# The Cybersecurity Landscape in Industrial Control Systems

*This paper surveys the state of the art in industrial control system (ICS) security, identifies outstanding research challenges in this emerging area, and explains the key concepts and principles for deployment of cybersecurity methods and tools to ICSs.*

By Stephen McLaughlin, Charalambos Konstantinou, Xueyang Wang, Lucas Davi, Ahmad-Reza Sadeghi, Michail Maniatakos, and Ramesh Karri

**ABSTRACT** | Industrial control systems (ICSs) are transitioning from legacy-electromechanical-based systems to modern information and communication technology (ICT)-based systems creating a close coupling between cyber and physical components. In this paper, we explore the ICS cybersecurity landscape including: 1) the key principles and unique aspects of ICS operation; 2) a brief history of cyberattacks on ICS; 3) an overview of ICS security assessment; 4) a survey of "uniquely-ICS" testbeds that capture the interactions between the various layers of an ICS; and 5) current trends in ICS attacks and defenses.

**KEYWORDS** | Computer security; industrial control; networked control systems; power system security; SCADA systems; security

## I. INTRODUCTION

Modern industrial control systems (ICSs) use information and communication technologies (ICTs) to control and automate stable operation of industrial processes [1], [2]. ICSs interconnect, monitor, and control processes in a variety of industries such as electric power generation, transmission and distribution, chemical production, oil and gas, refining and water desalination. The security of ICSs is receiving attention due to its increasing connections to the Internet [3]. ICS security vulnerabilities can be attributed to several factors: use of microprocessor-based controllers, adoption of communication standards and protocols, and the complex distributed network architectures. The security of ICSs has come under particular scrutiny owing to attacks on critical infrastructures [4], [5].

Traditional IT security solutions fail to address the coupling between the cyber and physical components of an ICS [6]. According to NIST [1], ICSs differ from traditional IT systems in the following ways. 1) The primary goal of ICSs is to maintain the integrity of the industrial process. 2) ICS processes are continuous and hence need to be highly available; unexpected outages for repair must be planned and scheduled. 3) In an ICS, interactions with physical processes are central and often times complex. 4) ICSs target specific industrial processes and may not have resources for additional capabilities such as security. 5) In ICSs, timely response to human reaction and physical sensors is critical. 6) ICSs use proprietary communication protocols to control field devices. 7) ICS components are replaced infrequently (15–20 years or longer). 8) ICS components are distributed and isolated and hence difficult to physically access to repair and upgrade.

Attacks on ICSs are happening at an alarming pace and the cost of these attacks is substantial for both governments and industries [7]. Cyberattacks against oil and gas infrastructure are estimated to cost the companies \$1.87 billion by 2018 [8]. Until 2001, most of attacks originated internal to a company. Recently, attacks

Manuscript received August 31, 2015; revised November 19, 2015; accepted December 19, 2015. Date of publication March 16, 2016; date of current version April 19, 2016. This work was supported in part by German Science Foundation as part of Project S2 within the CRC 1119 CROSSING; by the European Union's Seventh Framework Programme under Grant 609611, PRACTICE project; and by the Intel Collaborative Research Institute for Secure Computing (ICRI-SC). The NYU researchers were also supported in part by Consolidated Edison, Inc., under Award 4265141; by the U.S. Office of Naval Research under Award N00014-15-1-2182; and by the NYU Center for Cyber Security (New York and Abu Dhabi).

**S. McLaughlin** is with KNOX Security, Samsung Research America, Mountain View, CA 94043 USA (e-mail: s.mclaughlin@samsung.com).
**C. Konstantinou**, **X. Wang**, and **R. Karri** are with the Polytechnic School of Engineering, New York University, Brooklyn, NY 11201 USA (e-mail: ckonstantinou@nyu.edu; rkarri@nyu.edu).
**L. Davi** and **A.-R. Sadeghi** are with Technische Universität Darmstadt, Darmstadt 64289, Germany (e-mail: lucas.davi@trust.cased.de; ahmad.sadeghi@trust.cased.de).
**M. Maniatakos** is with the Electrical and Computer Engineering Department, New York University Abu Dhabi, Abu Dhabi, UAE (e-mail: michail.maniatakos@nyu.edu).

Digital Object Identifier: 10.1109/JPROC.2015.2512235

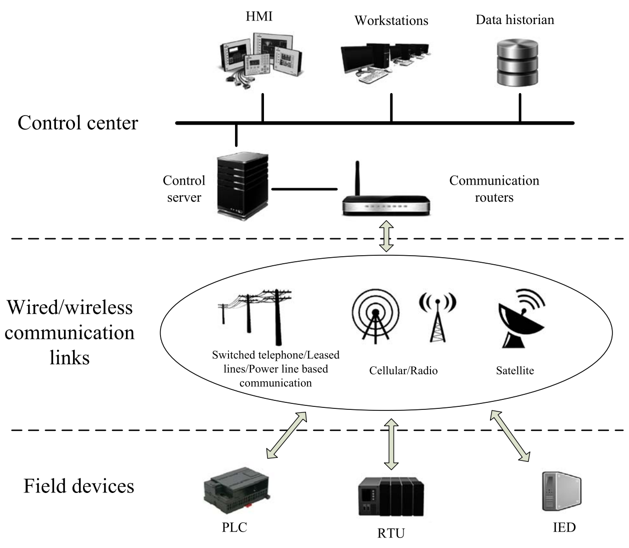


**Fig. 1.** ***General structure of an ICS. The industrial process data collected at remote sites are sent by field devices such as remote terminal units (RTUs), intelligent electronic devices (IEDs), and programmable logic controller (PLCs), to the control center through wired and wireless links. The control server allows clients to access data using standard protocols. The human–machine interface (HMI) presents processed data to a human operator, by querying the time-stamped data accumulated in the data historian. The gathered data are analyzed, and control commands are sent to remote controllers.***

external to a company are becoming frequent. This is due to the use of commercial off-the-shelf (COTS) devices, open applications and operating systems, and increasing connection of the ICS to the Internet.

In an effort to keep up with the cyberattacks, cybersecurity researchers are investigating the attack surface and defenses for critical infrastructure domains such as the smart grid [9], oil and gas [10], and water SCADA [11]. This survey will focus on the general ICS cybersecurity landscape by discussing attacks and defenses at various levels of abstraction in an ICS from the hardware to the process.

## A. Industrial Control Systems

The general architecture of an ICS is shown in Fig. 1. The main components of an ICS include the following.

- Programmable logic controller (PLC): A PLC is a digital computer used to automate industrial electromechanical processes. PLCs control the state of output devices based on the signals received from the sensors and the stored programs. PLCs operate in harsh environmental conditions, such as excessive vibration and high noise [12]. PLCs control standalone equipment and discrete manufacturing processes.
- Distributed control system (DCS): DCS is an automated control system in which the control elements are distributed throughout the system [13]. The distributed controllers are networked to remotely monitor processes. The DCS can remain operational even if a part of the control system fails. DCSs are often found in continuous and batch production processes which require advanced control and communication with intelligent field devices.
- Supervisory control and data acquisition (SCADA): SCADA is a computer system used to monitor and control industrial processes. SCADA monitors and controls field sites spread out over a geographically large area. SCADA systems gather data in real time from remote locations. Supervisory decisions are then made to adjust controls.

## B. History of ICS Attacks

In an ICS, the stable operation could be disrupted not only by an operator error or a failure at a production unit, but also by a software error/bug, malware, or an intentional cyber criminal attack [14]. Just in 2014, the ICS Cyber Emergency Response Team (ICS-CERT) responded to 245 incidents. Numerous cyberattacks on ICS are summarized in Fig. 2. We elaborate on four ICS attacks that caused physical damages.

In 2007, Idaho National Laboratory staged the Aurora attack, in order to demonstrate how a cyberattack could destroy physical components of the electric grid [15]. The attacker gained the access to the control network of a diesel generator. Then a malicious computer program was run to rapidly open and close the circuit breakers of the generator, out of phase from the rest of the grid, resulting in an explosion of the diesel generator. Since most of the grid equipment uses legacy communications protocols that did not consider security, this vulnerability is especially a concern [16].

In 2008, a pipeline in Turkey was hit by a powerful explosion spilling over 30 000 barrels of oil in an area above a water aquifer. Further, it cost British Petroleum $5 million a day in transit tariffs. The attackers entered the system by exploiting the vulnerabilities of the wireless camera communication software, and then moved deep into the internal network. The attackers tampered with the units used to alert the control room about malfunctions and leaks, and compromised PLCs at valve stations to increase pressure in the pipeline causing the explosion.

In 2010, Stuxnet computer worm infected PLCs in 14 industrial sites in Iran, including an uranium enrichment plant [4], [17]. It was introduced to the target system via an infected USB flash drive. Stuxnet then stealthily propagated through the network by infecting removable drives, copying itself in the network shared resources, and by exploiting unpatched vulnerabilities.

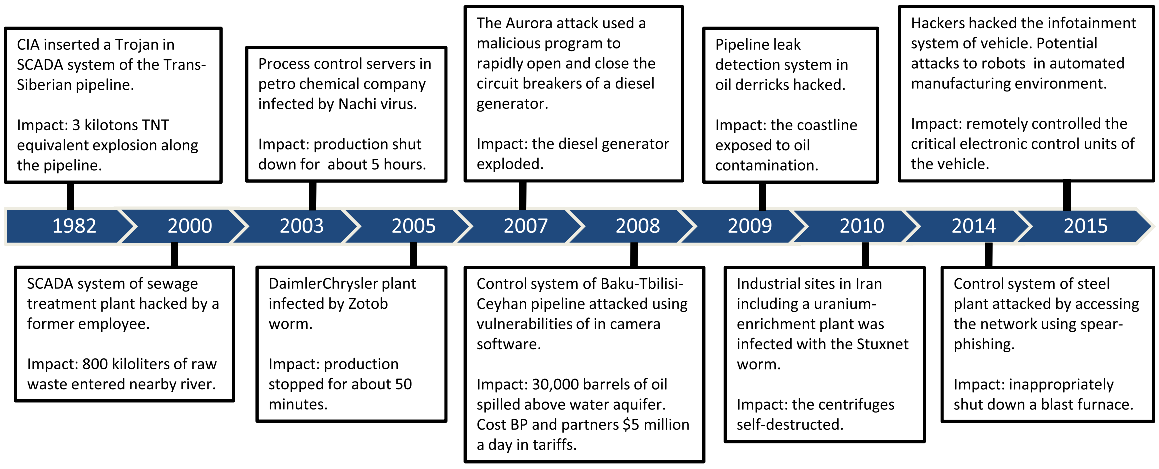

CIA inserted a Trojan in SCADA system of the Trans-Siberian pipeline.
Impact: 3 kilotons TNT equivalent explosion along the pipeline.
Process control servers in petro chemical company infected by Nachi virus.
Impact: production shut down for about 5 hours.
The Aurora attack used a malicious program to rapidly open and close the circuit breakers of a diesel generator.
Impact: the diesel generator exploded.
Pipeline leak detection system in oil derricks hacked.
Impact: the coastline exposed to oil contamination.
Hackers hacked the infotainment system of vehicle. Potential attacks to robots in automated manufacturing environment.
Impact: remotely controlled the critical electronic control units of the vehicle.
1982
2000
2003
2005
2007
2008
2009
2010
2014
2015
SCADA system of sewage treatment plant hacked by a former employee.
Impact: 800 kiloliters of raw waste entered nearby river.
DaimlerChrysler plant infected by Zotob worm.
Impact: production stopped for about 50 minutes.
Control system of Baku-Tbilisi-Ceyhan pipeline attacked using vulnerabilities of in camera software.
Impact: 30,000 barrels of oil spilled above water aquifer. Cost BP and partners $5 million a day in tariffs.
Industrial sites in Iran including a uranium-enrichment plant was infected with the Stuxnet worm.
Impact: the centrifuges self-destructed.
Control system of steel plant attacked by accessing the network using spear-phishing.
Impact: inappropriately shut down a blast furnace.

**Fig. 2.** ***Timeline of cyberattacks on ICS and their physical impacts.***

The infected computers were instructed to connect to an external command and control server. The central server then reprogrammed the PLCs to modify the operation of the centrifuges to tear themselves apart by the compromised PLCs [18].

In 2015, two hackers demonstrated a remote control of a vehicle [19]. The zero-day exploit gave the hackers wireless control of the vehicles. The software vulnerabilities in the vehicle entertainment system allowed the hackers to remotely control it, including dashboard functions, steering, brakes, and transmission, enabling malicious actions such as controlling the air conditioner and audio, disabling the engine and the brakes, and commandeering the wheel [20]. This is a harbinger of attacks in an automated manufacturing environment where intelligent robots cohabitate and coordinate with humans.

### C. Roadmap of This Paper

Cybersecurity assessment can reveal the obvious and nonobvious physical implications of ICS vulnerabilities on the target industrial processes. Cybersecurity assessment of ICSs for physical processes requires capturing the different layers of an ICS architecture. The challenges of creating a vulnerability assessment methodology are discussed in Section II. Cybersecurity assessment of an ICS requires the use of a testbed. The ICS testbed should help identify cybersecurity vulnerabilities as well as the ability of the ICS to withstand various types of attacks that exploit these vulnerabilities. In addition, the testbed should ensure that critical areas of the ICS are given adequate attention. This way one can lessen the costs for fixing cybersecurity vulnerabilities emerging from flaws in the design of ICS components and the ICS network. ICS testbeds are discussed in Section II. Discussion on how one can construct attack vectors appears in Section III. Attacks on ICSs have devastating physical consequences. Therefore, ICSs need to be designed for security robustness and tested prior to deployment. Control protocols should be fitted with security features and policies. ICSs should be reinforced by isolating critical operations by removing unnecessary services and applications from ICS components. Extensive discussion on vulnerability mitigation appears in Section IV, followed by final remarks in Section V.

## II. ICS VULNERABILITY ASSESSMENT

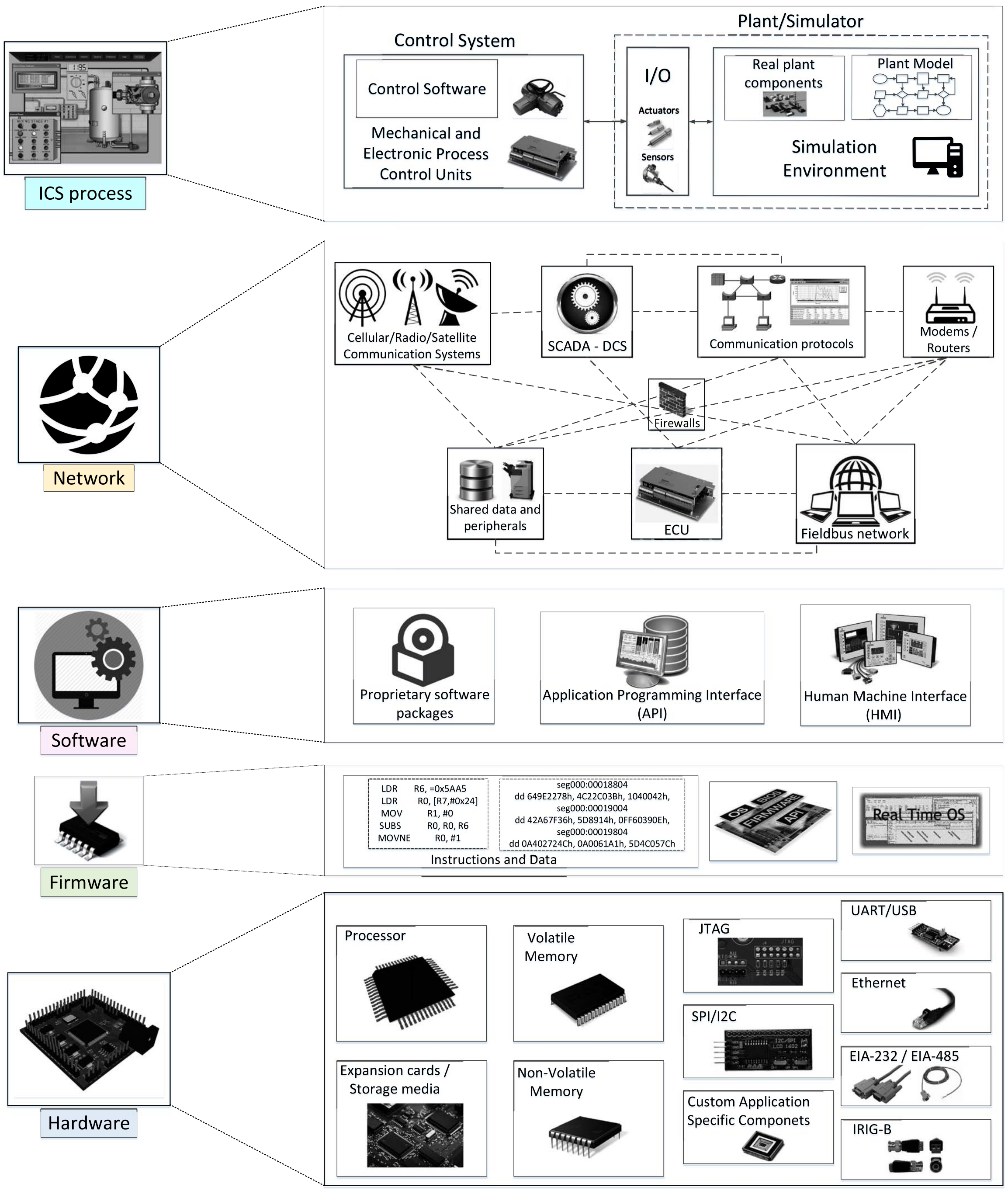
In this section, we review the different layers in an ICS, the vulnerability assessment process outlining the cybersecurity assessment strategy and discuss ICS testbeds for accurate vulnerability analyses in a lab environment.

### A. The ICS Architecture and Vulnerabilities

The different layers of ICS architecture are shown in Fig. 3.

*1) Hardware Layer:* Embedded components such as PLCs and RTUs are hardware modules executing software. Hardware attacks such as fault injection and backdoors can be introduced into these modules. These vulnerabilities in the hardware can be exploited by adversaries to gain access to stored information or to deny services.

The hardware-level vulnerabilities concern the entire lifecycle of an ICS from design to disposal. Security in the processor supply chain is a major issue since hardware trojans can be injected in any stage of the supply chain introducing potential risks such as loss of reliability and security [21], [22]. Unauthorized users can use JTAG ports—used for in-circuit test—to steal intellectual property, modify firmware, and reverse engineer logic [23]–[25]. Peripherals introduce vulnerabilities. For example, malicious USB drives can redirect communications by changing DNS settings or destroy the circuit board [26], [27]. Expansion cards, memory units,



```
LDR      R6, =0x5AA5
LDR      R0, [R7,#0x24]
MOV      R1, #0
SUBS     R0, R0, R6
MOVNE    R0, #1
```

```
seg000:00018804
dd 649E2278h, 4C22C03Bh, 1040042h,
seg000:00019004
dd 42A67F36h, 5D8914h, 0FF60390Eh,
seg000:00019804
dd 0A402724Ch, 0A0061A1h, 5D4C057Ch
```

**Fig. 3.** ***Layered ICS architecture and the vulnerable components in the ICS stack.***

and communication ports pose a security threat as well [28]–[30].

2) *Firmware Layer:* The firmware resides between the hardware and software. It includes data and instructions able to control the hardware. The functionality of firmware ranges from booting the hardware providing runtime services to loading an operating system (OS). Due to the real-time constraints related to the operation of ICSs, firmware-driven systems typically adopt a real-time operating system (RTOS) such as VxWorks. In any case, vulnerabilities within the firmware could be exploited by adversaries to abnormally affect the ICS process. A recent study exploited vulnerabilities in a wireless access point and a recloser controller firmware [31]. Malicious firmware can be distributed from a central system in an

advanced metering infrastructure (AMI) to smart meters [32]. Clearly, that vulnerabilities in firmware can be used to launch DoS attacks to disrupt the ICS operation.

*3) Software Layer:* ICSs employ a variety of software platforms and applications, and vulnerabilities in the software base may range from simple coding errors to poor implementation of access control mechanisms. According to ICS-CERT, the highest percentage of vulnerabilities in ICS products is improper input validation by ICS software, also known as the buffer overflow vulnerability [33]. Poor management of credentials and authentication weaknesses are second and third, respectively. These vulnerabilities in the implementation of software interfaces (e.g., HMI) and server configurations may have fatal consequences on the control functionality of an ICS. For instance, a proprietary industrial automation software for historian servers had a heap buffer overflow vulnerability that could potentially lead to a Stuxnet-type attack [34].

Sophisticated malware often incorporate both hardware and software. WebGL vulnerabilities are an example of hardware-enabled software attacks: access to graphics GPU hardware by a least-privileged remote party results in the exposure of GPU memory contents from previous workloads [35]. The implementation of the software layer in a HIL testbed should reflect how each added component to the ICS increases the attack surface.

*4) Network Layer:* Vulnerabilities can be introduced into the ICS network in different ways [1]: a) firewalls (that protect devices on a network by monitoring and controlling communication packets using filtering policies); b) modems (that convert between serial digital data and a signal suitable for transmission over a telephone line to allow devices to communicate); c) fieldbus network (that links sensors and other devices to a PLC or other controller); d) communications systems and routers (that transfer messages between two networks); e) remote access points (that remotely configure ICS and access process data); and f) protocols and control network (that connect the supervisory control level to lower level control modules). DCS and SCADA servers, communicating with lower level control devices, often are not configured properly and not patched systematically and hence are vulnerable to emerging threats [36].

When designing a network architecture for an ICS, one should separate the ICS network from the corporate network. In case the networks must be connected, only minimal connections should be allowed and the connection must be through a firewall and a DMZ.

*5) Process Layer:* All the aforementioned ICS layers interact to implement the target ICS processes. The observed dynamic behavior of the ICS processes must follow the dynamic process characteristics based on the designed ICS model [37]. ICS process-centric attacks may inject spurious/incorrect information (through specially crafted messages) to degrade performance or to hamper the efficiency of the controlled process [33]. Process-centric attacks may also disturb the process state (e.g., crash or halt) by modifying runtime process variables or the control logic. These attacks can deny service or change the industrial process without operator knowledge. Therefore, it is imperative to determine if variations in the system process are nominal consequences of an expected operation or signal an anomaly/attack. Process-centric/process-aware vulnerability analysis can contribute to practices that enable ICS processes to function in a secure manner. The vulnerabilities related to the information flow (e.g., dependencies on hardware/software/network equipment with a single point of failure) must be determined. The HIL testbed should properly emulate the target process, in order to effectively assess and mitigate process-centric attacks [38].

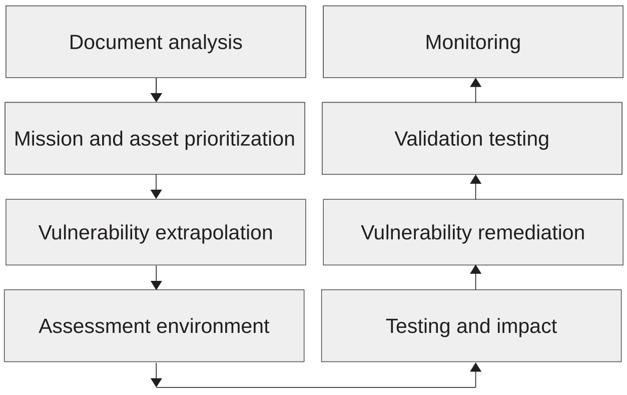


**Fig. 4.** ***Security assessment of ICS.***

## B. ICS Vulnerability Assessment

Fig. 4 presents the steps in the security assessment process whose aim is to identify security weaknesses and potential risks in ICSs. Due to the real-world consequences of ICS, security assessment of ICSs must account for all possible operating conditions of each ICS component. Additionally, since ICS equipment can be more fragile than standard IT systems, the security assessment should take into consideration the sensitive ICS dependencies and connectivity [39].

*1) Document Analysis:* The first step in assessing any ICS is to characterize the different parts of its architecture. This includes gathering and analyzing information in order to understand the behavior of each ICS component. For example, analyzing the features of IEDs used in power systems such as a relay controller entails collecting information about its communication,

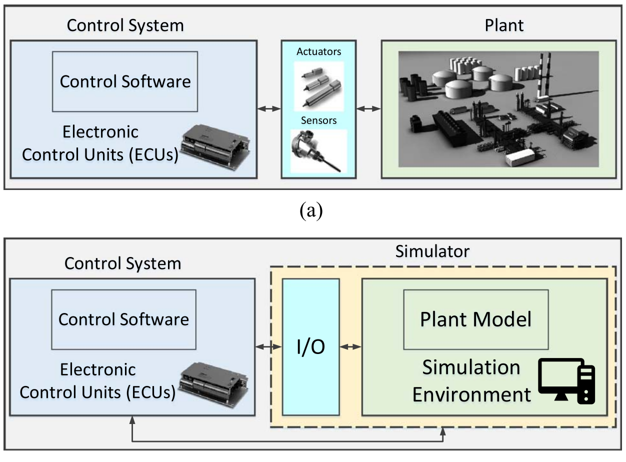


**Fig. 5.** ***(a) Real ICS environment versus (b) HIL simulation of ICSs.***

functionality, default configuration passwords, and supported protocols [40].

2) *Mission and Asset Prioritization:* Prioritizing the missions and assets of the ICS is the next step in security assessment. Resources must be allocated based on the purpose and sensitivity of each function. Demilitarized zones (DMZs), for instance, can be used to add a layer of security to the ICS network by isolating the ICS and corporate networks [41]. Selecting DMZs is an important task in this phase.

3) *Vulnerability Extrapolation:* Next the ICS should be examined for security vulnerabilities, to identify sources of vulnerability, and to establish attack vectors [42]. Design weaknesses and security vulnerabilities in critical authentication, application and communication security components should be investigated. The attack vectors should comprehensively explain the targeted components and the attack technique.

4) *Assessment Environment:* Depending on the type of industry and level of abstraction, assessment actions must be defined [37]. For example, in case when only software is used, the test vectors should address as many physical and cyber characteristics of the ICS as possible. By modeling and simulating individual ICS modules, the behavior of the system is emulated with regards to how the ICS and its internal functions react.

Due to the complexity and real-time requirements of ICSs, hardware-in-the-loop (HIL) simulation is more efficient to test system resiliency against the developed attack vectors [43]. HIL simulation adds the ICS complexity to the assessment platform by adding the control system in a loop, as shown in Fig. 5(b). To capture the system dynamics, the physical process is replaced with a simulated plant, including sensors, actuators, and machinery. A well-designed HIL simulator will mimic the actual process behavior as closely as possible. A detailed discussion of developing an assessment environment appears in Section II-C.

5) *Testing and Impact:* The ICS will be tested on the testbed to demonstrate the outcomes of the attacks including the potential effect on the physical components of the ICS [44]. In addition, the system-level response and the consequences to the overall network can be observed. The results can be used to assess the impact of a cyberattack on the ICS.

6) *Vulnerability Remediation:* Any weaknesses discovered in the previous steps should be carefully mitigated. This may involve working with vendors [45] and updating network policies [46]. If there is no practical mitigation strategy to address a vulnerability, guidelines should be developed to allow sufficient time to effectively resolve the issue.

7) *Validation Testing:* The mitigation actions designed to resolve security issues must then be tested. A critical part of this step is to reexamine the ICS and identify weaknesses.

8) *Monitoring:* Implementing all the previous steps is half the battle. Continuous monitoring and reassessing the ICS to maintain security is important [47]. Intrusion detection systems (IDSs) can assist in continuously monitoring network traffic and discover potential threats and vulnerabilities.

## C. ICS Testbeds

The assessment environment, i.e., the testbed, effects all the stages of the assessment methodology. Assessment methodologies that include the production environment or testing individual components of the ICS are not relevant. Although these methodologies are effective for IT systems, uniquely-ICS nature of using “data to manipulate physics” makes these approaches inherently hazardous. Therefore, we focus on lab-based ICS testbeds. A HIL testbed offers numerous benefits by balancing accuracy and feasibility. In addition, HIL testbeds can be used to train employees and ensure interoperability of the diverse components used in the ICS.

The cyber–physical nature of ICSs presents several challenges in the design and operation of an ICS testbed. The testbed must be able to model the complex behavior of the ICS for both operation and nonoperation conditions. It should address scaling since the testbed is a scaled down model of the actual physical ICS. Furthermore, the testbed must accurately represent the ICS in order to support the protocols and standards as well as to generate accurate data. It is also important for the testbed to capture the interaction between legacy and modern ICS. This interaction is important for both security assessment and compatibility testing of the ICS. Numerous other factors should be considered when designing an ICS testbed including flexibility, interface with IT systems, configuration settings, and testing for extreme conditions.

Assessment of ICS using software-only testbed and techniques is not frequently adopted. Software models and simulations cannot recreate real-world conditions since they include only one layer of the complex ICS architecture. Furthermore, the software models cannot include every possible cyber–physical system state of the ICS [48]. Software-only testbeds are also limited by the supported hardware. Finally, the limitations of the computational features supported by the software simulator might introduce delays, simplify assumptions, and use simple heuristics in the simulator engine (e.g., theoretical implementation of network protocols). Finally, in most cases, a software-only testbed gives the users a false sense of security regarding the accuracy of the simulation results. On the other hand, software-only assessment is advantageous in that one can study the behavior of a system without building it. Scilab and Scicos are two open-source software platforms for design, simulation, and realization of ICSs [49], [50].

It is clear that an ICS testbed requires real hardware in the simulation loop. Such HIL simulation symbiotically relates cyber and physical components [51]. A HIL testbed can simulate real-world interfaces including interoperable simulations of control infrastructures, distributed computing applications, and communication networks protocols.

1) *Security Objectives of HIL Testbeds:* The primary objective of HIL testbeds is to guide implementation of cybersecurity within ICSs. In addition, HIL testbeds are essential to determine and resolve security vulnerabilities. The individual components of an appropriate testbed should capture all the ICS layers, and the interactions are shown in Fig. 3.

Equipment and network vulnerabilities can be tested in a protected environment that can facilitate multiple types of ICS scenarios highlighting the several layers of the ICS architecture. For instance, the cybersecurity testbed developed by NIST covers several ICS application scenarios [52]. The Tennessee Eastman scenario covers the continuous process control in a chemical plant. The robotic assembly scenario covers the discrete dynamic processes with embedded control. The enclave scenario covers wide area industrial networks in an ICS such as SCADA.

2) *Benefits of a HIL Assessment Methodology:* The HIL assessment methodology has the following advantages:

- flexibility: HIL systems provide reconfigurable architectures for testing several ICS application scenarios (incorporating legacy and modern equipment);
- simulation: ICS phenomena are simulated faster than complex physical ICS events;
- accuracy: HIL simulators provide results comparable in terms of accuracy with the live ICS environment;
- repeatability: the controlled settings in the testbed increase repeatability;
- cost effectiveness: the combination of hardware HIL software reduces the implementation costs of the testbed;
- safety: HIL simulation avoids the hazards present when testing in a live ICS setting;
- comprehensiveness: it is often possible to assess ICS scenarios over a wider range of operating conditions;
- modularity: HIL testbeds facilitate linkages with other interfaces and testbeds, integrating multiple types of control components;
- network integration: protocols and standards can be evaluated creating an accurate map of networked units and their connection communication links;
- nondestructive test: destructive events can be evaluated (e.g., aurora generator test [53]) without causing damage to the real system;
- hardware security: HIL testbed allows one to study the hardware security of an ICS which has become a major concern over the past decade 7 (e.g., side-channel and firmware attacks [44]).

3) *Example ICS Testbeds:* Over 35 smart grid testbeds have been developed in the United States [54]. ENEL SPA testbed analyzes attack scenarios and their impact on power plants [55]. It includes a scaled down physical process, corporate and control networks, DMZs, PLCs, industrial standard software, etc. The Idaho National Laboratories (INL) SCADA Testbed is a large-scale testbed dedicated to ICS cybersecurity assessment, standards improvements, and training [56]. The PowerCyber testbed integrates communication protocols, industry control software, and field devices combined with virtualization platforms, real-time digital simulators (RTDSs), and ISEAGE WAN emulation in order to provide an accurate representation of cyber–physical grid interdependencies [57]. Digital Bond's Project Basecamp demonstrates the fragility and insecurity of SCADA and DCS field devices, such as PLCs and RTUs [58]. New York University (NYU) has developed a smart grid testbed to model the operation of circuit breakers and demonstrate firmware modification attacks on relay controllers [44]. Many hybrid laboratory-scale ICS testbeds exist in research centers and universities [54]. Besides laboratory-scale ICS testbeds with real equipment, many virtual testbeds are also being developed able to "create"

ICS components including virtual devices and process simulators [59].

Summarizing, given that many ICS attacks exploit vulnerabilities in one or more layers of an ICS, HIL ICS testbeds are becoming standard for security assessment, allowing development and testing of advanced security methods. Additionally, HIL ICS testbeds have been quantitatively shown to produce results close to real-world systems.

## III. ATTACKS ON ICSs

An important part of the assessment process is the identification of vulnerabilities in the ICS under test. In this section, we present the current and emerging threat landscapes for ICSs.

### A. Current ICS Threat Landscape

ICSs are vulnerable to traditional computer viruses [60]–[62], remote break-ins [63], insider attacks [64], and targeted attacks [65]. Industries affected by ICS attacks include nuclear power and refinement of fissile material [62], [65], transportation [63], [66], electric power delivery [67], manufacturing [60], building automation [64], and space exploration [61].

One class of attacks against ICS involves compromising one or more of its components using traditional attacks, e.g., memory exploits, to gain control of the systems behavior, or access sensitive data related to the process. We consider three classes of studies on ICS vulnerabilities. The first considers studies of the security and security readiness of ICS systems and their operators. The second class considers security vulnerabilities in PLCs. The third class considers vulnerabilities in sensors, in this case, focusing on smart electric meters, an important component of the smart grid infrastructure.

*1) ICS Security Posture:* There have been studies of the ICS security posture [68] and the conclusion is that there is substantial room for improvement. First, it was found that ICSs frequently rely on security through obscurity, due to their history of being proprietary systems isolated from the Internet. However, use of commodity OS (e.g., Microsoft Windows OS) and open, standard network protocols, have left ICS open not only to malicious attacks, but also to coincidental infiltration by Internet malware. For example, the slammer worm infected machines belonging to an Ohio nuclear power generation facility. These studies also showed a significant rise in ICS cybersecurity incidents; while only one incident was reported in 2000, ten incidents were reported in 2003. Penetration tests of over 100 real-world ICSs over the course of ten years, with emphasis on power control systems, corroborate these findings [69]. Besides identifying vulnerabilities throughout the ICS, the study shows that in most cases, these ICSs are at least a year behind the standard patch cycle. In some cases, the DMZ separating the ICS from the corporate network had not been updated in years leaving DoS attacks trivial. For example, in their evaluation of a network connected PLC, it was found that ping flooding with 6-kB packets was sufficient to render the PLC inoperable, causing all state to be lost and forcing it to be power cycled.

Another hurdle in improving ICS security are three commonly held myths [70]: 1) security can be achieved through obscurity; 2) blindly deploying security technologies improves security; the naive application of firewalls, cryptography, and antivirus software often leaves system operators with a false sense of security; and 3) standards compliance yields a secure system; the North-American Energy Reliability Corporations Cyber Infrastructure Protection standards [71] have been criticized for giving a false sense of security [72].

*2) Attacks on PLCs:* PLCs monitor and manipulate the state of a physical system. A popular Siemens PLC was shown to have vulnerabilities. The ISO-TSAP protocol used by these PLCs can implement a replay attack due to lack of proper session freshness [73]. It was also possible to bypass the PLC authentication, sufficient to upload payloads as described in Section III-B, and to execute arbitrary commands on the PLC. The Siemens PLCs used in correctional facilities have vulnerabilities allowing manipulation of cell doors [74].

*3) Attacks on Sensors:* Another critical element in an ICS are the sensors that gather data and relay it back to the control units. Consider smart meters that are widely a deployed element of the evolving smart electric grid [75]. A smart meter has the same form factor as a traditional analog electric meter with a number of enhanced features: time of use pricing [76], automated meter reading, power quality monitoring, and remote power disconnect. Security assessment of a real-world smart metering system considered energy theft by tampered measurement values [77]. The system under test allowed for undetectable tampering of measurement values both in the meter's persistent storage, as well as in flight, due to a replay attack against the meter-to-utility authentication scheme. A follow-up study examined meters from multiple vendors and found vulnerabilities allowing for a single-packet denial of service attack against an arbitrary meter, and full control of the remote disconnect switch enabling a targeted disconnect of the service to a customer [78].

### B. Emerging Threats

Here we introduce two new directions for attacks on ICSs. The first of these constructs payloads targeting an ICS that an adversary may not have full access to. The second class of attacks manipulate sensor inputs to misguide the decisions made by the PLCs.

1) *Payload Construction:* One type of attacks aims to gather intelligence about the victim ICS. For example, the Duqu worm seems to have gathered information about victim systems [79], before relaying it to command and control servers. The other type of attacks against ICS aims to influence the physical behavior of the victim system. Best known example of such an attack is the Stuxnet worm, which manipulated the parameters of a set of centrifuges used for uranium enrichment. Such an attack has two stages: the compromise and the payload. Traditionally, once an adversary has compromised an information system, delivering a preconstructed payload is straightforward. This is because the attacker usually has a copy of the software being attacked. However, for ICSs, this is not necessarily the case. Depending on the type of attack the adversary mounts, construction of the payload may be either error prone or nearly impossible. A payload is either indiscriminate or targeted.

An indiscriminate payload performs random attacks causing malicious actions within the machinery of a victim ICS. There are several ways malware can automatically construct indiscriminate payloads upon gaining access to one or more of the victim ICS PLCs [80]. The assumption here is that if the malware is able to write to the PLC code area, then it must also be able to read from the PLCs code area.[1] Given the ability to read the PLC code several methods may be used to construct indiscriminate payloads.

1) The malware infers basic safety properties known as *interlocks* [82] and generates a payload which sequentially violates as many safety properties as possible.
2) The malware identifies the main timing loop in the system. Consider the example of a traffic light, where the main loop ensures that each color of light is active in sequence for a specific period of time. The malware can then construct a payload that violates the timing loop, e.g., by allowing certain lights to overlap.
3) In the bus enumeration technique, the malware uses the standardized identifiers such as Profibus IDs to find specific devices within a victim system [83].

While these indiscriminate payload construction methods are generic, they have a number of shortcomings. First, in the case where the payload is unaware of the actual devices in the victim ICS, one cannot guarantee that the resulting payload will cause damage (or achieve any other objective). Second, they cannot guarantee that the resulting payload will be stealthy. Thus, the malicious behavior may be discovered before it becomes problematic. Finally, there is no guarantee that a payload can be constructed at all. If the malware is unable to infer safety properties, timing loops, or the types of physical machinery present, then it is not possible to construct a payload that exploits them.

[1]This assumption was confirmed in a study of PLC security measures placed as an ancillary section in an evaluation of a novel security mechanism [81]. The conclusion was that PLC access control policies are all or nothing, meaning that write access implies read access.

A targeted payload, on the other hand, attempts to achieve a specific goal within the physical system, such as causing a specific device to operate beyond its safe limits. The alternative is a targeted payload where the adversary is able to arbitrarily inspect the system under attack, i.e., he has a copy of the exploited software. For autonomous, malware-driven attacks against ICS, this is not the case. Embedded controllers used in ICS may be air-gapped, meaning that once malware infects them, it may no longer be able to contact its command and control servers. Additionally, possessing the control logic for a given PLC may not be sufficient to analyze the system manually, as the assembly-language-like control logic does not reveal which physical devices are controlled by which program variables. Malware can construct such a targeted payload against a compromised ICS [84]. They assume that the adversary launching the malware has imperfect knowledge about the physical machinery in the ICS, and is also mostly aware of their interactions. However, the adversary lacks two key pieces of information: 1) the complete and precise behavior of the ICS; and 2) the mapping between the memory addresses of the victim PLC and the physical devices in the ICS. This mapping is important, as often the variable names in a PLC code reveal nothing about the devices they control.

Assuming that the attacker can encode his limited knowledge of the victim plant into a temporal logic, a program analysis tool called SABOT can analyze the PLC code, and map behaviors of the memory addresses in the code to those in the adversary's temporal logic description of the system. The results show that by carefully constructing the temporal logic description of the system, the adversary can provide the malware with enough information to construct a targeted payload against most ICS devices.

These advances in payload generation defeat one of the main forms of security through obscurity: the inaccessibility and low-level nature of PLC code. The ability to generate a payload for a system without ever seeing its code represents a substantial lowering of the bar for ICS attackers, and thus should be a factor in any assessment methodology.

2) *False Data Injection (FDI):* In an FDI attack, the adversary selects a set of sensors that feed into one or more controllers. The adversary then supplies carefully crafted malicious values to these sensors, thus achieving a desired result from the controller. For example, if the

supplied malicious values tell the controller that a temperature is becoming too low, it will increase the setting on a heating element, even if the actual temperature is fine. This will then lead to undetected overheating.

The earliest FDI attack targeted power system state estimation [85]. State estimation is an important step in distributed control systems, where the actual physical state is estimated based on a number of observables. Power system state estimation determines how electric load is distributed across various high-voltage lines and substations within the power transmission network. Compromising a subset of phasor measurement units (PMUs) can result in incorrect state estimation. This work addressed two questions: 1) Which sensors should be compromised, and how few are sufficient to achieve the desired result? 2) How can the maliciously crafted sensor values bypass the error correction mechanisms built into the state estimation algorithm? By compromising only tens of sensors (out of hundreds or thousands) it is possible to produce inaccurate state estimations in realistic power system bus topologies.

FDI attacks on Kalman-filtering-based state estimation has been reported in [86]. Kalman filters are a more general form of state estimation than the linear, direct current (dc) system model. The susceptibility of a Kalman-filter-based state estimator to FDI attacks depends on inherent properties of the designed system [86]. The system is only guaranteed to be controllable via FDI attack if the underlying state transition matrix contains an unstable eigenvalue, among other conditions. This has important implications not only for attacks, but also for defenses against FDI attacks, since a system lacking an unstable eigenvalue may not be perfectly attacked.

## IV. MITIGATING ATTACKS ON ICSs

In this section, we review the following ICS defenses: software-based mitigation, secure controller architectures to detect intrusions, and theoretical frameworks to understand the limits of mitigation.

### A. Software Mitigations

Embedded systems software is programmed using native (unsafe) programming languages such as C or assembly language. As a consequence, it suffers from memory exploits, such as buffer overflows. After gaining control over the program flow the adversary can inject malicious code to be executed (code injection [87]), or use existing pieces (gadgets) that are already residing in program memory (e.g., in linked libraries) to implement the desired malicious functionality (return-oriented programming [88]). Moreover, return-oriented programming is Turing complete, i.e., it allows an attacker to execute arbitrary malicious code. The latter attacks are often referred to as code-reuse attacks since they use benign code of existing ICS software. Code-reuse attacks are prevalent and are applicable to a wide range of computing platforms. The Stuxnet is known to have used code-reuse attacks [89].

Defenses against these attacks focus on either the enforcing control-flow integrity (CFI) or randomizing the memory layout of an application by means of fine-grained code randomization. We elaborate on these two defenses. These defenses assume an adversary who is able to overwrite control-flow information in the data area of an application. There is a large body of work that prevents this initial overwrite; a discussion of these approaches is beyond the scope of this paper.

*1) Control-Flow Integrity:* This defense technique against code-reuse ensures that an application only executes according to a predetermined control-flow graph (CFG) [90]. Since code injection and return-oriented programming result in a deviation of the CFG, CFI detects and prevents the attack. CFI can be realized as a compiler extension [91] or as a binary rewriting module [90].

CFI has performance overhead caused by control-flow validation instructions. To reduce this overhead, a number of proposals have been made: kBouncer [92], ROPecker [93], CFI for COTS binaries [94], ROPGuard [95], and CCFIR [96]. These schemes enforce so-called coarse-grained integrity checks to improve performance. For instance, they only constrain function returns to instructions following a call instruction rather than checking the return address against a list of valid return addresses held on a shadow stack. Unfortunately, this tradeoff between security and performance allows for advanced code-reuse attacks that stitch together gadgets from call-preceded sequences [97]–[100]. Some runtime CFI techniques leverage low-level hardware events [101]–[103]. Another host-based CFI check injects intrusion detection functionality into the monitored program [104].

Until now, the majority of research on CFI has focused on software-based solutions. However, hardware-based CFI approaches are more efficient. Further, dedicated hardware CFI instructions allow for system-wide CFI protection using these instructions. The first hardware-based CFI approach [105] realized the original CFI proposal [90] as a CFI state machine in a simulation environment of the Alpha processor. HAFIX proposes hardware-based CFI instructions and has been implemented on real hardware targeting Intel Siskiyou Peak and SPARC [106], [107]. It generates 2% performance overhead across different embedded benchmarks by focusing on preventing return-oriented programming attacks exploiting function returns.

*Remaining Challenges*: Most proposed CFI defenses focus on the detection and prevention of return-oriented programming attacks, but do not protect against return-into-libc attacks. This is only natural, because the majority of code-reuse attacks require a few return-oriented gadgets to initialize registers and prepare memory before calling a system call or critical function. However, Schuster *et al.* [108] have demonstrated that code-reuse attacks based on only calling a chain of virtual methods allow arbitrary malicious program actions. In addition, it has been demonstrated that pure return-into-libc attacks can achieve Turing completeness [109]. Detecting such attacks is challenging: modern programs link to a large number of libraries, and require dangerous API and system calls to operate correctly [97]. Hence, for these programs, dangerous API and system calls are legitimate control-flow targets for indirect and direct call instructions, even if fine-grained CFI policies are enforced. In order to detect code-reuse attacks that exploit these functions, CFI needs to be combined with additional security checks, e.g., dynamic taint analysis and techniques that perform argument validation. Developing such CFI extensions is an important future research direction.

*2) Fine-Grained Code Randomization:* A widely deployed countermeasure against code-reuse attacks is the randomization of the applications' memory layout. The key idea here is one of software diversity [110]. The key observation is that an adversary typically attempts to compromise many systems using the same attack vector. To mitigate this attack, one can diversify a program implementation into multiple and different semantically equivalent instances [110]. The goal is to force the adversary to tailor the attack vector for each software instance, making the attack prohibitive. Different approaches can be taken for realizing software diversity, e.g., memory randomization [111], [112], based on a compiler [110], [113], [114], or by binary rewriting and instrumentation [115]–[118].

A well-known instance of code randomization is address space layout randomization (ASLR) which randomizes the base address of shared libraries and the main executable [112]. Unfortunately, ASLR is often bypassed in practice due to its low randomization entropy and memory disclosure attacks which enable prediction of code locations. To tackle this limitation, a number of fine-grained ASLR schemes have been proposed [115]–[120]. The underlying idea is to randomize the code structure, for instance, by shuffling functions, basic blocks, or instructions (ideally for each program run [117], [118]). With fine-grained ASLR enabled, an adversary cannot reliably determine the addresses of interesting gadgets based on disclosing a single runtime address.

However, a recent just-in-time return-oriented programming (JIT-ROP) attack, circumvents fine-grained ASLR by finding gadgets and generating the return-oriented payload on the fly [121]. As for any other real-world code-reuse attack, it only requires a memory disclosure of a single runtime address. However, unlike code-reuse attacks against ASLR, JIT-ROP only requires the runtime address of a valid code pointer, without knowing to which precise code part or function it points to. Hence, JIT-ROP can use any code pointer such as return addresses on the stack to instantiate the attack. Based on the leaked address, JIT-ROP can disclose the content of multiple memory pages, and generates the return-oriented payload at runtime. The key insight of JIT-ROP is that a leaked code pointer will reside on a 4-kB aligned memory page. This can be exploited leveraging a scripting engine (e.g., JavaScript) to determine the affected page's start and end address. Afterwards, the attacker can start disassembling the randomized code page from its start address, and identify useful return-oriented gadgets.

To tackle this class of code-reuse attacks, defenses have been proposed [122]–[124]. Readactor leverages a hardware-based approach to enable execute-only memory [124]. For this, it exploits Intel's extended page tables to conveniently mark memory pages as nonexecutable. In addition, an LLVM-based instrumented compiler 1) permutes function; 2) strictly separates code from data; and 3) hides code pointers. As a consequence, a JIT-ROP attacker can no longer disassemble a page (i.e., the code pages are set to nonreadable). In addition, one cannot abuse code pointers located on the application's stack and heap to identify return-oriented gadgets, since Readactor performs code pointer hiding.

*Remaining Challenges*: CFI provides provable security [125]. That is, one can formally verify that CFI enforcement is sound. In particular, the explicit control-flow checks inserted by CFI into an application provide strong assurance that a program's control flow cannot be arbitrarily hijacked by an adversary. In contrast, code randomization does not put any restriction on the program's control flow. In fact, the attacker can provide any valid memory address as an indirect branch target. Another related problem of protection schemes based on code randomization are side-channel attacks [126], [127]. These attacks exploit timing and fault analysis side channels to infer randomization information. Recently, several defenses started to combine CFI with code randomization. For instance, Mohan *et al.* [128] presented opaque CFI (O-CFI). This solution leverages coarse-grained CFI checks and code randomization to prevent return-oriented exploits. For this, O-CFI identifies a unique set of possible target addresses for each indirect branch instruction. Afterwards, it uses the per-indirect branch set to restrict the target address of the indirect branch to only its minimal and maximal members. To further reduce the set of possible addresses, it arranges basic blocks belonging to an indirect branch set into clusters (so that they are located nearby to each other), and also randomizes their location. However, O-CFI relies on precise static analysis. In particular, it statically determines valid branch addresses for return instructions which typically leads to coarse-grained policies.

Nevertheless, Mohan *et al.* [128] demonstrate that combining CFI with code randomization is a promising research direction.

## B. Novel/Secure Control Architectures

In this section, we consider mitigations for the problem raised in Section III-B1. The threat here is that an adversary may tamper with a controller's logic code, thus subverting its behavior. This can be generalized to the notion of an untrusted controller. We consider four novel architectures for this problem: TSV, a tool for statically checking controller code; $C^2$, a dynamic reference monitor for a running controller; S3A, a controller architecture that represents a middle ground between TSV and $C^2$; and finally, an approach for providing a trusted computing base (TCB) for a controller so that PLCs may dependably enforce safety properties on themselves.

*1) Trusted Safety Verifier (TSV) [81]:* As previously discussed, one method for tampering with an ICS process is to upload malicious logic to a PLC. This was demonstrated by the Stuxnet attack. TSV prevents uploading of malicious control logic by statically verifying that logic *a priori* [81]. TSV sits on an embedded devices next to a PLC and intercepts all PLC-bound code and statically verifies it against a set of designer supplied safety properties. TSV does this in a number of steps. First, the control logic is symbolically executed to produce a symbolic scan cycle. A symbolic scan cycle represents all possible single-scan cycle executions of the control logic. It then finds feasible transitions between subsequent symbolic scan cycles to form a temporal execution graph (TEG). The TEG is then fed into a model checker which will verify that a set of linear temporal logic safety properties hold under the TEG model. If the control logic violates any safety property, the model checker will return a counterexample input that would cause the violation, and the control logic would be blocked from running on the PLC. The main drawback of TSV is that often the TEG is a tree structure of bounded depth. Thus, systems beyond a certain complexity cannot be effectively checked by TSV in a reasonable amount of time.

*2) $C^2$ Architecture [129]:* It provides a dynamic reference monitor for sequential and hybrid control systems. Like TSV, $C^2$ enforces a set of engineer-supplied safety properties. However, enforcement in $C^2$ is done at runtime, by an external module positioned between a PLC and the ICS hardware devices. At the end of each PLC scan cycle, a new set of control signals are sent to the ICS devices. $C^2$ will check these signals, along with the current ICS state, against the safety properties. Any unsafe modifications of the plant state are denied. If at any step, an attempted control signal is denied by $C^2$, it will enact one of a number of deny disciplines to deal with the potentially dangerous operation. One of the main results from $C^2$ evaluation was that all deny disciplines should support notifying the PLC of the denial, so that it knows the plant did not receive the control signal. A key shortcoming of $C^2$ is that it can only detect violations immediately before they occur. What is preferable is a system that can give advanced warnings, like in the TSV's static analysis, but can work for complex ICS, like $C^2$.

*3) Secure System Simplex Architecture (S3A) [130]:* Similar to how TSV requires a copy of the control logic, S3A requires the high-level system control flow and execution time profiles for the system under observation. Similar to how $C^2$ performs real-time monitoring, S3A aims to detect when the system is approaching an unsafe state. However, different from $C^2$, S3A aims to give a deterministic time buffer before potentially entering the unsafe state [131]. While S3A has the advantage of more advanced detection, it cannot operate on arbitrarily complex systems like $C^2$ can. However, it is appropriate for more complex systems than TSV.

*Remaining Challenges*: In this review of TSV, $C^2$, and S3A, we see a tradeoff forming: complexity of the monitored system versus amount of advanced warning. TSV sits at one end of this spectrum, offering the most advanced warning for systems of bounded complexity, while $C^2$ sits at the other end, offering last second detection of unsafe states on arbitrarily complex systems. The S3A approach represents a compromise between the two, however, the more complex the system is, the more detailed the control flow and timing information fed to S3A must be, while in the future, computational power for verification may be substantial enough to allow for full TSV analysis of arbitrarily complex systems. However, for current, practical solutions, this is not a reasonable assumption.

Part of the reason none of the existing architectures can win both ends of the tradeoff is that they all exist outside the PLC. This also adds significant cost and complexity, as they must be physically integrated with an existing control system. An alternative approach is to construct future PLCs to provide a minimal trusted computing base (TCB). One such TCB with the goal of restricting the ability to manipulate physical machinery to a small set of privileged code blocks within the PLC memory is proposed in [132]. This TCB is not itself aware of the ICS physical safety properties. Instead, the goal of this TCB is to protect a privileged set of code blocks that are able to affect the plant, i.e., via control signals. The privileged code blocks then contain the safety properties. Thus, $C^2$ or S3A-like checks are done from within these blocks. This approach has the added benefit that a TSV-like verification of safety properties in the privileged blocks is substantially simpler than verifying an entire system, thus allowing for a static analysis of more complex system than TSV.

### C. Detection of Control and Sensor Injection Attacks

When considering attacks against ICS, there are two important channels that must be considered: the control channel and the sensor channel. A control channel attack compromises a computer, a controller, or individual upstream from the physical process. The compromised entity then injects malicious commands into the system. A sensor channel attack corrupts sensor readings coming from the physical plant in order to cause bad decision making by the controllers receiving those sensor readings.[2] In this section, we review detection of control channel attacks and FDI. Techniques for control channel attacks inherit from the existing body of work in network and host intrusion detection, whereas FDI detection stems largely from the theory of state estimation and control.

*1) Detecting Control Channel Attacks:* A survey of SCADA intrusion detection between 2004 and 2008 can be found in [133]. It presents a taxonomy in which detection systems are categorized based on the following.

- Degree of SCADA specificity: How well does the solution leverage some of the unique aspects of SCADA systems?
- Domain: Does the solution apply to any SCADA system, or is it restricted to a single domain, e.g., water.
- Detection principle: What method is used to categorize events: behavioral, specification, anomaly, or a combination?
- Intrusion-specific: Does the solution only address intrusions, or is it also useful for fault detection?
- Time of detection: Is the threat detected and reported in real time, or only as an offline operation?
- Unit of analysis: Does the solution examine network packets, API calls, or other events?

We find that among the categorized systems there are some deficiencies. First, they lack a well-defined threat model. Second, they do not account for the degree of heterogeneity found in real-world ICS, e.g., use of multiple protocols. Finally, the proposed systems were not sufficiently evaluated for false positives, and insufficient strategies for dealing with false positives were given.

We review recent work that aims at greater feasibility [134]. In this approach, a specification is derived for traffic behavior over smart meter networks, and formal verification is used to ensure that any network trace conforming to the specification will not violate a given security policy. The specification is formed based on: 1) the smart meter protocols (in this case, the ANSI C12 family); 2) a system model consisting of state machines that describe a meter's lifetime, e.g., provisioning, normal operation, error conditions, etc., as well as the network topology; and 3) a set of constraints on allowed behavior. An evaluation of a prototype implementation showed that no more than 1.6% of CPU usage was needed for monitoring the specification at meters. One potential limitation of this approach is the need for expert-provided information in the form of the system model and constraints on allowed behavior.

An alternative approach, which is not dependent on specifications, is given in [135]. This solution builds a model of good behavior through observation of three types of quantities visible to PLCs: sensor measurements, control signals, and events such as alarms. The behavioral model uses autoregression to predict the next system state. To avoid low-and-slow attacks that autoregression may not catch, upper and lower Shewart control limits are used as absolute bounds on process variables that may not be crossed. Their evaluation on one week of network traces from a prototype control system showed that most normal behaviors were properly modeled by the autoregression. There were, however, several causes of deviations including nearly constant signals that occasionally deviated briefly before returning to their prior constant value, and a counter variable that experienced a delayed increment. Such cases would represent false positives for the autoregression model, but would not necessarily trip the Shewart control limits.

*2) Detecting FDI:* While control channel attacks directly target controllers with malicious commands, FDI attacks can be more subtle, as they used forged sensor data to cause the controller to make misguided decisions. Detection of FDI attacks is thus deeply rooted in the existing discipline of state estimation discussed earlier. This will require a) a measurement model that relates the measured quantity to the physical value that caused it, i.e., heat propagation; and b) an error detection method to allow for faulty measurements to be discarded.

We described one attack against power grid state estimation in which estimation errors could be caused by tampering with a relatively small number of PMUs [85]. In one approach to detecting such an attack, one can use a small, strategically selected set of tamper-resistant meters to provide independent measurements [136]. These out-of-band measurements are used to determine the accuracy of the remaining majority of measurements contributing to the state estimation.

In a second approach [137], two security indices are computed for a given state estimator. The first index measures how well the state estimator's bad data detector can handle attacks where the adversary is limited to a few tampered measurements. The second index measures how well the bad data detector can handle attacks where the adversary only makes small changes to measurement magnitudes. Along with the grid topology information, these indices can be useful in determining how to

[2]More information about FDI can be found in Section III-B2.

allocate security functionality such as retrofitted encryption to various measurement devices.

The third approach differs from the above two in that it does not attempt to select a set of meters for security enhancements, but instead, places weights on the grid topology to reflect the trustworthiness of various PMUs [138]. The trust weights are integrated to a distributed Kalman filtering algorithm to produce a state estimation that more accurately reflects the trustworthiness of the individual PMUs. In the evaluation of the distributed Kalman filters, it was found that they converged to the correct state estimate in approximately 20 steps.

From these approaches to detecting both control channel and FDI attacks, one can see that an effective method toward detecting ICS intrusions involves monitoring the physical process itself, as well as its interactions with the controller and sensors.

## D. Theoretical ICS Security Frameworks

A number of recent advances have generalized the increasing body of results to provide theoretical frameworks. In this section, we will review such theoretical frameworks based on the following three approaches: 1) modeling attacker behavior and identifying likely attack scenarios; 2) defining the general detection and identification of attacks against ICS; and 3) the distribution of security enhancements in ICSs where controllers share network infrastructure. A common theme in these frameworks is the optimal distribution of security protections in large, legacy ICSs.

Adding protections like cryptographic communications to legacy equipment is expensive. Thus, it is preferable to secure the most vulnerable portions of an ICS. Teixeira *et al.* describe a risk management framework for ICSs [139]. Starting with the notion of security indices [137], this work looks at methods for identifying the most vulnerable measurements in both static and dynamic control systems. For static control systems, it is assumed that adversaries wish to execute the minimum cost attack. In the static case, the $\alpha_k$ index described in Section IV-C is sufficient, and methods are given for efficiently computing $\alpha_k$ for large systems.

In the case of dynamic systems, the maximum-impact, minimum-resource attacks are defined as a multiobjective optimization problem. For such a problem, the basic security indices do not suffice. Instead, the multiobjective problem is transformed into a maximum-impact, resource-constrained problem. An example is given where this is used to calculate the attack vectors for a quadruple-tank system. The resulting, optimal attack strategy can be used to allocate defenses such as data encryption in the ICS.

Another framework considers generalizing attacks against ICSs and describe the fundamental limitations of monitors against these attacks [140]. Assuming that an attack monitor is consistent, i.e., does not generate false positives, it is shown that some attacks are undetectable if there is an initial state which produces the same final state as an attack. Additionally, it is shown that some attacks cannot be distinguished from others. These results are applicable to stealthy [141], replay [142], and FDI attacks.

The previous two approaches considered ways of modeling attacks and attack likelihoods against individual control loops. However, in some systems, a number of otherwise independent control process are actually somewhat dependent due to the shared network. In this case, a distributed denial of service (DDoS) attack against one controller may affect others. The problem of interdependent control systems using a game-theoretic approach is addressed in [143]. The noncooperative game consists of two stages: 1) each control loop (player) chooses whether to apply security enhancements; and 2) each player applies the optimal control input to its plant.

In the nonsocial form of this game, players only attempt to minimize their own cost, which consists of the operating costs of the plant plus the cost of adding and maintaining security measures. For this form of the game, with $M$-players, there is shown to exist a unique equilibrium solution. The solutions to this game may not be globally optimal, due to externalities imposed by players that opt-out of security enhancements. To solve this, penalties are introduced for players that do not select security enhancements leading to a guaranteed unique solution that is also globally optimal. While such a game-theoretic approach is useful in distributing the cost of security enhancements, actually achieving robust control in distributed systems is more difficult, especially when the system in question is nonlinear. To this end, a modification to a traditional model-predicative control (MPC) problem has been suggested [144]. Adding a robustness constraint to the MPC problem can bound the values of future states.

These theoretical frameworks offer opportunities to understand and improve defenses. However, it is important to understand the assumptions behind the frameworks. For example, the above approaches assume the following.

1) Attackers are omniscient, knowing the exact measurement, control, and process matrices for each system, as well as all system states.
2) Attackers are nearly omnipotent, with the ability to compromise any measurement and control vector. There is an important exception here, which is that detectors are assumed to be immune to attackers.
3) Detectors do not create false positives and systems are completely deterministic (first two approaches).
4) Security enhancements can significantly mitigate DDoS attacks on various network architectures (third approach).

In any assessment procedure, the actual set of assumptions should be considered and compared with those of the theoretical framework being used in the assessment.

## V. CONCLUSION

ICT-based ICS can deliver real-time information, resulting in automatic and intelligent control of industrial processes. Inherently dangerous processes, however, are no longer immune to cyber threats, as vulnerable devices, formats, and protocols are not hosted on dedicated infrastructure due to cost pressures. Consequently, ICS infrastructure has become increasingly exposed, either by direct connection to the Internet, or via interfaces to utility IT systems. Therefore, inflicting substantial damage or widespread disruption may be possible with a comprehensive analysis of the target systems. Publicly available information, combined with default and well-known ICS configuration details, could potentially allow a resource-rich adversary to mount a large-scale attack.

This paper surveyed the state of the art in ICS security, identified outstanding research challenges in this emerging area, and motivated the deployment of cybersecurity methods and tools to ICS. All levels of the multilayered ICS architecture can be targeted by sophisticated cyberattacks and disturb the control process of the ICS. Assessing the vulnerabilities of ICS requires the development of a uniquely-ICS multilayered testbed that establishes as many pathways as possible between the cyber and physical components in the ICS. These pathways can assist in determining the real-world consequences in terms of the technical impacts and the severity of the outcomes. An important direction of research is to develop effective methods to detect ICS intrusions that involve monitoring the physical processes themselves, as well as their interactions with the controller and sensors. ■

## Acknowledgment

The authors from New York University (NYU) would like to thank S. Lee, P. Robison, P. Stergiou, and S. Kim from Consolidated Edison for their continuous support on the project, Platform Profiling in Legacy and Modern Control and Monitoring Systems.

## REFERENCES

[1] K. Stouffer, J. Falco, and K. Scarfone, “Guide to industrial control systems (ICS) security,” NIST Special Publication 800-82, 2011. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-82/SP800-82-final.pdf.

[2] E. Hayden, M. Assante, and T. Conway, “An abbreviated history of automation & industrial controls systems and cybersecurity,” 2014. [Online]. Available: https://ics.sans.org/media/An-Abbreviated-History-of-Automation-and-ICS-Cybersecurity.pdf.

[3] European Network and Information Security Agency (ENISA), “Protecting industrial control systems—Recommendations for Europe and member states,” 2011. [Online]. Available: https://www.enisa.europa.eu/.

[4] T. M. Chen and S. Abu-Nimeh, “Lessons from Stuxnet,” *Computer*, vol. 44, no. 4, pp. 91–93, 2011.

[5] P. Muncaster, “Stuxnet-like attacks beckon as 50 new SCADA threats discovered,” 2011. [Online]. Available: http://www.v3.co.uk/v3-uk/news/2045556/stuxnet-attacks-beckon-scada-threats-discovered.

[6] J. Weiss, “Assuring industrial control system (ICS) cyber security,” [Online]. Available: http://csis.org/files/media/csis/pubs/080825_cyber.pdf.

[7] R. Anderson *et al.*, “Measuring the cost of cybercrime,” in *The Economics of Information Security and Privacy*. Berlin, Germany: Springer-Verlag, 2013, pp. 265–300.

[8] Willis Group, “Energy market review 2014—Cyber-attacks: Can the market respond?” 2014. [Online]. Available: http://www.willis.com/.

[9] Y. Mo *et al.*, “Cyber-physical security of a smart grid infrastructure,” *Proc. IEEE*, vol. 100, no. 1, pp. 195–209, Jan. 2012.

[10] P. Radmand, A. Talevski, S. Petersen, and S. Carlsen, “Taxonomy of wireless sensor network cyber security attacks in the oil and gas industries,” in *Proc. 24th Int. Conf. Adv. Inf. Netw. Appl.*, 2010, pp. 949–957.

[11] S. Amin, X. Litrico, S. Sastry, and A. M. Bayen, “Cyber security of water SCADA systems—Part I: Analysis and experimentation of stealthy deception attacks,” *IEEE Trans. Control Syst. Technol.*, vol. 21, no. 5, pp. 1963–1970, 2013.

[12] “What is a programmable logic controller (PLC)?” [Online]. Available: http://www.wisegeek.org/what-is-a-programmable-logic-controller.htm.

[13] A. Scott, “What is a distributed control system (DCS)?” [Online]. Available: http://blog.cimation.com/blog/bid/198186/What-is-a-Distributed-Control-System-DCS.

[14] Kaspersky, “Cyperthreats to ICS systems,” 2014. [Online]. Available: http://media.kaspersky.com/en/business-security/critical-infrastructure-protection/Cyber_A4_Leaflet_eng_web.pdf.

[15] J. Meserve, “Mouse click could plunge city into darkness, experts say,” *CNN*, 2007. [Online]. Available: http://www.cnn.com/2007/US/09/27/power.at.risk/index.html.

[16] Security Matters, “The Aurora attack.” [Online]. Available: http://www.secmatters.com/casestudy10.

[17] D. Kushner, “The real story of Stuxnet,” 2013. [Online]. Available: http://spectrum.ieee.org/telecom/security/the-real-story-of-stuxnet.

[18] M. B. Line, A. Zand, G. Stringhini, and R. Kemmerer, “Targeted attacks against industrial control systems: Is the power industry prepared?” in *Proc. 2nd Workshop Smart Energy Grid Security*, 2014, pp. 13–22.

[19] C. Miller and C. Valasek, “Remote exploitation of an unaltered passenger vehicle,” 2015. [Online]. Available: https://www.defcon.org/html/defcon-23/dc-23-speakers.html#Miller.

[20] K. Thomas, “Hackers demo Jeep security hack,” 2015. [Online]. Available: http://www.welivesecurity.com/2015/07/22/hackers-demo-jeep-security-hack/.

[21] N. G. Tsoutsos, C. Konstantinou, and M. Maniatakos, “Advanced techniques for designing stealthy hardware trojans,” in *Proc. 51st Design Autom. Conf.*, 2014, pp. 1–4.

[22] Y. Jin, M. Maniatakos, and Y. Makris, “Exposing vulnerabilities of untrusted computing platforms,” in *Proc. 30th IEEE Int. Conf. Comput. Design*, 2012, pp. 131–134.

[23] K. Rosenfeld and R. Karri, “Attacks and Defenses for jtag,” 2010.

[24] M. F. Breeuwsma, “Forensic imaging of embedded systems using jtag (boundary-scan),” *Digital Investigation*, vol. 3, no. 1, pp. 32–42, 2006.

[25] J. Barnaby, “Exploiting embedded systems, Black Hat 2006,” 2006. [Online]. Available: http://www.blackhat.com/presentations/bh-europe-06/bh-eu-06-Jack.pdf.

[26] D. Schneider, “USB flash drives are more dangerous than you think.” [Online]. Available: http://spectrum.ieee.org/tech-talk/computing/embedded-systems/usb-flash-drives-are-more-dangerous-than-you-think.

[27] USB Killer. [Online]. Available: http://kukuruku.co/hub/diy/usb-killer.

[28] bunnie and xobs, 30c3d, "The exploration and exploitation of an SD memory card." [Online]. Available: http://bunniefoo.com/bunnie/sdcard-30c3-pub.pdf.

[29] S. Skorobogatov, "Flash memory 'bumping' attacks," in *Proc. Cryptogr. Hardware Embedded Syst.*, 2010, pp. 158–172.

[30] R. Templeman and A. Kapadia, "Gangrene: Exploring the mortality of flash memory," *HotSec*, vol. 12, p. 1, 2012.

[31] X. Wang, C. Konstantinou, M. Maniatakos, and R. Karri, "Confirm: Detecting firmware modifications in embedded systems using hardware performance counters," in *Proc. 34th IEEE/ACM Int. Conf. Comput.-Aided Design*, 2015, pp. 544–551.

[32] CRitical Infrastructure Security AnaLysIS, "Crisalis Project EU, Deliverable D2.2 Final Requirement Definition." [Online]. Available: http://www.crisalis-project.eu/.

[33] DHS, "Common cybersecurity vulnerabilities in industrial control systems," 2011. [Online]. Available: http://ics-cert.us-cert.gov/.

[34] D. Beresford, "The Sauce of Utter pwnage," 2011. [Online]. Available: http://thesauceofutterpwnage.blogspot.com/.

[35] Common Vulnerabilities and Exposures, "CVE-2011-2367." [Online]. Available: http://cve.mitre.org/.

[36] C. Nan, I. Eusgeld, and W. Kröger, "Hidden vulnerabilities due to interdependencies between two systems," in *Proc. Critical Inf. Infrastructures Security*, 2013, pp. 252–263.

[37] T. Macaulay and B. L. Singer, *Cybersecurity for Industrial Control Systems: SCADA, DCS, PLC, HMI, SIS*. Boca Raton, FL, USA: CRC Press, 2011.

[38] S. Shetty, T. Adedokun, and L.-H. Keel, "Cyberphyseclab: A testbed for modeling, detecting and responding to security attacks on cyber physical systems," in *Proc. 3rd ASE Int. Conf. Cyber Security*, 2014.

[39] Center for the Protection of National Infrastructure (CPNI), "Cyber security assessments of industrial control systems," 2010. [Online]. Available: https://ics-cert.us-cert.gov/sites/default/files/documents/Cyber_Security_Assessments_of_Industrial_Control_Systems.pdf.

[40] C1 Working Group Members of Power System Relaying Committee, "Cyber security issues for protective relays," 2008. [Online]. Available: https://www.gedigitalenergy.com/smartgrid/May08/7_Cyber-Security_Relays.pdf.

[41] R. E. Mahan *et al.*, "Secure data transfer guidance for industrial control and SCADA systems," 2011. [Online]. Available: http://www.pnnl.gov/main/publications/external/technical_reports/PNNL-20776.pdf.

[42] B. Rolston, "Attack methodology analysis: Emerging trends in computer-based attack methodologies and their applicability to control system networks," 2005. [Online]. Available: http://www5vip.inl.gov/technicalpublications/Documents/3494179.pdf.

[43] W. Grega, "Hardware-in-the-loop simulation and its application in control education," in *Proc. 29th Annu. Front. Edu. Conf.*, 1999, vol. 2, pp. 12B6–12B7.

[44] C. Konstantinou and M. Maniatakos, "Impact of firmware modification attacks on power systems field devices," in *Proc. 6th IEEE Int. Conf. Smart Grid Commun.*, 2015, pp. 1–6.

[45] M. Souppaya and K. Scarfone, "Guide to enterprise patch management technologies," 2013. [Online]. Available: http://dx.doi.org/10.6028/NIST.SP.800-40r3.

[46] European Network and Information Security Agency (ENISA), "Good practice guide for certs in the area of industrial control systems," 2013. [Online]. Available: https://www.enisa.europa.eu/.

[47] A. Nicholson, H. Janicke, and A. Cau, "Position paper: Safety and security monitoring in ICS/SCADA systems," in *Proc. 2nd Int. Symp. ICS SCADA Cyber Security Res.*, 2014, pp. 61–66.

[48] C. Konstantinou, M. Maniatakos, F. Saqib, S. Hu, J. Plusquellic, and Y. Jin, "Cyber-physical systems: A security perspective," in *Proc. 20th IEEE Eur. Test Symp.*, 2015, pp. 1–8.

[49] Scilab, "Open source and cross-platform platform." [Online]. Available: http://www.scilab.org/.

[50] Scicos, "Block diagram modeler/simulator." [Online]. Available: http://www.scicos.org/.

[51] H. K. Fathy, Z. S. Filipi, J. Hagena, and J. L. Stein, "Review of hardware-in-the-loop simulation and its prospects in the automotive area," *Ann Arbor*, vol. 1001, pp. 48 109–2125, 2006.

[52] National Institute of Standards and Technology, "A cybersecurity testbed for industrial control systems," 2014. [Online]. Available: http://www.nist.gov/manuscript-publication-search.cfm?pub_id=915876.

[53] M. Zeller, "Myth or reality-does the aurora vulnerability pose a risk to my generator?" in *Proc. 64th Annu. Conf. Protective Relay Eng.*, 2011, pp. 130–136.

[54] National Institute of Standards and Technology, "Measurement Challenges and Opportunities for Developing Smart Grid Testbeds Workshop" 2014. [Online]. Available: http://www.nist.gov/smartgrid/upload/SG-Testbed-Workshop-Report-FINAL-1-2-8-2014.pdf.

[55] I. N. Fovino, M. Masera, L. Guidi, and G. Carpi, "An experimental platform for assessing SCADA vulnerabilities and countermeasures in power plants," in *Proc. 3rd Int. Conf. Human Syst. Interactions*, 2010, pp. 679–686.

[56] Idaho National Laboratory, "National SCADA Test Bed (NSTB) Program." [Online]. Available: https://www.inl.gov/.

[57] A. Hahn, A. Ashok, S. Sridhar, and M. Govindarasu, "Cyber-physical security testbeds: Architecture, application, evaluation for smart grid," *IEEE Trans. Smart Grid*, vol. 4, no. 2, pp. 847–855, 2013.

[58] Digital Bond, "Project basecamp." [Online]. Available: http://www.digitalbond.com/tools/basecamp/.

[59] B. Reaves and T. Morris, "An open virtual testbed for industrial control system security research," *Int. J. Inf. Security*, vol. 11, no. 4, pp. 215–229, 2012.

[60] P. F. Roberts, "Zotob, PnP Worms Slam 13 DaimlerChrysler Plants," 2008. [Online]. Available: http://www.eweek.com.

[61] "Computer viruses make it to orbit," *BBC News*, Aug. 2008. [Online]. Available: http://news.bbc.co.uk/2/hi/7583805.stm.

[62] B. Krebs, "Cyber incident blamed for nuclear power plant shutdown," *The Washington Post*, Jun. 2008. [Online]. Available: http://www.washingtonpost.com/wp-dyn/content/article/2008/06/05/AR2008060501958.html.

[63] S. Grad, "Engineers who hacked into L.A. traffic signal computer, jamming streets, sentenced," *Los Angeles Times*. [Online]. Available: http://latimesblogs.latimes.com/lanow/2009/12/engineers-who-hacked-in-la-traffic-signal-computers-jamming-traffic-sentenced.html.

[64] N. Leall, "Lessons from an insider attack on SCADA systems," 2009. [Online]. Available: http://blogs.cisco.com/security/lessons_from_an_insider_attack_on_scada_systems/.

[65] K. Zetter, "Clues suggest Stuxnet virus was built for subtle nuclear sabotage," *Wired*, 2010. [Online]. Available: http://www.wired.com/threatlevel/2010/11/stuxnet-clues/.

[66] J. Leyden, "Polish teen derails tram after hacking train network," *The Register*, 2008. [Online]. Available: http://www.theregister.co.uk/2008/01/11/tram_hack/.

[67] J. Meserve, "Sources: Staged cyber attack reveals vulnerability in power grid," *CNN*, 2007. [Online]. Available: http://articles.cnn.com/2007-09-26/us/power.at.risk_1_generator-cyber-attack-electric-infrastructure?_s=PM:US.

[68] E. Byres and J. Lowe, "The myths and facts behind cyber security risks for industrial control systems," in *Proc. ISA Process Control Conf.*, 2003.

[69] J. Pollet, "Electricity for free? the dirty underbelly of SCADA and smart meters," in *Proc. Black Hat USA*, 2010.

[70] L. Piétre-Cambacèdés, M. Trischler, and G. N. Ericsson, "Cybersecurity myths on power control systems: 21 misconceptions and false beliefs," *IEEE Trans. Power Delivery*, vol. 26, no. 1, pp. 161–172, 2011.

[71] National Energy Regulatory Commission, "NERC CIP 002 1—Critical cyber asset identification," 2006.

[72] J. Weiss, "Are the NERC CIPS making the grid less reliable," in *Proc. Control Global*, 2009.

[73] D. Beresford, "Exploiting Siemens Simatic S7 PLCs," in *Proc. Black Hat USA*, 2011.

[74] T. Newman, T. Rad, and J. Strauchs, "SCADA & PLC vulnerabilities in correctional facilities," white paper, 2011.

[75] S. Blumsack and A. Fernandez, "Ready or not, here comes the smart grid," *Energy*, vol. 37, no. 1, pp. 61–68, 2012.

[76] C. S. King, "The economics of real-time and time-of-use pricing for residential consumers," American Energy Institute, Tech. Rep., 2001.

[77] S. McLaughlin, D. Podkuiko, and P. McDaniel, "Energy theft in the advanced metering infrastructure," in *Proc. 4th Int. Workshop Critical Infrastructure Protection*, 2009.

[78] S. McLaughlin, D. Podkuiko, S. Miadzvezhanka, A. Delozier, and P. McDaniel, "Multi-vendor penetration testing in the advanced metering infrastructure," in *Proc. 26th Annu. Comput. Security Appl. Conf.*, 2010.

[79] Laboratory of Cryptography and System Security (CrySyS), "Duqu: A Stuxnet-like malware found in the wild," Budapest Univ. Technol. Econ., Budapest, Hungary, Tech. Rep., 2011.

[80] S. McLaughlin, "On dynamic malware payloads aimed at programmable logic controllers," in *Proc. 6th USENIX Workshop Hot Topics Security*, 2011.

[81] S. McLaughlin, D. Pohly, P. McDaniel, and S. Zonouz, "A trusted safety verifier for

process controller code," in *Proc. 21st Annu. Netw. Distrib. Syst. Security Symp.*, 2014.

[82] C. Chevillat, D. Carrington, P. Strooper, J. Süß, and L. Wildman, "Model-based generation of interlocking controller software from control tables," in *Model Driven Architecture—Foundations and Applications*, Lecture Notes in Computer ScienceI. Schieferdecker, and A. Hartman, Eds. Berlin, Germany: Springer-Verlag, 2008, vol. 5095, pp. 349–360.

[83] PROFIBUS, "IMS research estimates top position for PROFINET," 2010. [Online]. Available: http://www.profibus.com/news-press/detail-view/article/ims-research-estimates-top-position-for-profinet/.

[84] S. McLaughlin and P. McDaniel, "SABOT: Specification-based payload generation for programmable logic controllers," in *Proc. 19th ACM Conf. Comput. Commun. Security*, 2012.

[85] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," in *Proc. 16th ACM Conf. Comput. Commun. Security*, 2009.

[86] Y. Mo and B. Sinopoli, "False data injection attacks in control systems," in *Proc. 1st Workshop Secure Control Syst.*, 2010.

[87] A. One, "Smashing the stack for fun and profit," *Phrack Mag.*, vol. 49, no. 14, 2000.

[88] R. Roemer, E. Buchanan, H. Shacham, and S. Savage, "Return-oriented programming: Systems languages applications," *ACM Trans. Inf. Syst. Security*, vol. 15, no. 1, pp. 2:1–2:34, 2012.

[89] A. Matrosov, E. Rodionov, D. Harley, and J. Malcho, "Stuxnet under the microscope," 2011.

[90] M. Abadi, M. Budiu, Ú. Erlingsson, and J. Ligatti, "Control-flow integrity: Principles, implementations, applications," *ACM Trans. Inf. Syst. Security*, vol. 13, no. 1, 2009.

[91] J. Pewny and T. Holz, "Compiler-based CFI for iOS," in *Proc. 29th Annu. Comput. Security Appl. Conf.*, 2013.

[92] V. Pappas, M. Polychronakis, and A. D. Keromytis, "Transparent ROP exploit mitigation using indirect branch tracing," in *Proc. 22nd USENIX Security Symp.*, 2013.

[93] Y. Cheng, Z. Zhou, Y. Miao, X. Ding, and R. Huijie Deng, "ROPecker: A generic and practical approach for defending against ROP attacks," in *Proc. 21st Annu. Netw. Distrib. Syst. Security Symp.*, 2014.

[94] M. Zhang and R. Sekar, "Control flow integrity for COTS binaries," in *Proc. 22nd USENIX Security Symp.*, 2013.

[95] I. Fratric, "ROPGuard: Runtime prevention of return-oriented programming attacks," 2012.

[96] C. Zhang *et al.*, "Practical control flow integrity & randomization for binary executables," in *Proc. 34th IEEE Symp. Security Privacy*, 2013.

[97] E. Göktas, E. Athanasopoulos, H. Bos, and G. Portokalidis, "Out of control: Overcoming control-flow integrity," in *Proc. 35th IEEE Symp. Security Privacy*, 2014.

[98] L. Davi, D. Lehmann, A.-R. Sadeghi, and F. Monrose, "Stitching the gadgets: On the ineffectiveness of coarse-grained control-flow integrity protection," in *Proc. 23rd USENIX Security Symp.*, 2014.

[99] N. Carlini and D. Wagner, "ROP is still dangerous: Breaking modern defenses," in *Proc. 23rd USENIX Security Symp.*, 2014.

[100] F. Schuster *et al.*, "Evaluating the effectiveness of current anti-rop defenses," in *Research in Attacks, Intrusions and Defenses*, Lecture Notes in Computer Science. Berlin, Germany: Springer-Verlag, 2014, vol. 8688, pp. 88–108.

[101] J. Demme *et al.*, "On the feasibility of online malware detection with performance counters," *ACM SIGARCH Comput. Architect. News*, vol. 41, no. 3, pp. 559–570, 2013.

[102] A. Tang, S. Sethumadhavan, and S. J Stolfo, "Unsupervised anomaly-based malware detection using hardware features," in *Research in Attacks, Intrusions and Defenses*. Berlin, Germany: Springer-Verlag, 2014, pp. 109–129.

[103] X. Wang and R. Karri, "Numchecker: Detecting kernel control-flow modifying rootkits by using hardware performance counters," in *Proc. 50th Design Autom. Conf.*, 2013, pp. 1–7.

[104] A. Cui and S. J. Stolfo, "Defending embedded systems with software symbiotes," in *Recent Advances in Intrusion Detection*. Berlin, Germany: Springer-Verlag, 2011, pp. 358–377.

[105] M. Budiu, Ú. Erlingsson, and M. Abadi, "Architectural support for software-based protection," in *Proc. 1st Workshop Architect. Syst. Support Improving Softw. Dependability*, 2006, pp. 42–51.

[106] L. Davi, P. Koeberl, and A.-R. Sadeghi, "Hardware-assisted fine-grained control-flow integrity: Towards efficient protection of embedded systems against software exploitation," in *Proc. 51st Design Autom. Conf. Special Session: Trusted Mobile Embedded Computing*, 2014.

[107] O. Arias *et al.*, "HAFIX: Hardware-assisted flow integrity extension," in *Proc. 52nd Design Autom. Conf.*, 2015.

[108] F. Schuster *et al.*, "Counterfeit object-oriented programming: On the difficulty of preventing code reuse attacks in C++ applications," in *Proc. 36th IEEE Symp. Security Privacy*, 2015.

[109] M. Tran *et al.*, "On the expressiveness of return-into-libc attacks," in *Proc. 14th Int. Conf. Recent Adv. Intrusion Detection*, 2011.

[110] F. B. Cohen, "Operating system protection through program evolution," *Comput. Security*, vol. 12, no. 6, 1993.

[111] S. Forrest, A. Somayaji, and D. Ackley, "Building diverse computer systems," in *Proc. 6th Workshop Hot Topics Oper. Syst.*, 1997.

[112] PaX Team, "PaX Address Space Layout Randomization (ASLR)." [Online]. Available: http://pax.grsecurity.net/docs/aslr.txt.

[113] M. Franz, "E unibus pluram: Massive-scale software diversity as a defense mechanism," in *Proc. Workshop New Security Paradigms*, 2010.

[114] T. Jackson *et al.*, "Compiler-generated software diversity," in *Moving Target Defense*, Advances in Information Security. Berlin, Germany: Springer-Verlag, 2011, vol. 54, pp. 77–98.

[115] V. Pappas, M. Polychronakis, and A. D. Keromytis, "Smashing the gadgets: Hindering return-oriented programming using in-place code randomization," in *Proc. 33rd IEEE Symp. Security Privacy*, 2012.

[116] J. D. Hiser, A. Nguyen-Tuong, M. Co, M. Hall, and J. W. Davidson, "ILR: Where'd my gadgets go?" in *Proc. 33rd IEEE Symp. Security Privacy*, 2012.

[117] R. Wartell, V. Mohan, K. W. Hamlen, and Z. Lin, "Binary stirring: Self-randomizing instruction addresses of legacy x86 binary code," in *Proc. 19th ACM Conf. Comput. Commun. Security*, 2012.

[118] L. Davi, A. Dmitrienko, S. Nürnberger, and A.-R. Sadeghi, "Gadge me if you can—Secure and efficient ad-hoc instruction-level randomization for x86 and ARM," in *Proc. 8th ACM Symp. Inf. Comput. Commun. Security*, 2013.

[119] S. Bhatkar, R. Sekar, and D. C. DuVarney, "Efficient techniques for comprehensive protection from memory error exploits," in *Proc. 14th USENIX Security Symp.*, 2005.

[120] C. Kil, J. Jun, C. Bookholt, J. Xu, and P. Ning, "Address space layout permutation (ASLP): Towards fine-grained randomization of commodity software," in *Proc. 22nd Annu. Comput. Security Appl. Conf.*, 2006.

[121] K. Z. Snow *et al.*, "Just-in-time code reuse: On the effectiveness of fine-grained address space layout randomization," in *Proc. 34th IEEE Symp. Security Privacy*, 2013.

[122] M. Backes and S. Nürnberger, "Oxymoron: Making fine-grained memory randomization practical by allowing code sharing," in *Proc. 23rd USENIX Security Symp.*, 2014.

[123] M. Backes *et al.*, "You can run but you can't read: Preventing disclosure exploits in executable code," in *Proc. 21st ACM Conf. Comput. Commun. Security*, 2014.

[124] S. Crane *et al.*, "Readactor: Practical code randomization resilient to memory disclosure," in *Proc. 36th IEEE Symp. Security Privacy*, 2015.

[125] M. Abadi, M. Budiu, Ú. Erlingsson, and J. Ligatti, "A theory of secure control-flow," in *Proc. 7th Int. Conf. Formal Methods Softw. Eng.*, 2005.

[126] R. Hund, C. Willems, and T. Holz, "Practical timing side channel attacks against kernel space ASLR," in *Proc. 34th IEEE Symp. Security Privacy*, 2013.

[127] J. Seibert, H. Okhravi, and E. Söderström, "Information leaks without memory disclosures: Remote side channel attacks on diversified code," in *Proc. 21st ACM SIGSAC Conf. Comput. Commun. Security*, 2014.

[128] V. Mohan, P. Larsen, S. Brunthaler, K. W. Hamlen, and M. Franz, "Opaque control-flow integrity," in *Proc. 22nd Annu. Netw. Distrib. Syst. Security Symp.*, 2015.

[129] S. McLaughlin, "Stateful policy enforcement for control system device usage," in *Proc. 29th Annu. Comput. Security Appl. Conf.*, 2013.

[130] S. Mohan *et al.*, "S3A: Secure system simplex architecture for enhanced security of cyber-physical systems," in *Proc. 2nd ACM Int. Conf. High Confidence Netw. Syst.*, 2013.

[131] S. Bak, K. Manamcheri, S. Mitra, and M. Caccamo, "Sandboxing controllers for cyber-physical systems," in *Proc. 2nd IEEE/ACM Int. Conf. Cyber-Phys. Syst.*, 2011.

[132] S. McLaughlin, "Blocking unsafe behaviors in control systems through static and dynamic policy enforcement," in *Proc. 52nd Design Autom. Conf.*, 2015.

[133] B. Zhu and S. Sastry, "SCADA-specific intrusion detection/prevention systems: A

survey and tanomy," in *Proc. 1st Workshop Secure Control Syst.*, 2010.

[134] R. Berthier and W. H. Sanders, "Specification-based intrusion detection for advanced metering infrastructures," in *Proc. 17th IEEE Pacific Rim Int. Symp. Dependable Comput.*, 2011.

[135] D. Hadziosmanović, R. Sommer, E. Zambon, and P. H. Hartel, "Through the eye of the PLC: Semantic security monitoring for industrial processes," in *Proc. 31st Annu. Comput. Security Appl. Conf.*, 2014.

[136] R. B. Bobba *et al.*, "Detecting false data injection attacks on dc state estimation," in *Proc. 1st Workshop Secure Control Syst.*, 2010.

[137] H. Sandberg, A. Teixeira, and K. H. Johansson, "On security indices for state estimators in power networks," in *Proc. 1st Workshop Secure Control Syst.*, 2010.

[138] T. Jiang, I. Matei, and J. S. Baras, "A trust based distributed Kalman filtering approach for mode estimation in power systems," in *Proc. 1st Workshop Secure Control Syst.*, 2010.

[139] A. Teixeira, K. C. Sou, H. Sandberg, and K. H. Johansson, "Secure control systems a quantitative risk management approach," *IEEE Control Syst. Mag.*, vol. 35, no. 1, pp. 24–45, Feb. 2015.

[140] F. Pasqualetti, F. Döfler, and F. Bullo, "Attack detection and identification in cyber-physical systems," *IEEE Trans. Autom. Control*, vol. 58, no. 11, pp. 2715–2729, Nov. 2013.

[141] S. Amin, X. Litrico, S. S. Sastry, and A. M. Bayen, "Stealthy deception attacks on water SCADA systems," in *Proc. 13th ACM Int. Conf. Hybrid Syst., Comput. Control*, 2010.

[142] Y. Mo and B. Sinopoli, "Secure control against replay attacks," in *Proc. 47th Annu. Allerton Conf. Commun. Control Comput.*, 2009.

[143] S. Amin, G. A. Schwartz, and S. S. Sastry, "Security of interdependent and identical networked control systems," *Automatica*, vol. 49, no. 1, pp. 186–192, Jan. 2013.

[144] H. Li and Y. Shi, "Robust distributed model predictive control of constrained continuous-time nonlinear systems: A robustness constraint approach," *IEEE Trans. Autom. Control*, vol. 59, no. 6, pp. 1673–1678, Jun. 2014.

## ABOUT THE AUTHORS

**Stephen McLaughlin** received the Ph.D. degree in computer science and engineering from the Pennsylvania State University, State College, PA, USA, in 2014.

He is a Senior Engineer at Samsung Research America, Mountain View, CA, USA. His work is concerned with the ongoing security and safe operation of control systems that have already suffered a partial compromise. His results in this area have been published in ACM Conference on Computer and Communications Security (CCS), Internet Society Network and Distributed System Security Symposium (NDSS), and IEEE Security and Privacy Magazine. As a part of the Samsung KNOX security team, he identifies and responds to vulnerabilities in Samsung's smartphones, mobile payments, and other devices near you right now.

**Lucas Davi** received the Ph.D. degree in computer science from the Technische Universitt Darmstadt, Darmstadt, Germany, in 2015.

He is an independent Claude Shannon research group leader of the Secure and Trustworthy Systems group at Technische Universitt Darmstadt. He is also a researcher at the Intel Collaborative Research Institute for Secure Computing (ICRI-SC). His research focuses on software exploitation technique and defenses. In particular, he explores exploitation attacks such as return-oriented programming (ROP) for ARM and Intel-based systems.

**Charalambos Konstantinou** (Student Member, IEEE) received the five-year diploma degree in electrical and computer engineering from the National Technical University of Athens, Athens, Greece. He is currently working toward the Ph.D. degree in electrical engineering at the Polytechnic School of Engineering, New York University, Brooklyn, NY, USA.

His interests include hardware security with particular focus on embedded systems and smart grid technologies.

**Ahmad-Reza Sadeghi** received the Ph.D. degree in computer science with the focus on privacy protecting cryptographic systems from the University of Saarland, Saarbrücken, Germany, in 2003.

He is a full Professor for Computer Science at the Technische Universitt Darmstadt, Darmstadt, Germany. He is the head of the System Security Lab at the Center for Advance Security Research Darmstadt (CASED), and the Director of Intel Collaborative Research Institute for Secure Computing (ICRI-SC) at TU Darmstadt. Prior to academia, he worked in Research and Development of Telecommunications enterprises, among others Ericsson Telecommunications. His research include systems security, mobile and embedded systems security, cyberphysical systems, trusted and secure computing, applied cryptography, and privacy-enhanced systems.

Dr. Sadeghi served as general/program chair as well as program committee member of many established security conferences. He also served on the editorial board of the *ACM Transactions on Information and System Security* (TISSEC), and as guest editor of the IEEE Transactions on Computer-Aided Design (Special Issue on Hardware Security and Trust). Currently, he is the Editor-in-Chief of IEEE Security and Privacy Magazine, and on the editorial board of ACM Books. He has been awarded with the renowned German prize Karl Heinz Beckurts for his research on trusted and trustworthy computing technology and its transfer to industrial practice. The award honors excellent scientific achievements with high impact on industrial innovations in Germany.

**Xueyang Wang** received the B.S. degree in automation from Zhejiang University, Zhejiang, China, in 2008 and the M.S. and Ph.D. degrees in computer engineering and electrical engineering from Tandon School of Engineering, New York University, Brooklyn, NY, USA, in 2010 and 2015, respectively.

His research interests include secure computing architectures, virtualization and its application to cybersecurity, hardware support for software security, and hardware security.

**Michail Maniatakos** received the B.Sc. degree in computer science and the M.Sc. degree in embedded systems from the University of Piraeus, Piraeus, Greece, in 2006 and 2007, and the M.Sc. and M.Phil. degrees and the Ph.D. degree in electrical engineering from Yale University, New Haven, CT, USA in 2009, 2010, and 2012.

He is an Assistant Professor of Electrical and Computer Engineering at New York University (NYU) Abu Dhabi, Abu Dhabi, UAE, and a Research Assistant Professor at the NYU Polytechnic School of Engineering, Brooklyn, NY, USA. He is the Director of the MoMA Laboratory (nyuad.nyu.edu/momalab), NYU Abu Dhabi. His research interests, funded by industrial partners and the U.S. Government, include robust microprocessor architectures, privacy-preserving computation, as well as industrial control systems security. He has authored several publications in IEEE transactions and conferences, and holds patents on privacy-preserving data processing,

Dr. Maniatakos is currently the Co-Chair of the Security track at IEEE International Conference on Computer Design (ICCD) and IEEE International Conference on Very Large Scale Integration (VLSI-SoC). He also serves in the technical program committee for various conferences, including IEEE/ACM Design Automation Conference (DAC), International Conference on ComputerAided Design (ICCAD), ITC, and International Conference on Compilers, Architectures and Synthesis For Embedded Systems (CASES). He has organized several workshops on security, and he currently is the faculty lead for the Embedded Security challenge held annually at Cyber Security Awareness Week (CSAW), Brooklyn, NY, USA.

**Ramesh Karri** received the Ph.D. degree in computer science and engineering from the University of California at San Diego, La Jolla, CA, USA in 1993.

He is a Professor of Electrical and Computer Engineering at Tandon School of Engineering, New York University, Brooklyn, NY, USA. His research and education activities span hardware cybersecurity: trustworthy ICs, processors, and cyberphysical systems; security-aware computer-aided design, test, verification, validation, and reliability; nano meets security; metrics; benchmarks; and hardware cybersecurity competitions. He has over 200 journal and conference publications including tutorials on trustworthy hardware in IEEE Computer (two) and Proceedings of the IEEE (five).

Dr. Karri was the recipient of the Humboldt Fellowship and the National Science Foundation CAREER Award. He is the area director for cyber security of the NY State Center for Advanced Telecommunications Technologies at NYU-Poly; Cofounded (2015–present) the Center for Cyber Security (CCS) (http://crissp.poly.edu/), co-founded the Trust-Hub (http://trust-hub.org/) and founded and organizes the Embedded Security Challenge, the annual red team blue team event at NYU, (http://www.poly.edu/csaw2014/csaw-embedded). His group's work on hardware cybersecurity was nominated for best paper awards (ICCD 2015 and DFTS 2015) and received awards at conferences (ITC 2014, CCS 2013, DFTS 2013 and VLSI Design 2012) and at competitions (ACM Student Research Competition at DAC 2012, ICCAD 2013, DAC 2014, ACM Grand Finals 2013, Kaspersky Challenge and Embedded Security Challenge). He co-founded the IEEE/ACM Symposium on Nanoscale Architectures (NANOARCH). He served as program/general chair of conferences including IEEE International Conference on Computer Design (ICCD), IEEE Symposium on Hardware Oriented Security and Trust (HOST), IEEE Symposium on Defect and Fault Tolerant Nano VLSI Systems (DFTS) NANOARCH, RFIDSEC 2015, and WISEC 2015. He serves on several program committees (DAC, ICCAD, HOST, ITC, VTS, ETS, ICCD, DTIS, WIFS). He is the Associate Editor of the IEEE Transactions on Information Forensics and Security (2010–2014), IEEE Transactions on Computer-Aided Design (2014–present), *ACM Journal of Emerging Computing Technologies* (2007–present), *ACM Transactions on Design Automation of Electronic Systems* (2014–present), IEEE Access (2015–present), IEEE Transactions on Emerging Technologies in Computing (2015–present), IEEE Design & Test (2015–present), and IEEE Embedded Systems Letters (2016–present). He is an IEEE Computer Society Distinguished Visitor (2013–2015). He is on the Executive Committee of IEEE/ACM Design Automation Conference leading the cybersecurity initiative (2014–present). He has delivered invited keynotes and tutorials on hardware security and trust (ESRF, DAC, DATE, VTS, ITC, ICCD, NATW, LATW).