April 2018
Volume 16 Issue 4

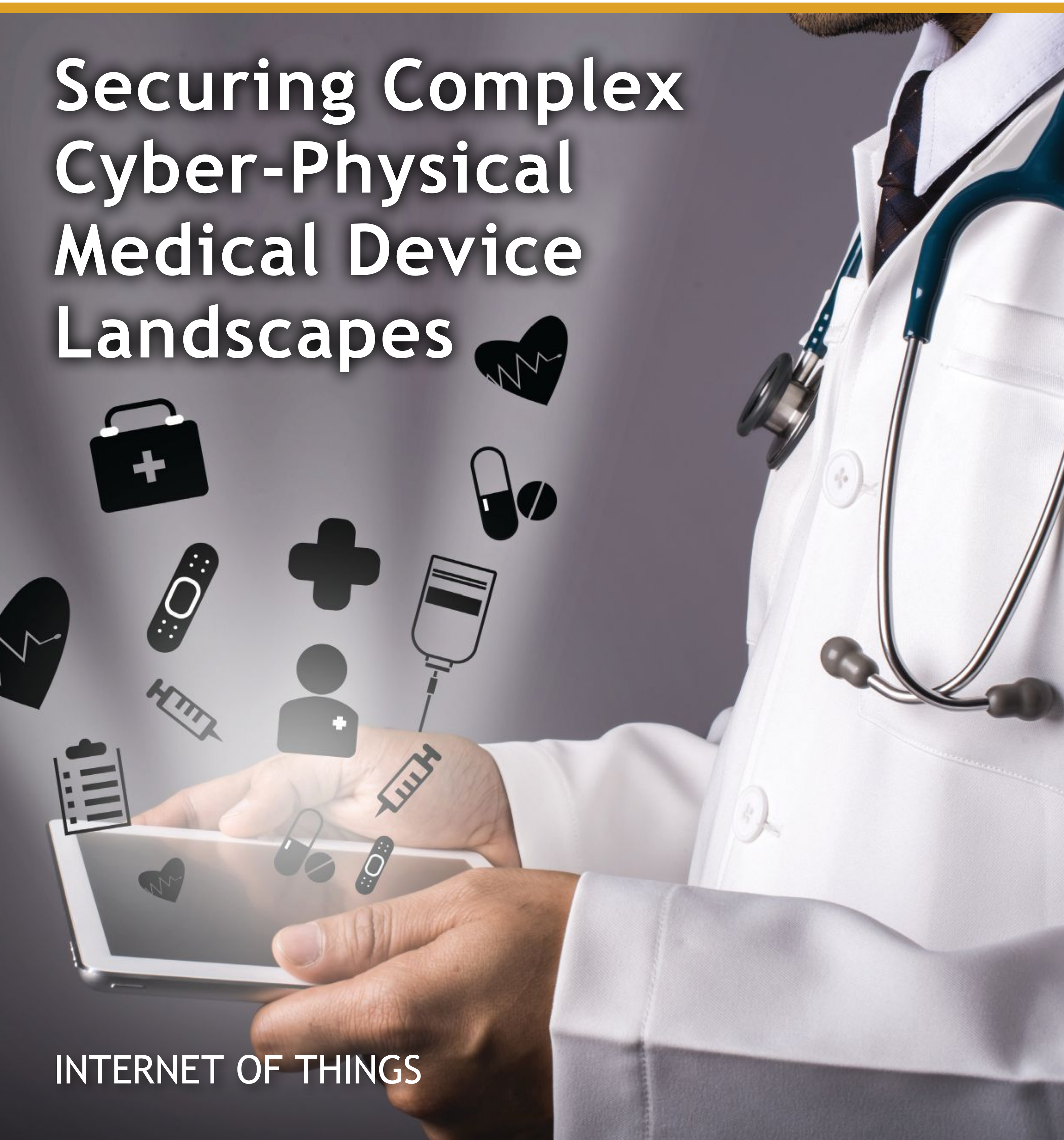
ISSA JOURNAL

The Dangers in Perpetuating a Culture of Risk Acceptance

Using PKI to Build a Secure Industrial Internet of Things

The Two Faces of Innovation: From Safe and Dumb to Vulnerable Smart Products and Infrastructure

Cyber-Physical Intelligence



Securing Complex Cyber-Physical Medical Device Landscapes

INTERNET OF THINGS

Securing Complex Cyber-Physical Medical Device Landscapes

By Ulrich Lang

The author presents innovative approaches to cybersecurity that should be considered to securely integrate medical device landscapes (and many other IoT environments) in the coming years as IoT rapidly matures.

Abstract

In this article we will present innovative approaches to cybersecurity that should be considered to securely integrate medical device landscapes (and many other IoT environments) in the coming years as IoT rapidly matures. The article is based on the results of several government-funded R&D projects, in particular a research project to secure a cyber-physical medical environment (for Defense Health Program, DHP), and a research project to automate access control policy testing (for National Institute of Standards and Technology, NIST).

The Internet of Things (IoT) is the network of physical devices, vehicles, home appliances, and other items embedded with electronics, software, sensors, actuators, and connectivity that enables these objects to connect and exchange data [1]. The IoT is going to be transformational—significantly impacting most industries and parts of society. Experts estimate that the IoT will rapidly grow to 30 billion objects within just two more years [2] (for comparison, in 2015 there were approximately 4.9 million things connected to the Internet).

Importantly, IoT will also play a major role in achieving "smart health care" to improve patient care/experience, efficiency, and outcomes. Hospitals already use many medical devices today (e.g., numerous monitoring and pump devices, etc.) though mostly not in a very interconnected fashion. Presently, in most cases, there is a human (e.g., nurse) in the loop to ensure safety because the devices in use have not been designed with the security in mind that is required for autonomous operation (e.g., monitoring). According to the latest research, US Department of Health plans to (eventually) save up to USD 300 billion from the national budget due to medical innovations [3].

Such future medical device landscapes pose many cybersecurity and privacy challenges because most of these are "cyber-physical" systems where cybersecurity breaches (and other failures) could directly impact the physical safety of patients.

Today, medical IoT is not fully matured yet, leading to a disconnect: Health IT is increasingly interconnected, while information security is not keeping up. This limits "smart health care" improvements and health IoT in general. In this emerging environment, hospitals need to prioritize patient safety first, which means medical devices are often not integrated; since humans are usually in the loop, there isn't much need for automation. This leads to inefficiencies, and patient care/experience is not as good as it could be. Yet, having humans in the loop actually creates its own risks.

In this article we will present innovative approaches to cybersecurity that should be considered to securely integrate medical device landscapes (and many other IoT environments) in the coming years as IoT rapidly matures. The article is based on the results of several government-funded R&D projects, in particular a research project to secure a cyber-physical medical environment (for the Defense Health Program[1]) and a research project to automate access control policy testing (for the National Institute of Standards and Technology[2]).

The presented approach comprises several parts:

1. **Integrated clinical environment:** OpenICE is an initiative to create a community implementation of an integrated clinical environment (ICE). The initiative encompasses not only software implementation but also an architecture for a wider clinical ecosystem to enable new avenues of clinical research. The OpenICE project is run by MD PnP. Our research uses the OpenICE reference implementation and DocBox's implementation as an ICE layer.
2. **Secure device communications:** The Data-Distribution Service (DDS) provides secure publish-subscribe communications for real-time and embedded systems. DDS introduces a virtual global data space where applications can share information by simply reading and writing data-objects addressed by means of an application-defined name (topic) and a key. DDS features fine and extensive control of QoS parameters. Our research uses RTI DDS Connext, a leading DDS implementation provided by Real-Time Innovations (RTI), Inc. OpenICE uses RTI DDS as a communications layer.
3. **Security policy automation** simplifies the management and technical implementation of security policies. It allows security professionals to manage rich security policies consistently in one place and often automatically technically enforce the managed policies across many devices, layers, and technologies. Our research uses ObjectSecurity OpenPMF, which generates technical policy enforcement for DDS, networks, etc., from generic security policies and imported information about users, systems, applications, networks, etc.

Cyber-physical systems

NIST defines cyber-physical systems as co-engineered interacting networks of physical and computational components [4], which will form the foundation for critical infrastructure and emerging/future smart services. Cyber-physical systems will improve quality of life in many areas, such as "smart" cities, transportation, hospitals, and energy. While cyber-physical systems can be used to improve safety (e.g., public safety), they also pose potential cybersecurity risks especially because cybersecurity could impact every patient's physical health and safety (in non-cyber-physical systems,

[1] ObjectSecurity LLC is a subcontractor to Real-Time Innovations (RTI), Inc. for this SBIR Phase II contract, focusing on the topics described in this article. More information about the R&D project: https://sbirsource.com/sbir/awards/167769-methodologies-and-tools-for-securing-medical-device-systems-in-integrated-clinical-environments-ice.

[2] This SBIR (Phase II) was awarded to ObjectSecurity LLC. RTI Inc was a subcontractor in Phase I. More information about the R&D project: https://objectsecurity.com/nist.

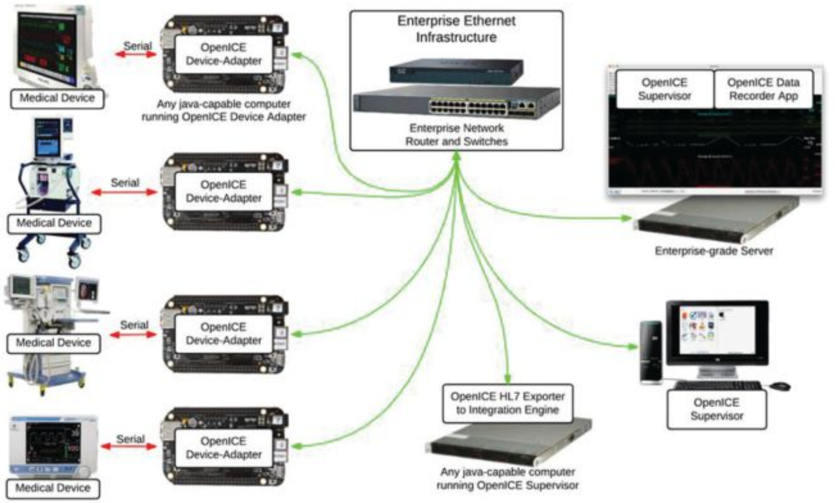

Medical Device
Serial
OpenICE Device-Adapter
Any java-capable computer running OpenICE Device Adapter
Medical Device
Serial
OpenICE Device-Adapter
Medical Device
Serial
OpenICE Device-Adapter
Medical Device
Serial
OpenICE Device-Adapter
Enterprise Ethernet Infrastructure
Enterprise Network Router and Switches
OpenICE Supervisor
OpenICE Data Recorder App
Enterprise-grade Server
OpenICE Supervisor
OpenICE HL7 Exporter to Integration Engine
Any java-capable computer running OpenICE Supervisor

Figure 1 – Example OpenICE system setup [5]

direct/immediate harm is usually directed towards stealing or changing sensitive, valuable information).

Smart health care: Integrated medical devices should detect and respond automatically

Smart health care will comprise at least:

- Networked, smart, and semi-autonomous devices
- Clinical, business, and building systems communication
- Real-time analytics, tracking, etc.
- Automation

Healthcare providers care about smart health care because it has the significant potential to measurably improve patient care, a patient's experience, and health provider efficiency. In conjunction, healthcare providers (especially hospitals) must implement significantly improved cybersecurity to ensure patient safety, security, and privacy when deploying these automated, autonomous IoT environments.

Figure 1 illustrates an example of an "integrated clinical environment" (ICE) based on OpenICE. The ambitious goal of OpenICE is to provide automated, autonomous communications between medical devices in hospitals. In this example, dongle devices (e.g., secured ARM7 platforms) are put in front of the serial connections of conventional medical devices. This is needed because many legacy medical devices that have no security will be used for the foreseeable future. In the example, medical devices (on the left) are connected to the OpenICE system via OpenICE device adapters; the adapters securely relay traffic between the devices and the ICE supervisor. The supervisor provides a "single pane of glass" view of each patient's heath by collecting, monitoring, and analyzing the data from every device in real-time, alerting the nurse when important changes in the patient's data arise.

Health IT needs better access control

HIPAA breach data underscores that a significant proportion of breaches are due to unauthorized access by hospital employees [6]. On top of that, many hacker attacks focus on stealing employee credentials to gain unauthorized access. Health IT access control simply isn't capable of providing the security that will be required: access is overprovisioned, too simplistic (identity-based access control (IBAC) and role-based access control (RBAC)), and unmanageable due to being fragmented and siloed. Add the IoT and ransomware attacks such as Mirai (2016) and WannaCry (2017) into the mix (which both affected medical devices) and it becomes even clearer that locking down access at all levels to the "minimum necessary" is sorely needed.

HIPAA fines are rarely administered but are a risk for health providers (fines have gone up to $4.8M for non-compliance after a breach). Furthermore, with improved access control (and other cybersecurity), hospitals could enable "smart business," resulting in a forecasted $8.5M in annual savings per US hospital (US$36 billion US total) [7].

Identity and access management (IAM) is an important program/initiative for health providers (and most organizations). However, in our experience, hospital IAM systems are often not fully mature (based on hospital IAM road-map projects the author has completed recently).

- IAM systems are distributed (there is a main IAM and numerous sub-IAMs)
- There are custom batch processes to keep information synchronized
- Onboarding/offboarding is done using manual processes with limited checks
- There is almost no work-flow automation
- There is a flat network with little isolation (including cyber-physical)
- There is almost no fine-grained, automated access control (which is partly IAM)

An example of a maturity score by IAM component for a typical hospital is shown in figure 2, clearly showing plenty of room for improvement around access control.

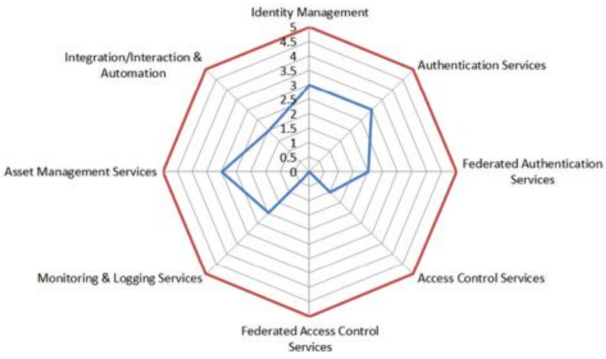


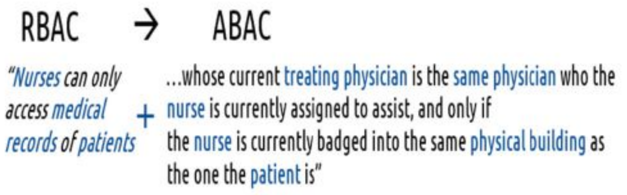
Fig 2 – Typical hospital IAM maturity score (source: ObjectSecurity)

Close-up on access control

Implementing effective access control across hundreds of devices and layers that form an interconnected IoT device landscape is highly complex. Protections are needed in many places and in many differing, overlapping, technical configurations. For the example clinical ICE environment presented in figure 1, consistent access policies would need to be authored and maintained to dynamically control information flows between the OpenICE device adapters, OpenICE supervisor systems, the enterprise infrastructure, electronic medical record (EMR) systems, identity and access management systems, etc.

Attribute-based access control (ABAC) has been around for a while as part of the solution to implement more granular and fine-grained access policies. According to some industry analysts, ABAC will be used increasingly to protect critical assets in coming years. While ABAC can be quite powerful if implemented correctly, it is also often far too hard to manage/author, implement, integrate, and audit (described more below). Adoption challenges are not only technical, but also psychological. ABAC still collides with the reactive cybersecurity "group think" that revolves around monitoring, coarse-grained blacklisting and RBAC.

The principles behind ABAC are simple: access is expressed as more or less Boolean rules that can draw on various information sources to determine an access decision. Figure 3 illustrates the shift (or rather extension) from RBAC to ABAC (note that the example is for illustrative purposes only, but will not fit to current business processes of most hospitals):



RBAC → ABAC

"Nurses can only access medical records of patients + ...whose current treating physician is the same physician who the nurse is currently assigned to assist, and only if the nurse is currently badged into the same physical building as the one the patient is"

Figure 3 – From RBAC to RBAC example

While such policies appear intuitive, the "plumbing" under the hood can be highly complex to manage these dynamic information sources for attribute values (i.e., the values need to be fetched at decision-making time by the ABAC system for the blue parts in the example policy in figure 3), which are then compared to the values in the policy by a "policy decision point" to determine an access decision.

Furthermore, it is usually hard to implement "defense in depth" using ABAC—there are too many different technical configurations across too many different devices and different layers. For example, how would you configure your firewalls from an ABAC rule like in the example above. Add to that the fact that IT landscapes (and users) will change over time, it is clear that ABAC could become an administrative nightmare if applied pervasively across many devices and layers. It is therefore often only used to add some granularity to user access (making up a small part of the overall access control challenges faced).

A close-up on security policy automation

Security policy automation is an approach to simplify the management and technical implementation of richer, more dynamic access policies. Depending on the tools used, it allows security professionals to manage rich security policies consistently in one place, using an intuitive, generic policy representation. Security policy automation is a policy management umbrella that helps define, manage, and enforce consistent policy management of rich policies (including access policies) across many devices, layers, and technologies and (often automatically) technically enforce the managed policies.

Some security policy automation tools additionally simplify policy management even in the face of dynamic changes. This is achieved by automatically detecting, importing, and analyzing information about the users, systems, applications, networks, etc. The imported information is used to fill in the concrete details about the generic policy authored by the security professional.

The most comprehensive policy automation round-trip includes all these features, as shown in figure 4 (overlaid on the medical IoT landscape in figure 3).

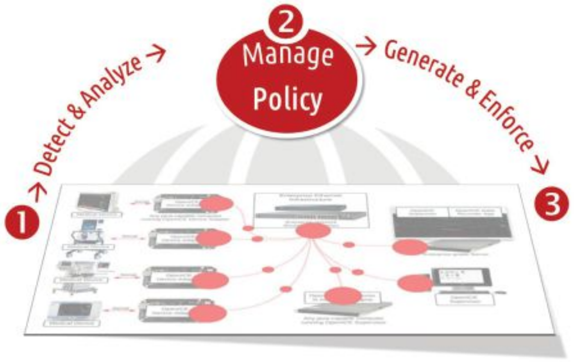


Figure 4 – Run-time security policy automation round trip

We have concretely implemented policy automation for an OpenICE medical device landscape as part of a government-funded R&D subcontract. The scenario included infusion pump devices, patient monitoring devices, and ICE managers. Device communications in OpenICE are handled by a real-time, secure data bus built using the Data Distribution Service (OMG DDS): a publish/subscribe middleware platform provided by the prime contractor, Real-Time Innovations. For our demo scenario, we decided to define security based on device attributes (identity, IP/MAC, DDS topics, etc.) rather than on user attributes (e.g., identity) because OpenICE already groups devices by patient, and simulating a full hospital work flow was beyond the scope of the demo test bed.

The policy automation solution included three main steps:

1) Import, Analyze, Visualize
2) Author Intuitive Policies

3) Enforce Policies

To simplify the process for users, the security policy automation system includes a work-flow automation feature and provides prebuilt policy automation work flows, leading the user seamlessly through the different policy automation steps. Users can also specify their own work flows as needed.

1) Import, analyze, visualize

- Imported network traffic logs provide information about the actual traffic flowing across the system.
- Information about DDS participants and publish/subscribe topics is automatically detected and captured by a DDS discovery tool. It provides information about which traffic is important (payload traffic vs. discovery "chatter," for example), and who is talking to whom.
- The policy automation system automatically configures the "building blocks" for policies based on the importers and exporters available. For example, "the requestor's IP address" is only available during policy authoring if IP addresses are imported or policies based on IP addresses will be enforced.
- Device types are automatically detected based on various indicators, including matching the device type profile in terms of traffic patterns and other factors. This allows the security policy automation system to automatically create a model of the devices that are on the network and what their interactions are.
- This model is visualized in 2D, 3D, and VR to give users visibility into medical device activities.
- Subsets of the datasets of the functional system model can be selected and analyzed, for example, kinds of traffic that should be whitelisted later.

2) Author intuitive policies

Thanks to the work flow automation feature, default generic policy templates are automatically loaded for the particular use-case scenarios (in our case hospital medical device landscapes). These templates include generic rules that apply to most hospitals. These generic policies use wildcards, mappers between building blocks, inheritance/aliases, etc. (e.g., a "patient monitor is allowed to send dosage data to infusion pumps"). Mappers, inheritance/aliases and wildcards can be used together with datasets in the functional system model to automatically generate detailed, technically enforceable policies from the authored policies.

In our demo test bed, some of the template rules are quite simple, for example, a wildcard rule that whitelists all detected user traffic (as opposed to middleware/network "chatter"). To modify the policy (if needed), several policy editors can be used interchangeably, including a graphical editor and a natural language text editor.

The policy automation system then automatically calculates the "low-level" technical policy model from the authored generic policy, the templates, the datasets, etc. This model is later used to generate concrete technical rules and configurations.

A formal verification tool can automatically verify that a generated policy meets specified invariants [8]. In addition, documentation is produced, including natural language text documents of the policy, and a compliance report that details how a policy was exactly generated from the various inputs.

3) Enforce policies (through software agents and native)

Finally, the policy automation system generates the actual technical enforcement. In our solution, enforcement can be via local software agents that intercept information flows, or by exporting "native" security configurations for existing security products and features.

For our OpenICE solution, we did both. First, we automatically generated policy configurations for security policy automation software agents, both for DDS (tied into DDS' security system via its access control plugin interface), and for the network (using our NetPEP), which interfaces directly with iptables, tcpdump, and syslog on the protected system.

Secondly, we also automatically generated textual configurations of various existing features to underscore the point that effective enforcement on multiple layers and devices is also feasible without the need to install a local software agent on each system. We generated OMG DDS-conformant *security permission.xml* and *governance.xml* files to lock down the middleware layer and iptables/arptables (Linux) and advanced firewall (Windows) configuration scripts to lock down the network layer. Those scripts and configurations needed to be pushed to each system manually (using a script).

We also demonstrated these features (for the abovementioned NIST SBIR) on our 20-node Raspberry Pi R&D cluster with individual touchscreens.[3]

Example in action

The following illustrates an example flow through the described approach from the perspective of the security professional tasked to implement technical security for the interconnected device environment.

First, users can optionally use work flow automation to guide them through the necessary security policy automation steps.

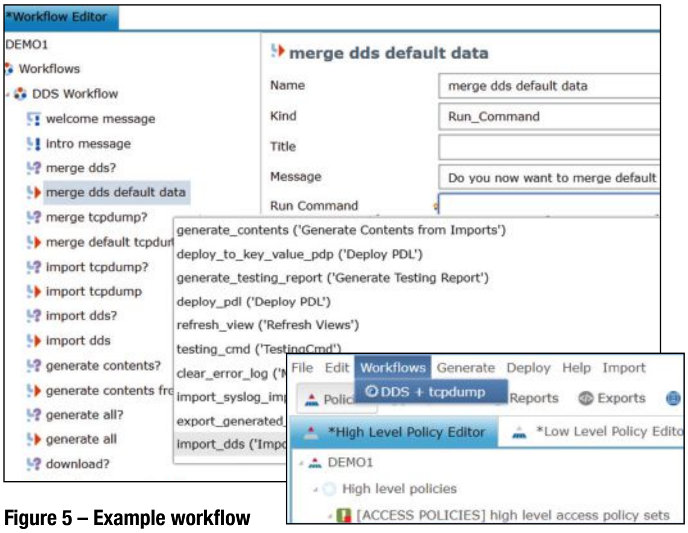
Continued on page 44

[3] See "ObjectSecurity R&D 20-node Raspberry Pi 3 Cluster," ObjectSecurity – objectsecurity.com/cluster.

Securing Complex Cyber-Physical Medical Device Landscapes

Continued from page 18

More advanced users can define their own work flows. In our implementation, work flows can consist of dialog boxes and actions: first, set up the security policy automation tool for DDS and tcpdump imports (including policy building blocks, data selection templates, policy templates, etc.) and import those data sources; then generate subsets of the imported data for whitelisting; finally, generate and download technical rules and configurations (figure 5).



Name: merge dds default data

Kind: Run_Command

Title:

Message: Do you now want to merge default

Run Command:

generate_contents ('Generate Contents from Imports')
deploy_to_key_value_pdp ('Deploy PDL')
generate_testing_report ('Generate Testing Report')
deploy_pdl ('Deploy PDL')
refresh_view ('Refresh Views')
testing_cmd ('TestingCmd')
clear_error_log ('N
import_syslog_im
export_generated
import_dds ('Impo

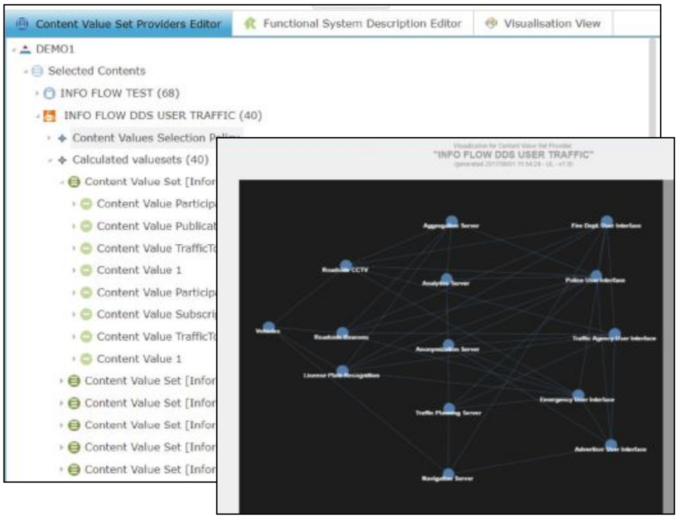
Figure 5 – Example workflow

DDS information is captured on the running system using a DDS discovery dump tool that essentially listens to all DDS discovery traffic and generates an XML file with information about all participants (including IP address), which topics they publish/subscribe, and (where known) on which ports (figure 6).

At the same time, network traffic is captured using tcpdump. In this example, both files are simply uploaded using drag-and-drop file uploaders (automatic detection and importing is an alternative, which would be preferable in more dynamic environments).

In general, the more data sources the security policy automation can tap into, the more detailed the "as is" picture gets—example information sources include network equipment and directory/IAM systems.

The imported information is then analyzed and merged automatically. For example, particular IP:port→IP:port network traffic is tagged information indicating matching DDS topic information flows (Side note: this is not as trivial as it sounds because DDS uses random ports for publishers).



DEMO1
Selected Contents
INFO FLOW TEST (68)
INFO FLOW DDS USER TRAFFIC (40)
Content Values Selection P
Calculated valuesets (40)
Content Value Set [Infor
Content Value Particip
Content Value Publicat
Content Value TrafficT
Content Value 1
Content Value Particip
Content Value Subscri
Content Value TrafficT
Content Value 1
Content Value Set [Infor
Content Value Set [Infor
Content Value Set [Infor
Content Value Set [Infor
Content Value Set [Infor

Figure 7 – Selected subsets of the imported data

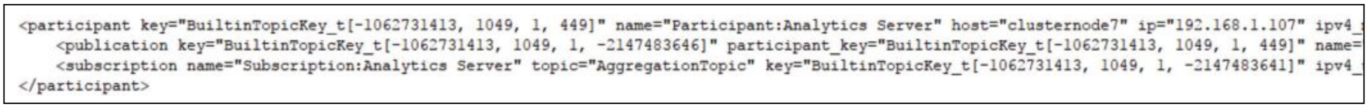
The next step of this example involves calculating/selecting relevant subsets of the imported information. In this simple example, assume we are interested in whitelisting all DDS traffic that actually convey application information (as opposed to discovery and other network "chatter"). The initial setup work flow step already set up numerous useful subsets. The calculated results can be visualized (figure 7).

Now that the data is imported and selected, it can be used in security policies. For example, a simple, generic policy (automatically set up as a default good practice policy) is to only allow all DDS traffic that actually convey application information. This minimizes lateral attacker movement if a system gets compromised. As shown in figure 8, policies include wildcards ("*"), which can be linked to the selected datasets (or the system can infer which dataset is applicable).

It is also possible to reuse (import) already existing policies, for example, from firewalls, IAM systems, XACML deployments, etc.

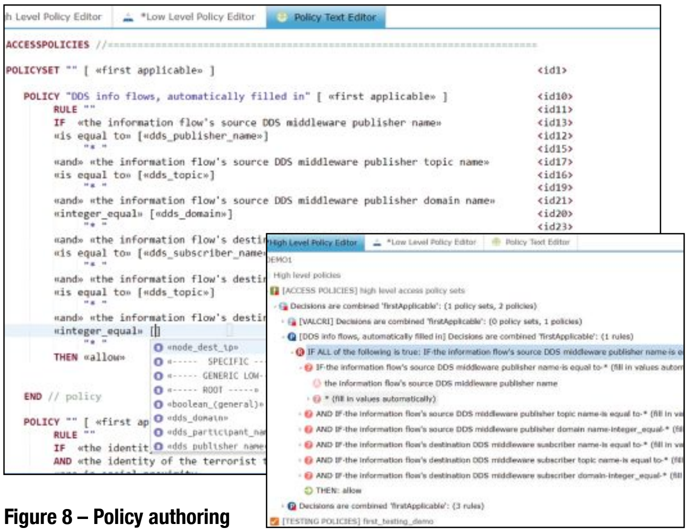
The next step then calculates the matching "low-level policies" – technical policies in a generic syntactic representation. Exporters take that information and turn it into the specific syntactic output required for actual enforcement, in our example just DDS security and host firewalls (figures 9 and 10).

In our implementation, these files can be simply downloaded from the web interface to be manually installed by the security professional. For a more integrated experience, automated configuration is of course possible using scripts and APIs.



```
<participant key="BuiltinTopicKey_t[-1062731413, 1049, 1, 449]" name="Participant:Analytics Server" host="clusternode7" ip="192.168.1.107" ipv4_
    <publication key="BuiltinTopicKey_t[-1062731413, 1049, 1, -2147483646]" participant_key="BuiltinTopicKey_t[-1062731413, 1049, 1, 449]" name=
    <subscription name="Subscription:Analytics Server" topic="AggregationTopic" key="BuiltinTopicKey_t[-1062731413, 1049, 1, -2147483641]" ipv4_
</participant>
```

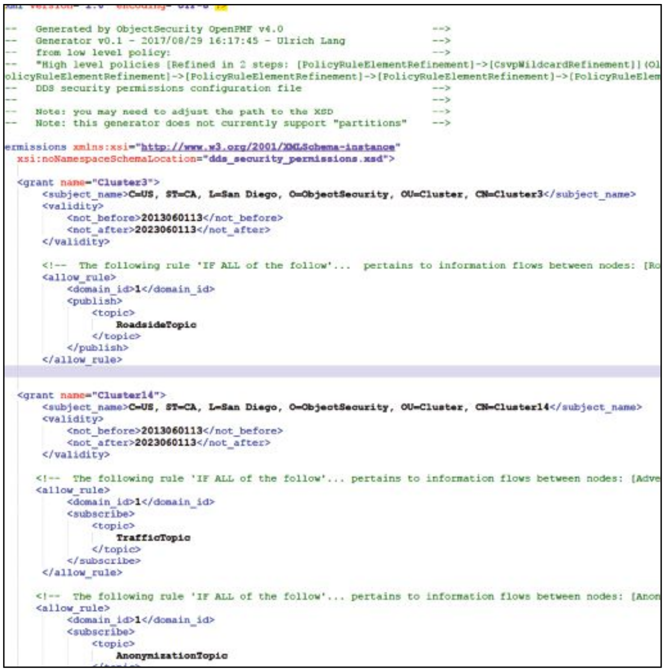
Figure 6 – DDS discovery traffic xml file



```
ACCESSPOLICIES //==========================================================

POLICYSET "" [ «first applicable» ]                                          <id1>

  POLICY "DDS info flows, automatically filled in" [ «first applicable» ]    <id10>
      RULE ""                                                                <id11>
      IF  «the information flow's source DDS middleware publisher name»      <id13>
      «is equal to» [«dds_publisher_name»]                                   <id12>
          "* "                                                               <id15>
      «and» «the information flow's source DDS middleware publisher topic name»   <id17>
      «is equal to» [«dds_topic»]                                            <id16>
          "* "                                                               <id19>
      «and» «the information flow's source DDS middleware publisher domain name»  <id21>
      «integer_equal» [«dds_domain»]                                         <id20>
          "* "                                                               <id23>
      «and» «the information flow's desti
      «is equal to» [«dds_subscriber_name
          "* "
      «and» «the information flow's desti
      «is equal to» [«dds_topic»]
          "* "
      «and» «the information flow's desti
      «integer_equal» []
          "* "          «node_dest_ip»
      THEN «allow»      «----- SPECIFIC --
                        «----- GENERIC LOW-
                        «----- ROOT -----»
  END // policy         «boolean_(general)»
                        «dds_domain»
  POLICY "" [ «first ap «dds_participant_na
      RULE ""
      IF  «the identit  «dds_publisher_name
      AND «the identity of the terrorist t
```

```
DEMO1
 High level policies
 [ACCESS POLICIES] high level access policy sets
   Decisions are combined 'firstApplicable': (1 policy sets, 2 policies)
     [VALCRI] Decisions are combined 'firstApplicable': (0 policy sets, 1 policies)
     [DDS info flows, automatically filled in] Decisions are combined 'firstApplicable': (1 rules)
       IF ALL of the following is true: IF·the information flow's source DDS middleware publisher name·is e
         IF·the information flow's source DDS middleware publisher name·is equal to·* (fill in values autom
           the information flow's source DDS middleware publisher name
           * (fill in values automatically)
         AND IF·the information flow's source DDS middleware publisher topic name·is equal to·* (fill in va
         AND IF·the information flow's source DDS middleware publisher domain name·integer_equal·* (fil
         AND IF·the information flow's destination DDS middleware subscriber name·is equal to·* (fill in va
         AND IF·the information flow's destination DDS middleware subscriber topic name·is equal to·* (fil
         AND IF·the information flow's destination DDS middleware subscriber domain·integer_equal·* (fil
         THEN: allow
     Decisions are combined 'firstApplicable': (3 rules)
 [TESTING POLICIES] first_testing_demo
```

Figure 8 – Policy authoring

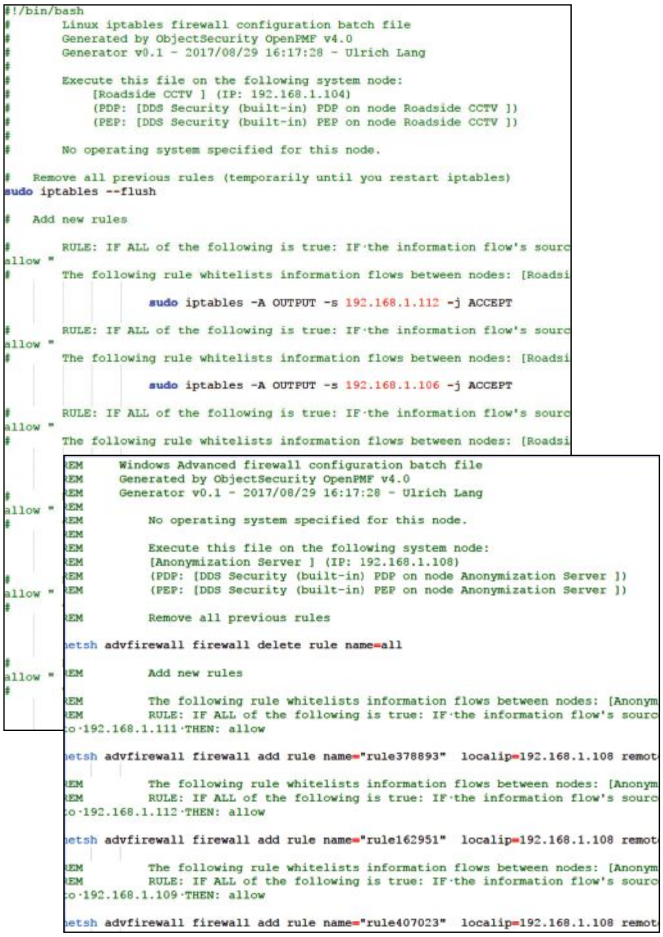


```
xml version="1.0" encoding="UTF-8"?>

--   Generated by ObjectSecurity OpenPMF v4.0                     -->
--   Generator v0.1 - 2017/08/29 16:17:45 - Ulrich Lang           -->
--   from low level policy:                                       -->
--   "High level policies [Refined in 2 steps: [PolicyRuleElementRefinement]->[CsvpWildcardRefinement]](Ol
olicyRuleElementRefinement]->[PolicyRuleElementRefinement]->[PolicyRuleElementRefinement]->[PolicyRuleElem
--   DDS security permissions configuration file                  -->
--                                                                -->
--   Note: you may need to adjust the path to the XSD             -->
--   Note: this generator does not currently support "partitions" -->

ermissions xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="dds_security_permissions.xsd">

 <grant name="Cluster3">
    <subject_name>C=US, ST=CA, L=San Diego, O=ObjectSecurity, OU=Cluster, CN=Cluster3</subject_name>
    <validity>
       <not_before>2013060113</not_before>
       <not_after>2023060113</not_after>
    </validity>

    <!--  The following rule 'IF ALL of the follow'...  pertains to information flows between nodes: [Ro
    <allow_rule>
       <domain_id>1</domain_id>
       <publish>
          <topic>
             RoadsideTopic
          </topic>
       </publish>
    </allow_rule>

 <grant name="Cluster14">
    <subject_name>C=US, ST=CA, L=San Diego, O=ObjectSecurity, OU=Cluster, CN=Cluster14</subject_name>
    <validity>
       <not_before>2013060113</not_before>
       <not_after>2023060113</not_after>
    </validity>

   <!--  The following rule 'IF ALL of the follow'... pertains to information flows between nodes: [Adve
   <allow_rule>
       <domain_id>1</domain_id>
       <subscribe>
          <topic>
             TrafficTopic
          </topic>
       </subscribe>
    </allow_rule>

   <!--  The following rule 'IF ALL of the follow'... pertains to information flows between nodes: [Anon
   <allow_rule>
       <domain_id>1</domain_id>
       <subscribe>
          <topic>
             AnonymizationTopic
```

Figure 9 – Standard DDS security permissions XML file

The more exporters are supported by the security policy automation solution, the more pervasive defense in depth can be achieved "at the click of a button" (e.g., network, host OS, middleware, application, databases, etc.).

Some security policy automation solutions often also come with their own decisioning/enforcement agents, for example XACML PDPs automated network configuration software agents and SDKs for developers to call. To prove this point during our research (and somewhat redundant), our example implementation also generates internal configurations for enforcement on the DDS layer and using our security policy automation solution's own decision/enforcement agent. The implementation also includes an automated network security enforcement agent, which interfaces with network tools (e.g., iptables and tcpdump) to allow push-button traffic capture and enforcement.



```
#!/bin/bash
#       Linux iptables firewall configuration batch file
#       Generated by ObjectSecurity OpenPMF v4.0
#       Generator v0.1 - 2017/08/29 16:17:28 - Ulrich Lang
#
#       Execute this file on the following system node:
#           [Roadside CCTV ] (IP: 192.168.1.104)
#           (PDP: [DDS Security (built-in) PDP on node Roadside CCTV ])
#           (PEP: [DDS Security (built-in) PEP on node Roadside CCTV ])
#
#       No operating system specified for this node.

#   Remove all previous rules (temporarily until you restart iptables)
sudo iptables --flush

#   Add new rules

#       RULE: IF ALL of the following is true: IF·the information flow's sourc
allow "
#       The following rule whitelists information flows between nodes: [Roadsi

                sudo iptables -A OUTPUT -s 192.168.1.112 -j ACCEPT

#       RULE: IF ALL of the following is true: IF·the information flow's sourc
allow "
#       The following rule whitelists information flows between nodes: [Roadsi

                sudo iptables -A OUTPUT -s 192.168.1.106 -j ACCEPT

#       RULE: IF ALL of the following is true: IF·the information flow's sourc
allow "
#       The following rule whitelists information flows between nodes: [Roadsi

#
allow "
#

#
allow "
#

#
allow "
#
```

```
REM     Windows Advanced firewall configuration batch file
REM     Generated by ObjectSecurity OpenPMF v4.0
REM     Generator v0.1 - 2017/08/29 16:17:28 - Ulrich Lang
REM
REM         No operating system specified for this node.
REM
REM         Execute this file on the following system node:
REM         [Anonymization Server ] (IP: 192.168.1.108)
REM         (PDP: [DDS Security (built-in) PDP on node Anonymization Server ])
REM         (PEP: [DDS Security (built-in) PEP on node Anonymization Server ])

REM         Remove all previous rules

netsh advfirewall firewall delete rule name=all

REM         Add new rules

REM         The following rule whitelists information flows between nodes: [Anonym
REM         RULE: IF ALL of the following is true: IF·the information flow's sourc
to·192.168.1.111·THEN: allow

netsh advfirewall firewall add rule name="rule378893"  localip=192.168.1.108 remot

REM         The following rule whitelists information flows between nodes: [Anonym
REM         RULE: IF ALL of the following is true: IF·the information flow's sourc
to·192.168.1.112·THEN: allow

netsh advfirewall firewall add rule name="rule162951"  localip=192.168.1.108 remot

REM         The following rule whitelists information flows between nodes: [Anonym
REM         RULE: IF ALL of the following is true: IF·the information flow's sourc
to·192.168.1.109·THEN: allow

netsh advfirewall firewall add rule name="rule407023"  localip=192.168.1.108 remot
```

Figure 10 –Standard iptables and Windows firewall rule scripts

This simple example was successfully implemented and tested using a 15-node interconnected DDS application. Note that this only illustrates one security policy automation example (using wildcards based on subsets of imported traffic, and only for host firewalls and DDS). Security policy automation often includes many more features to get from authored generic policies to specific technical implementations in many places. We just focused on a particularly intuitive/simple use case for illustrative purposes.

Conclusion

This article explains why medical device landscapes need better security, and why traditional approaches are insufficient. We describe how a security policy automation "umbrella" with work flow automation and prebuilt templates can seamlessly provide the necessary security to enable these interconnected cyber-physical devices to securely operate within hospitals.

Interestingly, while this article focuses mostly on the technical challenges and solutions, we also tackled several non-technical challenges over the course of our research efforts. First, it was difficult to determine policies that should be enforced. While hospitals broadly follow established business processes (especially at the lower echelons), it is not obvious which processes are firm enough to inform security policies. For example, is it realistic to restrict access to a patient record to staff that are not badged into the same building as the patient? Nurses may move around between buildings and may need to look up records in another building before attending to a patient. Even without firm access enforcement, rich access policies can still be useful to implement HIPAA "break the glass" procedures where suspicious or unauthorized access is granted, but needs to be documented/justified. Security policy automation should be used for this as well. Second, hospitals do not have large IT departments and IT budgets, so any successful security solution needs to be as seamless and transparent as possible. Also, legacy devices need to be supported (our ICE implementation uses dongles as wrappers).

In summary, a security policy automation "umbrella" solution can be used to improve medical device cybersecurity. It will provide more dynamic, fine-grained, comprehensive, and manageable access control, which minimizes the risk of lateral attacker movement and ensures HIPAA's "minimum necessary" requirement (access for the right information/devices/people/context only) is met. While this article did not cover user access control due to the specific demo scenario, our security policy automation solution has interfaced with IAM deployments in the past (this is actually one of the better-understood access control layers).

Our approach helps implement powerful technical security policies for users, devices, and applications, while at the same time reducing policy management efforts. The automated process is consistent, testable, documented, robust, and repeatable. The industry needs to move this way. We cannot manually manage technical policies for cyber-physical IoT—it is simply too dangerous.

Acknowledgements

This work is supported by the US Army Medical Research and Materiel Command under contract no. W81XWH-16-C-0175 (RTI, Inc.), and by the National Institute of Standards and Technology (NIST) under grant no. 70NANB16H192 (ObjectSecurity LLC). Also thanks to Paul Pazandak, PhD, Director of Research, Real-Time Innovations, Inc. for comments and edits.

—**Data-Distribution Service** (http://www.omg.org/dds/)

—**ObjectSecurity** (https://objectsecurity.com/)

—**OpenICE** (https://www.openice.info/)

—**Real-Time Innovations, Inc.** (https://www.rti.com)

References

1. "Internet of things," Wikipedia – https://en.wikipedia.org/wiki/Internet_of_things.
2. *Hsu, Chin-Lung; Lin, Judy Chuan-Chuan, "An Empirical Examination of Consumer Adoption of Internet of Things Services: Network Externalities and Concern for Information Privacy Perspectives," Computers in Human Behavior. 62: 516–527. doi:10.1016/j.chb.2016.04.023.*
3. "IoT In Healthcare Industry: See Why It Has A Promising Future," Cleveroad (De. 15, 2017) – https://www.cleveroad.com/blog/iot-in-healthcare-industry--see-why-it-has-a-promising-future.
4. "Cyber-Physical Systems," NIST – https://www.nist.gov/el/cyber-physical-systems.
5. "Example OpenICE System Setups," OpenICE – https://www.openice.info/docs/6_example-setups.html.
6. "Employee Snooping Most Common Cause of HIPAA Security Breaches," HIPAA Journal (Nov 22, 2013) – https://www.hipaajournal.com/employee-snooping-common-cause-hipaa-security-breaches/.
7. Mertz, Leslie. "Saving Lives and Money with Smarter Hospitals," IEEE Pulse (Nov. 7, 2014) – http://pulse.embs.org/november-2014/saving-lives-money-smart-hospital/.
8. This OpenPMF Security Policy Auditor tool was developed as part of another SBIR for NIST and is based on NIST Special Publication 800-192: "Verification and Test Methods for Access Control Policies/Models, NIST (June 2017) – https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-192.pdf.

About the author

Dr. Ulrich Lang (ObjectSecurity CEO, co-founder) received his PhD from the University of Cambridge Computer Laboratory (Security Group) on conceptual aspects of middleware security in 2003, after having completed a Master's Degree in Information Security (Royal Holloway, University of London) in 1997. Ulrich is a renowned thought leader in cybersecurity, privacy, and data analytics/AI and is on the Board of Directors of the Cloud Security Alliance (Silicon Valley Chapter). He may be reached at ulrich.lang@objectsecurity.com.