A Reference Separation Architecture for Mixed-Criticality Medical and IoT Devices

Todd Carpenter
Adventium Labs
Minneapolis, MN, USA
todd.carpenter@adventiumlabs.com

John Hatcliff
Department of Computer Science
Kansas State University
Manhattan, KS, USA
hatcliff@ksu.edu

Eugene Y. Vasserman
Department of Computer Science
Kansas State University
Manhattan, KS, USA
eyv@ksu.edu

ABSTRACT

Low cost embedded cyber-physical systems and ubiquitous networking has opened up a new world of connected devices in our homes, workplaces, automobiles, and medical clinics. Unfortunately, security has not progressed at the same pace, exposing unwitting users to loss of privacy, personal identifying information, and even safety. This is especially true for connected medical devices which interact directly with patients. Many of these medical devices lack even basic security, and instead rely on rigidly controlled environments of use, which are difficult to achieve and maintain. The Intrinsically Secure, Open and Safe Cyber-Physically Enabled, Life-Critical Essential Services (ISOSCELES) architecture is a reference implementation for future mixed-criticality medical and Internet of Things (IoT) system designs. This reference implementation will allow manufacturers to focus on the clinical side of their product, reducing the time and effort spent ensuring that security vulnerabilities in the resulting platform minimize adverse impacts on patient safety. ISOSCELES' separation architecture, backed by model-based analysis and configuration tools, simplifies product-line design and maintenance, since changes made to only one partition will have limited to no effect on other partitions.

CCS CONCEPTS

• **Security and privacy → Systems security**; *Security services*; • **Computer systems organization → Sensors and actuators**; **Reliability**; **Maintainability and maintenance**; *Availability*;

KEYWORDS

Medical CPS; Security; Safety; Separation architecture; IoT

1 INTRODUCTION

Recent trends resulting in part from Electronic Health Record expansion, are encouraging inter-networking of medical devices to reduce costs, minimize errors, increase patient safety, and improve outcomes. As complexity and network accessibility grows, the system and its components become more susceptible to attack, which can lead to reduced patient safety. Attacks on medical devices can compromise healthcare IT networks, a key part of the U.S. healthcare critical infrastructure [6]. Despite these risks, devices will continue to be networked together because of significant cost savings [3].

Consequently, security of networked medical devices is important to overall safety. This has prompted industry groups to develop cyber-security standards (e.g., AAMI TIR57 [4]), researchers to start addressing the challenges [10], and government agencies to issue guidance and requirements (e.g., [6]).

Without a substantial change in overall device architecture, addressing cyber-security issues will inevitably lead to increased development and approval costs. These are already prohibitive. For example, a report from 2010 pegs the average 510(k) approval cost at $24M per device [15]. Even larger firms claim challenges due to security requirements. Development staff often lack embedded security experience. Their typical design process uses mathematical, quantitative *safety* risk assessment models, which leads them to inappropriately dismiss security risks.

We propose and describe the Intrinsically Secure, Open and Safe Cyber-Physically Enabled, Life-Critical Essential Services (ISOSCELES) platform, a high-confidence, cyber-physical architecture for medical devices. ISOSCELES is intended to support the primary medical application of the device while enforcing safety and security properties. The ISOSCELES design provides strong time and space partitioning by combining separation kernels with virtualization provided by a Type I hypervisor [20] to isolate security and safety monitoring services from the functions they are monitoring. The combination provides the strong time and space partitioning capabilities needed for mixed-criticality in medical devices. Both separation kernels and Type I hypervisors are designed to run directly on the hardware. The hypervisor provides virtual abstractions of that hardware to multiple instances of guest operating systems, while a separation kernel supports multiple partitions for hosting both system and user functions.

To facilitate adoption, ISOSCELES is designed to reduce the costs of device development, deployment, and regulatory approval to increase the security of the overall U.S. healthcare infrastructure. Companies, especially startups, with new products under development and legacy products being upgraded, can use ISOSCELES to shorten development times and increase confidence that safety and security are jointly addressed.

ISOSCELES leverages time, space, and IO partitioning, common in other CPS such as avionics, to support medical devices containing multiple levels of software criticality, *without requiring that all the software is verified and validated to the highest level of criticality.* With strong partitioning, ISOSCELES-based medical devices will

SafeThings'17, November 5 2017, Delft, Netherlands

ACM ISBN 978-1-4503-5545-2/17/11…$$15.00
https://doi.org/10.1145/3137003.3137008

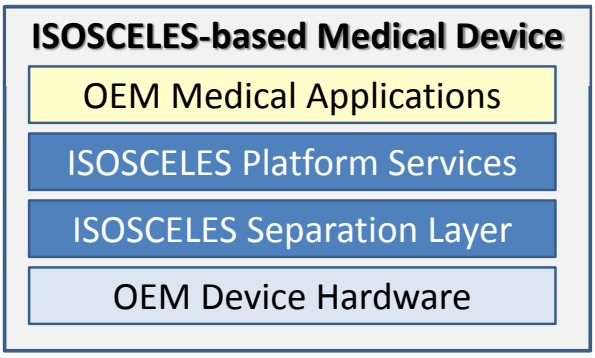

Figure 1: ISOSCELES layering structure

prevent the weakest link from corrupting the entire device, while reusing as much medical application logic and OEM hardware as possible without compromising safety (see Figure 1). Rather than verifying the entire system to the highest regulated criticality level, ISOSCELES components can be reviewed and approved to their own, appropriate regulatory level (including non-regulated). This will also significantly reduce the cost of software updates, as verification and approval will be limited to the partition that changes.

This work contributes by:

(1) Providing an overview of a reference architecture for medical devices and safety-critical IoT,
(2) Identifying services which are critical to such an architecture,
(3) Describing separation technologies applied to support the vision, and
(4) Identifying useful forms of tool support for designing safety-critical IoT.

2 BACKGROUND

There are three main challenges to basing the reference architecture on a separation kernel and hypervisor: the skill of the implementer, the available development time and funds, and supporting the regulatory path for devices based on this design.

2.1 Supplementing Expertise with Tools

Small medical startups might not have skills in these technologies. We address this challenge by developing and demonstrating a tool to model, using intuitive graphical visualizations, the hypervisor partitions and separation kernel rules (see Section 4). The modeling framework provides a developer- and evaluator- friendly view of the architecture into which a variety of information flows, access control, safety policies, and hazard analyses can be integrated. The lower-level separation kernel and hypervisor configuration rules can be generated from the high-level modeling view, providing strong traceability between the implementation and the specification, verification, and risk management artifacts.

ISOSCELES uses analytical models to support design, development, and risk assessment activities to enable users to create a device based on the reference architecture. The Architecture Analysis and Design Language (AADL) [22] provides the necessary expressive power for medical device interoperability, security, and safety properties [7, 9, 18]. AADL was created in response to the high cost associated with failed subsystem integration attempts due to ambiguous or poorly documented component interfaces, and is used in many complex system development efforts.

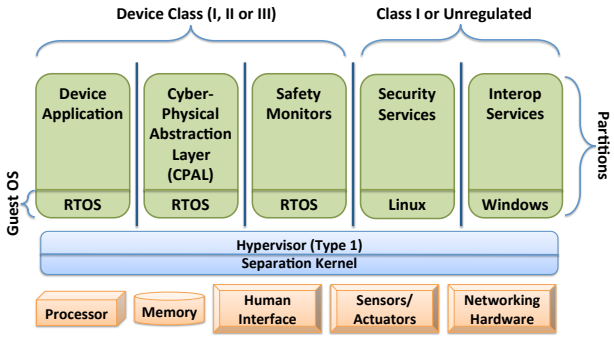

Figure 2: ISOSCELES implements partitioning via separation kernel and hypervisor technologies

2.2 Reducing Costs via Modularization/Reuse

The second challenge is the cost, both in terms of the processor performance required to host the reference architecture and the license cost of the separation kernel and hypervisor. The first issue to be relatively minor: while a small cost increase might be perceived for processors with "virtualization extensions" needed for strong partitioning, the cost savings from development, security by design, and simplified post-market support will well exceed the incremental increase in unit costs for the required hardware. ISOSCELES is developed to such that the choice of separation kernel is a pluggable module, as long as the kernel provides a minimum level of isolation and security functionality, implemented either fully within the kernel, or in the ISOSCELES modularization "glue" code. Depending on the risk profile of the device being built, the less formally proven separation mechanisms provided by open source hypervisors may still provide a much more safe and secure platform than the monolithic designs presently being fielded.

This combination of technologies provides several benefits to the user, including:

- An architecture designed to support both safety and security.
- Partitioning that allows components of lower-criticality and uncertain risk profiles to be cost-effectively combined with critical components; partitioning prevents faults that originate in lower criticality components from propagating into higher-criticality.
- Reusable safety and security services that detect and notify application components of faults. This allows developers to focus on the core medical functionality.
- Model-based architecture-centric development techniques that allow interfaces and the system architecture to be captured early in the development process. The models can be analyzed before implementation is complete to detect and correct inconsistencies across functional, safety, and security interfaces. This will significantly reduce costs.
- Models that provide high-level abstractions of systems and allow system wide safety and security policies to be specified in terms of these models.

2.3 Regulatory Submission Artifacts

Even when the technical benefits of separation can be realized, our conversations with manufacturers and regulators indicate that it is still a challenge to present regulatory submissions that treat the technology appropriately. In particular, the technology is novel enough in the medical space that manufacturers and regulators do not have a good grasp on how to phrase risk management and assurance case artifacts that (a) capture the increased assurance that partitioning provides and (b) are organized to support reuse of assurance for platform elements.

To facilitate adoption of ISOSCELES technology, the project is also preparing a variety of artifacts for manufacturers to use in supporting development, assurance, and regulatory submissions. These include requirements and design artifacts, testing infrastructure, hazard analysis results and assurance case structures for the platform elements.

3 ARCHITECTURE AND SERVICES

ISOSCELES relies on a combined separation kernel and Type I hypervisor to provide critical architecture services. The separation kernel enforces allowed inter-partition communication patterns. The hypervisor provides separate virtual machines to the application and service partitions. This allows developers to select appropriate guest operating systems for the different device applications. The combination provides the strong time and space partitioning capabilities needed for mixed-criticality in medical devices.

3.1 Separation Layer

A key feature of ISOSCELES is the combined use of a separation kernel and a Type I hypervisor as the foundation of the reference architecture. A Type I hypervisor runs directly on the hardware and provides virtual abstractions of that hardware, as virtual machines, to multiple instances of guest operating systems. A separation kernel also runs directly on the hardware and supports multiple partitions for hosting both system and user functions. A separation kernel has configurable rules that for each partition p govern (a) the allowed communication pathways between p and other partitions (including specific per-partition read/write policies for each pathway) and (b) p's access to physical hardware (see Figure 2). ISOSCELES uses *model-driven* analysis and auto-generation of key partitioning artifacts to simplify development of IoT platforms. The models are used for analysis of timing, safety, and security properties, and subsequently generate configuration tables used by the separation kernel and risk assessment analysis.

As the foundation for high-confidence real-time systems, separation kernels are subjected to very high levels of assurance. The separation kernel and the associated rules enforce *policies* such as: *Information flow:* only authorized subjects can exchange information, using preconfigured communication channels; *Data Partitioning:* objects can be isolated into separate partitions, such that subjects can only gain access to objects they are authorized to access; *Time Partitioning:* partitions are scheduled according to precise policies that prevent a failing or malicious process in one partition from utilizing computational resources that are budgeted for another partition; *Residual Information Protection:* covert channels cannot exist through unintended transfer of residual state information between partitions; *Damage limitation:* faults in one partition do not propagate to other partitions.

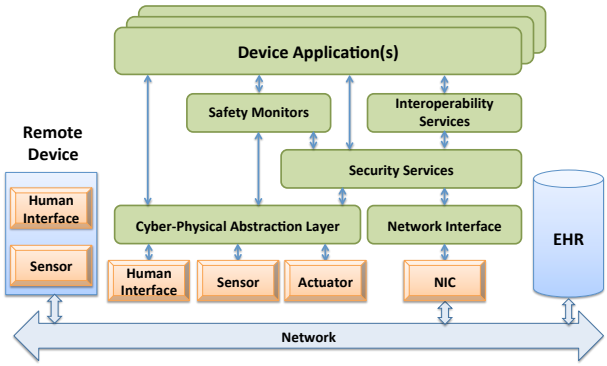

Figure 3: A rich set of service layers support CPS safety and security requirements

3.2 Service Layer

Above the separation layer, ISOSCELES (Figure 3) provides a set of service layers, which allow for configurable connections to the physical elements (sensors, actuators and human interface devices), the physical and data-link network layers, and the safety and security functionality. The services also enable the primary medical functionality of the device *and* enforce security and safety properties. ISOSCELES services rely on strong partitioning to isolate security and monitoring services from the functions they are monitoring, while enabling local and remote physical sensors, actuators, and user interface components to transparently support patient device networks. The *safety monitoring service* performs safety-critical functions and provides situationally safe responses when failures are encountered. The *security service* layer supports functions such as authentication and authorization for connections external to the device, and monitors for potential intrusion attempts.[1]

One of the benefits of mapping this software architecture to the ISOSCELES architecture shown in Figure 2 is the support for different virtualization partitions that can be optimized for the functional needs of the device. Different applications and support service levels will have different needs when it comes to real-time services (such as provided by an RTOS), which security properties are most important, and their potential safety levels and regulatory classes. In the absence of a partitioned architecture, a device requiring one real-time service and one high-safety/high-security function would require all functions to be built with the constraints of an RTOS and the testing requirements of the highest safety level of the device.

3.2.1 Security Services. Security services in the reference architecture can be used without a detailed understanding of their implementation. ISOSCELES can support security services such as:

- **Secure boot:** A device startup mechanism that cryptographically checks software on startup, reducing the chance that "root-kit" type malware can execute at start-up.
- **Secure software updates:** Provides a service that only permits trusted software to be installed on the device.

[1] Intrusion detection is not yet implemented

- **Key/certificate management:** A set of services that allow the manufacturer and end-user (e.g. HDO) to manage the required keys and certificates on the device. This includes certificates for code signing, identification, authentication, as well as a secure means to update them when required.
- **Confidentiality/Privacy:** Services to identify information that must be cryptographically protected when at rest, in transit, and securely deleted when no longer needed.
- **Authentication/authorization:** These services permit hospital-level configurability by allowing pluggable connectivity to enterprise single sign-on (SSO) systems where appropriate.
- **Whitelisting:** A mechanism to permit only explicitly allowed processes to execute on each virtual processor supported by the hypervisor.

3.2.2 Safety Monitoring Services. A configurable safety monitoring function maintains device outputs within safety limits. For example, a device application that controls therapy to a patient may be limited to a certain maximum output rate over a period of time. Inputs from a sensor may be monitored for changes that are not physiologically realistic, preventing bad data injected via a sensor from propagating to the application.

These services will be separated from the device application and interoperability services to protect against both accidental and malicious failures. In addition, this service layer will support watchdog timers, configurable fault response, and fault logging.

3.2.3 Communications and Interoperability Services. With the hypervisor isolating communications, ISOSCELES must have a dedicated external communication service. The configuration interface allows end-users to optimize settings to their local policy.

The ICE, an interoperability architecture standardized in ASTM F2761 (recognized by the US FDA), is the focus of interoperability safety/security standards and FDA considerations of interoperability. Emerging implementations of ICE [14, 16] utilize publish-subscribe architectures such as the OMG DDS for facilitating real-time communication between distributed components. A DDS-based ICE client is includes as part of the communication service, to enable devices built from the reference architecture to operate with multi-device medical "systems of systems." The same technique can be applied to support a wide variety of communication standards, such as HL7 [1], commonly used in the medical domain.

3.2.4 Accurate Time. Clock synchronization issues can be a significant potential source of patient harm [8]. ISOSCELES includes an accurate, synchronized time service to support device functionality and security services. Different time sources also have their own different trust levels.

3.3 Exemplar Design and Implementation

To demonstrate that the reference architecture can be the basis of a safe and secure medical device, we have developed an ISOSCELES-based PCA pump exemplar using COTS components. PCA pumps are well documented, including safety requirements, hazard analysis [5], and assurance case definitions [12]. Our exemplar exercises key portions of the safety, interoperability, and security services of

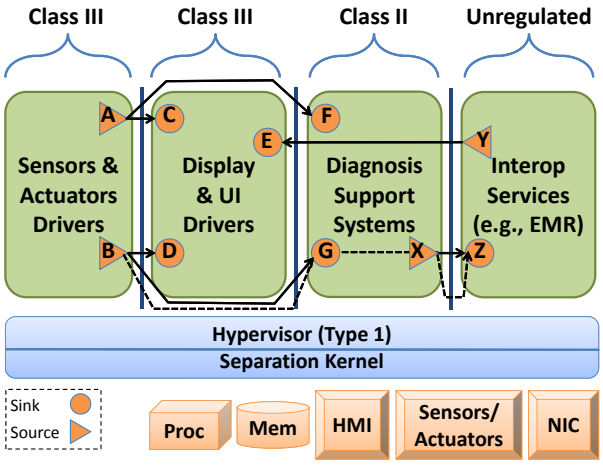

Figure 4: Separation kernel configuration tables govern run-time connections, auto-generated from the AADL models

the GIP architecture[2] and demonstrates local and remote sensing as part of the ISOSCELES platform's cyber-physical abstraction layer.

Requirements: We developed and collated functional, safety, and security requirements for the PCA pump from open sources, including the GIP project, as well as other models from the literature [10, 12, 21, 24], taking into account input from HDOs.

Separation Kernel & Hypervisor: The exemplar uses an established separation kernel and hypervisor that encompasses necessary security features, availability for desired target systems and development environments, and cost. ISOSCELES currently runs on both Xen-based[3] and seL4-based[4] separation frameworks. Both ARM and x86 hardware platforms are supported.

Security Services: We demonstrate security services that are relevant for a PCA pump. This includes secure boot, the ability to manage local keys/certificates, device-to-device authentication, and encryption of information on the network interface.

Safety Monitoring Services: ISOSCELES provides device safety services. For the PCA pump we provide drug flow rate limits, trust-level of drug formulary updates, and controlled rate-of-change of drug output. We intent to implement a programmable "safe mode" to provide an appropriate response in the presence of safety or security fault conditions.

CPAL: The cyber-physical abstraction layer is configurable abstraction for sensors, actuators and user interface devices. This service includes methods to collect and present medical CPS metadata. For the PCA pump we provide flow sensing, a pump actuator, and simple keypad and small display screen. Environmental metadata includes location information, patient association information, and network status information.

Communications/Interoperability Services: For the PCA pump exemplar, we implement protocols to interface with a supervisory controller such as defined by the ICE standard [2], or with an electronic health record system (using HL7 and IHE protocols [11]). The specific protocol is pluggable, and the one selected will depend on the maturity of the defining standards.

[2] http://rtg.cis.upenn.edu/gip.php3
[3] http://www.xenproject.org/
[4] https://sel4.systems/

4 TOOLS

We use the Architecture Analysis and Design Language (AADL) [22] for modeling the high-level architecture. The language and various annexes such as Error Modeling (EM) [23], are well-defined and tightly specified. AADL's features make it an attractive choice, including: component-and-connector model supporting the description of hardware/software components, structure and refinement, and a rich type system. It includes rigorous semantics for "plug-compatible" ports, which provide a solid foundation for ISOSCELES interface management. The EM Annex [23] supports modeling of different fault types, fault behavior of individual components, inter-component fault propagation, and specification of fault tolerance strategies expected in the actual system architecture. AADL has been successfully applied to similar problem domains: the SAVI effort [7] is developing model-based system engineering methods in air vehicle design to reduce defects in early design phases.

4.1 Modeling

4.1.1 Information Flow Models. The separation kernel in the base ISOSCELES software layer enforces communication paths and privileges between software services and device application software. We represent security policies as restrictions on allowed communications flows, such as which partitions are allowed to read or write information to another partition. Enforcing this at execution time can drastically reduce the attack surface of a networked device, e.g., by implementing the network interface in its own partition, with tightly constrained interactions to other partitions.

AADL also provides *flow annotations* to specify intra-partition and end-to-end information flows. For example, Figure 4 shows AADL annotations of inter-partition flows, in which information originating from two source ports (A, B) in the Sensors & Actuators Drivers partition can flow to sink ports in the Display & UI Drivers partition and the Diagnosis Support Systems partition. Flows from source port A are constrained to flow to sink ports C and F, while flows from source B can flow to sinks D and G. The Sensors & Actuators Drivers partition cannot send data directly to the Interop Services partition. We developed additional AADL model annotations for configuring reference architecture read/write policies. ISOSCELES support tools include an information flow framework that will generate configuration tables for the separation kernel and hypervisor, thereby tightly constraining the communication channels at implementation and deployment based on those defined at design and risk analysis time. No flows are allowed by the separation kernel unless explicitly specified in the AADL model.

Indirect information flow can also be captured using AADL end-to-end flows (represented using a dashed arrow; direct data flows are represented by solid arrows in Figure 4): data is allowed to flow from source A in Sensors & Actuators Drivers to sink G in Diagnosis Support Systems, from where it ultimately flows out of source port X and into sink Z. Sink port Z is in the Interop Services component, so data from Sensors & Actuators Drivers can make its way into Interop Services via Diagnosis Support.

Modeling the reference architecture in AADL permits analysis of several key system properties, such as timing and non-interference. Non-interference tools analyze the flow graphs and report on flows which directly connect partitions with different criticality labels (e.g., *safety critical* vs *non-safety critical* or *contains patient identifying information* and *no-PII*. Such flows could indicate a leak of information or integrity. If necessary, such flows will have to pass through *guards* designed to handle mixed criticality.

Timing properties include schedulability and communications latencies, jitter, and throughput. The separation kernel and virtualization can introduce latencies in the inter-partition communications of the device. These models can be used to expose this timing and allow developers to verify that timing requirements are satisfied.

4.1.2 Risk Management Models. Existing risk management approaches emphasize hazard analysis techniques such as FTA or FMEA, which are largely carried out as "thought experiments" to uncover how failures and faults originating deep within the system may cause unsafe effects. Analyses are typically performed manually, using tables and diagrams that are not directly integrated with formal architectural models or the system's implementation. Moreover, existing methodologies in the medical device space have not been customized to leverage ISOSCELES' fundamental architecture partitioning to limit fault propagation.

AADL's *EM* provides a foundation for addressing many of the challenges specified above. The AADL EM framework [23] enables fault behavior of individual system components and fault propagation across interacting components to be modeled. It also allows modeling of the aggregation of fault behavior and propagation in terms of the component hierarchy, and specification of fault tolerance strategies expected in the actual system architecture. The EM supports qualitative and quantitative assessments of system dependability, i.e., reliability, availability, integrity (safety, security), survivability, and compliance of the system to the specified fault tolerance strategies from an annotated architecture model of the embedded software, computer platform, and physical system.

We utilize the EM Annex to provide a more intuitive *model-based suite of hazard analysis techniques* that are integrated directly with the AADL architecture descriptions. These are used to auto-configure the separation kernel and to capture safety and security policies. Large sections of reports for common hazard analyses such as FMEA and FTA can be auto-generated from these EM annotations in our AADL model. We have extended the AADL EM annotations to provide direct support for hazard analysis in AADL models of ICE applications [17] using Leveson's STPA hazard analysis [13], and security analysis [19].

5 BENEFITS AND CONCLUSION

The combination of hypervisor and separation kernel addresses existing safety and security challenges in the IoT domain and provides many benefits to the end-user of the architecture, including:

- Inherent protection of the device application from information flows not explicitly specified via kernel configuration rules. This reduces the potential attack surface exposed to the network interfaces, and allows assurance efforts to focus on the explicitly declared communication paths.
- Controlled interfaces and enforcement of information flow policies between the cyber-physical domain (sensors, actuators, environmental state information) and the services and applications increases the difficulty of tampering with the integrity of the information to/from these external interfaces.

- Time, space, and IO partitioning and deterministic scheduling guarantees that test environments correspond to the actual device operation, and the addition or removal of functionality has predictable impacts on the primary application.
- Ability to host arbitrary "guest operating systems" to support mixed-criticality applications. For example, a non-safety critical Linux-based user interface application can be constrained to the partition running a Linux virtual machine that has no external networking access.
- The end-user does not have to test all software to the highest criticality level since applications with different criticality can be partitioned from each other. The hard partitioning and controlled interfaces protects from fault propagation.

The ISOSCELES architecture explicitly partitions the application domain or "business logic" from the CPS layer and from the separation and safety/security assurance subsystems. This makes ISOSCELES applicable across a variety of CPS domains that require mixed-criticality partitioning, including medical (as described in this paper), automotive, industrial, and building controls. Only slight changes to the overall implementation are required *even if the entire application domain-specific logic must be replaced*. This is the same property that supports simplified reasoning and easier change management, since different device functions can occupy different partitions, even within a single operational domain. We will demonstrate this versatility in future work.

Medical CPS often are mobile, as e.g., pumps in hospitals and clinics move between rooms. A room change might be grounds for re-authentication, since it could indicate a change of patient and/or clinical staff. While the ability to autonomously detect room or patient changes is a research topic beyond the scope of this paper, providing an infrastructure to support such sensing and decision making, is in scope. For example, low-cost ARM-based System on Chips (SoCs) can measure a rich variety of physical phenomena, including temperature, acceleration, GPS, and even spectroscopy. Future work will integrate location services and other context- and state-awareness into the platform. We intend to prototype a device "drop" sensor to issue an alert, and force re-authentication to clear the device alert. This will address the physical use case of a pump that is knocked over, which must be subsequently checked to acknowledge that external connections are unharmed. We will implement these concept sensors using the separation system so they can be enabled, disabled, or modified without affecting the medical application. This separation approach will also reduce the opportunity for side-channel attacks using these sensors, such as using the accelerometers to capture authentication keystrokes.

ACKNOWLEDGMENTS

This material is based on research sponsored by the Department of Homeland Security (DHS) Science and Technology Directorate, Homeland Security Advanced Research Projects Agency (HSARPA), Cyber Security Division (DHS S&T/HSARPA/CDS) BAA HSHQDC-14-R-B0005, the Government of Israel and the National Cyber Bureau in the Government of Israel via contract number D16PC00057, as well as the US National Science Foundation. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Department of Homeland Security, the U.S. Government, the Government of Israel or the National Cyber Bureau in the Government of Israel.

Several artifacts are being developed in collaboration with U.S. FDA engineers through the U.S. NSF FDA Scholar-in-Residence program that provides partial support for this work.

Results, including educational material, requirements and design documents, models, and exemplar artifacts will be made available as open source.[5] We are actively reviewing the architecture with the small and large companies in the medical device community, including members of the Medical Alley Association.[6]

REFERENCES

[1] Health Level Seven International. http://www.hl7.org/.
[2] F-2761: Medical Devices and Medical Systems - Essential safety requirements for equipment comprising the patient-centric integrated clinical environment (ICE) - Part 1: General requirements and conceptual model. Standard, 2009.
[3] The value of medical device interoperability. Technical report, Westhealth Institute, 2013.
[4] TIR57: Principles for medical device information security risk management. Technical report, AAMI, 2016.
[5] D. E. Arney, R. Jetley, P. Jones, I. Lee, A. Ray, O. Sokolsky, and Y. Zhang. Generic infusion pump hazard analysis and safety requirements version 1.0. Technical Report MS-CIS-08-31, University of Pennsylvania, 2009.
[6] FDA. Content of premarket submissions for management of cybersecurity in medical devices. http://www.fda.gov/downloads/MedicalDevices/DeviceRegulationandGuidance/GuidanceDocuments/UCM356190.pdf.
[7] P. H. Feiler, J. Hansson, D. de Niz, and L. Wrage. System architecture virtual integration: An industrial case study. Technical Report CMU/SEI-2009-TR-017, CMU, 2009.
[8] J. M. Goldman. Medical device interoperability ecosystem updates: Device clock time, value proposition, and the FDA regulatory pathway. NSF CPS Large Site Visit PRECISE Center at Penn, 2012.
[9] J. Hansson, B. Lewis, J. Hugues, L. Wrage, P. Feiler, and J. Morley. Model-based verification of security and non-functional behavior using AADL. *IEEE Security & Privacy Magazine*, PP(99), 2009.
[10] J. Hatcliff, A. King, I. Lee, M. Robkin, E. Vasserman, A. MacDonald, S. Weininger, A. Fernando, and J. M. Goldman. Rationale and architecture principles for medical application platforms. In *ICCPS*, 2012.
[11] Integrating the Healthcare Enterprise (IHE). IHE Patient Care Device (PCD) Technical Framework. http://ihe.net/Technical_Frameworks/#pcd.
[12] B. R. Larson, J. Hatcliff, and P. Chalin. Open source patient-controlled analgesic pump requirements documentation. In *SEHC Workshop*, 2013.
[13] N. G. Leveson. *Engineering a Safer World*. Engineering Systems. MIT Press, 2011.
[14] K. Li, S. Warren, and J. Hatcliff. Component-based app design for platform-oriented devices in a medical device coordination framework. In *ACM SIGHIT International Health Informatics Symposium*, 2012.
[15] J. Makower, A. Meer, and L. Denend. FDA impact on U.S. medical technology innovation. Technical report, 2010.
[16] MD PnP Program. Open-Source Integrated Clinical Environment. https://www.openice.info/.
[17] S. Procter and J. Hatcliff. An architecturally-integrated, systems-based hazard analysis for medical applications. In *MEMOCODE*, 2014.
[18] S. Procter, J. Hatcliff, and Robby. Towards an AADL-based definition of app architecture for medical application platforms. In *SEHC Workshop*, 2014.
[19] S. Procter, E. Y. Vasserman, and J. Hatcliff. SAFE and secure: Deeply integrating security in a new hazard analysis. In *SAW*, 2017.
[20] J. Sahoo, S. Mohapatra, and R. Lath. Virtualization: A survey on concepts, taxonomy and associated security issues. In *ICCNT*. IEEE, 2010.
[21] C. Salazar and E. Y. Vasserman. Retrofitting communication security into a publish/subscribe middleware platform. In *SEHC Workshop*, 2014.
[22] Society of Automotive Engineers. Architecture Analysis & Design Language (AADL). Aerospace Standard AS5506, 2004.
[23] Society of Automotive Engineers. AADL error-model annex proposed draft. Aerospace Standard AS5506/1B, 2014.
[24] E. Y. Vasserman and J. Hatcliff. Foundational security principles for medical application platforms. In *WISA*. 2014.

[5] https://www.adventiumlabs.com/isosceles
[6] https://www.medicalalley.org/