Abstract—Building complex secure applications with high assurance is difficult and requires experts. Security patterns and best practices have been proposed to assist architects in designing secure applications. However, these are usually written independently of the specific details of underlying platforms. This leads to a gap between patterns and the platforms, and does not directly support the design-level analysis and verification of systems to be built on those platforms. In this paper, we propose an approach to incrementally build an application design using design fragments, which are specializations of patterns for target platforms. Design fragments can be composed and reused during design, and directly support design-level security analyses. We also discuss our plan to evaluate our approach with a case study of the design and analysis of a smart electricity meter, using security requirements taken from industrial guidelines.

Keywords—Security Patterns; Capability; Composition; Verification; Platform;

I. INTRODUCTION

Building a secure application and providing assurance about its security properties requires significant effort and expertise. Secure platforms are trusted computing bases, and are a foundation for building large security-critical applications [1, 2]. Platforms encompass a wide range of artifacts, including hardware architectures, operating systems and libraries. In this paper, we use the term platform to refer to the operating system and its associated hardware architecture. The design of secure applications can be informed by security patterns [3], tactics [4] and best practices. However, these patterns and practices are usually informally described, have vague claims [3, 5] and abstract from the details of specific platforms. A study [6] to assess the applicability of security patterns reported that existing security patterns do not provide guidance on how to implement the patterns correctly. Furthermore, they do not directly support the design-level analysis and assurance of systems to be built on specific platforms.

Providing assurance about the security properties of a complex application is usually very costly. Formal verification provides the highest levels of assurance about the security of an application. However, formally verifying the complete functionality and security of the source code of a large system is infeasible. The current state-of-the-art limit is around 10,000 lines of code [1]. Instead, this paper aims for verification of system architectures and relies on trustworthy secure platforms to provide valid abstractions for design-level analysis.

Our goal is to bridge the gap between informal security patterns and the underlying platform, by providing platform-specific design fragments for security patterns. These design fragments are specializations of security patterns for a target platform, in this case a platform that adheres to the capability security model [7]. A capability is a communicable, difficult-to-forge token of authority that references an object with associated access rights. Informal security patterns can serve as inspiration for design. However, it is unclear what security properties they claim and what should be verified. Moreover, patterns usually have implicit assumptions. We try to make these assumptions explicit and formalize the patterns to allow formal verification of security properties. The design fragments are associated with reusable design-level verification procedures to substantiate the informal security-related claims and assumptions about the properties provided by the platform. Together these can be used to design and verify a variety of applications.

We model designs using the Serscs Access Modeller (SAM) notation [8]. We choose SAM as it is a modeling notation that is targeted for designing and verifying models for capability-based systems. We intend to leverage our group’s prior work, seL4 [1] – a formally verified capability-based operating system microkernel – as our target platform for implementation. Note that no formal correspondence has actually been established between a SAM-like model and seL4, so the validity of that becomes part of our future work.

We aim to provide reusable procedures to verify security properties for capability-specific design. The design fragment itself is verified this way, and ultimately the whole system is re-verified with a larger verification procedure, which incorporates the design fragment procedure. The specific approach for design-level verification depends on the security property of interest [9].

Composing verified capability-specific design fragments should help to achieve a more secure application design. Furthermore, reusing the verification procedures associated with these design fragments should reduce the cost and effort of verifying the whole application design. An approach to compose design fragments and their properties is essential to building high assurance secure applications.

We discuss related work in Section II and our research questions in Section III before describing our proposed approach in Section IV. We then present our expected
contributions and validation strategy before discussing the progress made towards realizing our vision.

II. RELATED WORK

Existing security patterns can be used to guide design but their description and security property claims are highly informal [3, 5]. Heyman et al. [10] presented an approach to formally verify security requirements for security patterns using Alloy. This includes “expectations”, which describe the expected behavior of the component, and “residual goals” which describe security concerns that need to be addressed but are not in the scope of the pattern. Other prior work also advocates the use of formal methods to verify security properties of patterns [5]. Users may define the behavior of a pattern in UML, which is transformed to Promela for verification with SPIN [11] model checker. However, this approach does not consider the security models, mechanisms and properties provided by the underlying security platform.

Some other efforts [12, 13] in the model-driven community focus on developing platform-specific security profiles, which are used in application design to facilitate security property analysis and to map models to platform-specific code. We focus on providing reusable capability-specific design fragments for patterns that can be composed during application design. Furthermore, we support design-level analysis and assurance of systems to be built on specific platforms. These, to the best of our knowledge, are not major concerns in these previous efforts.

The link between architecture tactics and code [14] or design patterns [15] has been explored through code mining. One observation is that there is wide variation in how tactics are implemented in different situations [15]. In our case study we choose one instantiation, for a capability-based platform.

Other research also advocates the importance of the underlying platform to fundamentally improve security [16]. However, they propose new hardware mechanisms to support security, whereas we rely on general purpose hardware and a formally verified microkernel.

Previous efforts in the design pattern community have worked on composition of design patterns. Yacoub et al. [17] categorize previous composition techniques as behavioral or structural. In behavioral composition, patterns are composed based on the object interactions. The structural composition composes patterns by modeling their structures as class diagrams. Yacoub et al. [17] also point out that composition considering both behavior and structure will be complex. These efforts focus on design patterns and correctness, in terms of implementation, of the design. However, composing security patterns to achieve security properties requires considering both behavioral and structural composition.

III. RESEARCH QUESTIONS

The research questions that we are tackling are as follows:

- How to verify design fragments against intended security properties?
- How can an application design and its verification benefit from composition of reusable verified design fragments?

IV. APPROACH

Our approach is shown in Fig. 1. We have assembled security patterns from the literature into a catalog, containing 130 security patterns. We have analyzed several patterns and instantiated them as concrete capability-specific design fragments, expressed using the SAM notation, and verified the properties of these patterns at the design-level using SAM’s Points-To analysis using Binary Decision Diagrams (BDD) [8, 20]. In the design of secure systems, the verified design fragments are composed to build an application design that satisfies the system security requirements. This composition aims to harden the security of the application against specific threats by mitigating those threats. Finally, we verify the composite system at the design level, to provide assurance that the system supports the intended properties.

A. Capability-specific design fragments for security patterns

We select security patterns from a catalog and then develop instances of design fragments from these patterns. The existence of an underlying platform makes this instantiation possible. We identify and explicitly state the assumptions of the patterns. These assumptions can be discharged by the mechanisms supported by the platform or carried through as operational constraints. We manually extract the important information from the textual form of the pattern and specialize it as a capability-specific design fragment. We use SAM to model our realization of patterns as capability-specific design fragments. These fragments make up our collection of reusable capability-specific design fragments. The security property analysis will provide feedback to improve the design instantiation of the patterns, e.g. to address counter-examples or security violations identified during the analysis process.

A SAM model represents the distribution of capabilities across components that make up a capability-based system. Such a model consists of a specification of all the components, a diagram, representing the capability distribution in SAM notation, generated from the specification and datalog statements that specify security goals of the system. The datalog statements are made up of rules and datalog statements that are used to verify the system. We refer to these statements as the design fragment’s verification procedures.

B. Composition of capability-specific design fragments

Designing a secure system may require the composition of several security patterns, as each pattern is concerned with different security properties. The first step in composing patterns is selecting the appropriate patterns; those that support the security requirements of the application or those that can mitigate attacks. Patterns can be composed with each other or applied to an existing model. There are two challenges in composing security patterns: producing a design that provides the intended security properties (and does not break the current
security properties) and ensuring that the patterns do not conflict. This suggests that security patterns be applied and composed incrementally.

There are a number of things we reuse when composing design fragments. First, we reuse the structure and behavior of the design fragment. In particular, we reuse the specification for each of components in the design fragment. Second, we reuse the verification procedure that we applied to the design fragment and apply it during the composite design verification.

![Diagram](image)

Fig. 1. Overview of our approach

In order to reason about whether the composed design provides the required security properties, we reuse our verified platform-specific design fragments and conduct a system-level analysis that the composed system supports the required security properties. The security property analysis can help to improve the design by identifying weaknesses and security violations, and can also provide assurance about the security of the design.

When composing security patterns to harden the security of an application, we must ensure that the resulting system not only mitigates the new attacks but also maintains the security properties that the previous version had achieved.

C. Security Property Analysis

We analyze security properties for individual design fragments, and also for the whole application architecture. Analyzing individual design fragments gives assurance about the security properties of the fragments. These fragments are composed together to form the application architecture design. Analyzing the whole application architecture provides evidence that the application retains the intended security properties despite transformations introduced in design or composition. This step is closely related to the capability-specific design fragments for the patterns, as the analysis performed on these designs will not only provide pointers to improve the designs but also provide evidence about the security property the fragments provide.

Different security properties may require different verification procedures. As an example, the absence of dataflow between two components might be verified by model checking, but the absence of information flow may require formal verification, in the form of formal proofs, at the code level. Formal verification techniques may also provide a higher level of assurance about the security properties of our designs. On the other hand, while analysis techniques such as scenario-based evaluation may be weaker, they will provide faster and less expensive forms of evidence.

We reuse the verification procedures for our capability-specific design fragments in order to verify the entire application architecture model. Full formal verification of a code-based can be very expensive or infeasible. However, design-level verification of some security properties for large systems is feasible and can be achieved through the use of verifiable designs for specific security patterns.

Points-To analysis using Binary Decision Diagrams (BDD) [20] is targeted to verify large scale application design. We use this technique to verify both the capability-specific design fragments and the application design resulting from the composition of design fragments. We define the security goals in datalog rules and use SAM’s Points-To analysis using BDD engine to analyze the security property. An example datalog rule is hasRef(source, target) which checks whether there is an access from source to target.

V. Expected Contributions

The expected contributions of this research are as follows:

- The concept of a design fragment as an instantiation of a security pattern for a specific platform, allowing design-level verification.
- An approach to incrementally build an application design using verified security patterns, and to concurrently build an associated assurance case to structure claims and supporting arguments about security properties.
- Example design fragments targeting a capability-based platform for several security patterns, and two composition tactics for design fragments.

VI. Evaluation Plan

There are four key parts to our evaluation. Firstly, we evaluate the capability-specific design fragments for patterns through the verification that will be done for each individual design fragments. This will evaluate whether the design fragments provide the security properties guaranteed. Secondly, the verification of the design fragments will be evaluated through simple case studies based on two criteria,
namely Feasibility and Correctness of produced design. Correctness of produced design evaluates whether the design achieved the security property claimed by the verified design fragments. Thirdly, the composition of verified design fragments will be evaluated by performing a security property analysis (i.e. verification) on the produced application design. Finally, the verification of the composition will be evaluated through a real-world case study, Smart Meter in the Advanced Metering Infrastructure, to be validated based on three criteria. These criteria are Feasibility, Correctness of produced design and Usability of the pattern composition approach.

VII. PROGRESS

We have started building a collection of capability-specific design fragments for security patterns. These will be modeled manually using SAM. Here we illustrate the Security Logger pattern [19] as a capability-specific design. One of the aims of the Secure Logger pattern is to decouple logging functionality from the application. The data must be cryptographically secure, with only authorized users able to view the contents of the log file. User sends a log command to secureLogger with the data. Upon receiving the command, secureLogger encrypts the data and sends it with the log command to logManager. logManager will then ask for a new instance of logger, the actual component that logs the data, from logFactory.

![Fig. 2. Secure logger design fragment in SAM](image)

From Fig. 2, logManager has a capability to logFactory and the newly created logger. The user does not have direct access to the logger instance and the file. We set an assertion in SAM that the user does not have any reference to the file. With that, we assume that the user has no access to the file, unless the user has been authorized to access it. The verification procedure for this design fragments is hasRef(<user>, <file>). This datalog rule checks whether there is an access from user to file.

ACKNOWLEDGMENT

NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

REFERENCES