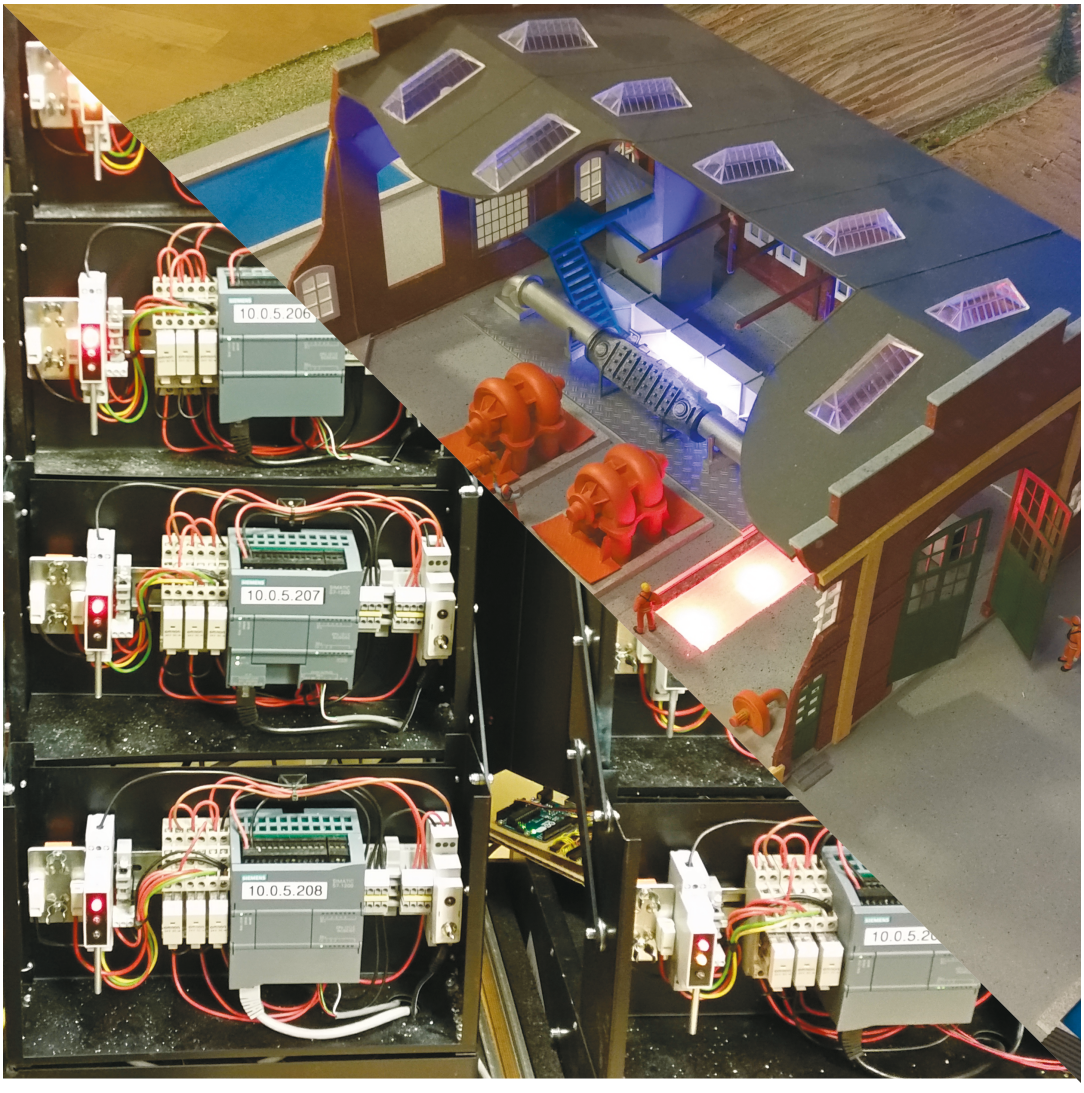
Virtual Industrial Control System Testbed

HANNES HOLM, MARTIN KARRESAND,
ARNE VIDSTRÖM, ERIK WESTRING



FOI-R--4073--SE
ISSN 1650-1942 March 2015

Hannes Holm, Martin Karresand, Arne Vidström, Erik Westring

Virtual Industrial Control System Testbed

Bild/Cover: Martin Karresand

Titel	En testmiljö för ett virtuellt industriellt informations- och styrsystem
Title	Virtual Industrial Control System Testbed
Rapportnr/Report no	FOI-R--4073--SE
Månad/Month	March
Utgivningsår/Year	2015
Antal sidor/Pages	81
ISSN	1650-1942
Kund/Customer	Myndigheten för samhällsskydd och beredskap (MSB)
Forskningsområde	4. Informationssäkerhet och kommunikation
FoT-område	
Projektnr/Project no	B341010
Godkänd av/Approved by	Christian Jönsson
Ansvarig avdelning	Informations- och aerosystem

Sammanfattning

Samhällskritiska funktioner såsom elektricitet och vattenrening är beroende av industriella informations- och styrsystem för att fungera. Fram tills nyligen bestod dessa system av specialkonstruerade isolerade komponenter. Industriella informations- och styrsystemhar har dock utvecklats på samma sätt som vårt övriga samhälle, och är nu ofta realiserade av komplexa ihopkopplade IT-system som på ett eller annat sätt är anslutna mot Internet. Detta medför att industriella informations- och styrsystem är sårbara för IT-attacker på liknande sätt som de flesta andra IT-system.

De extrema tillgänglighetskraven för industriella informations- och styrsystem i drift gör det är svårt att utföra IT-säkerhetsexperiment på dem, till exempel att leta efter sårbarheter eller testa försvarsmekanismer. För att möjliggöra sådana experiment använder praktiker och akademiker särskilda testmiljöer som är skapade för att efterlikna verkliga installationer av industriella informations- och styrsystem.

Denna studie undersöker vilka testmiljöer för industriella informations- och styrsystem som har föreslagits för vetenskaplig forskning. Särskild fokus ligger på fältutrustning, en särskild typ av komponent som anses vara synnerligen svår att integrera i testmiljöer. Studien jämför även dessa resultat med resultat från produktundersökningar, praktiska erfarenheter samt intervjuer med en tillverkare. Utfallen från dessa jämförelser är metoder och verktyg som kan användas för att skapa en naturtrogen testmiljö för industriella informations- och styrsystem.

Studien utfördes i samarbete med andra aktörer, i synnerhet Idaho National Laboratory.

Nyckelord: Industriella informations- och styrsystem, testmiljö, IT-säkerhet, cyber säkerhet, systematisk litteraturstudie

Summary

Critical societal functions such as electricity and water purification depend on Industrial Control Systems (ICS) to properly function. Not long ago, these ICS were realized by specially constructed isolated devices. Along with the rest of our society, ICS have evolved and are now often delivered by complex interconnected IT solutions including commercial-off-the-shelf technologies that in one way or another are connected to the Internet. As a consequence, ICS are vulnerable to IT attacks similarly to most other IT systems.

Due to the extreme availability requirements on ICS in operation, it is difficult to perform cyber security experiments on them, such as vulnerability discovery or tests of defense mechanisms. To accommodate such experiments, researchers and practitioners turn to testbeds that mimic real ICS.

This study first surveys ICS testbeds that have been proposed for scientific research. Special focus is given to field devices, a kind of ICS component that is considered particularly challenging to implement in testbeds. It then compares these results with findings from product surveys, practical experiences, and interviews with a manufacturer. The outcomes of this comparison are methods and tools for creating a high-fidelity ICS testbed.

The study was conducted in collaboration with other actors, in particular, the Idaho National Laboratory.

Keywords: Industrial Control Systems, testbed, IT security, cyber security, systematic literature review

Table of contents

Acronyms and abbreviations

ABI	Application Binary Interface
ACM	Association for Computing Machinery
ACORN	Automated Construction of Realistic Networks
AGA	American Gas Association
API	Application Programming Interface
APT	Access Policy Tool
ARINC	Avionics Application Standard Software Interface
ARM	Advanced RISC Machine
ASCII	American Standard Code for Information Interchange
ASIC	Application-Specific Integrated Circuit
ASSERT	Advanced System Security Education, Research, and Training
AVA	Control System Automated Vulnerability Assessment
CAN	Controller Area Network
CIM	Common Information Model
CIP	Common Industrial Protocol
CLR	Common Language Runtime
CORE	Common Open Research Emulator
COTP	Connection Oriented Transport Protocol
COTS	Commercial-Off-The-Shelf
CPU	Central Processing Unit
CSET	Cyber Security Evaluation Tool
CRATE	Cyber Range And Training Environment
DAQ	Data Acquisition
DCS	Distributed Control System
DETER	Cyber Defense Technology Experimental Research
DHS	US Department of Homeland Security
DNP3	Distributed Network Protocol
DNS	Domain Name System
FOI	Swedish Defense Research Agency
FPGA	Field-Programmable Gate Array
GUI	Graphical User Interface
HMI	Human-Machine Interface
ICCP	Inter-Control Center Communications Protocol
ICS	Industrial Control System
ICS-CERT	Industrial Control System Computer Emergency Response Team
IEC	International Electrotechnical Commission
IED	Intelligent Electronic Device
IEEE	Institute of Electrical and Electronics Engineers
IMUNES	Integrated Multiprotocol Network Emulator/Simulator
INL	Idaho National Laboratory

IP	Internet Protocol
ISA	International Society for Automation
ISO	International Organization for Standardization
IT	Information Technology
KVM	Kernel-based Virtual Machine
LAN	Local Area Network
LLVM	Low Level Virtual Machine
LM ATL	Lockheed Martin Advanced Technology Laboratories
MSB	Swedish Civil Contingencies Agency
MTU	Master Terminal Unit
NERC	North American Electric Reliability Corporation
NIST	National Institute of Standards and Technology
OB	Organization Block
OPC	Object Linking and Embedding for Process Control
OS	Operating System
OVAL	Open Vulnerability Assessment Language
PDC	Power distribution center
PLC	Programmable Logic Controller
PMU	Phasor Measurement Unit
QEMU	Quick Emulator
RDP	Remote Desktop Protocol
RESCH	REal-time SCHeduler framework
RINSE	Real-Time Immersive Network Simulation Environment
RISC	Reduced instruction set computing
RTU	Remote Terminal Unit
SCADA	Supervisory Control and Data Aquisition
SELENA	Scalable Emulation of LargE Network Architectures
SITL	System-in-the-loop
SMB	Server Message Block
SNMP	Simple Network Management Protocol
SSH	Secure Shell
STL	STatement List
TCP	Transmission Control Protocol
TSAP	Transport Services Access Protocol
UDP	User Datagram Protocol
VICS	Virtual Industrial Control System
VM	Virtual Machine
VMM	Virtual Machine Monitor
VNC	Virtual Network Computing
VPN	Virtual Private Network
VTC	Video Teleconferencing
WAN	Wide Area Network

XML	Extensible Markup Language

1 Introduction

Our society depends on various critical services such as electricity, water purification and transportation to properly function. Not long ago, the Industrial Control Systems (ICS) that supervised and controlled most of these critical services were realized by specially constructed isolated devices. Along with the rest of our society, ICS have evolved and are now often delivered by complex interconnected IT solutions including commercial-off-the-shelf technologies that in one way or another are connected to the Internet. The main reasons behind this evolution are increased functionality and increased effectiveness, as well as reduced costs. For example, IP-based remote control of railroad signaling and interlocking systems has increased the level of control of the railroad system. The benefits of using IT for critical infrastructure applications are thus clear.

However, the trend of interconnectivity and COTS has also brought about problems. Issues that are common in regular IT architectures, such as malware and misconfigurations, do now occur in ICS systems as well. Reduced availability due to such issues might be acceptable in regular IT architectures, but are generally completely unacceptable for IT that supports critical infrastructure services. For instance:

- Computers along railway tracks in Sweden send continuous data regarding the state of the track to remote railway operators. If there are more than 15 seconds between two points of data for a device, the corresponding track is considered faulty and all trains designated to traverse it are blocked [61].
- In the Energy Sector, digital protective relays are used to trip circuit breakers when power faults are detected - an event that can cause significant product damage and personnel harm. This function needs to be executed within a few milliseconds of the power fault to be of use.

To understand and manage the complexity of an IT architecture, e.g., to discover and mitigate security vulnerabilities within it, technical audits of different kinds are carried out. For instance, it is common practice to conduct penetration tests, audits that employ active network scanning and sometimes real cyber-attacks. These tests can however, due to their nature, decrease system availability in the short term. This is particularly evident for specific IT solutions used to support critical infrastructure services as these are often not able to withstand even the most basic scanning tools. For example, a study involving Programmable Logic Controllers (PLC, see Section 2.3) and the vulnerability scanner Nessus showed that the 18% of the tested PLCs crashed as a result of a scan [53]. As a consequence, technical audits are generally thought of as (at best) difficult for IT architectures that support critical infrastructure services.

1.1 Virtualization and testbeds

To study the vulnerability of IT architectures that are difficult to technically audit in the real world, many researchers attempt to copy real IT configurations and place these in isolated environments, also called *testbeds*, where experiments can be safely performed. Creating a test bed however comes with various challenges, in particular: (1) it can be difficult to achieve a realistic test bed scale, and (2) it can be difficult to achieve a realistic test bed configuration.

One way to achieve a large-scale realistic testbed is through virtualization. Virtualization is a technology which concerns isolating computer software in a means that enables layers of abstraction, both between different software and between software and hardware. For example, a virtual private network (VPN) adds a layer on top of a computer network that isolates its users from others on the network; the Comodo antivirus uses operating system-level virtualization to create a sandbox for isolated web browsing; VMware and VirtualBox use hardware virtualization to enable guest operating systems to interface with software and hardware; the Quick Emulator (QEMU) use instruction set virtualization to provide a complete emulation of computer hardware in software. Virtualizing a testbed is attractive for several reasons, for example:

- It enables running multiple parallel systems on single computer hardware.
- It enables configuration of systems and networks by the use of software scripts.
- It enables saving and loading the state/configuration of the system-of-systems.
- It isolates the activity in the testbed from the physical systems as well as external systems.

In other words, virtualization can potentially allow low-cost, replicable and safe security studies of IT architectures that have configurations valid to those of real ICSs.

1.2 Objective and research questions

The objective of this study is to *identify how a high-fidelity Virtual Industrial Control System testbed can be constructed* (hereafter referred to as VICS). The work was conducted by the Swedish Defence Research Agency (FOI) as part of a joint project in cooperation with the Idaho National Laboratory (INL) and is financed by the Swedish Civil Contingencies Agency (MSB) and the U.S. Department of Homeland Security (DHS). To meet this objective, the study first surveys existing ICS testbeds that have been proposed for scientific research and tries to answer the following five research questions (RQs):

- *RQ1: Which ICS testbeds have been proposed for scientific research?*
- *RQ2: Which research objectives do current ICS testbeds support?*
- *RQ3: How are ICS components implemented in current ICS testbeds?*
- *RQ4: How do existing ICS testbeds manage requirements?*
- *RQ5: Which methods are available for virtualizing ICS field devices?*

The first four RQs are addressed to gain an understanding of how previously constructed ICS testbeds for scientific research were designed. RQ5 follows from the fact that virtualization is a convenient way to manage testbeds and the fact that field devices (which is a critical testbed component, see Section 2.3) often have specialized hardware, software and logic that are unsupported by current virtualization technologies (such as VirtualBox, VMware or QEMU).

The answers to these five RQs are used to propose tentative means of constructing a high quality ICS testbed as well as means of measuring the fidelity of such a testbed.

The outcome of VICS is planned to be implemented in an existing testbed known as CRATE[1] (Cyber Range And Training Environment, see Appendix E) that is managed by the Swedish Defence Research Agency (FOI). CRATE has previously been used for a variety of cyber security experiments (see e.g. [40][41][85]). This is however outside the scope of this pre-study.

1.3 Related projects

The work was initiated at a meeting at Idaho National Laboratory involving all four stakeholder organizations (MSB, FOI, DHS and INL). At this meeting, the basic premises for the project were identified. These premises were then refined through a technical agreement document. The INL project that is related to VICS is known as Control System Automated Vulnerability Assessment (hereafter denoted as AVA).

The first official project meeting between FOI and INL was conducted on the 28th of November 2014 between the INL (AVA) principal investigator Craig Rieger and the FOI (VICS) principal investigator Hannes Holm. At this meeting it was concluded that there were opportunities to collaborate in several areas, and that contact should be maintained on a regular basis.

AVA underwent a feasibility pre-study during 2013 [42]. The objective of this pre-study was to "*perform a feasibility assessment on the implementation of a scalable, integrated capability for replicating ICS systems for rapid automated vulnerability assessment (AVA)*". It was sponsored by ICS-CERT and DHS, and

[1] www.foi.se/crate

carried out by INL, Lockheed Martin Advanced Technology Laboratories (LM ATL) and Draper Laboratory.

The AVA pre-study [42] focused on Supervisory Control and Data Acquisition (SCADA) and Distributed Control Systems (DSC) in the Energy Sector and other reference architectures taken from the DHS Cyber Security Evaluation Tool (CSET). The study explored a variety of different topics that are relevant to testbed creation and vulnerability assessment, including virtualization and data collection. It focused on providing snap-shot information on these topics and conclusions primarily based on expert competence.

The AVA pre-study [42] proposes to implement virtualized ICS architectures in a testbed developed by LM ATL called ACORN (Automated Construction of Realistic Networks). Implementation of some technologies in ACORN, particularly SCADA server applications and workstations, are judged easy as they build on COTS operating systems. Implementation of ICS specific devices (e.g., PLCs and RTUs) are however judged more difficult as they use specific real-time operating systems and sometimes proprietary protocols and functions. The study mentions three means of managing this issue: (1) involving developers that have emulation software for their specific ICS devices (e.g., ABB or Siemens), (2) developing a novel virtualization platform based on the Low Level Virtual Machine (LLVM) and QEMU, and (3) using real hardware platforms.

The AVA pre-study recommends data collection for a high-fidelity testbed to be performed using a combination of active (e.g., Nmap and OpenVAS) and passive (e.g., NetworkMiner and Wireshark) scanning combined with manual and offline configuration analysis.

Vulnerability assessment is recommended to be performed using off-the-shelf tools such as MetaSploit in combination with scripts developed by ICS-CERT and INL to find known vulnerabilities, and static or runtime analysis (e.g., the fuzzer developed by Wurldtech) to find novel vulnerabilities (also called “zero days”) [42].

VICS builds on the results from the AVA pre-study [42] as described in the following bullet list:

- *It provides a systematic literature review*. There are many initiatives in academia concerning ICS testbeds and virtualization. The present study complements [42] with a systematic literature review of scientific work done within the area. This is presented in Section 4.
- *It provides a second opinion on how to best create a high-fidelity ICS testbed*. In addition to providing a systematic review of ICS testbeds, the present study also serves to provide a second opinion on how to best create a high-fidelity ICS testbed. This is presented in Section 5.
- *It provides a survey of existing ICS manufacturers*. Products by different manufacturers offer different opportunities regarding testbed integration.

The present study provides an empirical survey of existing manufacturers. This is presented in Appendix A.

- *It provides a technical analysis of the Siemens S7-400 and S7-1200 PLCs*. Draper laboratories conducted physical assessments of an Allen Bradley ControlLogix 1756-L73 (Logix 5573) controller and paper studies of the Telvent Sage 303 RTU and ABB Harmony Bridge Controller with the purpose of gaining an initial understanding in the evaluation of virtualization software candidates. A researcher within the FOI project group has previously conducted physical assessments of the Siemens S7 series [99]. Experiences from these assessments complement those of the Draper laboratories. This is presented in Appendix C.

1.4 Disposition

This report is structured as follows. Section 2 provides an overview of ICS configurations and components. Section 3 provides an overview of methods that can be used to implement ICS components in a testbed. Section 4 describes a systematic literature review of existing ICS testbeds as well as means to virtualize, emulate and simulate embedded devices (e.g. PLCs)[2]. Section 5 describes means to implement a testbed as well as means for evaluate its fidelity. Finally, Section 6 concludes the report and presents possible future research directions.

There are five appendices to the report. Appendix A describes a survey of ICS field device manufacturers. Appendix B describes a more detailed description of a PLC than what is given in Section 2.3 (see footnote 2). Appendix C describes a technical analysis of the Siemens S7 series. Appendix D describes the categorization framework that was used during the systematic review. Finally, Appendix E describes CRATE, the testbed that is planned to facilitate the ICS testbed.

[2] Embedded devices are specifically addressed as these components often employ special hardware, software and logic that are considered very difficult to virtualize or emulate.

2 Industrial control systems

ICS are systems that connect the digital world with the physical world, and are for this purpose often referred to as cyber-physical systems [50]. These systems are common in critical infrastructure sectors such as the energy sector, the transportation systems sector and the water and wastewater systems sector. The configuration of an ICS depends on what sector and what context is concerned, however, most ICS systems involve similar components and architectures [90]. ICS typically involves components that enable remote monitoring and control of physical processes. If this is the case, the ICS is a so called Supervisory Control and Data Acquisition System (SCADA) [18].

An overview of a general SCADA system is presented in Figure 1 (taken from NIST 800-82 [90]). Other overviews and case studies describing SCADA configurations are given in e.g. [10][61][94]. According to [90], there are four overall areas of importance for ICS testbeds: the *control center*, the *communication architecture*, the *field devices* and the *physical process* itself. Apart from these four areas, there are also *business systems* that interact with ICS (e.g., an office network). Business systems are based on traditional IT components that for the most part can be incorporated into a testbed such as CRATE (see Appendix E) as-is. For this reason, they are left out of the scope of the present study.

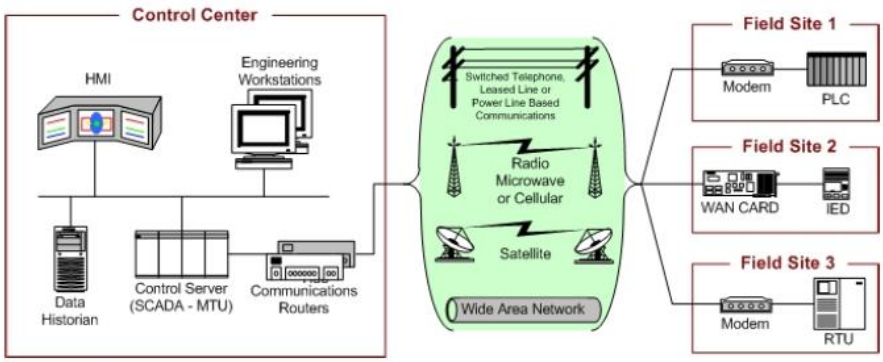


Figure 1. SCADA system general layout (taken from [90]).

2.1 Control center

The control center facilitates remote observation and control of physical processes such as voltage measurement and breaker control. This service includes, for example, systems that allow operators to interact with the process, systems that facilitate communication with field devices, systems for storing data

on the state of the process, and systems for designing the configuration of the ICS. Some important component types within the control center that are discussed by NIST 800-82 [90] are described below.

- The **Control Server** hosts supervisory control software that communicates with lower-level control devices. The control server accesses subordinate control modules over an ICS network.
- The **SCADA Server** or **Master Terminal Unit (MTU)** acts as the master in a SCADA system. RTU and PLC devices (described below) located at remote field sites usually act as slaves.
- The **Human-Machine Interface (HMI)** is software and hardware that allows human operators to monitor the state of a physical process under control as well as modify control settings and operations.
- The **Data Historian** is a centralized database for logging process information within an ICS. Information stored in this database can be accessed to support various analyses, such as statistical process control and enterprise level planning.
- The **Input/Output (IO) server** is a control component responsible for collecting, buffering and providing access to process information from control sub-components such as PLCs, RTUs and IEDs.

2.2 Communication architecture

The communication architecture enables different components within an ICS to exchange information such as control input or information updates. For example, the control center generally utilizes an Ethernet network to exchange information between control center systems, such as between HMI and MTU; similarly, the control center often utilizes modems to communicate with field devices that are located in geographically desolate places. Some types of communication architecture components that are discussed by NIST 800-82 [90] are described below.

- The **Fieldbus Network** links sensors and other devices to a PLC or other controller. Use of fieldbus technologies eliminates the need for point-to-point wiring between the controller and each device.
- The **Control Network** connects the supervisory control level to lower-level control modules.
- **Routers, switches and hubs** transfer messages within and between networks. Common uses for these devices include connecting a LAN to a WAN, and connecting MTUs and RTUs to a long-distance network medium for SCADA communication.
- **Firewalls** protect devices on a network by monitoring and controlling communication packets using predefined filtering policies. Firewalls are also useful in managing ICS network segregation strategies.

- **Modems** are used to convert between serial digital data and a signal suitable for transmission over a telephone line to allow devices to communicate.
- **Remote Access Points** are distinct devices, areas and locations of a control network that can be used to remotely configure control systems and access process data.

2.3 Field devices

Field devices process both digital and analog information through an embedded system (industrial computer) that contains both sensors and actuators. They are designed to function in harsh environments and to have extremely long time between failures: Sun et al. [93] show that manufacturers prescribe an average of 40 years, which translates to 15 years in practice according to the authors' measurements. Important kinds of field devices that are discussed by NIST 800-82 [90] are described below.

- A **Remote Terminal Unit (RTU)**, also called a remote telemetry unit, is a special purpose data acquisition and control unit designed to support SCADA remote stations. RTUs are often equipped with wireless radio interfaces to support remote situations where wire-based communications are unavailable.
- A **Programmable Logic Controller (PLC)** is a small industrial computer originally designed to execute the logic of physical hardware (e.g., relays, switches and mechanical timers/counters). PLCs have evolved into controllers with the capability of controlling complex processes, and they are common in SCADA systems. Other controllers used at the field level are process controllers and RTUs; they provide the same control as PLCs but are designed for specific control applications. In SCADA environments, PLCs are often used as field devices because they are more economical, versatile, flexible, and configurable than special-purpose RTUs.
- An **Intelligent Electronic Devices (IED)** is a sensor/actuator containing the intelligence required to acquire data, communicate with other devices, as well as perform local processing and control. An IED could combine an analog input sensor, analog output, low-level control capabilities, a communication system, and program memory in one device. The use of IEDs in SCADA and DCS systems allows for automatic control at the local level.

As can be seen in the text above, RTUs, PLCs and IEDs are similar devices – embedded devices with long lifespan that connect the digital world to the physical world. The term used to describe a field device generally depends on the context where it is applied. For example, the Swedish railroad uses Siemens S7

PLCs as transmitters/receivers of information from/to switches, which contain the actual application logic. For this purpose, they are denoted as RTUs by the rail operators.

A more detailed description of a PLC field device is given in Appendix B and a survey of PLCs is given in Appendix A. These are provided as field devices have more specialized hardware and software than what exists within the control center or the communication architecture (that primarily employ traditional COTS IT components). Furthermore, as PLCs are similar to RTUs and IEDs, this description also somewhat describes these types of components in further detail.

2.4 Physical process

There are various physical processes that are observed and controlled by ICS. A high-level overview can be given by studying the lists of critical infrastructure sectors that DHS[3] (16 sectors) and MSB[4] (11 sectors) provide. The lists provided by DHS and MSB greatly overlap and concern a wide variety of societal functions, from healthcare to commercial facilities. All of these sectors contain ICS in one way or another. However, usage of ICS is more central to the functionality of some sectors than others. While the present study does not limit itself to any single sector, its results are more valuable to these sectors. The sectors with the arguably most significant usage of ICS are described below (according to the MSB terminology):

- **Energy sector** (e.g., production and distribution of energy)
- **Municipal sector** (e.g., water distribution and wastewater management)
- **Transportation** (e.g., railroads, roads and air transportation)

The contents of this report are especially relevant for these sectors.

[3] http://www.dhs.gov/critical-infrastructure-sectors
[4] https://www.msb.se/sv/Forebyggande/Samhallsviktig-verksamhet/Om-samhallsviktig-verksamhet/

3 Integrating components in testbeds

This chapter describes three methods that can be used to implement ICS components in testbeds:

- *virtualization* (execute an existing platform in a virtual container),
- *simulation* (build a new platform that mimics the desired platform) and
- *hardware* (i.e., use the physical platform suggested by a vendor).

These methods are described in Section 3.1-3.3. A summary and a description of the terminology that is used in this report are given in Section 3.4.

3.1 Virtualization

Computer virtualization was initiated during the 1960s by IBM to provide concurrent, interactive access to mainframe computers in the form of Virtual Machines (VM) [17]. The purpose was to enable time- and resource-sharing of these mainframes within isolated copies of the underlying system without altering the end-user experience of interacting with a physical machine. Interest in virtualization then declined during the 1970s and 1980s when hardware got less expensive, but regained popularity during the 1990s along the release of a wide variety of hardware and operating systems – a trend that is continuing even now. [62]

There are several kinds of virtualization that have very different functionality and are applicable in different settings. For example, the Java Runtime Environment and the VMware workstation are both virtualization technologies, but serve highly different purposes. This diversity has created considerable confusion within academia and industry, which sometimes view virtualization as simply VMware or VirtualBox without considering how it actually works.

The purpose of this chapter is to introduce the reader to the concept of virtualization and describe how it is defined in the present study. The following overall definition of virtualization is applied in this report [62]:

> "*Virtualization is a technology that combines or divides computing resources to present one or many operating environments using methodologies like hardware and software partitioning or aggregation, partial or complete machine simulation, emulation, time-sharing, and many others*."

Nanda and Chiueh [62] present a survey of virtualization methodologies[5]. These methods are presented and related to the scope of the present report in sections 3.1.1-3.1.3.

3.1.1 Emulation

Emulation has the advantage that any kind of guest operating system with any kind of hardware requirements (in theory) can be executed on completely different physical hardware. For example, an Android-based smartphone running an ARM architecture could be emulated on a Windows-based PC running an x86 architecture. Emulation however has the disadvantage that there is a need to construct the translation framework. This is expensive and difficult with known architectures (e.g., ARM), and especially so for proprietary architectures (e.g., what is used by the Siemens Simatic S7 PLC). For proprietary cases, the instruction set has to be completely reversed, which can be a very troublesome task (see Appendix C). Emulation can also have a significant decrease in performance for the guest due to costly translation operations. Common emulation technologies are QEMU, Boch and Crusoe.

3.1.2 Hardware virtualization

Hardware virtualization involves allowing the guest machine to execute some instructions directly on hardware, whereas other instructions are trapped, translated by a Virtual Machine Monitor (VMM) and then executed on hardware (in a similar fashion to emulation) [28][62]. A typical instruction that is translated by the VMM is when there is a write to the kernel space memory space; a typical instruction that is executed directly by hardware is a read from user space memory.

Paravirtualization is a special case of hardware virtualization, where the guest operating system must be modified before it is able to run in the virtual machine. Such a VMM provides some way for the guest operating system to make special calls into it, asking for various tasks to be performed.

Hardware virtualization is typically divided into *hosted operating system* and *bare-metal* virtualization [62]. These technologies are described in the following two subsections.

[5] There are other relevant works, such as the taxonomy by Smith and Nair [84], the survey by Gu and Zhao [29], and the analysis by Robin and Irvine [74]. This report focuses on [62] as it was judged best suited for its scope; however, it includes other relevant work where necessary.

3.1.2.1 Hosted operating system

Hosted operating system virtualization concerns when the VMM runs on top of a host operating system (e.g. Windows 7). I/O operations that are trapped by the VMM are delivered to hardware through the host operating system. Examples of hosted operating system solutions include VMware workstation, VirtualBox and Microsoft Virtual PC. [62]

3.1.2.2 Bare-metal operating system

Bare-metal operating system virtualization involves when the VMM runs directly on hardware. This technology is more complicated than hosted operating systems as the VMM (rather than the host) has to handle all I/O instructions. It however in return enables a higher performance. Example bare-metal solutions include Xen, L4 and VMWare ESX. [62]

3.1.3 Other virtualization approaches

Three other forms of virtualization are discussed by Nanda and Chiueh [62]: operating system virtualization, programming language virtualization and library virtualization. These approaches are discussed next.

3.1.3.1 Operating System virtualization

Operating system virtualization involves creating a virtual copy of the hosting physical machine, but without the demand to setup a completely new machine (which is required for hardware virtualization). This technology can be provided either by the operating system itself, such as the FreeBSD Jail and Linux Kernel-based virtualization, or by third-party software such as the Comodo Internet Security Sandbox (that creates a virtual machine for “safe” web browsing) or Ensim (that virtualizes the machine’s native operating system into isolated and independent computing environments). [62]

3.1.3.2 Programming Language virtualization

Programming language virtualization involves a virtual machine that supports a set of predefined instructions. The two most common forms of programming language virtualization are the Java virtual machine and the Microsoft .NET Common Language Runtime (CLR). [62]

3.1.3.3 Library virtualization

Most applications use application libraries with various application programming interfaces (API) and/or application binary interfaces (ABI). Library virtualization builds on this fact by providing such library functions to environments that not normally supports them. The most common example of library virtualization is Wine. Wine is an implementation of the Windows API and can be used as a

library to execute Windows applications in Unix environments. Other examples are WABI, LxRun and Visual MainWin. [62]

3.2 Simulation

Simulation involves creating a model of a process or system that can be used for experimentation and evaluation in order to understand the behavior of the system and/or evaluate strategies for operating the system [66][78].

The purpose of simulation is thus not to enable execution of an existing platform (as virtualization), but rather to build a platform that mimics critical aspects of the desired platform. Simulation is used for a plethora of purposes, for instance, to mimic TCP/IP communication [96], power grids [63] and manufacturing processes [48].

3.3 Hardware

Using real hardware (i.e., the actual hardware suggested by a vendor) within a testbed naturally provides very high fidelity, but is in return very expensive. For example, reaching scale is costly, restoring configurations is difficult, and devices are bound to occasionally get bricked due to failed exploits.

3.4 Summary and terminology

Of the methods described in sections 3.1-3.3, hardware **virtualization** is desired as it enables high-performance execution of real applications in virtual containers. *To increase the readability of this report, hardware virtualization is hereafter simply denoted as virtualization*. **Emulation** also enables execution of real applications, but is slower than virtualization. **Simulation** is less desirable as it is expensive to develop new applications and models, and their fidelity is uncertain. **Hardware** is very expensive and cumbersome to employ and thus generally considered a last resort.

4 Systematic literature review

A systematic review of past scientific work was conducted to enable answering the research questions. The review follows the guidelines of Kitchenham [47], which has been used for several software related reviews in recent years.

The method of this review is described in Section 4.1; an overview of identified ICS testbeds (RQ1) is given in Section 4.2; testbed objectives (RQ2) are presented in Section 4.3; how testbeds have been implemented (RQ3) is described in Section 4.4; how testbed requirements are managed (RQ4) is presented in Section 4.5; how field devices can be virtualized (RQ5) is described in Section 4.6.

4.1 Search method

The review began with unstructured searches related to the topic with the purpose of identifying relevant keywords for systematic searches. As a pilot study, Scopus[6] was queried for articles with the chosen keywords[7] within their titles, keywords or abstracts, yielding a total of 123 matches.

The relevance of each of these 123 articles was independently judged based on title and abstract by randomly chosen pairs of researchers. Redundant judgments were used to enable measuring the group's internal agreement with the statistical metric Cohen's Kappa [15]. The results showed strong agreement[8], which is a sign that the group shares the same view on the project scope.

Twelve of the 123 identified articles, as well as fifteen articles cited by these works, were deemed as relevant and read in detail. This activity amounted to a categorization framework that was based partly on information about ICS [90], partly on information about virtualization [28][29][62][74], and partly on information provided by the articles themselves. This activity was also conducted with two reviewers per article, with the purpose of measuring the agreement regarding the employed categorization framework. Cohens Kappa indicated strong agreement also for these aspects (a Kappa of 0.83). The final categorization framework is presented in Appendix D.

[6] SCOPUS is a database that aggregates articles from most conferences and journals such as IEEE, ACM, Springer, Elsevier and Wiley.

[7] security AND (scada OR ics OR "smart grid" OR mtu OR plc OR rtu) AND (virtual OR simulat OR emulat)

[8] A Kappa of 0.88 on a scale from 0 (no agreement) to 1 (complete agreement).

A second more comprehensive review was conducted using Scopus and a set of refined keywords[9] that reflect the above mentioned two topics in a similar method to the pilot. This review identified 1335 articles. The relevance of these articles was judged by their abstracts and titles in the same means as during the pre-study, but due to the strong agreement (as shown by Cohen's Kappa) without redundant judgments. Out of the 1335 articles, 63 were judged as relevant and read in detail[10]. Of these articles, 52 articles were judged relevant after the more detailed review. Data were extracted from these articles based on the categorization framework presented in Appendix D. Of the 52 relevant articles, 40 concerned ICS testbeds and 12 concerned virtualization of embedded devices. The results from this literature review are presented in the following sections.

4.2 Overview of current ICS testbeds

The systematic literature review identified a total of 40 articles that concerned 30 ICS testbeds that were planned or currently operational at the time of the present study. An overview of these testbeds is described in Table 1.

As can be seen, almost half of the identified testbeds were located in the USA. Five testbeds were only planned ([12], [22], [27], [44] and [98]), while the remaining 25 were claimed to be operational to an extent that facilitated technical studies related to their stated purposes. It should be mentioned that there are various other testbeds, such as DETER [5] and the U.S. National SCADA testbed (that is run by the U.S. Department of Energy), that were not directly identified by the systematic review. There are two explanations behind this: (1) they had either not published their results in forums indexed by Scopus or (2) did not specifically concern ICS. The U.S. National SCADA testbed corresponds to the prior explanation; DETER is not a testbed that has been designed for the purpose of ICS tests and thus corresponds to the latter explanation. The testbeds that employ DETER, such as the testbed at the Technical Assessment Research Lab in China [25], view DETER as a tool that help realize an ICS testbed (similar to Matlab, OPNET or VirtualBox). The present report views DETER and other similar testbeds (e.g., Emulab, GENI and PlanetLab) in the same fashion as the ICS testbeds that use them.

[9] (scada OR ics OR mtu OR plc OR rtu OR IO OR "embedded device" OR "embedded system") AND ((virtuali OR simulat OR emulat OR hypervi OR VMM OR "virtual machine" OR "dynamic recompilation") OR (testbed OR "test bed" OR "cyber range"))

[10] Most of the articles judged as irrelevant concerned implementing the Java Virtual Machine on embedded systems or Power Line Communication.

Table 1. Overview of ICS testbeds.

ID	University/Organization	Country	References
1	American University of Sharjah	Abu Dhabi	[19]
2	Queensland University of Technology	Australia	[49]
3	RMIT University	Australia	[2],[70]
4	Research Institute of Information Technology and Communication	China	[98]
5	Technical Assessment Research Lab	China	[25]
6	Tsinghua University of Beijing	China	[14]
7	University of Zagreb	Croatia	[44]
8	Queen's University Belfast	Ireland	[102]
9	University College Dublin	Ireland	[88]
10	European Commission Joint Research Centre	Italy	[30],[83]
11	European Commission Joint Research Centre	Italy	[23]
12	Ricerca sul Sistema Energetico	Italy	[21]
13	American University of Beirut	Lebanon	[76]
14	University Kuala Lumpur	Malaysia	[80],[81]
15	TNO	Netherlands	[12]
16	ITER Korea	South Korea	[91]
17	Case Western Reserve University	USA	[58]
18	Iowa State University	USA	[33],[34]
19	ITESM Campus Monterrey	USA	[75]
20	Lewis Research Center	USA	[4]
21	Mississippi State University	USA	[59],[60],[71], [72],[97]
22	Ohio State University	USA	[31]
23	Pacific Northwest National Laboratory	USA	[22]
24	Sandia National Laboratories	USA	[95]
25	Tennessee Technological University	USA	[89]
26	The University of Tulsa	USA	[35]
27	UC Berkeley	USA	[27]
28	University of Arizona	USA	[55]
29	University of Illinois at Urbana-Champaign	USA	[6],[7],[20]
30	University of Louisville	USA	[37]

4.3 Objectives of ICS testbeds

An overview of the objectives that the creators of the testbeds present is given in Table 2. The most commonly mentioned objective is to use a testbed for

vulnerability analysis, with education and tests of defense mechanisms on a split second place. These objectives highlight the fact that most testbeds focus on cyber security rather than, for instance, performance issues due to UDP packet loss.

Table 2. Objectives of testbeds.

Objective	Testbeds
Vulnerability analysis	16
Education	9
Tests of defense mechanisms	9
Power system control tests	4
Performance analysis	1
Creation of standards	1
Honeynet	1
Impact analysis	1
Test robustness	1
Tests in general	1
Threat analysis	1

These objectives are in general described on a very superficial level. For example, the type of vulnerability analysis that is proposed is typically described with generic statements such as "*It is imperative to analyze the risk to SCADA systems in terms of vulnerabilities, threats and potential impact*" [12] and "*An evaluation of the security of SCADA systems is important*" [2]. However, as stated by Davis et al. [20], vulnerability analysis is not a simple matter:

> "*Determining the vulnerabilities of systems using these devices is a complicated process because of the complex hardware and software interactions that must be considered*"

As vulnerability analysis is a broad and difficult topic, there is a need to break it down into more tangible topics in order to yield useful testbed requirements. The same reasoning applies for other objectives, such as education and tests of defense mechanisms. This is discussed in Section 6.3.

4.4 Implementation of ICS testbed components

This section describes how the control center, communication architecture, field devices and observed/controlled process are implemented in the 30 surveyed testbeds. An overview of the results is described by Table 3. More detailed descriptions are provided in sections 4.4.1-4.4.4.

Table 3. Number of articles assessing different areas and methods of implementation (virtualization, emulation, simulation and hardware).

Area	Covered	Virtualization	Simulation	Emulation	Hardware
Control center	20	4	9	1	11
Communication architecture	22	6	10	3	11
Fields devices	23	0	14	0	14
Physical process	12	0	12	0	0

An overview of the product types that are described in articles concerning the 30 analyzed testbeds is given in Table 4. As can be seen, various abstraction levels and components are mentioned. The most commonly mentioned types of components are RTU, MTU, PLC, HMI and IED. These are all mentioned for more than one testbed. It is worth mentioning that these definitions are rather vague, especially to practitioners. For example, the Swedish railroad has Siemens S7 PLCs that are connected to switchgear. The purpose of these PLCs is to package/unpackage the proprietary data that the switchgear sends and receives by the MTU. For this reason, the Siemens S7 PLCs are denoted as RTUs by operators of the Swedish railroad (as they have a specific purpose).

Table 4. Overview of product types in testbeds.

Products	Testbeds
RTU	12
MTU	8
PLC	8
HMI	7
IED	4
DAQ	1
Data aggregator	1
HDBMS	1
OPC server/client	2
PDC	1
PMU	1
Relay	1
SCADA server/client	1
Not covered	13

There are several components in NIST 800-82 [90] that are not explicitly mentioned for any testbed. In particular, the data historian, IO server and control

server are not mentioned. The articles do not describe why this is the case. An explanation could however be that these components are thought of as integrated with the MTU.

An overview of the communication protocols that are used by the testbeds is given in Table 5. Modbus (Modbus ASCII, Modbus TCP or Modbus RTU) and DNP3 are by far the most commonly mentioned. OPC, IEC 60870 (including e.g. IEC 104), IEC 61850 (including e.g. GOOSE) and Profibus are also mentioned for more than one testbed. According to the American Gas Association's AGA-12 standard [1], there are between 150 and 200 SCADA protocols. There are thus a plethora of protocols that are not covered by current testbeds. How common these protocols are in practice is however unknown to the authors of this report.

Table 5. Overview of protocols in testbeds.

Protocol	Testbeds
Modbus	13
DNP3	12
OPC	5
IEC 60870	4
IEC 61850	3
Profibus	2
Fieldbus	1
FINS	1
GOOSE	1
ICCP	1
IEEE C37.118	1
CIP	1
RJ45	1
DeviceNet	1
Genius	1
Not covered	9

4.4.1 Control center

The control center concerns the servers and operator stations that are used to remotely observe and control field devices, such as MTU and data historian (see Section 2.1). An overview of how control center components are incorporated in testbeds can be seen in Table 3. Approximately two thirds of all testbeds contain descriptions regarding how their control center components are incorporated. Of

these, most utilize simulations (30%) and/or hardware (37%). It is interesting that so few (13%) testbeds choose to virtualize the control system components, something which to a large extent is possible as they typically involve COTS OS such as Windows and Linux (see Section 5.1).

The virtualization solutions that are mentioned concern DETER, Emulab, GENI, PlanetLab and VirtualBox. Simulation-based approaches concern LabVIEW, Mathworks Simulink, HoneyD in combination with IMUNES (FreeBSD jails), the RINSE network simulator and custom Python scripts. The emulation approach involves RINSE (it combines emulation and simulation). Hardware concerns standard x86-based computers such as CitectSCADA 6.1 on Windows XP (used as OPC server and HMI).

4.4.2 Communication architecture

The communication architecture involves components that realize communication within ICS, for instance, routers, switches and modems (see Section 2.2). 73% of all testbeds contain descriptions regarding how their communication architecture is incorporated. Of these, most utilize simulations (33%) and/or hardware (37%). As for control systems, many kinds of communication architectures are possible to easily virtualize. For example, Ethernet is commonly used within ICS and is easily virtualized through e.g. VirtualBox (see Section 5.2). Thus, it is interesting that few testbeds (20%) choose to do so.

Virtualization is proposed using DETER, GENI, Emulab or Virtualbox. Simulation is proposed using OPNET, SITL communication network simulator, Iperf (for background traffic), RINSE, OMNET++, PowerWorld simulator, Mathworks Simulink, the Inet framework, NS-2, Networksim, the c2windtunnel framework, IMUNES, and custom Python scripts. Emulation is proposed using CORE (in combination with OpenVZ) and RINSE. Hardware generally involves Ethernet devices such as routers and switches.

4.4.3 Field devices

Field devices concern the components that link the physical world to the digital world, for instance, a PLC or an RTU (see Section 2.3). 77% of the testbeds contain descriptions on how field devices are incorporated – a higher number than for control system, communication architecture or process. None of the testbeds contain virtualized or emulated[11] field devices. An explanation for this

[11] One testbed claims to utilize PLC emulation software (the RSEmulate from Allen-Bradley). Based on our paper studies, this software however simulates rather than emulates a PLC.

result is that ICS field devices generally are based on specialized, sometimes proprietary, hardware and software that are unsupported by common virtualization and emulation tools. Simulation (47% of all testbeds) and hardware (47% of all testbeds) are used instead.

Used simulation tools include STEP7 (of Siemens S7 PLCs), RSEmulate (by Allen-Bradley), LabVIEW, Scadapack LP PLC, Modbus Rsim, Soft-PLC, Python scripts with CORE, OpenVZ, PowerWorld server, and HoneyD in combination with IMUNES (FreeBSD jails). Hardware includes, for example, Allen Bradley Control Logix PLC, National Instruments NI-PXI, Omron PLC CJ1M-CPU11-ETN, CompactRIO from National Instruments, ABB 800F, Siemens OpenPMC, Siemens S7 PLC, Emerson Ctrl MD, and GE FANUC Rx3i.

4.4.4 Physical process

The physical process concerns the physical reality that the ICS observe and control (see Section 2.4). Less than half of the testbeds describe how the process is implemented. In all cases, implementation builds on simulation models (rather than actual physical processes).

The simulation approaches build on Matlab, Mathworks Simulink, Power Hardware-in-the-Loop (OPAL-RT), LabVIEW, PowerWorld, AnyLogic and EZJCOM, ANSYS, real time digital simulators, an Abacus solar array simulator, a library file (.dll) for EPANET, OMNET, and a custom application written in Java.

4.5 Managing testbed requirements

Siaterlis et al. [82] describe four overall requirements that cyber security testbeds should fulfill:

- **Fidelity**: Reproduce as accurately as possible the real system under study.
- **Repeatability**: Repeating tests produces the same or statistically consistent results.
- **Measurement accuracy**: Observing tests should not interfere with their outcome.
- **Safe execution of tests**: Cyber security tests often involve adversaries that exploit systems using malicious software. As it can be difficult to know the outcome of these activities beforehand, tests must ensure that the activity within the testbed is isolated.

Of these requirements, repeatability and measurement accuracy generally depend on activities outside of the technical scope of a testbed. For example, it is difficult to ensure that adversaries act in the same way during consecutive tests.

For this reason, repeatability and measurement accuracy are excluded from the scope of the present pre-study[12]. Safe execution of tests was a key topic during the development of CRATE (the testbed that is planned to host VICS, see Appendix E), and is thus already rather mature. For this reason, it is also excluded from the scope of the pre-study.

Ensuring testbed fidelity, i.e., that a testbed accurately reflects the desired real environment(s), is a critical task as the quality of any data produced from interaction with the testbed otherwise is uncertain. More than half (63%) of the testbeds are not discussed at all regarding fidelity (see Table 6). The remaining testbeds are analyzed in respect to fidelity in two different means: practical experiences and/or standards. The fidelity of 23% of the testbeds is argued based on real data gathered by the authors: either from quantitative data gathered from ICS systems in operation and/or from qualitative personal experiences or discussions with ICS manufacturers, providers and operators. For instance, "*Based on discussions with some industry partners and on our own experience*" [2] and "*In order to capture real image of the power network, a small part of power network was taken*" [19]. The remaining 13% that discuss fidelity base their testbed designs on standards developed by NIST (e.g., the NIST 800-82), ISA (e.g., the ISA-99) or IEC (e.g., the IEC Smart Grid Standardization Roadmap).

Table 6. Analysis of testbed fidelity.

Fidelity	**Testbeds**
Not covered	19
Study of real systems	7
Based on standards	4

Of the testbeds that are discussed in terms of fidelity, two provide specific metrics that can be used to replicate their results with some degree of accuracy. The first is Reaves and Morris [71] (a testbed at the Mississippi State University), who describe 11 metrics involving Modbus traffic (e.g., byte throughput, master-to-slave inter-arrival time, error count and packet size) in addition to comparing the result from attacks against testbed components (which in this case are simulated) compared to real components. These metrics were chosen based on the rule sets of model-based intrusion detection systems. The second is Siaterlis and Genge [83], who compare the execution time of their testbed to the required execution time of seven physical processes. Their results show that they fulfill the execution time for everything but the IEEE 118 bus model (the testbed has an execution time of 155ms and the IEEE bus system has a requirement of 24ms).

[12] They will however be studied in further detail in future work.

An important aspect of testbed fidelity concerns what data should be collected in order to recreate a valid testbed design. For example, how a network topology or machine configuration best should be captured. Of all testbeds, the Iowa State University testbed is the only one that discusses this topic [33]. Hahn and Govindarasu [33] discuss how different data collection tools are able to fulfill the NIST 800-115 [77] methodology and the NERC critical infrastructure protection requirements. They used Wireshark to analyze network traffic, The Open Vulnerability Assessment Language (OVAL) Interpreter for analyzing machine configurations, Nmap and Sandia's Antfarm for network and service discovery, Firewalk and the access policy tool (APT) for firewall rule set discovery, and Nessus for vulnerability scanning. The results showed that these tools overall had excellent support for regular IT solutions such as Windows operating systems, but poor support for ICS specific components such as PLCs. For instance, "*there appeared to be numerous communications employing proprietary protocols which Wireshark was unable to identify*" and "*Nmap was not able to identify 53 out of 157 the open ports utilized in the network. This occurrence is a result of the heavy utilization of proprietary and SCADA specific protocols which are not recognized by Nmap*". The analysis by Hahn and Govindarasu [33] is also limited as it does not study the potential to collect configuration data through agent based software, which is a common ICS industry practice (see Section 5.1).

4.6 Virtualization of embedded devices

The systematic literature review identified 12 articles that describe methods for virtualizing or simulating embedded devices. Of these articles, three present surveys (Section 4.6.1) and nine present implementations (Section 4.6.2).

4.6.1 Surveys of embedded device virtualization

Heiser [36] provides a number of examples of use cases of virtualization in embedded systems, and explain the motivation and benefits, as well as some of the differences to virtualization of non-embedded systems. For example, it is explained how virtualization enables data privacy layers in modern smartphones. However, the author does not discuss ICS field devices.

Gaska et al. [26] describe a survey of technologies and methods that can enable virtualization of avionics applications requiring multiple guest OS environments. The results show that there are many questions regarding embedded system virtualization that still are unanswered, for example, "*How will IT virtualization, ARINC 653 virtualization, and MILS/MLS accommodate multicore with 32 processor devices on chip?*".

Gu and Zhao [29] present a survey of existing virtualization technologies for real-time embedded systems. For example, one section presents Xen-based

solutions and another presents KVM-based solutions. The author does however not provide any in-depth discussion of any of the surveyed solutions. The most relevant part of [29] for the present study is a survey of virtualization technologies for “safety-critical systems”. Whether or not these surveyed virtualization technologies support field devices is however unknown.

4.6.2 Virtualization implementations

Zamorano and Puente [105] describe ASSERT, a virtualization methodology and platform that can be used to develop and implement real-time embedded systems. It however only supports the LEON processor family. It is also unclear how much effort it would require to build software for it, or how valid a developed embedded system would be.

Xu et al. [101] propose handling binary translation requests based on the page fault mechanism in the Linux kernel. The authors test their approach using the ARM platform and evaluate its performance in relation to emulating ARM using the Boch emulator. The results show that the authors approach is more efficient than the Boch emulator.

Yoo et al. [104] argue that the I/O model of current virtual machine monitors (e.g., Xen) is not suitable for real-time applications as it lacks predictability and does not guarantee deterministic I/O processing. The authors then propose a method for VMM resource scheduling given the presence of several VM guest OSs. Measurements by the authors show that their approach is promising yet requires work.

Sisu et al. [100] introduce a real-time multicore VM scheduling framework for the Xen VMM. Their approach primarily addresses the Xen scheduling issues in a similar fashion to Yoo et al. [104]. It does not address the problem that few operational field devices are possible to virtualize in Xen. The authors estimate the effectiveness of this approach through a resource scheduling experiment and find, similarly to Yoo et al. [104], that their approach is promising yet requires more work.

Similarly to Yoo et al. [104] and Sisu et al. [100], Åsberg et al. [3] address the timing issues of embedded virtualization. The authors approach does not require any kernel modifications, something which is accomplished by a scheduling framework called RESCH in combination with a type-2 hypervizor such as VirtualBox or VMware. Their approach does however not support embedded systems with special hardware and software not covered by common type-2 hypervizors.

Chunjie and Hui [13] address the heterogeneity of manufacture-dependent PLC programming languages by implementing the IEC 61131-3 standard in a virtual machine. This enables porting of IEC 61131-3 compatible applications to other

hardware platforms. Consequently, it is a means to standardize PLC programming rather than virtualize existing PLCs. The authors implement their approach on a C51 based embedded PLC platform and measure the execution time of different PLC instructions. A similar approach is presented by Zhang et al. [106], who also propose implementing IEC 61131-3 in a virtual machine (based on uCLinux). The authors do not attempt to estimate the effectiveness of this approach.

Zhang et al. [107] propose a virtualization approach for cyber-physical systems that build on QEMU and the source code of a controller to emulate a PLC and Matlab/Simulink to simulate a physical process. Thus, it is a kind of simulation rather than a means to virtualize or emulate existing field devices. QEMU and Matlab/Simulink are designed to communicate through a socket. They implement their approach for two real cyber-physical systems and find that their simulation is close to the real systems.

Son and Lee [86] has built a cross-platform virtualization software for embedded devices that is able to run code that has been compiled for its instruction set (a kind of programming language virtualization, see Section 3.1.3.2). The authors have created a compiler that can manage source code written in C, C++, Java or Objective-C. The effectiveness of this approach is however not examined.

4.6.3 Summary

As is presented in Section 4.6.1 and Section 4.6.2, there are a number of implemented virtualization technologies for embedded systems. However, none enables executing a real field device (such as a Siemens S7-1200) within a virtual or emulated container. Together with the fact that not a single of the identified testbeds virtualize or emulate field devices (see Section 4.4.3) this is a clear indication of the difficulty associated with virtualization and emulation of field devices.

5 Creation of an ICS testbed

This chapter describes how the control center, communication architecture, field devices and the physical process itself might be incorporated into an ICS testbed (sections 5.1-5.4).

The systematic review of existing testbeds identified very few tangible measures for fulfilling testbed fidelity (see Section 4.5). As a consequence, this chapter builds on additional literature reviews as well as practical experiences of ICS components and configurations in the project group in order to identify tentative means of creating a high-fidelity testbed:

- **Literature reviews**: The results from the systematic review (see Section 4) in combination with additional searches for specific works not covered by it (e.g., regarding network fidelity metrics, see 5.2.2).
- **Interviews with an ICS manufacturer**: Interviews with personnel at ABB Ventyx in Västerås (a product manager, an IT security architect and a lead developer) were conducted to assess the possibility to include ABBs SCADA system components in an ICS testbed, as well as improve the understanding of ABBs components and configurations.
- **First hand experiences with ICS components and configurations**: FOI has experience from testing, reversing, implementing and developing various ICS components and configurations (see e.g., Appendix C or [61]).

On an overall level, the amount of fidelity that is necessary depends on the objectives of the testbed. The systematic review suggests that the possible objectives are vulnerability analysis, education and tests of defense mechanisms (see Section 4.3).

All three objectives put one overall requirement on testbed fidelity: *The testbed should appear realistic when observed and interacted with*. This includes both the ability to appear realistic during normal usage and the ability to appear realistic given the presence of cyber attacks. Here, cyber attacks are defined according to [43], [57] and [79]: as a combination of **asset discovery** (e.g., mapping network topology and fingerprinting systems and services), **exploits** (e.g., an attack code that provides administrator privileges of a system through a buffer overflow vulnerability) and **activity as a result of system compromise** (e.g., downloading additional files or manipulation of data). In summary, this means that an ICS testbed should fulfill the following four general requirements:

1. The testbed should facilitate interaction between control center, field devices and the physical process using the same protocols and with the same outcome as a real ICS in operation.

2. An automated network scan should provide the same results (topology, systems, software and vulnerabilities) as if it was performed on the real ICS.
3. Sniffing network traffic should provide the same results as if it was done on the real ICS.
4. Exploits should provide the same results as if run on the real ICS.

A possible fifth requirement is that an ICS user (e.g., an operator) should not be able to tell the difference between testbed interfaces and real interfaces (e.g., the graphical user interface [GUI] of an HMI). However, none of the reviewed testbeds (including those addressing education) focuses on high-fidelity GUIs. Thus, it is not judged as necessary (although preferable) for an ICS testbed.

5.1 Integrating the control center

This section describes how control center components can be implemented in an ICS testbed (Section 5.1.1) and how the fidelity of these components can be managed (Section 5.1.2).

5.1.1 Method of implementation

Hardware and software within the control center generally build on "traditional" IT components such as Windows and Linux. For example, older variants of ABBs SCADA system Network Manager are run on physical Windows and Linux machines (e.g., Windows XP); newer Network Manager systems are run on virtualized Windows and Linux machines (e.g., Windows 7). Next to all control center hardware and software are thus supported by common virtualization technologies such as VirtualBox and thus possible to integrate in a testbed as-is. It is thus curious to why current testbeds choose to simulate control center components rather virtualize them. One explanation could be that it is difficult to obtain real control center components.

5.1.2 Managing fidelity

To yield a high-fidelity testbed in respect to the control center, it is necessary to be able to replicate ICS control centers in operation. This replication should concern three areas:

1. *machine configurations* (which operating systems that exist and what applications that are installed on them),
2. *application configurations* (how installed applications are configured), and
3. *application interactions* (how installed applications are used and communicate with other connected systems and software).

Machine configurations can be obtained by cloning hard drives, through automated network scanners such as Nessus or Nmap, or by feeding network traffic into special parsers such as NetworkMiner. A study by Holm et al. [39] of automated network scanners indicate that their accuracy greatly depends on whether or not they are allowed to login to the probed systems: their accuracy given login was 100% for operating systems (e.g. Windows 7), 92% for application servers (e.g. Apache Webserver) and 100% for application clients (e.g. Adobe Reader); the accuracy without login was 62.5% for operating systems, 67.3% for application servers and 0% for application clients. In another study, Holm et al. [41] focus on the ability of network scanners to correctly identify software vulnerabilities. The results show that a mean of 41% (given login) and 17% (without login) of all existing vulnerabilities are correctly identified by the tested scanners, and that 6% (given login) and 7% (without login) of all reported vulnerabilities actually are false alarms (i.e., do not exist in reality). The results described in [39] and [41] however concern "traditional" IT systems such as Linux and Windows OSs and Apache webservers. As reported by Hahn and Govindarasu [33], the accuracy would likely be significantly lower when probing ICS-specific software or sniffing ICS-specific protocols.

Application configurations can be obtained by cloning hard drives or by software agents that execute client-side code. For example, ABB has specially constructed scripts that allow replicating control centers. Services that extract application configurations can be agent-based (run locally on machines) or interact with machines through remote access services such as telnet, SSH, VNC or RDP.

Application interactions are more difficult to assess than machine and application configurations. This is the case as state transitions typically are not stored per default (e.g., the sequence of steps used by an individual when responding to an email). Some application usage can be captured with network sniffers (e.g., how individuals interact with shared folders); other application usage requires live observation of systems in operation using e.g. screen capturing software or memory dumps.

All of these aspects (application configurations, machine configurations and application interaction) are necessary to enable high fidelity in respect to "regular" control center operations (without considering any adversary).

It should be easy to provide an attacker with a realistic experience when attacking virtual control center components (such as probing or exploiting a Windows machine) as automated scanners (e.g., Nmap or Nessus) provide the same result as if scanning a physical system and exploits generally works the same on a virtual system as if run against a physical system. For example, application modules are loaded on the same logical address spaces no matter if the underlying OS is virtualized or not.

If the objective of the testbed is to function as a honeypot (deceive real attackers), virtualization is arguably only applicable given an attacker profile of a novice. Experienced attackers can typically easily fingerprint the usage of virtualization technologies such as VirtualBox as the guest OS requires special drivers, e.g., to interact with the virtual devices exposed by the virtual machine manager or to enable the VMM to impose special instructions on the guest OS (e.g., a shared clipboard between host and guest).

Apart from the issues concerning virtual machine fingerprinting, a control center contains various kinds of IT applications and protocols that trigger communication messages based on different activities. For example, SMB triggers based on access of (network) shared folders or printers, ARP requests and responses are used to map MAC addresses to IP addresses, and DHCP requests and responses are triggered when new system is connected to a LAN and when its IP lease timer has expired. A testbed should well reflect such traffic to be considered of sufficient fidelity [8]. High-fidelity network traffic is arguably of greater importance than high-fidelity local machine usage (e.g., writing a document in Microsoft Word) for three overall reasons:

- Adversaries often depend on network traffic to gain an understanding of a network and its systems and software.
- Some cyber attacks, such as sniffing and pass-the-hash[13] attacks, require operational network traffic to be successful.
- False alarms given by non-malicious activity is a significant issue for network intrusion detection systems (NIDS) such as Snort [38]. To study the effectiveness of a NIDS, there is a need to utilize realistic network traffic and system interaction. The fact that the criticized [54] DARPA 1998/1999 intrusion detection datasets [51][52] are still used (see e.g. [9][87]) is an indicator that this is a significant issue for the cyber security domain as a whole.

5.2 Integrating the communication architecture

This section describes how communication architecture components can be implemented in an ICS testbed (Section 5.2.1) and how the fidelity of these components can be managed (Section 5.2.2).

5.2.1 Method of implementation

The identified testbeds use simulations or real hardware rather than virtualization or emulation opportunities to implement communication architecture

[13] To bypass a Windows authentication function using an observed password hash sent over the SMB protocol.

components. This is odd considering that virtualization of the network and communication hardware of a SCADA system can most likely be done using the standard components of any competent virtualization software. They provide virtual switches and hubs as-is. A router can be created by installing an existing Linux router distribution; it only needs to be properly configured. Modems can be either simulated using a telnet or VPN type of connection, or using a hardware-in-the-loop model. The latter alternative is feasible since there is often only one modem connected to a system, used as a backup if the ordinary internet connection is down.

Regarding simulation and emulation: the main difference between network simulation and emulation is that the first does not run in physical time, which the latter does. In network simulation situations the simulated time used is often slowed down relative to the physical time.

There are also specialized virtual network components that can be used. For example, there are more than 60 WiFi software routers for Windows and *nix available[14,15]. There are also at least 10 network simulators or emulators available[16].

5.2.2 Managing fidelity

To yield a high-fidelity testbed in respect to its communication architecture, it is necessary to be able to replicate the components and network topologies of ICS communication architectures in operation. Similarly to the control center (see Section 4.4.1), this can be (somewhat) accomplished through network sniffing (e.g., Wireshark) and scripts that consult firewall rulesets (e.g., Firewalk or the access policy tool (APT)). The accuracy of such activities has not been quantified through scientific research [33]. However, as the communication architecture on overall concerns “traditional” components, it is likely similar to the statistics presented in Section 4.4.1.

To be of high fidelity in respect to regular usage (without an adversary), the virtual network should handle events such as (random) hardware failures and degrading performance due to ageing cables and components, i.e. intermittent failures. By measuring the performance of a real network and then creating a virtual network with the same performance signature, a high-fidelity copy of the real network can been constructed.

Ricciulli [73] suggests that the overall throughput and its standard deviation is to be used to measure the fidelity of a simulated network, together with the slowdown needed to accommodate for the heavier load put on the system when

[14] http://listoffreeware.com/list-of-best-free-virtual-router-software/
[15] https://en.wikipedia.org/wiki/List_of_router_and_firewall_distributions
[16] http://www.brianlinkletter.com/open-source-network-simulators/

simulating. The slowdown is implemented using a synchronizing clock keeping track of the simulated time, which runs in a different pace than the physical time. The same problem and solution is discussed by Perumalla et al. [67].

Poylisher et al. [69] present a rather detailed description of how a virtual network can be built. They explain the difference between a simulated network and an emulated one, where the first can reach a higher fidelity. The main difference between the two types is that network emulators run in real time while simulated networks run in simulated time. The authors use SNMP to control their network simulator, which is implemented by splitting the network stack in half. The upper half (closer to the applications) is executed by the operating systems that host the different applications. The lower half (closer to the physical layer) is simulated. Poylisher et al. have evaluated their virtual network simulator by measuring the latency introduced in the system at different traffic loads. The highest latency value is 0.51 seconds at 120Mbps. They do not mention any other network fidelity metric in their paper.

Yoginath et al. [103] addresses the problem with network fidelity where simulators run on physical multi-core machines. To properly mimic a real network the timing of all the virtual machines and their communication must be taken into account. They write that such simulators must address the "*concept of a intra-node simulation timeline and also ensure the simulation time-order of VM execution within each multi-core host node*."

Covington and Hanson [16] used link throughput, application throughput (sent and received), application response time, VTC (Video Teleconferencing) end-to-end delay, and VoIP jitter to measure the performance of their simulated network.

Sultan et al. [92] present a solution called TimeSync to the timing and synchronization problem in virtualized networks. They have concentrated on the situation where the simulated time is running slower than the real time. To evaluate their solution they use the measured end-to-end packet latency through their emulated network. They have also evaluated their solution by looking at the latency distortion induced by the size of the simulated network.

Chertov et al. [11] have compared the use of emulated and simulated networks when performing DoS attack experiments. They conclude that there is a large difference between the two types of network virtualization methods. The metrics they use are:

1. average goodput[17] in Kbps (Kbits per second) or Mbps;
2. average congestion window size in packets, computed for testbed experiments by taking an average of the congestion window values;

[17] The number of useful information bits delivered by the network to a certain destination per unit of time

3. CPU percentage utilization; and
4. packets per second received and sent on the test network interfaces.

Pediaditakis et al. [65] have created SELENA, a Xen-based network emulation framework, which can be used for general testing of network-based systems and actions. They have identified fidelity, scalability and reproducibility as the three main properties that have to be considered when designing high fidelity network simulators and experiments [65]:

- The *fidelity* metric characterizes the precision and accuracy of the experiments ability to replicate a real system. Using network experimentation fidelity as an example the precision can be measured by the differences of the timing properties between the real and the experimental system, the degree of reuse of traffic models and applications from the real system and how well the experimental topology mimics the real system.
- The *scalability* of an experimentation platform is also important for its usefulness as a substitute for real world networks and systems. There are three functional aspects to consider with regard to the scalability of an experimental platform: execution time scalability, resource scalability and fidelity at scale. These three aspects exhibit Pareto efficiency, that is, if one is improved the others are negatively affected. Execution time scalability is the physical time needed to replicate an experiment. The shorter the time, the more experiments per hour can be run. The resource scalability metric measures the experimental platform's ability to be efficient and minimize hardware requirements. The last aspect, fidelity at scale, measures how the fidelity of the experimental platform varies when the size of the experiment increases.
- The *reproducibility* of a network experimentation platform is the third key property, according to [65]. The property describes the fidelity of experiments over heterogeneous hardware platforms, and the platform's ability to reproduce earlier experiments and their results. Regarding the use of heterogeneous hardware platforms the goal should be the same perceived processing capacity of the experimental system regardless of the actual hardware used. At least the impact of the host's actual processing power should be controllable.

The most important things to consider when planning experiments in experimental network environments are the following questions: "*which are the metrics that better characterize the system's resulting behavior*" and "*what is the desired degree of similarity with a reference system*" [65]. For the platform to be of high fidelity the statistical properties of the answer to the first question should closely mimic those of a real system. The similarity test used could for example be the Kolmogorov-Smirnov test that compares

an empirical cumulative distribution function with a reference cumulative distribution function [65].

5.3 Integrating field devices

This section describes how field devices can be implemented in an ICS testbed (Section 5.3.1) and how the fidelity of these components can be managed (Section 5.3.2).

5.3.1 Method of implementation

The use of virtualization requires that the architecture of the guest machine is the same as the architecture of the host machine, or in some cases a subset of it. Most importantly, the range of possible machine code sets and CPU registers is impacted by this demand. I/O devices may differ between the guest machine and the host machine since they are more or less emulated by virtualization software anyway. Because of these similarity requirements, virtualization of field devices on regular computers is only feasible for devices built with the Intel IA32 or IA64 architectures. This is for example the case for the ABB RTU560. In all other cases we must either use emulation or simulation.

As long as timing requirements are not critical, emulation is almost always at least a theoretical option. The main drawback of emulation is that it is slower than virtualization. Emulation, as well as virtualization, also requires a detailed knowledge about the architecture of the device to be emulated. Perhaps the largest practical barrier to overcome is that the architecture may need exhaustive reverse engineering before it is sufficiently well documented. Such reverse engineering may be extremely time-consuming, bordering impossible, to accomplish.

Another decision to be made is at which abstraction level to perform the emulation. For example, a Siemens SIMATIC S7-400 PLC implements a kind of virtual machine itself. In this case the virtualization is more similar to a Java virtual machine than to a hardware virtualization product like Oracle VirtualBox. Thus, the S7-400 has two different architecture levels. The hardware one is based upon Infineon TriCore CPUs, while the virtual architecture runs a Siemens proprietary machine code called MC7. Emulation can be performed either at the hardware level or at the MC7 level.

Emulation at the hardware level of a field device assumes that the architecture of the true hardware is documented in detail. In the case of the Siemens S7-400 it also requires the reverse engineering and emulation of field programmable gate arrays (FPGAs) present at the circuit boards. This presents a problem of its very

own nature. The general field of FPGA reversing still seems to be in its infancy[18]. The next problem after FPGA reversing is that the emulation of the FPGAs runs the risk of being so slow as to be completely useless. However, it is also conceivable that the FPGA parts of the device can be safely ignored in the emulation, as long as the emulated device have no connection to an actual physical process.

The benefit of emulation at the hardware level is that it would enable the utilization of the original device firmware. This in turn may lead to a heavily reduced emulator size, as well as magnitudes higher emulation accuracy. On the other hand there may be intellectual property rights complications when original firmware is to be run in an emulated environment. Hopefully such problems can be sorted out in cooperation with each device manufacturer.

In any case, emulation requires a detailed understanding of at least some level of architecture, which may not be feasible in practice. The fact that not a single one of the surveyed 30 ICS testbeds attempt to virtualize or emulate field devices (see Section 4.4.3), in addition to the lack of research on this topic (see Section 4.6), are proof of these problems.

If emulation and virtualization are not feasible, simulation is the only option left in terms of yielding testbed scale without involving actual hardware. However, since simulation only is accurate on the surface it cannot be used for all low level security testing. It may be useful for other purposes though, such as education and testing at a higher abstraction level than attacks on software (e.g., configurations and functions).

The level of ambition when doing simulation can lead to very different amounts of complexity. For example, it may be desirable to fully implement real world communications protocols. This would enable communication between the simulator and completely external systems. In a fully simulated environment there may be no such needs, and then even the communication between different parts may be simulated. Another possibility is to implement known vulnerabilities into the network stack of the simulator, so various security testing tools and exploits can be run against it. The network stack can also be made sufficiently similar to the real device so that it looks the same when OS fingerprinting is performed against it.

5.3.2 Managing fidelity

Field devices are the most difficult type of ICS component to replicate due to their special software, hardware and logic. Running a local software agent to gather a field device's configuration is generally not feasible and replicating their

[18] http://www.hackitoergosum.org/2010/HES2010-sbourdeauducq-FPGA-Challenge.pdf

disks and memory is not only very difficult, but also generally impractical as their special hardware still is required to execute the copied configuration. While they have network interfaces that can be probed, doing so can cause them to crash [53]. Network sniffers can be used, but have little support for ICS-specific protocols [33]. Thus, there is a need to develop novel mechanisms for fingerprinting and recreating field devices. This is further discussed in Section 6.4.

We did not identify any metrics specifically developed for the fidelity of virtualized field devices. The explanation could be that there are no fully virtualized field device products on the market, or in academia. Exactly what to measure depends on the intended use of the field device. For example, the demands on a PLC used for the discovery of novel vulnerabilities are different from the demands on a PLC used to test the effects of a specific vulnerability on a whole plant. Yet another example is the testing of the effects of malware directed at devices, which places special demands on its own.

At the highest fidelity level, when the virtualized field device is used for discovery of novel attacks it should be possible to connect the device to a real system and it should work. Another metric can be formulated in the following way: the virtual device should not be possible for a skilled attacker to differentiate from a real one. At the other end of the fidelity scale it might be enough to get the correct response to standard interactions from a system or operator.

The main properties of field device fidelity metrics are functionality and correct timing (which is a key focus area for works that attempt to virtualize other kinds of embedded systems, see Section 4.6). Hence, a proper handling of latency is vital for any high-fidelity virtual application. The latency and its statistical properties should be controlled, for example its mean, median, variance and distribution. Also the changes over time of these metrics should be taken care of, for example the device's behavior under different load conditions.

Regarding the behavioral aspect, the fidelity of the virtual field device is related to its ability to mimic the physical device in all aspects, for normal input, as well as when its stimulus is faulty or missing. A list of tentative example metric components for a PLC follows:

- Are the formats of code, data and other blocks exactly similar or just functionally similar to the ones in a real PLC? The difference may be important for some vulnerability testing and complete accuracy may be crucial for malware testing.
- Can it handle PLC machine code fully, so any code developed for a real PLC can run on it? This is not the case for some PLC emulators that currently are available on the market.

- Can it handle undocumented aspects of PLC machine code? This may be important when testing malware, since malware may insert undocumented instructions to throw emulators, disassemblers or debuggers off.
- Does it execute faulty machine code in a similar way to a real PLC? This may also be important when testing malware, which may use such constructs to throw emulators, disassemblers or debuggers off. A real PLC might try to fix the problem and run the faulty machine code in the way it assumes the code was expected to run. Or simply do something other than crash - a behavior which must be replicated exactly in some cases.
- The four above aspects can be extended to include communication protocols. For example, undocumented and faulty protocol options.
- Are there any undocumented system areas with information that low level code can access, and are they similar to the ones in a real PLC? This may be very important in malware testing.
- Can the original PLC firmware be used? This most likely leads to significantly higher fidelity than other kinds of implementation.
- Does the PLC respond as its real counterpart when subjected to overload attacks? For example in regard to CPU and memory capacity.

5.4 Integrating the process

This section describes how physical processes be implemented in an ICS testbed (Section 5.4.1) and how the fidelity of such processes can be managed (Section 5.4.2).

5.4.1 Method of implementation

Leaving aside the fact that a testbed can be connected to a real physical process, there is a need to incorporate simulators. The systematic review identified a variety of simulators that can be used for this purpose, such as Matlab/Simulink models, OPAL-RT's Power Hardware-in-the-Loop, PowerWorld, and custom written applications.

The interviews with ABB Ventyx showed that they had several process simulators; for instance, a power system operator training simulator[19]. The respondents perceived it as non-trivial to alter these simulators as this would require not only altering the simulator itself, but all related testbed components as well (the configuration of field devices, the communication architecture and

[19] http://www.abb.com/industries/ap/db0003db004333/c125739a0067cb49c1257026003d4a31.aspx

the control center). Other ICS manufacturers are bound to have similar process simulators.

5.4.2 Managing fidelity

The difficulty involved with replicating the aspects of a physical process depends on the process in question. For example, a water storage tank is simple to simulate with high fidelity [60], whereas a power grid requires knowledge regarding a plethora of parameters to be simulated with high fidelity [68]. For more complex processes such as power grids, it is desirable to automatically assess configurations as high fidelity otherwise is difficult to achieve. It is not yet clear how this information can be automatically obtained from a general ICS in operation. However, it is clear that some ICS store relevant process information in control center components. For instance, the components of a power system at an electrical level and the relationships between each component are sometimes stored as an XML according to the Common Information Model (CIM, see the IEC 61970-301) [56] in control center components. This information can be extracted to enable modeling an electrical power grid in a simulator.

Similarly to replicating a physical process, how to study its fidelity also depends on the process that is concerned – a high voltage grid has different fidality requirements than a water distribution network. For example, how fidelity is managed in the power system domain is described by Pourbeik [68] and in a white paper by the North American Electric Reliability Corporation (NERC) [64]. The white paper by NERC provides a method and metrics for validating power system models and emphasizes that the system powerflow model should match the real world system. Pourbeik provide a survey of different means of measuring the fidelity of different power system models, such as transmission line models and power generator models. Fidelity is studied by first subjecting a simulation and a real world system to a series of stimuli that are thought to be representative of the use of the real world system (e.g., opening a breaker or injecting a power fault). The resulting data (e.g., power output response, voltage, field current or Watt) is then compared for the simulation and the real world system through graphical plots and statistical distribution fitting metrics.

An additional fidelity problem not considered by [64] or [68] concerns the fact that running process simulations in real-time is generally extremely CPU-intensive [83]. To manage this problem, experiments using process simulators (e.g., Matlab/Simulink models) often slow down the execution time of the simulation, thus decreasing the CPU load without reducing the validity of the experiment. This is unfortunately difficult to accomplish for ICS security testbeds as they often involve humans (such as adversaries) and real hardware devices, which both require the testbed to perform in real-time. Siaterlis and Genge [83] study this problem for the testbed at the European Commission Joint Research Centre in Italy by comparing the execution time of their Simulink

process models (in milliseconds) to the requirements of different physical processes such as power plants, railway systems and IEEE bus grid (see Section 4.5).

6 Conclusions and future work

The present study was conducted in cooperation with project AVA that is managed by INL and DHS. Overall, the findings and suggestions for future work match to those described in the AVA pre-study [42], and continued collaboration with INL and DHS is deemed as critical for the success of VICS. This is further discussed in Section 6.1. Apart from collaboration with AVA, other actors have been involved in VICS. For example, there have been meetings with ABB Ventyx in Västerås, a visit from David Bakken from the Washington State University, a presentation of DETER by Terry Benzel from the University of Southern California, and a master thesis is currently being conducted in cooperation with the Royal Institute of Technology (KTH) in Stockholm (see Section 6.5).

This study first examined what ICS testbeds currently exist (RQ1), what ICS objectives these propose (RQ2), how ICS components are implemented within them (RQ3), how they manage testbed requirements (RQ4), and what methods are available for the virtualization of ICS field devices (RQ5).

A total of 30 ICS testbeds were identified. The most common overall objectives of these testbeds are to facilitate vulnerability analysis, education and tests of defense mechanisms. ICS components are typically simulated, even in cases where virtualization is judged as feasible. The fidelity of these testbeds is seldom discussed (63%), and when it is discussed, there are only two articles (for two testbeds) that quantify fidelity. No existing methods for virtualizing operational field devices were identified.

This study then suggested means of creating an ICS testbed as well as means to examine the fidelity of such a testbed. Based on NIST 800-82 [90], an ICS testbed should consider four general areas: the control center, the communication architecture, field devices and the physical process itself (see Section 2). Based on the results from the literature review and experiences within the research group, the overall objectives for an ICS (security) testbed are facilitation of vulnerability analyses, education and tests of defense mechanisms (see Section 4.3). The discussion presented in this section builds on these areas and objectives.

Implementation opportunities based on four methodologies were considered: virtualization, emulation, simulation or hardware (see Section 3). An overview of suggested implementation methods and possible technical fidelity issues with these methods is described in Table 7.

The results show that *control center* and *communication architecture* components are possible to virtualize without too many technical issues. It is however not a straightforward task to simulate application interaction, such as interaction that leads to network traffic between different machines (see sections

5.1-5.2). The *physical process* is deemed best implemented as a simulation model, e.g. using Matlab/Simulink (see Section 5.4). While physical process simulators are key elements in most existing ICS testbeds (see Section 4.4.4), it is unknown to the authors of this report how much value they actually provide – discovery and exploitation of software and hardware vulnerabilities is not contingent on the availability of a physical process simulator. Future work should examine this property in-depth.

Table 7. Suggested implementation methods and perceived technical fidelity issues.

Area	**Implementation**	**Technical fidelity issues**
Control center	Virtualization	Application interaction
Communication architecture	Virtualization	Application interaction
Field devices	Virtualization, emulation, simulation or hardware	Development of simulator
Physical process	Simulation	Development of simulator

Implementation of *field devices* (e.g., a PLC or an RTU) depends on the kind of device that is considered. Modern field devices are often based on architectures and firmware that have current virtualization and/or emulation support[20]. The same applies for field devices that manufacturers have created emulation software for (it is however not certain that manufacturers would want to share such technology). Older or proprietary field devices (such as the Siemens S7 series) are however not supported by any current virtualization or emulation approaches. As a field device can be used for up to 40 years [93], there is bound to be a plethora of such devices in operation. For this reason, the AVA study [42] proposes using the emulator QEMU in combination with the compiler LLVM to emulate field devices. The study [42] recognizes that more research is required to validate the applicability of this approach. For this purpose, the present study conducted technical assessments of two models in the Siemens S7-400 and Siemens S7-1200 series (see Appendix C). The results from this analysis show that the S7-400 and S7-1200 build on proprietary and completely different machine code and that both are judged extremely difficult to emulate with a high degree of accuracy. In other words, the QEMU/LLVM approach would be very expensive to implement for the Siemens S7 PLC series. As this cost is directly influenced by the diversity of operational field devices, we conducted a survey of

[20] For example, the Schneider Electric Modicon Quantum PLC uses an x86 processor, whereas its Ethernet module uses vxWorks 5.4 and a PowerPC processor (MPC870): http://www.digitalbond.com/tools/basecamp/schneider-modicon-quantum/.

PLC manufacturers (see Appendix A). A total of 341 manufacturers with an unknown number of product families and models were identified. It is very likely that many of the field devices that the 341 identified manufacturers have developed are as difficult to emulate as the Siemens S7 series. Consequently, the present study argues that PLCs which are unsupported by current virtualization and emulation technologies should be simulated or implemented as hardware. Of these two approaches, simulators are judged as sufficient for all testbed purposes except software and hardware vulnerability discovery (see Section 6.5).

Based on the results gathered from the present study, six suggestions for future work are identified for the next phase of VICS (to be conducted during 2015 and 2016). These suggestions are described in Section 6.1 – 6.6. Finally, Section 6.7 reflects on the overall project goal – creating an ICS testbed – and discusses limitations of this study with respect to it.

6.1 Cooperation with INL and DHS

There is much to be gained if DHS, MSB, INL and FOI cooperate in the design and construction of an ICS testbed. Pooling resources such as personnel, data collection scripts, manufacturer and operator contacts enable better results at a lower cost. Furthermore, the testbeds that AVA and VICS plan to base the ICS testbeds on (ACORN and CRATE) both build on standard virtualization technologies. This enables porting entire machines and perhaps even complete testbed configurations.

The present report illustrates the value of this collaboration: it builds on and complements the pre-study of AVA [42]. It would not have been possible to achieve the results described in the present report without the collaboration with AVA. Overall, the results from the present study support the current and planned activities within AVA.

Future work should continue the collaboration that has been utilized so far between DHS, MSB, INL and FOI, as well as examine how to further improve it.

6.2 Involve ICS manufacturers and operators

Implementing ICS components that build on software and hardware which can be virtualized, such as ABBs Network Manager, are judged to require little effort and provide high fidelity, while development of novel (simulated) components of the same sort is believed to be expensive and provide uncertain (likely low) fidelity. For components that are difficult to virtualize, ICS manufacturers sometimes have emulators and simulators. Furthermore, to analyze (or replicate) their systems in operation, manufacturers have various tools and methods that facilitate high-fidelity data collection at a low cost without disrupting the studied

system. This, in addition to all the experience that manufacturers possess regarding ICS in general and ICS testbeds in particular, suggest that involving them is imperative for the success of VICS.

ICS manufacturers would obtain several advantages from participating in VICS, in particular:

- Vulnerability analysis of their components and configurations under controlled conditions. Discovery of novel vulnerabilities is coordinated with the manufacturer.
- The many individuals who participate in the education and awareness activities that are organized by FOI get to interact with the components and configurations provided by the manufacturer.
- FOI develops and tests novel defense mechanisms such as network intrusion detection systems. Participating manufacturers directly benefit from these mechanisms as they per default are customized for their solutions.

Similarly to ICS manufacturers, ICS operators have valuable deep knowledge on ICS components and configurations. The key difference is that ICS operators have specialized knowledge about their specific installations, whereas ICS manufacturers have more general knowledge of their developed systems. A high-fidelity testbed thus require involvement also of ICS operators. Operators would benefit from the participation in several ways, in particular:

- Vulnerability analysis of systems in operation under controlled conditions, as well as suggestions for how to mitigate discovered flaws.
- Participating operators directly benefit from defense mechanisms developed and tested by FOI as they per default are customized for their systems.

The Swedish national center for security in industrial control systems and critical infrastructures (NCS3) is managed by FOI and funded by MSB. NCS3 has connections with various operators and manufacturers. Thus, VICS would benefit from coordinating these activities with NCS3.

6.3 Identify tangible testbed objectives

The three identified testbed objectives (vulnerability analysis, education and tests of defense mechanisms) are described on a very superficial level for all existing testbeds (see Section 4.3). To be able to relate these objectives to actual testbed design decisions, there is a need to break them down and make them more tangible.

One means to make them more tangible is to employ taxonomies, e.g., the taxonomy for ICS vulnerability assessment which is presented by NIST 800-82 [90]:

- **Policy and Procedure Vulnerabilities** are vulnerabilities due to incomplete, inappropriate or nonexistent security documentation, including policy and implementation guides (procedures).
- **Platform Vulnerabilities** are vulnerabilities due to flaws, misconfigurations, or poor maintenance of their platforms, including hardware, operating systems, and ICS applications.
- **Network Vulnerabilities** are vulnerabilities due to flaws, misconfigurations, or poor administration of ICS networks and their connections with other networks.

These three topics contain a total of 71 more concrete types of vulnerability assessments that can be used to create better requirements for ICS testbeds. For instance, if one wishes to analyze the presence of the platform vulnerability buffer overflow, there is a need for real software to be in place. This would preferably involve hardware, and at worst virtualization or emulation: simulation simply would not be sufficient (as the software code-base would differ).

The above reasoning applies also to the other objectives such as education and tests of defense mechanisms. For example, an ICS security novice would not notice if an ICS environment is incomplete or if a PLC communicates a bit differently than what it typically does in the real system, whereas an ICS expert would call for a higher fidelity regarding both of these aspects to perceive a testbed as realistic.

Another means to make the objectives more tangible is to relate them to the activities at FOI. Within vulnerability analysis, FOI conducts both system-level (e.g. [61]) and component-level (e.g. [99]) vulnerability discovery. For the prior, simulated field devices are likely sufficient as the primary attack vector generally concerns the control center (which is judged possible to virtualize). For the latter, real software (and preferably hardware) is required. Within education, FOI provides courses and cyber defense exercises (CDX) that are aimed both at beginners, intermediate and advanced ICS and IT users. None of these educations are judged to require real field devices as they do not focus on vulnerability discovery on that level of abstraction. When it comes to tests of defense mechanisms, FOI has previously used CRATE to test a variety of cyber security mechanisms, such as NIDS [85], automated network scanners [41], and the accuracy of system-level vulnerability metrics [40]. Whether or not simulated field devices are adequate depend on what defense mechanism is concerned.

A third means to make the objectives more tangible is to survey the opinion of ICS manufacturers and operators. This is a necessity for several reasons, in particular: (1) these actors have significant practical experience on the matter and

(2) it facilitates increased involvement with them, which is imperative for the success of VICS (see Section 6.2).

6.4 Develop tools for fingerprinting and recreating ICS configurations

There is a need to develop a methodology for recreating ICS in operation. One important aspect within this methodology is to collect data on ICS configurations in a resource-effective, accurate and non-disruptive manner.

One important aspect within data collection concerns automatization using methods such as network scanning (e.g., Nmap or Nessus) and sniffing (e.g., Wireshark or NetworkMiner). Automated tools enable recreating operational ICS configurations for a lower cost and with higher fidelity than what is possible to achieve from interview-based methods. These methods might have sufficient accuracy for general IT applications [39][41], but lack the ability to accurately identify ICS specific components such as field devices [33]. Consequently, there is a need to develop ICS-specific tools that can be used to capture data in a means that a testbed can parse configurations from. A potential bonus result from this work could be novel intrusion detection systems, algorithms and rulesets, such as described by Hadžiosmanović et al. [32] and Fovino et al. [24].

Apart from data collection, there is a need to generate valid application interaction within the ICS testbed (see Section 5.1). CRATE, the testbed that VICS is planned to be based on (see Appendix E), currently has user agents (bots) that are able to access shared folders, read email, execute files and browse the web. These bots should be extended with the capability to interact in a means that is representative for ICS.

When data on application configurations and interactions have been collected, there is a need to project this as ICS testbed configurations. To fulfill this activity, there is a need to extend CRATE with such functionality.

Finally, there is a need to develop methodology and tools for ensuring that a testbed configuration is valid. Some preliminary metrics and thoughts on this subject are presented in sections 5.1-5.4.

6.5 Develop simulated field devices

Our results indicate that virtualizing or emulating field devices rarely is a feasible solution. An alternative means to reach testbed scale is to implement simulation models of field devices. Simulation is adequate when it doesn't compromise the fidelity of a testbed with respect to the objectives of an experiment. In practice,

we believe that simulation is acceptable for most objectives except software and hardware vulnerability discovery.

An acceptable simulated field device should be able to function as a real field device both under normal (non-malicious) usage and under the presence of an adversary (see Section 5):

1. **Replacing a real field device with the simulated field device should not impact the ICS**. The simulated device should not impact the state of the process (e.g., be able to react to a power fault in the same way as a real PLC) or the state of the control center (e.g., act on commands from SCADA [for instance, open a breaker] and send measurements to SCADA [such as voltage]). What applications, protocols, and logic (according to IEC 61131-3) that should be supported depend on the desired ICS configuration. One approach could be to incorporate existing simulators such as the Modbus Rsim or the Siemens SIMATIC PLCSIM simulator. Ideas for other approaches are given by the testbeds that have chosen to simulate field devices (see Section 4.4.3, e.g., [2], [70], [35] and [49]) as well as Chunjie and Hui [13] and Zhang et al. [106], who propose means of implementing IEC 61131-3 in virtual machine containers. Discussions with operators and manufacturers would also be beneficial to conduct.
2. **An adversary scanning a simulated field device should not be able to tell the difference between it and its portrayed real field device**. Adversaries use both active (e.g., Nmap or Nessus) and passive (e.g., Wireshark) data collection methods to scan IT components such as field devices. *To be of high fidelity in regard to active scanning*, the field device should respond in the same way as its portrayed real device when subjected to network traffic. One means of accomplishing this task is to use Honeyd[21], a software that can be used to simulate everything from an entire operating system to the network stack. For example, Honeyd can be used to simulate network service responses of a Siemens S7-1200. FOI has previously both been a user and a developer of Honeyd. *To be of high fidelity in regard to passive scanning*, a field device should employ the same network stack as its portrayed real device. This is the case as adversaries profile applications based on protocol implementation (i.e., actual network traffic) rather than protocol specification. Thus a field device should not only fulfill the first requirement described above, but its protocol should be (at best) *identical* to the desired implementation. Fidelity in respect to passive scanning is thus more difficult to achieve than for active scanning, but arguably also less important as active scanning is more accurate and thus more frequently used by adversaries.

[21] http://www.honeyd.org/

3. **An adversary exploiting a simulated field device should not be able to tell the difference between it and its portrayed real field device**. As of March 2015, the Open Source Vulnerability Database[22] contains 1030 vulnerabilities that contain the word "SCADA" in their descriptions. Of the vulnerabilities that concern field devices, some enable attackers to disrupt a device (such as bricking it or blocking network communications); others' enable attackers to install new instructions on it. For instance, CVE 2014-5074 enables an attacker to remotely shut down a Siemens S7-1500 PLC, and the malware Stuxnet infects memory block DB890 of the Siemens S7-300 PLC[23] (to periodically adjust motor rotation speed). A simulated Siemens S7-1500 or S7-300 in a Linux kernel would per definition not allow these attacks to succeed as its codebase would differ from the real devices. For this reason, it can be very difficult to enable many exploits to work on a simulation in a way that an attacker expects. One means of accomplishing this task could be to create a ruleset corresponding to known exploits and implement this ruleset in the simulated field device. Incoming packets are matched against the ruleset and if an exploit is deemed successful, an appropriate outcome is triggered. For instance, an exploit corresponding to CVE 2014-5074 would serve to shut down the simulator. An example ruleset and pattern matching system that could be used for this purpose is the NIDS Snort.

VICS is currently co-supervising a master thesis with Professor Lars Nordström from the department of Industrial Information and Control Systems at KTH that concerns creating simulated field devices. This master thesis is planned to be completed by July 2015.

6.6 Automated vulnerability discovery

The AVA study [42] suggests that automated discovery of vulnerabilities within ICS configurations as a key ICS testbed component. The present study left out this aspect due to the time and resource constraints that are involved. We agree with [42] that it is a key activity and thus aim to involve it in future work. This component should involve both identifying publicly known vulnerabilities and novel vulnerabilities ("zero days"), as well as suggesting mitigations for such flaws.

[22] http://osvdb.org
[23] http://www.symantec.com/connect/blogs/exploring-stuxnet-s-plc-infection-process

6.7 Develop a functional testbed

The final envisioned result of VICS is a framework including methods and tools for replicating ICS in operation within an isolated virtual environment building on CRATE. Extensive work is required to accomplish this task and the activities described in sections 6.1-6.6 denote the first steps to realize it. A comprehensive implementation plan relating these activities to the larger goal should be created. This plan should be iteratively updated with technical specifications and design documents.

Furthermore, this report has focused on the various technical issues related to implementing an ICS testbed. However, there are many other issues that were not considered, in particular:

- It is uncertain how an ICS testbed best should be administrated and managed when it is operational (and in the long run).
- If no manufacturers or operators want to participate with their components and configurations, there is a need to implement SCADA simulators. It is uncertain how such an activity best should be addressed[24].
- When extracting ICS data, there is a need to anonymize gathered data without compromising its validity. There are to the authors' knowledge no standard methods that can be used for this purpose.

Issues such as these need be addressed by future work.

[24] That said, the value of a testbed without any interest from manufacturers or operators is questionable.

7 References

[1] Cryptographic protection of scada communications - retrofitting serial communications. Technical Report 12, American Gas Association (AGA), 2006.

[2] Abdulmohsen Almalawi, Zahir Tari, Ibrahim Khalil, and Adil Fahad. Scadavt-a framework for scada security testbed based on virtualization technology. In *Local Computer Networks (LCN), 2013 IEEE 38th Conference on*, pages 639–646. IEEE, 2013.

[3] Mikael Asberg, Nils Forsberg, Thomas Nolte, and Shinpei Kato. Towards real-time scheduling of virtual machines without kernel modifications. In *Emerging Technologies & Factory Automation (ETFA), 2011 IEEE 16th Conference on*, pages 1–4. IEEE, 2011.

[4] RF Beach, GL Kimnach, TA Jett, and LM Trash. Evaluation of power control concepts using the pmad systems test bed. In *Energy Conversion Engineering Conference, 1989. IECEC-89., Proceedings of the 24th Intersociety*, pages 327–332. IEEE, 1989.

[5] Terry Benzel. The science of cyber security experimentation: the deter project. In *Proceedings of the 27th Annual Computer Security Applications Conference*, pages 137–148. ACM, 2011.

[6] David C Bergman. Power grid simulation, evaluation, and test framework. 2010.

[7] David C Bergman, Dong (Kevin) Jin, David M Nicol, and Tim Yardley. The virtual power system testbed and inter-testbed integration. In *CSET*, 2009.

[8] Vincent H Berk, Ian Gregorio-de Souza, and John P Murphy. Generating realistic environments for cyber operations development, testing, and training. In *SPIE Defense, Security, and Sensing*, pages 835908–835908. International Society for Optics and Photonics, 2012.

[9] Jakub Breier and Jana Branišová. Anomaly detection from log files using data mining techniques. In *Information Science and Applications*, pages 449–457. Springer, 2015.

[10] AG Bruce. Reliability analysis of electric utility scada systems. In *Power Industry Computer Applications., 1997. 20th International Conference on*, pages 200–205. IEEE, 1997.

[11] 3) Chertov, R. (1, 4) Fahmy, S. (1, and 5) Shroff, N.B. (2. Fidelity of network simulation and emulation: A case study of tcp-targeted denial of service attacks. *ACM Transactions on Modeling and Computer Simulation*, 19(1), 2008.

[12] Henrik Christiansson and Eric Luiijf. Creating a european scada security testbed. In *Critical Infrastructure Protection*, pages 237–247. Springer, 2008.

[13] Zhou Chunjie and Chen Hui. Development of a plc virtual machine orienting iec 61131-3 standard. In *Measuring Technology and Mechatronics Automation, 2009. ICMTMA'09. International Conference on*, volume 3, pages 374–379. IEEE, 2009.

[14] Wang Chunlei, Fang Lan, and Dai Yiqi. A simulation environment for scada security analysis and assessment. In *Measuring Technology and Mechatronics Automation (ICMTMA), 2010 International Conference on*, volume 1, pages 342–347. IEEE, 2010.

[15] Jacob Cohen. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological bulletin*, 70(4):213, 1968.

[16] D.A. Covington and M.D. Hanson. High-fidelity modeling and scalable simulation for tactical network design. *MILCOM 2008 - 2008 IEEE Military Communications Conference*, page 1, 2008.

[17] Robert J. Creasy. The origin of the vm/370 time-sharing system. *IBM Journal of Research and Development*, 25(5):483–490, 1981.

[18] Axel Daneels and Wayne Salter. What is scada. In *International Conference on Accelerator and Large Experimental Physics Control Systems*, pages 339–343, 1999.

[19] Khalid W Darwish, AR Al Ali, and Rached Dhaouadi. Virtual scada simulation system for power substation. In *Innovations in Information Technology, 2007. IIT'07. 4th International Conference on*, pages 322–326. IEEE, 2007.

[20] CM Davis, JE Tate, H Okhravi, C Grier, TJ Overbye, and D Nicol. Scada cyber security testbed development. In *Proceedings of the 38th North American power symposium (NAPS 2006)*, pages 483–488, 2006.

[21] G Dondossola, F Garrone, and J Szanto. Cyber risk assessment of power control systemsâ€"a metrics weighed by attack experiments. In *Power and Energy Society General Meeting, 2011 IEEE*, pages 1–9. IEEE, 2011.

[22] Thomas Edgar, David Manz, and Thomas Carroll. Towards an experimental testbed facility for cyber-physical security research. In *Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research*, page 53. ACM, 2011.

[23] I Nai Fovino, Marcelo Masera, Luca Guidi, and Giorgio Carpi. An experimental platform for assessing scada vulnerabilities and countermeasures in power plants. In *Human System Interactions (HSI), 2010 3rd Conference on*, pages 679–686. IEEE, 2010.

[24] Igor Nai Fovino, Andrea Carcano, T De Lacheze Murel, Alberto Trombetta, and Marcelo Masera. Modbus/dnp3 state-based intrusion detection

system. In *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, pages 729–736. IEEE, 2010.

[25] Haihui Gao, Yong Peng, Kebin Jia, Zhonghua Dai, and Ting Wang. The design of ics testbed based on emulation, physical, and simulation (eps-ics testbed). In *Intelligent Information Hiding and Multimedia Signal Processing, 2013 Ninth International Conference on*, pages 420–423. IEEE, 2013.

[26] Thomas Gaska, Brian Werner, and David Flagg. Applying virtualization to avionics systems-the integration challenges. In *Digital Avionics Systems Conference (DASC), 2010 IEEE/AIAA 29th*, page 5, 2010.

[27] Annarita Giani, Gabor Karsai, Tanya Roosta, Aakash Shah, Bruno Sinopoli, and Jon Wiley. A testbed for secure and robust scada systems. *ACM SIGBED Review*, 5(2):4, 2008.

[28] Robert P Goldberg. Architecture of virtual machines. In *Proceedings of the workshop on virtual computer systems*, pages 74–112. ACM, 1973.

[29] Zonghua Gu and Qingling Zhao. A state-of-the-art survey on real-time issues in embedded systems virtualization. 2012.

[30] Michele Guglielmi, Igor Nai, Andres Perez-Garcia, and Christos Siaterlis. A preliminary study of a wireless process control network using emulation testbeds. In *Mobile Lightweight Wireless Systems*, pages 268–279. Springer, 2010.

[31] Feng Guo, Luis Herrera, Mohammed Alsolami, He Li, Pu Xu, Xintong Lu, Andong Lang, Jin Wang, and Zhijun Long. Design and development of a reconfigurable hybrid microgrid testbed. In *Energy Conversion Congress and Exposition (ECCE), 2013 IEEE*, pages 1350–1356. IEEE, 2013.

[32] Dina Hadžiosmanovic, Lorenzo Simionato, Damiano Bolzoni, Emmanuele Zambon, and Sandro Etalle. N-gram against the machine: On the feasibility of the n-gram network analysis for binary protocols. Ín *Research in Attacks, Intrusions, and Defenses*, pages 354–373. Springer, 2012.

[33] Adam Hahn and Manimaran Govindarasu. An evaluation of cybersecurity assessment tools on a scada environment. In *Power and Energy Society General Meeting, 2011 IEEE*, pages 1–6. IEEE, 2011.

[34] Adam Hahn, Ben Kregel, Manimaran Govindarasu, Justin Fitzpatrick, Rafi Adnan, Siddharth Sridhar, and Michael Higdon. Development of the powercyber scada security testbed. In *Proceedings of the sixth annual workshop on cyber security and information intelligence research*, page 21. ACM, 2010.

[35] Michael Haney and Mauricio Papa. A framework for the design and deployment of a scada honeynet. In *Proceedings of the 9th Annual Cyber and Information Security Research Conference*, pages 121–124. ACM, 2014.

[36] Gernot Heiser. Virtualizing embedded systems: why bother? In *Proceedings of the 48th Design Automation Conference*, pages 901–905. ACM, 2011.

[37] Jeffrey Hieb, James Graham, and Sandip Patel. Security enhancements for distributed control systems. In *Critical Infrastructure Protection*, pages 133–146. Springer, 2008.

[38] Hannes Holm. Signature based intrusion detection for zero-day attacks:(not) a closed chapter? In *System Sciences (HICSS), 2014 47th Hawaii International Conference on*, pages 4895–4904. IEEE, 2014.

[39] Hannes Holm, Markus Buschle, Robert Lagerström, and Mathias Ekstedt. Automatic data collection for enterprise architecture models. *Software & Systems Modeling*, 13(2):825–841, 2014.

[40] Hannes Holm, Mathias Ekstedt, and Dennis Andersson. Empirical analysis of system-level vulnerability metrics through actual attacks. *Dependable and Secure Computing, IEEE Transactions on*, 9(6):825–837, 2012.

[41] Hannes Holm, Teodor Sommestad, Jonas Almroth, and Mats Persson. A quantitative evaluation of vulnerability scanning. *Information Management & Computer Security*, 19(4):231–247, 2011.

[42] Idaho National Laboratory (INL). Control system automated vulnerability assessment study. Technical report, Idaho National Laboratory (INL), 2013.

[43] Erland Jonsson and Tomas Olovsson. A quantitative model of the security intrusion process based on attacker behavior. *Software Engineering, IEEE Transactions on*, 23(4):235–245, 1997.

[44] B Jurisic, N Holjevac, and B Morvaj. Framework for designing a smart grid testbed. In *Information & Communication Technology Electronics & Microelectronics (MIPRO), 2013 36th International Convention on*, pages 1247–1252. IEEE, 2013.

[45] Rao Kalapatapu. Scada protocols and communication trends. *ISA EXPO*, 2004.

[46] Dong-joo Kang and Rosslin John Robles. Compartmentalization of protocols in scada communication. *International Journal of Advanced Science and Technology*, 8, 2009.

[47] Barbara Kitchenham. Procedures for performing systematic reviews. *Keele, UK, Keele University*, 33:2004, 2004.

[48] Pär Klingstam and Per Gullander. Overview of simulation tools for computer-aided production engineering. *Computers in Industry*, 38(2):173–186, 1999.

[49] Nishchal Kush, Andrew J Clark, and Ernest Foo. Smart grid test bed design and implementation. 2010.

[50] Edward A Lee. Cyber physical systems: Design challenges. In *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, pages 363–369. IEEE, 2008.

[51] Richard Lippmann, Joshua W Haines, David J Fried, Jonathan Korba, and Kumar Das. The 1999 darpa off-line intrusion detection evaluation. *Computer networks*, 34(4):579–595, 2000.

[52] Richard P Lippmann, David J Fried, Isaac Graf, Joshua W Haines, Kristopher R Kendall, David McClung, Dan Weber, Seth E Webster, Dan Wyschogrod, Robert K Cunningham, et al. Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, volume 2, pages 12–26. IEEE, 2000.

[53] Stefan Lüders. Cern tests reveal security flaws with industrial network devices. *The Industrial Ethernet Book*, 35(CERN-OPEN-2006-074):12–23, 2006.

[54] Matthew V Mahoney and Philip K Chan. An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection. In *Recent Advances in Intrusion Detection*, pages 220–237. Springer, 2003.

[55] Malaz Mallouhi, Youssif Al-Nashif, Don Cox, Tejaswini Chadaga, and Salim Hariri. A testbed for analyzing security of scada control systems (tasscs). In *Innovative Smart Grid Technologies (ISGT), 2011 IEEE PES*, pages 1–7. IEEE, 2011.

[56] Alan W McMorran. An introduction to iec 61970-301 & 61968-11: The common information model. *University of Strathclyde*, 93:124, 2007.

[57] Miles A McQueen, Wayne F Boyer, Mark A Flynn, and George A Beitel. Time-to-compromise model for cyber risk reduction estimation. In *Quality of Protection*, pages 49–64. Springer, 2006.

[58] DH Moore, JM Murray, FP Maturana, T Wendel, and KA Loparo. Agent-based control of a dc microgrid. In *Energytech, 2013 IEEE*, pages 1–6. IEEE, 2013.

[59] Thomas Morris, Anurag Srivastava, Bradley Reaves, Wei Gao, Kalyan Pavurapu, and Ram Reddi. A control system testbed to validate critical infrastructure protection concepts. *International Journal of Critical Infrastructure Protection*, 4(2):88–103, 2011.

[60] Thomas Morris, Rayford Vaughn, and Yoginder S Dandass. A testbed for scada control system cybersecurity research and pedagogy. In *Proceedings of the*

Seventh Annual Workshop on Cyber Security and Information Intelligence Research, page 27. ACM, 2011.

[61] Karin Mossberg Sonnek, Hannes Holm, Johan Lindgren, Fredrik Lindgren, and Erik Westring. Foi-r–4029–se, ncs3 - informations- och styrsystem inom spårbunden trafik, en kartläggning. Technical report, Swedish Defence Research Agency (FOI), 2014.

[62] Tzi-cker Nanda and Susanta Chiueh. A survey on virtualization technologies. *RPE Report*, pages 1–42, 2005.

[63] Sani R Nassif and Joseph N Kozhaya. Fast power grid simulation. In *Proceedings of the 37th Annual Design Automation Conference*, pages 156–161. ACM, 2000.

[64] North American Electric Reliability Corporation (NERC. Power system model validation - a white paper by the nerc model validation task force of the transmission issues subcommittee. Technical report, North American Electric Reliability Corporation (NERC, 2010.

[65] Dimosthenis Pediaditakis, Charalampos Rotsos, and Andrew William Moore. Faithful reproduction of network experiments. In *Proceedings of the Tenth ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ANCS '14, pages 41–52, New York, NY, USA, 2014. ACM.

[66] C Dennis Pegden, Randall P Sadowski, and Robert E Shannon. *Introduction to simulation using SIMAN*. McGraw-Hill, Inc., 1995.

[67] K.S. Perumalla and S. Sundaragopalan. High-fidelity modeling of computer network worms. Coll. of Comput., Georgia Inst. of Technol., Atlanta, GA, USA, 2004.

[68] P Pourbeik. Approaches to validation of power system models for system planning studies. In *Power and Energy Society General Meeting, 2010 IEEE*, pages 1–10. IEEE, 2010.

[69] A. Poylisher, C. Serban, J. Lee, T. Lu, R. Chadha, C.-Y.J. Chiang, K. Jakubowski, and R. Orlando. Virtual ad hoc network testbeds for high fidelity testing of tactical network applications. 2009.

[70] Carlos Queiroz, Abdun Mahmood, and Zahir Tari. Scadasimâ€"a framework for building scada simulations. *Smart Grid, IEEE Transactions on*, 2(4):589–597, 2011.

[71] Bradley Reaves and Thomas Morris. An open virtual testbed for industrial control system security research. *International Journal of Information Security*, 11(4):215–229, 2012.

[72] Ram Mohan Reddi and Anurag K Srivastava. Real time test bed development for power system operation, control and cyber security. In *North American Power Symposium (NAPS), 2010*, pages 1–6. IEEE, 2010.

[73] L. Ricciulli. High-fidelity distributed simulation of local area networks. *Proceedings 31st Annual Simulation Symposium*, page 165, 1998.

[74] John S Robin and Cynthia E Irvine. Analysis of the intel pentium's ability to support a secure virtual machine monitor. Technical report, DTIC Document, 2000.

[75] Erick A Salazar and Manuel E Macás. Virtual 3d controllable machine models for implementation of automations laboratories. In *Frontiers in Education Conference, 2009. FIE'09. 39th IEEE*, pages 1–5. IEEE, 2009.

[76] Naoum Sayegh, Ali Chehab, Imad H Elhajj, and Ayman Kayssi. Internal security attacks on scada systems. In *Communications and Information Technology (ICCIT), 2013 Third International Conference on*, pages 22–27. IEEE, 2013.

[77] Karen A Scarfone, Murugiah P Souppaya, Amanda Cody, and Angela D Orebaugh. Sp 800-115. technical guide to information security testing and assessment. 2008.

[78] Thomas J Schriber. Introduction to simulation. In *Proceedings of the 9th conference on Winter simulation-Volume 1*, page 23. Winter Simulation Conference, 1977.

[79] Gregg Schudel and Bradley Wood. Adversary work factor as a metric for information assurance. In *Proceedings of the 2000 workshop on New security paradigms*, pages 23–30. ACM, 2001.

[80] A Shahzad, S Musa, A Aborujilah, and M Irfan. A new cloud based supervisory control and data acquisition implementation to enhance the level of security using testbed. *Journal of Computer Science*, 10(4):652, 2013.

[81] AAmir Shahzad, Shahrulniza Musa, Abdulaziz Aborujilah, and Muhammad Irfan. Secure cryptography testbed implementation for scada protocols security. In *Advanced Computer Science Applications and Technologies (ACSAT), 2013 International Conference on*, pages 315–320. IEEE, 2013.

[82] Christos Siaterlis, Andres Perez Garcia, and Béla Genge. On the use of emulab testbeds for scientifically rigorous experiments. *Communications Surveys & Tutorials, IEEE*, 15(2):929–942, 2013.

[83] Christos Siaterlis and Béla Genge. Cyber-physical testbeds. *Communications of the ACM*, 57(6):64–73, 2014.

[84] James E Smith and Ravi Nair. The architecture of virtual machines. *Computer*, 38(5):32–38, 2005.

[85] Teodor Sommestad and Amund Hunstad. Intrusion detection and the role of the system administrator. *Information Management & Computer Security*, 21(1):30–40, 2013.

[86] YunSik Son and YangSun Lee. Smart virtual machine code based compilers for supporting multi programming languages in smart cross platform. *International Journal of Software Engineering & Its Applications*, 8(5), 2014.

[87] Georgios Spathoulas, Sokratis K Katsikas, and Anastasios Charoulis. A test-bed for intrusion detection systems results post-processing. In *Public Key Infrastructures, Services and Applications*, pages 170–183. Springer, 2014.

[88] Alexandru Stefanov and Chen-Ching Liu. Cyber-power system security in a smart grid environment. In *Innovative Smart Grid Technologies (ISGT), 2012 IEEE PES*, pages 1–3. IEEE, 2012.

[89] Joseph Stites, Ambareen Siraj, and Eric L Brown. Smart grid security educational training with thundercloud: A virtual security test bed. In *Proceedings of the 2013 on InfoSecCD'13: Information Security Curriculum Development Conference*, page 105. ACM, 2013.

[90] Keith Stouffer, Joe Falco, and Karen Scarfone. Guide to industrial control systems (ics) security. *NIST Special Publication*, 800(82):16–16, 2007.

[91] JH Suh, JS Oh, J Choi, J Goff, J Tao, EH Song, P Fu, GS Lee, and KS Eom. Korean r&d on the converter controller for iter ac/dc converters. In *Fusion Engineering (SOFE), 2011 IEEE/NPSS 24th Symposium on*, pages 1–5. IEEE, 2011.

[92] F. (1) Sultan, A. (2) Poylisher, J. (3) Lee, C. (4) Serban, C.J. (5) Chiang, and R. (6) Chadha. Timesync: Enabling scalable, high-fidelity hybrid network emulation. In *MSWiM'12 - Proceedings of the 15th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, number MSWiM'12 - Proceedings of the 15th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, pages 185–194, (1)Applied Communication, Sciences (ACS), 2012.

[93] Yibin Sun, Taotao Ma, Bingfei Huang, Wei Xu, Bin Yu, and Yingwei Zhu. Risk assessment of power system secondary devices for power grid operation. In *Electricity Distribution (CICED), 2012 China International Conference on*, pages 1–5. IEEE, 2012.

[94] Chee-Wooi Ten, Chen-Ching Liu, and Govindarasu Manimaran. Vulnerability assessment of cybersecurity for scada systems. *Power Systems, IEEE Transactions on*, 23(4):1836–1846, 2008.

[95] Vincent E Urias and Brian P Van Leeuwen. Supervisory command and data acquisition (scada) system cyber security analysis using a live virtual and constructive (lvc) testbed. Technical report, Sandia National Laboratories (SNL-NM), Albuquerque, NM (United States), 2012.

[96] András Varga et al. The omnet++ discrete event simulation system. In *Proceedings of the European simulation multiconference (ESMâ€™2001)*, volume 9, page 65. sn, 2001.

[97] Rayford B Vaughn, Thomas Morris, and Elena Sitnikova. Development & expansion of an industrial control system security laboratory and an international research collaboration. In *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*, page 18. ACM, 2013.

[98] Yu Fei Wang, Tao Zhang, Yuan Yuan Ma, and Bo Zhang. An information security assessments framework for power control systems. *Advanced Materials Research*, 805:980–984, 2013.

[99] Arne Widström. Foi-r–4029–se, möjligheter och problem vid analys av fientlig kod riktad mot siemens s7-serie. Technical report, Swedish Defence Research Agency (FOI), 2012.

[100] Sisu Xi, Meng Xu, Chenyang Lu, Linh TX Phan, Christopher Gill, Oleg Sokolsky, and Insup Lee. Real-time multi-core virtual machine scheduling in xen. In *Embedded Software (EMSOFT), 2014 International Conference on*, pages 1–10. IEEE, 2014.

[101] Fan Xu, Li Shen, and Zhiying Wang. A dynamic binary translation framework based on page fault mechanism in linux kernel. In *Computer and Information Technology (CIT), 2010 IEEE 10th International Conference on*, pages 2284–2289. IEEE, 2010.

[102] Yi Yang, Kieran McLaughlin, Sakir Sezer, Timothy Littler, Eul Gyu Im, Bernardi Pranggono, and HF Wang. Multiattribute scada-specific intrusion detection system for power networks. *IEEE transactions on power delivery*, 29(3):1092–1102, 2014.

[103] S.B. Yoginath, K.S. Perumalla, and B.J. Henz. Runtime performance and virtual network control alternatives in vm-based high-fidelity network simulations. In *Simulation Conference (WSC), Proceedings of the 2012 Winter*, pages 1–13, Dec 2012.

[104] Seehwan Yoo, Miri Park, and Chuck Yoo. A step to support real-time in virtual machine. In *Consumer Communications and Networking Conference, 2009. CCNC 2009. 6th IEEE*, pages 1–7. IEEE, 2009.

[105] Juan Zamorano and Juan Antonio de la Puente. Design and implementation of real-time distributed systems with the assert virtual machine.

In *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*, pages 1–7. IEEE, 2010.

[106] Minghui Zhang, Yanxia Lu, and Tianjiao Xia. The design and implementation of virtual machine system in embedded softplc system. In *Computer Sciences and Applications (CSA), 2013 International Conference on*, pages 775–778. IEEE, 2013.

[107] Yu Zhang, Fei Xie, Yunwei Dong, Gang Yang, and Xingshe Zhou. High fidelity virtualization of cyber-physical systems. *International Journal of Modeling, Simulation, and Scientific Computing*, 4(02), 2013.

Appendix A. Survey of field devices

To create a testbed that is representative of the real world, there is a need to understand it. For this purpose, we surveyed the market of field devices (PLCs, RTUs and IEDs). We focused on field devices rather than control center and communication architecture components as the latter both can be virtualized through VirtualBox (and thus are possible to implement in CRATE as-is), while the prior often have specialized hardware, software and logic that are unsupported by current virtualization technologies (and thus require extending the functionality of CRATE).

The survey was carried out by extracting a list of field device manufacturers from three web sites[25,26,27] that contained information regarding ICS hardware and software suppliers[28]. All hits containing the string "manufacture" were chosen and the lists were parsed into a simple database format with one hit per row. We manually deleted rows with obvious duplicates.

The resulting list contains 341 unique company names, of which the ten largest manufacturers of ICS equipment world-wide[29] are ABB, Alstom, Emerson Electric, General Electrics, Honeywell, Omron, Rockwell Automation, Schneider Electrics, Siemens and Yokogawa. The product ranges of these companies were not studied in depth. However, it is safe to say that there is a large variety of PLCs around. For example, there are five different Siemens S7 PLC product series alone (with an unknown number of submodels, see Appendix C). As field devices have extremely long lifespans (manufacturers claim up to 40 years [93]), there is bound to be products from different developers, series and generations in real-world operation. For example, the Swedish railroad system has operational switchgear that was developed between the 1960s (without embedded computers altogether) to the 2010s (that consist of a collection of servers, including common IT protocols such as DHCP and SMB) [61].

Consequently, if the market shares are fairly evenly distributed among the ICS manufacturers, virtualization, emulation or simulation should be used as a general, possibly modularized, platform instead of product specific platforms.

---

[25] http://www.plcs.net/chapters/links.htm,

[26] http://www.thomasnet.com/products/controllers-programmable-logic-plc-18190900-1.html

[27] http://www.automation.com/suppliers/automation-product-manufacturers/product-category/programmable-logic-controllers-plcs

[28] A future project will survey the Swedish ICS market, perhaps through a questionnaire in conjunction with an FOI course that is attended by personnel from the major ICS operators in Sweden.

[29] According to http://www.reportlinker.com/p02131164-summary/SCADA-Market-by-Components-PLC-RTU-HMI-Communication-Systems-Architecture-Hardware-Software-Services-Application-Oil-Gas-Power-Water-Wastewater-Transport-Manufacturing-Chemicals-and-Geography-Analysis-Forecast-to.html

Appendix B. Overview of a PLC

A PLC (Programmable Logical Controller) is a small industrial computer that is designed to operate in harsh environments (e.g., cold, hot or moist), where it performs logical simple instructions and is connected to real physical hardware like water dams or oil platforms.

PLCs have analog and/or digital inputs and outputs. They can also have relay outputs, serial communication, and motion and process control. Modern PLCs often have a LAN port to support IP based traffic (TCP/IP). PLCs can have different extension modules like additional physical outputs or extended communication capabilities (e.g., an additional Ethernet port).

The internal properties of a PLC include timers, hardware and software interrupts, ladder logic programming capabilities, different functional blocks (e.g., organization blocks (OB), see the next section), counters and real-time characteristics.

There are a plethora of PLC manufacturers and most of them have many different PLCs for different working environments. As it is a computer it contains a CPU, firmware, memory modules and a hard drive function (typically a flash chip). Commonly used CPUs are Motorola 68000, Motorola 68020, Motorola 68030, Motorola 68040, Intel 80C186, Intel 80C386 and Intel 80C486. FPGAs and ASICS are also used.

The tasks that are to be conducted by a PLC are determined during its scan cycles, where the PLC reads inputs, writes outputs, executes user program instructions and performs system maintenance and background processing. A scan cycle is triggered by a timer, hardware interrupt or software interrupt (specific condition). A PLC typically has real-time requirements regarding when tasks must be completed. Thus, it is not enough to complete a task - it has to be done in a specified time frame.

A scan cycle is carried out in the following manner: First the PLC reads the physical inputs and stores them in the process image input area memory (in RAM). It is important to write the inputs into the process memory so they do not change during the execution phase. If different calculations use the same input data it needs to stay the same. It then executes the user program instructions and calculations. It starts with the lowest number OB block and steps through them in numerical order. An OB cannot call another OB to perform an operation. That rule ensures that all OBs will be executed in sequential order. Then, the output values are updated in the process image area. Finally, the resulting outputs are written to the physical outputs.

There is a diversity of automation (industrial) protocols used by PLCs [45], [46]. Some are standardized[30] (e.g., Modbus and DNP3); some are proprietary and undocumented (see e.g. Appendix D). Commonly used wired protocols include CIP (Common Industrial Protocol), Modbus (RTU, ASCII or TCP), DNP3, IEC 60870-5, IEC 61850, IEC 62351 (a security layer added to other protocols such as GOOSE), OPC, Profinet IO and CAN. Commonly used wireless protocols include ZigBee and HART.

The protocols have a diversity of functionality. Important ones are querying a state for a field device, query if a switch is on or off. Other uses are to set a switch to either on or off or to collect measurement data, for example temperature readings or a water level. The main functions are either collecting data or setting data on the device.

The protocols support a variety of other functionalities that are rarely used under regular run operation. Two examples are to update the PLC firmware or to change the ladder logic in the PLC.

[30] The actual implementation of a standardized protocol can however be proprietary.

Appendix C. Siemens S7 technical analysis

Siemens SIMATIC S7 is the sixth and current generation of Siemens control systems, with its oldest predecessor SIMATIC G launched in 1959[31]. The S7 generation in turn consists of a number of product lines: S7-200, S7-300, S7-400, S7-1200 and S7-1500. The S7-200 is an obsolete product line which has already been superseded by the S7-1200. These two represent the low end of S7 PLCs. The S7-300 contains the middle range PLCs and the S7-400 the high end. The most recent product line is the S7-1500, launched in 2013. Until 2020 it will only complement the S7-300 and S7-400 product lines but then finally succeed them[32].

The following description is based upon FOI studies of the S7-400 and the S7-1200 PLCs. Parts of the results have previously been documented in an FOI report [99].

The S7-400 and the S7-1200 are very different when compared at a low level. They use different machine languages and different communication protocols for the configuration of the PLCs.

The S7-400 protocol stack consists of several layers. Ethernet is at the lowest level, followed in turn by IP, TCP, ISO-TSAP (ISO Transport Services Access Protocol) and ISO 8073, also known as COTP (Connection Oriented Transport Protocol). The role of the last two protocols is more or less to enable the use of ISO-style connection oriented protocols on top of TCP. Finally, above COTP, we find the proprietary Siemens S7 protocol. The S7 protocol is somewhat extensive, but large parts of it have been reverse engineered by researchers outside of Siemens.

The S7-1200 protocol stack also consists of several layers, with Ethernet at the lowest level followed by IP, TCP, ISO-TSAP and COTP. The protocol on top of COTP is however completely different from the old S7 protocol used by the S7-400. A small part of the protocol has been reverse engineered for internal use at FOI. This resulted in the discovery of two novel denial of service vulnerabilities in the S7-1200 PLCs. It also made it possible to create a handful of demonstration tools for control systems security courses held at FOI. The S7-1200 protocol was partly reversed in both its second and third versions. The third version has some, although lacking, protection against replay attacks. This protection was also circumvented through the FOI research.

---

31 http://www.siemens.com/innovation/en/publikationen/publications_pof/pof_spring_2005/history_of_industrial_automation.htm

32 https://support.industry.siemens.com/cs/#document/67856446?lc=de-WW

The S7-400 PLCs implement a kind of virtual machine on top of the hardware CPU. For example, inside each of the S7-400 CPU units available at FOI are two Infineon TriCore processors. While a long unconditional jump in this architecture begins with the machine code byte 0x1d, a similar unconditional jump inside the PLC virtual machine begins with the machine code bytes 0x70 0x0b. The machine code inside the virtual machine is called MC7 (Machine Code 7) in the S7-400 architecture. The corresponding assembler language is called STL (STatement List). The STL language is entirely documented by Siemens, while the MC7 machine code is undocumented. However, independent researchers have been able to reverse engineer large parts of MC7. Further, some of the blank spaces have been reverse engineered at FOI for internal use.

The S7-1200 PLCs utilize a completely different kind of machine code compared to the S7-400 PLCs. At this point we have not been able to identify the exact architecture of the hardware CPUs used in the S7-1200 series. Neither do we have any information about the PLC machine code other than that it is dissimilar to MC7. The two machine codes may even be the same in this architecture, with no virtual machine layer present. Reversing the machine code for S7-1200 is harder than reversing MC7 because there is no STL for the S7-1200 product line. It follows that one cannot easily insert known instructions and then investigate the corresponding machine code. Instead, an unknown number of unknown instructions are generated from each atomic higher level construct with the available development tools.

We have also studied various block headers and footers internal to the S7-400 system. These are undocumented by Siemens as well, and they also vary in exact layout depending on where in the system they are found. We have been able to reverse a few of these in quite good detail, enabling us to decode some further information from the Stuxnet worm than previously published.

Too much of the S7-400 MC7 level architecture is still completely undocumented to realistically start the design of a high accuracy emulator at the MC7 level. The S7-1200 is still mostly a large unknown, which may finally turn out to be either easier or harder to emulate with high accuracy. Another possibility is to emulate at the hardware level instead of the MC7 level, which is something we have only recently started to investigate.

Finally, it might be possible to build something based upon the Siemens SIMATIC PLCSIM simulator. The feasibility of this solution depends both on the low level accuracy of PLCSIM and on the cooperation of Siemens. At present we however have very limited low level knowledge regarding the PLCSIM.

Appendix D. Categorization framework

#	Variabel level 1	Variabel level 2	Description	Example
1	Type of contribution	Testbed/Other	If the paper discusses a testbed	Testbed
2	Objectives	-	What testbed objectives that are given	Vulnerability analysis, security forensics
3	Fidelity	Discussion	If fidelity is discussed in the paper	Is discussed
		Data collection	Suggestions for data collection (to yield realistic testbeds)	Portscanning with Nmap
		Metrics	Specific metrics for measuring fidelity that are mentioned	Packet arrival rate for Modbus TCP
4	Implemented	-	If the testbed has been implemented	Yes
5	Protocols	-	ICS-specific protocols that are mentioned	OPC, Modbus
6	Devices	-	ICS-specific product types that are mentioned	MTU, Database Historian
7	Control System	Virtualization	Virtualization solutions that are employed	VMware, VirtualBox
		Simulation	Simulation slutions that are employed	Matlab, Opnet
		Emulation	Emulation solutions that are employed	QEMU
		Hardware	Employed hardware	ABB WS500
8	Field devices	Virtualization	Virtualization solutions that are employed	VMware, VirtualBox
		Simulation	Simulation slutions that are employed	Matlab, Opnet
		Emulation	Emulation solutions that are employed	QEMU
		Hardware	Employed hardware	Siemens S7 PLC
9	Process	Simulation	Simulation slutions that are employed	Matlab, Opnet
		Hardware	Employed hardware	Power system
10	Communication	Virtualization	Virtualization solutions that are employed	VMware, VirtualBox
		Simulation	Simulation slutions that are employed	Matlab, Opnet
		Emulation	Emulation solutions that are employed	QEMU
		Hardware	Employed hardware	Cisco router
11	Other	-	Information that do not fit any other category	-

Appendix E. Cyber Range And Training Environment (CRATE)

A cyber range makes it possible to design and deploy IT environments of considerable size and complexity, and expose them to cyber security threats under realistic conditions without putting operational systems at risk. The Swedish Defence Research Agency (FOI) develops and maintains a cyber range named Cyber Range And Training Environment (CRATE)[33], which is used by FOI for cyber security research and training.

In terms of hardware, the infrastructure consists of a server room, some 350 servers, network switches, network cables and auxiliary equipment (e.g., specially designed portable devices that allow secure remote access). However, the main component and the more costly and complicated part of CRATE is the software framework necessary to enable the use of the hardware for cyber security research and training. The framework consists of a set of tools, which include for example:

- A web-based interface (CrateWeb) to specify desired computer networks
- Software packages and desktop software applications of various types and versions.
- A library of virtual machines (VMs) with different operating systems and installed applications.
- Scripts to automatically deploy virtual machines and networks.
- A scripting infrastructure to configure individual virtual computers (e.g., network interfaces, hostnames, users and passwords).
- Tools to monitor and log events taking place in the infrastructure.
- Tools supporting the analysis and synchronous replay of data streams.

CRATE is also equipped with prototypic software-based user agents that can generate user activities in the environment (e.g., sending emails or surf the web) and prototypic systems for producing and observing typical cyber-attacks (e.g., computer viruses).

CrateWeb, the web-based configuration tool (see Figure 2) can be used to design computer networks consisting of any combination of virtual machines deployed on the 350 servers. It can also be used to deploy a wide range of operating systems and applications with different vulnerabilities, and specify the users of each system. Some manual configurations and tuning of scripts may still be required to deploy uncommon configurations, operating systems and applications that currently are not covered by CRATE's application library. The goal is that

[33] www.foi.se/crate

every configuration activity performed regularly should be handled through the management interface.

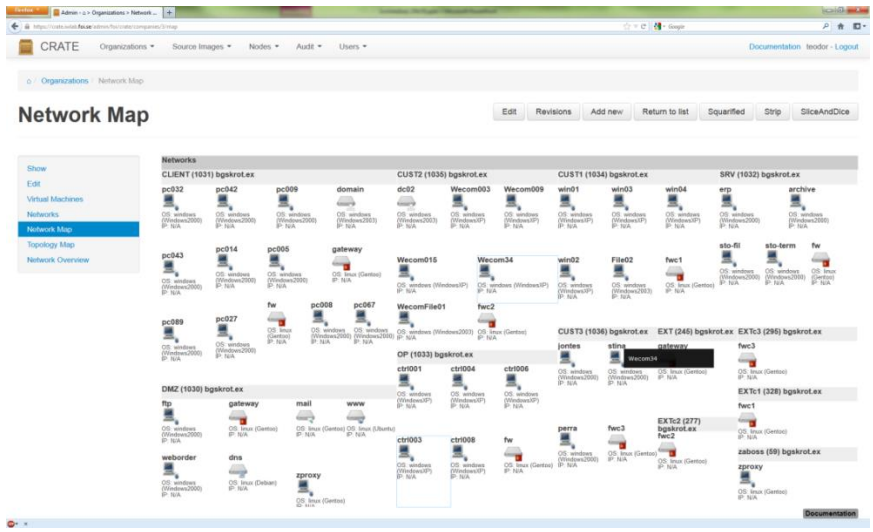


Figure 2. Screenshot of the web-based configuration tool of CRATE (CrateWeb), illustrating the virtual machines and network topology of a computer network.

An overview of CRATE's technical architecture can be seen in Figure 3. Networks with virtual machines, configured using CrateWeb, are deployed by a *command and control server* through a separate *administration network* to any of the 350 existing servers. The specific configuration of coexisting virtual machines then operate in an Internet-wise isolated environment denoted as the *game network*. Systems within the game network can then be accessed by infrastructure administrators through the application programming interface provided by the hypervisor VirtualBox[34] (e.g., to set up computer networks). The operations performed include (but are not limited to):

- Import and export of virtual machines.
- Manipulation of a virtual machine's virtual hardware.
- Execution of programs inside virtual machines.
- File operations, for example copying files and folders to and from virtual machines.
- Access to the virtual machines' native graphical user interfaces (through a remote desktop protocol server built into VirtualBox).

[34] VirtualBox is the virtualization technology that is used for hosts in CRATE. See https://www.virtualbox.org/.

Actors (scripted and human) in the exercise or experiment interface the infrastructure through the standard interfaces of a computer under the restrictions (e.g., firewall rules) defined in the infrastructure. Consequently, activity within the game network can be observed from either the infrastructure administrators' point of view (with no restrictions) or from the perspective of normal users' or attackers' point of view (under the restrictions given by the infrastructure administrator).

Since the game network is an isolated environment, Internet backbone technologies that are taken for granted, such as DNS Root name servers[35] and a router infrastructure, have been implemented in CRATE.

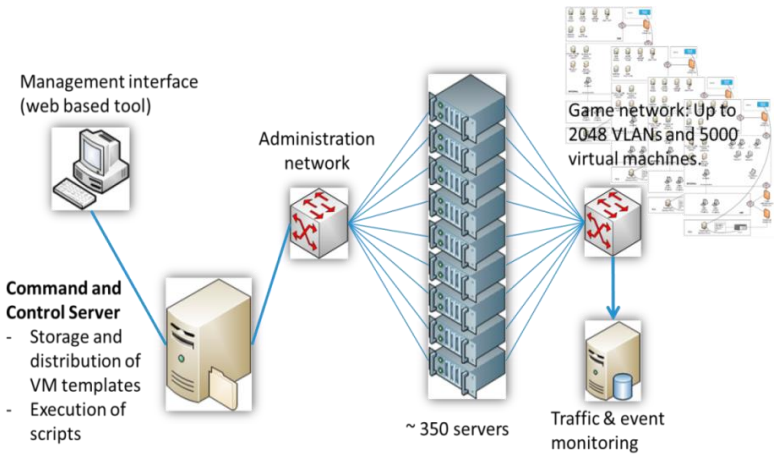


Figure 3. An overview of CRATE's technical architecture.

In its current state, CRATE makes it possible to virtualize large computer networks, and efficiently deploy and configure a large number of virtual machines to create sizeable computer networks of various types. For instance, in an experiment performed in 2012, more than one thousand virtual machines were deployed in over eighty different computer networks. The computer networks can be designed from scratch to fit some particular need or be generated based on templates of standardized environments (e.g. representing cyber environments of industrial production facilities, schools, hospitals or newspapers). CRATE has been designed with cyber security testing in mind (for instance with respect to software tools and machine templates). Thus, while other domains may also benefit from the emulation of large computer networks, CRATE is primarily constructed to meet the needs associated with research and education related to cyber security.

[35] http://www.internetsociety.org/internet-domain-name-system-explained-non-experts-daniel-karrenberg

Critical societal functions such as electricity and water purification depend on Industrial Control Systems (ICS) to properly function. Not long ago, these ICS were realized by specially constructed isolated devices. Along with the rest of our society, ICS have evolved and are now often delivered by complex interconnected IT solutions including commercial-off-the-shelf technologies that in one way or another are connected to the Internet. As a consequence, ICS are vulnerable to IT attacks similarly to most other IT systems.

Due to the extreme availability requirements on ICS in operation, it is difficult to perform cyber security experiments on them, such as vulnerability discovery or tests of defense mechanisms. To accommodate such experiments, researchers and practitioners turn to testbeds that mimic real ICS.

This study first surveys ICS testbeds that have been proposed for scientific research. Special focus is given to field devices, a kind of ICS component that is considered particularly challenging to implement in testbeds. It then compares these results with findings from product surveys, practical experiences, and interviews with a manufacturer. The outcomes of this comparison are methods and tools for creating a high-fidelity ICS testbed.

The study was conducted in collaboration with other actors, in particular, the Idaho National Laboratory.

FOI
Swedish Defence Research Agency
SE-164 90 Stockholm

Phone: +46 8 555 030 00
Fax: +46 8 555 031 00

www.foi.se