Formal Security Analysis and Open Standards

An Approach and Case Study

The Open Group FACE™ and SOSA™ US Air Force Technical Interchange Meeting (TIM) Paper by:

Michael R. Clark, Riverside Research
mclark@riversideresearch.org

Anthony Pierce, Riverside Research
apierce@riversideresearch.org

September 2022

Table of Contents

Executive Summary

Security is an important technical concept in the Sensor Open System Architecture™ (SOSA) consortium. Defining security can be extremely difficult but is important as otherwise we can never know if system developers can achieve security of their products. In this paper, we introduce the concept of formal methods, discuss how open architecture standards, such as the SOSA Technical Standard can contribute to formal modeling for security assessments and how formal methods can support the development of open architecture standards. Furthermore, we present an example use case using a formal methods technique known as the Universal Composability (UC) framework. This document is intended for anyone working on open architecture standards to become more aware of what formal methods are and to help cybersecurity practitioners who work on open architecture standards know how formal methods can support standards development. It is not intended as an exhaustive survey on formal techniques or a primer on the UC framework, but rather details the mutual benefits we see of applying formal methods in standards and system design with a relevant use case example.

Introduction

The SOSA Security Subcommittee (SSC) was the first subcommittee organized within the SOSA Consortium. *Securability* is a SOSA quality attribute that is placed second in order of precedence only to *Interoperability*. As described in Technical Standard 1.0, *Securability* "ensures that the fundamental architecture is one that has minimal attack surfaces" and that "systems can be designed so that they can adapt to an evolving threat environment". This illustrates the Consortium's commitment to security of sensor systems. The SSC has long held that we cannot guarantee security of a sensor system, but rather that we would first and foremost ensure that elements of the technical standard would not prevent security accreditation and second that the technical standard provides necessary building blocks to system designers so that they can secure their system according to unique program needs. This is important as we cannot guarantee that following the SOSA Technical Standard will result in an authorization to operate (ATO).

This raises several interesting and challenging questions with respect to developing a technical standard with security built in rather than bolted on. How do we know we have provided the right building blocks for a broad number of use cases and missions? How can we be sure that elements of the standard do not lead to cases where securability is impossible? How do we define security properties of modules in an implementation-agnostic way to protect vendor intellectual property? How do we define security properties of a Modular Open Systems Approach (MOSA) when the composition of modules is not known *a priori*?

The qualitative answer to these questions is by ensuring broad participation in the subcommittee by industry and government. Hopefully this broad participation with diverse experiences and expertise helps us arrive at good conclusions that we codify in the technical standard. Is there a more quantitative approach? We believe there is, through the use of formal methods. Formal methods refer to mathematically rigorous techniques applied to various stages of system development, including specification, development, and verification. Via mathematical rigor, formal methods allows us to prove correctness, safety and security properties of a system. Formal methods will not guarantee achieving an ATO but can provide strong assurances about system properties.

Based on our own prior experience in MOSA, we are building formal analysis tools to model security of interaction-centric open systems. In this paper we describe how open standards can support formal security modeling through the development of artifacts that support the standard (e.g., interface description language files) and through the use of a common syntax for requirements (e.g., Easy Approach to Requirements Syntax or EARS). We also describe how formal methods support securing open standards through modeling of adversarial effects and providing a model of what it means to be secure. We report on a case study we conducted with SOSA security in mind where we apply the UC formal analysis framework to a multi-level secure message passing server. We wrap up by describing our next steps to continue building out a framework for using formal methods to help develop security of open standards.

Open Standards Supporting Formal Security Modeling

Open Standards are great for seeing how something is developing and how it's properties can guarantee certain features that a program or project may want. All the important factors of the system outline what it should do and what it should not do. This sets a standard or an ideal of what a system certified to the standard will do and will not. With this as the premise we can then create a formal security model of the standard and see what to protect and what still has work left to do. Finally with this security model that matches the ideal of what the system should be we can then take vendor created implementations and see how closely it resembles that ideal model. Modeling is done without exposing the vendor's IP through the use of abstractions while still being able understand the security model of the device on how it deviates from the specifics of the standard's ideal model.

First to cover is the fact of how important requirements formatting is for automatically generating the formal security model quickly. In the SOSA Technical Standard, for example they use the Easy Approach to Requirements Syntax, also known as EARS, to create the requirements for the standard. EARS helps make the requirements easy to read and helps the reader understand what the requirement is asking of them. It also gives the creator a format to fit for creating requirements that are testable. The EARS format, an example shown in Table 1, typically flows as follows:

- Starts with a trigger and or condition that states what caused this to start
- Flows into the subject of the requirement and the action they are to perform or not
- Followed by the object they may be performing the action against.

The interesting thing about the EARS format is that the only 2 required pieces for the format are the subject and action pieces. As not all requirements have a trigger, condition, or object that needs to act against. EARS makes requirements easy to understand for those who are not familiar with the reasons for why a rule is there and helps eliminate the ambiguity that can occur for a rule that is using unformatted natural language. Next we will discuss how standard artifacts can improve the accuracy and speed at which we make these formal security models.

Table 1: Breakdown of SOSA Rule into the EARS Format

EARS Breakdown of a SOSA Rule			
"When requested, the SOSA Security Services module shall authenticate a digest for a software package identifier."			
Trigger	Actor	Action	Object
When requested	SOSA Security Services module	authenticate	digest for a software package

When talking about standard artifacts, these are the pieces of information about the requirements that we generate throughout the process that help give more insight to what the rule specifically requires for a module. They give context and more understanding to how we should implement these rules, for example we have the interface description language (IDL) files an example is shown in Listing 1. These give insight into how the interfaces look, for example what to expect to come into the system and what is output. These let us understand from a module level all the vulnerabilities that may be possible through these interactions from a

formal security model perspective. This lets us abstract away the actual implementation details those vendors, who want to conform to the standard, have in their conformant or aligned products to check for security guarantees while not giving away any of their IP to do this formal analysis of their system. Formal analysis supports the test suites which in them have the exact details of what a vendor will need to pass to meet conformance to the standard. This test suite gives exact insight into what in the system is under test and what interfaces are for the standards body and what are extras that may or may not have different security properties from what just a direct implementation of the standard may have. These are just some of the ways that open standards allow us to quickly generate formal security models to prove the security guarantees of the open standard. Next we discuss how formal methods can support securing open standards.

Listing 1: Sample IDL File of the GetKey_Request interaction

```
//! Source file: FACE/DM/DataModel/GetKey_Request.idl

#ifndef _FACE_DM_DATAMODEL_GETKEY_REQUEST

#define _FACE_DM_DATAMODEL_GETKEY_REQUEST

#include <FACE/DM/DataModel/UniqueIdentifier_Type.idl>

module FACE {

   module DM {

      module DataModel {

         module T_GetKey_Request {

            typedef sequence<UniqueIdentifier_Type>
Seq_UniqueIdentifier_Type;

            struct GetKey_Request {

               Seq_UniqueIdentifier_Type keyID;

            };

         };

         typedef T_GetKey_Request::GetKey_Request GetKey_Request;

      };

   };

};

#endif // _FACE_DM_DATAMODEL_GETKEY_REQUEST
```

Formal Methods Supporting Securing Open Standards

One of the primary benefits of formal methods is the rigor that comes with applying formal techniques to a problem. Unfortunately, of the open architecture standards we are aware of, few develop rigorous security models. The E-safety Vehicle Intrusion Protected Applications (EVITA) standard is one exception. EVITA has specified both a metamodel for security and an access control model in UML, followed by a set of formally specified security property predicates. This allows system developers to verify that their implementations meet the security requirements of the standard. While the SOSA SSC has not developed formal security models to date, there are many potential benefits to doing so.

Security of a system is not a binary property, rather it is a function of risk. A good formal approach to security must be able to capture this fact. A challenge in applying formal methods to a technical standard is that while the standard specifies interfaces and enables composability based on those interfaces, we do not necessarily know in advance how modules and interfaces will be composed to create a sensor system. Thus, to formally reason about the security design of a standard, we must be able to reason about compositions of modules, and potentially even restrict certain compositions. Furthermore, additional complications arise since implementations of the standard (or portions thereof) do not necessarily exist. This is why security models for standards (like that of EVITA) focus on formally specified security properties, which implementers can verify once their implementation exists (actually, we would recommend that verification begins during the design stages of an implementation against a standard). A third challenge is that threats (and therefore risk) changes over time. As attacks only get better over time, a particular threat that was too unlikely to necessitate mitigation at one point in time can rapidly become a high-impact threat. Thus, we must update security models periodically.

We are currently developing a new approach to applying formal methods to secure open standards. Rather than specifying security properties, which often turn out to be binary and are less amenable to risk management, we are interested in the effects of an adversary on a system and quantifying the degradation of security in the face of the effects, especially as effects compound across system layers. Rather than focusing on proving absence of known, common implementation flaws, we use an interaction-centric modeling paradigm. This does not require any *a priori* structure to system composition, but rather can rigorously prove security in the presence of specified adversarial effects, and then study the impact of arbitrary composition on security. As described in the previous section, the technical standard itself and its artifacts help generate the models used for analyzing security. The methods we are developing are nascent in system security analysis, but in fact, cryptographers have used the methods to prove security of cryptographic protocols for over two decades. The next section describes a recent case study we conducted with applications to multi-level security in SOSA.

An Approach for Making It Happen

To make formal security analysis of open standards more accessible, we are developing case studies and tools to apply formal methods for security on SOSA that will eventually extend to other Open Architecture standards. We have developed a case study of a multi-level secure (MLS) message passing server (MPS) in the UC framework. Security proofs in the UC framework have two key components, an ideal world specification and a real world specification. The ideal world specification can be viewed as a perfect implementation of a design (e.g., an open standard). This ideal specification is allowed to make simplifying assumptions to make the specification as simple as possible. The goal is to make it so simple that security analysis of the ideal world specification is trivial. This can be done, for example, by leveraging a completely trusted, incorruptible entity in the specification. The real world specification cannot make simplifying assumptions, and therefore, security analysis of a real world specification is much more difficult. This would include, for example, a vendor's specific implementation of a specific design, protocol, or system component (e.g., the same open standard). The ideal and real world specifications have the same application programming interface (API) or direct interfaces. To use the UC framework to prove security properties of a real world specification, we also specify adversarial interfaces, which are the interfaces that an adversary exercises to achieve its goals (whatever those goals may be, a UC proof accounts for adversarial goals generically, not specifically). The remainder of this section presents an example use case of the UC framework applied to a multi-level secure message passing server. In this example, there are two security domains (Low and High). On either domain, the system takes an input and returns a response. For this example, we let the input be a key/value pair, which is stored by the system. If the value is blank, however, then the supplied key is used to retrieve a previously stored value. The content below can be thought of more as a proof sketch than a full proof, as presentation of the entire proof is beyond the scope of this paper. The elements of the proof are shown in Figure 1.

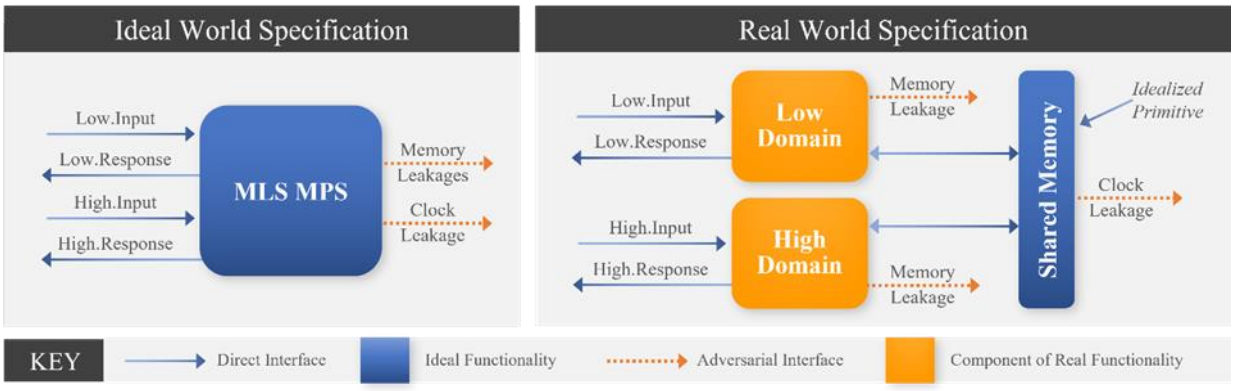

Figure 1: The full diagram of the Universal Composability framework for the Multi-Level Secure (MLS) Message Passing Server (MPS)

Figure 1 shows the two key components of a UC proof, including the direct interfaces and adversarial interfaces.

In the ideal world specification for the MLS MPS, we have modeled the high and low domain with a single functionality and implement this functionality using no shared resources. This ensures that the two domains are completely isolated which ensures there is no leakage between domains. For the adversarial interfaces of

the ideal world specification, we ensure no useful information is revealed to an adversary by returning random values for memory leakages and random values for clock leakages.

In the real world specification for the MLS MPS, we split the high and low domains into their own functionalities. This would mimic using separate processes for each domain and potentially pinning each process to a separate core of a modern CPU. For the real world specification, however, we have chosen to implement memory as a shared resource that each domain can access. This mimics a shared memory system common in many computing architectures. To isolate the two domains, we have each domain independently generate a cryptographic key on startup. Before writing to the shared memory resource, each domain encrypts all information it writes to shared memory and tags that memory according to the domain that wrote it. When retrieving information from shared memory, each domain process would retrieve the contents and check the tag. If the tag is correct, it would decrypt the memory and process the information accordingly. The shared memory and memory encryption illustrates how real world specifications in UC are much more complex than their ideal world counterpart. For the adversarial interfaces of the real world specification, memory leakages to the adversary are encrypted values, and therefore appear random without knowledge of the key. Clock leakages are also random in this world due to process context switching and other randomness that occurs in typical computing architectures. If we wanted to study cache timing side-channel vulnerabilities and implemented our shared memory system using a cache, that could additionally allow us to study how an attacker could combine information to attempt to retrieve sensitive information from the system.

Developing UC security proofs has historically been done using pen-and-paper, which can lead to mistakes. We are currently collaborating with researchers at Boston University who are developing a domain specific language (DSL) for specifying UC security proofs. We have prototyped our MLS MPS proof in their DSL and are currently working with them to verify the proof. Once complete, we plan to release our proofs so they can serve as an example to future users who would like to use the UC framework to perform security analyses.

Where We Go Next

Formal approaches to cybersecurity have seen growing adoption in recent years. This adoption has focused primarily on implementations contained in code bases of varying sizes. For open standards like the SOSA Technical Standard, this approach will work for vendors who implement the standard but has limited impact on standard developers. What we have presented in this paper is a different approach that we think can change this significantly. Our approach comes from the cryptographic protocol community, where the more important aspects of the formal models are the interfaces and the behaviors. This aligns better with open standards like the SOSA Technical Standard. As we further flesh out our approach and the supporting tools and use cases, we will identify opportunities to use formal analysis to assist with standards development.

Adversarial modeling in open standards is a topic area that practitioners have not explored in enough depth in our opinion. The cryptographic community has developed a number of adversary models of ranging capabilities that have served that community well. These include the honest-but-curious model, where adversaries follow protocols as specified but glean what information they can during protocol execution to attempt to violate security properties, the malicious model, where adversaries deviate from protocols in any way possible to attempt to violate security properties, and a middle-ground model known as the covert model, where adversaries deviate from protocols only if their likelihood of getting caught doing so is minimal. Such a capability focused approach lends itself nicely to our approach to formal methods applied to systems. Developing the right metamodel or taxonomy for system-level adversary modeling is of great interest to us moving forward.

Referenced Documents

(Please note that the links below are good at the time of writing but cannot be guaranteed for the future.)

- Group, September 2021; refer to: www.opengroup.org/library/c212
- Mavin, Alistair, et al. "Easy approach to requirements syntax (EARS)." 2009 *17th IEEE International Requirements Engineering Conference*. IEEE, 2009
- "What is Formal Methods?", published by National Aeronautics and Space Administration (NASA), accessed May 2022; refer to: https://shemesh.larc.nasa.gov/fm/fm-what.html
- "Security and trust model", published by E-safety Vehicle Intrusion Protected Applications, accessed May 2022; refer to: https://www.evita-project.org/index.html
- Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. 2014. "Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors." ACM SIGARCH Computer Architecture News 42, 3 (2014), 361–372
- Thomas Dullien. 2017. "Weird machines, exploitability, and provable unexploitability." IEEE Transactions on Emerging Topics in Computing 8, 2 (2017), 391–403
- Canetti, Ran. "Universally composable security." Journal of the ACM (JACM) 67.5 (2020): 1-94
- "EasyUC", accessed July 2022; refer to: https://github.com/easyuc/EasyUC

About the Authors

Dr. Michael R. Clark is an Associate Director of Secure and Resilient Systems at Riverside Research. He has been involved with the SOSA Consortium since 2018 and currently serves as the lead of the Security Subcommittee. He led the development of the inter-module interaction security (or data-in-transit) portion of the SOSA Technical Standard, and has helped with other portions including authentication, authorization, key management, and secure system startup.

Mr. Anthony Pierce is a Cybersecurity Researcher of Secure and Resilient System at Riverside Research. He has been involved with the SOSA Consortium since 2019 and assists in the creation of security requirements as part of the Security Subcommittee. He is leading the development of authentication for Security Services in the SOSA Technical Standard.

About The Open Group FACE™ Consortium

The Open Group Future Airborne Capability Environment™ Consortium (the FACE™ Consortium), was formed as a government and industry partnership to define an open avionics environment for all military airborne platform types. Today, it is an aviation-focused professional group made up of industry suppliers, customers, academia, and users. The FACE Consortium provides a vendor-neutral forum for industry and government to work together to develop and consolidate the open standards, best practices, guidance documents, and business strategy necessary for acquisition of affordable software systems that promote innovation and rapid integration of portable capabilities across global defense programs.

Further information on the FACE Consortium is available at www.opengroup.org/face.

About The Open Group SOSA™ Consortium

The Open Group SOSA™ Consortium enables government and industry to collaboratively develop open standards and best practices to enable, enhance, and accelerate the deployment of affordable, capable, interoperable sensor systems. The SOSA Consortium is creating open system reference architectures applicable to military and commercial sensor systems and a business model that balances stakeholder interests. The architectures employ modular design and use widely supported, consensus-based, nonproprietary standards for key interfaces.

Further information on the SOSA Consortium is available at www.opengroup.org/sosa.

About The Open Group

The Open Group is a global consortium that enables the achievement of business objectives through technology standards. With more than 870 member organizations, we have a diverse membership that spans all sectors of the technology community – customers, systems and solutions suppliers, tool vendors, integrators and consultants, as well as academics and researchers.

The mission of The Open Group is to drive the creation of Boundaryless Information Flow™ achieved by:

- Working with customers to capture, understand, and address current and emerging requirements, establish policies, and share best practices
- Working with suppliers, consortia, and standards bodies to develop consensus and facilitate interoperability, to evolve and integrate specifications and open source technologies
- Offering a comprehensive set of services to enhance the operational efficiency of consortia
- Developing and operating the industry's premier certification service and encouraging procurement of certified products

Further information on The Open Group is available at www.opengroup.org.