The first hosts devices close to the physical process. As outlined in [4], such devices are: Sensors, Actuators, Programmable Logic Controllers (PLCs) and Remote Terminal Units (RTUs).

The Process Network contains the servers used to manage industrial processes. According to [5], it typically hosts: SCADA Servers, Distributed Control Servers (DCS Servers), the Human Machine Interface (HMI), the Engineer's Workstation, Historian Servers.

Industry and critical infrastructures have used serial communications for years. Today, TCP and IP are becoming increasingly important and involved in ICSs. This change has three main advantages both at business and network management level. First, the two systems can share network infrastructures reducing costs for communication lines and exploit cheaper TCP/IP-based components. Second, common elements such as network, database, and security can be managed by the same trained experts [6]. Finally, there is a much easier communication and information flow between corporate offices and plants personnel [7]. On the other hand, there are also disadvantages. Several IT threats now affect also ICS systems. Stuxnet and Flame are two main examples. Both malwares exploit Windows vulnerabilities to attack or steal information from industrial systems. More generally, works as [8] identify how Internet worms are now concrete threats for ICSs.

2.2 Device Fingerprinting

Device fingerprinting is a set of activities aimed at describing hardware and software components on a computer network. There are two type of fingerprinting techniques: active and passive. Active fingerprinting actively queries the system to obtain a set of information required to define the fingerprint. Passive fingerprinting acquires information in a less intrusive way by only observing the existing communication. An active fingerprinter can collect all the information necessary for describing a fingerprint therefore, this approach has higher chances to succeed.

There are two main scenarios in which fingerprinting plays an important role: penetration testing and infrastructure protection. Penetration testing methodologies always present such activities as the first step towards vulnerability identification [3] (e.g. tools like Nmap [9], P0f [10], or XProbe2++ [11] are widely used for this purpose). Also Nessus [12] implements several fingerprinting methods. Fingerprinting tools provide different information regarding operating systems and applications running on network hosts (e.g. OS and services versions). The analysis allows pentesters to organize and tune following attacks. The literature proposes also to use fingerprinting together with common security systems (e.g. IPSs, IDSs, and spam filters) to improve their effectiveness and accuracy. For example, CISCO IPSs can leverage passive OS fingerprinting and OS mappings on a host to determine the relevance of a specific attack signature [13].

Within an IDS, fingerprinting could be an additional method to resolve cases where the alerted anomaly has a high probability to be a false positive. The approach described in [14] uses the passively detected OS fingerprint of the end host to correctly resolve ambiguities between different network stack implementations. In this work, the authors provide to IDS sensors additional knowledge about the network stack implementation of the end host and thus improve the chances of detecting the attacker. Finally, fingerprinting techniques are used to mitigate specific sets of cyber-attacks. This is the case of the work described in [15]. In their paper, the authors propose a preliminary architecture that applies spam detection filtering at the router-level using light-weight signatures for spam senders. Such signatures can be used to identify spamming hosts based on the specific operating system and version from which the email is sent.

The most widely adopted fingerprinting technique uses the following information of the TCP/IP protocol headers: *initial packet size*, *initial TTL*, *windows size*, *max segment size*, *windows scaling value*, *"don't fragment" flag*, *"sackOK" flag*, and *"nop" flag*. Together these fields form a 67-bits signature capable to identify an operating system working in a standard network. Common operating systems implement TCP/IP protocols with, at least, a difference in one of the characteristics listed above. Fingerprinting tools leverage this information to identify the specific operating system.

A large part of the research on TCP/IP stack fingerprinting focuses on optimizing the standard methodology. This is usually performed by refining stored information about connection negotiation and similar mechanisms implemented by the TCP/IP protocols [16]. In [14] the authors succeed in increasing the level of confidence in TCP/IP standard fingerprinting by looking at several specific characteristic of protocol implementations. The work is mostly focused on the SYN/SYN-ACK phase of the TCP three-way handshake. Several authors present works on exploiting implementation differences of other protocols to define fingerprints. A practical work on this topic is [17] in which application-level features of protocols like HTTP, FTP, SMTP are used to improve the recognition of software installed on standard PCs. This is the approach used by Shodan [18]. Shodan is search engine that, instead of indexing web page content, analyzes banner information. This feature allows it to scan the Internet and identify machines with specific characteristics. On the other hand, there are several works that try to take advantage of hardware properties to define suitable signatures for IT systems. In [19] authors show that Ethernet devices can be uniquely identified and tracked using as few as 25 Ethernet frames. An alternative approach implies recording small deviations called clock skews in devices hardware [20]. Works like [21] explain in detail how it is possible to build a highly-reliable classification technique based on packet payload inspection. In [22] authors present an operating system detection method based on temporal response analysis. Moreover, there are fingerprinting approaches based on port scanning. Some works as [23] exploits the use of standard network ports by known services to recognize systems and applications. Finally, the creation and maintenance of the fingerprints is one of the major problem of fingerprinting. Machine learning is one of

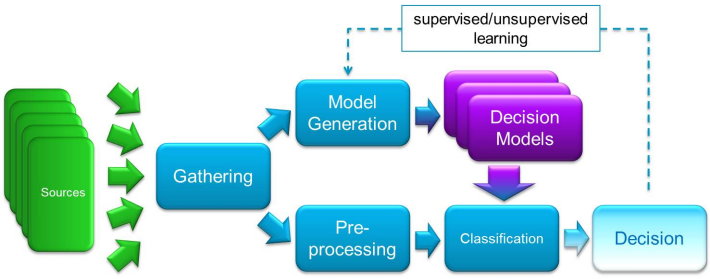

Fig. 1. Fingerprinting tools' reference architecture

the solution used to improve these two activities. For example, in [24] the authors propose probabilistic learning to develop a Naïve Bayesian classifier that passively infers a host operating system from packet headers.

3 A Reference Model for Device Fingerprinting

Despite different purposes and information sources, device fingerprinting activities have several common tasks. Due to this reason, it is possible to summarize such activities into few logical modules (or components) which can be found in all fingerprinting tools. A reference model allows to abstract details from specific solutions and makes them easier to compare. Each component of the model receives an input from the previous component, elaborates the provided data and forward the information. Our proposal for a fingerprinting reference architecture is provided in Fig. 1.

Every fingerprinting tool depends on one or more information *Sources*. A source can be uniform or heterogeneous, depending on the target of the analysis or the overall tool complexity. As already described in Sec. 2.2, fingerprinters usually exploit TCP/IP protocol information. Furthermore, some tools try to fingerprint different levels of the ISO/OSI stack (e.g. application headers, Ethernet headers, etc.). Finally, network topologies, time patterns, or also network ports usage can be potentially valuable sources as well.

The *Gathering* module implements the techniques used to collect the required data. This module relies directly on the information source and works capturing the valuable data and discarding the useless ones. Gathering module implementations differ according to the choice to do active or passive fingerprinting. During the active fingerprinting, the Gathering module manages the probing activity of the tool (i.e. by creating suitable packets and sending them to a device in order to study the responses). This is the case of Nmap and XProbe2++. The information collected, once labeled and organized, will form the dataset used by the tool for the analyses. Not all the traffic always contains valuable sources of information. Because of this, the Gathering module has to filter out useless or unknown communication and bad traffic.

The *Model Generation* performs data organization and storing. Data provided by the Gathering module is sometimes labeled by humans. For example, in the device fingerprinting training, fingerprints captured by tools are linked to information regarding the system that has generated them. Also, some tools automatically refine the dataset using machine learning techniques as in [17], [24]. Models generated by fingerprinting tools must be consistent and unambiguous. Exploiting few characteristics in generating models can cause overlaps in the dataset. Avoiding identical signatures or ambiguous data structures is the key element for fingerprinting reliability. For example, P0f provides a command for a signature collision check.

The *Decision Models* are the outputs of the Model Generation phase. The models represent the knowledge of a fingerprinter. A fingerprinter uses such knowledge as an input in the classification process. Usually, models are organized in signatures and stored in files. Both Nmap and P0f have a signature for each operating system. More advanced tools exploit different kinds of signature at the same time. This is the case of XProbe2++ and SinFP. The former has a variable number of signature items per OS, related to several tests it can perform [11]. The latter exploits two different datasets keeping active and passive signatures separated [25].

The *Pre-processing* component organizes information to be a suitable input for the analysis. Most of the times, fingerprinters exploit the Model Generation module to create temporary Decision Model of the system under examination. In few cases, such as XProbe2++, the Pre-processing component selects one by one the information needed for the comparison.

The *Classification* module implements the analysis of the collected information with respect to the dataset. This is the last component of the architecture. The classification may imply different kinds of actions. For example, P0f performs simple comparisons among signatures. Nmap and XProbe2++ exploit different methods like fuzzy signature matching. Moreover, XProbe2++ refines its classification by using decision trees. Most advanced tools add awareness about fingerprinting mitigation techniques to Classification modules. This is used to detect information artificially manipulated against fingerprinting. In particular, a software called "scrubber" is often used to confuse fingerprinting. As explained in [26] a scrubber is an active mechanism that converts heterogeneous network flows into well-behaved flows that cannot be equivocally interpreted. XProbe2++ implements systems to avoid such problem [11].

Finally, the *Decision* made by the Classification module gives the input for the manual or automatic refinement process of the dataset. Numerous tools calculate percentages of uncertainty for the provided results.

4 Fingerprinting Applicability on ICS

4.1 Overview

Fingerprinting a standard LAN works for a number of reasons. First of all, these networks often contain similar devices (e.g. personal computers) running com-

mon operating systems. Such software is widely studied and all fingerprinting tools provide comprehensive fingerprints about it. Communications usually implement open protocols like HTTP, SMTP, etc. Some of these already provide information about the systems they are working on. Moreover, PCs working on a network continuously open and close communications making connection setup information (SYN, SYN-ACK and RST packets) largely available.

To the best of our knowledge, there are only few examples of applying standard techniques to the industrial environment. This is because, there are few applicable solutions. In [27], authors argue that using already available information (as opposed to actively querying for it) is always preferable to minimize any risk to interfere with industrial operations. On the other hand, existing passive fingerprinting tools rely on the presence of Decision Models with already-known system and device fingerprints. None of such fingerprinting tools today provides such Decision Models (e.g. signatures of industrial devices) per se.

To the best of our knowledge, there is still no evidence that standard TCP/IP fingerprinting methodologies would work also for PLCs and RTUs.

4.2 Tests

We conducted preliminary tests to study how standard fingerprinting tools behave in an ICS environment. The tests involved different kinds of tools and PLCs (Siemens SIMATIC S7-1200 and ABB 800M). Our tests show that passive tool P0f working on real PCAP traces recognized only standard components like windows machines (SCADA servers or HMI) but was not able to provide any information about PLCs. Even probing devices, Nmap, Xprobe2++ and SinFP were not able to correctly identify any industrial device. Exploiting Siemens PLC's web server running on port 80, Nmap succeeded in recognizing its version and labeling it as "Siemens Simatic S7-1200 PLC httpd". Despite that, the tool fingerprinted the device as a QEMU showing its limitations in solving ambiguities related to industrial devices. It is worth noting that these tests were also useful to evaluate how feasible standard signatures' structures are for ICSs. For example, we set up a signature for the Siemens PLC and we added it to the Xprobe2++ database. Despite this new information, the tool was not able to recognize the same device. Further analyses showed that the field "icmp_echo_tos_bits" was filled in randomly by the PLC invalidating the signature enough to be discarded in favor of a different one (in this specific case an "HP JetDirect" printer).

4.3 IT/ICS Comparison

The tests were useful to identify several characteristics of ICSs that makes device fingerprinting more challenging compared to regular Internet or company LAN settings.

- **Device heterogeneity** while standard fingerprinting mostly focuses on PCs and their operating systems, in an industrial environment we deal with many different embedded devices.

- **Proprietary protocol** related to device heterogeneity, it makes difficult to exploit application layer information.
- **Device computational power** industrial devices do not have the capabilities to deal with massive network traffic and can not set up as many connections as standard PCs. This makes it difficult to use active fingerprinting.
- **Long-running TCP sessions** standard fingerprinting mostly use specific network packets (e.g. SYN, SYN-ACK, etc.). Typically, once a PLC and a control server set up a connection, the TCP session remains open for a very long time (days or even weeks) making it impossible to see those packets.

On the other hand, there are a few characteristics that fingerprinting tools can exploit in industrial environments.

- **Long life-cycle of devices** devices' working period without updates is usually long enough to guarantee a fingerprint to remain valid for years
- **Predictable behavior of components and Stable topology** once established, the process control remains the same and components behave accordingly by continuously performing stable instructions and communications [28]. This property can be exploited to create signatures based on traffic and communications patterns.
- **Protocol specification** some ICS protocols, like Modbus and Profinet, provide a way to query components in order to have information about their hardware and software.

It is worth noting that exploiting protocol specifications to query a device still has to deal with its computational power constraints. Also in this case, there is no guarantee to not interfere with process control.

5 ICS Fingerprinting based on the Reference Model

To the best of our knowledge, there are only few examples of ICS fingerprinters. PLCscan and Modbuspatrol [29] exploit the "Read Device Identification" function in Modbus allowing users to query a device for information. However, the tools are biased to a specific protocol, exploiting the last characteristic described in Sec. 4. Furthermore, Shodan has been proven to be effective with ICS systems [30]. Several banners used by PLCs contains special keywords (e.g. brand names) that often allow an easy recognition. Standard tools provide still insufficient support for ICS environments (e.g. Nmap provides only a few scripts to be used with Siemens components and Shodan works just with well-known application protocols).

A comprehensive approach to the development of ICS fingerprinters has to face all the challenges outlined in Sec. 4. For this reason, we use the reference architecture defined in Sec. 3 to structure the discussion about ICS fingerprinting and to describe the properties required for an industrial environment.

Sources: In ICS environments, there is no guarantee to see any information useful to exploit standard TCP/IP signatures due to long TCP sessions and consequently few useful packets. The only way to avoid such constraint is to set up a new connection with the target. Application layer protocols might show useful fingerprinting data especially if we deal with protocols that already provide instruments to facilitate fingerprinting (e.g. Modbus and Profinet). In several cases there still is the problem of unknown protocols. We believe that using temporal, traffic and communication patterns can be a viable solution to implement ICS fingerprinting. The use of this information does not suffer from any of the problems listed before. Moreover, we suggest to exploit the substantial stability and regularity of ICS networks' communication to identify component roles inside the system.

Gathering: with respect to standard fingerprinting we propose to reverse the balance in using active and passive techniques. We argue to exploit passive fingerprinting more than active to avoid any interference with the system under analysis. ICS often monitor or control processes in systems where a component failure may have disastrous consequences (or may be otherwise very undesirable). Because of this, a generic active probing of systems (like scanning for open ports and then opening arbitrary TCP connections), may have undesired consequences, such as network delays or unexpected component behaviors [31]. However, we know that PLCScan and Modbuspatrol actively query devices for information. It is worth noting that, in this case, Modbus provides the function to perform such query thus it is unlikely to cause problems to the infrastructure. Most preferably, the sniffer used to capture network traffic has to be transparent to ICS components. without injecting any kind of traffic in the network and sending responses to any incoming message. This often guarantees no interferences with ICS operations. Collected information regarding traffic flows has a strong constraint as it relies on the position the sniffer has in the network. For instance, two traffic captures taken from the Field Network and the Process Network are hardly comparable. Thus, the Gathering module has to provide some general information about its position and accordingly label captured data. Fingerprinting based on packets' information does not suffer such problem and allows a simpler and more performing implementation of the module.

Model Generation: signatures are the most widely adopted structures to organize fingerprinting information. However, creating signatures (e.g. the standard TCP/IP) works well only with a fingerprinting methodology that relies on several precise properties. In Sec 4.3 we show that this is not always possible in ICS environments. Querying devices for information usually provides a structured and detailed list of data that does not need any further specification. In other cases, fingerprinters involving temporal, traffic and communication patterns deal with one comprehensive set of characteristics about an ICS infrastructure that cannot be reduced to a simple signature. For example, we may want to describe a communication between two ICS components with respect to the behavior that other devices have in the same network. In this case, the purpose of our analysis is to spot the differences that make such communication unique in an

ICS system (e.g. amount of packets sent, transferred bytes, etc.) and look for the same behavior in another networks. To extract and store this information we need a comprehensive data structure that outlines architecture, properties, and trends of an ICS infrastructure.

For this reason, a *Decision Model* can be either a simple signature or a more complex set of heterogeneous information. In the second case data can be related to both the component and the system within it is deployed.

Pre-processing: after the gathering phase, communication information undergoes a further refinement process. This process depends on the structure of the Decision Model and on the classification algorithms used by the ICS fingerprinter.

Classification: without precise signatures it is difficult to make ICS fingerprinters deal with operating systems and services profiling. When ICS protocols provide a way to query devices for information, a comprehensive fingerprinting analysis is possible. In this case, we argue that the primary target of ICS fingerprinting is to recognize component's vendor, hardware (e.g. device model), and software. If it is not possible to query the device for that data, the information we can obtain is usually not complete enough to detail the component. Consequently, other possible targets in ICS fingerprinting are: component type identification (e.g differentiate between SCADA servers and PLCs), component role identification (e.g. differentiate between main PLCs and normal PLCs), network topology identification (e.g. differentiate situations in which PLCs communicate only with a SCADA server or schemas in which PLC coordinate with each other), and gathering general information about the process (e.g. ICS working on energy systems perform updates and send messages often than in water infrastructures). We can achieve the first three targets by looking at communication patterns while the last one can be the result of a temporal analysis on the observed traffic.

Decision: the Decision is the output of the ICS fingerprinter. Depending on the exploited information or the complexity of the Decision Model it can be difficult to reliably update the dataset in an automatic way. Adding unverified information into the Decision Models increases the risk of false positive and break the integrity of the dataset. However, due the heterogeneity of ICS environments, the amount of information owned is a key element toward finding matches to unknown devices. Storing new Decision Models and keeping them distinct from the original dataset can be a solution. This method allows the fingerprinter to use new models only in specific cases (e.g. solving ambiguities if the main dataset does not give reliable results).

6 Conclusions and Future Works

In this paper we analyzed the concept of device fingerprinting for ICS networks and discussed feasibility and requirements fingerprinters need to work in an industrial environment.

We started our research testing widely used fingerprinting tools with industrial devices. These tests showed that current tools are not yet tuned on such components. Consequently, our study focused on understanding how we can modify already in place fingerprinting schemes and methodologies to be effective with ICS.

The value of this work is twofold. First, we analyzed specific features of ICS systems and the challenges they pose to traditional fingerprinting. This study describes main differences and analogies with IT networks and lays the foundation for further comparative analyses. Second, we created a fingerprinting reference model by generalizing the operations performed by state of the art fingerprinting techniques. This reference model allowed us to propose and organize a guideline for the development of ICS-specific fingerprinting techniques.

We are currently working on the implementation of a proof-of-concept tool, based on the proposed reference architecture. Such fingerprinter will analyze communication patterns and will exploit a SCADA/ICS Context Model to elaborate and use information about traffic flows. This tool will take into account challenges imposed by the working environments and will implement a way to identify components types and roles inside an ICS infrastructure.

Acknowledgement This work was conceived within the "CRitical Infrastructure Security AnaLysIS" (CRISALIS) FP7 European project [32]. CRISALIS aims at providing new means to secure critical infrastructure environments from targeted attacks, carried out by resourceful and motivated individuals.

References

1. R. Robles, M. Choi, E. Cho, S. Kim, G. Park, and S. Yeo, "Vulnerabilities in SCADA and critical infrastructure systems," *International J. of Future Generation and Networking*, 2008.
2. C. Ten, C. Liu, and G. Manimaran, "Vulnerability assessment of cybersecurity for SCADA systems," *Power Systems, IEEE Trans.*, 2008.
3. C. Pfleeger, S. Pfleeger, and M. Theofanos, "A methodology for penetration testing," *Computers & Security*, 1989.
4. M. Endi, Y. Elhalwagy, and A. Hashad, "Three-layer PLC/SCADA system architecture in process automation and data monitoring," in *Computer and Automation Engineering (ICCAE)*. IEEE, 2010.
5. I. N. Fovino, A. Coletta, and M. Masera, "Taxonomy of security solutions for the SCADA sector," 2010.
6. R. Clark, S. Hakim, and A. Ostfeld, *Handbook of Water and Wastewater Systems Protection*. Springer, 2011.
7. R. McClanahan, "The benefits of networked SCADA systems utilizing IP-enabled networks," in *Rural Electric Power Conference*. IEEE, 2002.
8. K. Munro, "Scada - a critical situation," *Network Security*, 2008.
9. G. Lyon, "Nmap security scanner," Feb. 2013. [Online]. Available: http://nmap.org/
10. M. Zalewski, "p0f: Passive OS fingerprinting tool," 2006. [Online]. Available: 2002-02-01). http://lcamtuf.coredump.cx/p0f.shtml

11. F. Yarochkin, O. Arkin, M. Kydyraliev, S. Dai, Y. Huang, and S. Kuo, "Xprobe2++: Low volume remote network information gathering tool," in *Dependable Systems & Networks, 2009. DSN'09. IEEE/IFIP*.
12. R. Deraison, H. Meer, and C. V. D. Walt, *Nessus network auditing*. Syngress Media Incorporated, 2004.
13. C. S. Inc., "User guide for Cisco security manager 4.3," 2012.
14. G. Taleck, "Ambiguity resolution via passive os fingerprinting," in *Recent Advances in Intrusion Detection*. Springer, 2003.
15. H. Esquivel, T. Mori, and A. Akella, "Router-level spam filtering using TCP fingerprints: Architecture and measurement-based evaluation," in *Proceedings of the Sixth Conference on Email and Anti-Spam*, 2009.
16. V. Paxson, "Automated packet trace analysis of TCP implementations," in *ACM SIGCOMM Computer Communication Review*, 1997.
17. P. Haffner, S. Sen, O. Spatscheck, and D. Wang, "ACAS: automated construction of application signatures," in *Proceedings of the 2005 ACM SIGCOMM workshop on Mining network data*.
18. J. Matherly, "Expose online devices," May 2013. [Online]. Available: http://www.shodanhq.com/
19. R. Gerdes, T. Daniels, M. Mina, and S. Russell, "Device identification via analog signal fingerprinting: A matched filter approach," in *Network and Distributed System Security Symposium (NDSS)*, 2006.
20. T. Kohno, A. Broido, and K. Claffy, "Remote physical device fingerprinting," *Dependable and Secure Computing, IEEE Trans.*, 2005.
21. A. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," *Passive and Active Network Measurement*, 2005.
22. F. Veysset, O. Courtay, and O. Heen, "New tool and technique for remote operating system fingerprinting," *Intranode Software Technologies*, 2002.
23. D. Moore, K. Keys, R. Koga, E. Lagache, and K. Claffy, "The coralreef software suite as a tool for system and network administrators," in *Proceedings of the 15th USENIX conference on System administration*, 2001.
24. R. Beverly, "A robust classifier for passive TCP/IP fingerprinting," *Passive and Active Network Measurement*, 2004.
25. P. Auffret, "Sinfp, unification of active and passive operating system fingerprinting," *Journal in computer virology*, 2010.
26. D. Watson, M. Smart, G. Malan, and F. Jahanian, "Protocol scrubbing: network security through transparent flow modification," *IEEE/ACM Trans. on Networking (TON)*, 2004.
27. A. Mahmood, C. Leckie, J. Hu, Z. Tari, and M. Atiquzzaman, "Network traffic analysis and SCADA security," *Handbook of Information and Communication Security*, 2010.
28. D. Hadziosmanovic, D. Bolzoni, S. Etalle, and P. Hartel, "Challenges and opportunities in securing industrial control systems," in *Proceedings of the IEEE Workshop on Complexity in Engineering, COMPENG 2012, Aachen, Germany*.
29. S. Gordeychik, "SCADA strangelove or: How i learned to start worrying and love nuclear plants," Feb. 2013.
30. ICS-CERT, "ICS-ALERT-11-343-01 Control System Internet Accessibility," U.S. Department of Homeland Security, Dec 2011.
31. D. Duggan, M. Berg, J. Dillinger, and J. Stamp, "Penetration testing of industrial control systems," *Sandia National Laboratories*, 2005.
32. "CRitical Infrastructure Security AnaLysIS (CRISALIS)," 2012. [Online]. Available: http://www.crisalis-project.eu/