CHERI Macaroons: Efficient, host-based access control for cyber-physical systems

Michael Dodson
University of Cambridge, UK
md403@cl.cam.ac.uk

Alastair R. Beresford
University of Cambridge, UK
arb33@cl.cam.ac.uk

Alexander Richardson
University of Cambridge, UK
alexander.richardson@cl.cam.ac.uk

Jessica Clarke
University of Cambridge, UK
jessica.clarke@cl.cam.ac.uk

Robert N. M. Watson
University of Cambridge, UK
robert.watson@cl.cam.ac.uk

***Abstract*—Cyber-Physical Systems (CPS) often rely on network boundary defence as a primary means of access control; therefore, the compromise of one device threatens the security of all devices within the boundary. Resource and real-time constraints, tight hardware/software coupling, and decades-long service lifetimes complicate efforts for more robust, host-based access control mechanisms. Distributed capability systems provide opportunities for restoring access control to resource-owning devices; however, such a protection model requires a capability-based architecture for CPS devices as well as task compartmentalisation to be effective.**

This paper demonstrates hardware enforcement of network bearer tokens using an efficient translation between CHERI (Capability Hardware Enhanced RISC Instructions) architectural capabilities and Macaroon network tokens. While this method appears to generalise to any network-based access control problem, we specifically consider CPS, as our method is well-suited for controlling resources in the physical domain. We demonstrate the method in a distributed robotics application and in a hierarchical industrial control application, and discuss our plans to evaluate and extend the method.

***Index Terms*—CHERI, Macaroons, cyber-physical, industrial control, robotics, access control, security**

1. Introduction

Cyber-Physical Systems (CPS) tightly couple hardware and software with sensing and manipulation of the physical environment. Examples include automobile Electronic Control Units (ECUs), Industrial Control Systems (ICS), Internet of Things (IoT), and robotics. CPS often have limited compute and memory resources, real-time and high-availability constraints, and safety-critical functions, requiring them to meet rigorous certifications. Further, many CPS applications require decades of durability, raising (unsolved) questions about toolchain and patch support [1]. Finally, CPS applications often consist of distributed, heterogeneous devices integrated over multiple networks and protocols. Together, these factors create a challenging environment for applying common security paradigms (e.g., encryption, compartmentalisation, network segmentation, and high patch cadence).

Most industrial solutions side-step these challenges by emphasising boundary control at the Operational Technology (OT) network. For example, ICS are often organised in 'automation cells', aggregating ICS devices, sensors, and actuators performing a common task (Figure 1) [2]. Within the cell, communication between devices is unencrypted and unauthenticated, often using decades-old serial protocols modified for TCP or UDP transport. Therefore, any compromise of one component in the cell effectively results in a compromise of the whole cell. Similar boundary control issues have been identified for automotive applications [3]. Emphasis on boundary control makes weak assumptions about the adversary and results in a failure mode that is at odds with the philosophy of graceful degradation underpinning many CPS designs. This philosophy is especially prevalent in safety-critical industries, in which systems employ hardware redundancy and byzantine fault tolerance to protect against probabilistic failure, ensuring the system as a whole can still perform the required actions despite multiple component failures.

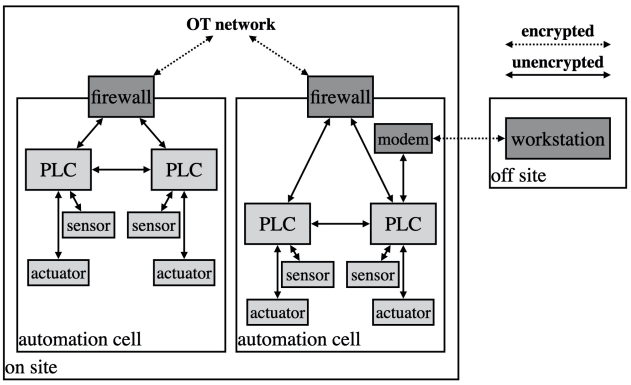

Figure 1. Example of an ICS 'automation cell' architecture, with unencrypted and unauthenticated communication between devices within a given boundary. Remote access to a given device, often required by the vendor, may occur through a modem that is within the boundary, bypassing the firewall.

Inside the boundary, progress is being made on behaviour-based anomaly detection, where network traffic, including serial communication with sensors and actuators, is aggregated and compared with physics-based models of system behaviour to detect anomalies [4]. While

these systems make stronger assumptions about the adversary, they are often as complex as the systems they are designed to protect and add to existing integration and long-term support challenges.

Boundary control and intrusion detection may contribute to a defence-in-depth strategy; however, without host-based security solutions they both force CPS devices to expand their trust boundary to encompass not just the boundary control or intrusion detection device, but all other devices under the same aegis. This has been exploited in the wild at both the network and device boundary. At the network boundary, many automotive hacks are initiated by remote access to an insecure ECU resulting in full control of steering, acceleration, and braking [3]. Similarly, OT network compromises leveraging BLACKENERGY 3 malware gave hackers arbitrary control over sections of the Ukrainian power grid in 2015 and again with a more targeted attack in 2016 [5]. At the device level, lack of compartmentalisation in the TCP stack resulted in multiple remote code execution vulnerabilities in the VxWorks Real-Time Operating System (RTOS) [6]. Lack of compartmentalisation also enabled an attack against the Schneider Triconex Safety Instrumentation System, where malware masqueraded as a log reading application and modified safety-critical programmable logic [7].

To address these concerns, we propose binding local and distributed capabilities to restore access control authority to resource owners, particularly for those resources that have physical effects. In this paper we: (1) introduce architectural capabilities as tokens to protect access to physical resources; (2) propose a model for implementing hardware-backed tokens in distributed systems; and (3) demonstrate efficient translation between hardware and network tokens in concrete prototypes.

2. Capability-based access control for CPS

Capabilities are unforgeable tokens that prove the presenter is authorised to access a named resource [8]. Capabilities are an attractive option for access control in the CPS domain where resource owners and users are likely to be devices, rather than people, and have static relationships known at design or install-time. These static, pre-defined relationships mitigate known challenges in capability systems associated with audit and revocation and simplify the mechanisms necessary to assign capabilities to authorised principals. Capabilities also align well with principles of delegation, which are prevalent in distributed and hierarchical CPS. Further, though many relationships in CPS are device-to-device, capabilities also map naturally to the way humans tend to authenticate with CPS. For example, a car key is a capability held by the driver, granting access to resources that are mostly delegated through primary and secondary ECUs. Finally, capabilities are abstractions that make it easy to reason about and trace resource access, even when that capability has been delegated or transferred between mediums.

Historically, capability-based access control systems were developed for single-device, multi-user hardware, such as mainframes [8]. More recently, network bearer tokens (e.g., cookies) have been used to support access control in single-device, multi-user, client-server interactions [9]. In contrast, CPS are multi-user, multi-device systems requiring heterogeneous devices to communicate over networks in peer-to-peer and hierarchical relationships. Therefore, a capability-based access control system for CPS must locally bind capabilities to physical resources controlled by the hardware and be able to delegate authority by passing those capabilities across networks.

We implement this design pattern by tightly coupling CHERI (Capability Hardware Enhanced RISC Instructions) architectural capabilities [10] with Macaroon network tokens [9] to provide hardware-enforced access to physical resources in distributed CPS.

3. CHERI: Local capabilities for CPS

CPS generally lack either hardware (e.g., MMU) or software (e.g., virtualisation) solutions to enforce memory segregation or task compartmentalisation; therefore, tasks share flat, physical memory spaces, allowing a compromise of one to affect all. This is a microcosm of the larger boundary control problem described in Section 1 and is an inadequate protection model on which to build a local capability system. Specifically, if all tasks on a device share the same physical memory and register files, there is no way to prevent a compromised task from acquiring and using a capability created by any co-located task.

CHERI capabilities provide both the abstract mechanism for physical resource access control and the basis for efficient task compartmentalisation in CPS. CHERI enables fine-grained memory protection and scalable software compartmentalisation via hardware-enforced, bounded pointers. CHERI capabilities maintain address boundaries and permissions (e.g., read, write, execute) to control memory accesses. Tagged memory and CHERI-aware instructions guarantee unforgeability, provenance (ensuring a capability can only be constructed via other valid capabilities), and monotonicity (ensuring that the bounds and permissions of a new capability cannot exceed the capability from which it is derived) [10].

A CHERI-aware kernel or bootloader achieves compartmentalisation of tasks by providing capabilities that precisely define the memory regions to which the task has access and the permissions it has for those memory regions. For example, the task's program counter capability will be execute-only and have a base and length confined to that task's code in the binary file. Similar capabilities can be provided to enable precise access to data regions and memory-mapped I/O and services.

While CHERI capabilities nominally replace pointers and naturally act as tokens granting access to regions of memory, they can be used as tokens to access any abstract resource, including allowing the bearer to read a sensor or command an actuator (see Figure 2). Access control for physical resources can be implemented similar to memory-mapped I/O, reserving a region of address space to represent continuous, discrete, or enumerated physical abstractions. For example, a motor with speed settings between 0 and 10 can map to a capability with a length of 0x0a and a base selected from within the reserved address space. Leveraging CHERI monotonicity for delegation is intuitive, as the capability can be used to derive a new capability with a reduced addressable range, corresponding to a reduction in the set of speeds the bearer can command. Similar mappings can be used for

enumerated quantities. For example, the motor commands `ON`, `+INCREMENT`, `−INCREMENT`, `STOP` can map to a capability with a length of 0x04. The capability can be efficiently delegated by reducing the length field, with the default or emergency capability (i.e., `STOP`) represented by the least privileged derivative of the original capability (i.e., a length of 0x01).

Together, task isolation and physical resource tokens create a hardware-enforced, efficient access control mechanism for CPS, precluding any principal from reading data or issuing a command without an explicitly granted capability derived from a trusted source (e.g., the kernel or bootloader). This not only prevents an adversary on or off the device from leveraging memory safety vulnerabilities to manipulate physical resources, but creates a chain of provenance and intentionality that makes it easy to reason about delegated authority between principals.

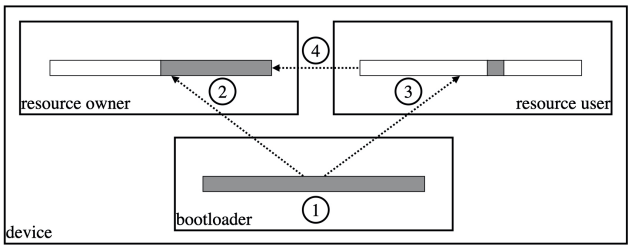

Figure 2. Example of a bootloader generating capabilities for the resource owner and user. The bootloader uses a 'root' capability for the entire memory space (1) to derive a capability for the resource owner covering its entire memory space (2) and a capability for the resource user specifically for the resource it is allowed to use (3). The resource user provides this capability to the resource owner as a token when requesting access to the resource (4).

4. Macaroons: Network capabilities for CPS

Guaranteeing CHERI capability properties of monotonicity and provenance depends on hardware enforcement of microarchitectural features such as tagged memory; therefore, the access control mechanisms described above only apply for processes with shared memory, and we need a separate mechanism to carry access control tokens between networked devices. Having considered various distributed Public Key Infrastructures (PKIs) and bearer tokens (e.g., JSON Web Tokens (JWTs)), we chose Macaroons based on their semantic similarity to CHERI capabilities and the limited cryptographic and maintenance burden placed on the resource owner [9], [11].

Macaroons are bearer credentials designed for decentralised, network-based access control requiring efficient delegation and attenuation of privilege. Macaroons are constructed using a chain of keyed Hash-based Message Authentication Codes (HMACs) to protect provenance and integrity. The initial Macaroon, generated by the resource owner, consists of a plaintext identifier and its hash signature. The resource owner or any holder of the token can add an arbitrary number of plaintext caveats to extend 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 new Macaroon, preventing any intermediary or user from removing caveats once appended to the chain. A caveat can attenuate privilege (e.g., limit the bearer to read-only access), restrict context (e.g., require the token to be used within a limited time), or link to a Macaroon generated by a third party (e.g., an authentication service). The provenance and integrity of the resulting Macaroon, when returned to the resource owner, can be easily verified using the secret key to recompute the HMAC chain. The resource owner then checks that each caveat is met and grants or denies access to the user presenting the Macaroon [9].

Unlike PKI and JWT schemes, the resource owner only requires facilities for generating keys and performing HMACs. These operations are at least one order of magnitude faster than those required for even lightweight PKI schemes [9], and hardware acceleration is widely available, even for low-cost microcontroller systems. Further, the simplicity of the operation limits the Trusted Computing Base (TCB) required for Macaroon generation and verification, reducing the risk of frequent, future patches.

Because the semantics of Macaroons and CHERI capabilities overlap, mapping a verified Macaroon to a CHERI capability (or vice versa) is straightforward (see Figure 3), allowing us to consider abstract tokens represented by CHERI capabilities on the hardware and by Macaroons on the network. Further, as many CPS communication graphs are static and known at design-time, the cost of executing such a mapping can be bounded, based on the number of capabilities that need to be mapped and the number of possible caveats added to the Macaroon.

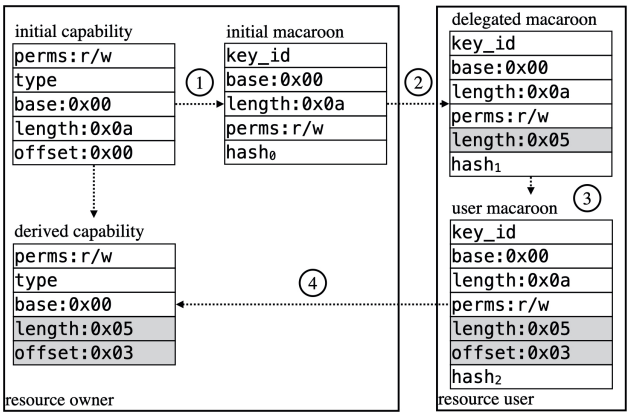

Figure 3. Example of mapping between capabilities and Macaroons. The resource owner creates a capability representing a physical resource and maps to an initial Macaroon (1). The resource owner attenuates the Macaroon by reducing the length field and passes it to the resource user (2). The resource user attenuates the Macaroon by specifying the requested action in the form of an offset (3). The resource user passes the Macaroon back to the resource owner as a request for access. The resource owner verifies the Macaroon and uses the verified Macaroon and initial capability to derive a new capability to access the requested resource (4).

5. Authentication and token distribution

For CHERI capabilities, we assume that the kernel or bootloader provides processes with appropriately bounded and permissive capabilities. Most CPS have limited, static, well-defined behaviour, so it is reasonable to assume that either the kernel or bootloader can be configured at design-time or install-time to correctly assign capabilities.

For Macaroons, an explicit authentication step is required to ensure our method generalises to arbitrary prin-

cipals on an arbitrary network. The goal is to provide a Macaroon from the resource owner to an authenticated and authorised resource user and then ensure the user can return the Macaroon to the owner as part of a service request, all while maintaining the integrity of the Macaroon and preventing reuse by a Man-in-the-Middle (MitM) attacker. Given our effort to limit both the implementation complexity and patch pressure at the resource owner, we selected Kerberos to facilitate this authentication step. In its simplest form, the Kerberos protocol is initialised by two principals (a server and client) sharing secret keys with the Kerberos server. When the client is ready to initiate a service request from the server, the client authenticates with the Kerberos server and receives a session key encrypted under its own key and a session key encrypted under the server's key, which it sends to the server. After a freshness check, the server and client initiate an encrypted session. Kerberos is attractive for the CPS use case, as it removes almost the entire authentication burden from the resource owner while still allowing it to locally enforce access control to its resources.

6. Case studies

To explore the implementation and security implications of combining CHERI capabilities, Macaroons, and a suitable authentication framework such as Kerberos, we perform two case studies: a distributed Robot Operating System (ROS)-based application and a hierarchical industrial control application.

6.1. Distributed systems: ROS

ROS is a library framework for developing and integrating robotics applications, and our first case study is based on its 2nd generation (ROS2[1]). The core ROS library is written in C, with a C++ API library for developing ROS clients. 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).[2] The resource owner is a `motor control` node, which encapsulates independent control of the two wheels, thereby controlling robot speed and direction (by differential movement of the wheels). The resource user is a `user input` node, which encapsulates obtaining, validating, and transmitting user input to control the robot. Communication between the two nodes is via local wifi. ROS does not provide any facility for encrypting communication between nodes. Figure 4 shows the architecture and the interaction between principals.

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 Localisation and Mapping (SLAM) input to close the control loop.

1. https://index.ros.org/doc/ros2/
2. http://emanual.robotis.com/docs/en/platform/turtlebot3/overview/

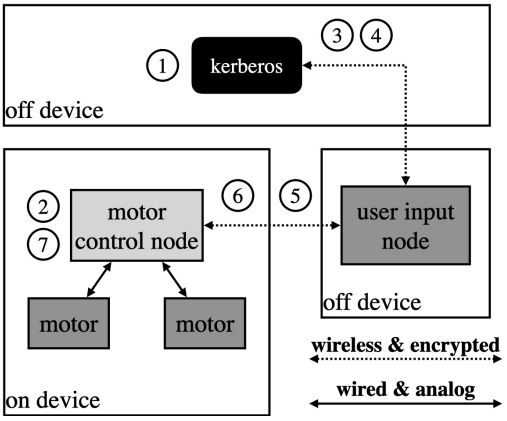

Figure 4. Architecture of the ROS case study, with the `motor control` node locally communicating with two motors and remotely communicating with the `user input` node. As part of initialisation, both nodes need to share a secret with Kerberos (1). The `motor control` node generates CHERI capabilities and corresponding Macaroons (2). The `user input` node authenticates with Kerberos (3) and receives a session keys for initiating an encrypted session with the `motor control` node (4). Once the encrypted session is initiated, the `motor control` node passes requested Macaroons to the `user input` node (5), caveated with the session identifier and sequence number. The `user input` node caveats the Macaroon with an incremented sequence number and the requested action and returns it to the `motor control` node via the encrypted channel (6). Finally, the `motor control` node validates the Macaroon, derives a restricted CHERI capability, and executes the request (7).

The motors are the resource for which access is required. To simplify comparison with the unmodified implementation, we implement one token for linear speed and one token for angular speed to control these resources, rather than implementing one token for each motor. The `motor control` node (the resource owner) derives a CHERI capability and Macaroon for each resource.

The `user input` node authenticates with the Kerberos node and initiates an encrypted session with the `motor control` node. Having received the Macaroons, the `user input` node adds the desired linear or angular velocity as a caveat to the appropriate Macaroon and returns it to the `motor control` node. The `motor control` node verifies the received Macaroon, uses it to derive a new capability (with additional restrictions based on Macaroon caveats), and executes the request. hardware enforcement of CHERI properties ensures that any attempt to expand the bounds or set an out-of-bounds offset will throw a hardware exception and prevent derivation and/or use of the new capability.

6.2. Hierarchical systems: ICS remote access

The second case study is based on a common ICS architecture, where devices coordinating on a task are organised within cells and are connected to the larger OT network through a firewall or security appliance [2]. As discussed in Section 1, any compromise of the boundary compromises the entire cell. Similarly, lack of compartmentalisation within the Programmable Logic Controller (PLC) allows any compromised task to manipulate the resources owned by other tasks.

For our example, the ICS device is a PLC controlling a motor-operated valve. The PLC provides commands

to open and shut the valve and senses the position of the valve as either open, shut, or intermediate. The PLC stores calibration constants for the motor speed and the valve position sensors. A vendor may need to read state information, change constants, or run a diagnostic test, but should not be able to command valve motion. Valve motion can only be commanded locally at the PLC.

The `PLC` is the resource owner and the remote `workstation` is the resource user. We divide the `PLC` into two, isolated compartments (`local controller`, `control loop`), which is efficient with CHERI, even on a PLC with a flat, physical memory address space and limited computational power. Motor commands and valve state are the physical resources for which access control is required, each of which can be represented as a CHERI capability by the `PLC`. The calibration constants and testing semaphore (held by the `PLC` during normal operation or by the `workstation` during testing), are memory resources that can also be represented as CHERI capabilities by the `PLC`. The `local controller` should be the only avenue through which valve state can be manipulated. Figure 5 shows the architecture and the interaction between principals.

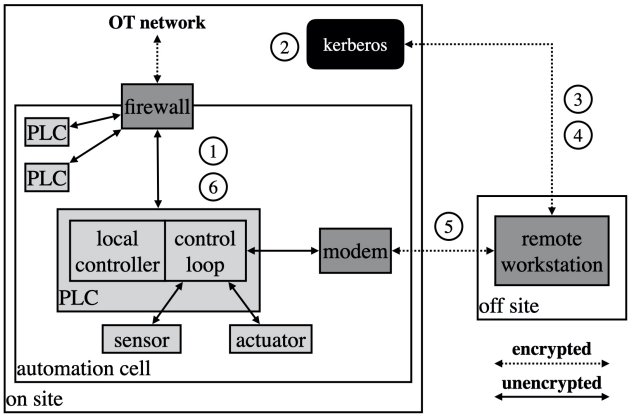

Figure 5. Architecture of the ICS case study, with a `PLC` communicating with a remote `workstation`. As part of initialisation, the `control loop` creates Macaroons for specific resources (1) and shares them with Kerberos (2). Simultaneously, the `workstation` shares its secret with Kerberos (2). The `workstation` authenticates with Kerberos (3) and receives a delegated Macaroon for the requested resource, caveated with an initial sequence number and an expiration time (4). The `workstation` caveats the Macaroon with an incremented sequence number and the requested action and returns it to the `PLC` via the partially unencrypted communication channel (5). Finally, the `PLC` derives a restricted CHERI capability and executes the request (6).

The `PLC` is a device with limited compute and memory resources and real-time constraints; therefore, to limit the cryptographic burden on the `PLC`, we eliminate the encrypted session between the resource owner and user and rely on the Kerberos node to delegate Macaroons. Specifically, the `control loop` creates CHERI capabilities for each resource, delegates the capability for valve operation to the `local controller`, and shares Macaroons for the remaining resources (e.g., valve state, calibration constants, testing semaphore) with the Kerberos server. When the `workstation` authenticates with Kerberos, Kerberos provides a Macaroon for the requested service, and attenuates permissions (e.g., read-only) as appropriate. The `workstation` then adds the desired command or request to the Macaroon as a caveat and sends it to the `PLC` over the existing communication channel.

As above, the `control loop` receives and verifies the Macaroon, uses it to derive a new capability, and executes the request. As the capability to command the valve was never mapped to a Macaroon, the hardware will not allow any Macaroon to be mapped to a capability with permission to change valve state.

While a fully encrypted session between the `PLC` and the `workstation` would add confidentiality protection, the modified Kerberos protocol provides similar guarantees with respect to tampering by a MitM or abuse by the resource user without adding any additional complexity to the resource owner.

7. Prototyping and evaluation

We currently have a working prototype of ROS2 and Macaroons running on CheriBSD, a CHERI-aware port of FreeBSD.[3] Our CHERI CPU is emulated, but Field Programmable Gate Array (soft) cores are available for both high-end and low-resource, embedded designs. Additionally, a high-end, superscalar prototype CHERI CPU is expected to be available from Arm in 2021.[4] Our Macaroons port is based on libmacaroons.[5] Porting ROS2 and libmacaroons to CHERI-aware libraries was straightforward and consisted mostly of updating memory allocators and other code segments that hard-code assumptions related to pointer size. Where our changes were generalisable, we submitted pull requests to the applicable projects, which have all been merged. ROS2 and its microcontroller derivative, micro-ROS[6] both support real-time operation, and we plan to evolve our prototype to incorporate real-time constraints, assessing the impact of capabilities on scheduling and execution of tasks with hard, soft, or best-effort timing requirements. We also plan to port OpenPLC[7] for our ICS prototype. While OpenPLC does not support real-time operation, it emulates PLC network behaviour sufficiently well to be the basis for many academic studies and several commercial products.

The prototypes will support evaluations of performance, security, and porting effort. Prior work on a CHERI-aware real-time system has already quantified the worst-case context-switch and inter-process communication overhead introduced by CHERI capabilities in embedded systems [12]. We plan to quantify how the number of capabilities maintained by a resource owner, the number of derived Macaroons, and the number of Macaroon caveats affect start-up and run-time performance. These affect the number of calls to CHERI-aware instructions, key generation routines, and HMACs performed by the host, respectively. We will also use our prototypes to compare CHERI-aware and CHERI-unaware threat models.

Currently, CheriBSD's default behaviour is to raise a signal that terminates the current process when the hardware identifies a capability violation. This is not ideal behaviour in many CPS scenarios requiring graceful

3. https://github.com/CTSRD-CHERI/cheribsd
4. https://developer.arm.com/architectures/cpu-architecture/a-profile/morello
5. https://github.com/rescrv/libmacaroons
6. https://micro-ros.github.io/
7. https://www.openplcproject.com/

degradation, and CheriBSD supports installing a custom signal handler, allowing a system designer to catch and handle these signals in different ways. For example, a compartment implementing a TCP stack may simply drop incoming packets attempting to overflow buffers, while a compartment executing a control loop may revert to a minimal, safe state or perform a reset. Our prototypes will be the basis for exploring various operational models for CHERI-related violations in CPS. Similarly, lessons learned will provide feedback for future iterations of the CHERI protection model to better support CPS use cases and hardware-enforced tokens in distributed systems.

8. Conclusion

In this paper we present a method for fine-grained access control in CPS using distributed capabilities. We leverage the semantic similarity between CHERI architectural capabilities and Macaroons to create unforgeable tokens for which both provenance and integrity can be efficiently verified by resource-owning hardware prior to granting access to a requested resource. Our method provides effective access control in the presence of strong adversaries on the network and on the resource-owning hardware. We demonstrate the method using two CPS case studies, as the tight hardware/software coupling, limited compute and memory resources, and real-time constraints of many CPS systems provide a challenging environment for implementing host-based access control mechanisms. In future work, we intend to show how the method extends to general resource access control scenarios.

Acknowledgments

Michael Dodson is supported by a scholarship from the Gates Cambridge Trust; Alastair R. Beresford is partially supported by EPSRC [grant number EP/M020320/1]; Jessica Clarke is partially supported by EPSRC [grant number EP/R513180/1]. This work was supported in part by the Defense Advanced Research Projects Agency (DARPA) under contract HR0011-18-C-0016 ("ECATS"). The views, opinions, and/or findings contained in this report are those of the authors and should not be interpreted as representing the official views or policies of the funders, the Department of Defense, or the U.S. Government. Approved for Public Release, Distribution Unlimited. We are grateful to Martin Kleppmann for suggesting the similarities between CHERI capabilities and Macaroons.

References

[1] É. Leverett, R. Clayton, and R. Anderson, "Standardisation and certification of the 'Internet of Things'," *Workshop on the Economics of Information Security (WEIS)*, 2017. [Online]. Available: https://perma.cc/5Y9R-9DD3

[2] Siemens, "Reliable and robust industrial networks for the oil and gas industry," Siemens, Tech. Rep., 2019. [Online]. Available: https://assets.new.siemens.com/siemens/assets/api/uuid:b4588b31-5319-4404-a9e4-16736611932e/version:1570519475/whitepaper-reliable-networks-for-oil-gas-en.pdf

[3] C. Miller and C. Valasek, "A survey of remote automotive attack surfaces," *Black Hat USA*, 2014.

[4] J. Giraldo, D. Urbina, A. Cardenas, J. Valente, M. Faisal, J. Ruths, N. O. Tippenhauer, H. Sandberg, and R. Candell, "A survey of physics-based attack detection in cyber-physical systems," *ACM Computing Surveys*, vol. 51, no. 4, pp. 1–36, Jul. 2018.

[5] "CRASHOVERRIDE: Analysis of the threat to electric grid operations," *Dragos Inc.*, 2017. [Online]. Available: https://perma.cc/E7K5-9T8M

[6] B. Seri, G. Vishnepolsky, and D. Zusman, "URGENT/11 technical white paper," *Armis*, 2019. [Online]. Available: https://go.armis.com/urgent11

[7] "TRISIS malware: Analysis of safety system targeted malware," *Dragos Inc.*, 2017. [Online]. Available: https://perma.cc/K9EM-CABV

[8] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, vol. 63, no. 9, 1975.

[9] A. Birgisson, J. G. Politz, Ú. Erlingsson, A. Taly, M. Vrable, and M. Lentczner, "Macaroons: Cookies with contextual caveats for decentralized authorization in the cloud," *Network and Distributed System Security (NDSS) Symposium*, 2014.

[10] R. N. M. Watson, S. W. Moore, P. Sewell, and P. G. Neumann, "An Introduction to CHERI," University of Cambridge, Tech. Rep. UCAM-CL-TR-941, 2019.

[11] S. Sciancalepore, M. Pilc, S. Schröder, G. Bianchi, G. Boggia, M. Pawłowski, G. Piro, M. Płóciennik, and H. Weisgrab, "Attribute-based access control scheme in federated IoT platforms," *Interoperability and Open-Source Solutions for the Internet of Things (InterOSS-IoT)*, 2017.

[12] H. Xia, J. Woodruff, H. Barral, L. Esswood, A. Joannou, R. Kovacsics, D. Chisnall, M. Roe, B. Davis, E. Napierala, J. Baldwin, K. Gudka, P. G. Neumann, A. Richardson, S. W. Moore, and R. N. M. Watson, "CheriRTOS: A Capability Model for Embedded Devices," in *2018 IEEE 36th International Conference on Computer Design (ICCD)*. Orlando, FL, USA: IEEE, Oct. 2018, pp. 92–99.