Technical Report

UCAM-CL-TR-963
ISSN 1476-2986

Number 963

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

Capability-based access control for cyber physical systems

Michael G. Dodson

October 2021

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

https://www.cl.cam.ac.uk/

This technical report is based on a dissertation submitted July 2021 by the author for the degree of Doctor of Philosophy to the University of Cambridge, Queens' College.

Technical reports published by the University of Cambridge Computer Laboratory are freely available via the Internet:

https://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

Abstract

Capability-based access control for cyber physical systems

Michael Dodson

Cyber Physical Systems (CPS) couple digital systems with the physical environment, creating technical, usability, and economic security challenges beyond those of information systems. Their distributed and hierarchical nature, real-time and safety-critical requirements, and limited resources create new vulnerability classes and severely constrain the security solution space. This dissertation explores these challenges, focusing on Industrial Control Systems (ICS), but demonstrating broader applicability to the whole domain.

We begin by systematising the usability and economic challenges to secure ICS. We fingerprint and track more than 10 000 Internet-connected devices over four years and show the population is growing, continuously-connected, and unpatched. We then explore adversarial interest in this vulnerable population. We track 150 000 botnet hosts, sift 70 million underground forum posts, and perform the largest ICS honeypot study to date to demonstrate that the cybercrime community has little competence or interest in the domain. We show that the heterogeneity, cost, and expertise required for large-scale attacks on ICS are economic deterrents when targets in the IoT domain are available.

The ICS landscape is changing, however, and we demonstrate the imminent convergence with the IoT domain as inexpensive hardware, commodity operating systems, and wireless connectivity become standard. Industry's security solution is boundary defence, pushing privilege to firewalls and anomaly detectors; however, this propagates rather than minimises privilege and leaves the hierarchy vulnerable to a single boundary compromise.

In contrast, we propose, implement, and evaluate a security architecture based on distributed capabilities. Specifically, we show that object capabilities, representing physical resources, can be constructed, delegated, and used anywhere in a distributed CPS by composing hardware-enforced architectural capabilities and cryptographic network tokens. Our architecture provides defence-in-depth, minimising privilege at every level of the CPS hierarchy, and both supports and adds integrity protection to legacy CPS protocols. We implement distributed capabilities in robotics and ICS demonstrators, and we show that our architecture adds negligible overhead to realistic integrations and can be implemented without significant modification to existing source code.

Acknowledgements

There are many fingerprints on this work that are not my own, and there are many people to and for whom I am very grateful.

First, I thank God for the gifts he has given me, for bringing me to live in a place that I love, and for helping me grow through work that I enjoy.

I am grateful to my supervisor, Alastair Beresford, for teaching me how and why to do research. You regularly pushed me into free and open spaces, and your encouragement helped me grow from rather than flounder in the disorientation. Thank you for your wisdom and guidance, and for the confidence you gave me to try new things.

I have been blessed by the diverse mix of research ideas and experiences that make up the Digital Technology Group. I would like to thank Daniel Thomas, Stan Zhang, Diana Vasile, Jovan Power, Martin Kleppmann, and Daniel Hugenroth for their friendship in and out of the lab. Your ideas, expertise, and support are evident throughout this thesis.

From the Cybercrime Centre, I would like to thank Richard Clayton and Ross Anderson for letting me come alongside the group and benefit from their vast experience. I am also particularly grateful to Alexander Vetterl, who became both a friend and invaluable mentor during my project, and to Ilia Shumailov, who never let me rest on my oars.

The opportunity to work with the CHERI research team was a humbling and pleasurable challenge. I am thankful to Robert Watson and Simon Moore for their guidance, and I am indebted to Brett Gutstein, Alexander Richardson, and Jessica Clarke for multiple heavy lifts getting me up to speed with CHERI. I want to give particular thanks to Hesham Almatary: you have been a great research partner and friend, and you have given more than you have received. Thank you.

Outside the lab, I am grateful to the Gates Cambridge Trust for their generous funding of my PhD, to Éireann Leverett for sharing his friendship, experience, and creativity, and to John and Diane Lister for always providing us a home. I am also grateful to my PhD examiners, Professors Simon Moore and Steven Murdoch, for their attentive and interested assessment of my work, feedback, and general encouragement.

Finally, to my wife, Jennifer: eighteen-ish months of intermittent lockdowns was not how we expected to spend our years here, yet the long-sought, but unplanned opportunity to work sitting next to you was an encouragement and a joy. This dissertation is the most tangible artefact of that era, and it is, of course, dedicated to you.

Contents

Glossary

ACL Access Control List.

API Application Programming Interface.

AUTOSAR AUTomotive Open System ARchitecture.

AWS Amazon Web Services.

CA Certificate Authority.

CAN Controller Area Network.

CCCC Cambridge Cybercrime Centre.

CFI Control Flow Integrity.

CFU Compact Field Unit.

CHERI Capability Enhanced RISC Instructions.

CIA Confidentiality, Integrity, Availability.

COTS Commercial Off-The-Shelf.

CPS Cyber-Physical System.

DLR Device-Level Ring.

FDI False Data Injection.

FPGA Field Programmable Gate Array.

HMI Human-Machine Interface.

ICS Industrial Control System.

IDS Intrusion Detection System.

IIC Industrial Internet Consortium.

IIRA Industrial Internet Reference Architecture.

IoT Internet of Things.

ISA Instruction Set Architecture.

IT Information Technology.

MAC Message Authentication Code.

MCS Mixed Criticality System.

MMU Memory Management Unit.

NAT Network Address Translation.

NIST National Institute of Standards and Technology.

NVD National Vulnerability Database.

OPC Open Platform Communications.

PLC Programmable Logic Controller.

PLOC Physical Lines of Code.

PPI Pay-Per-Install.

RBAC Role-Based Access Control.

ROS Robot Operating System.

RTOS Real-Time Operating System.

SAS Shared Access Signature.

SCADA Supervisory Control and Data Acquisition.

SDSI Simple Distributed Security Infrastructure.

SFI Software Fault Isolation.

SIEM Security Information and Event Management.

SLAM Simultaneous Localisation and Mapping.

SMB Server Message Block.

SPKI Simple Public Key Infrastructure.

SSH Secure SHell.

TGT Ticket-Granting Ticket.

TLS Transport Layer Security.

TOFU Trust-on-First-Use.

UA Unified Architecture.

WCET Worst-Case Execution Time.

chapter 1

Introduction

Cyber-Physical System (CPS) is an umbrella term for a domain that tightly couples hardware and software with sensing and manipulation of the physical environment. CPS covers the natively-digital robotics and Internet of Things (IOT) subdomains, and increasingly includes Industrial Control Systems (ICS), vehicular control, and avionics, all of which are integrating digital devices into historically analogue systems.

Most CPS are distributed systems, consisting of heterogeneous devices connected and cooperating in a hierarchical system-of-systems. Historically, these connections were point-to-point serial links, but they increasingly use IP-based networks (including wireless) to communicate between devices and to networks up and down the hierarchy. Size, cost, and power constraints drive many CPS designers to microcontrollers with limited compute resources and flat, physical memory spaces (i.e., no Memory Management Units (MMUs)) [219], precluding standard Information Technology (IT) memory safety techniques. Additionally, rigorous testing to certify CPS' safety-critical functions, combined with deployed lifetimes measured in decades, create strong economic pressure against patching [114]. This patch resistance is in tension with the increasing connectivity of these devices, as efficient Internet-wide scanning makes it trivial to find vulnerable, Internet-connected devices [56, 136]. Finally, CPS are integrations of devices from multiple manufacturers cooperating on critical tasks without mechanisms to ensure trusted inter-device communication. For example, the electronic suites in many modern vehicles are integrations of dozens of Electronic Control Units (ECUs) from multiple manufacturers communicating using the unsecured, broadcast Controller Area Network (CAN) bus protocol. In summary, CPS operate in challenging environments that constrain efforts to apply well-established IT security practices, resulting in systems with course-grained authority mechanisms, lowering the bar for malicious actors to access a device, manoeuvre within a network, and modify system behaviour.

Despite these challenges, IoT is the only CPS subdomain that has seen significant exploitation. While there have been limited attacks on ICS with catastrophic, physical effects (e.g., Stuxnet's destruction of Iranian Uranium enrichment centrifuges [101] and the CRASHOVERRIDE attack on the Ukrainian power grid [38]), these have been infrequent, targeted nation-state efforts. Scaled attacks against critical infrastructure (e.g., oil, gas, and electrical distribution; hospitals; water and sanitation), have all been ransomware attacks against Windows-based infrastructure, targeting the data necessary for safe operation, but without specific knowledge or targeting of the physical systems themselves. Examples

include Norsk Hydro, which was forced to shut down global aluminium operations [48], and Colonial Pipeline, which was forced to shut down fuel distribution along the entire US east coast [81]. To date, there have not been any attacks against automotive, robotics, or aviation subdomains in the wild.

One explanation for this relative peace (compared, at least, to IoT or IT) could be that these CPS subdomains have addressed their security challenges and only deploy secure systems. For example, the ICS subdomain long claimed that their systems were protected by an 'air gap', and that vigilantly maintaining the gap between their local network and the Internet allowed them to avoid common IT security paradigms, such as patching and encryption, that appeared to conflict with other operational requirements. While an 'air gap' may be an effective component of a defence-in-depth strategy, the common reliance on an 'air gap' alone was publicly and sensationally shown to be ineffective against nation-state attackers by the Stuxnet attack [101], and more broadly and systematically dismantled by Leverett *et al.*'s demonstration that thousands of ICS from many critical industries were directly connected to the Internet without any access control or communication security [115]. In fact, multiple studies on the security of the ICS subdomain conclude that the "space is in disarray" [136, 163]. Similar proclamations have been made about other CPS subdomains based on catalogues of proof-of-concept malware samples and attack demonstrations [68, 134, 192].

We explore an alternative explanation, focusing on the economics of such attacks against CPS. We present a novel security economics model for large-scale cybercrime against both the IT and CPS domains, showing that factors such as a fragmented population, high subject matter expertise requirements, and the limited availability of public tools and exploits push would-be attackers toward the larger, and more homogeneous IoT and IT domains. Our model also helps predict the effects of future changes to the subdomain. For example, we demonstrate that the ICS subdomain is converging with the IoT subdomain in terms of homogeneity and connectivity, and we postulate that this convergence will both make ICS devices attractive targets in their own right, and that they will be wrapped up in larger attacks against the IoT subdomain. We take the first step to explore this hypothesis using ICS and IoT honeypots and targeted scanning of Internet-connected hosts, and we present the first empirical evidence of Internet-facing convergence of ICS and IoT networks, including networks hosting ICS and malware-infected IoT.

While the security model that relies solely on an 'air gap' is generally defunct, the operational requirements and pressures that make it so attractive survive in both deployed security solutions and those actively being researched. Specifically, deterministic timing, high availability requirements, and brownfield deployments (i.e., backfitting new equipment into existing, legacy systems) all discourage solutions that require authentication, authorisation, or encryption between controllers, sensors, and actuators. Instead, the industrial and automotive subdomains rely heavily on boundary defence and network segmentation: collecting devices with a common task within a common boundary and assuming complete trust between them. While this is an improvement over fallacious 'air gap' security assumptions, it suffers from the same problem: any compromise of the boundary compromises the whole cell. Similarly, behaviour-based anomaly detection systems, while making stronger assumptions about the adversary, are as complex as the systems they protect and present maintenance and operational challenges without addressing the fundamental problems of privilege and trust.

While network security (either boundary defence or anomaly detection) may be im-

portant parts of defence-in-depth, they do not constitute a robust security architecture for distributed systems, as they outsource privilege to the boundary or monitor device, which become single points of failure or concentrated targets. This has been exploited in high-profile proof-of-concept automotive hacks [134] and real-world, nation-state attacks on critical infrastructure (e.g., Ukrainian power grid [38]).

In contrast, we propose, implement, and evaluate a security architecture based on distributed capabilities, creating a true, least privilege decomposition at every level of the CPS hierarchy. We show that object capabilities, representing attributes of physical resources (e.g., motor speed, valve position, coil energisation), can be constructed, delegated, and used anywhere in a distributed CPS. We demonstrate a novel composition of Capability Hardware Enhanced RISC Instruction (CHERI) architectural capabilities and Macaroon cryptographic capabilities, providing hardware-enforced access control to objects on a device and cryptographic protections across the network, guaranteeing unforgeability, provenance, and monotonicity of our capabilities at every level of the hierarchy. We also show that this composition of capabilities allows CPS to leap-frog use of MMUs, providing software-defined access control at a finer granularity than is possible with an MMU, while simplifying reasoning about authority on a device and across a networked CPS.

1.1 Contributions

In this dissertation we propose, implement, and evaluate a distributed capability architecture for CPS, providing an end-to-end, minimally-privileged access control solution for physical resources in a highly-constrained security environment. The architecture is motivated by a demonstration of the growing vulnerability of the CPS domain, based on both the changing hardware and software landscape of the domain and the economics of attacking the CPS population.

- We provide the first longitudinal study of Internet-connected ICS devices, including a demonstration that individual ICS devices can be retroactively fingerprinted and tracked over several years using publicly available scanning data (Chapter 3).
- We develop a security economics model for characterising and predicting the success of large-scale attacks against an Internet-connected population, and we provide an empirical demonstration of our model using the population of Internet-connected ICS devices (Chapter 3).
- We perform the largest, high-interaction ICS honeypot study to date and are the first to integrate ICS and IoT honeypot data to demonstrate that the convergence of vulnerable ICS and malware-infected IoT networks is already apparent (Chapter 4).
- We provide a novel systematisation of CPS access control challenges to explain why common access control measures (e.g., Access Control Lists (ACLs)) have proved unable to protect CPS systems after boundary compromise, and we demonstrate that capabilities can uniquely meet these challenges with distributed policies and enforcement (Chapter 5).
- We propose a novel approach for composing local architectural capabilities and distributed cryptographic capabilities to allow uniform designation and consistent

enforcement of access control to distributed CPS physical resources, and we provide a concrete implementation of our design pattern using CHERI architectural capabilities and Macaroon cryptographic capabilities (Chapter 6).

- Finally, we perform two case studies, one using a robotics application and another using an industrial application, to show that a distributed CPS with a capability-based security architecture is performant and addresses CPS security challenges with minimal perturbation to application code (Chapter 6).

Throughout, we endeavour to respect the operational challenges of the CPS domain, particularly the constraints of brownfield deployments and the safety- or life-critical nature of some of the devices. In our longitudinal and honeypot studies, we strictly adhere to ethical data-gathering standards to avoid any perturbation of live system behaviour. In implementing and evaluating our distributed capability architecture, our emphasis was on compatibility with existing devices, software, and performance requirements. We sought to demonstrate that tangible security benefits could be obtained by upgrading individual devices in a brownfield environment and that the hardware and software solutions were sufficiently practical to consider our architecture in a production system. To this end, we used existing operating systems, protocol libraries, and cryptographic libraries, all of which are well known and used in production systems. Similarly, while we made extensive use of prototype hardware, we have a reasonable expectation that production hardware supporting architectural capabilities will be available within a few years. While few benchmark suites exist for our applications, we leveraged the unit and integration tests packaged with the software and evaluated performance against existing systems.

1.2 Publications

Much of the research described in this thesis has been published in the proceedings of peer-reviewed journals, conferences, and workshops. Below is a list of those publications and an enumerated distinction between my own contributions and those of my co-authors.

1. Michael Dodson, Alastair R. Beresford, and Daniel R. Thomas. 'When will my PLC support Mirai? The security economics of large-scale attacks against Internet-connected ICS devices.' In *Proceedings of the 15th Symposium on Electronic Crime Research (eCrime '20)* [56].

   This paper forms the basis of Chapter 3. Daniel Thomas provided the initial idea to perform a longitudinal study, and he set up the initial database to find overlap between ICS devices and Mirai hosts. I performed all data collection, fingerprint generation, and processing. I also performed the case studies and created the security economics model used throughout the paper.

   I'm grateful to Daniel Thomas for his initial idea, for setting me up with the Mirai database, and his continued support throughout the data processing stage. I am also grateful to Alastair Beresford, Éireann Leverett, Alexander Vetterl, and Ross Anderson for their feedback and regular discussions.

2. Michael Dodson, Alastair R. Beresford, and Mikael Vingaard. 'Using Global Honeypot Networks to Detect Targeted ICS Attacks.' In *Proceedings of the 12th International Conference on Cyber Conflict (CyCon '20)* [57].

This paper forms the basis of Chapter 4. Mikael Vingaard, from Industrial Defenica (formerly SecuriOT), owns and maintains the honeypot network and provided the primary dataset used in this chapter. I performed the data processing, identified relevant attacks, and executed the comparison with other studies and datasets.

I am grateful to Alastair Beresford for initiating our relationship with Industrial Defenica and for his feedback and support. I am also grateful for Mikael Vingaard's willingness to share his dataset and help interpreting the data.

3. Michael Dodson, Alastair R. Beresford, Alexander Richardson, Jessica Clarke, and Robert N. M. Watson. 'CHERI Macaroons: Efficient, host-based access control for cyber-physical systems.' In *Proceedings of the IEEE European Symposium on Security and Privacy Workshops (EuroS&PW '20)* [58].

   This paper contributes to the content of Chapter 6. Alexander Richardson and Jessica Clarke helped immeasurably with porting the Robot Operating System (ROS) to run on CheriBSD, which required changes to ROS libraries, CheriBSD, and parts of the CHERI toolchain. Martin Kleppmann provided the initial inspiration for mapping CHERI capabilities to Macaroon cryptographic capabilities. I conceived of the idea to use capabilities to represent physical resource attributes and developed the application to demonstrate the concept.

   I am grateful to Martin Kleppmann for the initial idea and to Alexander Richardson and Jessica Clarke for their support in porting ROS to CheriBSD. I am also grateful to Alastair Beresford and Robert Watson for regular discussions and feedback.

4. Michael Dodson, Hesham Almatary, Alastair R. Beresford, and Robert N. M. Watson. 'A distributed capability security architecture for cyber physical systems.' Submitted to *IEEE Transactions on Information Forensics and Security*.

   This paper forms the basis of Chapter 5 and contributes to the content of Chapter 6. Hesham Almatary was responsible for porting the FreeRTOS library operating system to CHERI (CheriFreeRTOS) and supported the evaluation of the Modbus demonstration. He also architected the CHERI compartmentalisation framework for CheriFreeRTOS. I developed the idea of using architectural capabilities to construct complex object capabilities and mapping these across to network capabilities, and I developed the application to test and evaluate these mappings.

   I am grateful to Hesham Almatary for his partnership throughout this work, both in providing a technical basis for the demonstration and his discussion and support while implementing and evaluating the demonstration. I am also grateful to Alastair Beresford and Robert Watson for their regular feedback and discussions.

CHAPTER 2

Background

This chapter provides background information and the current state of research to support and contextualise the dissertation.

While the dissertation deals broadly with Cyber Physical Systems (CPS), our primary focus is on Industrial Control Systems (ICS), whose design and operational constraints exacerbate all the challenges of securing CPS. Therefore, we start this chapter with an ICS primer (Section 2.1), which includes an overview of the state of ICS security (Section 2.1.1). We also discuss Internet-wide scanning (Section 2.1.2) and honeypots (Section 2.1.3), tools and techniques we use in Chapters 3 and 4 to understand the current state of ICS security and to motivate development of a distributed security architecture for CPS.

A distributed security architecture for CPS presents challenges both on the device and across the network. We address these challenges in detail in Chapter 5; however, two of these challenges, the limited availability of memory management hardware and lack of trust between networked devices, require broader contextualisation, which we provide here. Specifically, we discuss memory management in terms of Memory Management Units (MMUs), Memory Protection Units (MPUs), and Capability Hardware Enhanced RISC Instruction (CHERI) architectural capabilities (Sections 2.2 and 2.3). We then discuss trust between networked devices, comparing public key and token based authentication and authorisation mechanisms, and then comparing tokens used or proposed for the CPS domain (Section 2.4).

2.1 Industrial Control Systems (ICS)

ICS are used to command, manage, or regulate devices or physical systems in industry (e.g., chemical processing, automotive manufacturing), infrastructure (e.g., power generation and distribution, gas pipeline control) and building automation (e.g., elevator control, fire suppression, climate control). ICS are often composed of many devices and systems, and may be physically distributed over a large area. Devices communicate using ICS-specific protocols, most of which are legacy point-to-point or broadcast protocols designed with the assumption that devices are connected with dedicated cabling; however, many of these protocols are now layered on top of Ethernet and TCP or UDP, and devices use existing IP-based networks, including the Internet, to communicate.

Control systems generally leverage a feedback loop, where sensed input from the physical process under control is evaluated and used to make adjustments to that physical

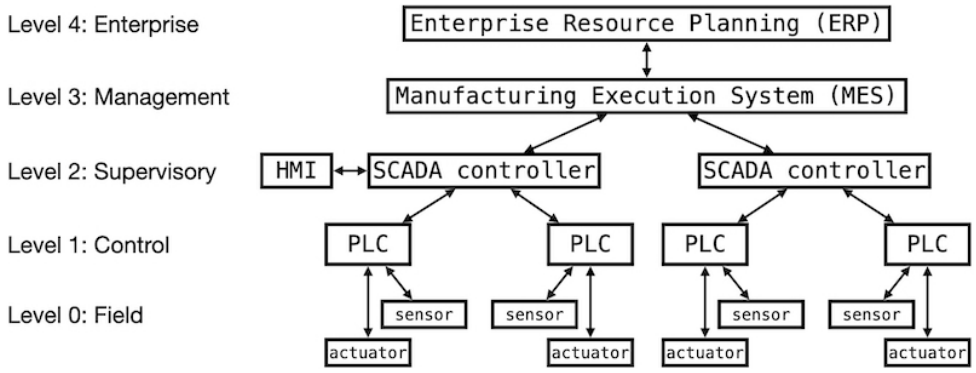

Figure 2.1: Simple example of a SCADA hierarchy.

process to ensure it behaves within some user-defined bounds. This can be continuous, discrete, or a combination of the two. For example, a motor controller senses the speed of the motor shaft and adjusts the AC frequency provided to the motor. This adjustment increases or decreases the shaft frequency to match a user-defined setpoint. Similarly, a heating controller senses temperature and turns the heating unit on or off to maintain temperature at a user-defined setpoint. The first example senses and actuates continuously, while the second may sense continuously but actuate discretely. Often such devices have additional, safety-related logic to support their main task. For example, a heating system may shut down if the input voltage drops below a safety setpoint because as the voltage decreases, the system may increase its current draw to maintain constant power, and this could lead to a fire. Devices directly in control of a physical process are often simple, logic-based Programmable Logic Controllers (PLCs). These devices are under the supervision of controllers with a more complex or expressive language for controlling multiple end units and interfacing with higher-level applications. Collections of such PLCs and supervisory controllers, along with peripheral devices such as the Human Machine Interface, are often referred to as Supervisory Control and Data Acquisition (SCADA) systems. Figure 2.1 shows a sample hierarchy of devices in a simple SCADA system.

ICS present unique design and management challenges compared to standard Information Technology (IT) systems: ICS devices are required to meet real-time guarantees and severe testing regimes before being placed in service, generally prioritise availability of a system over confidentiality, have lifetimes measured in decades, and are often distributed in remote locations [128]. These constraints lead vendors to be conservative, using stable technologies and maximising backward compatibility. Further, as ICS are often systems-of-systems, integrated testing is challenging: full test beds rarely exist, and testing on site may present both commercial and safety risks. These challenges create inertia against change, including software updates, especially those that may affect a certified system.

2.1.1 ICS security

Though there are several possible classifications of attacks against ICS, they can be helpfully organised under the standard categories of Confidentiality, Integrity, and Availability (CIA). While ICS face many of the same CIA threats as IT systems (e.g., eavesdropping, replay, and Denial of Service (DoS)), attacks on ICS can have catastrophic physical consequences, including threats to safety and life. For example, the Maroochy sewage incident in 2000 involved a disgruntled employee with a maintenance laptop who went to remote water

treatment sites and caused over one million litres of raw sewage to be released into local waterways [191]. More recently, the 2015 BlackEnergy 3 attack against the Ukrainian power grid caused extended power outages by first manipulating the electrical grid and disconnecting customers, and then destroying the ability to remotely restore the grid. One year later, the more sophisticated CRASHOVERRIDE/Industroyer attack again caused power outages in Ukraine [38].

In addition to unique consequences, control systems are open to unique classes of attack because they depend on physical sensor measurements for state awareness. For example, False Data Injection (FDI), first introduced via control theory research, is an integrity attack on a system's sensor measurements that can manipulate the system state without detection by algorithms designed to identify bad sensor measurements [32]. Such an attack may be possible in systems where the state cannot be directly measured, but is inferred from sensor measurements. For example, the pitot probe on the nose of an aeroplane measures static and dynamic pressure, and the relation between these two pressures is used to infer airspeed. Similarly, the voltage magnitude and phase angle at a given bus in a power grid may be inferred based on real power measurements at various points in the grid. In both cases, the inferred state, not the direct sensor measurement, is used for subsequent control decisions (e.g., to accelerate, to bring another generator online). With sufficient knowledge of how that inference occurs, it may be possible to manipulate sensors and actuators to cause arbitrary and targeted changes in system *behaviour* without causing detectable changes to observed system *state*, in which case, the manipulation will remain undetected. These covert attacks were first introduced by Liu *et al.* [121] and Bobba *et al.* [21], and they remain the dominant topic in theoretical control system security research as well as the related work on mitigation through anomaly detection [77, 208].

ICS security became a 'hot' topic following public identification of the Stuxnet worm in 2010 [31, 101, 109], which was a state-sponsored attack on the Iranian Uranium enrichment activities. While there had been previous attacks on and disruptions of digital national infrastructure [3, 28], Stuxnet surpassed them all in the breadth of knowledge and access required for successful execution. Subsequent attacks on national infrastructure, particularly against Ukraine's power generation and distribution infrastructure in 2015 and 2016 [38, 47, 117], continue to highlight risks associated with digital ICS, with special emphasis on the vulnerability of critical infrastructure. Governments and industry have responded with guidance and regulation, such as the National Institute of Standards and Technology (NIST) special publication 800-82 'Guide to Industrial Control Systems (ICS) security' [197], a direct descendent of [64], and IEC 62443 'Industrial Automation and Control Systems Security' [95].

Simultaneously, researchers have highlighted the growing risk of Internet-connected ICS devices, of which tens of thousands expose remote monitoring and control services without any authentication or access control mechanisms [115, 136]. While many of these devices may be exposed inadvertently, due to lack of knowledge on the part of the user or poor design on the part of the vendor, there is a growing demand to connect *everything* to the Internet in industrial environments as part of Industry 4.0 [214]. Industry 4.0 relies on a fully-connected ecosystem of Industrial Internet-of-Things (IIoT), a blend of traditional ICS and commodity IoT devices, abandoning the standard communication hierarchy of most industrial environments and replacing it with an industrial cloud, which allows individual ICS/IIoT devices, including controllers, sensors, and actuators, to be deployed, connected, and scheduled to maximise production efficiency, without the constraints of

fixed hierarchies such as that in Figure 2.1 [120, 122].

As mentioned in Chapter 1, many industrial security solutions focus on securing the network boundary, as this minimises perturbations to existing devices and systems. Much academic security research similarly attempts to support existing integrations, but seeks to avoid this ICS security paradigm of a "hard, outer shell" with a "soft, gooey center" [106], correctly recognising that it inadequately protects ICS networks, cannot prevent FDI or other integrity attacks, and provides no protection against protocol attacks [106]. For example, researchers are pursuing behaviour-based anomaly detection: inspecting network traffic and comparing it against expected traffic produced by a physics-based model of system behaviour [77, 208]. Related work adds strategic and correlated sensors to a system to improve the speed and accuracy of anomaly detection at the cost of additional complexity in design, operation, and maintainability [79]. Much of this work is rooted in the same control theory that first identified stealthy control system attacks [200, 201].

While these solutions make stronger assumptions than their industrial counterparts (e.g., the adversary breaches the boundary), they are often as complex as the systems they are designed to protect and add to existing integration and support challenges. To avoid the complexity and uncertainty of probabilistic detection methods, researchers are beginning to investigate provable security guarantees for ICS. Giraldo *et al.* propose designing systems such that the physical limitations of a system's actuators make a generic adversary's goals unachievable [78]. They perform two case studies, one involving simultaneously filling and draining a set of tanks (approximating an industrial water filtration system) and another involving four platooning vehicles. In both cases, they assume the adversary can control actuation (e.g., pump speed or vehicle acceleration), and in the tank example they assume the adversary can control sensing (e.g., tank level). They identify the entire state space reachable by the adversary, given the physical limitations of the system, and then constrain system actuation until the reachable state space is provably disjoint from the set of undesirable states (e.g., overflowing tank or vehicle collision). The authors admit, however, that their current method may constrain actuation to the point that a system can no longer achieve its objective. For example, constraints on acceleration/deceleration may conflict with requirements ensuring the lead vehicle can stop quickly enough to avoid collisions with obstacles. Similarly, constraints on pump speed may conflict with requirements to rapidly fill a tank in the event of a leak to prevent the filtration media from drying out. It is also unclear how such methods will scale to more complex systems and whether it will always be possible to constrain the system to ensure the set of reachable states is disjoint from the set of undesirable states.

2.1.2 Internet-wide scanning

A growing number of ICS are connectable; that is, they are designed for IP-based network connectivity and often use a locally hosted webserver for Human-Machine Interaction (HMI). Partly as a result of this connectability, the number of Internet-connected ICS devices is growing. Understanding the use and behaviour of these devices is necessary to understand the security implications of this Internet-connected population, and Internet-wide scanning is a tool to help gain that understanding.

Internet-wide scanning of IPv4 addresses is a common tool leveraged by both malicious and non-malicious users. On one hand, it is a foundational element of botnet propagation. For example, every host infected by the Mirai botnet performs its own stateless scanning

of the Internet to find and infect other, vulnerable devices [8]. On the other hand, it is used by network defenders for asset discovery and network mapping and by security researchers to identify otherwise-opaque distributed systems [60, 123].

Network scanning relies on rapid use of existing protocols to identify and interact with hosts across a range of IP addresses. NMap was one of the first widely available, open source scanners, and remains both popular and powerful. Amongst other things, it uses stateful scanning to identify hosts and open ports, fingerprints operating systems, and finds hosts with known vulnerabilities [123, 150]. NMap was originally designed for probing individual hosts and mapping local networks, rather than the whole Internet, and its stateful mechanisms make it an impractical tool for scanning large blocks of IP addresses. By comparison, ZMap, another open-source tool, abandons stateful scanning, generates Ethernet frames directly to maximise use of the client's compute and network resources, and randomly orders target IP addresses to avoid overloading target networks. As a result, ZMap can perform host and port discovery across a block of IP addresses over 1 300 times faster than NMap [60, 204]. Because ZMap's stateless nature precludes use beyond host and port discovery, ZGrab builds on ZMap with protocol-specific, stateful scanning to request more information about a host [136]. For example, the HTTP module will obtain HTTP headers from a webserver, and ICS modules will obtain device-specific data such as that shown in Tables 3.1 through 3.4. While stateful, ZGrab's underlying use of ZMap to identify hosts open on a given port before attempting stateful interaction makes it considerably faster than NMap over large blocks of IP addresses.

Standard search engines, such as Google, crawl and index content that can be consumed by a web browser, such as web and FTP servers, which operate on a limited number of ports (e.g., HTTP/80, HTTPS/443, FTP/21). The ability to rapidly scan and index hundreds or thousands of ports and protocols opened the door for new search engines that consolidated port and protocol information for hosts across the Internet. For example, Shodan and Censys discover webcams and other IoT devices, routers, cloud storage buckets, and building automation devices [30, 186]. These search engines allow users to bypass implementing their own scanning infrastructure and are used by cybercriminals, voyeurs (looking for open webcams is by far the top search on Shodan), network defenders, security researchers, and government analysts [56, 115, 163]. That said, Internet-wide scanning is limited to hosts that are externally addressable via an IPv4 address. While targeted ranges of the IPv6 space can be scanned, it is not feasible to scan the entire IPv6 space due to its size: there are 2^{128} possible IPv6 addresses compared to 2^{32} possible IPv4 addresses, which require about 45 minutes to scan with ZMap [60] on a single port. Similarly, devices behind a firewall or NAT may not be externally discoverable via scanning, though the existence of such hosts can be inferred by monitoring network traffic [17].

In Chapter 3 we use Shodan and Censys to perform a longitudinal study of Internet-connected ICS devices, leveraging historical data from these search engines to monitor changes to the population size and stability as well as patch cadence. We also use ZGrab to scan ICS protocol ports on IP addresses suspected of hosting Mirai-infected devices to identify networks hosting both ICS devices and malware-infected IoT devices.

2.1.3 Honeypots

Honeypots are computer security systems that emulate production systems and decoy attackers away from the production system, provide warning of an intrusion, or allow

attacker behaviour to be studied [126, 211]. Honeypots have been designed to emulate individual computers, such as laptops; servers; IoT and ICS devices [169, 211]; and larger systems, such as electrical substations [174]. As a security device, they can be used as part of a defence-in-depth strategy alongside segmented networks and firewalls. As a research tool, they are often used as standalone devices directly connected to the Internet.

Honeypots can be characterised by their purpose and level of interaction [211]. The purpose of interaction refers to whether the honeypot is part of a production system, designed as part of a security solution for a given network or device, or a research device, designed to attract attackers and study their behaviour [193]. The level of interaction refers to how well the honeypot emulates the target device, which determines how easy it is for the attacker to identify that they are interacting with a honeypot. The level of interaction is generally categorised as low, medium, or high, though these categories are not well defined. A low-interaction honeypot may be a simple script that only emulates a login screen but no stateful device behaviour. A high-interaction honeypot may be an actual device or system, not an emulation, which is instrumented to record details of attacker behaviour on the system [65, 161].

Because honeypots have no purpose on a network except to deceive potential attackers, any interaction with such a honeypot by an attacker demonstrates either that the attacker lacks knowledge or is indiscriminate. If an attacker has sufficient knowledge and a specific target, then they can interact directly with the target device on a network and leave any honeypots untouched. If the attacker has less knowledge, but still has a specific target, they may have to scan a network to find the target device. In this case, they will interact with the honeypot, which will notify the defender of the attacker's presence, even if the attacker is able to avoid further interaction with the honeypot. In a less discriminate scenario, where the attacker is looking for any vulnerable device, they may go further and continue to interact with the honeypot, attempting to exploit vulnerabilities. Therefore, Internet-connected, research honeypots have been effectively used to detect and monitor large-scale, indiscriminate attacks [211], but not knowledgeable, targeted attacks [29, 66].

Within the ICS community, there are several, open-source honeypots available. Conpot is a low-interaction honeypot capable of responding accurately to network scans [169]. It is easy to set up and scales well, making it a good candidate to research Internet-wide scanning [29, 66, 136]; however, its inability to interact with an attacker limits its utility in detecting and characterising ICS attacks, and studies using Conpot have yet to identify any new or targeted ICS attacks [29, 66, 136]. MiniCPS is a framework for higher-interaction honeypots and runs actual programmable logic [9]; however, it has yet to be used in a study to detect previously unknown ICS attacks and its hardware emulation may be detectable by a capable attacker [211]. In Chapter 4 we use a network of 120 high-interaction ICS honeypots to attempt to identify both targeted and large-scale manipulation of Internet-connected ICS devices.

2.2 Memory management

One of the challenges faced by CPS, particularly those devices with limited compute and memory resources, is limited memory protection. Specifically, devices may not have memory management hardware and/or may not trust the kernel to mediate memory sharing between mutually-suspicious processes. We discuss these challenges in more detail in Chapter 5. Here we provide the background necessary to support that discussion. We

first describe the history of memory segmentation and paging, leading to the ubiquitous use of virtual memory in general-purpose computing. We then compare the memory management hardware available in general-purpose and embedded computing, particularly focusing on memory protection. In the next section, we introduce capabilities, and focus on the use of CHERI architectural capabilities as the basis for alternate memory protection and compartmentalisation models for both general-purpose and embedded computing.

2.2.1 Memory segmentation and paging

The advent of multiprogramming systems, where concurrently running programs shared main memory and processing resources, created a need for abstracting the physical resources away from the programs, which in turn created a need for a supervisor with knowledge of the physical resources to manage their multiplexing. For processing resources, the supervisor manages scheduling. For the memory resources, the supervisor implements an indirection that evolved into virtual memory, which we discuss further below.

Multiprogramming systems required a mechanism for loading objects from concurrently running programs into main memory and ensuring those programs could not access one another's objects unless the object was explicitly shared. The initial and natural semantics for this was to associate an arbitrarily-sized memory *segment* with an object, load the entire segment into memory, and then refer to it using a base address and an index into the segment. The supervisor was responsible for managing segment loading and access, including evicting a segment from main memory when it was no longer in use or when the memory was needed to support a segment for another program [52]. Because the physical memory layout was managed by the supervisor and abstracted away from the program, a layer of indirection was required: the program specified a *virtual address*, which consisted of an object identifier and index into the object, and the supervisor maintained *segment tables*, which were used to translate the object identifier into a physical base address. Having identified the base address, the index provided by the program could be used directly as an offset from the base address to find the desired word of memory. This translation effectively virtualised the memory, as each concurrently running program could behave as if it owned the entire memory space.

The arbitrary size of memory segments made inefficient use of memory and caused external fragmentation. External fragmentation occurs over time when objects of non-uniform size are loaded into and evicted from main memory, resulting in non-uniform ranges of contiguous, free memory between objects. Eventually, though the total free memory may be sufficient to load an object, there may not remain a sufficiently large contiguous range of free memory. In this case, the supervisor can either defragment the memory or evict an existing object. Defragmenting requires the supervisor to suspend the program, move existing objects to adjacent memory regions and consolidate free memory, and then resume the program and the loading operation. Evicting requires the supervisor to select an object to be overwritten based on a specified algorithm (e.g., least recently used) and may require suspending the program while the object is written back to secondary memory. If the evicted object is still in use by another program, the supervisor may need to reload the object when the other program is resumed (possibly requiring another eviction or defragmenting event). This back and forth of loading and evicting is called thrashing, and the performance penalty can cripple a system [51].

Memory *paging*, in contrast, always loads constant size blocks of contiguous ad-

dresses (pages) from secondary memory into main memory. Objects smaller than a page still require use of a full page of main memory. Objects larger than a page are divided into multiple pages and loaded into free pages of main memory, which are not required to be contiguous. Instead of segment tables, the supervisor maintains *page tables* to support translation from a virtual to a physical address.

Paging solves the external fragmentation problem, because unallocated memory will always be a multiple of the page size, and the pages representing an object do not need to be contiguous in main memory. It also simplifies allocation because the memory manager does not need to identify a contiguous free block to fit a full segment, but rather it only has to ensure a sufficient number of free pages exist to hold the entire object. While it eliminates the inefficiencies of external fragmentation, paging introduces internal fragmentation, which occurs when an object uses less than a full page of memory; the remaining memory in the page cannot be used by any other object. This creates a tension in the design of a memory paging system. Smaller pages reduce the memory wasted by internal fragmentation, but they increase the required number of page table entries, which themselves must be stored in memory. Conversely, larger pages require smaller page tables, but increase internal fragmentation. This tension becomes more acute as the size of main memory increases. Modern systems overcome this tension in two ways. First, many systems support two or more page sizes simultaneously (e.g., 4 KiB and 2 MiB). Second, many systems support multi-level page tables, increasing the number of indirections required to translate a virtual to a physical address, but limiting the size of individual page tables. Specifically, multi-level page tables are designed such that each page table consumes a single page of memory, avoiding the need to store a single, large, sparse page table in memory [51, 156].

Early multiprogramming systems with virtual memory (e.g., MULTICS) relied on both the segment and paging mechanisms, allowing the programmer to think intuitively in terms of segments and the supervisor to manage memory efficiently with pages: objects were represented as a contiguous set of virtual addresses in a segment table, the index in the segment table redirected to a page table, and the index in the page table contained the physical memory address of the desired word [37, 91]. As described above, modern virtual memory systems have inherited the use of multiple levels of page table indirection to efficiently manage large memory spaces, but no longer support the semantics of memory segments. Memory segments survive, however, in binary file formats, such as ELF [156].

2.2.2 MMUs and MPUs

As virtual memory and paging became ubiquitous in multiprocessing systems, virtual memory translation and permissions checks have increasingly been offloaded to dedicated logic blocks, called MMUs, which cache virtual to physical address translations in Translation Lookaside Buffers (TLBs). MMUs also generally 'walk' the page table indirections for TLB misses, though some systems still perform this translation in software [96]. While not strictly necessary, MMUs make virtual memory systems practical, especially in systems with multi-level page tables, requiring multiple indirections to translate virtual to physical memory. MMUs also ensure a given process has the correct permissions to access a given page of memory, effectively providing coarse-grained memory protection and faulting when a process attempts an illegal operation.

While virtual memory and MMUs are considered necessary in general-purpose comput-

ing, they have not historically been implemented in embedded devices, including many CPS devices. Baremetal or Real-Time Operating System (RTOS) applications were not historically considered multiprogramming environments (i.e., they did not have multiple programs competing for compute and memory resources); therefore, the added complexity and overhead of virtual memory and MMUs was unnecessary. However, this paradigm is no longer representative: embedded systems increasingly require resource mediation between multiple tasks or processes, and even devices with only one application often integrate a kernel and third-party libraries, which may not be equally trustworthy. To avoid the cost and complexity of MMUs, some microcontrollers implement MPUs, which allow low-level software (e.g., bootloader or privileged kernel) to statically divide the physical memory into regions with different permissions. The number of regions is architecture-dependent, but is generally between 8 and 16. Like an MMU, the MPU performs a permission check before allowing a memory operation, ensuring the process or task has the required privilege to perform the specified operation on the given region of memory. The MPU allows the kernel to protect its own memory from applications, allows applications to be protected from one another, and can be used to protect applications from untrusted libraries [119].

2.3 Capabilities

A *capability* demonstrates that a *subject* has *authority* to *invoke* a *resource* [135, 175]. That is, when a subject presents a capability, the capability proves the subject has the specified access rights for the named resource [175]. Capabilities are the building blocks for a minimally privileged system and inherently preclude the confused deputy problem [135].

The confused deputy problem originates in a system that separates the designation of an object from the authority to operate on that object. In the canonical example, a compiler requires supervisory privilege to write to a log file (e.g., for billing purposes). A user with insufficient privilege to access the log file directly invokes the compiler, but provides the log file as the output file. The compiler, using its supervisory privilege, then overwrites the log as requested. While input sanitisation may mitigate this vulnerability, nothing inherent in an Access Control List (ACL)-based, ringed protection model precludes such confusion. This confusion is impossible in a capability system because designation and authority are inextricably linked; therefore, the user can only designate objects (such as the output file) if it also holds the authority to invoke that object [83]. Chapter 5 elaborates on the particular relevance of the confused deputy in CPS.

In the following sections, we provide a brief introduction to CHERI architectural capabilities, a discussion of the availability of CHERI-aware hardware, and an overview of CHERI-based compartmentalisation as an alternative to MMUs or MPUs.

2.3.1 CHERI architectural capabilities

CHERI is a hybrid capability architecture, enabling fine-grained memory protection and scalable software compartmentalisation via hardware-enforced, bounded pointers.

A CHERI capability designates and authorises invocation of a memory object, bounding a designated memory region and providing specific authorities for accessing that memory (e.g., load, store, execute). CHERI relies on tagged memory and CHERI-aware instructions to provide hardware-enforcement for unforgeability, provenance, and monotonicity [215].

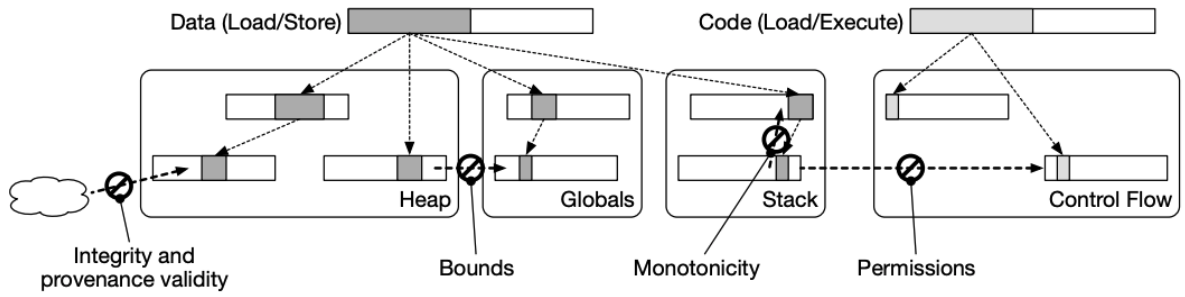

Figure 2.2: CHERI capabilities provide hardware-enforced unforgeability, provenance, monotonicity, and permissions checking for memory pointers [215]. Figure used with permission.

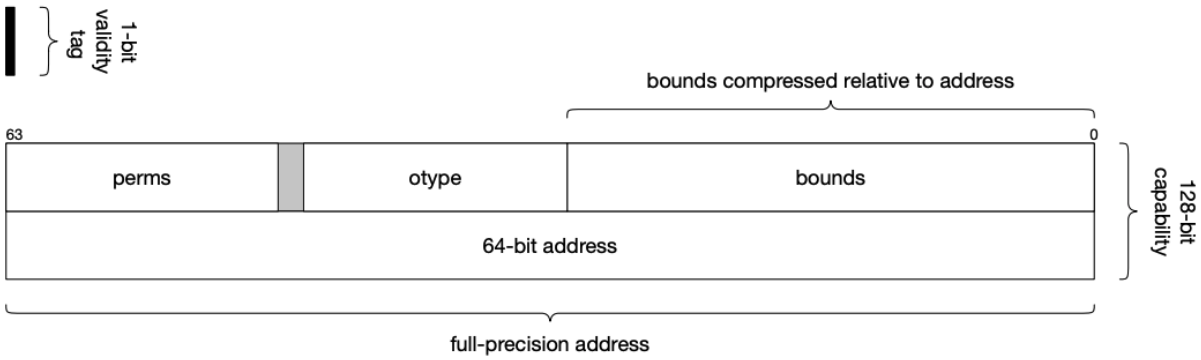

Figure 2.3: 128-bit CHERI capability with the memory address, bounds, metadata, and out-of-band validity bit for a 64-bit memory address [215]. Figure used with permission.

Enforcing provenance ensures a capability can only be constructed via other valid capabilities, while enforcing monotonicity ensures that the bounds and permissions of a new capability cannot exceed those of the capability from which it was derived. Figure 2.2 visualises how these properties are enforced in a C/C++ runtime, and Figure 2.3 provides an example of a CHERI capability in memory.

In Chapter 6 we discuss the use of CHERI in CPS to provide spatial memory safety and compartmentalisation without MMU or kernel support.

2.3.2 CHERI availability

The obvious limitation of a CHERI-dependent architecture is the lack of production processors implementing CHERI Instruction Set Architecture (ISA) extensions.

ISA extensions currently exist for MIPS, RISC-V, and Armv8-A, and numerous Field Programmable Gate Array (FPGA) cores are available for MIPS and RISC-V to support architectural capability research.

Production hardware is on the horizon, however. Arm Morello is a prototype, superscalar, CHERI-aware System-on-Chip (SoC), which will ship in 2022 [118] as part of the Digital Security by Design challenge [54]. Further, capability extensions were discussed as part of the Armv9 roadmap during Arm's 2021 Vision Day announcement of the Armv9 architecture [10]. Capability extensions are also being considered for production M-profile cores [146]. Overall, there appears to be a good chance that CHERI (or similar) architectural capabilities will be available in both high-performance CPU and microcontroller production hardware by the end of the decade.

2.3.3 CHERI-based compartmentalisation

As discussed above, CPUs with MMUs and operating systems that support virtualised memory provide coarse-grained memory protection via memory pages: a fixed-size, contiguous block of physical memory addresses mapped into a process' virtual memory space. The operating system mediates a process' access to these pages and controls what permissions a process has over that memory (e.g., read, write). Smaller CPUs and microcontrollers targeting embedded or real-time applications may have a MPU instead of a MMU. These devices do not support memory virtualisation, but the MPU provides statically-configurable memory partitions and allows kernel space to be separated from user space, for example, but their limited number, static constraints, and performance costs historically precluded practical, software-defined compartmentalisation [219]. Recent research efforts have improved MPU usability, however, helping developers map their applications to separate memory spaces [34], or implementing reference monitors in a separate memory space from the kernel, providing strong security guarantees without burdening the CPS developer with compartment management [103].

CHERI capabilities provide opportunities for multiple compartmentalisation models that are more fine-grained than the memory protection provided by MMUs and more intuitive and dynamic than those provided by MPUs. These models are active research topics. The former was first explored in the context of a CHERI-aware version of FreeBSD, relying on the dynamic linker and CHERI capability properties to control the flow of execution and isolate one process' code and data segments from other processes [167]. The latter was first explored in the context of a CHERI-aware RTOS, which compartmentalised tasks in an otherwise flat, physical memory space [219].

In Chapter 6 we leverage early work on a new compartmentalisation model applied to a CHERI-aware version of FreeRTOS, and we discuss related work on compartmentalisation and control flow in that context.

2.4 Network authentication and authorisation

In addition to memory management challenges, networked CPS also face trust challenges. Specifically, many CPS integrate multiple devices from different suppliers performing distinct tasks to accomplish a common goal, resulting in an environment of mutual dependence without mutual trust. As discussed in Section 2.1.1, boundary defence mechanisms partially mitigate this by limiting the number of devices that must trust one another, and anomaly detection devices attempt to mitigate this further by identifying suspicious network traffic or device behaviour. However, both of these mitigations outsource privilege and can be complex to design, maintain, and operate. This dissertation attempts to minimise privilege and restore access control to resource-owning devices, which requires mechanisms to establish and transmit identity and authority across a network. Here we provide an overview of these problems and common mechanisms used to authenticate and authorise devices on a network.

2.4.1 Network authentication

In terms of network security, authentication means "verifying the identity of the communicating principals to one another" [144]. There are many cryptographic mechanisms, both

symmetric and asymmetric, for protecting the confidentiality and integrity of data across a network; however, the validity of such protections depends on proper authentication of principals, which depends on key establishment and distribution. If this challenging problem is solved, then continued use of the keys (or ephemeral session keys derived from those initial keys) is "keeping authentication alive" [210].

The naive mechanism for key distribution is manual provisioning: manually storing key material at a given principal for subsequent use. For very small networks, this method is intuitive and easy to manage. For example, when a user is using secure shell (SSH) to remotely log into one host from another, and particularly when both hosts are under the user's own control, it is common for that user to manually store the public key from one host in the `authorized_keys` file of the other host. While manual provisioning scales poorly and is not suitable for dynamic environments where hosts may be added to or removed from the network regularly, it may be sufficient for small, static networks [6].

An automated derivative of manual provisioning is Trust-On-First-Use (TOFU), where one principal simply 'imprints' on the first key it receives from another principal. The key could be transferred over the same channel used for subsequent communication, or via a side channel, such as physically mating the two devices before installation in different parts the network. In the former case, TOFU assumes the network adversary is not yet present during key provisioning to either passively snoop the key or actively provide its own. Both cases require a mechanism to reset the device and allow a new key to be provisioned (e.g., if the key is compromised or the device imprints on the wrong key) [6, 194]. Like manual provisioning, however, TOFU does not work in a dynamic environment because it assumes key provisioning happens before an adversary is present, which does not hold in an environment where users are being added or removed over time.

Use of a trusted third party simplifies the problem of key management in a dynamic environment. For example, the Kerberos protocol uses a trusted third party (in the form of one of more authentication and ticket granting servers) to allow two principals to mutually establish identities and derive ephemeral session keys without sharing secrets with one another during provisioning. It does require both participants to have shared a secret (a long-term key) with the trusted third party, but this is more scalable and requires trusting fewer principals than a system requiring all participants to share secrets with one another during provisioning or whenever a new principal is added [6, 144]. Kerberos supports the separation of authentication from key establishment and distribution by making use of cryptographic tokens. To briefly summarise the protocol, principal A first establishes its identity (via password or asymmetric key) with the authentication server and is granted a cryptographic token, called a Ticket Granting Ticket (TGT). The TGT is encrypted under A's long-term key and demonstrates A's successful authentication. Then principal A sends a request to the ticket granting server for access to principal B's services, and provides the TGT as proof of its identity. The ticket granting server creates an ephemeral session key and provides this to A along with another cryptographic token, called a ticket, containing A's identity and the session key, encrypted under B's long-term key. Principal A forwards the ticket to B, and the protocol continues by establishing freshness and then moving on to communication under the session key. Thus, A and B have mutually authenticated and obtained a session key for subsequent communication without previously sharing a secret between them. Kerberos scales well, but a compromise of the trusted third party can be catastrophic, as the adversary would hold the session keys used to establish any subsequent communication between principals.

Public Key Infrastructures (PKIs) are another example of systems using trusted third parties to establish identity before generating session keys for subsequent communication. For client/server interactions, such as initiating a Transport Layer Security (TLS) session, the client reaches out to the server, and the server responds with its public key certificate, which binds the server's identity to its public key. To verify the identity of the server and the validity of the certificate, the client works backward through one or more certificates that signed the server's certificate until it reaches the root certificate issued by a Certificate Authority (CA). Like the secret shared between a principal and a Kerberos server, the client must have received this root certificate during provisioning through some trusted channel. For example, web browsers embed the root certificates for common Internet CAs. Having verified the certificate and associated public key of the server, the client then uses the public key to encrypt material that will be used to generate an ephemeral session key with the server [6]. The client will generally authenticate with the server through some other mechanism, such as a password, over the now-encrypted channel.

While PKIs scale well (even to Internet scale), they are also subject to abuse at scale. Specifically, since the purpose of a public key certificate is to tightly bind a verified identity to a public key, any malicious effort to actually compromise or call into question the integrity of that binding can affect millions of clients and servers. This abuse can happen at either end of the chain of trust. For example, at the CA end, compromise of the DigiNotar CA, identified in 2011, allowed an intelligence agency to monitor the Gmail accounts of at least 300,000 people. At the client end, intelligence agencies have asked or compelled browser vendors to include government-owned CA root certificates in the browser, allowing them to issue valid certificates for domains they do not own, supporting Man-in-the-Middle attacks on TLS [6]. Further, the complexity of managing PKIs is known to motivate shortcuts and risky behaviours that introduce vulnerabilities, such as CAs issuing wildcard certificates [26], or users becoming accustomed to clicking through browser certificate warnings [195]. Efforts such as certificate transparency provide some mitigation for concerns associated with the chain of trust (at the cost of added PKI complexity), but cannot directly address management shortcuts or poor user behaviour [111, 195].

2.4.2 Network authorisation

These four mechanisms for establishing identity (manual provisioning, TOFU, Kerberos, and PKIs) are examples of using cryptography for authentication on a network. They do not establish authorisation, or the privileges a principal has to access resources owned by another device on the network. Rather, having authenticated through the use of a cryptographic key or token, access control to resources is generally managed locally, through a mechanism such as an ACL. That is, if A remotely connects to B and establishes its identity via a cryptographic key or token, B manages subsequent requests for access to B's resources as if A is a local user.

This paradigm presents two challenges in a distributed system. The first is that the system requires a mechanism for propagating the names of principals and resources amongst the interacting principals. For the simple example above, B must either already know of A's existence and associate A's identity with specific resources and authority, or else B must consider A part of a larger group, inheriting that group's authority over specific resources. Similarly, A must be aware of the resources available and the authority each principal is willing to give A over its resources [135]. The second challenge is delegation. Because of

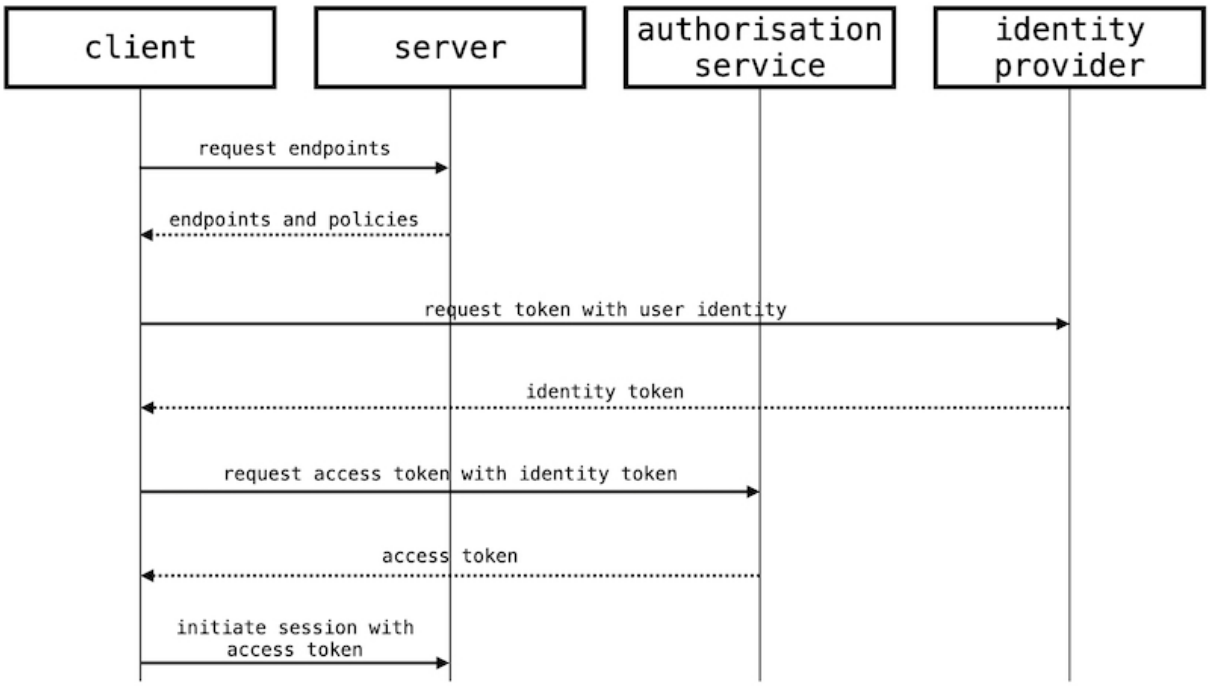

Figure 2.4: OPC UA client obtaining and using an access token [185].

the naming dependency, delegating access to a resource either requires propagating a new name (e.g., A asking B to give C access to a subset of resources to which A has access) or acting as a proxy (e.g., A accessing B's resources on behalf of C). Propagating names is error prone and expensive, and proxied requests lack transparency and risk introducing a confused deputy (see Section 2.3).

Cryptographic tokens conveying authority attempt to solve these problems by removing the need to associate principal names with resources and permissions at the resource-owning host. Instead, a resource user presents a token designating a resource and a set of privileges to the resource owner, and the owner verifies the integrity and provenance of the token and grants or denies access to the resource. For example, JSON Web Tokens (JWTs) are JSON objects that convey a set of 'claims' about a named resource between two parties, where a claim is usually a privilege for that named resource. The JWT is cryptographically protected by a digital signature or Message Authentication Code (MAC) [97]. The expected use case is a client authenticating with an authorisation server, receiving a JWT encapsulating claims for specific resources, and presenting that JWT to another server as part of a request for access to that server's resources. Figure 2.4 shows how this is used within Open Platform Communications (OPC) Universal Architecture (UA), an interoperability standard for industrial automation. The 'Access Token' in the figure is a JWT. OPC UA is discussed in more detail in Chapter 6.

While JWTs and similar tokens avoid the problem of propagating names and support simple delegation, they have no mechanism for attenuation. For example, if a client holds a JWT granting read and write access to a resource and wants to delegate read access to another client, it must obtain a new JWT from the authorisation server. In contrast, Simple Public Key Infrastructure (SPKI)/Simple Distributed Security Infrastructure (SDSI) is an effort to to decentralise authorisation, delegation, and attenuation. Though a key rather than a token scheme, SPKI/SDSI not only allows authorisations to be delegated from one key to another, but it supports transferring only a subset of authorisations, effectively

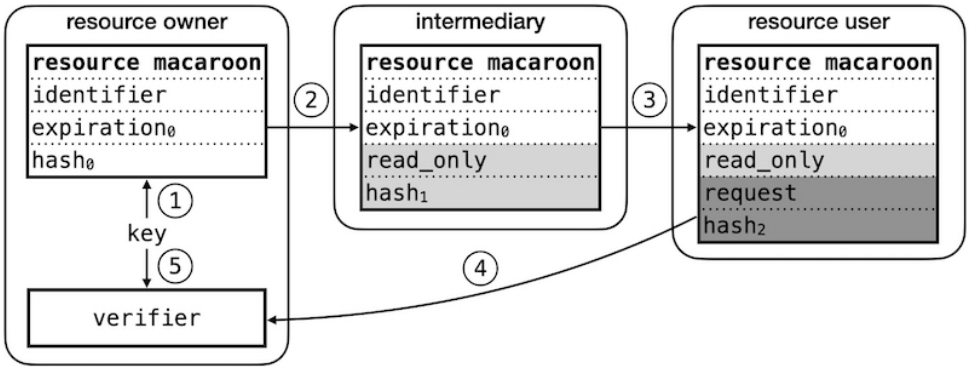

Figure 2.5: Macaroon delegation example. The resource owner generates a secret key and initial Macaroon (1), and attenuates and provides it to an intermediary (2). The intermediary applies additional caveats and provides it to the resource user (3). The user applies a request as a caveat and returns the Macaroon to the owner (4) for verification (5).

attenuating the delegated authorisations [62].

Macaroons are cryptographic tokens, like JWTs, but have the delegation and attenuation power of SPKI/SDKI. The distributed security architecture we implement in Chapter 6 makes use of Macaroons, so we provide an extended overview here.

Macaroons are bearer credentials designed for decentralised, network-based access control requiring efficient delegation and attenuation of authority. Macaroons are constructed using a chain of keyed Hash-based Message Authentication Codes (HMACs) to protect provenance and integrity. The initial Macaroon is generated by the resource owner and consists of a plaintext identifier (designating a resource) and a hash signature generated with a secret key. The key remains with the resource owner; only the Macaroon is distributed to users. Privilege is attenuated by concatenating plaintext 'caveats' and extending the HMAC chain. For each caveat, the signature of the existing Macaroon is removed and used as the key input to the HMAC of the attenuated Macaroon. This allows any Macaroon holder to add a caveat, but prevents anyone without the original secret key from removing caveats. A caveat can attenuate authority (e.g., read-only), restrict context (e.g., expiration), or link to a third party (e.g., an authentication service). A resource user presents a Macaroon as part of a request to invoke a resource, and the resource owner verifies the provenance and integrity of the Macaroon using the original secret key to recompute the HMAC chain. Having verified the HMAC chain, the resource owner checks that each caveat is met and grants or denies the invocation [20]. A basic example of this exchange is shown in Figure 2.5.

As stated above, a caveat can link to a third party, requiring the resource user to discharge some task (e.g., authentication) prior to presenting the resource Macaroon to the resource owner. A basic example of this exchange is shown in Figure 2.6. A resource user holding a resource Macaroon reads a third-party caveat and interacts with the third-party to satisfy the caveat's requirement. To affirm the user has discharged the task, the third party issues a *discharge Macaroon* to the user. When requesting access to the resource, the user presents the discharge Macaroon alongside the resource Macaroon to the resource owner. While verifying the resource Macaroon, the resource owner will come to the third-party caveat, recursively verify the discharge Macaroon, and then continue to verify the resource Macaroon. These third-party caveats allow authentication (or other tasks) to be decoupled and off-loaded from the resource owning system, while still giving the host

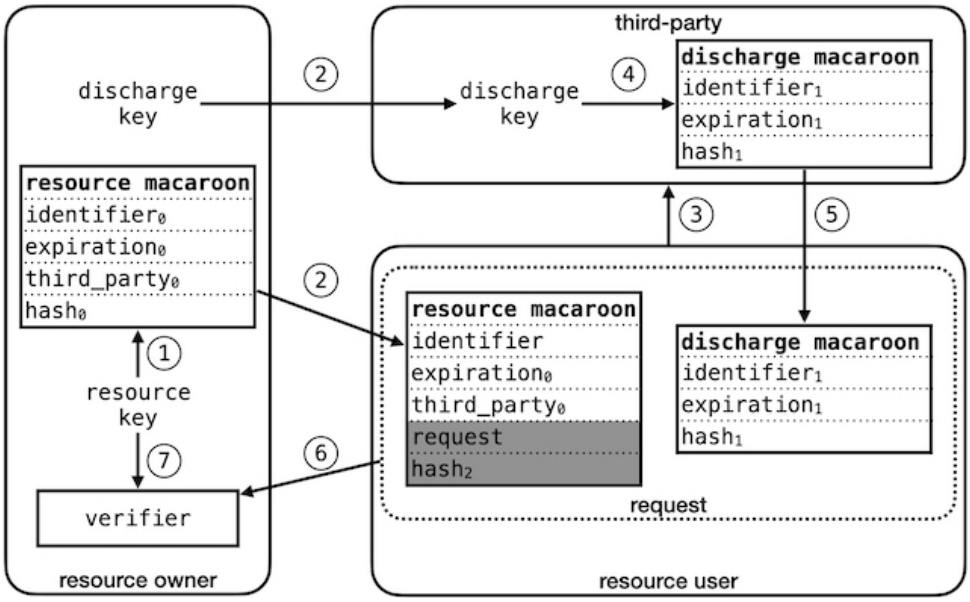

Figure 2.6: Example usage of a third-party Macaroon. The resource owner generates a resource key, used to create the initial resource Macaroon (1). The resource owner also creates a discharge key which is shared with the third party while the initial resource Macaroon is shared with the resource user (2). The resource user sends an authentication request to the third party (3). The third party uses the discharge key to generate a discharge Macaroon (4) and provides the discharge Macaroon to the resource user (5). The resource user applies the request as a caveat to the resource Macaroon and returns both Macaroons to the resource owner (6). The resource owner recursively verifies the resource and discharge Macaroons (7).

full control over access to its objects.

In Chapter 6 we use Macaroons because of their low cryptographic burden and ease of brownfield deployment; however, there are other network tokens that could be used to either replace or complement our use of Macaroons. For example, Kerberos can be used to complement Macaroons by providing a discharge Macaroon, rather than a ticket, when a user authenticates with the Kerberos server. Compared to SPKI/SDSI, Macaroons provide the same expressiveness with a lower cryptographic burden [20]; however, replacing Macaroons with SPKI/SDSI certificates would require little effort in a deployment of our end-to-end capability system.

Many CPS reference architectures already rely on network tokens for access control across the network. For example, Azure Sphere uses Shared Access Signature (SAS) tokens [131] and OPC UA uses JWTs [97], to establish authorised access [55, 129, 130, 181]. SAS tokens are part of Microsoft's Azure ecosystem and require a centralised Azure Active Directory instance. As stated above, JWTs can be decentralised, but lack the delegation properties of Macaroons and SPKI/SDSI certificates (e.g., one user holding a JWT cannot attenuate the authorisations of that JWT and pass it to another user). Taking these restrictions into account, either could be used to replace Macaroons in an implementation of our end-to-end capability system with little effort.

2.5 Summary

This chapter briefly summarised the background and the current state of research to support and contextualise the rest of the dissertation.

We provided an introduction to CPS and ICS, including a discussion of the state of ICS security. We also provided a primer on tools such as Internet-wide scanning and honeypots, which we use in Chapters 3 and 4 to understand the current state of ICS security and to motivate the development of a distributed security architecture for CPS.

We discussed the memory management and network trust challenges that such a distributed architecture presents in CPS, which are a subset of the challenges we present in Chapter 5. In both cases, we briefly described the history and evolution of mechanisms widely used in general-purpose computing and demonstrated why such mechanisms were either impractical or inadequate in the CPS domain. We then provided a longer summary of CHERI architectural capabilities and Macaroon cryptographic tokens, the technologies on which we base our distributed security architecture in Chapter 6.

chapter 3

The security economics of large-scale attacks against Internet-connected ICS devices

The vulnerability of Internet-connected Industrial Control Systems (ICS) is a well-studied problem. These devices present open industrial protocol ports to the Internet without any authentication or encryption [136], allowing an attacker to modify the device logic or directly manipulate device behaviour [149]. ICS devices also often expose ports and services to the Internet that are not specific to industry (e.g., webservers) [115, 163], and the slow patch cadence for this domain results in long-term risk from common Information Technology (IT) vulnerabilities (e.g., privilege escalation). Previous studies took snapshots of the Internet-connected ICS device population to identify the number and types of devices and infer risk [115, 136, 163]. These studies concluded that the ICS "space is in disarray" [136], that the possibility of people with "criminal intent or a political agenda... act[ing] against these devices is significant" [163], and that malicious actors "might already be doing this" [115]; however, the predicted consequences have not materialised. We reassess this risk with an end-to-end assessment of the Internet-connected ICS landscape: we track the device population and device owner behaviour over a four-year period and perform a technical and criminological assessment of the adversarial community to understand why an apparently-vulnerable population is largely ignored.

The Internet-connected ICS device population shares many characteristics with other populations that *have* attracted large-scale criminal activity, such as size, stability, and lack of long-term support. The population is approaching 100 000 devices, of which we tracked approximately 10% over four years. We find that over 60% of tracked devices are continuously connected at the same IP address for years, fewer than 30% ever receive a software update, and that the overall population is growing with the addition of the newest hardware and software from major manufacturers.

Despite these vulnerabilities, we show that the actual risk to ICS devices is low. We provide results from the largest, high-interaction ICS honeypot study to date; monitor scanning malware for knowledge of industrial protocols; and aggregate ICS interest from a database of 70 million hacker forum posts. Collectively, the data demonstrate that the cybercrime community has little competence or interest in ICS.

To explain the limited interest, we introduce a security economics model for character-

ising and predicting large-scale adversarial interest in a population of Internet-connected devices. The model was developed by studying successful, wormable malware such as Conficker, Mirai, and WannaCry, and identifying aspects of the device population or adversarial community that enabled large-scale exploitation. The model exposes several factors that currently deter large-scale malicious interest in the ICS domain, such as fragmentation in the ICS population, limited compute and memory resources on most ICS devices, and high cost of entry. The model also helps anticipate the effect of changes to the ICS population or adversarial community. We map ongoing and predicted changes to the industrial landscape and show that the ICS community is converging with the Internet-of-Things (IoT) community in terms of homogeneity and connectivity. We conclude that such a convergence makes the ICS domain an attractive target for attackers displaced by increased security or competition within the IoT domain; and even if ICS devices are not directly targeted, such convergence creates a new risk of being swept up in large-scale attacks targeting the billions of non-industrial IoT devices.

We make the following contributions in this chapter:

- A security economics model for characterising and predicting the success of large-scale attacks against a device population (Section 3.2) and an empirical demonstration of our model against the ICS device population (Sections 3.3 and 3.4).
- The first longitudinal study of Internet-connected ICS devices and a demonstration that individual ICS devices can be retroactively tracked for years using publicly-available scanning data (Section 3.3).
- An economic and usability explanation for the growth of the Internet-connected ICS population (Section 3.3.5).
- A consolidation of existing literature associated with ICS runtimes, Industry 4.0, and Industrial IoT (IIoT), and an analysis of how these changes affect cybercriminal interest in the domain (Section 3.5).

This chapter is largely derived from a paper presented at the peer-reviewed *15th Symposium on Electronic Crime Research (eCrime '20)* [56]. My co-author Daniel Thomas provided the initial idea to perform a longitudinal study, and he set up the initial database to find overlap between ICS devices and Mirai hosts. I performed all data collection, fingerprint generation, and processing. I also performed the case studies and created the security economics model used throughout the paper.

3.1 History of connected ICS

ICS are used to command, manage, or regulate physical systems in industry, critical infrastructure and building automation. ICS are often composed of many devices and may be distributed over a large area. Devices communicate using industry-specific protocols, most of which are legacy point-to-point or broadcast protocols designed for use with dedicated cabling; however, many protocols are now layered on top of TCP or UDP, and devices use existing IP-based networks, including the Internet, to communicate. As neither these legacy protocols nor the applications that use them support or implement authentication or encryption, a remote adversary can take control of a device and cause physical damage by sending well-formed packets [136] or modifying programmable logic [177,

Table 3.1: Sample Ethernet/IP data. 'Host IP' has been redacted and 'Serial number' has been modified to remove identifying information.

Property	Value
Host IP	[REDACTED]
Vendor ID	Rockwell Automation/Allen-Bradley
Product / Firmware	1766-L32BXBA / B14.00
Firmware Version	B14.00
Serial number	0x4062f367
Device [Local] IP	192.168.52.2

Table 3.2: Sample S7comm data. 'Host IP' has been redacted and 'Serial number of module' has been modified to remove identifying information.

Property	Value
Host IP	[REDACTED]
PLC name	S7-300
Module type	CPU 315-2 DP
Module	6ES7 315-2AH14-0AB0 v.0.2
Basic Firmware	v.3.0.3
Serial number of module	S C-A9VV07081020

217]. Even when securable protocols are used, they are often misconfigured in ways that reduce or eliminate any security benefit [49].

Shodan [186] and Censys [30, 61] provide Internet-wide views of devices responding on ICS protocol ports by scanning the IPv4 address space and making response data publicly available. Table 3.1 is a sample response for the query 'port:44818'. Port 44818 corresponds to the Common Industrial Protocol over Ethernet, known as Ethernet/IP, used in time-critical process automation applications, and about 8 500 unique devices are identified by Shodan. Protocols which provide similar levels of detail in the query response include Siemens S7 (port 102), typically used to control manufacturing processes (see Table 3.2), and BACnet (port 47808), designed for large-scale building automation (see Table 3.3) [136]. Other protocols, such as Modbus (port 502) and DNP3 (port 20000), provide less information (see Table 3.4), but still reveal whether a device using that protocol is alive at a given IP address.

As neither these legacy protocols nor the applications that use them support authentication or encryption mechanisms, an adversary can take control of a device and cause physical, possibly catastrophic, damage by sending targeted packets [136], or simply too many packets [149]. For at least one protocol used in building automation and control (BACnet), a detailed threat analysis [90] and summary of known protocol-level attacks [157] is readily available. Additionally, as many ICS devices are designed to be configured via the ICS protocol, Programmable Logic Controller (PLC) code or other configuration data can be directly downloaded or uploaded via the ICS protocol port [177, 217].

Prior studies quantified the size of the Internet-connected ICS population and demonstrated that the population was growing in absolute terms: studies in 2011, 2014, and 2016 identified approximately 8 000, 13 500, and 60 000 devices, respectively [115, 136, 163]. At

Table 3.3: Sample BACnet data. 'Host IP' has been redacted and 'Description' has been modified to remove identifying information.

Property	Value
Host IP	[REDACTED]
Object Name	W_Montana_HS_and_MS_200
Location	Maintenance Office
Vendor Name	Tridium
Application Software	Tridium 3.5.406
Firmware	3.5.406.1
Model Name	NiagaraAX Station
Description	Tridium WEBS-600

Table 3.4: Sample Modbus data. 'Host IP' has been redacted to remove identifying information.

Property	Value
Host IP	[REDACTED]
Product code	TM221CE16T
Vendor	Schneider Electric
Revision	V1.0

the time of writing, the tools used in these studies identify approximately 100 000 devices. Recent work evaluating Internet exchange traffic indicates that these 100 000 devices may be only a small percentage of the industrial devices using the Internet to communicate via insecure protocols [17] (e.g., because Shodan cannot identify devices behind Network Address Translation (NAT) or firewall devices).

Identifying the application of Internet-connected ICS devices is challenging because, with the exception of BACnet devices, device responses are not sufficiently descriptive and most device types are not sufficiently prescriptive to infer an application. One study used WHOIS records to identify business subscribers and confirmed that Internet-connected ICS devices were located in medical centres, energy companies, and water and sanitation organisations [136]. Another study downloaded PLC configurations from Allen-Bradley PLCs, and subject matter experts used indicators such as variable names to classify up to 60% of the devices as critical infrastructure [177, 217]. These methods are approximate and rely on inference, but they provide strong evidence that a non-trivial number of vulnerable, Internet-connected ICS devices are installed in industrial and infrastructure applications. In our study, we attempt to confirm the commercial application of these devices by evaluating population stability and IP assignment types (Section 3.3.4).

Academic and industrial literature effectively blames the user for the current state of Internet-connected affairs, but a sizeable body of usable security literature demonstrates that wholesale objection to user behaviour is ineffective [4]. From the user's perspective, the actual cost to maintain recommended levels of security may significantly outweigh the perceived or actual harm of a compromise; therefore, many users make the rational choice *not* to implement security recommendations [88]. Further, users have a limited time budget to spend on security, and often the advice, demands, and worst-case scenarios presented by the security community are not sufficiently persuasive to convince users to

take time away from some other activity to apply to security [87]. While much usable security literature deals with passwords and basic cyber-hygiene, the arguments can also be applied to ICS, where users are advised either not to make use of the Internet at all, which does not accord with their expectations, or are given the burden of implementing and maintaining new security infrastructure (e.g., virtual private networks (VPNs)) not included with the devices themselves. For example, a used, low-end Siemens SCALANCE security appliance is currently selling for over $400 on E-Bay, which is about the same cost as the industrial controller it is designed to protect, and two security appliances would be required to support communication between controllers over the Internet. In our study, we address the usability of security recommendations for ICS devices to explain the existing population of Internet-connected devices and to demonstrate that the growing trend for more connectable ICS devices will almost certainly result in a greater number of Internet-connected devices (Section 3.3.5).

Similarly, the literature documenting the risks of Internet-connected ICS (e.g., [115, 136, 163]) has not documented any instance of actual, much less worst-case, harm to an Internet-connected ICS. Without such demonstration, arguments to invest time and money in security infrastructure do not make economic sense for many device owners [23]. We provide initial results from three studies looking at scalable and targeted ICS attacks to demonstrate that there is little interest in attacking ICS devices at scale, but there is evidence for limited, targeted use of open industrial protocol ports to manipulate device behaviour (Section 3.4).

3.2 Security economics model

To better understand the risk of large-scale exploitation to Internet-connected ICS devices, we looked to successful exploitation of other Internet-connected populations. Using the categories in Anderson *et al.*, we pair at least one case study with each large-scale criminal activity: ransomware, cryptomining malware, Distributed Denial-of-Service (DDoS)-for-hire, spam, and disruption/destruction [7]. We use these studies to develop a security economics model for evaluating the risk of large-scale exploitation to an Internet-connected population of devices. Table 3.5 compares the technical details of the case studies. Table 3.6 summarises the security economics model, comparing the characteristics of the target population and attacker for each case study.

3.2.1 Case studies

Here we present case studies of large-scale attacks used to establish our model, focusing on named attacks or campaigns to support reproducibility and traceability. Further, we preferentially select recent attacks or attacks that affected cyber physical systems, as our primary use for the model is to explain and predict cybercriminal interest in Internet-connected ICS devices.

There have been several high-profile attacks against ICS devices (e.g., Stuxnet [101], BlackEnergy 3 [5], CRASHOVERRIDE [38]); however, these were targeted attacks and used industrial, Windows-based infrastructure to attack ICS devices that were not directly connected to the Internet; therefore, they are not considered here.

Conficker Conficker emerged in October 2008, using multiple propagation, exploit, and self-update techniques to infect millions of Windows computers. Patches were released by Microsoft in the same month. At its peak in 2009, F-Secure reported nearly 9 million infections [148], and a decade later there were still 500 000 active infections [24]. The worm took advantage of a large population, known vulnerabilities, slow patch cycles, and a collection of existing exploits, including some from Metasploit [142]. Conficker was originally deployed by Ukrainian cybercriminals, but crime-related payloads only appeared in the fifth variant with limited distribution [24, 102].

Mirai The Mirai botnet emerged in late 2016 and used aggressive scanning and brute force password searches to infect hundreds of thousands of Linux-based IoT devices, such as routers and webcams. At its peak in 2017, an estimated 600 000 hosts were infected [8]. In 2020 the Cambridge Cybercrime Centre (CCCC) [53] still observed approximately 150 000 unique, infected IPs per day scanning a monitored /14 network. The success of Mirai has been attributed to the efficient use of Internet-wide scanning, insecurity of the target devices, ease of porting to a large number of device types, and a lack of IoT patching infrastructure. Further, the original source code was publicly released on a hacking community forum, allowing individuals to create botnets without possessing the skill to develop the tool themselves [8].

Brickerbot Brickerbot malware was identified in early 2017 and used Mirai-like techniques to infect Linux-based IoT devices and perform Permanent DoS attacks [25, 164]. The author of the malware claimed he was mitigating the spread of Mirai and that he had 'bricked' over 10 million devices [33]. The malware continued to spread for over a year before being intentionally retired by its author. Even if the number of infections is exaggerated, Brickerbot provides an example of reusing or co-opting an existing botnet.

WannaCry and NotPetya WannaCry and NotPetya are ransomware strains exploiting a vulnerability in Microsoft's implementation of the Server Message Block (SMB) protocol to spread across the Internet and between computers on a local network. The exploit, known as EternalBlue, was allegedly developed by the NSA and publicly released by an organisation called ShadowBrokers [63, 143].

WannaCry appeared in May 2017 and infected hundreds of thousands of Windows machines. Though Microsoft rapidly patched the vulnerability, even for officially-unsupported Windows versions, the attack was finally halted by a security researcher registering a kill-switch domain, preventing the malware from spreading further. The original version of the malware was flawed, and encrypted computers could not be decrypted, though subsequent versions allowed users to pay a ransom and recover data [216].

NotPetya appeared in June 2017 and similarly used EternalBlue to spread between Windows machines [158]. It was a highly modified version of the Petya ransomware strain and it propagated through popular accounting software in Ukraine [147]. The reliance on accounting software for initial distribution limited the direct impact to tens of thousands of victims. NotPetya turned out to be a disruption attack masquerading as ransomware, as the malware provided no means to decrypt infected computers [80, 125].

Webstresser Webstresser was one of the largest DDoS-for-hire, or 'booter', services leveraging UDP amplification to direct traffic at a target. In the six months before its

Table 3.5: Comparison of technical details of case studies in Section 3.2.1.

		Conficker	Mirai	Brickerbot	WannaCry	NotPetya	Webstresser	Cryptomining
Potential use	Ransomware	✓			✓	✓		
	Cryptojacking		✓					✓
	DDoS		✓				✓	
	Spam and advertising scams	✓	✓					
	Disruption/destruction	✓		✓	✓	✓		
Botnet-based		✓	✓	✓				✓
Method of infection	Exploit functionality						✓	✓
	Exploit vulnerability	✓	✓	✓	✓	✓		✓
	Unnecessarily-open ports		✓	✓				
	Weak passwords	✓	✓	✓				
	Websites/JavaScript							✓
	Phishing/weaponised files	✓				✓		
Persistence	Host-based payload	✓	✓	✓	✓	✓		✓
	Host-based execution							✓
	Direct manipulation						✓	
	Multi-variant/adaptable	✓	✓		✓		✓	✓

takedown in April 2018, Webstresser executed over 400 000 attacks [36]. UDP amplification attacks use common Internet protocols and services (e.g., DNS, NTP) to create DDoS attacks without a dedicated botnet. The attacks leverage traffic asymmetries with these services, where a client request of a few bytes results in a server response of tens or hundreds of bytes. The attacker spoofs the source IP in the request, so the response from the service is directed at the target. The attacker is not making use of any particular vulnerability, but rather exploiting characteristics of the service that cannot be 'patched' without fundamental changes to the service [205]. Further, booting appears to be a lucrative and easy service to set up, and, until recently, did not result in adverse consequences for providers [36, 98, 107].

Illicit cryptomining Illicit cryptomining computes cryptocurrency hashes using host resources without permission from the owner. With the introduction of cryptocurrencies with changing proof-of-work (PoW) algorithms (e.g., Monero), binary-based cryptomining with botnets has become profitable. As a result, many cryptomining campaigns use a combination of publicly available mining tools and pay-per-install (PPI) botnets [154]. This ecosystem is designed to adapt when the PoW algorithm changes: miners are updated to the new algorithm and the cryptomining campaigns pay the botnet owners to install the new miner on botnet hosts. Pastrana *et al.* estimate that as of November 2018, such campaigns had illicitly mined over $57 million in Monero [154].

Table 3.6: Large-scale cybercrime enablers. (●) denotes a fully-met enabler, (◐) denotes a partially-met enabler, and (○) denotes a un-met enabler. Un-shaded columns are populated based on the enablers identified in the case studies (Section 3.2.1). Shaded columns represent new analyses of the current Internet-connected ICS population (Sections 3.3 and 3.4) and a predictive evaluation of the future population based on current trends in ICS connectivity (Section 3.5).

Category	Enabler	Conficker	Mirai	Brickerbot	WannaCry	NotPetya	UDP amp	Cryptomining	ICS: current	ICS: future
Vulnerable population	Internet-connected	●	●	●	●	●	●	●	●	●
	Large and/or stable	●	●	●	●	●	●	●	●	●
	Slow patching/no support	●	●	●	○	○	●	●	●	●
	Software and/or hardware homogeneity	●	●	●	●	●	●	●	○	◐
	Onboard resources	●	●	●	●	●	●	●	○	◐
	Known vulnerabilities	●	●	●	●	●	◐	●	●	●
	Predictable system/device response	◐	●	●	●	●	●	●	○	◐
Attacker incentives	Financial	◐	●	○	◐	◐	●	●	○	◐
	Ideological	○	○	●	◐	◐	○	○	○	◐
	Low exploitation cost	●	●	●	●	●	◐	◐	○	◐
	Low consequence to attacker	●	◐	●	●	●	○	●	○	◐
Attacker tools and resources	Develops exploits	◐	○	◐	●	●	◐	◐	○	●
	Adapts existing exploits and tools	○	●	○	●	●	◐	●	○	●
	Buys or co-opts access	○	◐	◐	○	○	◐	●	◐	◐

3.2.2 Large-scale cybercrime enablers

From these case studies, we identify three, largely-orthogonal categories that are present in each case: a *vulnerable population*, clear *attacker incentives*, and a degree of domain-specific *knowledge, tools, and resources*. Within each of these categories, the case studies reveal several sub-properties, or 'enablers', that characterise the larger category. The categories and constituent enablers are discussed further below. Table 3.6 maps the individual case studies into our model, showing the enablers that are fully or partially met for each of the attacks or campaigns. We mark enablers as 'partially-met' if there is uncertainty (e.g., whether the WannaCry incentive was financial or disruptive), if the enabler is possible but not well-documented (e.g., co-opting access to Mirai botnets), or if the enabler is possible but not well-executed (e.g. Conficker spreading faster than intended). We also include columns for Internet-connected ICS devices based on a study of the current population (Section 3.3) and the anticipated, future population (Section 3.5). For the latter, we consider enablers to be partially met if they are expected to be possible but are not yet demonstrated.

We argue that enablers from these three categories must be present in high measure for successful, large-scale exploitation of a population. We also show that there is some element of progress; namely, that the existence of a vulnerable population creates incentives that lead to attackers either developing or adapting tools to target that population. Further, without intervention, a positive feedback loop exists, such that a larger attacker population will be attracted to a domain, bringing broader incentives and skills. The following sections describe the categories in more detail, emphasising each *constituent enabler* corresponding to a row of Table 3.6.

Vulnerable population All cited cases involve a *large*, *Internet-connected* population of devices (e.g., Windows computers, IoT devices). While some of these cases could spread via removable media, none primarily rely on such methods for propagation; in all cases, hosts are directly addressable and have open ports. Similarly, these populations all have (or had) *unpatched vulnerabilities* or are fundamentally constructed in such a way that precludes simple mitigation. The populations also have a strong, initial *homogeneity*. To perform the tasks required by the malware authors, devices require sufficient *onboard resources*. In the case of ransomware, this may include sensitive information. For a DDoS botnet, this includes sufficient spare processing and memory resources to avoid impairing normal device function, as the nature of a botnet is to be a parasite. Finally, the device or system response in all cases is *predictable*, though in some cases the malware appears to have spread faster (e.g., Conficker) or wider (e.g., NotPetya) than initially intended.

Attacker incentives The most obvious incentive for cybercrime is *financial*, as demonstrated by ransomware, spam botnets, and DDoS-for-hire services. There may be other incentives, however. So-called 'script kiddies' may seek prestige in their community (e.g., developing DDoS infrastructure to boot rivals from online games [36]). Alternately, *ideology* (e.g., Brickerbot) or grey conflict (e.g., NotPetya) may provide the incentive. An attacker must be able to turn the attack into the capital of the incentive. For example, a ransomware campaign against devices without screens requires an additional step to inform the device owner of the attack and provide a remediation mechanism [68]. Similarly, attackers must ensure gains exceed costs, including the financial *cost to develop and deploy* the attack and the potential *personal consequences* of executing an attack. There is currently little appetite for pursuing low level cybercriminals [6, 7], and even the authors of Mirai avoided jail time, despite a conviction [108]; however, there have been recent cybercrime prosecutions, including those associated with booter services [36, 98, 107], which may be the leading edge of change.

Attacker tools and resources Most perpetrators of cybercrime do not appear to *find and exploit vulnerabilities* themselves, but *rely on and adapt tools* from sources with greater skill and different motivations. WannaCry's reliance on EternalBlue, which has a nation-state pedigree, is a prime example [63, 143]. Similarly, the original Mirai source code was published by an individual with perhaps greater skill than many of the code's subsequent users. The Mirai author even mocked the 'skid[s]' who would use it: "I know every skid and their mama, it's their... dream to have something besides qbot"[183]. This highlights the fact that even if a given population of devices appears to be too challenging to exploit on a given day, a high-capability tool placed in the hands of low-capability adversaries can change the threat model in a very short period. Finally, in some cases attackers *buy or co-opt* access, such as in binary-based criminal cryptomining, which relies on PPI botnets [154].

3.2.3 Security economics model summary

Our model consists of the categories, enablers, and descriptions elaborated above and exercised for several case studies in Table 3.6. It provides a qualitative foundation for security economics assessments of target and adversarial populations to predict the likelihood of large-scale cybercriminal interest in the target population, or to assess the

factors driving an existing interest in a target population. In the following sections we apply the model to Internet-connected ICS devices and potential adversaries.

3.3 Longitudinal study of ICS

To apply the model in Section 3.2 to Internet-connected ICS devices, we performed a longitudinal study to identify device characteristics and infer device owner behaviour over a four year period. To identify enablers for large-scale exploitation, we leverage historical data from both Shodan and Censys to track individual devices, allowing us to investigate population growth, stability, and patch cadence.

3.3.1 Internet-wide ICS scanning

Shodan and Censys scan dozens of ports over the IPv4 address space and make results publicly available. Both request information from ICS hosts using several protocols. The six most common industrial protocols are Ethernet/IP (port 44818), BACnet (port 47808), S7comm (port 102), Modbus (port 502), DNP3 (port 20000), and Niagara Fox (ports 1911 and 4911). Shodan scans all six protocols, while Censys scans all except Ethernet/IP. We focused on Ethernet/IP, S7comm, and BACnet because devices communicating over these protocols self-report more information than other protocols.

Threat to validity While ports 44818, 102, and 47808 are the default (and most common) ports used for Ethernet/IP, S7comm, and BACnet devices, respectively, the port number may be configurable and Internet-connectable devices can communicate using these protocols on alternate ports. Gasser *et al.* demonstrated that only about 50% of Internet-connected BACnet devices use port 47808, and the remainder use ports 47809 to 47823 [76]; however, they showed that devices on alternate ports were equally susceptible to protocol-based attacks. Shodan and Censys only scan the default ports (i.e., 44818, 102, and 47808), thereby missing up to half of Internet-connected ICS devices. Because the devices on these alternate ports appear to have the same characteristics as the devices on the default ports, and to minimise our own active scanning (based on ethical considerations discussed at the end of this chapter), we chose to limit our datasets to those provided by Shodan and Censys, but consider our conclusions to apply to the broader Internet-connected ICS population.

A similar argument applies to the possibility that ICS devices are being connected via IPv6 rather than IPv4. As discussed in Chapter 2, Internet-wide scanning of IPv6 is infeasible due to the number of addresses; therefore, Shodan and Censys may be missing a large population of Internet-connected ICS devices. As above, the use of IPv6 (like the use of an alternate protocol port) has no correlation with improved security practices; therefore, the possibility of a large population of IPv6-connected devices does not affect the validity of our conclusions. Supporting this, recent work monitoring industrial traffic at Internet Exchange Points (IXPs) indicates that the number of Internet-connected ICS devices using insecure protocols over the wider Internet is significantly larger than that observable via Internet-wide scanning, as IXP monitoring is able to identify devices hidden behind NAT devices and firewalls or devices connected via IPv6 [17]. For our purposes, the key takeaway from the IXP monitoring is that a device hidden from Internet-wide scanning is not necessarily used in a more secure manner. Rather, it appears many users

Table 3.7: Summary of observed ICS device population sizes as an average over the duration of the study.

Source	Manufacturer	Devices per Snapshot	Fingerprints per Snapshot
Censys	Siemens	3 881	1 105
	Tridium	2 117	1 882
Shodan	Siemens	3 227	1 025
	Tridium	2 380	2 079
	Allen-Bradley	6 985	5 674

simply connect their ICS devices to their local access points with default network settings; this may result in an IPv4 or IPv6 address and may or may not hide the device behind a NAT or firewall, but the addressing and boundary schemes do not provide an inherent or correlated improvement in security practice.

3.3.2 Fingerprinting

Globally-unique, static fingerprints support tracking devices over time to identify changes to the Internet-connected ICS population and to observe user behaviour. Device fingerprinting using historical data relies on identifying device-specific, immutable characteristics in existing datasets. From the three protocols of interest, we developed fingerprints for the manufacturer with the largest number of devices responding on that protocol: Allen-Bradley for Ethernet/IP, Siemens for S7comm, and Tridium for BACnet. Allen-Bradley and Siemens devices provide a serial number that appears to be globally unique, with the exception of honeypots (e.g., default instances of Conpot [169]). Tridium devices do not report serial numbers, but do report several fields of user-configurable data. By evaluating data returned from hundreds of devices, we chose the 'object name' field, as it is generally configured by the user and does not appear to have a default value. Using 'object name' produces globally-unique fingerprints for about 90% of devices, with the exception being devices with general entries such as 'device'. Table 3.7 compares the number of ICS devices to the number of devices with unique fingerprints, showing that we are able to uniquely identify a large percentage of the ICS device population to support our longitudinal study.

Threat to validity Our fingerprints are limited by the lack of ground truth data and reliance on user-configurable data instead of inherent characteristics of each device (e.g., clock skew [141] or factory calibration [220]); however, the stability observed over several years indicates that the fingerprints are sufficient for our study. Additionally, the conclusions drawn in this section are based on the ability to track individual devices to determine their stability, longevity, and patch cadence. For Tridium and Allen-Bradley devices, the ratio of fingerprintable devices to total devices is high (approximately 90% and 80%, respectively). In contrast, the ratio for Siemens devices is low (approximately 30%), introducing potential bias to our analysis. This would be a concern if the lack of a fingerprintable serial number on the remaining 70% of Siemens devices was evidence of greater security awareness amongst device owners; however, the fact that these devices are connected to the Internet at all implies that they are not configured more securely, but

Table 3.8: Comparison of devices identified by Shodan and Censys between December 2018 and March 2020.

Vendor	Number of Devices		
	Intersection	Shodan Only	Censys Only
Siemens	1 305	468	418
Tridium	3 016	331	206

that the device type simply does not report a serial number by default. Based on this, we consider conclusions regarding the fingerprintable Siemens devices likely apply to the entire Internet-connected Siemens population, but further work is required to demonstrate this conclusively.

3.3.3 Comparison of Shodan and Censys

Other studies have qualitatively demonstrated inconsistencies between Shodan and Censys [82, 136], while our fingerprint allows us to quantify and characterise the difference.

Shodan returns all hosts open on a given port, whether or not they respond correctly to protocol-specific requests. For example, on 10 June 2019, Shodan returned 55 989 results for the query 'port:44818' (Ethernet/IP), of which only 8 501 responses could be parsed as Ethernet/IP data. The remainder are likely incorrectly configured hosts unrelated to the ICS domain. We find all Censys returns to be valid, in that all returns provide parsable, protocol-specific responses, going some way to validating results from studies based on ZGrab, the underlying scanning tool for Censys [61, 136]. In contrast, studies relying solely on Shodan data that do not distinguish between an open port and a correctly-communicating device may over-predict connected devices numbers.

To characterise the difference between Shodan and Censys, we used our fingerprint to compare the set of devices returned by each source. Table 3.8 provides a high-level summary of the comparison, which was limited to devices with a unique fingerprint obtained between December 2018 and March 2020.

Table 3.8 shows broad agreement between the two search engines. The larger difference between the Siemens results may be due to intentionally-limited commands used by Censys [136]. A high-level analysis of the devices identified by one search engine over another did not reveal any obvious reasons for the difference; however, an analysis accounting for the IP blocks from which each search engine scans may be more instructive [82].

We also compared the data for devices identified by both search engines and found nearly perfect agreement. For example, there was 100% agreement for the Siemens model numbers (e.g., 6ES7 315-2EH13-0AB0). Similarly, all but two of the Tridium firmware versions agreed, and the difference appears to be due to a recent software update that had been indexed by Censys but not Shodan. This is further supported by Table 3.7, which shows a consistent fingerprint-to-device ratio between Shodan and Censys.

3.3.4 ICS device owner behaviour

While previous studies accurately state that devices are vulnerable to manipulation as long as their industrial protocol port is open to the Internet, they do not attempt to

understand why or how these devices are connected. We demonstrate the ability to infer behavioural characteristics of device owners, including how the devices were connected to the Internet, the software update frequency, and the age of devices being connected.

Data collection methodology Censys provides access to past scanning results via Google BigQuery. We took retroactive, quarterly snapshots from December 2015 to March 2020, resulting in a dataset of all S7comm and BACnet device indexed by Censys on the first day of each quarter. Shodan does not provide direct access to historical scans, but does support historical queries of individual IP addresses. We took daily Shodan snapshots between December 2018 and July 2019, and then occasional snapshots until March 2020, and used the IP addresses in those snapshots to perform historical queries. The resulting dataset includes three to four years of data for ICS devices recently indexed by Shodan; however, any device disconnected before December 2018 would not have been in the list of IPs for which historical queries were performed. To overcome this limitation, IP addresses indexed by Censys between December 2015 and March 2020 were used to supplement the list of IP addresses indexed by Shodan, and this composite list of IP addresses was used to perform Shodan historical queries, surfacing Shodan data for ICS devices that were no longer indexed by Shodan in December 2018. This technique was used for Siemens and Tridium devices only, as Censys does not scan port 44818, used by Allen-Bradley devices.

Threat to validity The use of IP addresses from Censys in Shodan historical queries biases any further comparison between the two datasets, and we did not use this technique in Section 3.3.3. In this section, our goal is to maximise the data available from any IP address at which an ICS device has been hosted and we consider the inclusion of IP addresses from Censys the best way to do so, especially in the case of Siemens devices, as only Shodan accurately parses the Siemens firmware information. Similarly, performing quarterly Censys snapshots risks missing devices connected for short periods (i.e., less than 90 days, between quarterly snapshots); however, daily Shodan snapshots showed that almost all devices remained at the same IP address and were connected for extended periods, so we considered more frequent Censys snapshots to be unnecessary.

Stability and IP assignment We used our fingerprint to identify the time each device spent in the population and the stability of the device's IP address. Table 3.9 documents the percent of devices seen more than once with a *stable presence* (i.e., devices in the population for more than one year) and with a *stable IP* (i.e., devices only observed at one IP address). As discussed above, the Shodan data for Allen-Bradley is skewed toward devices with shorter connection times, as it only incorporates the history of devices connected between December 2018 and March 2020, whereas the remaining rows include all devices connected between December 2015 and March 2020.

This is the first confirmation of the stability of the Internet-connected ICS population, demonstrating that most devices have stable IPs, are continuously connected, and are connected for extended periods. Combined with the lack of patching, this means that an attacker has a nearly unlimited opportunity to reconnoitre, develop, and deploy attacks against this population.

Further, these results provide some insight into the different device environments. For example, nearly 60% of the Allen-Bradley devices are connected via cellular Internet Service Providers (ISPs), such as Verizon or AT&T Wireless, which, combined with the

Table 3.9: Summary of ICS device population behaviours over the course of the study. All values represent a percentage (%) of fingerprintable devices.

Source	Vendor	Firmware update	Stable IP	Stable presence	Replaced devices
Censys	Siemens	-	58.7	71.3	-
	Tridium	29.6	66.3	64.4	-
Shodan	Siemens	0.5	73.0	40.4	1.3
	Tridium	22.3	82.7	53.7	5.7
	Allen-Bradley[a]	1.8	85.1	63.1	5.0

[a] Historical query based on Shodan IP addresses only

stable IP addresses, suggests they are field devices; whereas fewer than 5% of Siemens devices have cellular ISPs, but double the percentage of dynamic IP addresses, implying they may be inadvertently connected to the Internet, as a dynamic IP has limited value for remote monitoring and control without a service like DynDNS.

Table 3.9 shows a higher stable IP address percentage, but a lower stable presence percentage for Shodan compared to Censys. This is an artefact of our data collection methodology. By performing historical Shodan searches on individual IP addresses obtained from Censys, we expect IP address stability for the Shodan results to increase, since the set of devices is now the union of those identified by Censys and Shodan, whereas the Censys percentages would not include any device only identified by Shodan. Similarly, because we are performing historical searches on IP addresses identified by Censys as early as December 2015, but no longer observed by Shodan in December 2018 (our first Shodan snapshots) we would expect the stable presence percentage to decrease, as the search window is effectively truncated by 15 months.

Firmware Devices that use Ethernet/IP, S7comm, or BACnet over IP provide detailed firmware and/or application software version information in response to a protocol-specific request (e.g., Table 3.1). In all cases, the version information provides sufficient granularity to correlate with release notes and Common Vulnerabilities and Exposures (CVEs).

Table 3.9 shows the percentage of devices seen more than once for which at least one *firmware update* was observed. We used Censys to track Tridium firmware and Shodan to track Tridium, Siemens, and Allen-Bradley firmware. While Censys does index Siemens data, the firmware version is not correctly parsed. Censys confirmed to us that the data was incorrect and the issue was being tracked, but stated that they did not keep raw historical data for re-processing.

Overall, a stark contrast is evident between building automation controllers (Tridium) and Programmable Logic Controllers (PLCs) (Allen-Bradley and Siemens). The difference may be accounted for by the centrality (logically and physically) of building automation controllers compared to PLCs, which may make it easier to install updates and anticipate or test effects on the rest of the system. Similarly, maintenance contracts for building automation systems may include the entire, integrated system, so the third-party is responsible for and licensed to install updates and test the whole system. Conversely, maintenance contracts for physical processes under PLC control may be limited to the

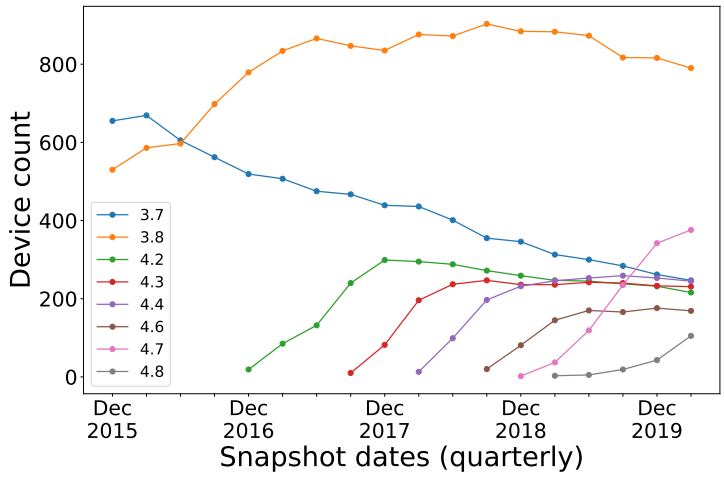

Figure 3.1: Tracking firmware versions of Tridium devices running the Niagara framework shows that few devices are updated and that the population includes the newest device hardware and software.

single device or process, so the third-party maintainer is not responsible for the integrated testing that may be required following a software update. Finally, it may be that the risk and consequence of disruption from patching is less in the building automation domain than in other industrial domains.

While it may seem that a failure to patch is a secondary problem while the protocol port remains open, these observations raise two immediate concerns. First, vulnerabilities may provide an attacker greater access than the protocol alone. For example, over 70% of the Tridium devices identified by Shodan report a firmware version against which there is at least one critical-severity CVE allowing an attacker to gain full administrative privileges (CVE-2017-1674). Second, we hypothesise that these results are at least qualitatively applicable to non-Internet-connected ICS devices, as software updates may need to pass through an integrator, be installed by a third party, and require integrated testing after installation, all of which incur delay and engender resistance.

Figure 3.1 shows frequency plots of firmware versions over time for Tridium devices running the Niagara framework, demonstrating limited, long-term patching behaviour. The figure also shows that connecting ICS devices directly to the Internet is not a legacy problem: even the newest devices with the most recent firmware (version 4.8 was released in 2019) have their industrial protocol ports directly connected to the Internet. It is encouraging to see that manufacturers are releasing updates that are incorporated in new device installations; however, the infrastructure to distribute and install patches in live systems remains inadequate.

Table 3.9 also shows the percentage of *replaced devices* observed between December 2015 and March 2020. We tracked this metric separately, but include it in this section because most replacements reported newer firmware. Replacements were identified by fingerprints and timestamps. If one device at an IP address was no longer observed (at any IP address) after the initial observation of a second device at that IP address, the first device was considered replaced. The method is unable to account for DHCP churn and likely under-predicts replacements; however, the majority of devices have stable IP addresses, so the replacement percentages are at least qualitatively instructive and show that device upgrades do not compensate for lack of patching.

3.3.5 ICS usability and security economics

The vulnerability of Internet-connected ICS devices raises an obvious question: why are these devices connected to the Internet in the first place?

We are not aware of empirical research explicitly asking this question to device owners; however, we can map existing usable security and security economics research from the IT domain into the industrial domain. Vendor guidance for ICS devices can be summarised as follows [16, 187, 206]:

1. Do not connect to the Internet
2. If you must connect, use segmentation and isolation
3. Use role-based access control and strong passwords

The guidance is sound but fails to consider user expectations and security models [4, 87], and it does not account for the cost/benefit decisions users are forced to make when determining the resources to invest in security [88, 139]. For example, there is currently no evidence of large-scale criminal interest in Internet-connected ICS devices (see Section 3.4), so the likelihood of an attack against an arbitrary industrial system appears low. Further, the security community has not demonstrated a common-case consequence for attacks against such devices. While the consequence of state-sponsored attacks against high-profile targets are well publicised (e.g., Stuxnet [101], Triton/Trisis [207]), it is unclear what attacks and consequences might be expected by smaller, low-profile organisations. With low likelihood and no demonstrable consequence, demanding a wide-spread investment in security is ineffective [88, 139].

The first recommendation does not accord with user expectations, as demonstrated by the fact that many devices are not only intentionally connected, but are packaged with cellular modems for that purpose. Also, given that many devices are equipped with webservers and industrial protocol ports enabled and open out-of-the-box, the recommendation is not enforced by default. Instructions for setting up webservers on many devices fail to mention that the industrial protocol port is open and accessible from the same physical port as the webserver; therefore, users may follow guidance for securing their webserver and inadvertently connect their unprotected, industrial protocol port directly to the Internet. This is highlighted by recent work showing 92% of reachable, Internet-connected OPC UA deployments (a securable industrial protocol) had critical security misconfigurations [49].

The second recommendation generally comes alongside additional security appliances that the user needs to buy in order to adequately secure the ICS device (e.g., security processors [187]). Basic security appliances can cost as much as the ICS device being protected, and generally two are required to secure each end of the communication link. Having purchased a security appliance, the user now has to consider the expertise and expense to maintaining it. Given the high cost and lack of identified risk, users may be making a rational decision not to invest in further security while still benefiting from the connectivity of their new device [88].

This recommendation highlights an additional problem with ICS devices: the lack of host-based security. Even with a segmented network, industrial protocols are such that all devices within a network segment must fully trust all other devices within the segment, creating usability challenges if the user expects one device to be accessible from outside the

segment (e.g., for remote monitoring or maintenance) without providing access to another device in the segment. Recommended solutions involve either shrinking the segment size to a single device, or adding additional security appliances to allow remote access to a single device, both of which add short and long-term costs to an installation.

The third recommendation has the same problems as the second: because most ICS devices lack host-based access control, implementing a robust access control policy across a facility with a collection of heterogeneous devices generally requires buying and maintaining additional infrastructure.

In summary, the number and growth of Internet-connected ICS devices is not surprising. Some of these devices may be inadvertently connected, due to failures in usable security. Others may be intentionally connected based on either rational, risk-based decision-making or inadequate understanding of security. Regardless, we should expect that the number of connected devices will continue to increase as the number of connectable devices grows.

3.4 Abusing ICS devices

The Internet-connected ICS population is large, stable, slow to patch, and growing, which meets several of the enablers for large-scale cybercrime (Section 3.2). The *ICS: current* column of Table 3.6 provides a comparison of Internet-connected ICS devices against other populations targeted by large-scale attacks, and shows that there are several missing enablers: homogeneity, predictable response, and onboard resources. The model predicts that these missing enablers will inhibit cybercriminal interest in the ICS domain. In this section, we attempt to empirically test that prediction by studying the adversarial population to complete the Table 3.6 analysis.

In this section, we are specifically interested in large-scale exploitation of ICS devices directly connected to the Internet. As noted in Section 3.2.1, there have been several high-profile attacks against ICS devices (e.g., Stuxnet [101], BlackEnergy 3 [5]); however, these used industrial, Windows-based infrastructure to attack ICS devices that were not directly connected to the Internet. Similarly, there have been high-profile ransomware attacks against industrial organisations (e.g., Norsk Hydro [48]), but, again, these attacks targeted Windows-based infrastructure. While they effectively shut down associated industrial processes, they were not attacks against ICS devices themselves.

First, we collaborated with SecuriOT[1] to evaluate a year of data from a global network of 120 high-interaction ICS honeypots to see whether malicious actors were attempting to modify the behaviour of ICS devices with protocol-specific commands (Section 3.4.1). Second, we used a database of over 70 million cybercrime forum posts to measure interest in and identify any explicit exploitation of ICS devices (Section 3.4.2). Finally, we evaluated whether the ICS devices, or the routers and modems through which they connected to the Internet, were infected with Mirai (Section 3.4.3). This was a proxy to see whether such devices were generally targets for botnets.

These evaluations provide no evidence of cybercriminal interest or expertise in attacking ICS devices at scale, nor that exposed industrial protocol ports are being used as a vector for attacking such devices. We acknowledge the challenges associated with proving a negative, but believe our conclusions are supported by consistent data from largely orthogonal domains (honeypots, cybercrime forums, and malware).

[1] https://www.securiot.se/

3.4.1 Honeypots

There are several ICS honeypot studies that identify large amounts of scanning traffic; however, none of these identify any instance of malicious manipulation of an ICS device through the industrial protocol [19, 136, 162, 212], though there are indications that this capability exists [66]. These studies were conducted over short periods, had a limited number of devices, were not geographically distributed, or used honeypots that were easily fingerprinted (e.g., they were hosted on Amazon Web Services and used default Conpot [169] configurations); therefore, the lack of interaction may not be surprising.

We collaborated with SecuriOT to overcome some of these shortfalls. SecuriOT develops deception technologies that mimic PLCs, remote terminal units, and firewalls, and they run 120 Internet-connected honeypots in 22 different countries as part of their own intelligence gathering operation. SecuriOT provided us one year of labelled interactions with their honeypot network, covering March 2018 through March 2019. We did not have access to raw packets, so our assessment was limited to packet header data, a fingerprint of the tool used for the interaction, and their analysts' classifications of interactions as either reconnaissance or exploitation.

In one year, SecuriOT recorded approximately 200 000 packets from 80 000 interactions, of which about 1 000 attempted to modify equipment behaviour. All but nine of these were Mirai variants initiating an SSH session with a router honeypot. The nine interactions targeting an ICS protocol used S7comm, Modbus, and IEC-104. According to SecuriOT, these attacks were either DoS or command replay attacks.

The results confirm there is no large-scale effort to indiscriminately compromise ICS devices, though there are many reconnaissance efforts. However, this data may provide the first evidence of efforts to maliciously manipulate Internet-connected ICS device behaviour through ICS protocols, demonstrating that attackers are interested in protocol-specific attacks and that high-interaction honeypots can successfully deceive them [57].

If Internet-connected ICS devices are only subject to targeted attacks, it is not surprising that the number of observed compromises is low, as they likely represent either an attacker testing against a random system or mistaking the honeypot for a specific target. This demonstrates one of the limitations of honeypots as intelligence gathering tools for targeted attacks: while they might reveal an attacker capability or interest, they cannot be used for quantitative measurement in the same way as honeypots targeted by indiscriminate malware like Mirai.

Chapter 4 expands on the honeypot network setup, datasets, and conclusions.

3.4.2 Hacker forums

To evaluate cybercrime community interest, we used the CCCC's CrimeBB dataset [53, 155], which contains over 70 million posts from over ten years, scraped from dozens of publicly-accessible cybercrime forums, including Hackforums, where the Mirai source code was originally shared. Analysing hacker forums has been used to characterise and predict cybercrime behaviours in several domains, including cryptomining malware [154], eWhoring [93], and booter services [36]. We searched for ICS-related terms (e.g., 'ICS', 'Shodan', vendor and protocol names), identified relevant posts, and evaluated the entire thread of which the post was a part.

The initial search returned over 13 000 posts, from which we filtered out posts that

were either not in English or obviously not applicable.[2] We manually evaluated the remaining posts for relevance before extracting applicable threads. We discarded threads that consisted solely of extracts from news articles or other websites without follow-on posts (e.g., copied Stuxnet articles), threads with hyperbolic claims (e.g., selling an exploit for $1 million), requests for information about hacking without follow-up, and conflations of ICS with IoT (e.g., dozens of posts with links to open webcams).

We identified fewer than 30 relevant threads, including tutorials, discussions of vulnerabilities, demonstrations of compromise, and offers of credible service. There were several Shodan tutorials, scrapers for obtaining Shodan API keys from GitHub, IP addresses of Internet-connected ICS devices, and videos of people manipulating thermostats via web interfaces. Two threads competently discussed how resource constraints and long lifetimes impeded security implementation in the ICS domain. One user claimed to work for a power utility and offered physical access to smart meters, though we found no evidence of anyone offering to pay for his services. Finally, we found two threads where managers of ICS equipment were, apparently inadvertently, providing details of their equipment and security practices; in both cases, they were complaining about the restrictions placed on them by system administrators.

Compared to other cybercrime domains, such as eWhoring, which has over 6 500 tutorial-related posts on Hackforums [93], the ICS picture is one of limited interest and competence. In other domains we observe correlations between posting and actual malicious activity, which is absent in the ICS domain. For example, a spike in Monero-related posts directly corresponds to a demonstrable increase of Monero crypto-mining malware in the wild [154]. Further, these other communities stimulate their own interest and profitability: the distribution of tools (e.g., source code) and knowledge (e.g., tutorials) allow less experienced actors to join and potentially profit, turning niche domains into large-scale attack targets. For ICS, we do not observe even a nascent or incubating community that would support development and distribution of such tools and knowledge.

3.4.3 Mirai

Given the cybercrime community's penchant for effective modification and reuse of existing malware, one might expect that initial efforts at large-scale exploitation of Internet-connected ICS devices would be based on modified IoT-targeting malware. Mirai is a prime candidate for such modification based on its flexibility, simplicity, and demonstrated effectiveness [8]. As all known Mirai variants target Linux-based hosts, Mirai would have to be modified for new target hardware and software, as few ICS devices are known to be Linux-based. It is also unclear whether a Mirai-like exploit could infect a device through the industrial protocol port, though proof-of-concept, wormable PLC malware has been demonstrated [68]. Such a worm may also be able to infect and spread via non-industrial protocols on an ICS device (e.g., FTP, HTTP). To detect such modifications to existing malware, one could either directly look for modified Mirai source code in the wild or indirectly look for infected ICS hosts. In this section, we describe the latter method.

Devices infected with Mirai scan for other potential hosts with a distinctive scanning packet, and we can use that distinctiveness to identify infected hosts [8]. The CCCC [53]

[2]For example, the search term 'ics' returned 1,033 posts, of which 286 referred to Android Ice Cream Sandwich, 111 referred to Internet connection sharing, and 165 were in Russian (while Russian posts may be relevant, we determined it was not worth translating them based on our evaluation of the English-language posts).

monitors a /14 network, collects these packets, and makes the data available to researchers. We used a real-time feed of source IP addresses suspected of hosting Mirai and identified approximately 150 000 unique, infected IPs per day. We used ZGrab[3] to scan these Mirai source IP addresses on industrial protocol ports to identify whether any ICS devices were hosted at the infected IP addresses.

Fewer than 1% of the Mirai source IP addresses provided a parsable response to our scans, indicating that there is only a small population of IP addresses hosting an ICS device and a Mirai host. Notably, none of the Mirai hosts were scanning ICS ports, strongly implying that they were not ICS devices, but rather 'normal' Mirai hosts (e.g., routers) sharing an IP address with an ICS device.

Mirai is frequently modified for different targets, and its scanning behaviour is unique and easy to detect, making it a good starting point for investigating whether cybercriminals are adapting existing malware for the ICS domain. This negative result is not generalisable to all possible ICS malware; however, the combination of passive honeypots with an active search for infected devices with ICS knowledge (e.g., the ability to parse industrial protocols) may provide leading indicators of a shift in cybercriminal focus.

3.4.4 Summary

In Section 3.3 we characterised the large, stable, and vulnerable population of Internet-connected ICS devices; however, our security economics model (Table 3.6) predicted that the population was insufficiently homogeneous, resourced, and predictable to attract attention from the cybercrime community. In this section, we used orthogonal investigative methods to empirically confirm that the cybercrime community has little competence or interest in the ICS domain. While our focus is on largely undefended, Internet-connected ICS devices, our assessment is confirmed by security companies responding to incidents on defended networks [202]: ransomware attacks against Windows infrastructure are the only large-scale attacks affecting industrial systems.

3.5 Changing industrial landscape

While the current landscape makes ICS devices unattractive targets for scaled attacks, expected changes to industry are likely to overcome missing enablers in the *ICS: current* column of Table 3.6. In this section we describe these changes, and in the next section we show how these changes may make the future Internet-connected ICS device population an attractive target for cybercrime.

ICS runtimes Industrial runtimes are third-party software that provide common development and execution environments across large numbers of heterogeneous device families. For example, the CODESYS runtime is used in ICS devices and development environments from Schneider Electric, Beckhoff Automation, Bosch, and WAGO, all major ICS vendors [151]. Runtimes are a growing target for security researchers because they create larger, homogeneous populations and broaden attack surfaces by adding complexity and new interfaces. For example, security researchers have developed several exploits in versions 2 and 3 of CODESYS, supported by at least 360 device types [1, 151].

[3] https://github.com/zmap/zgrab2

Industry 4.0 Industry is trending toward systems with greater connectivity, greater flexibility, and denser information flows, resulting in an increased number of devices in a system and an increased number of *connectable* devices in a system. Data-driven decision making (e.g., condition-based maintenance) requires more sensors providing high resolution data [127], and the desire for reconfigurable manufacturing environments encourages connectable devices, wireless connections, and Software Defined Networks (SDNs) [218]. The net result of these pressures is a greater number of connectable devices in a given industrial environment.

Common IoT and IIoT platforms Technology companies are capitalising on this IIoT market, deploying ready-made solutions for device manufacturers, integrators, and operators. For example, Microsoft's Azure Sphere is a hardware, software, and cloud solution for IoT and IIoT devices [130]. Similarly, Huawei has developed open source operating system and cloud solutions for IoT and IIoT [94]. These solutions are designed to make development and testing easy: development boards are inexpensive and readily available, and the software stacks include full, open source operating systems. These products greatly simplify the effort to develop and test malicious software by increasing platform availability and providing higher level and more powerful resources.

3.6 Economics of attacking ICS

In this section, we attempt to answer two questions using our security economics model:

1. Why is the current Internet-connected ICS device population not a target of large-scale cybercrime?

2. Will the changing industrial landscape make attacks against the future ICS device population more likely?

Using the *ICS: current* and *ICS: future* columns of Table 3.6 we address these questions by looking at each category, describing the enablers that are currently missing for the ICS population (*Current situation*) and showing how those enablers are or are not satisfied by expected changes to the industrial landscape (*Outlook*).

3.6.1 Vulnerable population

Current situation While the Internet-connected ICS population is approaching 100 000, it is fragmented amongst dozens of vendors, with a heterogeneous collection of proprietary hardware and software; even if an attacker developed wormable malware for a particular device family, the vulnerable population may only include a few thousand devices. For example, the largest population of Internet-connected ICS devices from a single vendor (Allen-Bradley) consists of fewer than 10 000 devices. Compared to 2.5 billion Android devices [160], 1.5 billion desktop computers [75], 1.3 billion iOS devices [35], and 1.2 billion in-home IoT devices [74], the ICS population is irrelevant to a large-scale attacker.

Additionally, many ICS devices are installed in a unique, physical system, and the response to a manipulation of a given ICS device would be difficult to predict. This is a concern for authors of parasitic malware, because unpredictable system response may

result in malware being identified and removed. This contrasts with many standalone IT and IoT assets with predicable and easily tested responses. For example, Mirai on a webcam is unlikely to affect the behaviour of any other device on the local network.

Outlook The fragmented nature and limited resources of the existing ICS population are overcome by the growing use of runtimes across manufacturers, the concentration of IIoT hardware and software amongst a limited number of vendors, and the increased hardware and software resources available from those IIoT platforms.

Devices with Linux-based or other widely used, open source operating systems provide the attacker with a large community of people looking for vulnerabilities and developing packaged exploits (e.g., Metasploit). Further, given the slow patch cadence of the Internet-connected ICS population, an attacker can expect vulnerabilities identified in white hat communities to remain exploitable for a long time. Additionally, many IIoT devices (e.g., sensors) are more self-contained than traditional ICS devices (e.g., PLCs), making it easier to predict the device and system response.

A growing population of increasingly homogeneous devices with common runtimes or open-source operating systems satisfies, in part or full, all the missing enablers from the *Vulnerable population* category of Table 3.6. Further, as ICS hardware and software converge with the IoT domain, ICS devices risk being swept up in large-scale attacks targeting the billions of non-industrial IoT devices.

3.6.2 Attacker incentives

Current situation Monetising an attack against ICS devices at scale is challenging for several reasons. First, many ICS devices have limited onboard resources, making them unattractive from a cryptomining perspective. Second, ICS devices are unlikely to store high-value data, such as purchase orders, personal details, or financial records, commonly used to support a ransomware campaign. Industrial organisations are generally well-equipped to handle device failures without significant impact to operations, and an encrypted device could simply be treated as a failed device [68].

While the potential financial benefit from exploiting a large population of ICS devices appears to be low, the cost to develop and deploy an exploit appears to be high. The direct cost of buying devices for development and testing may run to thousands of dollars. Further, actually developing a wormable ICS exploit has been shown to require substantial subject matter expertise and time [68, 105, 192].

Finally, there may be personal consequences for the attacker. Authorities have successfully identified and prosecuted those responsible for several cyber attacks against industrial systems [3, 12, 99, 191]. This contrasts sharply with non-industrial cybercrime: while there have been some high-profile cases [98, 107], existing infrastructure to identify and successfully prosecute offenders remains weak [7].

Overall, the high cost to develop exploits, uncertain payoff, and risk of prosecution make Internet-connected ICS devices unattractive targets, when compared with IT and IoT assets. Why spend months developing an exploit for hundreds of devices when off-the-shelf exploits exist for target populations of hundreds of thousands?

Outlook Larger compute and memory resources on IIoT devices make them more attractive hosts for parasitic malware, creating viable financial incentives. Additionally,

the full operating systems running on these devices and development board availability simplify malware development and testing. Similarly, these platforms make it easier to modify and test existing malware (e.g., Mirai), reducing uncertainty and limiting the cost to develop an exploit.

While attacks against ICS have been prosecuted in the past, many large-scale cybercrimes do not directly affect the host, and those responsible have not been aggressively pursued [7]. Cybercrime making use of ICS devices may similarly be low risk, provided the parasitic malware does not adversely affect the industrial process.

Devices with greater compute and memory resources and commodity operating systems satisfy several missing enablers in the *Attacker incentives* category of Table 3.6. Given the currently-limited pursuit of cybercriminals targeting IoT devices, we also consider the *low consequence to attacker* enabler to be partially met.

3.6.3 Attacker tools and resources

Current situation Wormable ICS malware has yet to be packaged for an unskilled attacker to deploy [68, 192]. Subject matter experts developing such proof-of-concept malware have demonstrated that such development is plausible, but admit it is not currently practical. In contrast, off-the-shelf tools exist for attacking IT and IoT at scale. Further, numerous examples show that an adversary interested in attacking industry at scale need not target ICS devices directly, as off-the-shelf malware for Windows-base infrastructure is sufficient to shut down industrial processes for extended periods [48, 71].

Outlook As discussed above, devices with Linux-based or other widely used, open source operating systems provide attackers with a much greater opportunity to develop or use existing exploits, which is exacerbated by the slow patch cadence demonstrated for Internet-connected ICS devices. The increased homogeneity and use of commodity operating systems simplifies the effort to develop or modify exploits targeting ICS devices, satisfying the remaining enablers in the *Attacker tools and resources* category of Table 3.6.

3.6.4 Summary

In answer to the questions posed at the beginning of this section, our model demonstrates:

1. The current Internet-connected ICS population is not a target for large-scale cybercrime due to its fragmented nature, the high cost to develop and test exploits, and the unpredictable monetisation and consequences of attack.

2. Our model predicts that adversarial interest in this population will grow with trends toward connectable devices; homogeneous hardware, software, and development environments; greater onboard resources; and commodity operating systems.

3.7 Ethical considerations

We obtained IRB approval for searching and processing CrimeBB data as well as scanning IP addresses suspected of hosting Mirai.

For CrimeBB, our primary concerns were violating user privacy (e.g., deanonymising identities) and loss of data control (e.g., allowing another organisation to use the data

in a way that violates the CCCC's legal framework [53]). To mitigate these concerns, we restricted our search terms, stored and processed the data on a specified server, and followed strict guidelines to avoid privacy violations, such as specifically avoiding correlations between different forums and avoiding direct quotations from posts.

For scanning, our primary concern was adverse physical effects on ICS (e.g., causing devices to reset or slow down [149]). To mitigate these concerns, we followed the recommended practices in Durumeric *et al.* [60] and used the ZGrab scanner,[4] the tool used by Censys [61, 136]. We reasoned that as Censys performs Internet-wide scanning on a near-daily basis against every IP address and port number we planned to scan, we would not cause any additional adverse effects. Where possible, we used Censys' data directly and only performed our own scans when better-than-daily resolution was necessary to eliminate uncertainty from DHCP churn. We used a superset of the blocklists maintained by Censys, the CCCC, and the OARC-DNS 'don't probe' list [152]. We also hosted a website at the scanner's IP address that provided an explanation of our research and contact information for individuals or organisations to lodge complaints or request to be added do our blocklist. We received no complaints or requests to be added to the blocklist.

3.8 Summary

In this chapter, we presented the first multi-year, longitudinal study of the Internet-connected ICS population. We demonstrated the ability to passively track thousands of ICS devices over many years, showing that the population is growing, that device owners rarely install software updates, and that most devices are continuously connected. Despite these vulnerabilities, we find that the fragmentation of the ICS community, the high cost to develop and test exploits, and the unpredictable monetisation and consequences of attacking ICS make them an unattractive target for the cybercrime community, especially given the continued vulnerability of the larger and more homogeneous IoT population. Our conclusions are supported by technical and criminological analyses of the cybercrime community, using honeypots, malware tracking, and hacker forums to demonstrate that there is currently little competence or interest in the ICS domain amongst cybercriminals.

To explain this limited interest, we introduced a security economics model for characterising and predicting large-scale adversarial interest in Internet-connected populations. We developed the model by studying successful, large-scale attacks and empirically verified it using our longitudinal study. The model provides a concise explanation for the apparent reluctance of cybercriminals to target ICS devices.

While the current ICS device population may not be an attractive target for large-scale attacks, we surveyed ongoing and expected changes to the industrial environment that will boost the number of connectable devices and will move industry toward more homogeneous hardware, software, and development environments, greater compute and memory resources, and commodity operating systems. Our model predicts that these changes bring the ICS community in line with other, targeted populations, such as IoT; therefore, it is reasonable to expect greater attention from the cybercrime community directed toward ICS. Further, even if ICS devices are not directly targeted, convergence with IoT creates a new risk of being swept up in large-scale attacks targeting the billions of non-industrial IoT devices.

[4] https://github.com/zmap/zgrab2

CHAPTER 4

Using global honeypot networks to detect targeted ICS attacks

Defending Industrial Control Systems (ICS) in the cyber domain is both helped and hindered by bespoke systems integrating heterogeneous devices for unique purposes. Because of this fragmentation, observed attacks against ICS have been targeted and skilled, making them difficult detect and profile. Further, the proprietary nature of most industrial software and the relatively low profile of industrial devices result in limited vulnerability hunting and disclosure [1, 151, 184]. For example, the most popular proprietary and open-source real-time operating systems (RTOSes), VxWorks and FreeRTOS, respectively, have a total of 66 entries in the National Vulnerability Database (NVD) at the time of writing, compared with over 3 200 records for Windows 10 and over 1 100 records for Ubuntu 18.04. Further, a majority of the identified vulnerabilities in VxWorks and FreeRTOS come from a small number of large disclosures, and both have at least one disclosure in the last two years with more than 10 vulnerabilities, including vulnerabilities leading to remote code execution and Denial of Service (DoS) attacks. In both cases, most vulnerabilities were related to memory safety and existed in the software for more than a decade. Because these RTOSes are highly configurable, it is hard to estimate the number of affected devices; however, it likely exceeds two billion [100, 184]. For comparison, the initial install target for Windows 10 was only one billion devices [133].

Even when vulnerabilities are identified, the industrial community demonstrates a strong resistance to patching, partly due to the high cost of regression testing and recertification by both the vendor and user [114]. Additionally, industrial networks have limited host-based security or logging opportunities, complicating forensics efforts. Even when forensics is possible, industrial network compromises are generally business-sensitive, so post-exploit forensics efforts rarely result in public disclosure of vulnerabilities, though ICS security companies often publish summary reports, such as those for Triton/Trisis [207]. Finally, few industrial protocols employ authentication or encryption; therefore, ICS devices will consider any well-formed packet to be valid, including those that request information or command changes of state [68], allowing malicious manipulation of device behaviour without actually exploiting any specific vulnerability. Together, these factors result in a vulnerable industrial environment and create unique security challenges.

Successful attacks against ICS have all targeted specific organisations and devices (e.g., Stuxnet [31], Triton/Trisis [207], CRASHOVERRIDE [113]) or have targeted vendors

directly [173]; therefore, unlike other domains where attacks are large-scale and indiscriminate, such as the Internet of Things (IoT) domain, there are few means for researchers to gather open-source intelligence on ICS attack methods, motivations, and campaigns. In domains such as IoT, honeypots have been effective tools to track and profile malicious behaviour [211], but they rely on either indiscriminate or easily deceived attackers, neither of which apply to current ICS adversaries. To date, the use of ICS honeypots for security research has been largely limited to monitoring Internet-wide scanning.

Despite these challenges, we show that a geographically distributed network of high-interaction ICS honeypots can be an effective tool for identifying and profiling new, targeted attacks against ICS devices. We make the following contributions in this chapter:

1. A description of the largest, high-interaction ICS honeypot study to date.
2. A discussion of multiple, new ICS exploits (zero days) identified by the honeypot network.
3. An assessment of the growing overlap between ICS and IoT-aware scanning and botnet infections.
4. An explanation of the limitations of previous ICS honeypot studies and recommendations for successful networks of ICS honeypots for security research.

This chapter is largely derived from a paper presented at the peer-reviewed *12th International Conference on Cyber Conflict (CyCon '20)* [57]. My co-author Mikael Vingaard, from Industrial Defenica, owns and maintains the honeypot network and provided the primary dataset used in this chapter. I performed the data processing, identified relevant attacks, and executed the comparison with other studies and datasets.

4.1 Background

4.1.1 Targeted ICS attacks

Most, if not all, successful attacks against ICS have been targeted, in that the attackers wish to create adverse physical effects in a specific organisation, and they knowledgeably targeted specific devices. Examples include Stuxnet attacks against Siemens PLCs [31]; Triton/Trisis attacks against specific models of Schneider Electric's Triconex Safety Instrumented System [207]; and CRASHOVERRIDE attacks against the Ukrainian power grid [38]. The targeted and highly resourced nature of these attacks complicates efforts to identify and track real-world ICS exploitation, as the number of attacks is limited, and attackers have the ability and motivation to limit their exposure. As a result, ICS honeypot studies to date have not identified any attempt to maliciously modify ICS behaviour nor have they been effectively used to disclose new ICS exploits to the community.

4.1.2 Large-scale ICS attacks

Researchers have demonstrated scalable, proof-of-concept malware for PLCs that modify programmable logic and automatically spread to other devices (e.g., to demand a ransom) [68, 192]. To date, no such large-scale, indiscriminate ICS malware has been observed in the

wild. Further, while a decade of security research has demonstrated that tens of thousands of vulnerable ICS devices are directly connected to the Internet [115, 136], there has been little evidence of malicious attempts to modify the behaviour of such devices.

The lack of criminal or other large-scale malicious interest in vulnerable ICS devices can be attributed to several economic factors, which we describe in detail in Chapter 3 and summarise here:

1. High cost of entry: The cost of hardware for development and testing and the time to gain sufficient knowledge and experience to exploit such devices are significantly higher than other domains (e.g., IoT).

2. Fragmented population: While there may be over 100 000 Internet-connected ICS devices, the population is divided amongst dozens of manufacturers running proprietary or baremetal software on different chipsets.

3. Limited resources: ICS devices have limited compute and memory resources making them poor hosts for resource-intensive tasks such as cryptomining, and they are unlikely to store sensitive information typically used in ransomware attacks. Limited resources and proprietary software make general computing malware unlikely to succeed on ICS devices.

These economic factors are changing as industry seeks new ways to use digital technology. Industry 4.0 and Industrial IoT (IIoT) are converging with the IoT domain [218], creating a larger, more homogeneous environment of low-cost devices with general purpose compute and memory resources. In short, these changes are expected to overcome the economic factors currently inhibiting large-scale malicious interest in the ICS domain. As IIoT and IoT converge and industrial environments become increasingly attractive to cybercriminals and others looking to exploit devices at scale, ICS honeypots will be effective tools to identify and profile these attacks, as they are currently within the IoT domain.

4.2 SecuriOT deception technology

4.2.1 ICS honeypots

Previous ICS honeypot studies had several limitations that reduced the likelihood of an attacker being deceived into interacting with the honeypot, such as geographic concentration, the use of cloud hosts, the use of low-interaction honeypots, and short study durations. In this paper we demonstrate that these limitations can be overcome, showing that a sufficiently sized, Internet-connected ICS honeypot network can be effective in detecting and monitoring previously unknown, targeted attacks.

4.2.2 SecuriOT honeypots

Low-interaction honeypots can be inexpensively deployed at scale, but they are easy to identify. Further, because they do not emulate device state, they cannot be used to profile an attacker's behaviour (e.g., attempts to modify programmable logic). High-interaction honeypots overcome these limitations, but can be expensive to develop, deploy, and maintain. To address the limitations of both low- and high-interaction honeypots,

SecuriOT developed a reconfigurable device that supports multiple interaction levels with a common interface and management framework [179]. The device can be configured with templates to emulate an ICS device for low-interaction contexts, like Conpot [169], but can also act as a proxy to a production device. When acting as a proxy, the honeypot redirects traffic to a production device and acts as a man-in-the-middle between the network and the device. This proxy mode allows an adversary to exercise the full behaviour of the target device while providing the honeypot's full logging and alert functionality.

As shown in Figure 4.1, each physical device is capable of hosting multiple virtual IP addresses and up to three templates simultaneously, allowing a single physical device to appear as multiple devices on a given network.

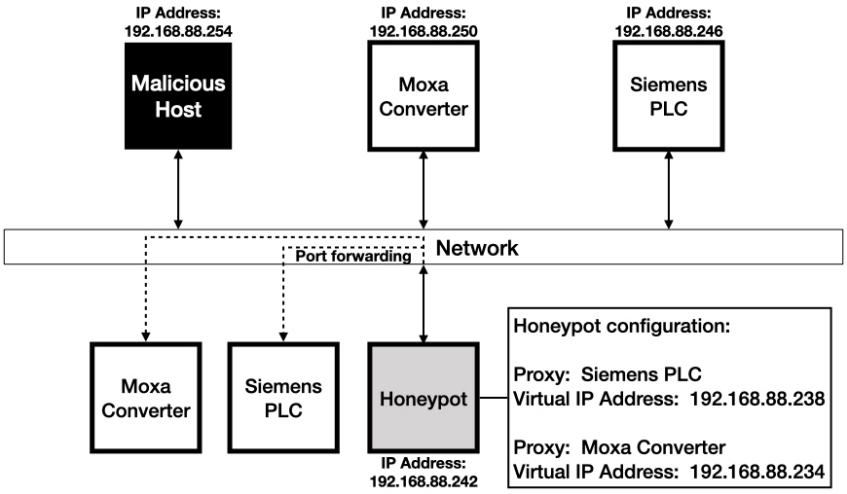

Figure 4.1: Sample deployment of a SecuriOT honeypot, showing the ability to emulate multiple virtual devices and act as a proxy for a custom device.

Each physical honeypot interfaces with a Security Information and Event Management (SIEM) system, which logs interactions and raises alerts. Since the honeypot is passive and has no production function on the network, any interaction with a virtual device is suspicious, as it implies a host is either scanning the network segment or directly interacting with the honeypot. The SIEM is also used to manage device configurations.

4.2.3 SecuriOT honeypot network

While SecuriOT's ICS honeypots are primarily designed for installation in production systems, the ability to act as a proxy and simultaneously support multiple virtual devices makes them a good foundation for a network of research honeypots. As part of their own intelligence-gathering operation, SecuriOT runs a network of 120 such virtual honeypots with IP addresses geolocated in over 20 countries. Each virtual IP routes traffic to a honeypot acting as a proxy to a production ICS device. Devices include PLCs, RTUs, and serial-to-Ethernet converters from vendors such as Siemens, Moxa, and Phoenix Contact. The virtual honeypots are supported by up to 15 production devices communicating over the following protocol/port combinations: S7comm/102, BACnet/47808, SOAP/37215, IEC-104/2404, DNP3/20000, and Modbus/502. S7comm, IEC-104, DNP3, and Modbus are used in several industrial environments, including manufacturing, automation, and power and water utilities. BACnet is used in large-scale building automation. SOAP on port 37215 is used for configuration and management of certain routers.

Table 4.1: Fields provided per packet from SecuriOT's honeypot network.

Field	Example	Field	Example
Date	2018-03-31	Source country	Japan
Time	06:33:49	Destination country	United States
Source IP address	[REDACTED]	Source AS number	AS63949
Source port	51667	Source AS name	Linode, LLC
Destination port	102	Scanning tool	ZMAP
Protocol	S7comm	Campaign	TA-VV
Packet action	Reconnaissance		

The honeypots perform full packet captures and SecuriOT performs post-processing, such as fingerprinting the tool used to interact with the honeypot (e.g., NMAP [150]), identifying campaigns, and classifying packets as either reconnaissance or exploitation. The result is a dataset with the fields shown in Table 4.1.

4.3 Data analysis and discussion

Our dataset consists of 13 months of packets captured between March 2018 and March 2019 from SecuriOT's network of 120, globally distributed, high-interaction ICS honeypots. The dataset consists of approximately 200 000 packets that we group into approximately 80 000 interactions. In this section, we present our analysis of the data and discuss our findings. We start with a dataset overview, including a comparison with previous, similar surveys. We then demonstrate malicious use of industrial protocols and discuss the relationships between attackers and targets. We conclude with a demonstration of large-scale attacks against non-industrial protocols recorded by the honeypot network and present early evidence that the ICS domain is affected by malicious, large-scale interest in IoT.

4.3.1 Dataset overview

Table 4.2 provides a summary of the interactions with the SecuriOT network of industrial honeypots over the period of observation. The data demonstrates the breadth of interest in Internet-connected ICS devices: thousands of individual hosts (IP addresses) are scanning industrial protocols from dozens of Autonomous Systems (ASes) in dozens of countries.

A majority of these interactions originate from well-known research scanners and are expected to be benign (e.g., Censys [30]), which is consistent with previous observations [29, 66, 136]. Both SecuriOT and the Cambridge Cybercrime Centre (CCCC) [53] maintain lists of known scanners against which source IP addresses were compared to generate the 'Known scanners' percentages in Table 4.2. Similarly, Table 4.2 shows that a vast majority of interactions are initiated by well-known scanning tools (e.g., NMAP [150]).

Following previous studies, we classify multiple received packets from a given IP address as part of a single 'interaction'. Comparing interactions rather than packets is preferable because the number of packets to perform a given task can vary for different scanning tools and protocols. We define an interaction as a single scanning or exploitation event. For example, the Siemens module from the ZGrab scanner sends about 12 packets to each scanned IP address, while scanning a single port with ZMap only sends 2 packets (TCP SYN and RST) to each scanned IP address [136, 204]. Each of these would be considered

Table 4.2: Summary of interactions with SecuriOT's network of industrial honeypots.

Protocol/Port	Total packets	Related interactions	Source IP addresses	Source ASes	Source countries	Known scanners	Known tools
Modbus/502	54 682	18 980	1 321	91	31	69.5%	99.9%
BACnet/47808	50 276	20 097	1 073	35	16	84.7%	100.0%
S7comm/102	43 203	18 422	998	85	30	49.9%	99.5%
DNP3/20000	32 534	13 283	1 040	124	42	39.1%	99.9%
SOAP/37215	12 975	7 403	337	85	29	0.0%	51.2%
IEC-104/2404	8 797	3 404	214	162	23	7.0%	99.6%

Table 4.3: Comparison of survey methods, types, and sizes.

Source	Method	Dataset type	Dataset size
SecuriOT	Honeypot	Packets	202 467
SecuriOT	Honeypot	Interactions	81 589
Mirian *et al.* [136]	Telescope	Packets	2 100
Mirian *et al.* [136]	Honeypot	Interactions	5 252
Ferretti *et al.* [66]	Honeypot	Packets	n/a[a]
Ferretti *et al.* [66]	Honeypot	Interactions	4 986
Cabana *et al.* [29]	Telescope	Packets	$197M$[b]

[a] Raw data not available.
[b] Estimated based on graphical data.

one interaction.

4.3.2 Comparison to earlier studies

Different studies use different methodologies and focus on different protocols; therefore, direct comparison is challenging. Even surveys covering the same timeframe, but using different methodologies, can produce different results (e.g., network telescopes versus honeypots [136]). We approach such comparisons with caution, and only draw qualitative conclusions. We selected studies for comparison for the following reasons: Mirian *et al.* is regularly used for comparison in other studies [136]; Ferretti *et al.* is a more recent study of similar size to Mirian *et al.* and has a global scope [66]; and Cabana *et al.* is the largest, low-interaction ICS honeypot study in the literature [29]. Notably, all three of these studies use low-interaction honeypots, whereas our study uses high interaction honeypots. There is not a comparable survey in the academic literature of a large-scale, high-interaction ICS honeypot network.

Table 4.3 compares these surveys and data collection methods, showing broad agreement in the observed scanning frequency against each protocol. Table 4.4 also demonstrates that ranking based on interactions results in a different ordering than ranking based on packets, as different protocols have different packet densities.

While the distribution of our scanning traffic is largely consistent with previous studies, the data shows both growth and asymmetry in the DNP3 scanning traffic that has not been previously identified or evaluated. Mirian *et al.* only identified 5.1% of network telescope traffic as targeting DNP3 in 2015 [136], whereas Cabana *et al.* observed over 22% of network telescope data targeting DNP3 in 2019 [29]. Similarly, over 16% of the

Table 4.4: Comparison of ranked popularity of scanned industrial protocols from multiple surveys.

Source	Dataset type	Ranked popularity				
SecuriOT	Packets	Modbus	BACnet	S7comm	DNP3	IEC-104
SecuriOT	Interactions	BACnet	Modbus	S7comm	DNP3	IEC-104
Mirian *et al.* [136]	Packets	Modbus	BACnet	S7comm	DNP3	Ethernet/IP
Mirian *et al.* [136]	Interactions	S7comm	Modbus	BACnet		
Ferretti *et al.* [66]	Packets	Modbus	BACnet	S7comm	Ethernet/IP	IEC-104
Ferretti *et al.* [66]	Interactions	BACnet	Modbus	Ethernet/IP	S7comm	IEC-104
Cabana *et al.* [29]	Packets	BACnet	Modbus	DNP3	S7comm	Ethernet/IP

Table 4.5: Attacks using industrial protocols. The packet count does not include transport layer handshakes (e.g., initial SYN packet for protocols layered on TCP).

Date	Source country	Destination country	Protocol	Attack type	Source AS number	Number of packets
2 Apr 2018	United States	China	IEC-104	DoS	AS394828	2
17 Apr 2018	China	Poland	IEC-104	DoS	AS4134	1
20 Apr 2018	Russia	United States	S7comm	Replay	AS60307	8
27 Jun 2018	Ukraine	China	IEC-104	DoS	AS15626	4
8 Aug 2018	Vietnam	France	S7comm	DoS	AS38731	1
8 Aug 2018	Vietnam	Lithuania	S7comm	DoS	AS38731	1
9 Aug 2018	Vietnam	Poland	Modbus	DoS	AS38731	1
9 Aug 2018	Vietnam	France	Modbus	DoS	AS38731	1
19 Nov 2018	Seychelles	Czech Republic	Modbus	DoS	AS29073	1

interactions recorded by SecuriOT honeypots targeted the DNP3 protocol. Further, as shown in Table 4.2, while the total number of DNP3 interactions is only 70% of the number of Modbus interactions (13 283 vs. 18 980), the number of IP addresses scanning for DNP3 is nearly 80% of that for Modbus (1 040 vs. 1 321), and the number of ASes from which those IP addresses originate is 136% of those for Modbus (124 vs 91). This statistic is also reflected in the number of source countries in which those IP addresses are geolocated (42 vs. 31). The asymmetry is even more pronounced when comparing DNP3 with BACnet or S7comm. Despite the challenges in quantitative comparisons between studies, there is clear evidence from multiple studies demonstrating a wider, as well as a growing, interest in DNP3 compared to other industrial protocols.

4.3.3 Targeted attacks via industrial protocols

SecuriOT concludes that only 20 of the 200 000 captured packets make use of an industrial protocol with clear malicious intent. These 20 packets can be grouped into nine attack interactions, which are summarised in Table 4.5. Based on feedback from vendors and vulnerability databases, four of the nine interactions represent previously unknown attacks, or zero days, and one represents the first documentation of a previously identified proof-of-concept attack in the wild [159]. The attack types include DoS and command replay.

The DoS attacks took several forms. In one case, a specially crafted packet forces a device to violate its real-time constraints, providing a low-bandwidth DoS attack on the process control. In another case, the attack targets devices with incomplete implementations of the protocol stack; the attack provides valid, but unimplemented

commands, and adversely affects the device's process control. The attacker specifically targeted vulnerable device types, so this is not a case of accidental DoS. In a third case, a buffer overflow affects the device's network communication capability, but does not affect the device's process control.

Since many industrial protocols lack authentication or encryption, the receipt of any packet with a parsable command may be considered valid. In some cases, though, manufacturers have implemented protection to prevent a replay of previous commands or commands recorded in a test environment. The replay attack identified by SecuriOT was successful against a device for which the manufacturer claimed replay protection.

For most of the attacks, the source IP address was only active for the attack itself; the honeypot network has no record of other interactions from that IP address. This is not unexpected: an attacker may use one (or multiple) IP addresses for reconnaissance and then use a fresh IP address for the actual attack to avoid blocklists. For three of the attacks, however, consistent activity was observed from the source IP address. Specifically, the IP addresses used for the attacks originating in Vietnam, Ukraine, and the Seychelles performed regular scanning over the entire study duration.

These vulnerabilities and associated exploits were responsibly disclosed by SecuriOT to the device manufacturers.

4.3.4 Large-scale attacks via industrial protocols

SecuriOT's honeypots also exposed a non-industrial protocol port and captured data associated with the Okiru-Satori variant of the Mirai botnet, which is indiscriminate and targets any vulnerable device across any network to which an infected device is connected.

While Okiru-Satori does not target industrial protocols, the convergence of IIoT and IoT domains may result in industrial devices being included in large-scale, non-industrial attacks. This is already the case for Windows-based industrial infrastructure. For example, the ransomware attack against the Windows-based infrastructure at Norsk Hydro in early 2019 prevented the safe and effective use of industrial devices [48]. As IIoT devices incorporate common operating systems with general purpose processing (e.g., Linux-based Azure Sphere [130]), they are more likely to become inadvertent victims of large-scale attacks targeting the IoT population. In this section, we discuss interactions with Mirai hosts and show that overlap already exists with industrial protocol scanners. Section 3.4.3 provides background on the Mirai botnet and the CCCC dataset of suspected Mirai hosts.

Many Mirai variants emerged after the public release of the Mirai source code. Variants target different device types and architectures and exploit different vulnerabilities. The Okiru-Satori variant was identified in 2017 and targeted Huawei routers on port 37215 using a previously unidentified vulnerability (CVE-2017-17215) [92]. As shown in Table 4.6, SecuriOT's honeypots recorded 7 403 interactions from 337 IP addresses on port 37215. Of these, SecuriOT identified 222 malicious interactions, based on attempts to brute force passwords, make use of the vulnerabilities exploited by Okiru-Satori, or modify firmware. While the malicious packets make up more than 30% of the total traffic on port 37215, only 3.0% of the total interactions are malicious, as password searches and firmware downloads necessarily require more packets than scanning.

Notably, while the scanning of port 37215 was recorded on 266 days, the honeypots were only configured as vulnerable routers over short periods in April and July 2018, resulting in only 15 days of malicious interactions. As discussed below, some of the apparently benign

Table 4.6: Summary of interactions on port 37215.

	Packets	Interactions	Source IP Addresses	Source ASes	Dates of interaction
Overall	12 975	7 403	337	85	266
Malicious	3 919	222	13	2	15

Table 4.7: SecuriOT honeypot data corresponding to source IP addresses and dates from the CCCC Mirai host dataset.

Protocol/Port	Total packets	Related interactions	Source IP addresses	Source ASes	Dates
SOAP/37215	792	789	26	11	40
DNP3/20000	116	71	4	2	4
BACnet/47808	2	2	1	1	1
Modbus/502	1	1	1	1	1

scanning might have transitioned to exploitation had the scanner found the honeypot in a vulnerable configuration.

To study the overlap between Mirai hosts and hosts aware of industrial protocols, we combined the CCCC database of suspected Mirai hosts [53] with SecuriOT's honeypot data, correlating source IP addresses and interaction dates. Table 4.7 summarises the results of this comparison from the perspective of the SecuriOT honeypots. For example, the first row should be interpreted as 792 packets received by SecuriOT honeypots on port 37215 from 26 IP addresses that the CCCC suspected to be hosting Mirai on the day of the interaction with the honeypot.

Comparing 789 SOAP/37215 interactions in Table 4.7 with 222 malicious interactions in Table 4.6 demonstrates that the CCCC suspects many of the benign interactions with SecuriOT honeypots to have originated from Mirai hosts that simply did not find the SecuriOT honeypot to be vulnerable. This is consistent with the knowledge that the SecuriOT honeypots were only configured as vulnerable routers during limited periods.

Table 4.7 also shows that the CCCC suspects 74 industrial protocol interactions (i.e., over DNP3, BACnet and Modbus) with SecuriOT honeypots to have originated from IP addresses hosting Mirai. As there is no known variant of Mirai that targets ICS devices, the scanning traffic by Mirai hosts against industrial protocols implies either that these scanners share an IP address with a Mirai host (e.g., a scanner behind an infected router) or that the scanner uses a similar technique to that employed by Mirai, though we are not aware of any such benign, Internet-wide scanners.

This overlap between SecuriOT's honeypot data and the CCCC Mirai database, though limited, suggests that the gap between ICS-aware and IoT-aware hosts is narrowing.

4.4 Recommendations for honeypot networks

SecuriOT's honeypot network exposed four zero day attacks against devices running common ICS protocols, such as S7comm and Modbus. By comparing our study with previous studies that did not identify similar exploits (e.g., [29, 66, 136]), we provide the following recommendations for deploying networks of ICS honeypots for security research:

1. Honeypot networks should be geographically dispersed. We identified nine attacks against devices in six countries, and none of the attacks originated in the same country as the target. Several honeypot studies located most or all targets in the United States [66, 136]; however, of our nine identified attacks, only one target was located in the United States.

2. Honeypots should be hosted at realistic IP addresses. Several previous ICS honeypot studies used Amazon Web Services (AWS) or other cloud providers to host honeypots [29, 66, 136]. ICS devices are unlikely to be connected via a cloud service provider, so the use of AWS or similar is a red flag to an attacker.

3. Honeypots should be high-interaction. Low-interaction honeypots can often be fingerprinted and generally do not allow an attacker to interact with the device beyond the initial login screen or protocol handshake. To deceive targeted attackers and understand their intentions (e.g., modifying firmware or programmable logic), high-interaction honeypots are necessary.

4. Honeypot use should be continuous. This provides both authenticity and a larger window for an attacker to identify and target a honeypot. Unlike large-scale attacks scanning for any vulnerable device, targeted attackers are looking for specific devices and it may take considerable time before accidentally targeting a honeypot.

4.5 Summary

We have demonstrated that a network of high-interaction honeypots can identify and profile previously unknown, targeted ICS attacks. Specifically, our honeypot network was used to expose four zero day attacks against devices running common ICS protocols such as S7comm and Modbus.

We also demonstrated that the gap between ICS-aware and IoT-aware hosts is narrowing, showing that IoT malware is co-located with ICS devices and scanners. Bridging this gap is the first major hurdle in attacking ICS devices at scale. Thus far, ICS devices have not been subjected to indiscriminate targeting, but the convergence of IIoT and IoT domains will make industrial devices more attractive targets, even if only as a vulnerable sub-population amongst the growing IoT population.

Finally, we discussed the limitations of previous honeypot studies and provided recommendations for developing effective honeypot networks as intelligence-gathering tools.

chapter 5

Access control challenges in distributed CPS

Cyber Physical Systems (CPS) is an umbrella term for a domain that tightly couples hardware and software with sensing and manipulation of the physical environment. CPS covers the natively-digital robotics and IoT domains, and increasingly includes Industrial Control Systems (ICS), vehicular control, and avionics, all of which are integrating digital devices into historically-analogue systems.

In Chapter 1 we described the evolution of CPS hardware, software, and communication technology. The practicalities of backfitting new technologies into brownfield deployments of safety- or life-critical systems often create conflicting incentives. For example, attempts to simplify backfits strongly incentivise maintaining existing, insecure protocols while simultaneously incentivising shifts from point-to-point serial connections to existing IP-based networks. Similarly, integrations of coordinating devices from different vendors that are likely to be replaced over time incentivise coarse-grained access control and disincentivise patching, due to the challenge of integrated regression testing [114].

Many industrial solutions attempt to side-step these challenges by securing the network boundary. For example, Siemens, a major vendor of ICS devices, recommends organising devices into 'automation cells': collecting ICS devices, sensors, and actuators performing a common task within a common boundary (see Figure 5.1) [188]. Security begins at the boundary, and communication within the cell assumes complete trust. While this is an improvement over fallacious 'air gap' security assumptions [115], it suffers from the same problem: any compromise of the boundary compromises the whole cell. An adversary within the cell is in full command and can read sensors and manipulate actuators. Inside the boundary, researchers are pursuing behaviour-based anomaly detection: inspecting network traffic and comparing it against expected traffic produced by a physics-based model of system behaviour [77]. While these systems make stronger assumptions (e.g., the adversary breaches the boundary), they are often as complex as the systems they are designed to protect and add to existing integration and support challenges.

These solutions are not robust security architectures for distributed systems, as they outsource privilege to boundary or monitoring devices that become single points of failure or even concentrated targets. This has been exploited in high-profile proof-of-concept automotive hacks [134] and real-world, nation-state attacks on critical infrastructure (e.g., CRASHOVERRIDE attacks on the Ukrainian power grid [38]). Similarly, lack of

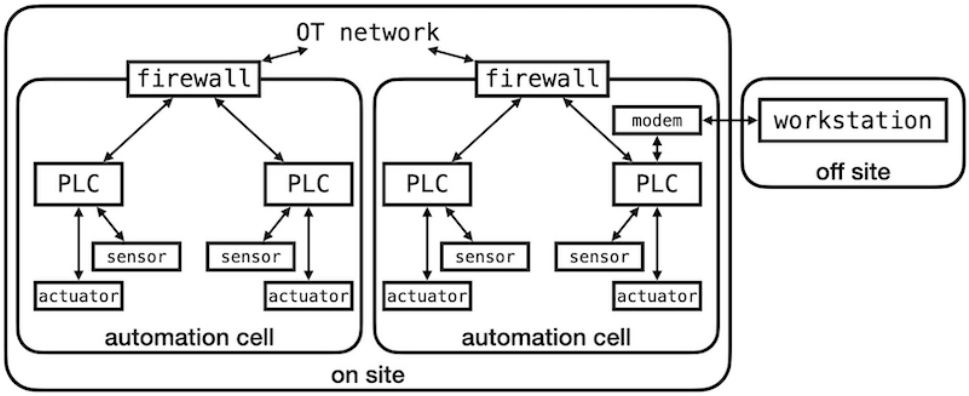

Figure 5.1: Example of an ICS 'automation cell' architecture [58].

compartmentalisation on the device (another boundary control problem) enabled proof-of-concept, remote code execution attacks against billions of embedded systems via popular Real-Time Operating System (RTOS) networking libraries [27, 168, 184], as well as real-world, nation-state attacks against industrial safety systems (e.g., TRITON/TRISIS attacks against the Schneider Electric Triconex Safety Instrumented System [207]).

Common Information Technology (IT) security paradigms, such as Access Control Lists (ACLs) and ringed protection methods, do not provide viable security alternatives, as many embedded systems lack hardware support (e.g., no Memory Management Unit (MMU)) and there is either no kernel to monitor access (e.g., baremetal systems) or the kernel is less trusted than the application. Further, these paradigms do not extend to the networked domain, as there is no network-wide monitor to regulate one device's access to another device's resources. Finally, these paradigms fail to enforce least privilege in the CPS domain: there is no straightforward way to distinguish between 'safe' unprivileged operations and 'unsafe' privileged operations, since the benign use and malicious use target the same operations (e.g., changing motor speed, manipulating valve position).

Shifting to per-device protections improves robustness and is a foundation for a distributed, compartmentalised design that tolerates partial compromise. However, this requires a distributed access control architecture and resilient individual hosts, neither of which is facilitated by a boundary control approach. Capability-based access control systems, on the other hand, inherently isolate and minimise privilege [135, 175]. Recent capability system research, and the availability of commodity hardware, provide new opportunities to solve CPS security challenges up and down the hierarchy with a capability-based architecture. Capabilities are unforgeable tokens that designate and convey access rights to an resource, and capability properties are such that holding a capability is a guarantee that a user has specific authorities to access the designated resource. This is particularly well suited for CPS' systems-of-systems designs, as capabilities provide a common mechanism for designating objects (including physical resources) that can be applied at the hardware, between modules, within cells, and across systems; that is, capabilities compose at every level of distributed CPS.

In this chapter, we motivate a security architecture based on distributed capabilities, creating a least privilege decomposition at every level of the CPS hierarchy. In Chapter 6 we show that object capabilities, representing physical resources, can be constructed, delegated, and used anywhere in a distributed CPS, and we provide a concrete implementation and evaluation of this design pattern.

The primary contribution of this chapter is a systematisation of CPS access control challenges and a demonstration that capabilities can uniquely meet these challenges with distributed policies and enforcement.

This chapter is largely derived from a paper that has been submitted to the peer-reviewed *IEEE Transactions on Information Forensics and Security*. I performed the systematisation of CPS access control challenges presented in that paper. My collaborators were responsible for other work in that paper, which is described in Chapter 6.

5.1 Threat model

Our threat model assumes colluding attackers on the network and the hardware. On the network, we assume a Dolev-Yao attacker who can eavesdrop and manipulate network traffic and interact with or impersonate both the resource owner and user [59]. The network attacker's primary goal is to successfully manipulate the behaviour of the resource-owning device by constructing, manipulating, or replaying legitimate requests from authorised resource users. The hardware attacker is a process running on the same hardware and sharing the same physical memory as the resource owner and/or user (e.g., a TCP/IP library). Both the network and hardware attackers are capability-aware; however, we assume the attacker does not initially hold a capability allowing it to create or derive its own valid capabilities for physical resources. This assumption is enforced by compartmentalisation and initial capability list distribution, as discussed in Section 6.1.1.

5.2 CPS access control challenges

In this section we systematise CPS access control challenges to motivate the utility of a capability-based security architecture for minimising privilege up and down the CPS hierarchy.

Challenge 1: *Memory management* Many CPS lack MMUs to support memory virtualisation and software-defined, hardware-enforced memory isolation (e.g., memory paging). This also precludes MMU-based compartmentalisation strategies. Therefore, CPS processes or tasks share a flat, physical memory space with one another and with the kernel (if present), allowing a compromise of one to affect all. Some CPS do contain Memory Protection Units (MPUs), which provide coarse-grained, hardware-enforced memory boundaries (usually between 8 and 16 regions). MPUs allow kernel space to be separated from user space, for example, but their limited number, static constraints, and performance costs preclude practical, software-defined compartmentalisation [34, 219]. This is a microcosm of the larger boundary control problem described in Section 2.1.1.

Challenge 2: *Kernel trust* In the IT domain, an operating system is responsible for protecting itself from a set of mutually-suspicious applications and for protecting those applications from one another. Embedded devices, like those in CPS, tend to either run baremetal, without any operating system, or they run a single application compiled into a monolithic binary with the kernel. For baremetal systems, there is no kernel available to mediate between processes and tasks. Kernel-enforced ACLs or hardware-enforced divisions between user and supervisor modes (where available, see the

Memory management challenge) might limit the damage from a malicious (or vulnerable and exploited) application; however, high-profile privilege escalation and code execution attacks against embedded systems usually originate in the kernel, or in libraries packaged with the kernel. For example, URGENT/11 and Ripple20 disclosed several remote code execution attacks originating in TCP/IP stacks packaged with embedded system kernels and together affected several billion devices [168, 184].

Challenge 3: *Privilege separation* Even where hardware or software support for separating privileged and unprivileged modes exists, the security guarantees provided to the CPS domain are weaker than those in the IT domain because routine CPS operations are the very behaviours an adversary will attempt to exploit. For example, a motor controller makes small adjustments to frequency during every control loop to maintain a constant rotor speed, and this is the very signal the adversary targets to modify the system behaviour. Privilege levels provide no security if routine behaviour of unprivileged applications control the 'unsafe' behaviour that the adversary is targeting. The only behaviour left to hide behind a privilege barrier is, potentially, the ability to modify programmable logic or firmware. While this precludes more persistent attacks, it is a weaker security position than ringed protection provides in the IT domain.

Further, if the application performs routine operations at higher privilege, the system may become a confused deputy [83, 84], executing commands from external sources (e.g., sensors and actuators) at a privilege higher than that held by the source. This also applies on the network: the reliance on highly-privileged boundary control devices rather than any host-based access control risks devices within the network interpreting commands passing through the boundary as having the same privilege as the boundary device.

Challenge 4: *Suspicious dependence* In the IT domain, mutually-suspicious applications generally share the same platform, but perform unrelated tasks. Where those tasks are related, the kernel mediates resource access. For networked applications, the client-server paradigm creates a level of trust between the client and the server and is suspicious of all other applications on the client or server machines and any entity on the network between the two. In contrast, CPS often integrate multiple devices from different suppliers performing distinct tasks to accomplish a common goal.

This is a worst-case scenario of mutual dependence and suspicion. For example, a common design pattern in ICS is the Device-Level Ring (DLR), which connects a controller, actuator, and sensor in a bi-directional, ringed network. Each device forwards all messages to the other devices along both directions of ring to ensure a disruption of a single link does not prevent sensory and control messages from reaching necessary recipients. Any device on the ring can manipulate their own information or the information they forward. The devices are critically dependent on one another, but have no inherent mechanism for trusting one another. There is no trustworthy network analogue of the kernel to mediate network interactions; though Intrusion Detection Systems (IDS) provide some mitigation, but add complexity and new attack vectors [77, 172].

This challenge extends to individual devices. Many Programmable Logic Controllers (PLCs) integrate a CPU module with additional modules (e.g., analogue I/O, network communications) on a common backplane. While these modules may have a common manufacturer, the modules themselves and the unencrypted backplane communication remain a source of suspicion that has generated a niche IDS industry [140, 209].

5.3 CPS access control solutions

In this section we compare possible solutions to the challenges in Section 5.2, evaluating hardware and software improvements, ACLs, and capability-based access control. For consistency, we use the convention from Miller *et al.* that a *subject invokes* a *resource* with some *authority* [135].

5.3.1 Improved hardware and software

The obvious solution to the *Memory management* and *Kernel trust* challenges is to improve the hardware and software, and this has been pursued by some vendors (e.g., Microsoft's Azure Sphere [130]); however, these improvements are not able to solve the *Privilege separation* and *Suspicious dependence* challenges on their own. Specifically, they do not provide a mechanism for reducing the authority of routine operations and cannot create trust between distributed devices within a system. For example, though the connection between an Azure Sphere device and Azure Hub may provide a mechanism for secure update and data transfer, it is mediated by a non-real-time operating system and is designed to connect an edge device to the cloud. It is not a real-time solution supporting system interconnections, such as those in a DLR.

While IT hardware and software solutions cannot comprehensively solve our challenges, they necessarily form part of any solution. We show in the next two sections that all practical solutions to the *Kernel trust* and *Privilege separation* challenges require *some* additional hardware (e.g., MMU, Instruction Set Architecture (ISA) extensions), and that all practical solutions to the *Privilege separation* challenge also require some piece of trusted software, though it may only be a bootloader or nanokernel.

5.3.2 ACLs and ringed protection

An ACL is a 'list of principals that are authorised to have access to some object' [175]. ACLs are the most common host-based access control mechanism, and are used on *NIX and Windows systems. The ACL is monitored by a gatekeeper (e.g., the kernel), which mediates a subject's invocation of a resource, and is controlled by a resource owner, who is authorised to modify the list of subjects and their authority for the resource. In this section, we discuss the extent to which ACLs can mitigate the four challenges and some additional problems presented by ACLs in the CPS domain.

Challenges 1 and 2: *Memory management* and *Kernel trust* ACLs do not provide, but rather rely on solutions to these challenges. A trusted kernel or other gatekeeper is required to mediate ACL-based invocation of a resource, which necessarily requires hardware-enforced isolation in the form of compartments or privilege levels to ensure subjects cannot arbitrarily modify ACLs.

Challenge 3: *Privilege separation* Assuming the *Memory management* and *Kernel trust* challenges can be met by improved hardware and software, ACLs controlling invocation of a CPS device's resources seem a natural mechanism for limiting authority, since they perform that function in most IT systems. In fact, CPS vendors recommend ACLs with Role-Based Access Control (RBAC) to minimise the burden and potential to

introduce errors as users come and go [15, 189, 206]. Despite this, ACLs cannot address the *Privilege separation* challenge, as they cannot reduce the authority required for routine tasks; therefore, even fine-grained designation of subjects and resources (a deviation from RBAC) would provide ineffective mitigation.

ACLs also require common namespaces to designate subjects and resources. That is, subjects must know what resources to designate, and ACL gatekeeper must know the authority of any subject. Managing namespaces is generally handled by the kernel [135]. In a networked CPS, however, there is no trusted manager to propagate changes to the set of subjects or resources to other devices. In many cases, propagating such changes is expensive or impossible, and is further complicated by mutual suspicion between devices (the *Suspicious dependence* challenge). For example, in a car relying on ACLs to manage interactions between Electronic Control Unites (ECUs), a vendor adding resources to an ECU to support new functionality must either create new subjects or allow the new resources to be accessible by existing subjects (i.e., existing users/groups). The latter minimises perturbations but unnecessarily expands authority, while the former requires propagation of new subjects to dozens of other ECUs. The design pressure is to avoid changes to authority by adding new resources to existing groups. In CPS, this is compounded by the strong encouragement to use RBAC [15, 189, 206], which aggregates users into coarse-grained roles and competes with the goal of minimising authority.

Challenge 4: *Suspicious dependence* ACLs cannot solve the *Suspicious dependence* challenge because they cannot convey authority across a network. Instead, as described in Section 2.4.2, tokens are used to convey an authenticated identity (e.g., a Kerberos ticket [196]), and an ACL is used to manage access to the resource at the host. This begins to anticipate our use of capabilities to convey authority, below, but requires a translation to an ACL at the resource-owning host, which end-to-end capability systems avoid.

5.3.3 Capability-based access control

In this section, we show that capabilities have the potential to address all four CPS security challenges described in Section 5.2; however, no existing system addresses them all. In Chapter 6, we discuss these limitations and introduce our end-to-end, capability-based security architecture for CPS.

Challenge 1: *Memory management* Capability systems provide opportunities to minimise authority without ringed protection and with a very limited trust boundary. For software-based systems (e.g., KeyKOS, seL4), trusted operations are limited to those specifically managing capabilities (e.g., creating, attenuating, destroying) [22, 104]. Protecting these capability operations ensures only the minimal authority is provided for subsequent invocations. Hardware/software co-designs take this further by handling capability operations with hardware instructions, eliminating the need for trusted software operations [145]. Further still, CheriOS, a single-address-space operating system, is a capability system that leverages Capability Hardware Enhanced RISC Instructions (CHERI) to provide efficient, fine-grained memory protection without an MMU [41].

Challenge 2: *Kernel trust* Capabilities provide a foundation for fine-grained access control without a trusted kernel. As above, the only entity which *must* be trustworthy is

that which performs capability operations. In a system where capabilities are software-enforced, such as seL4 or Fiasco.OC, this is the microkernel. In a hardware-enforced system, this is limited to the low-level code (e.g., bootloader).

Capabilities inextricably link the designation of a resource with the authority to invoke that resource; therefore, there is no need to maintain a shared namespace, normally a kernel responsibility, nor is the kernel able to invoke resources for which it does not hold a capability. In fact, the kernel is not even aware of resources for which it holds no capability, as there is no separate facility for designation. Capabilities also provide a simple foundation for baremetal access control, avoiding gatekeepers and namespace propagation (as required for ACLs) and minimising trusted code.

Capability systems also benefit from the use of access-controlled delegation channels, which confine the delegation of authority via capabilities to communication channels that are themselves invoked via capabilities. This is a necessary property of practical capability systems to solve the confinement problem [135] and is enforced by modern, capability-based operating systems (e.g., seL4 [86, 104]). This property allows explicit definition of on-device delegation channels without having to trust the kernel. On the network, however, the problem is more challenging and is discussed further below.

Challenge 3: *Privilege separation* Capability systems completely bypass the *Privilege separation* challenge because the privilege level (if any) at which a device is running to perform routine operations does not affect the authority used to execute a request. When a subject invokes a resource via a capability, the request will be executed with the authority provided by the capability, not the privilege level of the executor.

Put another way, capabilities prohibit ambient authority, which exists on systems where a subject does not select an authority as part of invoking a resource (e.g., UNIX filesystem). Such systems are susceptible to the confused deputy problem because a subject (the deputy) asked to perform an invocation on behalf of another will perform that invocation with the deputy's own authority, not the authority of the requesting subject [83]. Because capabilities bind the designation of a resource and the authority to invoke it, when a subject designates a resource, the subject also selects the authority by which it will invoke that resource. This extends to any deputised actions: the originating subject passes explicit authority to the deputy as part of designating the resource on which the deputy is to act. By explicitly selecting (and delegating) authority, the confused deputy problem is precluded and the need for privilege levels is eliminated (though systems such as seL4 may still use privilege separation to protect kernel operations on capabilities). By eliminating the need for privilege levels, we also reduce our dependence on supporting hardware (e.g., MMUs), mitigating the *Memory management* challenge.

Challenge 4: *Suspicious dependence* Capability systems mitigate the *Suspicious dependence* challenge by avoiding namespaces and explicitly selecting authority when designating a resource, allowing a user to invoke a resource anywhere on the network without a change in designation or access control method.

In an ACL system, subject and resource designators need to be propagated through the system, resulting in a design pressure toward coarse grained groupings and authority for those groups. In a capability system, this problem simply does not occur. Any device can add subjects or resources with arbitrarily-fine authority granularity, and the associated capabilities for the new resources only need to be propagated to devices that will use

them, fully minimising authority and defaulting to no authority. Using the earlier example, backfitting an upgraded ECU into an existing system that will make *no* use of the new resources is trivial, as it requires no propagation and grants no additional authority.

Capabilities also simplify reasoning about distributed authority. A subject either holds or does not hold a capability to invoke a resource elsewhere in the system, and that capability can be verified, delegated, transmitted, and revoked. We can effectively ignore both network and local transmission when reasoning about authority. Further, authority can be delegated to arbitrary depth without compromising either the explicit nature of capabilities or the ability to reason about authority between subjects and resources.

Most reference architectures and vendors already convey identity across CPS networks with capability-like tokens (see Section 6.4), so the infrastructure and philosophy are already in place to support distributed capabilities. We implement a Kerberos-like mechanism for token distribution in our first case study (Section 6.3.1).

5.4 Summary

In this chapter, we identified four unique security challenges faced by the CPS domain: limited memory management hardware, the lack of a trusted kernel, the inability to separate privilege, and the mutual dependence and suspicion of networked CPS devices.

With a trustworthy kernel and hardware-enforced memory isolation, ACLs go some way to confining authority on a device; however, they cannot address the confused deputy problem or convey authority across a distributed CPS. In other words, assuming the *Memory management* and *Kernel privilege* challenges are solved, ACLs still cannot address the *Privilege separation* and *Suspicious dependence* challenges.

Capabilities present a stronger foundation for CPS access control. They provide the building blocks for protecting resource invocation both locally and across a network without memory management hardware or kernel intervention, addressing the *Memory management* and *Kernel trust* challenges. Capabilities also have the potential to fully address the *Privilege separation* challenge, and they go a long way to addressing the *Suspicious dependence* challenge, though some additional mitigations are necessary to enforce confinement across the network (e.g., expiration or rotation to limit delegation). Capabilities enforce true minimisation of privilege to protect against both CPS device and network breaches, and they compose well at every level of a distributed network hierarchy, from an individual device module to a system-of-systems.

In Chapter 6 we implement an end-to-end capability system for distributed CPS and show that such as system is performant and that it can be implemented with minimal modification to existing software.

CHAPTER 6

A DISTRIBUTED CAPABILITY ARCHITECTURE FOR CPS

In Chapter 5 we demonstrated that capabilities have the potential to solve many of the unique access control challenges faced by Cyber Physical Systems (CPS), and that other access control solutions are inadequate. In this chapter, we propose and implement an end-to-end CPS security architecture based on distributed capabilities, creating a least privilege decomposition at every level of the CPS hierarchy, which tolerates both network and device compromise (see Figure 6.1). We show that object capabilities, representing physical resource attributes, can be constructed, delegated, and used anywhere in a distributed CPS. We compose architectural and cryptographic capabilities, providing hardware-enforced access control to objects on a device and cryptographic protections across the network, guaranteeing unforgeability, provenance, and monotonicity of capabilities at every level of the hierarchy. We also show that these object capabilities allows CPS to leap-frog use of Memory Management Units (MMUs), providing software-defined access control at a finer granularity than is possible with an MMU, while simplifying reasoning about authority on a device and across a networked CPS.

We evaluate two concrete implementations that use Capability Hardware Enhanced RISC Instruction (CHERI) architectural capabilities and Macaroon cryptographic capabilities. The first is a Robot Operating System (ROS)-based application and the second is an industrial controller built on a Real-Time Operating System (RTOS) (specifically, FreeRTOS). We show that our architecture adds negligible overhead (single digit µsec) to realistic, real-time integrations and that our design pattern can be implemented without significant modification to existing source code.

The primary contributions of this chapter are:

1. a novel approach for composing local architectural capabilities and distributed cryptographic capabilities to allow uniform designation and consistent enforcement of access control to CPS physical resources, and

2. a concrete demonstration and security evaluation of our design pattern, showing that a distributed CPS with a capability-based security architecture is performant and addresses CPS security challenges with minimal perturbation to application code.

This chapter is largely derived from two papers:

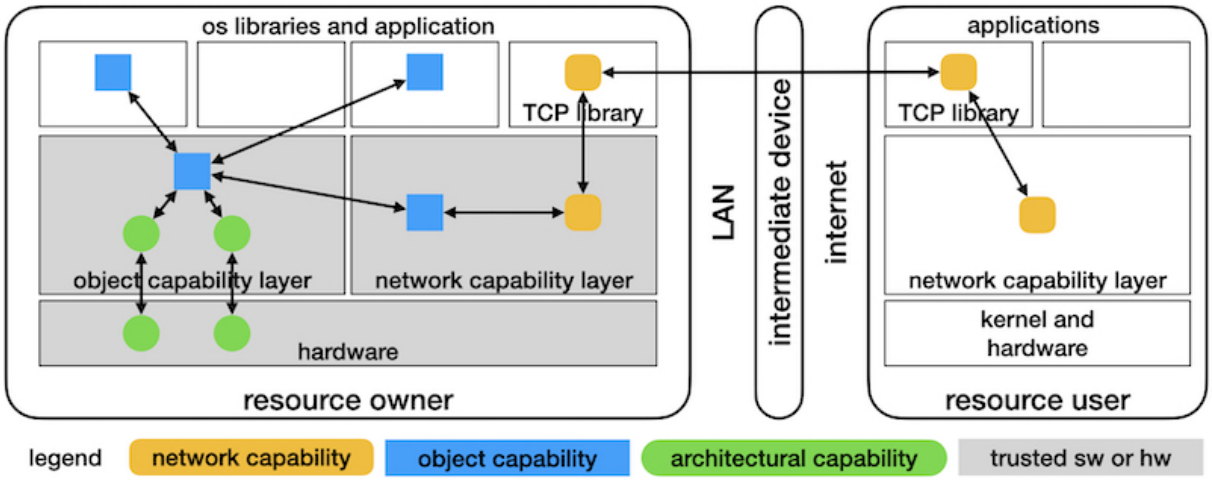

Figure 6.1: End-to-end design pattern for a distributed CPS capability system. This parallels the 'automation cell' architecture in Figure 5.1, where the resource owner is the PLC, the user is the workstation, the intermediate device is the modem or firewall, and each of these can issue commands to manipulate resources. In the capability architecture, only the shaded entities can create or modify capabilities to manipulate resources.

1. The first was presented at the peer-reviewed *IEEE European Symposium on Security and Privacy Workshops (EuroS&PW '20)* [58]. My co-authors Alexander Richardson and Jessica Clarke helped immeasurably with porting ROS to run on CheriBSD, which required changes to ROS libraries, CheriBSD, and parts of the CHERI toolchain. I conceived of the idea to use capabilities to represent physical resource attributes and developed the application to demonstrate the concept.

2. The second has been submitted to *IEEE Transactions on Information Forensics and Security* and is in peer review. My co-author Hesham Almatary was responsible for porting the FreeRTOS library operating system to CHERI (CheriFreeRTOS) and supported the evaluation of the Modbus demonstration. He also architected the CHERI compartmentalisation framework for CheriFreeRTOS. I developed the idea of using architectural capabilities to construct complex object capabilities, and I developed the application to test and evaluate these mappings.

6.1 Implementation

CPS are multi-user, multi-device systems with networked, peer-to-peer, and hierarchical relationships. Therefore, an end-to-end, capability-based access control system for CPS must locally bind capabilities to physical resources and be able to delegate authority by passing those capabilities across networks.

In this section, we show that such a design pattern can be implemented by deriving software-defined object capabilities from CHERI architectural capabilities [215], and tightly coupling these object capabilities with Macaroon cryptographic capabilities [20] to provide hardware-enforced access to physical resources in networked CPS (see Figure 6.2).

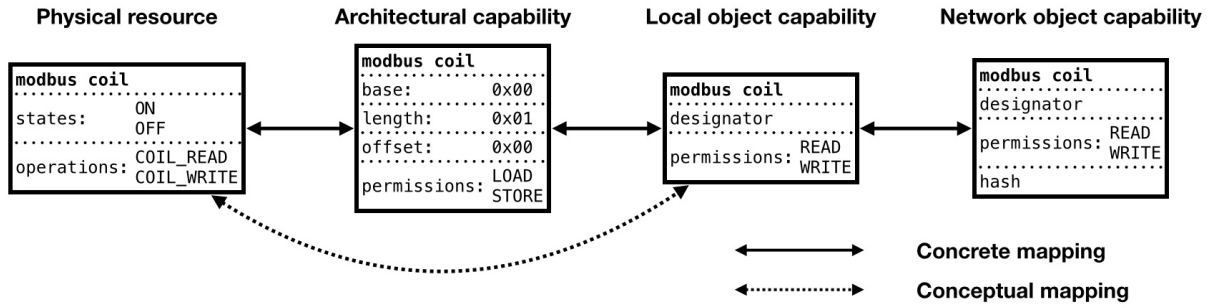

Figure 6.2: Relationship between a physical resource and its capabilities. Conceptually, the physical resource is represented by a single object capability; concretely, that object capability is constructed from one or more CHERI capabilities. The local and network object capabilities map in a 1:1 relationship.

6.1.1 CHERI: Local capabilities for CPS

Early capability systems were hardware/software co-designs (e.g., CAL-TSS, CAP [116]). Such co-designs continue in modern research systems (e.g., M^3 [11], SemperOS [89]); however, modern systems with significant usage outside academia use kernel-enforced, software-defined capabilities (e.g., seL4 [104], Fiasco.OC [110]), but rely on MMUs.

In CPS, the lack of hardware-enforced memory protection (the *Memory management* challenge) explicitly precludes direct use of existing capability systems (e.g., seL4). Combined with the general lack of a trusted kernel (the *Kernel trust* challenge), many CPS provide a weak foundation on which to build a local capability system. Specifically, if all tasks on a device share the same physical memory, and the kernel cannot be trusted to mediate access to capabilities, there is no way to prevent a compromised task from acquiring and using a capability held by another task.

We overcome these limitations with CHERI architectural capabilities (see Section 2.3.1). We leverage CHERI capabilities in three ways: as primitive memory object capabilities, as the basis for software-defined compartments, and as the basis for software-defined object capabilities. As primitive memory object capabilities, CHERI provides spatial memory safety for all pointers, and can generally be implemented by recompiling existing code with the CHERI LLVM-based toolchain, though modifications may be required to custom memory allocators [50, 167]. CHERI capabilities are distributed by low-level initialisation code (e.g., bootloader), supporting baremetal systems or those without a trusted kernel, helping to mitigate the *Kernel trust* challenge. We discuss CHERI compartmentalisation and object capabilities further below.

CHERI-based compartmentalisation We propose and implement a CHERI-based compartmentalisation framework in FreeRTOS as a lightweight, capability-based security framework to compartmentalise untrusted components from critical ones in a system without an MMU.

Each compartment is a separate protection domain. Like other capability-based systems, each compartment has a list of capabilities (c-list). A c-list forms the protection domain of each compartment and strictly defines its authorities to access its own resources and those in other compartments. This mechanism supports rapid inter-compartment communication without expensive, kernel-mediated context switches.

The current design provides isolation between C-based functions, objects, and files.

Isolation is guaranteed by the hardware, which ensures that each compartment can only access data and functions through its c-list. A system designer defines the files constituting a component at build time and the run-time framework automatically generates a c-list for each component. C-lists are assigned during the run-time loading process of compartments (by a separate capability-aware loader library) and do not rely on a trusted kernel.

Attempts to access data or functions outside a compartment's c-list trigger a protection domain switch, supporting default or custom handlers to restore system operation. The implementation discussed below catches these signals and makes compartment-specific recovery decisions, such as saving state and restarting a critical task, or shutting down and isolating a compromised TCP/IP library to allow a critical task to continue in an uninterrupted, if degraded, state.

Software-defined object capabilities CHERI architectural capabilities can also be used to construct software-defined, object capabilities with hardware-enforced access control. This is a new use for CHERI capabilities.

In capability-based operating systems, the kernel allocates and mediates access to capabilities [86, 116]. This is not feasible in many CPS due to the *Memory management* and *Kernel trust* challenges; therefore, we implement object capabilities as an independent library that can be isolated from the kernel and the application via CHERI compartmentalisation. The library maps abstract objects to one or more CHERI capabilities, is lightweight and application-specific, and can be implemented using a shim layer to minimise or eliminate changes to existing code.

Traditional capability systems have intrinsic object types and the ability to create new object types [104, 116]. In such systems, a capability pointing to an object of a given type is *sealed* by a special capability specific to that type. Sealed capabilities are tokens that both designate and confer the authority to invoke an object, but do not provide direct access to that object. Only the type manager or kernel can unseal that capability and access the underlying object.

CHERI Instruction Set Architecture (ISA) extensions support sealing and unsealing capabilities, allowing the creation of a similar type system for mediating access to objects of a given type; however, for the simple state elements and operations in most CPS, such expressiveness is unnecessary, and CHERI capabilities can be used directly to represent CPS objects. For example, CPS often have state elements representing physical objects that can be `ENERGISED` or `DEENERGISED` (e.g., a solenoid coil). A coil can be represented by a single CHERI capability designating a one-byte allocation with only `LOAD` and `STORE` permissions. To limit the object capability such that the holder can only `READ` the coil state, the CHERI capability permissions can be reduced to `LOAD`, and the hardware will prevent any holder of that object capability from modifying the coil's state.

To minimise changes to existing applications, we implement a shim layer that replaces permissive pointers with limited object capabilities. For example, in the Modbus demonstration detailed below, the `libmodbus` Application Programming Interface (API) requires the Modbus server (the resource owner) to provide `libmodbus` access to the entire system state (coils, discrete inputs, and registers) for any operation on any object, such as changing a coil from `DEENERGISED` to `ENERGISED`. The shim layer replaces that state object with the CHERI capability designating the single coil and the permitted operations on that coil. When `libmodbus` attempts to change coil state, the CHERI-aware hardware ensures that it can only perform the permitted operations on the limited memory designated by

the capability. The shim layer allows existing applications and libraries to be used with object capabilities at the cost of additional indirection.

The simplicity of these shim layers relies on the observation that operations on CPS objects map cleanly to existing CHERI permissions, allowing the hardware to act as the type manager. Most ICS protocols, such as Modbus, DNP3, and S7comm, roughly map to reading or writing simple state objects, allowing our design pattern to be generalised; therefore, modifying our Modbus shim layer for DNP3 or S7comm would affect fewer than 200 lines of code that map the protocol functions to CHERI functions and protocol objects to memory objects.

6.1.2 Composing local and network capabilities

Like other object capability systems, the protection of our local objects is confined to an individual device with hardware to enforce protection (e.g., CHERI ISA-extensions, MMU for seL4). To address the *Suspicious dependence* challenge we need a mechanism to convey local object capabilities between resource-owning and resource-using devices. Macaroons, described in Section 2.4.2, are well suited to our capability design pattern (and to CPS generally) because they present limited cryptographic and key management burdens on the CPS device and because of their semantic similarity to CHERI capabilities [20, 178]. In this chapter, we demonstrate an efficient mapping between CHERI capabilities, software-defined object capabilities, and Macaroon cryptographic capabilities.

CHERI-based, local object capabilities and Macaroon-based, network object capabilities have semantic similarities, supporting an efficient mapping back and forth (see Figure 6.3). The use of semantically-similar capabilities also simplifies reasoning about access control across a distributed CPS. This contrasts with ACL-based systems, where the designation and authority mechanism fundamentally changes at the interface between the network and device. Further, as many CPS communication graphs are static and known at design-time, we can bound the cost of executing the mapping.

Facilities and procedures for generation, distribution, and management of cryptographic capabilities, including manually loading tokens in edge devices, already exist in most CPS reference architectures, and can be implemented at low cost, even in small, incremental deployments (see Section 6.4); therefore, the infrastructure is already in place to make the practical step from a local to a fully distributed CPS capability system.

Capability violations (e.g., attempts to access an object without a valid capability) are handled with application-defined, custom handlers. For architectural and object capabilities, this mechanism is described in Section 6.1.1. For network capabilities, this is handled by the network capability layer. Custom handlers support graceful degradation at each boundary of both the network and the device. For example, the handler can drop attempts by the TCP/IP library to transmit an invalid command to the critical control loop. However, after a threshold of consecutive, invalid requests, the handler can consider the library (or an upstream device) to be compromised and isolate it from the control loop, allowing the critical task to continue, though possibly in a degraded condition.

6.2 Security evaluation

Given the implementation of our design pattern, we can begin to assess its security properties based on the threat model presented in Section 5.1. For the network attacker

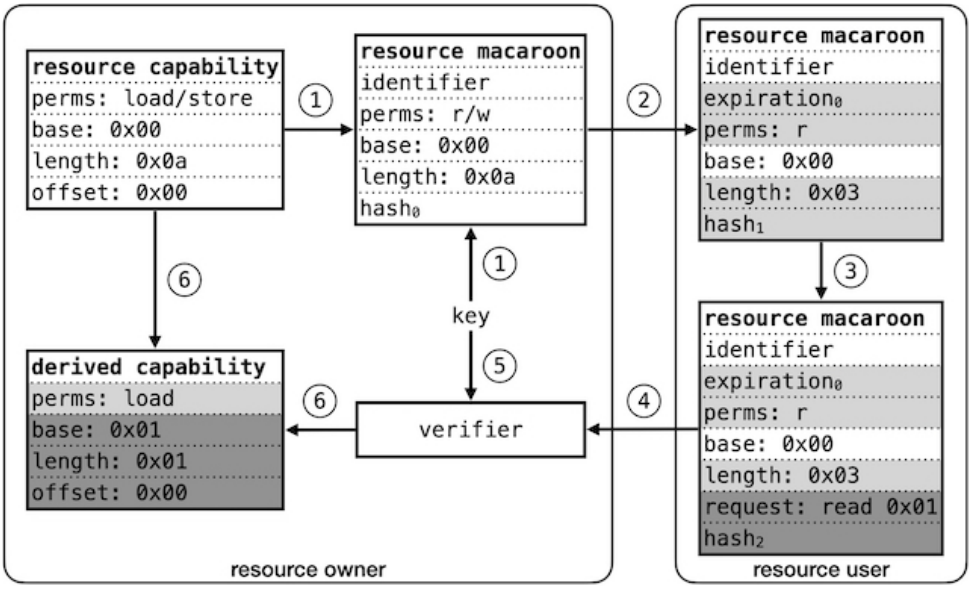

Figure 6.3: Composing distributed capabilities. The CHERI-based capability is mapped to a Macaroon-based capability (1). The Macaroon is provided to the resource user (2), who caveats the Macaroon with a request (3), and returns it to the owner (4). The owner verifies the Macaroon (5), derives an attenuated local capability (6), and invokes the resource using the derived capability.

targeting Macaroon-based network capabilities and the on-device attacker targeting either Macaroons or CHERI-based object capabilities, we consider Denial-of-Service (DoS), integrity attacks (e.g., expanding capability permissions), and replay.

A network attacker between the resource user and owner or an on-device attacker targeting Macaroons can initiate a DoS by sinkholing communication or corrupting data. Corruption is detected by the resource user via failed Macaroon verification. The attacker does not have the key that initiated the HMAC chain; therefore, the attacker intercepting a Macaroon cannot successfully remove caveats or generate a new, valid Macaroon granting more authority. Similarly, the attacker cannot modify the contents of a Macaroon in transit to change a caveat or the command included in the Macaroon. Any attempt to modify or generate a new Macaroon will be identified by a failed verification. Replay attacks can be limited by adding expiration caveats or by including sequence numbers as caveats, neither of which can be modified by the attacker.

An on-device attacker can perform a DoS, of sorts, by attempting an illegal operation on a CHERI capability (e.g., attempting to expand permissions), which will clear that capability's tag and prevent subsequent dereferencing. Attempts to dereference a capability with a cleared tag result in a signal from the hardware, which is caught by the kernel or a custom handler. CHERI provides no mechanism to prevent replay; therefore, if an attacker is granted valid capabilities during operation, the attacker can replay previous, authorised invocations of a physical resource. We limit this by restricting capabilities to the minimal authorities and address range necessary to execute any invocation before the capability leaves the security layer. We can also leverage capability rotation to limit the time available to replay collected capabilities. Unlike the network scenario, CHERI compartmentalisation prevents any snooping, however, so the attacker would have to be explicitly granted capabilities for later reuse.

In summary, composing capabilities prevents a network attacker and significantly limits a co-located process from manipulating any resource protected by a capability. Macaroons ensure the provenance and integrity of the capability on the network, and CHERI ensures the provenance and integrity of the capability on the hardware; therefore, an adversary cannot forge, modify, or effectively reuse a capability captured on the network, and the adversary is limited to reusing the minimally-permissioned capabilities explicitly granted to them on the hardware. Further, CHERI architectural capabilities provide a class fix for spatial memory safety vulnerabilities that may be introduced by the use of languages such as C/C++, which are common in the CPS domain.

In this evaluation, we do not assess the mechanism or protocol used to transfer the initial Macaroon between the resource owner and resource user; we assume that the transfer is performed during provisioning such that the network attacker cannot observe it, and that the material is stored on each device such that the hardware attacker cannot access it at rest. This is consistent with reference architectures and industrial provisioning guidelines, as discussed in Section 6.4.

6.3 Case studies

To explore the implementation and security implications of an end-to-end capability architecture, we perform two case studies. The first case study demonstrates the feasibility of our architecture in a widely-used robotics framework. The second is a detailed case study of a real-time industrial control application using a lightweight industrial protocol.

6.3.1 ROS: Proof-of-concept

ROS is a library framework for developing and integrating robotics applications, and our first case study is based on its second generation (ROS2) [171]. The core ROS library is written in C, with a C++ API library. ROS encapsulates functionality in 'nodes' which communicate with one another as peers using a message-passing, publish/subscribe protocol. For this case study, we implement a two-node derivative of TurtleBot (a canonical ROS example and demonstrator) [170]. The resource owner is a `motor control` node, which encapsulates independent control of the two wheels, thereby controlling robot speed and direction. The resource user is a `user input` node, which encapsulates obtaining and transmitting user input to control the robot. Communication between the two nodes is via local WiFi. At the time of writing, ROS does not have a standard facility for encrypting communication between nodes. Figure 6.4 shows the architecture and the interaction between principals.

Our application runs on a CHERI-aware version of FreeBSD (CheriBSD) [40]. ROS' primary target operating system is Ubuntu; therefore, some modifications to ROS libraries were required to support FreeBSD (all changes have been up-streamed), but no changes were required to support CHERI. We based our Macaroons implementation on `libmacaroons`, a C library implementing all Macaroons functionality.

In the unmodified TurtleBot, the `user input` node sends a message with a linear and angular velocity to the `motor control` node, which checks an md5 hash of the message and executes the command. There is no direct communication from the `motor control` node back to the `user input` node. The user relies on either visual or Simultaneous

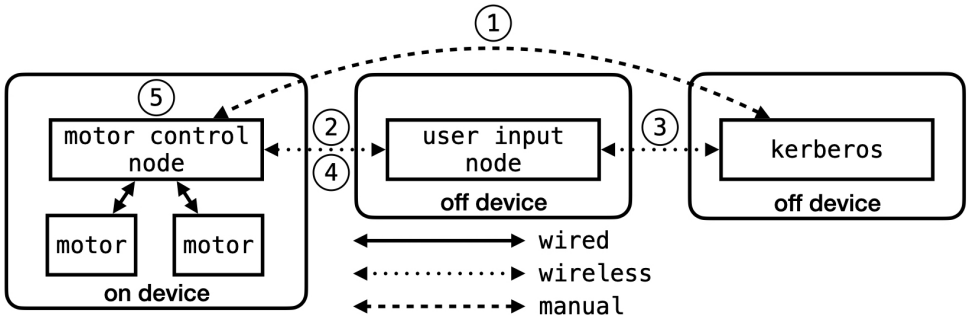

Figure 6.4: Architecture of the ROS case study. During provisioning, the `motor control` node shares a secret with Kerberos (1) to construct third-party Macaroons. At runtime, the `motor control` node generates and shares resource Macaroons (2). The `user input` node authenticates with Kerberos and receives a third-party Macaroon (3). The `user input` node caveats the resource Macaroon with the requested action, binds it to the third-party Macaroon, and returns both to the `motor control` node (4). Finally, the `motor control` node validates the Macaroons, derives a restricted CHERI capability, and executes the request (5).

Localisation and Mapping (SLAM) input to close the control loop. The motors are the resource for which access is required.

To simplify comparison with the unmodified implementation, we implemented one object capability for linear speed and one for angular speed to control these resources, rather than implementing one for each motor. The `motor control` node (the resource owner) derives a CHERI capability and Macaroon for each resource. The Macaroons can be openly shared with potential resource users, as they can only be used after authentication with the Kerberos server.

During provisioning, the `motor control` node shares a secret with the Kerberos server, which derives a third-party Macaroon. When the `user input` node authenticates with Kerberos, it receives this third-party Macaroon, which is caveated with an expiration. The `user input` node already has the resource Macaroons, and when ready to issue a command it 1) adds the desired linear or angular velocity as a caveat to the resource Macaroon, 2) binds the third-party Macaroon to the resource Macaroon, and 3) returns the third-party and resource Macaroons to the `motor control` node. The `motor control` node recursively verifies the received Macaroons (ensuring the integrity of the request and the authenticity of the user), uses the resource Macaroon to derive a restricted CHERI capability, and executes the request. CHERI hardware enforcement prevents attempts to access a different resource or expand permissions on a given resource.

We tested the proof-of-concept on the CHERI-MIPS CPU running on a Terasic DE4 FPGA [39]. The CHERI-MIPS CPU is a proof-of-concept processor for the CHERI-MIPS ISA, and it includes counters for fine-grained performance measurements, such as CPU cycle counts, instructions executed, and L2 cache misses. We focused on cycle count as an initial performance measurement, and we evaluated the cycles required to execute a command from the `user input` node to the `motor control` node. Our initial measurements demonstrated that CheriBSD and ROS, particularly ROS' publish/subscribe protocol and associated hashing of all messages, hid any overhead from the use of CHERI-based object capabilities and Macaroon-based cryptographic capabilities. While this was a positive observation for the practicality of the design pattern, it limited the benefit of performing a more detailed performance evaluation and encouraged us toward a more constrained case study.

In summary, our ROS proof-of-concept demonstrated that our design pattern could integrate into CPS frameworks without modifying existing code. It also demonstrated that our design pattern could be incorporated into many CPS applications without performance impact, specifically those running on commodity operating systems without real-time constraints. Other CPS applications are far more constrained, with strict execution timing and protocols transmitting only a few bytes per communication; therefore, to demonstrate our end-to-end capability design pattern is performant across the CPS domain, we performed a second case study with a lightweight, real-time application.

6.3.2 FreeRTOS + Modbus: Full case study

Modbus is a messaging protocol for client/server communication in critical infrastructure (particularly electrical generation and distribution). The protocol is an open standard developed by Modcon in 1979 and maintained by the Modbus Organization. It is supported by hundreds of manufacturers and at least 1,500 industrial devices, including PLCs, Remote Terminal Units (RTUs), and gateways [203]. The device is the Modbus 'server' and maintains state, while the 'client' is a computer, Human-Machine Interface (HMI), or other device that sends requests to read or change state. Modbus has three primary state abstractions: coils, discrete inputs, and registers. Coils are single bits that generally correspond to some form of actuation (e.g., turning on a motor). Discrete inputs are also a single bit, can only be read by the client, and generally correspond to some external input to the server (e.g., the status of a switch). Holding registers are 16 bits and can be read or written by the client; they can be used for passing digital data, such as changing server settings. Input registers are also 16 bits, but can only be read by the client. Modbus was originally designed for RS485 serial communication (Modbus RTU) [137], but has been extended for use over Ethernet, and specifically for TCP transport (Modbus TCP) [138].

We based our implementation on `libmodbus`, a C library implementing all Modbus functionality [165]. Our client and server applications exercise the entire range of Modbus functions. `libmodbus` handles both state manipulation and communication between client and server, as shown in Figure 6.5. Modbus does not provide any security features, so confidentiality, integrity, and availability can all be trivially compromised by an attacker on the network. Additionally, `libmodbus` initially generates and subsequently receives the pointer to the full state of the server during each request/reply interaction, allowing `libmodbus` to make arbitrary changes to the server's state.

In this section we document the detailed implementation of our design pattern and its effect on system performance.

6.3.2.1 Implementation

We implemented and tested our Modbus demonstration on CheriFreeRTOS [44], a CHERI-aware version of FreeRTOS [69]. `libmodbus` and `libmacaroons` did not require any modifications to compile for CHERI, but did require minor modifications to run on FreeRTOS. We implemented object and network capabilities as two security layers between the application and `libmodbus` (see Figure 6.1). These shims were designed to minimise modifications to either the application or the libraries.

The object capability layer mediates access to software-defined objects between the application and `libmodbus`, ensuring `libmodbus` only receives a capability designating the specific object and authorities necessary to execute a given request. As discussed in

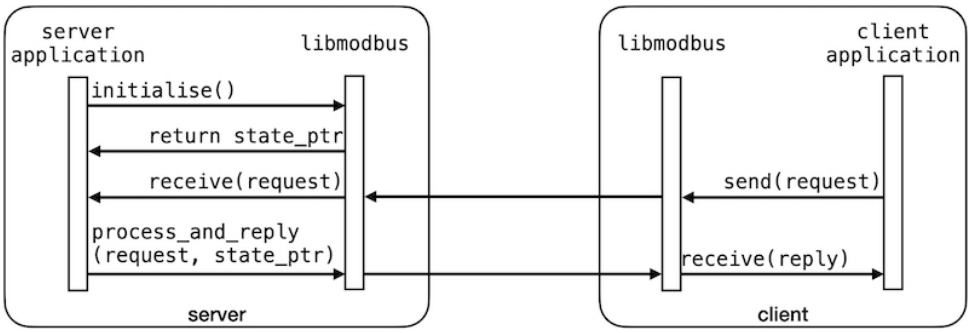

Figure 6.5: Example of initialisation and request/response interaction between applications and the `libmodbus` library.

Section 6.1.1, only the object capability layer retains privileges to read and write to the entire memory range representing the server's state.

During initialisation of the server's state, `libmodbus` returns pointers for each of the four pieces of state: coils, discrete inputs, input registers, and holding registers. Initially, these capabilities grant authority to `EXECUTE`, `LOAD`, `STORE`, `LOAD_CAPABILITY`, and `STORE_CAPABILITY`. `EXECUTE` allows the memory address to be loaded into the program counter. `LOAD` and `STORE` support dereferencing the capability and either loading a value from or storing a value to the capability's memory address. `LOAD_CAPABILITY` and `STORE_CAPABILITY` support dereferencing the capability and either loading another capability from or storing another capability to the capability's memory address. The object capability layer immediately reduces the authorities to `LOAD` and `STORE`, eliminating the possibility of an adversary attempting to execute code stored in the data regions pointed to by these capabilities. Also, by removing the privileges to `LOAD_CAPABILITY` and `STORE_CAPABILITY`, these pointers cannot be reassigned or used as an indirection to another pointer (which, in turn, could be used to execute code, etc.).

During operation, the object capability layer intercepts the call from the server application to `libmodbus` when processing a request from the client. The layer masks out permissions and memory regions of the state pointers to ensure only the explicit function and object in the request are accessible to `libmodbus`. Figure 6.6 illustrates an example request to read a coil at memory address `0x00`: the object capability layer removes all permissions from the CHERI capabilities for the discrete inputs and registers, removes `STORE` permission from coil capability, and reduces the memory bounds on the coil capability to the one byte of memory holding the coil state. As discussed in Section 6.2, a compromised `libmodbus` can only repeat specific commands previously observed, so this attenuation limits a compromised library's ability to perform a replay attack.

The network capability layer uses Macaroons as capabilities to mediate access to software-defined objects across the network. Having demonstrated the utility of Kerberos and third-party Macaroons in the ROS proof-of-concept (Section 6.3.1), for our Modbus demonstration we assume Macaroons are distributed as part of system provisioning, consistent with the recommendations in several CPS reference architectures (Section 6.4). Prior to granting it to the client, the Macaroon can be attenuated by limiting allowable commands (e.g., allowing the client to read, but not write), limiting the resources to which access is authorised (e.g., to specific coils but not registers), or by limiting the lifetime of the Macaroon and forcing an expiration and rotation. When sending a Modbus request to the server, the client's network capability layer adds the requested function to the

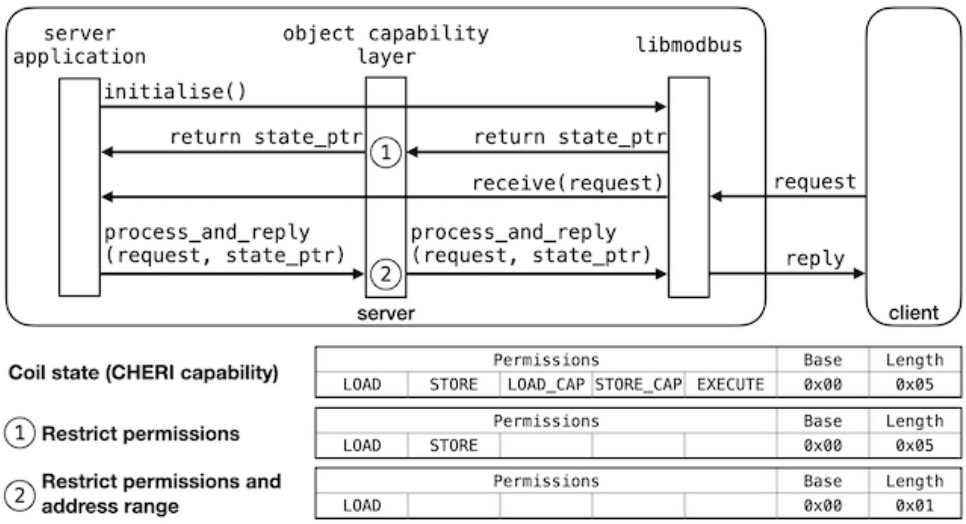

Figure 6.6: Example of the object capability layer minimising access to physical resources by limiting permissions and addresses accessible by `libmodbus`.

Macaroon as a caveat, thereby protecting it from tampering during transit, and forwards the Macaroon and request to the server. The server's network capability layer verifies the integrity of the Macaroon, verifies that the requested function and physical resource are not excluded by any Macaroon caveats, and forwards the request to the server application. As discussed in Section 6.2, Macaroons protect against tampering and reuse by adversaries on the network or resident at either endpoint.

While the object and network capability layers compose easily and work powerfully together, there is no need for them to explicitly interact. The network capability layer will ensure that a requested function and memory access is not precluded by any Macaroon caveat before forwarding the request to the application, and the object capability layer will explicitly restrict permissions on the state based on the requested function before allowing hardware to process the request. As a result of this independence, either or both of the layers can be used, depending on the context. For example, a gateway protecting a legacy device may implement the network capability layer to provide local access control for requests coming across the network, but it will not implement the object capability layer, as its hardware does not have direct control over the physical resources owned by the legacy device. Similarly, a device communicating via shared memory (e.g., a VME-based system) may only need the object capability layer, as all requests are passed via shared memory rather than network messages.

6.3.2.2 Implementation decisions and trade-offs

We aimed to demonstrate that capability-based access control could be implemented without modification to existing libraries and applications. A less-restrictive integration, or a bottom-up rewrite of the applications and libraries would improve both performance and efficiency. In this section we highlight some of these trade-offs.

To to minimise changes to existing libraries and applications, we implemented local object capabilities as a shim layer between the application and `libmodbus`. Implementing object capabilities as a shim layer requires the layer to have a full knowledge of the application state and to be able to intercept all operations on that state, effectively

requiring the shim layer to implement all functions in the Modbus API. For example, the call to `modbus_mapping_new()` allocates memory for Modbus state (a defined number of coils, discrete inputs, and registers) and returns a CHERI capability to a structure holding the uninitialised state. The shim layer intercepts the returned pointer, copies the structure, and frees the memory from the original allocation; ensuring only the shim layer has full access to the state. For subsequent calls from the application to Modbus functions that read or modify state, the shim layer only provides a capability with the necessary address ranges and permissions to execute the request. For example, if the application calls `modbus_write_register()`, the shim layer intercepts the call and the argument to write, derives a CHERI capability restricted to the specific address of the register and `WRITE` permission, and passes that to the Modbus library along with the original argument from the application. The application and library never receive a capability with unrestricted permissions to the entire state structure; however, the application and library *are* able to collect these restricted capabilities over time, allowing an adversary to slowly broaden access. While undesirable, even such a collection limits the adversary to legal actions on the state. For example, there is no Modbus function to write to a discrete input; therefore, an adversary would never observe, and never be able to collect, a capability that provided `WRITE` permission on a discrete input. The obvious benefit of using a shim is the transparency to the application and library; it can be added to any existing application in a backward compatible manner, without modifying the existing code or introducing side-effects to networked, co-dependent devices. The obvious downside is the need for indirection with every function call and the added complexity of implementing an intercepting function for every library API call. A more subtle downside is the ability for an adversary to collect capabilities without any mechanism for revocation. These issues are corrected by using a more traditional and formal definition of object capabilities.

A more traditional definition of object capabilities may also be helpful to support more expressive operations on our state objects. For Modbus, and most other ICS protocols, each request is a single operation that roughly maps to reading or writing a simple state object, which maps nicely to CHERI capabilities. Some CPS objects, however, may benefit from more expressive permissions. For example, it may be desirable to represent operations such as `INCREASE_SPEED`, `DECREASE_SPEED` and `EMERGENCY_STOP` for an object representing control of a motor's speed. These all require `STORE` CHERI permissions to a register or other memory object; however, CHERI permissions are not sufficiently expressive to distinguish between a write that increases, decreases, or zeros the value held by that memory object. In such cases, we can revert to a more conventional object capability model. We can define an object type with `permission` and `CHERI capability` members. The `permission` member can hold arbitrary permissions available for the given object type, and the `CHERI capability` member holds a CHERI capability to the memory object with the minimal CHERI permissions necessary to perform the operations allowed by the `permission` member. The CHERI capability permissions will necessarily subsume the operations allowed by the `permission` member. A CHERI capability pointing to the object is then sealed. This sealed token is the capability for that object. Only the capability manager can unseal the token, however, so all operations on the object, including reducing permissions to support delegation, must pass through the capability manager. Unlike our existing Modbus demo, where the CHERI toolchain reinterprets all integer pointers as CHERI capabilities, the use of sealed tokens as object capabilities changes the semantics for accessing objects, requiring more invasive changes to applications and

Table 6.1: Physical Lines of Code (PLOC) for the libraries and modules of the Modbus server implementation. The lines of code calculation was performed with `cloc`.

Library/Module	PLOC
Server application	770
Object capability layer	166
Network capability layer	751
`libmodbus`	5 097
`libmacaroons`	5 202

libraries. For our Modbus demonstration, this design pattern was unnecessary. Further, even where more expressiveness is possible, it may not be necessary, and the simpler design pattern (using CHERI capabilities as object capabilities) may still provide substantial security improvements without modifying existing code.

6.3.2.3 Performance evaluation

Our hypothesis is that an end-to-end capability system, implemented with CHERI and Macaroons, provides efficient memory safety, compartmentalisation, and host-based access control for distributed CPS that face the limitations and challenges discussed in Section 5.2. In this section, we evaluate those claims using our Modbus implementation. Design criteria for CPS generally include real-time constraints and guarantees that critical tasks will be scheduled and complete. We use a micro and a macrobenchmark to address these criteria, evaluating the time required to process each Modbus function at the server and the maximum throughput between a client and server. For each performance measurement, we compare performance with and without CHERI, object, and network capabilities. Finally, we look at specific security claims for our implementation.

Setup Our Modbus server runs as a CheriFreeRTOS task. The compiler toolchain is mature and open-source [45]. Our hardware platform was a CHERI-aware, RISC-V FPGA core [43] running at 100MHz on a Xilinx UltraScale FPGA. The FPGA's Ethernet is configured in SGMII mode at 1000 Mbps. The core is a five-stage, in-order, 64-bit processor for 'low-end to medium' applications and is roughly comparable with higher-end Arm Cortex-M cores (potential targets for CHERI-ARM-M [146]). The core and detailed instructions for replicating this setup are available [42]. The Modbus client runs on a Linux-based PC. The client and server were independently connected to our University network, with a `ping` consistently less than 1 msec between them.

We compiled six executable variants for comparison. The Modbus server, FreeRTOS codebase, and toolchain were the same for all variants. The baseline variant (`base`) is the primary point of reference and is compiled without CHERI, object, or network capabilities. Other variants build on that baseline and are designated as `base + [capability type]` where `[capability type]` is a combination of `CHERI`, `obj`, and `net`. There is no `base + obj` variant because object capabilities build on CHERI. In reporting results, Table 6.3 treats the `net` variant as a baseline for comparison with other network capability variants; a direct comparison against the `base` variant is not instructive as it involves a jump from micro- to milliseconds. As a rough comparison, Table 6.1 shows the lines of code for each library and module in use, and Table 6.2 shows the executable size for each variant.

Table 6.2: Executable sizes for each variant of the Modbus server with `-Os` optimisation. Object capabilities depend on CHERI architectural capabilities. Network capabilities do not, and therefore were tested both with and without CHERI capabilities.

Executable variant	Size (KiB)
`base`	486
`base + CHERI`	486
`base + CHERI + obj`	490
`base + net`	605
`base + CHERI + net`	605
`base + CHERI + obj + net`	609

Table 6.3: Result from throughput and request processing tests. Throughput measures the average time per request and includes time at both the Modbus client and server (i.e., generating, transmitting, receiving, processing, etc. at both ends). Request processing also measures average time per request, but only includes the time to process the request at the server. Note that request processing is measured in microseconds while throughput is measured in milliseconds.

Executable variant	Request processing		Throughput	
	(µs/request)	(% overhead)	(ms/request)	(% overhead)
`base`	12.8	0.0	4.00	0.0
`base + CHERI`	19.3	35.3	4.00	0.0
`base + CHERI + obj`	29.4	129.4	4.00	0.0
`base + net`	1 380	0.0	13.0	0.0
`base + CHERI + net`	1 930	39.9	15.0	15.4
`base + CHERI + obj + net`	1 950	41.8	15.0	15.4

The use of CHERI architectural capabilities alone has no discernible effect on executable size, and integrating object capabilities adds little. Unsurprisingly, integrating network capabilities results in a significant increase (about 23%) due to the relatively-large size of `libmacaroons`; however, this is a fixed addition, and would be less apparent in a larger and more functional `base` application.

Request processing test To understand the specific impact of CHERI, object, and network capabilities on individual operations, we focused narrowly on the time to process an individual Modbus request at the server (i.e., ignoring the time to receive a request and transmit a reply). In the baseline variant, processing a request involves a single function call into the Modbus library, a switch statement to distinguish between Modbus functions, and then dereferencing one or more state pointers to read or write state. As both the function call and state manipulation involve pointer operations, we expect these to incur CHERI-related overheads based on the larger size of a CHERI capability and the additional capability-based operations necessary to set up a function stack. Object capabilities add another layer of indirection and several operations on CHERI capabilities. Similarly, network capabilities add a layer of indirection; however, this will be dominated by the cost of Macaroon verification hashes.

The 'Request processing' column of Table 6.3 shows the average time and overhead to process a request. As expected, the use of CHERI capabilities and object capabilities affected the request processing time, as almost every step involves either an implicit or

explicit indirection (e.g., function call, pointer dereference). Despite this, the absolute overhead is only 7 µsec for CHERI and 17 µsec for object capabilities, which is orders of magnitude less than the round-trip between client and server, as shown below.

The overhead associated with network capabilities is also expected, as the hashing required to verify a Macaroon dominates the few instructions necessary to process the Modbus request. CHERI adds further overhead to the software-based hash implementation due to extensive pointer manipulation. We maintain this software implementation to demonstrate that network capabilities can be implemented without modification to existing libraries; however, the hash could be hardware accelerated. Open-source SHA-1 hardware accelerators process a 512 bit block in about 80 cycles, or less than 1 µsec on our hardware [198], reducing the network capability overhead from milli- to microseconds. Hardware acceleration would also eliminate the software implementation's CHERI overhead.

We also stressed the scheduler to demonstrate how CHERI capabilities affect the entire system. While object and network capabilities are confined to the processing of a request, CHERI capabilities replace all pointers, with system-wide effects on interrupts, function calls, and memory allocation. We tested execution periods from 100 msec (FreeRTOS recommendation) to 20 msec and measured the time between the Modbus server task waking and blocking. We found that CHERI overheads associated with interrupts and scheduling had no observable effect on task execution time.

Throughput test Having identified the specific effects of capabilities on processing a single function at the Modbus server, our second test measured end-to-end performance. The Modbus server was configured to process and respond to each Modbus request as soon as it arrived (i.e., the task only blocked if waiting for receipt of a request from the client). The Modbus client iterated over a set of Modbus requests to exercise all Modbus functions and state elements. The client sent each request immediately after receiving and validating the server response to the previous request, and the client recorded the time (in µsec) to complete the set of requests using `gettimeofday()`.

The 'Throughput' columns of Table 6.3 show the results for each variant, both as an average time per request and as an overhead percentage compared to the baseline variant. While CHERI has a negligible effect on the baseline, network capabilities unsurprisingly incur significant overhead due to calculating multiple hashes at the client and server. As mentioned above, hardware acceleration could reduce the overhead of network capabilities to low µsec, so both the `base + net` and `base + CHERI + net` throughput would approach the values for `base` and `base + CHERI`.

Macaroons also increase network utilisation, as a network packet containing a Macaroon is an order of magnitude larger than a packet solely containing a Modbus command (i.e., hundreds versus tens of bytes); however, the difference in transmission time, even for low bandwidth connections, is still orders of magnitude lower than the expected network latency in a real-world system. Despite this, in a hard real-time system, it would be appropriate to constrain the number of Macaroon caveats, which would constrain the maximum Macaroon size and the time to verify the Macaroon.

6.3.2.4 Security

As stated in Section 6.2, CHERI provides a class fix for spatial memory safety attacks, and object capabilities provide fine-grained access control for software-defined objects across distributed CPS. As a coarse check of our implementation's efficacy, we evaluated

Table 6.4: Capability-based mitigations for Modbus CVEs.

Exploit effect	Total	CHERI	Object/network	None
Code execution	20	8	12	0
DoS	29	6	23	0
Privilege escalation	19	0	19	0
Information exposure	18	4	13	1
Command replay	1	0	1	0

each Modbus-related CVE in the National Vulnerability Database (NVD) [46], and either directly tested or indirectly assessed whether our implementation provides (or would provide) protection.

There are 87 Modbus-related CVE entries in the NVD at the time of writing. Thirteen are related to drivers or other software on a Windows-based workstation, not an actual embedded device; eight apply to gateways, translating between protocols; and the remainder (66) apply to embedded devices, including PLCs and industrial switches. Table 6.4 categorises the stated effect of these vulnerabilities on the system and identifies those which would likely be mitigated by either CHERI or a combination of object and network capabilities. Those mitigated by CHERI capabilities are all memory safety violations (e.g., buffer overflow), while those mitigated by object and network capabilities involve unauthorised access to an object or designating non-existent objects.

We implemented and tested two memory safety CVEs against `libmodbus` (CVE-2019-14462, CVE-2019-14463), which allow buffer overreads and information exposure. CHERI capabilities identify and reject the overread, ensure no state corruption, and allow the server to continue functioning normally. We evaluated the remaining CVEs to the extent practical, based on mitigation documentation from the vendor or vulnerability writeups from those credited with CVE discovery. Only one CVE, involving cleartext transmission, cannot be mitigated by our design pattern, as we do not encrypt data; however, we prevent adversaries requesting such information without holding a capability.

6.3.3 Summary

While CPS span a broad range of domains, network types, and microarchitectures, our demonstrations are representative of two broad classes of CPS, robotics and ICS, using canonical examples and the most widely used open-source frameworks and operating systems. We demonstrate that CHERI and object capabilities add insignificant overhead, even to lightweight, real-time systems. While network capabilities add milliseconds of overhead, we identify open-source hardware acceleration that would reduce this to the µsec level. Finally, through testing and evaluation, we show that our security architecture has the potential to mitigate almost all Modbus-related CVEs.

6.4 Existing architectural support

This section demonstrates the existing infrastructure to support our implementation in the CPS domain as well as the changes necessary and security improvements available from an end-to-end capability architecture.

The infrastructure required to support Macaroons is minimal: provided a communication path already exists between resource owners, resource users, and (if desired) trusted third parties, the only additional requirement is the ability for the resource owner to share a Macaroon with the resource user or to share a single secret with a trusted third party. In a resource-constrained or *ad hoc* network, this can be executed with a Trust-On-First-Use (TOFU) mechanism [194]. In larger and more stable environments, this task can be performed using existing authentication mechanisms. In this section, we evaluate several CPS reference architectures and some concrete examples to show the feasibility of implementing CHERI and Macaroons across the CPS spectrum, from the most resource-constrained networks to large, distributed, multi-site industrial networks. Further, we show how CHERI and Macaroons provide more efficient mechanisms to implement vendors' own recommendations associated with boundary control and token-controlled access. Similarly, we show that CHERI and Macaroons allow access control to be pushed down to the fieldbus level (i.e., level 0 in Figure 2.1), which is either unsupported or impractical in the reference architectures and concrete systems we evaluate.

6.4.1 Open Platform Communications Universal Architecture (OPC UA)

"OPC UA is a platform-independent standard through which various kinds of systems and devices can communicate" and "is applicable to components in all industrial domains" [153]. OPC UA provides abstractions and protocols for synchronous and asynchronous communication between clients and servers. The security model assumes authentication and authorisation services are independently available as part of the service discovery process, and that they are facilitated using certificates, to establish identity, and JSON Web Tokens (JWTs) [97], to establish authorised access [55, 181] (see Figure 6.7). For systems without a trusted third party or certificate authority, OPC UA also provides explicit instructions for self-signing and securely exporting and importing certificates [181].

An OPC UA system is well-equipped to use Macaroons as a one-for-one substitute for JWTs. The benefit of Macaroons in this case is the support for third party caveats and discharge Macaroons. Third party caveats allow resource owners and users to maintain resource Macaroons with long expirations (limiting the rotation demand on edge devices of a distributed CPS), while forcing a resource user to authenticate with the authorisation service and obtain an ephemeral discharge Macaroon, which is provided by the user to the resource owner alongside the resource Macaroon. This allows most of the security and reliability guarantees of OPC UA to be extended to edge devices, which are currently unable to participate in the authenticated and authorised links between clients and servers due to resource and real-time constraints. Conversely, CHERI-based object capabilities can easily map to JWTs instead of Macaroons, allowing our end-to-end capability architecture to be integrated into the existing OPC UA framework.

6.4.2 Industrial Internet Reference Architecture (IIRA)

The Industrial Internet Consortium (IIC) provides a multi-volume publication containing technical reports with guidance, recommendations, and best practices for Industrial Internet of Things (IIoT). One of these is the IIRA, which provides a framework for developing IIoT systems from business, usage, functional, and implementation 'viewpoints' [166].

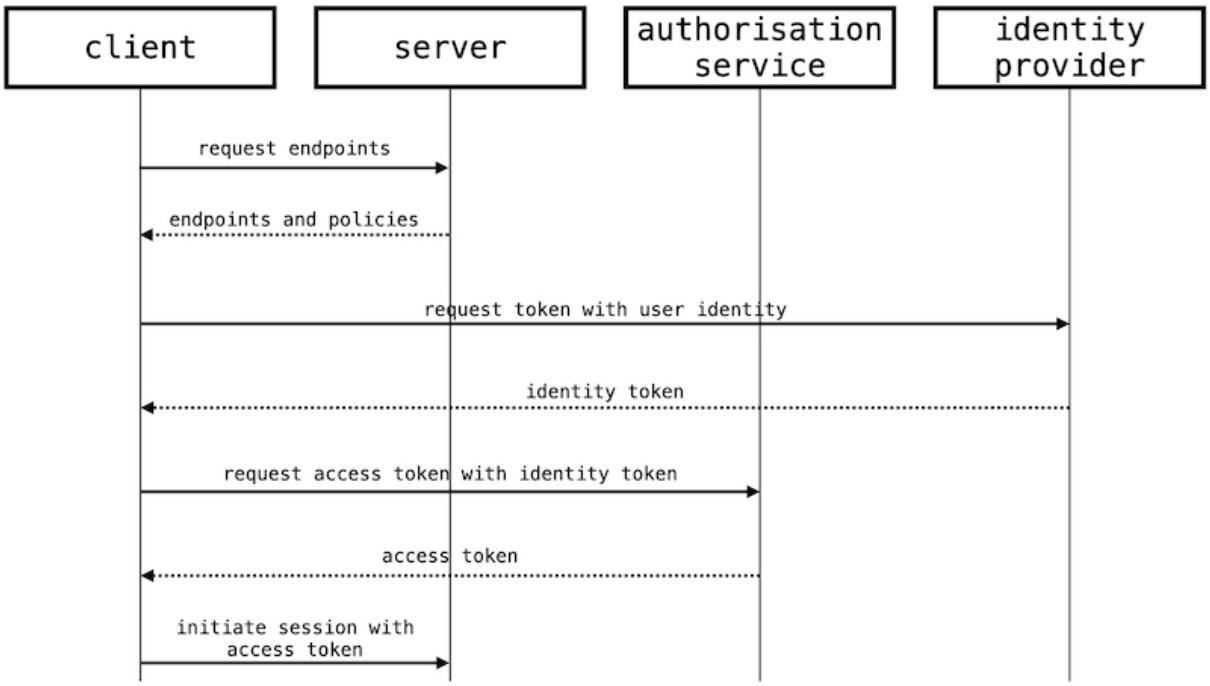

Figure 6.7: OPC UA client obtaining and using an access token [185].

The implementation viewpoint includes three design patterns for large IIoT systems: a three-tiered architecture, separating the edge, platform, and enterprise devices; a gateway-mediated architecture for edge devices, separating edge networks from wide-area networks with parameterising gateways; and a layered databus architecture, for low-latency, peer-to-peer communication across logical system layers [166]. The IIRA security framework documents the need for authentication and authorisation, even at resource-constrained edge devices, and recommends the gateway-mediated design pattern to secure brownfield deployments where legacy devices cannot be individually secured. The security framework also provides recommendations for isolation within edge devices, either through process isolation, containerisation, virtualisation, or physical separation [180].

The IIRA has much in common with OPC UA from a security standpoint, including recommendations to use device specific authentication and authorisation, even for resource-limited devices. The gateway-mediated design pattern in the IIRA would support Macaroon usage, even for backfitting in brownfield deployments. CHERI also provides a low-cost mechanism for recommended isolation, even in resource-constrained edge devices.

6.4.3 Microsoft Azure

Microsoft provides a reference architecture for connecting "sensors, devices, and intelligent operations" with Microsoft Azure IoT services [132]. The reference architecture is focused on Microsoft Azure cloud services and, where appropriate, Microsoft Azure Sphere hardware, software, and development environments [130]. Like OPC UA, Microsoft's reference architecture recommends a gateway-mediated architecture using Azure IoT Hub and per-device authentication and authorisation using Azure Active Directory. The reference architecture also has an extensive discussion on device provisioning to 'home' an edge device with its gateway by exchanging key and token material. Provisioning is expected to be done in person, before a device is installed. The Azure application architecture

guide emphasises the use of authorisation tokens over methods requiring key exchange and encourages just-in-time rather than standing privileges [129].

The reference architecture and application architecture guides' emphasis on gateway-mediation and in-person provisioning all clearly support Macaroon-based token authorisation, and the application architecture guide explicitly encourages token use, though a different token scheme is recommended (Shared Access Signature (SAS) [131]). Further, the reference architecture encourages shifting complex identity decisions away from edge devices [132], which makes the simplicity and minimal cryptographic requirements of third party caveats and discharge Macaroons particularly attractive.

6.4.4 AUTomotive Open System ARchitecture (AUTOSAR)

AUTOSAR is an open, standardised software architecture for automotive Electronic Control Units (ECUs) developed by an international partnership of automotive stakeholders. It uses several abstraction layers between the actual microcontroller and the runtime environment to provide common services and features amongst heterogeneous ECUs. While AUTOSAR does not specifically address interconnections between ECUs across an automobile (e.g., topologies, gateways, boundary control), it does provide software modules and communication paths for common protocols, including libraries for protecting end-to-end communication. Notably, end-to-end protection in this context refers to the integrity of the communication rather than the confidentiality, and authentication and authorisation are not explicitly addressed [73, 112]. For safety-critical applications, AUTOSAR provides software facilities for segregating critical from non-critical software modules on a single ECU and for Byzantine fault tolerance [112]. Recently, AUTOSAR released a new standard to address vehicle-to-x interfaces [72]. While still not addressing car-wide architectures, it does discuss multi-ECU interactions, such as handling ECU degradation by transferring responsibility for degraded functionality to a different ECU.

CHERI compartmentalisation is a strong candidate to support AUTOSAR's design goals of an adaptable, upgradeable, and durable software architecture. CHERI's efficient, hardware-enforced compartmentalisation supports the strong separation desired between critical and non-critical software modules, and also allows for unrelated modules to be isolated from one another. For example, the microcontroller abstraction layer can be largely isolated from the communication software modules, mitigating network stack exploits (e.g., Ripple20 [168], URGENT/11 [184]) and potentially reducing the regression testing required for software updates. While AUTOSAR does not explicitly discuss tokens, the libraries supporting end-to-end protection are capable of performing all operations required for generating, adding caveats to, and validating Macaroons.

6.4.5 Manufacturing on a Shoestring

The architectures described above require system-wide planning and implementation, special infrastructure, and considerable expertise to install and maintain. Many Small and Medium-sized Enterprises (SMEs) are only willing to take incremental and low-risk steps to digitise their manufacturing and automation, and are unlikely to have the in-house skill to manage a large, digital infrastructure [176]. The Digital Manufacturing on a Shoestring project is a tactical, rather than strategic, approach to help SMEs obtain some of the benefits of Industry 4.0. The Shoestring project is trialling low-cost, commercial off-the-shelf (COTS) building blocks that can be incrementally integrated via a common

communication backbone [85, 127]. Current building block demonstrators are based on Single Board Computers (SBCs) (e.g., Raspberry Pi) and target tasks such as real-time job tracking, machine monitoring, and digitised work instructions.

The incremental, low-risk, tactical digitisation efforts of SMEs present a challenging security engineering environment; however, even these scenarios provide opportunities for our capability-based access control approach. SBCs effectively act as gateways to unsecured, legacy edge devices, and provide sufficient resources to support our Macaroon-based system. These small deployments could use TOFU authentication to share secrets and Macaroons between the resource owner and resource user, allowing the resource user to generate a discharge Macaroon as part of a service request to the resource owner. Such a system would allow incremental deployment of COTS devices providing the desired digital functionality and also providing local (if not actually host-based) access control without additional infrastructure or significant expertise. The availability of CHERI CPUs on such SBCs would provide efficient memory safety and compartmentalisation benefits; however, since the SBC is only a gateway and not the actual resource owner, it cannot provide hardware-enforced access control to physical resources.

6.4.6 Vendor examples

Siemens and Rockwell Automation both provide suites of industrial controllers, switches, security appliances, and infrastructure for controlling and coordinating arbitrarily large industrial processes. Both encourage and provide centralised services to support authorisation and authentication at the device-level, though this does not extend to many edge devices or any communication using a fieldbus protocol. Finally, both provide means of provisioning devices prior to installation, which provides the mechanism and opportunity to install any Macaroon-related material [13, 187].

Siemens products are designed for network and processing redundancy, allowing controllers to transmit commands and information on redundant networks (see Figure 6.8) [190]. Macaroons can be supported on redundant networks, as identical Macaroons can be transmitted on the redundant links. Redundant controllers would need to share keys for generating Macaroons so each could verify Macaroons created by the other. Further, CHERI compartmentalisation could be used within the communication module to isolate redundant channels, eliminating some use cases for redundant controllers.

Siemens has recently released a new product, the Compact Field Unit (CFU), to digitise and consolidate communication with field devices (see Figure 6.9). The CFU is marketed as a device to simplify and add flexibility to deployed field devices, such as sensors and actuators [190]. CFUs provide both a new use-case and a simple and inexpensive infrastructure for supporting Macaroons. The consolidation of communication to field devices further complicates the access control problem, as an attacker with access to the gateway can interfere with communication to any of the field devices. Macaroons allow each CFU to manage access control decisions locally, even if they are connected to legacy (even analogue) field devices without the ability to process Macaroons themselves.

Rockwell Automation provides multiple reference architectures dedicated to Device-Level Rings (DLRs) (see Figure 6.10). DLRs provide efficient redundancy by allowing all network traffic to travel around the ring in both directions; therefore, all devices participating in the ring continue to see all traffic during a single link failure [14]. DLRs, like the redundant Siemens networks described above, provide opportunities for deploying

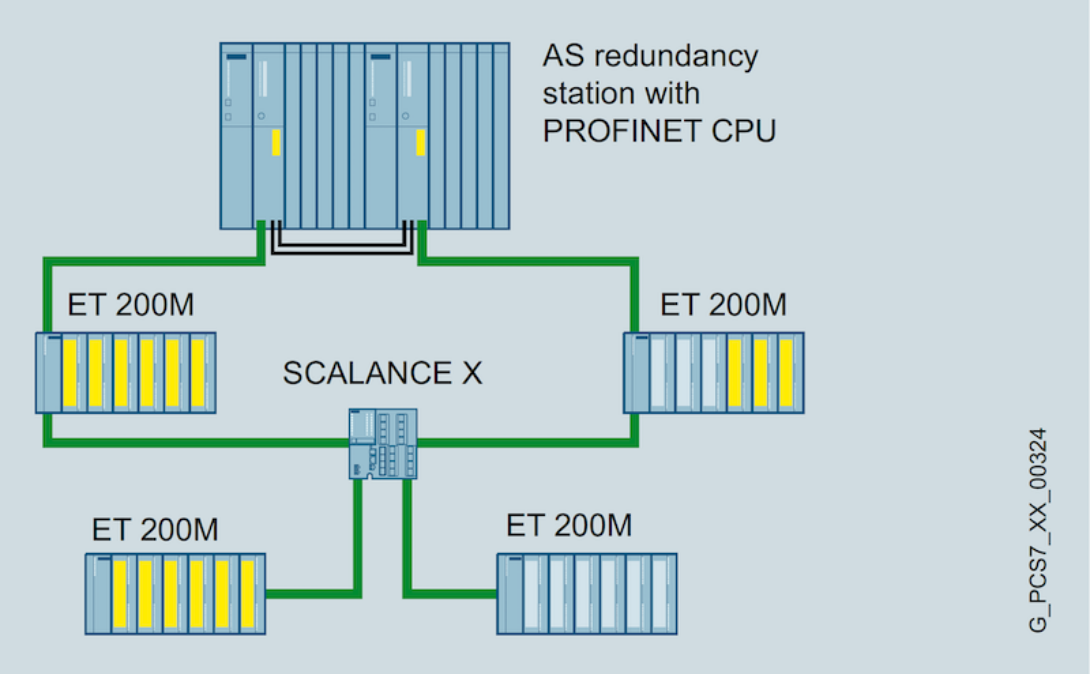

Figure 6.8: Siemens diagram showing redundant network transmission from redundant controllers [190].

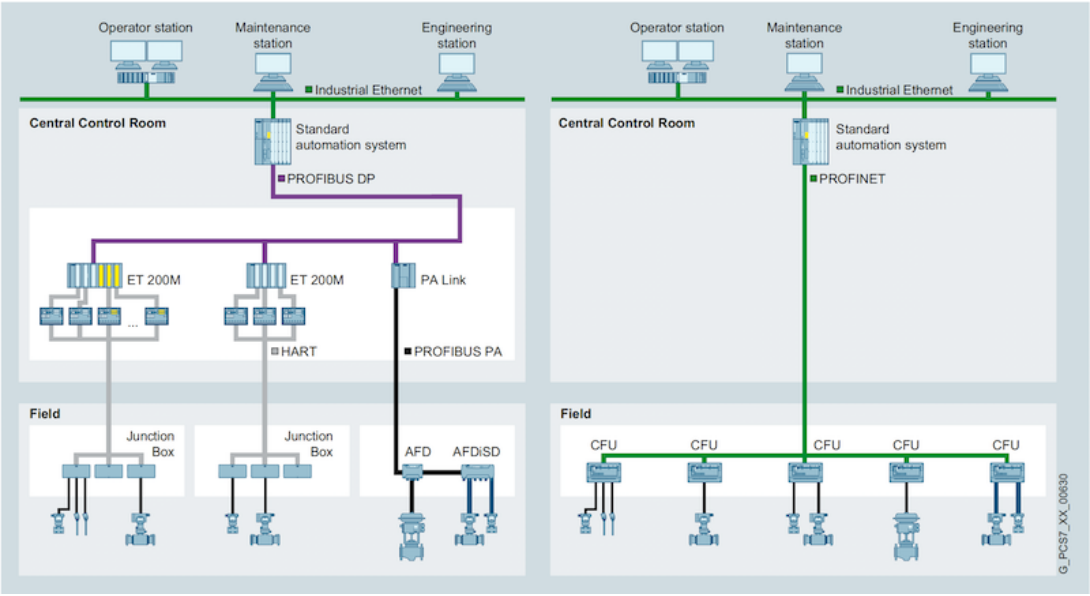

Figure 6.9: Siemens diagram comparing the multiple digital and analogue paths required by legacy field networks with CFU digitisation and consolidation [190].

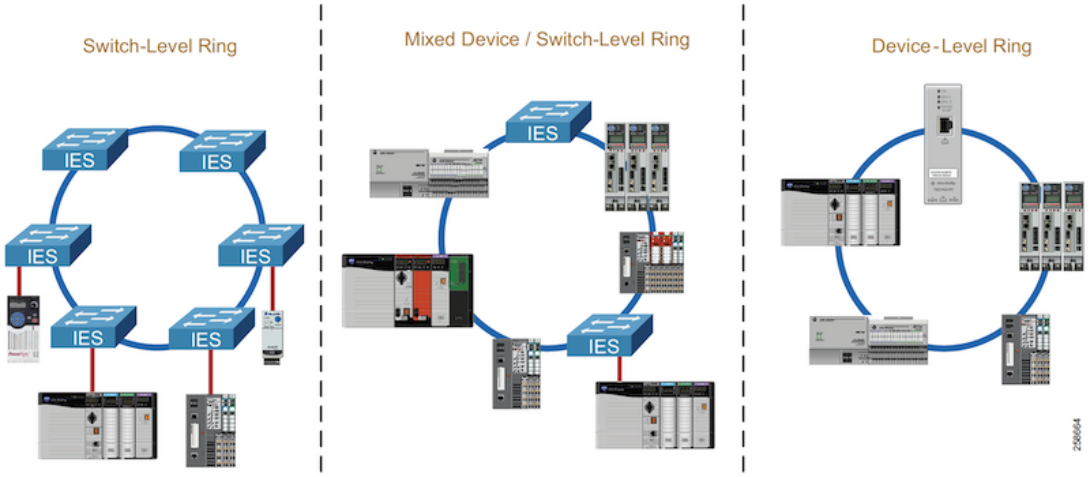

Figure 6.10: Rockwell Automation diagram showing multiple ring architectures [14].

CHERI and Macaroons to field devices. The resource owner is able to generate and transmit the Macaroon around the ring for potential users, and resource users can add desired commands and return the Macaroon to the resource owner. Macaroons would be transmitted in both directions and the owners and users would need logic to check and discard the Macaroon received second (e.g., by adding a counter to the Macaroon's information field or as a caveat).

6.5 Related work

The content of this chapter relates to both historical and recent work in the fields of capability systems, memory safety, boundary defence, and network authorisation. We briefly discuss related work below and compare it with the work presented in this chapter.

6.5.1 Capability systems

seL4 is a well-known capability system, and it carries the added assurance of formal verification, but it has seen few practical or commercial deployments. It demonstrated that software-defined, object capabilities can be used for provable spatial and temporal isolation in multi-criticality, embedded systems; however, the seL4 security model requires MMU support [104, 124], which disqualifies it from many CPS applications (seL4 without an MMU is frequently requested [70]). Further, seL4 capabilities are constrained to specific kernel objects, precluding creation of the software-defined objects we use in our end-to-end system [199]. Other software-based capability systems (e.g., Fiasco.OC [110], M^3 [11]) are similarly constrained. Only CHERI provides both hardware-enforced capabilities and the building blocks for complex, software-defined capabilities necessary to implement our end-to-end design pattern.

Xia *et al.* demonstrated that CHERI architectural capabilities can be used for fine-grained memory isolation between tasks in a RTOS without an MMU [219], but they provided no mechanism for constructing capabilities representing abstract objects, which is necessary for a capability-based access control solution.

Finally, none of the solutions above have direct knowledge of the network, and therefore do not provide native support for capability-based access control between distributed CPS devices; though it is possible they could be extended with our network capability scheme.

6.5.2 Other distributed capability models

In this chapter and in Chapter 5 we use 'distributed' to refer to an end-to-end capability system for CPS consisting of a network of heterogeneous devices; however, the capability literature also uses 'distributed' to refer to a capability system on a System-on-Chip (SoC) or Network-on-Chip (NoC), with multiple, heterogeneous processing elements (e.g., CPU, DSP, GPU, or FPGA) on the same wafer or in the same package. For example, Baumann *et al.* introduced Barrelfish, an operating system to demonstrate their multikernel model and design principles of explicit communication, hardware-neutral structure, and state replication for heterogeneous, interconnected processing elements. Barrelfish relies on independent operating system instances on each core and uses seL4-like capabilities to control access to resources. While the independent kernel instances manage capability operations on a given core, per-core 'monitors' are necessary to handle the global coordination of shared resources, such as memory mapping or capability retyping, for which the monitors collectively ensure consistency of locally-replicated state [18]. SemperOS, based on the M^3 microkernel, similarly relies on message passing and replicated state to implement a distributed capability system on a SoC or NoC. Unlike Barrelfish, SemperOS does not require a kernel instance for each processing element. Instead, SemperOS uses one kernel instance per group of processing elements and relies on extra, trusted hardware components (Data Transfer Units (DTUs)) at each processing element to allow the kernel to manage distributed resources in the group. For global resources, where coordination between groups is required, SemperOS employs consistency mechanisms like Barrelfish, but relies on DTUs rather than software-based monitors [89].

These systems both deploy trusted monitors on each processing element that maintain consistency through an algorithmic commitment mechanism. This is effective because the systems, like most monolithic OSes, assume that the kernel is trusted and that one of the kernel's jobs is to mediate access to shared resources between mutually distrusting applications. As we discuss in Chapter 5, this assumption does not hold in CPS, where the networked devices are mutually dependent, but also mutually suspicious, and there is no network monitor or mechanism for implementing a trusted monitor on each device.

6.5.3 Memory safety and Control Flow Integrity (CFI)

The 1990s and 2000s produced several efforts to run untrusted code efficiently and safely within a trusted memory space. Software Fault Isolation (SFI) leverages special control sequences of existing instructions to control jumps into and constrain untrusted code [182, 213]. It is both efficient (5%–7% overhead) and implementable in commodity hardware; however, it is designed to sandbox untrusted software in a trusted memory space and has no way to express or control privilege for shared resources between two mutually-distrusting processes, which is a necessary component of our end-to-end capability architecture. Further, while SFI provides limited CFI, full protection for CFI using control sequences of existing instructions incurs significantly more overhead ($\tilde{1}5\%$) [2, 182]. In contrast, CHERI provides both memory protection and CFI for mutually-distrusting processes with overheads comparable to the most efficient SFI implementations [167].

6.5.4 Compartmentalisation

Historically, the limited number, static constraints, and performance costs of MPUs have precluded wide-spread, practical compartmentalisation with microcontrollers [219]; however, several recent research efforts improve MPU usability, helping developers map their applications to separate memory spaces [34]. Further, CPS researchers have shown promising use cases for MPUs, such as implementing reference monitors in a separate memory space from the RTOS, providing strong security guarantees without burdening the CPS developer with compartment management [103]. These improvements are either unnecessary or more intuitive with CHERI-based compartmentalisation, which simplifies compartment creation and management to support reference monitors or other security appliances running alongside an application.

6.5.5 Boundary defence

Many industrial solutions rely on network boundary defences to mitigate the prevalence of legacy protocols. As stated in Chapter 5, Siemens recommends organising devices with a common task into 'automation cells' with a security appliance at the boundary [188]. Azure Sphere recommends a hub-connected architecture, where the edge device maintains a secure connection to cloud-based services, supporting data transfer and over-the-air software upgrades; however, unless sensor and actuator communications also occur via the cloud, those communications will still rely on legacy communication mediums and protocols. Other CPS reference architectures similarly secure higher layers of the communication hierarchy, but do not protect communication for these 'layer 0' devices. In all cases, communication amongst 'layer 0' devices assumes complete trust. Chapter 2 similarly discusses the benefits and limitations of anomaly detection within the boundary [77], which make stronger assumptions about the adversary, but add design and support challenges.

While boundary defence and anomaly detection contribute to defence-in-depth, they cannot minimise privilege in a system, are focal points for attack, and add complexity. Our end-to-end capability architecture is designed to minimise privilege at every layer of the hierarchy without adding such complexity.

6.5.6 Network tokens

As discussed in Section 6.4, many CPS reference architectures rely on network tokens to provide authenticity and integrity for network communications. For example, Azure Sphere uses SAS tokens [131] and OPC UA uses JWTs [97] to establish authorised access [55, 129, 130, 181]. We chose Macaroons as our network token to minimise overheads, as discussed in Section 6.1.2; however, SAS tokens and JWTs could be composed with CHERI capabilities in our end-to-end capability architecture, allowing our design pattern to integrate into these reference architectures.

6.6 Summary

Boundary defence and intrusion detection provide no defence-in-depth for distributed CPS because they do not minimise privilege or encourage layered isolation. We proposed and implemented a new approach, demonstrating that distributed capability systems not only

minimise privilege up and down the CPS hierarchy but also solve specific CPS challenges, such as securing the routine performance of privileged operations.

We provided the first least privilege decomposition of CPS workloads, demonstrating that capabilities allow this fundamental security principle to be implemented at the hardware, software-object, and network layers. Specifically, we showed that object capabilities representing physical resources can be constructed, delegated, and used across a distributed CPS, and that a composition of CHERI architectural capabilities and Macaroon cryptographic capabilities guarantees capability properties of unforgeability, provenance, and monotonicity at every level of the hierarchy. Using concrete implementations of a robotic vehicle using ROS and an industrial controller running on FreeRTOS, we showed that this architecture was performant and practical to implement. Further, we demonstrated that the architecture overcomes failures in existing security architectures, in that it mitigates and tolerates adversarial attacks at both device and network boundaries.

CHAPTER 7

Conclusions and further work

The aim of this work was to understand and improve the security of Cyber Physical Systems (CPS). We started with a security economics and usable security assessment of the Internet-connected Industrial Control System (ICS) population. We documented the vulnerability of the population, demonstrated why such a vulnerable population develops and grows, and explained the historical and counter-intuitive lack of interest in attacking it. Despite the lack of active, scaled attacks, we showed that hardware, software, and network convergence amongst CPS is already increasing the security risk to subdomains such as ICS as they become more like IoT. Existing solutions focused on boundary defences are insufficient to handle this convergence, so we proposed a new CPS security architecture that minimises privilege and supports the distributed, hierarchical nature of many CPS. First, we systematised the unique security challenges facing CPS and explained why hardware and software improvements and common security solutions from the Information Technology domain were insufficient to address those challenges. Then we proposed, implemented, and evaluated a distributed capability system for CPS, creating a least privilege decomposition at every level of the CPS hierarchy. Finally, we evaluated our implementation, demonstrating it was both practical to implement in existing CPS architectures and performant under realistic operating conditions. In the following sections, we summarise our conclusions and provide recommendations for further research.

7.1 Conclusions

In Chapter 3, we asked why a decade of catastrophic predictions about the security of Internet-connected ICS devices had failed to materialise. To answer this question we performed the first longitudinal study of Internet-connected ICS devices, showing that we could retroactively fingerprint and track over 10 000 devices over several years using public datasets. We demonstrated the Internet-connected ICS device population is growing steadily, that a majority of devices are continuously connected at the same IP address, and that fewer than 30% of devices are ever patched. We then developed a security economics model for large-scale attacks against Internet-connected populations and showed that the fragmentation of the ICS community, the high cost to develop and test exploits, and the unpredictable monetisation and consequences of attacking ICS currently make them an unattractive target for the cybercrime community, especially given the continued vulnerability of the larger and more homogeneous IoT population. We confirmed this

assessment by evaluating hacker forums, malware, and ICS honeypots, demonstrating there was little interest or competence in attacking the ICS population.

Chapter 4 expanded on the ICS honeypot assessment. Despite demonstrating the current lack of interest in maliciously manipulating the behaviour of ICS devices at scale, it provided the first evidence of the convergence between ICS and IoT networks. Specifically, we showed that ICS devices were hosted on the same networks as malware-infected IoT devices. While the network convergence alone is capable of disrupting industrial processes, in Chapter 3, we demonstrated the imminent convergence of ICS devices themselves with the IoT domain, as inexpensive hardware, commodity operating systems, and ubiquitous, wireless connectivity become standard in the ICS domain. Our security economics model predicts that such a convergence will either result in ICS devices becoming targets for large-scale attacks, or that ICS devices will be swept up in attacks against the larger IoT population. This network convergence may be the leading edge of a broader and more disruptive convergence between CPS subdomains.

As stated above, industry's solution to these security challenges is boundary defence, pushing privilege and responsibility to firewalls and anomaly detectors; however, this propagates rather than minimises privilege and leaves the entire hierarchy vulnerable to a single boundary compromise. In contrast, we proposed and implemented a security architecture based on end-to-end, distributed capabilities.

In Chapter 5 we systematised the unique access control challenges faced by the CPS domain, including the lack of memory management hardware and trusted kernels, the inability to separate privilege, and the mutual dependence and mutual suspicion of networked CPS devices. We compared access control solutions to mitigate these challenges, focusing on Access Control Lists (ACLs) and capability systems, and we demonstrated that capabilities present a stronger foundation for access control in distributed CPS. Specifically, they provide the building blocks for protecting resource invocation both locally and across a network without memory management hardware or kernel intervention, they bypass concerns associated with CPS' routine, privileged operation, and they are suitable tools for establishing trust between heterogeneous, networked devices.

Finally, Chapter 6 presented a design pattern for a distributed security architecture based on capabilities. We showed that object capabilities, representing physical resource attributes, could be constructed, delegated, and used anywhere in a distributed CPS by composing hardware-enforced architectural capabilities and cryptographic network tokens. We concretely implemented our design pattern using Capability Hardware Enhanced RISC Instruction (CHERI) architectural capabilities and Macaroon cryptographic capabilities, and we provided two case studies: a robotic vehicle using the Robot Operating System (ROS) running on a CHERI-aware version of FreeBSD, and an industrial controller running on a CHERI-aware version of FreeRTOS. In neither case did the open source libraries or applications require modification for use with CHERI. The overhead from adding CHERI-based object capabilities was unobservable across the network in both cases. While the overhead from Macaroon cryptographic capabilities was unobservable across the network in the ROS case study, for the industrial controller Macaroons added about 2 milliseconds to the request/reply round trip between client and server, which is comparable to the expected latency in an industrial network.

These case studies showed that our architecture was performant and practical to implement. Further, we demonstrated that the architecture overcomes failures in existing security architectures, in that it mitigates and tolerates adversarial attacks at both

device and network boundaries. In summary, our architecture provides defence-in-depth, minimising privilege at every level of the CPS hierarchy, and both supports and adds integrity protection to legacy CPS protocols.

7.2 Further work

While both the motivation for and implementation of our distributed security architecture is generalisable to the whole CPS domain, the ICS subdomain was our primary focus, as it presents the greatest existing threat and challenge of the CPS subdomains. As discussed in Chapter 3, we expect the economics of attacking this domain to change as ICS hardware and software converge with IoT, and this opens several possible areas of research. First, our longitudinal study was limited to directly addressable devices, but recent work monitoring traffic at Internet Exchange Points (IXPs) shows that the actual number of ICS devices insecurely using the Internet for monitoring and control may be an order of magnitude higher than the roughly 100 000 devices we identified [17]. Further, data from IXPs is closer to ground truth than that available through Internet-wide scanning, as the identified devices are actually using ICS protocols to communicate, whereas Internet-wide scanning only demonstrates a device is open on a given port and understands a given protocol. Therefore, researchers should identify systematic mechanisms for ethically monitoring, fingerprinting, and indexing ICS and IoT traffic at these IXPs to support longitudinal studies of device behaviour and ICS/IoT network convergence. This may provide a better insight into the state of ICS security than what we were able to achieve in Chapters 3 and 4, and it would certainly be the best mechanism for monitoring future convergence. Similarly, as IPv6 becomes more prevalent, Internet-wide scanning becomes impractical, and IXP monitoring will become the primary mechanism for measuring the population size and behaviour of Internet-connected devices.

While the challenges discussed in Chapter 5 make CPS an attractive target for an end-to-end, distributed capability architecture, there appear to be opportunities to generalise this design pattern. For example, in Section 6.5, we described Barrelfish and SemperOS, two implementations of the *other* use of the term 'distributed capabilities' to refer to heterogeneous processing devices on a single System- or Network-on-Chip (SoC or NoC) [18, 89]. These systems use a trusted hardware- or software-based monitor at each processing element to generate trust and ensure consistency of shared state between elements. CHERI compartmentalisation would be ideally suited for protecting the software-based trusted monitor and may be sufficient to eliminate the motivation for a hardware-based monitor. Further, it may be possible to replace the monitors altogether with a mechanism similar to our use of Macaroon cryptographic capabilities, allowing one processing element to access the resources of another element if it holds an unforgeable and appropriately-permissioned token. This would also require the development of a consensus protocol using cryptographic capabilities, such as Macaroons, as the monitors manage both access control and consensus for replicated state. If these hurdles were overcome, however, such a mechanism would allow the design patterns used by Barrelfish and SemperOS to extend between multiple chips and multiple devices, as it would eliminate the need for local control of the software or hardware reference monitors to ensure trust.

More ambitiously, CHERI may provide the necessary primitives to remove seL4's dependence on MMUs, eliminating one of the major hurdles to wider adoption of seL4 in embedded devices. Currently, seL4 relies on the MMU to isolate users from operations on

capabilities performed by the kernel. CHERI-based compartmentalisation could provide this isolation. Further, seL4's fundamental objects, which are designated and manipulated via capability invocations, could be constructed as CHERI-based objects, as described in Chapter 6. In addition to supporting wider adoption, eliminating MMU dependence and using CHERI-based object capabilities may strengthen the provable security properties of seL4, as it supports finer-grained and hardware-enforced access control to memory objects. Further, seL4's new Mixed Criticality System (MCS) kernel may provide the practical mechanisms necessary to implement CHERI-based temporal safety. Specifically, the MCS kernel introduces capabilities for processing time and supports provable Worst-Case Execution Time (WCET) calculations. At a minimum, these provide the ability to deterministically bound the time to perform mark and sweep operations to revoke architectural capabilities, mitigating use-after-free memory safety vulnerabilities. Existing proof-of-concepts for CHERI temporal safety make use of MMUs [67]; therefore, this work would need to be extended to support the goal of removing seL4's MMU dependence.

Finally, CHERI-based compartmentalisation could also provide a foundation for limiting regression testing of CPS integrations following software perturbation. The cost of regression testing, which may require complex test harnesses or regulatory recertification, creates patch resistance in the CPS domain [114]. Using CHERI to formally prove that a given software perturbation cannot affect certain physical behaviour may reduce the cost of such regression testing, avoid recertification requirements, or obviate the need maintain test harnesses over long periods. Similarly, CHERI may support the development of MCS generally (in addition to the specific seL4 MCS kernel). The benefit of MCS comes from reducing the amount of hardware required in a system, because it avoids having to split low and high criticality software between devices. The complexity of MCS comes from having to demonstrate that a failure or compromise of lower criticality software will not affect the performance of higher criticality software. As with seL4, hardware enforcement of CHERI-compartmentalisation and access to CHERI-based object capabilities may simplify such a demonstration. Limiting the cost of regression testing is particularly important in MCS because it is often the lower criticality software (e.g., TCP/IP libraries) that receive frequent updates, and the desire to avoid requalifying the higher criticality software creates a perverse incentive not to patch the vulnerable, network facing software.

The convergence of hardware and software use amongst the CPS subdomains is both a threat and an opportunity. The threat is primarily for older subdomains, like ICS, that continue to use protocols and design systems assuming that the 'air gap' threat model is still valid, and that devices within a boundary can be treated as fully trusted. The opportunity comes from the greater resources available on the commodity hardware and the larger community of developers and security researchers associated with commodity software. These resources make a distributed capability architecture possible for CPS, and we hope this dissertation is a useful and practical contribution to improving security in a domain historically constrained by technical, usability, and economic security challenges.

Bibliography

[1] *3S CoDeSys Vulnerabilities*. Cybersecurity and Infrastructure Security Agency (CISA). 2013. URL: https://perma.cc/F8W4-7H75 (visited on 28/10/2019) (cit. on pp. 56, 61).

[2] Martín Abadi et al. 'Control-Flow Integrity Principles, Implementations, and Applications'. In: *ACM Transactions on Information and System Security* 13.1 (Oct. 2009), pp. 1–40. ISSN: 1094-9224, 1557-7406. DOI: 10.1145/1609956.1609960. URL: https://dl.acm.org/doi/10.1145/1609956.1609960 (visited on 11/05/2021) (cit. on p. 101).

[3] Marshall Abrams and Joe Weiss. *Malicious Control System Cyber Security Attack Case Study – Maroochy Water Services, Australia*. Mitre Corp, 2008. URL: https://perma.cc/CTX9-A673 (cit. on pp. 21, 58).

[4] Anne Adams and Martina Angela Sasse. 'Users Are Not the Enemy'. In: *Communications of the ACM* 42.12 (1999). ISSN: 00010782. DOI: 10.1145/322796.322806. URL: http://portal.acm.org/citation.cfm?doid=322796.322806 (visited on 07/06/2019) (cit. on pp. 40, 52).

[5] 'Analysis of the Cyber Attack on the Ukrainian Power Grid'. In: *Electricity Information Sharing and Analysis Center (E-ISAC)* (2016). URL: https://perma.cc/74V7-TN7J (visited on 11/03/2020) (cit. on pp. 41, 53).

[6] Ross Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. John Wiley & Sons, 2020 (cit. on pp. 30, 31, 45).

[7] Ross Anderson et al. 'Measuring the Changing Cost of Cybercrime'. In: *Proceedings of the 11th Workshop on the Economics of Information Security (WEIS '12)* (2019). URL: https://perma.cc/ZKB4-C6Q6 (cit. on pp. 41, 45, 58, 59).

[8] Manos Antonakakis et al. 'Understanding the Mirai Botnet'. In: *Proceedings of the 26th USENIX Security Symposium (USENIX '17)*. 2017. URL: https://www.usenix.org/system/files/conference/usenixsecurity17/sec17-antonakakis.pdf (visited on 10/06/2019) (cit. on pp. 23, 42, 55).

[9] Daniele Antonioli, Anand Agrawal and Nils Ole Tippenhauer. 'Towards High-Interaction Virtual ICS Honeypots-in-a-Box'. In: *Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security & Privacy (CPS-SPC '16)*. 2016. DOI: 10.1145/2994487.2994493. URL: http://dl.acm.org/citation.cfm?doid=2994487.2994493 (visited on 20/12/2019) (cit. on p. 24).

[10] *Arm Vision*. The Architecture for the Digital World. 2021. URL: https://www.arm.com/campaigns/arm-vision (visited on 25/06/2021) (cit. on p. 28).

[11] Nils Asmussen et al. 'M3: A Hardware/Operating-System Co-Design to Tame Heterogeneous Manycores'. In: *Proceedings of the 21st International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '16)*. Atlanta, Georgia, USA: ACM Press, 2016. ISBN: 978-1-4503-4091-5. DOI: 10.1145/2872362.2872371. URL: http://dl.acm.org/citation.cfm?doid=2872362.2872371 (visited on 16/10/2020) (cit. on pp. 81, 100).

[12] *Attacks on Arkansas Power Grid*. Federal Bureau of Investigation (FBI). 2015. URL: https://perma.cc/DNU6-EM65 (visited on 28/10/2019) (cit. on p. 58).

[13] Rockwell Automation. *Deploying CIP Security within a Converged Plantwide Ethernet Architecture*. 2020. URL: https://literature.rockwellautomation.com/idc/groups/literature/documents/td/enet-td022_-en-p.pdf (visited on 23/06/2021) (cit. on p. 98).

[14] Rockwell Automation. *Deploying Device Level Ring within a Converged Plantwide Ethernet Architecture*. 2019. URL: https://belorg.by/wp-content/uploads/rockwell/td/enet-td015_-en-p.pdf (visited on 23/06/2021) (cit. on pp. 98, 100).

[15] Rockwell Automation. *Deploying Network Security within a Converged Plantwide Ethernet Architecture*. 2018. URL: https://literature.rockwellautomation.com/idc/groups/literature/documents/td/enet-td019_-en-p.pdf (visited on 23/06/2021) (cit. on p. 76).

[16] Rockwell Automation. *Industrial Security: Protecting Networks and Facilities against a Fast-Changing Threat Landscape*. 2016. URL: https://www.researchgate.net/profile/Ljubomir-Jacic/post/Results-from-implementation-of-the-Internet-of-Things-Does-it-really-has-an-advantage-for-the-industry/attachment/59d63d8579197b807799a502/AS%3A420045256708096%401477158001350/download/Industrial+Security.pdf (visited on 23/06/2021) (cit. on p. 52).

[17] Giovanni Barbieri et al. 'Sorry, Shodan Is Not Enough! Assessing ICS Security via IXP Network Traffic Analysis'. In: *arXiv*. 2nd July 2020. arXiv: 2007.01114. URL: http://arxiv.org/abs/2007.01114 (visited on 03/10/2020) (cit. on pp. 23, 40, 46, 107).

[18] Andrew Baumann et al. 'The Multikernel: A New OS Architecture for Scalable Multicore Systems'. In: *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles (SOSP '09)*. Big Sky, Montana, USA: ACM Press, 2009, p. 29. ISBN: 978-1-60558-752-3. DOI: 10.1145/1629575.1629579. URL: http://portal.acm.org/citation.cfm?doid=1629575.1629579 (visited on 04/06/2021) (cit. on pp. 101, 107).

[19] Amine Belqruch and Abdelilah Maach. 'SCADA Security Using SSH Honeypot'. In: *Proceedings of the 2nd International Conference on Networking, Information Systems & Security*. 2019, pp. 1–5. DOI: 10.1145/3320326.3320328. URL: http://dl.acm.org/citation.cfm?doid=3320326.3320328 (visited on 18/06/2019) (cit. on p. 54).

[20] Arnar Birgisson et al. 'Macaroons: Cookies with Contextual Caveats for Decentralized Authorization in the Cloud'. In: *Proceedings of the 2014 Network and Distributed System Security Symposium (NDSS '14)*. San Diego, CA: Internet Society, 2014. DOI: 10.14722/ndss.2014.23212. URL: https://www.ndss-symposium.org/ndss2014/programme/macaroons-cookies-contextual-caveats-decentralized-authorization-cloud/ (visited on 21/04/2020) (cit. on pp. 33, 34, 80, 83).

[21] Rakesh B. Bobba et al. 'Detecting False Data Injection Attacks on DC State Estimation'. In: *Proceedings of the 1st Workshop on Secure Control Systems (CPSWEEK)*. 2010. URL: https://tcipg.org/sites/default/files/papers/Bobba-SCS10-final-web.pdf (visited on 23/06/2021) (cit. on p. 21).

[22] Alan C. Bomberger et al. 'The KeyKOS Nanokernel Architecture'. In: *Proceedings of the USENIX Workshop on Micro-Kernels and Other Kernel Architectures*. 1992. URL: https://www.cs.utexas.edu/~lorenzo/corsi/439/ref/keykos.pdf (visited on 23/06/2021) (cit. on p. 76).

[23] Joseph Bonneau et al. 'Passwords and the Evolution of Imperfect Authentication'. In: *Communications of the ACM* 58.7 (2015). ISSN: 00010782. DOI: 10.1145/2699390. URL: http://dl.acm.org/citation.cfm?doid=2797100.2699390 (visited on 17/06/2019) (cit. on p. 41).

[24] Mark Bowden. 'The Worm That Nearly Ate the Internet'. In: *The New York Times. Opinion* (29th June 2019). ISSN: 0362-4331. URL: https://perma.cc/WU7P-LBAD (visited on 30/10/2019) (cit. on p. 42).

[25] *BrickerBot Permanent Denial-of-Service Attack*. Cybersecurity and Infrastructure Security Agency (CISA). 2017. URL: https://www.us-cert.gov/ics/alerts/ICS-ALERT-17-102-01A (visited on 30/10/2019) (cit. on p. 42).

[26] Marcus Brinkmann et al. 'ALPACA: Application Layer Protocol Confusion - Analyzing and Mitigating Cracks in TLS Authentication'. In: *Proceedings of the 30th USENIX Security Symposium (USENIX '21)*. 2021. URL: https://alpaca-attack.com/ALPACA.pdf (visited on 23/06/2021) (cit. on p. 31).

[27] Sharon Brizinov. *Lingering RTA ENIP Stack Vulnerability Poses Risk to ICS Devices*. Claroty. 17th Nov. 2020. URL: https://www.claroty.com/2020/11/17/blog-research-rta-enip-stack-vulnerability/ (visited on 30/11/2020) (cit. on p. 72).

[28] Eric Byres and Justin Lowe. 'The Myths and Facts behind Cyber Security Risks for Industrial Control Systems'. In: *Proceedings of the VDE Kongress* 116 (2004). URL: https://www.researchgate.net/publication/237240867_The_Myths_and_Facts_behind_Cyber_Security_Risks_for_Industrial_Control_Systems (visited on 23/06/2021) (cit. on p. 21).

[29] Olivier Cabana et al. 'Detecting, Fingerprinting and Tracking Reconnaissance Campaigns Targeting Industrial Control Systems'. In: *Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Ed. by Roberto Perdisci et al. 2019. DOI: 10.1007/978-3-030-22038-9_5. URL: http://link.springer.com/10.1007/978-3-030-22038-9_5 (visited on 09/08/2019) (cit. on pp. 24, 65–67, 69, 70).

[30] *Censys*. 2020. URL: http://censys.io/ (visited on 02/12/2019) (cit. on pp. 23, 39, 65).

[31] Thomas Chen and Saeed Abu-Nimeh. 'Lessons from Stuxnet'. In: *Computer* 44.4 (2011). ISSN: 0018-9162. DOI: 10.1109/MC.2011.115. URL: http://ieeexplore.ieee.org/document/5742014/ (visited on 14/06/2019) (cit. on pp. 21, 61, 62).

[32] Michelle S. Chong, Henrik Sandberg and Andre M. H. Teixeira. 'A Tutorial Introduction to Security and Privacy for Cyber-Physical Systems'. In: *Proceedings of the 18th European Control Conference (ECC '19)*. Naples, Italy: IEEE, June 2019, pp. 968–978. ISBN: 978-3-907144-00-8. DOI: 10.23919/ECC.2019.8795652. URL: https://ieeexplore.ieee.org/document/8795652/ (visited on 22/08/2019) (cit. on p. 21).

[33] Catalin Cimpanu. *BrickerBot Author Retires Claiming to Have Bricked over 10 Million IoT Devices*. BleepingComputer. 2017. URL: https://perma.cc/6WUV-99VG (visited on 30/10/2019) (cit. on p. 42).

[34] Abraham A. Clements, Naif Saleh Almakhdhub and Saurabh Bagchi. 'ACES: Automatic Compartments for Embedded Systems'. In: *Proceedings of the 27th USENIX Security Symposium (USENIX '18)*. 2018. URL: https://www.usenix.org/system/files/conference/usenixsecurity18/sec18-clements.pdf (visited on 23/06/2021) (cit. on pp. 29, 73, 102).

[35] Juli Clover. *Apple Now Has 1.3 Billion Active Devices Worldwide*. 2018. URL: https://perma.cc/X599-Q3N8 (visited on 13/11/2019) (cit. on p. 57).

[36] Ben Collier et al. 'Booting the Booters: Evaluating the Effects of Police Interventions in the Market for Denial-of-Service Attacks'. In: *Proceedings of the Internet Measurement Conference (IMC '19)*. 2019. DOI: 10.1145/3355369.3355592. URL: http://dl.acm.org/citation.cfm?doid=3355369.3355592 (visited on 31/10/2019) (cit. on pp. 43, 45, 54).

[37] Fernando J. Corbató and Victor A. Vyssotsky. 'Introduction and Overview of the Multics System'. In: *Proceedings of the November 30–December 1, 1965, Fall Joint Computer Conference, Part I (AFIPS '65)*. Las Vegas, Nevada: ACM Press, 1965, p. 185. DOI: 10.1145/1463891.1463912. URL: http://portal.acm.org/citation.cfm?doid=1463891.1463912 (visited on 17/06/2021) (cit. on p. 26).

[38] *CRASHOVERRIDE: Analysis of the Threat to Electric Grid Operations*. Dragos Inc. 2017. URL: https://perma.cc/E7K5-9T8M (visited on 08/11/2019) (cit. on pp. 13, 15, 21, 41, 62, 71).

[39] *CTSRD-CHERI/Cheri-Cpu*. Capability Hardware Enhanced RISC Instructions, 2021. URL: https://github.com/CTSRD-CHERI/cheri-cpu (visited on 25/06/2021) (cit. on p. 86).

[40] *CTSRD-CHERI/Cheribsd*. Capability Hardware Enhanced RISC Instructions, 5th May 2021. URL: https://github.com/CTSRD-CHERI/cheribsd (visited on 16/05/2021) (cit. on p. 85).

[41] *CTSRD-CHERI/Cherios*. Capability Hardware Enhanced RISC Instructions, 27th Nov. 2020. URL: https://github.com/CTSRD-CHERI/cherios (visited on 16/05/2021) (cit. on p. 76).

[42] *CTSRD-CHERI/FETT*. Capability Hardware Enhanced RISC Instructions, 17th May 2021. URL: https://github.com/CTSRD-CHERI/FETT (visited on 16/05/2021) (cit. on p. 91).

[43] *CTSRD-CHERI/Flute*. Capability Hardware Enhanced RISC Instructions, 8th May 2021. URL: https://github.com/CTSRD-CHERI/Flute (visited on 16/05/2021) (cit. on p. 91).

[44] *CTSRD-CHERI/FreeRTOS-Demos-CHERI-RISC-V*. Capability Hardware Enhanced RISC Instructions, 15th May 2021. URL: https://github.com/CTSRD-CHERI/FreeRTOS-Demos-CHERI-RISC-V (visited on 16/05/2021) (cit. on p. 87).

[45] *CTSRD-CHERI/Llvm-Project*. Capability Hardware Enhanced RISC Instructions, 10th May 2021. URL: https://github.com/CTSRD-CHERI/llvm-project (visited on 16/05/2021) (cit. on p. 91).

[46] *CVE - Modbus*. MITRE CVE Search. 17th May 2021. URL: https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=modbus (visited on 16/05/2021) (cit. on p. 94).

[47] *Cyber-Attack against Ukrainian Critical Infrastructure*. Cybersecurity and Infrastructure Security Agency (CISA). 2016. URL: https://perma.cc/FQ7X-SQQ7 (visited on 19/06/2019) (cit. on p. 21).

[48] *Cyber-Attack on Hydro*. Norsk Hydro. 2019. URL: https://perma.cc/Z4NV-W9WP (visited on 12/03/2020) (cit. on pp. 14, 53, 59, 68).

[49] Markus Dahlmanns et al. 'Easing the Conscience with OPC UA: An Internet-Wide Study on Insecure Deployments'. In: *Proceedings of the Internet Measurement Conference (IMC '20)*. 27th Oct. 2020. DOI: 10.1145/3419394.3423666. URL: https://dl.acm.org/doi/10.1145/3419394.3423666 (visited on 30/10/2020) (cit. on pp. 39, 52).

[50] Brooks Davis et al. 'CheriABI: Enforcing Valid Pointer Provenance and Minimizing Pointer Privilege in the POSIX C Run-Time Environment'. In: *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS '19)*. Providence, RI, USA: ACM Press, 2019. ISBN: 978-1-4503-6240-5. DOI: 10.1145/3297858.3304042. URL: http://dl.acm.org/citation.cfm?doid=3297858.3304042 (visited on 04/09/2019) (cit. on p. 81).

[51] Peter J. Denning. 'Virtual Memory'. In: *ACM Computing Surveys (CSUR)* 2.3 (1970), pp. 153–198. URL: https://dl.acm.org/doi/pdf/10.1145/356571.356573 (cit. on pp. 25, 26).

[52] Jack B. Dennis and Earl C. Van Horn. 'Programming Semantics for Multiprogrammed Computations'. In: *Communications of the ACM* 9.3 (Mar. 1966), pp. 143–155. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/365230.365252. URL: https://dl.acm.org/doi/10.1145/365230.365252 (visited on 17/06/2021) (cit. on p. 25).

[53] *Description of Available Datasets*. Cambridge Cybercrime Centre. 2020. URL: https://www.cambridgecybercrime.uk/datasets.html (visited on 03/12/2020) (cit. on pp. 42, 54, 55, 60, 65, 69).

[54] *Digital Security by Design Challenge*. UK Research and Innovation. 2021. URL: https://www.ukri.org/our-work/our-main-funds/industrial-strategy-challenge-fund/artificial-intelligence-and-data-economy/digital-security-by-design-challenge/ (visited on 25/06/2021) (cit. on p. 28).

[55] *Discovery and Global Services*. OPC Foundation, 2017. URL: https://reference.opcfoundation.org/v104/GDS/docs/ (visited on 23/06/2021) (cit. on pp. 34, 95, 102).

[56] Michael Dodson, Alastair R. Beresford and Daniel R. Thomas. 'When Will My PLC Support Mirai? The Security Economics of Large-Scale Attacks against ICS'. In: *Proceedings of the 2020 APWG Symposium on Electronic Crime Research (eCrime '20)*. 2020. URL: https://www.cl.cam.ac.uk/~md403/papers/ics_security_economics_ecrime_2020.pdf (visited on 23/06/2021) (cit. on pp. 13, 16, 23, 38).

[57] Michael Dodson, Mikael Vingaard and Alastair R. Beresford. 'Using Global Honeypot Networks to Detect Targeted ICS Attacks'. In: *Proceedings of the 12th International Conference on Cyber Conflict (CyCon '20)*. 2020. URL: https://ieeexplore.ieee.org/abstract/document/9131734/ (visited on 23/06/2021) (cit. on pp. 16, 54, 62).

[58] Michael Dodson et al. 'CHERI Macaroons: Efficient, Host-Based Access Control for Cyber-Physical Systems'. In: *Proceedings of the IEEE European Symposium on Security and Privacy Workshops (EuroS&PW '20)*. 2020. DOI: 10.17863/CAM.54214. URL: https://ieeexplore.ieee.org/abstract/document/9229739/ (visited on 23/06/2021) (cit. on pp. 17, 72, 80).

[59] Danny Dolev and Andrew Yao. 'On the Security of Public Key Protocols'. In: *IEEE Transactions on Information Theory* 29.2 (1983). URL: https://ieeexplore.ieee.org/abstract/document/1056650/ (visited on 23/06/2021) (cit. on p. 73).

[60] Zakir Durumeric, Eric Wustrow and J. Alex Halderman. 'Zmap: Fast Internet-Wide Scanning and Its Security Applications'. In: *Proceedings of the 22nd USENIX Security Symposium (USENIX '13)*. 2013. URL: https://zmap.io/paper.pdf (visited on 18/05/2020) (cit. on pp. 23, 60).

[61] Zakir Durumeric et al. 'A Search Engine Backed by Internet-Wide Scanning'. In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15)*. 2015. DOI: 10.1145/2810103.2813703. URL: http://dl.acm.org/citation.cfm?doid=2810103.2813703 (visited on 26/02/2019) (cit. on pp. 39, 48, 60).

[62] Carl Ellison et al. 'SPKI Certificate Theory'. In: *IETF RFC 2693* (1999). URL: https://www.ietf.org/rfc/rfc2693.txt (visited on 05/06/2021) (cit. on p. 33).

[63] *Equation: Advanced Cyberespionage Group Has All the Tricks in the Book, and More*. Symantec. 2015. URL: https://perma.cc/S9MF-6BK5 (visited on 30/10/2019) (cit. on pp. 42, 45).

[64] Joe Falco, Albert Wavering and Frederick Proctor. *IT Security for Industrial Control Systems*. NIST IR 6859. Gaithersburg, MD: National Institute of Standards and Technology, 2002, NIST IR 6859. DOI: 10.6028/NIST.IR.6859. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/IR/nistir6859.pdf (visited on 13/06/2019) (cit. on p. 21).

[65] Wenjun Fan et al. 'Enabling an Anatomic View to Investigate Honeypot Systems: A Survey'. In: *IEEE Systems Journal* 12.4 (2018). ISSN: 2373-7816. DOI: 10.1109/JSYST.2017.2762161 (cit. on p. 24).

[66] Pietro Ferretti, Marcello Pogliani and Stefano Zanero. 'Characterizing Background Noise in ICS Traffic through a Set of Low Interaction Honeypots'. In: *Proceedings of the 5th ACM Workshop on Cyber-Physical Systems Security & Privacy (CPS-SPC '19)*. 2019. DOI: 10.1145/3338499.3357361 (cit. on pp. 24, 54, 65–67, 69, 70).

[67] Nathaniel Wesley Filardo et al. 'Cornucopia: Temporal Safety for CHERI Heaps'. In: *Proceedings of the 2020 IEEE Symposium on Security and Privacy (S&P '20)*. 2020, pp. 608–625. URL: https://ieeexplore.ieee.org/abstract/document/9152640/ (visited on 23/06/2021) (cit. on p. 108).

[68] David Formby, Srikar Durbha and Raheem Beyah. 'Out of Control: Ransomware for Industrial Control Systems'. In: *Proceedings of the RSA Conference (RSA '17)*. Vol. 4. 2017. URL: https://perma.cc/XD8V-LP5M (cit. on pp. 14, 45, 55, 58, 59, 61, 62).

[69] *FreeRTOS - Market Leading RTOS (Real Time Operating System) for Embedded Systems with Internet of Things Extensions*. FreeRTOS. 17th May 2021. URL: https://www.freertos.org/ (visited on 16/05/2021) (cit. on p. 87).

[70] *Frequently Asked Questions on seL4*. seL4. 17th May 2021. URL: https://docs.sel4.systems/projects/sel4/frequently-asked-questions.html (visited on 16/05/2021) (cit. on p. 100).

[71] Laurence Frost et al. *Renault-Nissan Resumes Nearly All Production after Cyber Attack*. Reuters. 2017. URL: https://perma.cc/Y2J7-8PXU (visited on 12/03/2020) (cit. on p. 59).

[72] Simon Fürst. 'Autosar the next Generation – The Adaptive Platform'. In: *CARS@EDCC* (2015). URL: http://conf.laas.fr/cars/CARS/CARS@EDCC2015_files/AUTOSAR_CARS%40EDCC%202015.pdf (visited on 23/06/2021) (cit. on p. 97).

[73] Simon Fürst et al. 'AUTOSAR - A Worldwide Standard Is on the Road'. In: *International VDI Congress Electronic Systems for Vehicles* (2009). URL: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.462.325&rep=rep1&type=pdf (visited on 23/06/2021) (cit. on p. 97).

[74] Gartner. *Global Connected IoT Devices by Type 2017 and 2018*. Statista. 2019. URL: https://perma.cc/6SW2-4LH9 (visited on 13/11/2019) (cit. on p. 57).

[75] Gartner. *PCs Installed Base Worldwide 2013-2019*. Statista. 2019. URL: https://perma.cc/CSV6-5VF5 (visited on 13/11/2019) (cit. on p. 57).

[76] Oliver Gasser et al. 'The Amplification Threat Posed by Publicly Reachable BACnet Devices'. In: *Journal of Cyber Security and Mobility* 6.1 (2017). ISSN: 2245-1439. DOI: 10.13052/jcsm2245-1439.614. URL: http://www.riverpublishers.com/journal_read_html_article.php?j=JCSM/6/1/4 (visited on 04/04/2019) (cit. on p. 46).

[77] Jairo Giraldo et al. 'A Survey of Physics-Based Attack Detection in Cyber-Physical Systems'. In: *ACM Computing Surveys (CSUR)* 51.4 (2018). DOI: 10.1145/3203245. URL: http://dl.acm.org/citation.cfm?doid=3236632.3203245 (visited on 21/08/2019) (cit. on pp. 21, 22, 71, 74, 102).

[78] Jairo Giraldo et al. 'DARIA: Designing Actuators to Resist Arbitrary Attacks against Cyber-Physical Systems'. In: *Proceedings of the 2020 IEEE European Symposium on Security and Privacy (EuroS&P '20)*. 2020. URL: https://users.soe.ucsc.edu/~alacarde/papers/DARIA.pdf (visited on 12/10/2020) (cit. on p. 22).

[79] Jairo Giraldo et al. 'The More the Merrier: Adding Hidden Measurements to Secure Industrial Control Systems'. In: *Proceedings of the 7th Symposium on Hot Topics in the Science of Security (HotSoS '20)*. Lawrence Kansas: ACM, 21st Sept. 2020, pp. 1–10. ISBN: 978-1-4503-7561-0. DOI: 10.1145/3384217.3385624. URL: https://dl.acm.org/doi/10.1145/3384217.3385624 (visited on 01/06/2021) (cit. on p. 22).

[80] GReAT. *Schroedinger's Pet(Ya)*. SecureList. 2017. URL: https://perma.cc/U6HH-HDZJ (visited on 30/10/2019) (cit. on p. 42).

[81] Andy Greenberg. 'The Colonial Pipeline Hack Is a New Extreme for Ransomware'. In: *Wired* (2021). ISSN: 1059-1028. URL: https://www.wired.com/story/colonial-pipeline-ransomware-attack (visited on 15/06/2021) (cit. on p. 14).

[82] Guannan Guo et al. 'A Survey of Industrial Control System Devices on the Internet'. In: *Proceedings of the 2018 International Conference on Internet of Things, Embedded Systems and Communications (IINTEC '18)*. IEEE, Dec. 2018, pp. 197–202. DOI: 10.1109/IINTEC.2018.8695276 (cit. on p. 48).

[83] Norm Hardy. 'The Confused Deputy (or Why Capabilities Might Have Been Invented)'. In: *ACM SIGOPS Operating Systems Review* 22.4 (Oct. 1988). ISSN: 0163-5980. DOI: 10.1145/54289.871709. URL: https://dl.acm.org/doi/10.1145/54289.871709 (visited on 05/10/2020) (cit. on pp. 27, 74, 77).

[84] Norman Hardy. 'KeyKOS Architecture'. In: *ACM SIGOPS Operating Systems Review* 19.4 (Oct. 1985), pp. 8–25. ISSN: 0163-5980. DOI: 10.1145/858336.858337. URL: https://dl.acm.org/doi/10.1145/858336.858337 (visited on 06/10/2020) (cit. on p. 74).

[85] Gregory Hawkridge et al. 'Tying Together Solutions for Digital Manufacturing: Assessment of Connectivity Technologies & Approaches'. In: *Proceedings of the 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA '19)*. IEEE, 2019 (cit. on p. 98).

[86] Gernot Heiser. *The seL4 Microkernel – An Introduction*. Whitepaper. 2020. URL: https://sel4.systems/About/seL4-whitepaper.pdf (visited on 23/06/2021) (cit. on pp. 77, 82).

[87] Cormac Herley. 'More Is Not the Answer'. In: *IEEE Security and Privacy (S&P)* 12.1 (2014). ISSN: 1540-7993. DOI: 10.1109/MSP.2013.134. URL: http://ieeexplore.ieee.org/document/6671584/ (visited on 17/06/2019) (cit. on pp. 41, 52).

[88] Cormac Herley. 'So Long, and No Thanks for the Externalities: The Rational Rejection of Security Advice by Users'. In: *Proceedings of the 2009 Workshop on New Security Paradigms (NSPW '09)*. 2009, pp. 133–144. URL: https://dl.acm.org/doi/abs/10.1145/1719030.1719050 (visited on 23/06/2021) (cit. on pp. 40, 52).

[89] Matthias Hille et al. 'SemperOS: A Distributed Capability System'. In: *Proceedings of the 2019 USENIX Annual Technical Conference (USENIX ATC '19)*. 2019, pp. 709–722. URL: https://www.usenix.org/system/files/atc19-hille.pdf (visited on 05/10/2020) (cit. on pp. 81, 101, 107).

[90] David G. Holmberg. *BACnet Wide Area Network Security Threat Assessment*. NIST IR 7009. Gaithersburg, MD: National Institute of Standards and Technology, 2003. DOI: 10.6028/NIST.IR.7009. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/IR/nistir7009.pdf (visited on 28/03/2019) (cit. on p. 39).

[91] Anatol W. Holt. 'Program Organization and Record Keeping for Dynamic Storage Allocation'. In: *Communications of the ACM* 4.10 (Oct. 1961), pp. 422–431. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/366786.366795. URL: https://dl.acm.org/doi/10.1145/366786.366795 (visited on 17/06/2021) (cit. on p. 26).

[92] *Huawei Home Routers in Botnet Recruitment*. Check Point Research. 21st Dec. 2017. URL: https://research.checkpoint.com/2017/good-zero-day-skiddie/ (visited on 30/11/2019) (cit. on p. 68).

[93] Alice Hutchings and Sergio Pastrana. 'Understanding eWhoring'. In: *Proceedings of the 2019 IEEE European Symposium on Security and Privacy (EuroS&P '19)*. 2019. DOI: 10.17863/CAM.39108. URL: https://dl.acm.org/authorize?N695003 (visited on 08/11/2019) (cit. on pp. 54, 55).

[94] *IoT: Driving Verticals to Digitization*. Huawei. 2019. URL: https://www.huawei.com/minisite/iot/en/ (visited on 31/10/2019) (cit. on p. 57).

[95] *ISA-62443: Security for Industrial Automation and Control Systems*. International Society of Automation. URL: https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa99 (visited on 19/06/2019) (cit. on p. 21).

[96] Bruce Jacob, David Wang and Spencer Ng. *Memory Systems: Cache, DRAM, Disk*. Morgan Kaufmann, 2010 (cit. on p. 26).

[97] Michael Jones, John Bradley and Nat Sakimura. *JSON Web Token (JWT)*. RFC7519. May 2015. DOI: 10.17487/RFC7519. URL: https://www.rfc-editor.org/info/rfc7519 (visited on 18/06/2020) (cit. on pp. 32, 34, 95, 102).

[98] U.S. Department of Justice. *Former Operator of Illegal Booter Services Pleads Guilty to Conspiracy to Commit Computer Damage and Abuse*. 27th Feb. 2019. URL: https://perma.cc/HC5G-KYL4 (visited on 28/10/2019) (cit. on pp. 43, 45, 58).

[99] U.S. Department of Justice. *Former Systems Administrator Sentenced to Prison for Hacking into Industrial Facility Computer System*. 16th Feb. 2017. URL: https://perma.cc/PWP7-SPKA (visited on 28/10/2019) (cit. on p. 58).

[100] Ori Karliner. *FreeRTOS TCP/IP Stack Vulnerabilities*. Zimperium Mobile Security Blog. 4th Dec. 2018. URL: https://blog.zimperium.com/freertos-tcpip-stack-vulnerabilities-details/ (visited on 18/12/2019) (cit. on p. 61).

[101] Stamatis Karnouskos. 'Stuxnet Worm Impact on Industrial Cyber-Physical System Security'. In: *Proceedings of the 37th Conference of the IEEE Industrial Electronics Society (IECON '11)*. 2011. DOI: 10.1109/IECON.2011.6120048. URL: http://ieeexplore.ieee.org/document/6120048/ (visited on 14/06/2019) (cit. on pp. 13, 14, 21, 41, 52, 53).

[102] Gregg Keizer. *Conficker Cashes in, Installs Spam Bots and Scareware*. 17th Apr. 2009. URL: https://www.computerworld.com/article/2524137/conficker-cashes-in--installs-spam-bots-and-scareware.html (visited on 04/11/2019) (cit. on p. 42).

[103] Arslan Khan et al. 'M2MON: Building an MMIO-Based Security Reference Monitor for Unmanned Vehicles'. In: *Proceedings of the 30th USENIX Security Symposium (USENIX '21)*. 2021. URL: https://atc.usenix.org/system/files/sec21fall-khan-arslan.pdf (visited on 05/11/2021) (cit. on pp. 29, 102).

[104] Gerwin Klein et al. 'Comprehensive Formal Verification of an OS Microkernel'. In: *ACM Transactions on Computer Systems* 32.1 (26th Feb. 2014), pp. 1–70. ISSN: 07342071. DOI: 10.1145/2560537. URL: http://dl.acm.org/citation.cfm?doid=2584468.2560537 (visited on 13/08/2019) (cit. on pp. 76, 77, 81, 82, 100).

[105] Johannes Klick et al. 'Internet-Facing PLCs - a New Back Orifice'. In: *Blackhat USA*. 2015. URL: https://perma.cc/XK4N-UPV4 (cit. on p. 58).

[106] Eric D Knapp and Joel Thomas Langill. *Industrial Network Security: Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems*. Syngress, 2014 (cit. on p. 22).

[107] Brian Krebs. *DDoS-for-Hire Service Webstresser Dismantled*. Krebs on Security. 2018. URL: https://perma.cc/586G-TKGG (visited on 28/10/2019) (cit. on pp. 43, 45, 58).

[108] Brian Krebs. *Mirai Botnet Authors Avoid Jail Time*. Krebs on Security. 2018. URL: https://perma.cc/ECB4-T8FC (visited on 31/10/2019) (cit. on p. 45).

[109] David Kushner. 'The Real Story of Stuxnet'. In: *IEEE Spectrum* 50.3 (2013). ISSN: 0018-9235. DOI: 10.1109/MSPEC.2013.6471059. URL: http://ieeexplore.ieee.org/document/6471059/ (visited on 14/06/2019) (cit. on p. 21).

[110] *L4 Runtime Environment (L4Re)*. 28th Dec. 2020. URL: https://l4re.org/doc/ (visited on 02/02/2021) (cit. on pp. 81, 100).

[111] Ben Laurie. 'Certificate Transparency: Public, Verifiable, Append-Only Logs'. In: *ACM Queue* 12.8 (Aug. 2014), pp. 10–19. ISSN: 1542-7730, 1542-7749. DOI: 10.1145/2668152.2668154. URL: https://dl.acm.org/doi/10.1145/2668152.2668154 (visited on 24/06/2021) (cit. on p. 31).

[112] *Layered Software Architecture*. 53. AUTOSAR, 2019. URL: https://www.autosar.org/fileadmin/user_upload/standards/classic/4-3/AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf (visited on 23/06/2021) (cit. on p. 97).

[113] Robert M. Lee. *CRASHOVERRIDE: Analyzing the Malware That Attacks Power Grids*. Dragos Inc. 2017. URL: https://perma.cc/JV89-A9T4 (visited on 19/06/2019) (cit. on p. 61).

[114] Éireann Leverett, Richard Clayton and Ross Anderson. 'Standardisation and Certification of the 'Internet of Things''. In: *Proceedings of the 16th Workshop on the Economics of Information Security (WEIS '17)*. 2017. URL: https://perma.cc/5Y9R-9DD3 (cit. on pp. 13, 61, 71, 108).

[115] Éireann P. Leverett. 'Quantitatively Assessing and Visualising Industrial System Attack Surfaces'. In: *University of Cambridge MPhil Thesis* (2011). URL: https://perma.cc/83Z9-Q5J9 (visited on 26/02/2019) (cit. on pp. 14, 21, 23, 37, 39, 41, 63, 71).

[116] Henry M. Levy. *Capability-Based Computer Systems*. Digital Press, 2014. URL: https://homes.cs.washington.edu/~levy/capabook/Preface.pdf (visited on 23/06/2021) (cit. on pp. 81, 82).

[117] Gaoqi Liang et al. 'The 2015 Ukraine Blackout: Implications for False Data Injection Attacks'. In: *IEEE Transactions on Power Systems* 32.4 (2017). ISSN: 0885-8950, 1558-0679. DOI: 10.1109/TPWRS.2016.2631891. URL: http://ieeexplore.ieee.org/document/7752958/ (visited on 19/06/2019) (cit. on p. 21).

[118] Arm Limited. *Arm Architecture Reference Manual Supplement - Morello for A-Profile Architecture*. DDI0606. 14th Dec. 2020. URL: https://developer.arm.com/documentation/ddi0606/latest (cit. on p. 28).

[119] Arm Limited. *Arm Cortex-M55 Processor Technical Reference Manual Revision R0p2*. Technical Reference Manual. 2020. URL: https://developer.arm.com/documentation/101051/0002 (cit. on p. 27).

[120] Qiang Liu et al. 'An Access Control Model for Resource Sharing Based on the Role-Based Access Control Intended for Multi-Domain Manufacturing Internet of Things'. In: *IEEE Access* 5 (2017). ISSN: 2169-3536. DOI: 10.1109/ACCESS.2017.2693380. URL: http://ieeexplore.ieee.org/document/7904674/ (visited on 14/06/2019) (cit. on p. 22).

[121] Yao Liu, Peng Ning and Michael K. Reiter. 'False Data Injection Attacks against State Estimation in Electric Power Grids'. In: *ACM Transactions on Information and System Security* 14.1 (May 2011), pp. 1–33. ISSN: 1094-9224, 1557-7406. DOI: 10.1145/1952982.1952995. URL: https://dl.acm.org/doi/10.1145/1952982.1952995 (visited on 01/06/2021) (cit. on p. 21).

[122] Javier Lopez and Juan E. Rubio. 'Access Control for Cyber-Physical Systems Interconnected to the Cloud'. In: *Computer Networks* 134 (2018). ISSN: 13891286. DOI: 10.1016/j.comnet.2018.01.037. URL: https://linkinghub.elsevier.com/retrieve/pii/S1389128618300501 (visited on 14/06/2019) (cit. on p. 22).

[123] Gordon Fyodor Lyon. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Sunnyvale, CA, USA: Insecure, 2009. 468 pp. ISBN: 978-0-9799587-1-7 (cit. on p. 23).

[124] Anna Lyons et al. 'Scheduling-Context Capabilities: A Principled, Light-Weight Operating-System Mechanism for Managing Time'. In: *Proceedings of the 13th European Conference on Computer Systems (EuroSys '18)*. 23rd Apr. 2018. DOI: 10.1145/3190508.3190539. URL: https://dl.acm.org/doi/10.1145/3190508.3190539 (visited on 03/10/2020) (cit. on p. 100).

[125] *Malware Initial Findings Report (MIFR) - 10130295*. Cybersecurity and Infrastructure Security Agency (CISA), 2017. URL: https://perma.cc/QV8W-QAKM (visited on 11/08/2019) (cit. on p. 42).

[126] William Martin. 'Honey Pots and Honey Nets - Security through Deception'. In: *SANS Institute* (2003). URL: https://www.sans.org/reading-room/whitepapers/attacking/honey-pots-honey-nets-security-deception-41 (visited on 09/12/2019) (cit. on p. 24).

[127] Duncan McFarlane et al. 'Digital Manufacturing on a Shoestring: Low Cost Digital Solutions for SMEs'. In: *Proceedings of the 2019 International Workshop on Service Orientation in Holonic and Multi-Agent Manufacturing*. Ed. by Theodor Borangiu et al. 2019. DOI: 10.1007/978-3-030-27477-1_4. URL: http://link.springer.com/10.1007/978-3-030-27477-1_4 (visited on 15/08/2019) (cit. on pp. 57, 98).

[128] Stephen McLaughlin et al. 'The Cybersecurity Landscape in Industrial Control Systems'. In: *Proceedings of the IEEE* 104.5 (2016). ISSN: 0018-9219, 1558-2256. DOI: 10.1109/JPROC.2015.2512235. URL: http://ieeexplore.ieee.org/document/7434576/ (visited on 12/06/2019) (cit. on p. 20).

[129] Microsoft. *Azure Application Architecture Guide*. 2020. URL: https://docs.microsoft.com/en-us/azure/architecture/guide/ (visited on 10/06/2020) (cit. on pp. 34, 97, 102).

[130] Microsoft. *Azure Sphere*. 2019. URL: https://azure.microsoft.com/en-us/services/azure-sphere/ (visited on 31/10/2019) (cit. on pp. 34, 57, 68, 75, 96, 102).

[131] Microsoft. *Grant Limited Access to Data with Shared Access Signatures (SAS) - Azure Storage*. 2020. URL: https://docs.microsoft.com/en-us/azure/storage/common/storage-sas-overview (visited on 11/05/2021) (cit. on pp. 34, 97, 102).

[132] Microsoft. *Microsoft Azure IoT Reference Architecture*. Version 2.1. Microsoft, 2018. URL: http://download.microsoft.com/download/A/4/D/A4DAD253-BC21-41D3-B9D9-87D2AE6F0719/Microsoft_Azure_IoT_Reference_Architecture.pdf (visited on 10/06/2020) (cit. on pp. 96, 97).

[133] Carrie Mihalcik. *Microsoft Aims for 1 Billion Devices Running Windows 10*. CNET. URL: https://www.cnet.com/news/microsoft-aims-for-1-billion-devices-running-windows-10/ (visited on 26/02/2020) (cit. on p. 61).

[134] Charlie Miller and Chris Valasek. 'A Survey of Remote Automotive Attack Surfaces'. In: *Black Hat USA*. 2014. URL: https://www.hardworkingtrucks.com/wp-content/uploads/sites/6/2014/09/Remote_Automotive_Attack_Surfaces.pdf (visited on 23/06/2021) (cit. on pp. 14, 15, 71).

[135] Mark S. Miller, Ka-Ping Yee and Jonathan Shapiro. *Capability Myths Demolished*. SRL2003-02. Johns Hopkins University Systems Research Laboratory, 2003. URL: http://www-users.cselabs.umn.edu/classes/Fall-2019/csci5271/papers/SRL2003-02.pdf (visited on 30/09/2020) (cit. on pp. 27, 31, 72, 75–77).

[136] Ariana Mirian et al. 'An Internet-Wide View of ICS Devices'. In: *Proceedings of the 14th Conference on Privacy, Security and Trust (PST '16)*. 2016. DOI: 10.1109/PST.2016.7906943 (cit. on pp. 13, 14, 21, 23, 24, 37–41, 48, 54, 60, 63, 65–67, 69, 70).

[137] Modbus. *MODBUS Application Protocol Specification*. v1.1b3. 2012. URL: https://modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf (visited on 23/06/2021) (cit. on p. 87).

[138] Modbus. *MODBUS Messaging on TCP/IP Implementation Guide*. v1.0b. 2006. URL: https://modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf (visited on 23/06/2021) (cit. on p. 87).

[139] Tyler Moore and Ross Anderson. 'Economics and Internet Security: A Survey of Recent Analytical, Empirical and Behavioral Research'. In: *Harvard University* TR-03-11 (2011). URL: https://perma.cc/AH7R-XRYT (cit. on p. 52).

[140] John Moulder et al. 'PLC Backplane Analyzer for Field Forensics and Intrusion Detection'. U.S. pat. 9,032,522 B1. Sandia Corporation. 12th May 2015. URL: https://patentimages.storage.googleapis.com/78/70/8c/1b68ce613e64bb/US9032522.pdf (visited on 16/10/2020) (cit. on p. 74).

[141] Steven J. Murdoch. 'Hot or Not: Revealing Hidden Services by Their Clock Skew'. In: *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06)*. 2006. DOI: 10.1145/1180405.1180410. URL: http://portal.acm.org/citation.cfm?doid=1180405.1180410 (visited on 15/02/2019) (cit. on p. 47).

[142] Ben Nahorney. *The Downadup Codex: A Comprehensive Guide to the Threat's Mechanics*. Symantec. 2009. URL: https://perma.cc/3ACC-U5R4 (visited on 08/11/2019) (cit. on p. 42).

[143] Ellen Nakashima and Craig Timberg. *NSA Officials Worried about the Day Its Potent Hacking Tool Would Get Loose. Then It Did.* Washington Post. 2017. URL: https://perma.cc/V8D9-GCHS (visited on 08/11/2019) (cit. on pp. 42, 45).

[144] Roger M. Needham and Michael D. Schroeder. 'Using Encryption for Authentication in Large Networks of Computers'. In: *Communications of the ACM* 21.12 (Dec. 1978), pp. 993–999. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/359657.359659. URL: https://dl.acm.org/doi/10.1145/359657.359659 (visited on 05/06/2021) (cit. on pp. 29, 30).

[145] Roger M. Needham and Robin D. H. Walker. 'The Cambridge CAP Computer and Its Protection System'. In: *ACM SIGOPS Operating Systems Review* 11.5 (1977). URL: https://dl.acm.org/doi/pdf/10.1145/1067625.806541 (visited on 05/11/2020) (cit. on p. 76).

[146] Peter G. Neumann et al. *CHERI: Capability Hardware Enhanced RISC Instructions*. 2020. URL: https://eri-summit.darpa.mil/docs/ERIsummit2020/posters/37P_SSITH_Rebello_CHERI_Poster.pdf (cit. on pp. 28, 91).

[147] *New Petya / NotPetya / ExPetr Ransomware Outbreak*. Kaspersky Daily. 2017. URL: https://perma.cc/BM7P-P2R9 (visited on 30/10/2019) (cit. on p. 42).

[148] *News from the Lab Archive*. F-Secure. 2009. URL: https://perma.cc/A2DK-3JUJ (visited on 30/10/2019) (cit. on p. 42).

[149] Matthias Niedermaier et al. 'You Snooze, You Lose: Measuring PLC Cycle Times under Attacks'. In: *Proceedings of the 12th USENIX Workshop on Offensive Technologies (WOOT '18)*. 2018 (cit. on pp. 37, 39, 60).

[150] *Nmap: The Network Mapper*. NMap Project. 2020. URL: https://nmap.org/ (visited on 02/12/2019) (cit. on pp. 23, 65).

[151] Alexander Nochvay. *Security Research: CODESYS Runtime, a PLC Control Framework*. Kaspersky ICS CERT, 2019. URL: https://perma.cc/325P-N7AV (visited on 08/11/2019) (cit. on pp. 56, 61).

[152] *OARC's DNS Don't-Probe List*. DNS-OARC. 2020. URL: https://www.dns-oarc.net/oarc/services/dontprobe (visited on 15/05/2020) (cit. on p. 60).

[153] *Overview and Concepts*. OPC Foundation, 2017. URL: https://reference.opcfoundation.org/v104/Core/docs/Part1/ (visited on 23/06/2021) (cit. on p. 95).

[154] Sergio Pastrana and Guillermo Suarez-Tangil. 'A First Look at the Crypto-Mining Malware Ecosystem: A Decade of Unrestricted Wealth'. In: *Proceedings of the Internet Measurement Conference (IMC '19)*. 2019. URL: https://dl.acm.org/authorize?N695072 (visited on 09/08/2019) (cit. on pp. 43, 45, 54, 55).

[155] Sergio Pastrana et al. 'CrimeBB: Enabling Cybercrime Research on Underground Forums at Scale'. In: *Proceedings of the 2018 World Wide Web Conference (WWW '18)*. 2018. DOI: 10.1145/3178876.3186178. URL: http://dl.acm.org/citation.cfm?doid=3178876.3186178 (visited on 09/08/2019) (cit. on p. 54).

[156] David A. Patterson and John L. Hennessy. *Computer Organization and Design: The Hardware Software Interface*. Morgan kaufmann, 2016 (cit. on p. 26).

[157] Matthew Peacock. 'Anomaly Detection in BACnet/IP Managed Building Automation Systems'. In: *Edith Cowan University PhD Thesis* (2019). URL: https://ro.ecu.edu.au/theses/2178/ (visited on 23/06/2021) (cit. on p. 39).

[158] *Petya Ransomware Outbreak: Here's What You Need to Know*. Symantec. 2017. URL: https://perma.cc/7JP5-9D6D (visited on 30/10/2019) (cit. on p. 42).

[159] *PLC Cycle Time Influences (Update A)*. Cybersecurity and Infrastructure Security Agency (CISA). 2019. URL: https://www.us-cert.gov/ics/advisories/ICSA-19-106-03 (visited on 29/02/2020) (cit. on p. 67).

[160] Emil Protalinski. *Android Passes 2.5 Billion Monthly Active Devices*. VentureBeat. 7th May 2019. URL: https://perma.cc/2D3D-S7KN (visited on 12/11/2019) (cit. on p. 57).

[161] Niels Provos and Thorsten Holz. *Virtual Honeypots: From Botnet Tracking to Intrusion Detection*. Pearson Education, 2007 (cit. on p. 24).

[162] Bob Radvanovsky. 'Project RUGGEDTRAX SCADA/ICS Analysis'. In: *Infracritical Technical Report* (2015). URL: https://perma.cc/7AN4-KR8K (visited on 08/11/2019) (cit. on p. 54).

[163] Bob Radvanovsky and Jake Brodsky. 'Project SHINE (SHodan INtelligence Extraction)'. In: *Infracritical Technical Report* (2014). URL: https://perma.cc/HA8J-5SNZ (cit. on pp. 14, 23, 37, 39, 41).

[164] Radware. *BrickerBot Results in PDoS (Permanent Denial of Service) Attacks*. 2017. URL: https://perma.cc/3MTJ-7GWL (visited on 30/10/2019) (cit. on p. 42).

[165] Stéphane Raimbault. *Libmodbus*. 14th May 2021. URL: https://github.com/stephane/libmodbus (visited on 16/05/2021) (cit. on p. 87).

[166] *Reference Architecture*. G1. Industrial Internet Consortium, 2019. URL: https://www.iiconsortium.org/IIC_PUB_G1_V1.80_2017-01-31.pdf (visited on 23/06/2021) (cit. on pp. 95, 96).

[167] Alexander Richardson. *Complete Spatial Safety for C and C++ Using CHERI Capabilities*. UCAM-CL-TR-949. University of Cambridge, 2020. URL: https://svr-www-00.cl.cam.ac.uk/techreports/UCAM-CL-TR-949.pdf (visited on 01/10/2020) (cit. on pp. 29, 81, 101).

[168] *Ripple20*. JSOF. 2020. URL: https://www.jsof-tech.com/ripple20/ (visited on 20/06/2020) (cit. on pp. 72, 74, 97).

[169] Lukas Rist et al. *CONPOT ICS/SCADA Honeypot*. 2019. URL: http://conpot.org/ (visited on 08/04/2019) (cit. on pp. 24, 47, 54, 64).

[170] *ROBOTIS E-Manual*. ROBOTIS. 17th May 2021. URL: https://emanual.robotis.com/ (visited on 16/05/2021) (cit. on p. 85).

[171] *ROS 2 Documentation*. 17th May 2021. URL: https://index.ros.org/doc/ros2/ (visited on 16/05/2021) (cit. on p. 85).

[172] Juan Enrique Rubio et al. 'Analysis of Intrusion Detection Systems in Industrial Ecosystems'. In: *SECRYPT*. 2017. DOI: 10.5220/0006426301160128 (cit. on p. 74).

[173] *Russian Government Cyber Activity Targeting Energy and Other Critical Infrastructure Sectors*. Cybersecurity and Infrastructure Security Agency (CISA). 2018. URL: https://www.us-cert.gov/ncas/alerts/TA18-074A (visited on 26/02/2020) (cit. on p. 62).

[174] Ross Rustici and Israel Barak. *ICS Threat Broadens: Nation-State Hackers Are No Longer the Only Game in Town*. Cybereason. URL: https://www.cybereason.com/blog/industrial-control-system-specialized-hackers (visited on 09/12/2019) (cit. on p. 24).

[175] Jerome H. Saltzer and Michael D. Schroeder. 'The Protection of Information in Computer Systems'. In: *Proceedings of the IEEE* 63.9 (1975). DOI: 10.1109/PROC.1975.9939 (cit. on pp. 27, 72, 75).

[176] Benjamin Schönfuß et al. 'Prioritising Low Cost Digital Solutions Required by Manufacturing SMEs: A Shoestring Approach'. In: *Proceedings of the International Workshop on Service Orientation in Holonic and Multi-Agent Manufacturing*. Ed. by Theodor Borangiu et al. 2019. DOI: 10.1007/978-3-030-27477-1_22. URL: http://link.springer.com/10.1007/978-3-030-27477-1_22 (visited on 15/08/2019) (cit. on p. 97).

[177] Carl D. Schuett. 'Programmable Logic Controller Modification Attacks for Use in Detection Analysis'. In: *Air Force Institute of Technology Master's Thesis* (2014). URL: https://perma.cc/Q7GZ-8JQM (cit. on pp. 38–40).

[178] Savio Sciancalepore et al. 'Attribute-Based Access Control Scheme in Federated IoT Platforms'. In: *Interoperability and Open-Source Solutions for the Internet of Things*. Ed. by Ivana Podnar Žarko et al. Vol. 10218. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 123–138. DOI: 10.1007/978-3-319-56877-5_8. URL: http://link.springer.com/10.1007/978-3-319-56877-5_8 (visited on 19/06/2021) (cit. on p. 83).

[179] *SecuriOT Honeypot: Powered by Industrial Defenica*. Industrial Defenica. URL: https://www.honeypot.dk (visited on 28/12/2019) (cit. on p. 64).

[180] *Security Framework*. G4. Industrial Internet Consortium, 2016. URL: https://www.iiconsortium.org/stay-informed/IISF.htm (visited on 23/06/2021) (cit. on p. 96).

[181] *Security Model*. OPC Foundation, 2017. URL: https://reference.opcfoundation.org/v104/Core/docs/Part2/ (visited on 23/06/2021) (cit. on pp. 34, 95, 102).

[182] David Sehr et al. 'Adapting Software Fault Isolation to Contemporary CPU Architectures'. In: *Proceedings of the 19th USENIX Security Symposium (USENIX '10)*. 2010. URL: https://www.usenix.org/legacy/events/sec10/tech/full_papers/Sehr.pdf (visited on 05/11/2021) (cit. on p. 101).

[183] Anna Senpai. *[FREE] World's Largest Net: Mirai Botnet, Client, Echo Loader, CNC Source Code Release*. Hack Forums. 30th Sept. 2016. URL: https://perma.cc/SV6Z-LX92 (visited on 10/06/2019) (cit. on p. 45).

[184] Ben Seri, Gregory Vishnepolsky and Dor Zusman. *URGENT/11*. Armis. 2019. URL: https://www.armis.com/research/urgent11/ (visited on 23/06/2021) (cit. on pp. 61, 72, 74, 97).

[185] *Services*. OPC Foundation, 2017. URL: https://reference.opcfoundation.org/v104/Core/docs/Part4/ (visited on 23/06/2021) (cit. on pp. 32, 96).

[186] *Shodan*. 2020. URL: https://www.shodan.io/ (visited on 03/12/2020) (cit. on pp. 23, 39).

[187] Siemens. *Network Security*. 2020. URL: https://cache.industry.siemens.com/dl/files/269/109766269/att_1040304/v1/Brochure-Network-Security-EN.pdf (visited on 23/06/2021) (cit. on pp. 52, 98).

[188] Siemens. *Reliable and Robust Industrial Networks for the Oil and Gas Industry*. 2019. URL: https://assets.new.siemens.com/siemens/assets/api/uuid:b4588b31-5319-4404-a9e4-16736611932e/version:1570519475/whitepaper-reliable-networks-for-oil-gas-en.pdf (visited on 07/05/2020) (cit. on pp. 71, 102).

[189] Siemens. *Setting up Role-Based Access Control for Siemens Digital Grid Products*. 2016. URL: https://docplayer.net/140082109-Setting-up-role-based-access-control-for-siemens-digital-grid-products.html (visited on 23/06/2021) (cit. on p. 76).

[190] Siemens. *SIMATIC PCS7 Process Control System*. Catalog ST PCS 7. 2019. URL: https://cache.industry.siemens.com/dl/files/632/109745632/att_1070789/v1/KG_STPCS7_en_2021_Web.pdf (visited on 23/06/2021) (cit. on pp. 98, 99).

[191] Jill Slay and Michael Miller. 'Lessons Learned from the Maroochy Water Breach'. In: *Proceedings of the International Conference on Critical Infrastructure Protection*. Ed. by Eric Goetz and Sujeet Shenoi. Vol. 253. 2007. DOI: 10.1007/978-0-387-75462-8_6. URL: http://link.springer.com/10.1007/978-0-387-75462-8_6 (visited on 14/06/2019) (cit. on pp. 21, 58).

[192] Ralf Spenneberg, Maik Brüggemann and Hendrik Schwartke. 'PLC-Blaster: A Worm Living Solely in the PLC'. In: *Black Hat Asia*. 2016. URL: https://perma.cc/XWU5-TZ7L (visited on 28/10/2019) (cit. on pp. 14, 58, 59, 62).

[193] Lance Spitzner. *The Value of Honeypots, Part One: Definitions and Values of Honeypots*. Symantec. 2001. URL: https://www.symantec.com/connect/articles/value-honeypots-part-one-definitions-and-values-honeypots (visited on 09/12/2019) (cit. on p. 24).

[194] Frank Stajano and Ross Anderson. 'The Resurrecting Duckling: Security Issues for Ad-Hoc Wireless Networks'. In: *Proceedings of the International Workshop on Security Protocols*. Ed. by Bruce Christianson et al. Red. by Gerhard Goos, Juris Hartmanis and Jan van Leeuwen. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 172–182. DOI: 10.1007/10720107_24. URL: http://link.springer.com/10.1007/10720107_24 (visited on 18/05/2020) (cit. on pp. 30, 95).

[195] Emily Stark et al. 'Does Certificate Transparency Break the Web? Measuring Adoption and Error Rate'. In: *Proceedings of the IEEE Symposium on Security and Privacy (S&P '19)*. May 2019. DOI: 10.1109/SP.2019.00027 (cit. on p. 31).

[196] Jennifer G. Steiner, Clifford Neuman and Jeffrey I. Schiller. 'Kerberos: An Authentication Service for Open Network Systems'. In: *Usenix Winter* (1988). URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.39.606&rep=rep1&type=pdf (visited on 23/06/2021) (cit. on p. 76).

[197] Keith Stouffer et al. 'Guide to Industrial Control Systems Security, Revision 2'. In: *NIST Special Publication* 800.82 (2015). URL: https://csrc.nist.gov/publications/detail/sp/800-82/rev-2/final (visited on 23/06/2021) (cit. on p. 21).

[198] Joachim Strömbergson. *Secworks/Sha1*. 12th May 2021. URL: https://github.com/secworks/sha1 (visited on 16/05/2021) (cit. on p. 93).

[199] Trustworthy Systems Team. *seL4 Reference Manual*. Version 11.0.0. Data61, 20th Nov. 2019. URL: https://sel4.systems/Info/Docs/seL4-manual-latest.pdf (visited on 05/10/2021) (cit. on p. 100).

[200] Andre Teixeira et al. 'Revealing Stealthy Attacks in Control Systems'. In: *Proceedings of the 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton '12)*. Monticello, IL, USA: IEEE, Oct. 2012, pp. 1806–1813. DOI: 10.1109/Allerton.2012.6483441. URL: http://ieeexplore.ieee.org/document/6483441/ (visited on 13/08/2019) (cit. on p. 22).

[201] André Teixeira et al. 'Attack Models and Scenarios for Networked Control Systems'. In: *Proceedings of the 1st International Conference on High Confidence Networked Systems (HiCoNS '12)*. Beijing, China: ACM Press, 2012, p. 55. ISBN: 978-1-4503-1263-9. DOI: 10.1145/2185505.2185515. URL: http://dl.acm.org/citation.cfm?doid=2185505.2185515 (visited on 01/06/2021) (cit. on p. 22).

[202] *The ICS Landscape and Threat Activity Groups*. Dragos Inc. 2019. URL: https://perma.cc/RFL4-HWD8 (visited on 11/03/2020) (cit. on p. 56).

[203] *The Modbus Organization*. 17th May 2021. URL: https://modbus.org/ (visited on 16/05/2021) (cit. on p. 87).

[204] *The ZMap Project*. 2020. URL: https://zmap.io/ (visited on 02/12/2019) (cit. on pp. 23, 65).

[205] Daniel R. Thomas, Richard Clayton and Alastair R. Beresford. '1000 Days of UDP Amplification DDoS Attacks'. In: *Proceedings of the 2017 APWG Symposium on Electronic Crime Research (eCrime '17)* (2017). DOI: 10.1109/ECRIME.2017.7945057. URL: https://ieeexplore.ieee.org/document/7945057/ (visited on 10/06/2019) (cit. on p. 43).

[206] Tridium. *Niagara 4 Hardening Guide*. 2019. URL: https://www.tridium.com/content/dam/tridium/en/documents/document-lists/Niagara%204%20Hardening%20Guide_2019.pdf (visited on 23/06/2021) (cit. on pp. 52, 76).

[207] *TRISIS Malware: Analysis of Safety System Targeted Malware*. Dragos Inc. 2017. URL: https://perma.cc/K9EM-CABV (visited on 08/11/2019) (cit. on pp. 52, 61, 62, 72).

[208] David I. Urbina et al. 'Limiting the Impact of Stealthy Attacks on Industrial Control Systems'. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS '16)*. Vienna, Austria: ACM Press, 2016, pp. 1092–1105. ISBN: 978-1-4503-4139-4. DOI: 10.1145/2976749.2978388. URL: http://dl.acm.org/citation.cfm?doid=2976749.2978388 (visited on 09/08/2019) (cit. on pp. 21, 22).

[209] Jorge M. Urrea et al. 'Control System Backplane Monitoring with FPGA'. U.S. pat. 10,409,274 B1. National Technology & Engineering Solutions of Sandia, LLC. 10th Sept. 2019. URL: https://patents.google.com/patent/US10409274B1/en (visited on 23/06/2021) (cit. on p. 74).

[210] Paul C. van Oorschot. *Computer Security and the Internet: Tools and Jewels*. 2020. URL: http://people.scs.carleton.ca/~paulv/toolsjewels.html (visited on 23/06/2021) (cit. on p. 30).

[211] Alexander Vetterl and Richard Clayton. 'Honware: A Virtual Honeypot Framework for Capturing CPE and IoT Zero Days'. In: *Proceedings of the 2019 APWG Symposium on Electronic Crime Research (eCrime '19)*. 2019 (cit. on pp. 24, 62).

[212] Alexandru Vlad, Sebastian Obermeier and Der-Yeuan Yu. 'ICS Threat Analysis Using a Large-Scale Honeynet'. In: *Proceedings of the 3rd International Symposium for ICS & SCADA Cyber Security Research (ICS-CSR '15)*. 2015. DOI: 10.14236/ewic/ICS2015.3. URL: http://ewic.bcs.org/content/ConWebDoc/55096 (visited on 18/06/2019) (cit. on p. 54).

[213] Robert Wahbe et al. 'Efficient Software-Based Fault Isolation'. In: *Proceedings of the 14th ACM Symposium on Operating Systems Principles*. 1993. URL: https://dl.acm.org/doi/abs/10.1145/168619.168635 (visited on 23/06/2021) (cit. on p. 101).

[214] Jiafu Wan et al. 'Software-Defined Industrial Internet of Things in the Context of Industry 4.0'. In: *IEEE Sensors Journal* (2016). ISSN: 1530-437X, 1558-1748, 2379-9153. DOI: 10.1109/JSEN.2016.2565621. URL: http://ieeexplore.ieee.org/document/7467436/ (visited on 14/06/2019) (cit. on p. 21).

[215] Robert N. M. Watson et al. *An Introduction to CHERI*. UCAM-CL-TR-941. University of Cambridge, 2019. URL: https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-941.pdf (cit. on pp. 27, 28, 80).

[216] *What You Need to Know about the WannaCry Ransomware*. Symantec. 2017. URL: https://perma.cc/J6HD-HFYR (visited on 30/10/2019) (cit. on p. 42).

[217] Paul M. Williams. 'Distinguishing Internet-Facing ICS Devices Using PLC Programming Information'. In: *Air Force Institute of Technology Master's Thesis* (2014). URL: https://perma.cc/W7YL-7J7T (cit. on pp. 39, 40).

[218] Martin Wollschlaeger, Thilo Sauter and Juergen Jasperneite. 'The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0'. In: *IEEE Industrial Electronics Magazine* (2017). ISSN: 1932-4529. DOI: 10.1109/MIE.2017.2649104. URL: http://ieeexplore.ieee.org/document/7883994/ (visited on 28/10/2019) (cit. on pp. 57, 63).

[219] Hongyan Xia et al. 'CheriRTOS: A Capability Model for Embedded Devices'. In: *Proceedings of the 36th IEEE International Conference on Computer Design (ICCD '18)*. Orlando, FL, USA: IEEE, Oct. 2018, pp. 92–99. ISBN: 978-1-5386-8477-1. DOI: 10.1109/ICCD.2018.00023. URL: https://ieeexplore.ieee.org/document/8615673/ (visited on 10/07/2019) (cit. on pp. 13, 29, 73, 100, 102).

[220] Jiexin Zhang, Alastair R. Beresford and Ian Sheret. 'SENSORID: Sensor Calibration Fingerprinting for Smartphones'. In: *Proceedings of the IEEE Symposium on Security and Privacy (S&P '19)*. 2019. URL: https://ieeexplore.ieee.org/abstract/document/8835276/ (visited on 23/06/2021) (cit. on p. 47).