Optimising Architectures for Performance, Cost, and Security

Rajitha Yasaweerasinghelage[1,2](✉), Mark Staples[1,2], Hye-Young Paik[1,2], and Ingo Weber[1,2]

[1] Data61, CSIRO, Level 5, 13 Garden Street, Eveleigh, NSW 2015, Australia
{rajitha.yasaweerasinghelage,mark.staples, hye-young.paik,ingo.weber}@data61.csiro.au

[2] School of Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia

Abstract. Deciding on the optimal architecture of a software system is difficult, as the number of design alternatives and component interactions can be overwhelmingly large. Adding security considerations can make architecture evaluation even more challenging. Existing model-based approaches for architecture optimisation usually focus on performance and cost constraints. This paper proposes a model-based architecture optimisation approach that advances the state-of-the-art by adding security constraints. The proposed approach is implemented in a prototype tool, by extending Palladio Component Model (PCM) and PerOpteryx. Through a laboratory-based evaluation study of a multi-party confidential data analytics system, we show how our tool discovers secure architectural design options on the Pareto frontier of cost and performance.

Keywords: Software architecture · Software performance · Data security · Architecture optimisation

1 Introduction

Many software systems today are complex, with thousands of deployed components and many stakeholders [19]. With increasing complexity, there is increasing development cost. Non-functional requirements for systems often include response time, cost of development and operation, and security. When developing systems, software architecture should support these requirements effectively.

There are inter-dependencies and trade-offs between quality attributes like performance, cost, and security. For example, secure components are generally more costly than non-secure components. Prior work reports costs of \$10,000 per line of code to develop highly-secure components, compared to \$30–\$40 per line of code for less-secure components [7,11]. When designing systems with critical requirements for performance, cost, and security, architects try to achieve optimal trade-offs between them. In a large design space, with many components and design options, finding designs with good trade-offs is challenging, even for experienced architects. Manually assessing and comparing quality attributes for even a small number of design alternatives is difficult and error-prone.

Model-based design is now a common practice, and helps architects explore options during design. Many architecture modelling and optimisation methods have been studied [2–4]. There are well-established methods for optimising deployment architecture based on the performance of the system [13,16], costs of development, deployment, and maintenance [16], and other constraints such as energy consumption [21]. However, security constraints and policies are not yet well-treated in existing literature on architectural optimisation [1].

In this paper, we propose a new approach for optimising for performance, cost, and security in architectural design. We demonstrate the feasibility of the approach by implementing a prototype which extends the Palladio Component Model [4] and PerOpteryx optimisation tool [13] to support static *taint* analysis. One challenge in designing secure systems is defining and evaluating system security. Optimisation techniques require automated assessments. Static taint analysis is a simple automatic security analysis approach. Taint analysis is not a perfect model of security, but is a popular technique for identification of architecture-level vulnerabilities related to data propagation in the design phase [22]. Although our prototype uses taint analysis, our approach is more general and we discuss the use of other techniques for security analysis.

The main contributions of this paper are: an approach for architectural optimisation for cost, performance, and security; a model and method for taint analysis for security analysis for Palladio and PerOpteryx; and an evaluation of the approach on an industrial use case demonstrating feasibility and the ability to generate useful insights: in the case study, best performance and cost were achieved by non-secure architectures, but secure architectures were not far behind. Also, the approach discovered distinctive design options on the Pareto frontier of cost and performance for secure designs.

The paper structured is as follows. In Sect. 2, we introduce existing technologies relevant to the proposed approach. Then we provide an overview of the proposed method in Sect. 3. Section 4 provides details about modelling and optimisation through a running example. We discuss and compare literature closely related to this work in Sect. 5, propose suggestions for future work in Sect. 6 and conclude the paper with Sect. 7.

2 Background

This section reviews: architecture performance modelling; architecture design space exploration and deployment optimisation; and static taint analysis.

2.1 Architecture Performance Modelling

Architectural models capture the system structure by representing the links between components. Performance characteristics are associated with these components and their composition. Popular frameworks for architectural modelling are the Palladio Component Model (PCM) [18], and Descartes Modelling Language [12]. Architectural models can incorporate additional non-functional attributes associated with the system structure, such as latency, resource usage, cost and throughput. The resulting models can be used by simulation engines or analytical solvers to analyse non-functional properties [5]. Simulation-based prediction can be time-consuming, but provides more flexibility for modelling.

Palladio Component Model (PCM). [18] is the platform used in this paper to model architecture performance characteristics. Palladio was selected as it is freely available, supports simulation, provides a familiar 'UML-like' interface for model creation, and has the flexibility to incorporate extensions such as architectural optimisation tools [8,13], new qualities [21], and new kinds of systems [23]. The modelling concepts in Palladio align with component-based development paradigm and support component reuse across models.

2.2 Architecture Design Space Exploration and Deployment Architecture Optimisation

Automated software architecture exploration based on architecture models is increasingly popular in industry. Aleti et al. [1] surveys existing methods.

PerOpteryx. [13] is an automated design space exploration tool for PCM, capable of exploring many degrees of freedom. PerOpteryx starts with a PCM instance and a set of design decision models that describe how the architecture can be changed. Automated search over this design space is performed using a genetic algorithm. For each generation in the search, a Palladio instance is generated and analysed to evaluate quality attributes such as performance and cost.

PerOpteryx is capable of optimising multiple quality attributes by searching for Pareto-optimal candidates. A candidate is Pareto optimal if there exists no other candidate that is better across all quality metrics. A set of Pareto-optimal candidates approximate the set of globally Pareto-optimal candidates [8].

2.3 Static Taint Analysis

Defining meaningful quantitative metrics for security is challenging. There have been a number of approaches proposed, but in our opinion, there is no single generic method suitable for all applications (see Sect. 5). In this paper, to simplify our demonstration of security analytics for optimisation, we use taint analysis. Taint analysis results in a binary secure/not-secure evaluation for a system, which is arguably the most challenging kind of metric for use in optimisation. Taint analysis is simple but useful in identifying fundamental issues in the data flow of the system, as a form of information flow analysis [17,22].

Taint is used to represent the reach of an attack within a system. As shown in Fig. 1, taint starts at a taint source (node 1), which could be a component exposed to the external environment, then flows to connected components. Taint

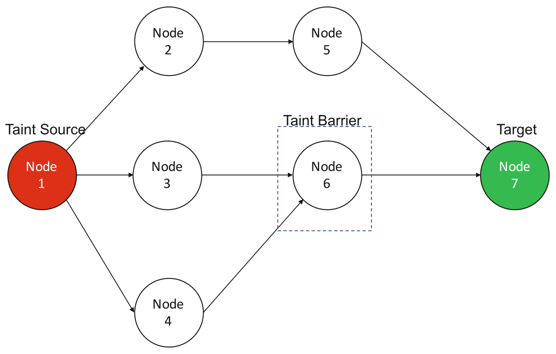

Fig. 1. Graph taint analysis, illustrating an insecure system. Bad 'taint' from the source Node 1 to the critical target Node 7, via a path through Node 2, despite being blocked from flowing through the taint barrier at Node 6.

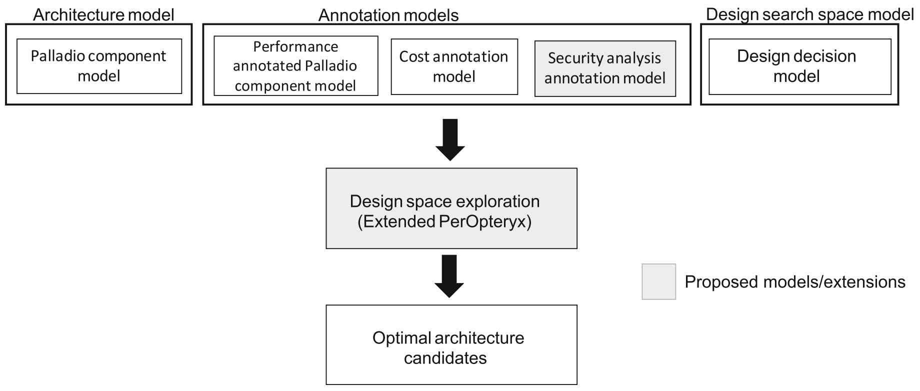
Architecture model
Palladio component model
Annotation models
Performance annotated Palladio component model
Cost annotation model
Security analysis annotation model
Design search space model
Design decision model
Design space exploration (Extended PerOpteryx)
Optimal architecture candidates
Proposed models/extensions

Fig. 2. Method overview, highlighting extensions proposed in this paper.

blockers (e.g. node 6) are secure components which prevent further propagation of taint. A system security property defines the set of critical components (e.g. node 7) which must remain free of taint after maximal propagation of taint through the system. The system in Fig. 1 is not secure, because taint can flow through non-secure components (e.g. nodes 2, 5) to the critical component.

3 Method Overview

Our approach, shown in Fig. 2, combines architecture-level performance modelling, simulation and optimisation. We use three types of models to represent the system: the initial architecture model, annotation models, and the design search space model. We use the Palladio Component Model (PCM) tool for the underlying architecture model. To define annotation models, we annotate PCM with information about three quality attributes; performance, cost, and security.

The performance annotation model is supported directly in PCM, and the Palladio cost extension is used for cost annotations. The security model is defined separately. In Sect. 3.1, we describe how each quality attribute is modelled.

For the design search space model, we used Palladio Design Decision Diagrams. These are used to generate candidate architectures in the optimisation phase. Some design options are specific to security architecture. For example, a component might be modelled as being a secure component that works as a taint barrier. So, the default Palladio Design Decision Diagrams need to be extended to accommodate these model elements.

For design space exploration, we use PerOpteryx optimisation tool with modifications to use these extended security annotation models. The output is a set of generated Pareto-optimal candidate architectures, which can be used by experts to select the final design.

3.1 Quality Attribute Modelling for the Optimisation

The first step of the proposed approach is to model each quality attribute.

Performance Modelling. We used PCM performance analysis, as discussed in the literature [9], which has been shown to be sufficiently accurate for various types of applications, including the example system discussed in this paper. This demonstrates that our approach allows the reuse of previously-developed Palladio performance models.

The security level of a component may affect the resource utilisation of the component, impacting the overall performance of the system. (For example, encrypting communications may incur a performance overhead.) In such cases, a component with one kind of functionality is modelled with different performance (and security) properties as design alternatives, and are used for design exploration during optimisation.

Cost Modelling. We use the existing and well-studied Palladio cost modelling extension for modelling cost attributes. This can capture different types of costs such as component costs, variable and fixed resource costs, and networking costs.

The security level of a component can impact its cost. For example, secure components are more expensive to develop than less-secure components. We model a component with one kind of functionality as multiple alternative components that represent different levels of security each with a corresponding cost in the cost model. Then we use those component alternatives when exploring options during optimisation.

Security Modelling. A key contribution of this paper is integrating security analysis into automatic design space exploration. Unlike other quality attributes such as performance and cost, security is not easily quantifiable. Security analyses often only make Boolean judgements about system security (i.e., secure, or not), but some analyses give continuous metrics of security (e.g., expected time to next attack). In this paper, we demonstrate our approach using taint analysis as the basis for security analysis. However, our general approach could be adapted to use other security analysis metrics, as discussed in Sect. 5.

4 Modelling and Optimising

The prototype for our approach uses taint analysis (see Sect. 2.3) as the security analysis technique. As our goal is to optimise performance and cost while satisfying a security requirement, we developed an extension for integrating taint analysis with existing Palladio Models and incorporating taint properties into the PerOpteryx optimisation. To describe the modelling and optimisation process, we use a running example based on a privacy-preserving computing system called N1Analytics[1] [9]. This section provides details about the extension and how it works for the running example. Finally, we discuss how the architecture of the N1 Analytics system can be optimised for performance, cost, and taint properties.

4.1 Running Example

N1Analytics is a platform that allows statistical analyses using data distributed among multiple providers, while maintaining data confidentiality between the providers. Following the main principles of N1Analytics systems, we designed an initial abstract architecture, to illustrate some of the critical features of our proposed approach. It should be noted that this abstract architecture differs from actual N1Analytics implementations.

Base Deployment Architecture. Figure 3 presents the federated deployment architecture of the N1Analytics platform. Data providers and coordinators are the two main building blocks. In an analytics operation, the coordinators have the private key to decrypt the computed results but do not have access to plain or encrypted input data. They only have a partial output that is not itself capable of revealing plaintext results.

The private key is not accessible to the data providers, so they cannot violate the privacy of the encrypted input data shared with them. Data providers and coordinators may have a set of worker nodes to perform their operations. It is possible to move data between nodes, as long as they preserve the protocol: the coordinator should not have access to the encrypted data, and data providers should not have access to the private keys.

[1] https://www.n1analytics.com.

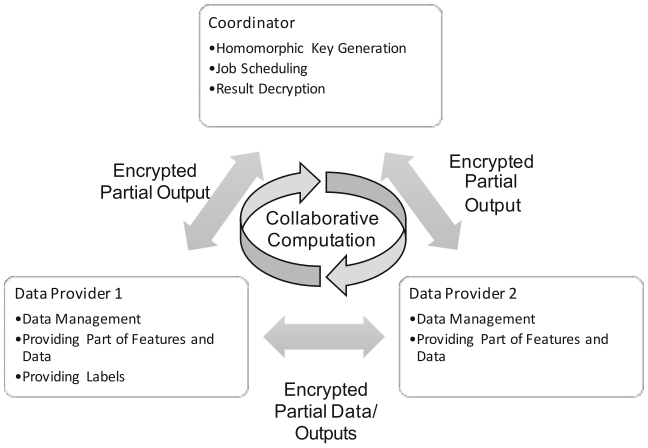

Fig. 3. N1Analytics platform distributed architecture

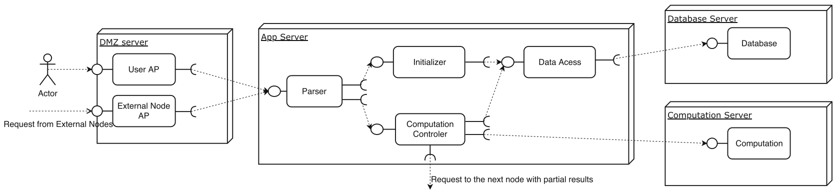

Fig. 4. N1Analytics component architecture in UML notation

Component Architecture. To simplify the demonstration, we modify the architecture of the N1Analytics system used in our earlier work [24] by assuming that the basic component architecture of the coordinator and each data provider is similar. Even so, the resource utilisation and the functionality of each node are different. Notably, the computation overhead and workflow of each node are significantly different. We model each node separately to reflect those differences. Figure 4 presents the architecture model we considered.

4.2 Modelling System for Optimisation

Performance Modelling. We modelled performance characteristics following the general Palladio approach. Our model of the N1Analytics system is similar to that presented in our earlier work [24], but introduces changes to demonstrate cost-based optimization and security-critical components.

In [24], the N1Analytics system was deployed in a test environment, and the resource utilisation of each development component was measured. Then, each development component was mapped to a functional component to be used in the model architecture. The architecture is modelled in PCM using functional components, and the resource utilisation of each component is derived from microbenchmark results. Resource utilisation is defined as a function of workload and the size of the data set. The resource environment definition, usage model, and allocation model were defined based on the design specification of the system. We reuse their published abstract model[2], but with minor modifications to introduce a user access point component, a parser, and database access component for demonstrating data propagation design options.

Cost Modelling. We used the standard Palladio Cost modelling approach. Note that if a single component can have multiple levels of security, it needs to be modelled as multiple alternative components with different cost properties. Similarly, when introducing additional components such as secure load balancers and secure bridging interfaces, base costs and operating costs need to be specified accordingly. There will also be an overhead for operation cost, because some secure components may have higher resource utilisation.

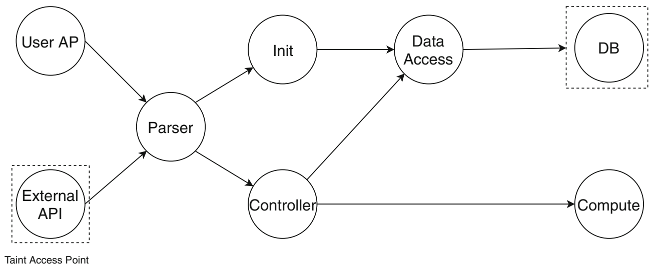

Fig. 5. Taint graph

Security Modelling - Modelling Taint Properties. We extended PCM to define taint properties of the system. These properties are then used in the optimisation algorithm. First, the extension retrieves the candidate system architecture and converts to a taint graph as shown in Fig. 5.

In the proposed method, each software component can be *taint safe* or *taint unsafe*. Assigning this state to a component, based on whether it is secure or not, is a decision for the model designer, as discussed further in Sect. 6. Taint safe components act as a taint barrier preventing taint propagation from that point onwards. In this study, our cost models assume that taint safe components cost more than their taint unsafe counterparts.

From an initial taint setting, we analyse the graph by graph search, spreading taint except through taint safe components. The search includes cyclic dependencies which might spread taint over multiple passes. The results about whether security critical components become tainted are provided to the optimisation engine (see Sect. 4.4).

[2] https://doi.org/10.6084/m9.figshare.5960014.v1.

When modelling the N1Analytics architecture, we represent each component twice, with secure and non-secure alternatives, each with a different cost. Our not-unrealistic assumption is that a secure component is ten times more expensive than its non-secure version. Additionally, to explore the impact of security-specific design patterns, we define two optional secure bridge components in front of the parser and the data access component. Our experiments are executed with and without these secure bridging components.

4.3 Additional Design Options

We modelled additional architectural design alternatives related to data propagation of the system and basic security policies in Design Decision Diagrams. These define the exploration space for architecture optimisation.

In this paper, we include design options directly related to the security properties. The design options model their impact on the overall performance, cost, and security of the analysed architecture. These design options are used alongside other general architecture design options.

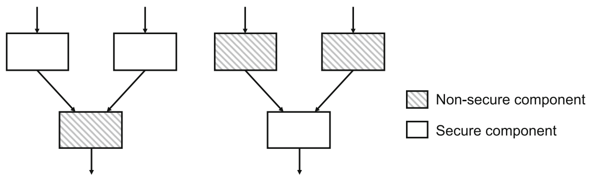

Fig. 6. Design option - taint blockers/secure components

Taint Blockers/ Secure Components. Developing a secure component is significantly more expensive than developing a component using a standard development process. To be cost-optimal, only a limited number of components can be secure.

As illustrated in Fig. 6, a component can be made taint safe to act as a taint barrier protecting critical components and thus ensuring system security. A secure component may have higher resource utilisation compared to less-secure components due to validity checks, or encryption, and this is also reflected in the performance models.

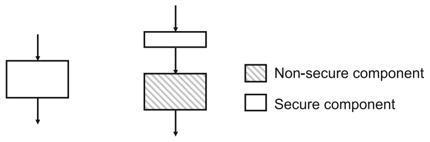

Fig. 7. Design option - secure bridging interfaces

Secure Bridging Interfaces. There is a significant cost of securing components if those components are large. One design strategy to prevent taint propagation is to implement secure bridging interfaces in-between components, as shown in Fig. 7. A typical bridging interface component is small compared to major functional components because it focuses on enforcing key security properties. Being smaller, their development cost can be significantly lower. On the other hand, introducing a bridging interface component adds new fundamental cost for developing the component, increases resource utilisation, and may act as a performance bottleneck.

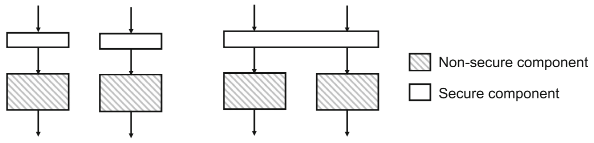

Fig. 8. Design option - secure component access interfaces and secure load balancers

Secure Component Access Interfaces and Secure Load Balancers. Similar to the secure bridging interface components, a design strategy might be to introduce secure common interfaces/load balancers, or to bring existing common interfaces/ load balancers to a higher security level (see Fig. 8). Generally, these components are smaller than major functional components, and so have significantly lower development cost. However, these components also can be bottlenecks to the system and incur additional base development cost. In addition, as load balancer interfaces can be concurrently accessed by multiple components with different resource utilisation, we have to consider such interactions when optimising the system under different workloads.

4.4 Model Optimisation

We started the optimisation with the architecture shown in Fig. 4. Even though the proposed approach can handle multiple components defined as taint starting points or security critical systems, for the simplicity of illustration we define the *external access* component as the taint starting point and the *database* component as the only security-critical component. In the initial architecture, all components are non-secure.

In the Design Decision Model, we allow every component except access points and databases to be made taint safe or taint unsafe. Additionally, we defined optional trusted bridge components before the parser and computation controller. Access points, databases, and computation components should only be allocated to the DMZ server, database server, and computation server respectively. Other components can be allocated to any server.

We modelled the example system using Palladio Workbench version 4.0 using SimuCom Bench for performance analysis with Sensor Framework as the persistence framework. For design space exploration we used PerOpteryx version 4.0 with slight modifications for accommodating taint analysis when optimising. We executed the optimisation on a machine with a 2.7 GHz Intel Core i5 CPU and 8 GB main memory. It took approximately 4 h to run 500 iterations of the simulation.

4.5 Results

The selection of optimal components for a system depends on its requirements. Here we assume the reasonable goal is the *lowest-cost secure architecture that achieves a response time above a given threshold.*

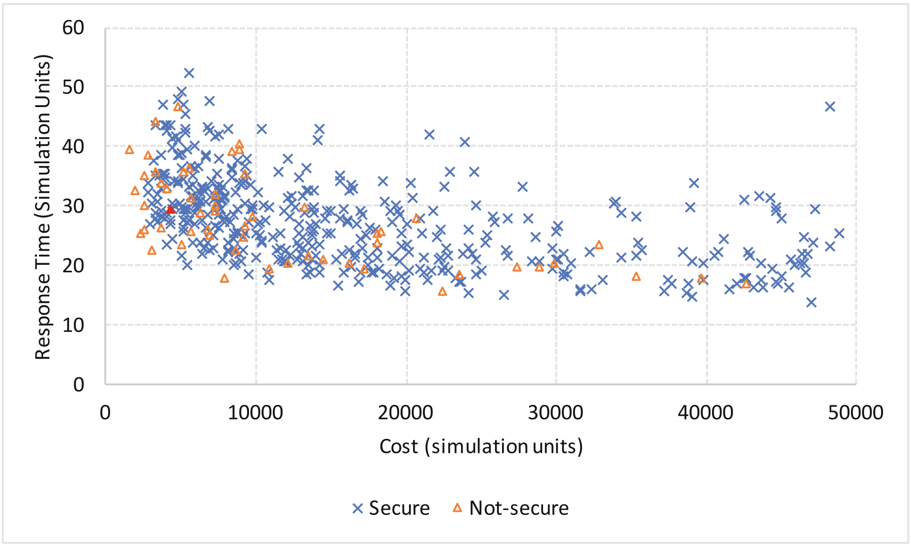

Fig. 9. Response time and cost of candidate architectures generated by PerOpteryx. (Color figure online)

Figure 9 plots the identified candidate architectures as a distribution of response time and cost. The red dot indicates the initial architecture configuration (i.e. Fig. 4) fed into the system. Secure candidates are shown as blue diagonal crosses, and non-secure candidates are shown with an orange plus. As can be seen, the genetic algorithm generated fewer non-secure candidates than secure candidates. Importantly, the results show that when the architecture is secure the system tends to be more expensive and have inferior performance. In other words, *if security is ignored when picking a candidate architecture, one would likely pick a non-secure architecture.* However, there are secure alternatives with just slightly inferior cost and performance.

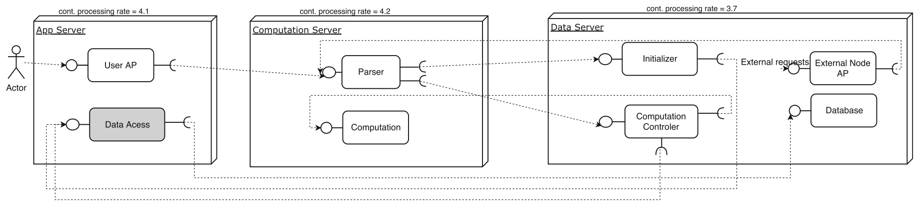

Fig. 10. Secure candidate architecture with low cost where the simulated cost is 2,778 units. Simulated response time of this architecture is 33.9 units.

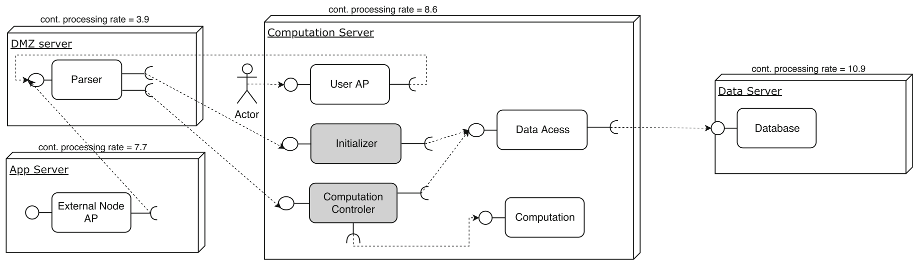

Fig. 11. Secure candidate architecture with low simulated response time of 13.4 units where simulated cost is 46,692 units.

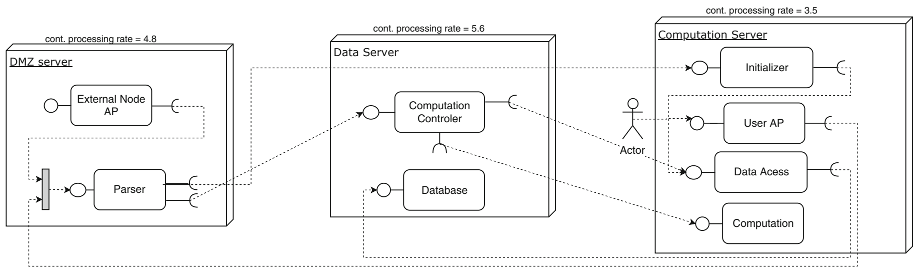

Fig. 12. Secure intermediate point where a bridge component has been introduced. Cost is 2,832 units and response time is 32.1 units.

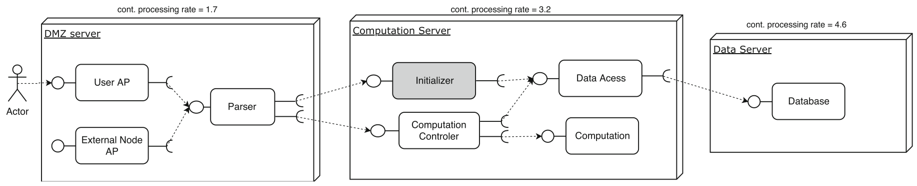

Fig. 13. Generated non-secure architecture. Simulated cost is low as 1,580 and response time is 37.8 units. The system is non-secure despite one component being secure.

For some concrete examples, Fig. 10 shows the cheapest secure architecture that costs 2,778 units but has 33.9 units response time. Figure 11 illustrates the best performing secure architecture identified, which has a response time of 13.4 units but costs of 46,692 units. Figure 13 shows a non-secure architecture which has cost low as 1,580 while response time is 37.8. From these examples, it is evident that this method is capable of generating wide range of feasible candidate architectures based on given design options. This is true for all the candidates.

Identifying vastly different architectures with similar performance, cost and security can be beneficial in some cases. The difference between those architectures can be measured by calculating the edit distance between two Palladio instances by aggregating the weighted difference of each design option. We assigned a lower weight for differences in the resource environment and higher weight for structural changes to identify architectures with vastly different structural changes. Figures 10 and 12 show a pair of such alternative architectures we identified by comparing distance between alternatives, i.e., structurally quite different but with similar performance and cost, and both secure.

5 Related Work

Here we compare our work to related security modelling and analysis approaches.

Design Space Exploration for Security. Eunsuk Kang [10] identifies the importance of design space exploration for security and outlines key elements of a framework intended to support it. The main focus of his work is on low-level system design and configuration, which is not directly applicable to architecture level design exploration.

Security Modelling Using Palladio Component Model. Busch et al. [6] provide a Palladio extension to predict the mean time to the next security incident. Their methodology is to model what to protect (e.g., data of a database), different ways to access the protected data (e.g., hacking the fronted and then hacking the non-public database), attacker's experience, available knowledge about the system, and the quality of the components in the system. The model can then predict the mean time to the next security incident.

Busch et al.'s approach facilitates security comparison of different architectures and can be used to identify secure architectures. The main limitation is the difficulty of identifying specific model parameters such as the experience of an attacker or quality of a component. It is also complicated to model insider attacks. Nonetheless, the approach defines a metric for system security that might be able to be incorporated into the general approach proposed in this paper.

Quantifying Security. Sharma et al. [20] propose to use Discrete-Time Markov Chains (DTMCs) to model software architecture. This is a hierarchical model that captures quality attributes of components, including security. They quantify security through a model that represents the probability of exposing the vulnerability of a component in a single execution and its effect on system security. This model considers how often a certain component is accessed, which is ignored in our approach based on the assumption that an attacker accesses a component as often as needed. Sharma et al. [20] designed the model to consider the system as broken if at least one component is successfully attacked. Yet, as the systems we consider are typically deployed on several machines, a broken component does not mean that the whole system is compromised. Hence, we designed our approach to consider the control flow of a system as could be followed by an attacker.

Madan et al. [15] propose a Semi-Markov process-based model to quantify security for intrusion-tolerant systems. This model is based on two state-transition models describing how the system behaves under attack. Their scope is Denial-of-Service (DoS) and attacks to compromise the system. The objective of the models is to calculate the Mean Time To Security Failure, to quantify the security of the system. In contrast to this model, our approach can assess the security of component-based architectures and is not restricted to monolithic systems.

SECOMO. SECOMO (Security Cost Model) [14] is a cost modelling technique associated with a framework for risk management in telecommunications. It estimates the effort required to conduct a risk management project in a networked environment. This estimation forms a basis for other task estimations such as the cost, human resources and duration of the project. The estimations are calculated using network size and parameters called scale factors and effort multipliers, which combined together can provide a measure for the security task complexity.

6 Discussion and Future Work

Unlike performance and cost, security is not easily quantifiable. Although security must be considered when making architecture design decisions, the complicated nature of security makes it difficult to follow traditional automated design optimisation practices. In this paper, we demonstrated that, instead of directly modelling the security of architecture, it is possible to perform architecture optimisation using security analysis techniques in conjunction with other quantifiable system properties (cost, performance). We used taint analysis as an example architecture security analysis technique to demonstrate the proposed approach.

Based on system security requirements and a domain of operation, we expect it would be possible to use alternative security analysis techniques such as those discussed in Sect. 5 in place of taint analysis. By using alternative security analysis techniques, users may better identify security vulnerabilities relevant to their domain. We plan to extend this work by developing a wider range of security analysis techniques to be used along with Palladio component model, covering different aspects of security analysis.

In an architectural model, secure components may have higher cost, because of the time and resources required to secure and provide assurance for that component. This may include formal security evaluation techniques such as Evaluation Assurance Level (EAL). These assumptions of increased cost are reasonable, but could be refined or tailored in specific industries or organisations if empirical cost data is available. The security of a component can also depend on the domain. For example, a component might be sufficiently secure for a small-scale software system with no significant security threats, but be non-secure for a highly security-critical system in a hostile environment.

PerOpteryx performs a heuristic search on the design space. So it is not guaranteed to find the optimal or simplest viable architecture. Different initial architectures may converge to different sub-optimal Pareto candidates. The system also does not find a single optimal architecture, but instead defines a range of optimal alternatives on the Pareto frontier. It is the architect's responsibility to choose the final architecture. The architectures discussed here are for illustration purposes only. In real-world scenarios, all the relevant components need to be modelled with higher detail in order to get more accurate results.

Taint analysis technique we chose for the evaluation of the proposed approach outputs a binary value for the security. In the real world, architects may want to use continuous values such as mean time for an attack (see Sect. 5). In such cases, they can apply the same principles we propose and optimise the system for multi-objectives considering security as another dimension because PerOpteryx inherently supports multi-objective optimisations.

7 Conclusion

This paper proposes a new method that incorporates security analysis techniques, in addition to cost and performance (latency), when automatically exploring and optimising system architecture designs. We demonstrate our approach using taint analysis, a basic architecture security analysis technique where secure components stopped propagation of taint from attackers to security-critical components, as the basis for security analysis. We prototyped the approach by extending the Palladio Component model and PerOpteryx systems. The extensions include support for our security modelling and analysis. We reported on the experiment and demonstrate the feasibility of using the approach, illustrating contrasting examples of generated secure system architectures on the cost/performance Pareto frontier.

The evaluation was performed on an industrial example of a secure system architecture for a privacy-preserving computing system. The case study highlighted the usefulness of the approach, by finding that best performance and cost were achieved by non-secure architectures – secure architectures were not far behind, and a variety of distinct design options were identified. Our approach is aimed at supporting architects in identifying and selecting good architecture during the design phase, considering security, cost and performance. In future work, we plan to augment the prototype with support for other security models and analysis techniques.

References

1. Aleti, A., Buhnova, B., Grunske, L., Koziolek, A., Meedeniya, I.: Software architecture optimization methods: a systematic literature review. IEEE Trans. Softw. Eng. **39**(5), 658–683 (2013)
2. Ardagna, D., Casale, G., Ciavotta, M., Pérez, J.F., Wang, W.: Quality-of-service in cloud computing: modeling techniques and their applications. J. Internet Serv. Appl. **5**, 5–11 (2014)
3. Balsamo, S., Di Marco, A., Inverardi, P., Simeoni, M.: Model-based performance prediction in software development: a survey. IEEE Trans. Softw. Eng. **30**(5), 295–310 (2004)
4. Becker, S., Koziolek, H., Reussner, R.: The Palladio component model for model-driven performance prediction. J. Syst. Softw. **82**(1), 3–22 (2009)
5. Brunnert, A., et al.: Performance-oriented DevOps: a research agenda. arXiv preprint arXiv:1508.04752 (2015)
6. Busch, A., Strittmatter, M., Koziolek, A.: Assessing security to compare architecture alternatives of component-based systems. In: International Conference on Software Quality, Reliability and Security. IEEE (2015)
7. Colbert, E., Boehm, B.: Cost estimation for secure software & systems. In: ISPA/SCEA 2008 Joint International Conference (2008)
8. De Gooijer, T., Jansen, A., Koziolek, H., Koziolek, A.: An industrial case study of performance and cost design space exploration. In: International Conference on Performance Engineering. ACM (2012)
9. Djatmiko, M., et al.: Privacy-preserving entity resolution and logistic regression on encrypted data. In: Private and Secure Machine Learning (PSML) (2017)
10. Kang, E.: Design space exploration for security. In: 2016 IEEE Cybersecurity Development (SecDev), pp. 30–36. IEEE (2016)
11. Klein, G., et al.: seL4: formal verification of an OS kernel. In: Symposium on Operating Systems Principles. ACM (2009)
12. Kounev, S., Brosig, F., Huber, N.: The Descartes modeling language. Department of Computer Science, University of Wuerzburg, Technical report (2014)
13. Koziolek, A., Koziolek, H., Reussner, R.: PerOpteryx: automated application of tactics in multi-objective software architecture optimization. In: Proceedings of the QoSA & ISARCS. ACM (2011)
14. Krichene, J., Boudriga, N., Fatmi, S.: SECOMO: an estimation cost model for risk management projects. In: International Conference on Telecommunications, ConTEL 2003, vol. 2. IEEE (2003)
15. Madan, B.B., Goševa-Popstojanova, K., Vaidyanathan, K., Trivedi, K.S.: A method for modeling and quantifying the security attributes of intrusion tolerant systems. Perform. Eval. **56**(1–4), 167–186 (2004)

16. Martens, A., Koziolek, H., Becker, S., Reussner, R.: Automatically improve software architecture models for performance, reliability, and cost using evolutionary algorithms. In: International Conference on Performance Engineering (ICPE) (2010)
17. Newsome, J., Song, D.X.: Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. In: NDSS, vol. 5. Internet Society (2005)
18. Reussner, R.H., et al.: Modeling and Simulating Software Architectures: The Palladio Approach. MIT Press, Cambridge (2016)
19. Safwat, A., Senousy, M.: Addressing challenges of ultra large scale system on requirements engineering. Procedia Comput. Sci. **65**, 442–449 (2015)
20. Sharma, V.S., Trivedi, K.S.: Architecture based analysis of performance, reliability and security of software systems. In: International Workshop on Software and Performance. ACM (2005)
21. Willnecker, F., Brunnert, A., Krcmar, H.: Predicting energy consumption by extending the Palladio component model. In: Symposium on Software Performance (2014)
22. Yang, Z., Yang, M.: LeakMiner: detect information leakage on android with static taint analysis. In: 2012 Third World Congress on Software Engineering (WCSE). IEEE (2012)
23. Yasaweerasinghelage, R., Staples, M., Weber, I.: Predicting latency of blockchain-based systems using architectural modelling and simulation. In: International Conference on Software Architecture (ICSA) (2017)
24. Yasaweerasinghelage, R., Staples, M., Weber, I., Paik, H.Y.: Predicting the performance of privacy-preserving data analytics using architecture modelling and simulation. In: International Conference on Software Architecture (ICSA) (2018)