Evaluating Security Properties of Architectures in Unpredictable Environments: A Case for Cloud

Funmilade Faniyi, Rami Bahsoon
University of Birmingham
Edgbaston, Birmingham
B15 2TT, UK
f.faniyi@gmail.com, r.bahsoon@cs.bham.ac.uk

Andy Evans
Xactium Limited
PCI House, Woodseats Close
Sheffield, S8 0TB, UK
andy.evans@xactium.com

Rick Kazman
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
kazman@sei.cmu.edu

***Abstract*—The continuous evolution and unpredictability underlying service-based systems leads to difficulties in making exact QoS claims about the dependability of architectures interfacing with them. Hence, there is a growing need for new methods to evaluate the dependability of architectures interfacing with such environments. This paper presents a method for evaluating the security quality attribute of architectures in service-based systems.**

The proposed method combines some properties of the Architectural Tradeoff Analysis Method (ATAM) and security testing using Implied Scenario. In particular, the scenario elicitation process of ATAM is improved by utilising Implied Scenario technique to generate scenarios which may be undetected using plain ATAM. An industrial case study of a problem related to securing data at the Software-as-a-Service layer on Force.com Cloud platform is adopted to validate the new method. The results indicate that our method found four additional security scenarios beyond the plain ATAM, resulting in four new risks and two new tradeoff points.

***Keywords*-ATAM, Implied Scenario, Security, Cloud Architectures, Dynamic Architectures**

I. Introduction

The success recorded by some of the dominant Cloud service providers (e.g. Amazon EC2, Salesforce, Google App Engine) has motivated many organisations to consider adopting Cloud services as a means of reducing IT operational cost amongst other benefits. The risk of data misuse, data loss and security breaches however, remains a major concern for most Cloud users. An example is the case of a phishing attack targeted at Salesforce.com which led to the theft of customers' emails and addresses [1].

The inability of Cloud providers to meet security requirements which are mandated by territorial laws governing critical applications such as financial and health systems has necessitated the design of secure architectures interfacing the Cloud to exploit the economies of scale offered by Cloud Computing. One approach adopted by such Cloud interfacing architectures is to store some sensitive data in their local datacentres while other non-sensitive data are stored in the Cloud. Two examples in the literature adopting this approach are the work of [1] and [2]. While these results provide some insight into addressing the security challenges impeding Cloud adoption, existing architecture evaluation methods are not suitable for verifying whether such architectures are indeed secure.

We argue that the existing architecture evaluation methods have limitations when assessing architectures interfacing with unpredictable environments such as the Cloud. This is because the Cloud environment is fundamentally different from the classical environments for which most software evaluation methods were developed [3]. The unpredictability of this environment is attributed to the dynamic elasticity, scale, and continuous evolution of the Cloud topology (e.g. due to new services, mashups, unpredictable modes of service use, fluctuations in QoS provision due to unpredictable load/growth etc.). As a result, architectures interfacing such unpredictable environments are expected to encounter many uncertainties resulting from emerging behaviour from concurrent and unexpected modes of interaction of their components with the Cloud environment and its various components. These challenges call for holistic approaches combining aspects of both dynamic and static evaluation.

The aim of this paper is to present a methodology for evaluating the security quality attribute of architectures in unpredictable environments such as the Cloud. Our approach extends well established methods in software architecture namely: Architectural Tradeoff Analysis Method (ATAM) [4] and Implied Scenario for security testing [5]. We empirically evaluated the new method by considering a non-trivial industrial case study with the goal of securing data on the Force.com Cloud platform. The combination of both approaches (ATAM and Implied Scenario) has proven to be effective, where the benefits outweigh the cost. This is because the combination has provided the architect with a systematic and well informed methodology for detecting the "right" scenarios, which are critical for revealing security threats and weaknesses in the key architecture design decisions. The results indicated that our method was able to detect critical security scenarios not captured with the use of static analysis alone. The method was beneficial to the architect for understanding the architecture from both static and dynamic analysis perspectives. Furthermore, proven ex-

isting tools were utilised to facilitate the evaluation process, thus making the method easily accessible to architects.

The rest of the paper is structured as follows. In section 2 we provide the background required to understand ATAM and Implied Scenario. Our proposed methodology is presented in section 3. The case study adopted for evaluating our method is presented in section 4, while the architecture evaluation is the focus of section 5. In section 6 we discuss the lessons learned from our experience. Section 7 covers a review of related work and we conclude in section 8.

II. Background

A brief overview of the ATAM and Implied Scenario are presented as a precursor for our discussion about the new evaluation methodology that systematically blends these methods together.

A. ATAM

The ATAM is considered a mature and validated scenario-based Software Architecture (SA) evaluation method [6]. The inputs of the ATAM are scenarios elicited by stakeholders and documented descriptions of the architecture. The goal of the ATAM is to analyse architectural approaches with respect to scenarios generated from business drivers for the purpose of identifying risk points in the architecture [4]. This is achieved by a disciplined reasoning about SA relating to multiple quality attributes [6]. There are two important classifications of risk points in ATAM namely *sensitivity points* and *tradeoff points*. A sensitivity point refers to a parameter of the architecture that affects the achievement of one quality attribute. On the other hand, a tradeoff point refers to a parameter of the architecture that affects the achievement of more than one quality attribute, where one improves and the other degrades. These risk points, together with extensive documentations of the architecture, scenarios, and quality-attributes analyses are the products of ATAM. A more detailed description of ATAM is presented in [7].

B. Implied Scenario Testing

Scenarios generally refer to a description of an interaction between a user and the system or interaction among components of a system [8]. Implied scenarios are unanticipated interactions among components which arise when multiple well understood scenarios are combined together. An implied scenario could take different forms: a useful scenario which could aid understanding of the system, a trivial scenario which could be ignored or an unacceptable scenario which could pose risks to the system [5], [9].

A framework for synthesising behaviour models for scenario-based specifications and detecting implied scenarios was proposed by [9]. This framework was integrated with the Labelled Transition System Analyser (LTSA)[1] open source tool [10] allowing the representation of a system

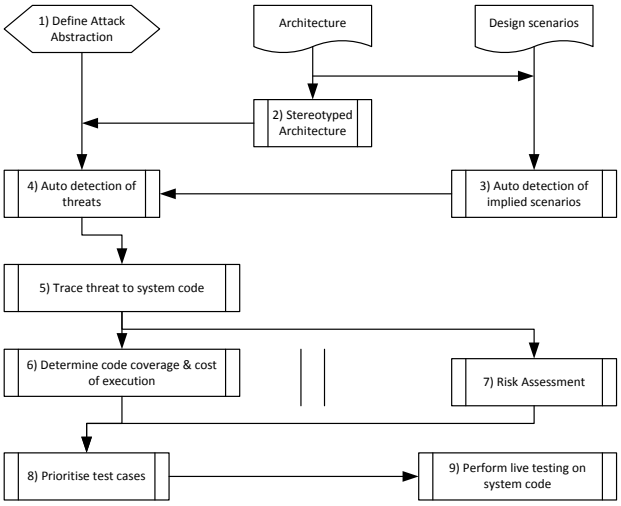


Figure 1. Implied Scenario Security Testing Methodology [11]

model via Message Sequence Charts (MSC). Thereafter, the application of Implied Scenario for testing architectures for security risks was motivated by [11]. A model-based technique for applying the concept to security testing was later proposed by [5]. This technique utilised the framework proposed by [9] as a mechanism for automating the Implied Scenario detection process. The seamless plug-in of the LTSA tool with the Acme[2] architectural modelling tool offers the benefit of easily representing architectures modelled with Acme in a way suitable for implied scenarios analysis.

Our proposed methodology builds on these evaluation methods and lessons learned from their applications (e.g., via case studies and illustrative examples).

III. Proposed Methodology

As earlier motivated, our methodology combines the strengths of the ATAM with the dynamic scenario generation process of the Implied Scenario technique. Our motivation for combining aspects of both methods is based on observable weaknesses in each of them. Precisely, we noted that the ATAM suffers from the following weaknesses:

- The iterative nature of the ATAM means it is necessary to have a substantial number of human experts on the team at different times to provide insight into the implications of different architectural approaches. From our experience in evaluating architectures (in general), it is often expensive to speculate the availability of such domain experts for small-mid size software projects due to budgetary or time constraints.
- Also, the architectural analysis in the ATAM takes into consideration only a few highly prioritised top-level

[1] http://www.doc.ic.ac.uk/ltsa/

[2] http://www.cs.cmu.edu/ acme/

scenarios elicited during the utility tree and brainstorming steps. We argue that the selection of only a few top-level scenarios may lead to missing some critical security scenarios which may later be exploited by attackers.

On the other hand, the Implied Scenario evaluation technique is a semi-automated process that combines both tool-based scenario analysis and subjective expert judgment [11]. When used singlehandedly it may be susceptible to the following weaknesses:

- The sole use of Implied Scenario as an evaluation technique is not as mature as ATAM in terms of disciplined reasoning about architecture design decisions, although it suggests the use of expert judgment to select and prioritise security scenarios.
- There is as yet no provision to understand the impact of implied scenarios on other quality attributes aside from security and reliability.

Therefore, we propose the following enrichment to ATAM with specific consideration for evaluating security properties of architectures:

1) The steps of the ATAM evaluation technique should be performed in a logical and iterative sequence to assess the security attribute of the architecture and its tradeoff against other non-security quality attributes.
2) The risks, sensitivity points and tradeoff points, identified in the ATAM process should be used to refine the architecture in order to mitigate their impact.
3) Next, the interaction among the security critical components in the revised architecture from Step 2 should be modelled using MSCs and fed into the LTSA tool for the purpose of detecting implied scenarios.
4) If any implied scenario is detected go to Step 5 or else terminate the evaluation process.
5) The detected implied scenario(s) is/are analysed by members of the ATAM team and classified appropriately either as non-risk or risk. If the implied scenario(s) is/are considered as risk points, the expertise of the ATAM team is exploited to take one or more of the follow actions.
    a) Evaluate the cost of refining the architecture to address the risk.
    b) Prioritisation of the risks based on risk assessment outcome.
    c) Implement mitigation measures to reduce likely impact of risk.
    d) Ignore the implied scenario if it is considered trivial.
6) In the presence of critical security risks, the ATAM team may choose to repeat step 2 - 5 to possibly uncover additional subtle implied scenarios.

Implied scenarios that were derived by modelling the concurrent interaction of architectural components improves the scenario elicitation process of the ATAM (step 3-5) by generating additional subtle scenarios which may not be discovered using ATAM alone. The scenarios generated using the implied scenario method as a follow-up to ATAM serve to provide a better understanding of security risks. Within a dynamic unpredictable environment such as Cloud Computing, there are benefits of using this method to unveil security risks arising from concurrent component interactions which may otherwise only be discovered after deployment.

The method benefits from the expertise of the ATAM team to decide the best course of action after the detection of implied scenarios (step 4-5).

We suggest that the method should not be followed in a rigid sequence for all cases. In our opinion, it is logical to first account for architectural risks by subjective human judgment before applying tool-based dynamic analysis to detect implied scenarios. The reverse of the process may be beneficial in cases where little is known about the architecture style or the evaluator is not fully competent to carry out a proper ATAM security evaluation. In the rest of this paper, we shall refer to this method as *ATMIS* (Architectural Tradeoff Method using Implied Scenario).

IV. Case Study

We adopted an industrial case study provided by Xactium Limited[3] as a means of evaluating the ATMIS method. The following criteria were used to measure the achievement of the goals of study:

1) Does the ATMIS method detect relevant security risks in the architecture following initial refinement using ATAM?
2) How critical are the detected security risk points?
3) Could any detected security risks have been detected using other static methods (i.e. plain ATAM and non-ATAM methods)?

A. The Problem

Smart bank has recently decided to adopt Force.com SaaS Cloud provider as the platform to deliver a robust, scalable and on-demand service provisioning for the risk management aspect of its business.

In order to achieve the goal of the study, the first step taken was to understand the requirements. Following this exercise, a subset of elicited requirements is presented below:

R1: Smart bank's policies prohibit certain sensitive data from being hosted on any server outside the its country of operation. Thus, data fields specified as sensitive should not be accessible in clear text outside this territory.

R2: Security mechanisms should not degrade defined performance thresholds. Specifically, response time for

[3]http://www.xactium.com

add, delete, update and display data operations should not exceed 5 secs at peak period and 2.5 secs during normal operations.

R3: Critical components within the architecture should not constitute a single point of failure thereby affecting the system's uptime.

R4: Any mechanism adopted should not impact the programmability of the platform. Thus, native Force.com functions and reporting tools should not require additional language extensions.

R5: It should be easy to manage security keys and add new data fields to the sensitive dataset without compromising the security of the system.

Requirements R1, R3 and R5 have security implications which must be factored into the design decisions taken to realise the architecture. In addition, requirements R2, R3 and R4 should be factored into performance, availability and modifiability design decisions of the architecture respectively.

B. Architectural Design

Members of the architecture team elicited and reviewed several candidate architectures prior to the evaluation process (E.g. [1]). The candidate architecture adopted to meet the requirements specified by Smart bank is shown in Figure 2.

The security objective of this architecture is to ensure that data fields marked as *sensitive* by the bank administrator (denoted by Admin in Figure 2) are stored in encrypted format, while other non-sensitive data fields are stored in clear text within the Cloud. It should be impossible for an attacker to link the encrypted sensitive data fields in the Cloud to any local data stored within the bank's subsystem. Some of the important features of this architecture are elaborated upon below.

- *Web Service*; a middleware between the Cloud infrastructure and the bank's subsystem for exchanging metadata required for synchronising the state of the two subsystems.
- *Security Manager*; is the component responsible for managing the security of data passed on to the Cloud and also making it readable when fetched by Cloud users. It consists of 3 main sub-components namely: data filter, encryption and decryption components.
- *Data Repository*; this refers to the database located within the bank's premises storing data field definition, security keys and sensitive data.
- *Administration Tool*; is used by the bank's IT administrator to flag data fields as either sensitive or non-sensitive and also to manage cryptographic keys on behalf of users.

C. Component Description

The main sub-components of the Security Manager are described here to provide sufficient information for security analysis.

- Data Filter Component; requires a record consisting of data fields. Suppose we have a record, R with fields, $f_1, f_2, ..., f_n$. The data filter component queries the data repository for the type (i.e. sensitive or non-sensitive) of each f_i in R. Each f_i marked as sensitive is passed to the encryption component for encryption and the returned ciphertext, c is sent to the Cloud in place of the original value of f_i.
  The reverse is the case when the data is fetched from the Cloud. If the field f_i was previously encrypted then its ciphertext value, c is passed to the decryption component and the decrypted result (i.e. original value of f_i) is presented to the user.
- Encryption Component; this component requires the data filter component for the case where the field f_i is a sensitive field. The encryption component encrypts the value assigned to field f_i using the following scheme:

$$h \leftarrow H(f_i||n_a)$$

$$c \leftarrow Enc(h||s, K)$$

  where, $||$ is the concatenation operator
  n_a is a nonce (a random number generated and used only once)
  H is a hash function
  f_i is the original value of the sensitive field
  K is the user's symmetric encryption key
  s is a unique identifier generated per encryption
  Enc is the encryption function, and
  c is the resultant ciphertext
  The value of c is returned to the data filter component while those of f_i, n_a and s are stored securely. The ciphertext corresponding to the value of each sensitive field, along with the values of the non-sensitive fields in the record are passed on to the Cloud.
- Decryption Component; requires the ciphertext, c stored in the Cloud. The value of c is decrypted to its original plaintext using the key, K, assigned to the user requesting the data. The decryption scheme is as follows:

$$Dec(c, K) \rightarrow h||s$$

  The component retrieves the values of f_i and n_a based on the value of s. The verification, $h = H(f_i||n_a)$ is performed to ensure the encrypted value has not been compromised. If the verification succeeds, the actual value f_i is displayed to the user.

Figure 3a depicts the interaction among these components. We reiterate here that while the detailed description of components informs proper analysis, our objective is not to

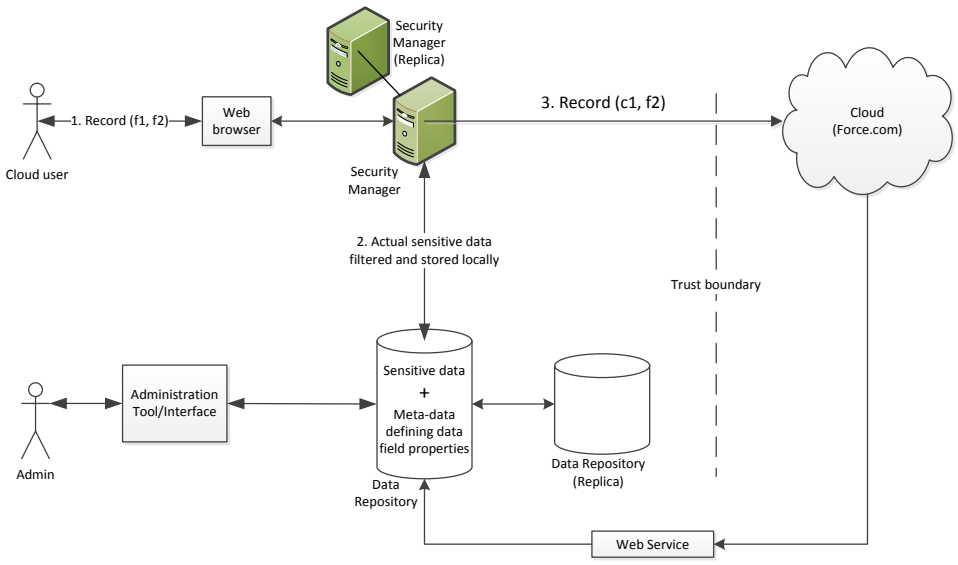


Figure 2. Smart Bank Cloud Architecture

design cryptographic protocols, rather we seek to identify security risks in the architecture by a systematic application of the ATMIS method.

V. Architectural Evaluation

A. Direct Evaluation (ATAM)

Space limitation constrains us from providing full details of the ATAM analysis. Hence, we provide only relevant information required to understand how we arrived at the ATAM deliverables; especially risks, sensitivity and tradeoff points.

1) Security Quality-Attribute Characterisation: The work of [7] provided characterisation for the availability, performance and modifiability quality attributes. We found only one such characterisation for the security quality attribute [12]. Since our work is concerned with the security properties beyond the infrastructural level, it was necessary for us to design a characterisation of the security quality attribute which we used to elicit security-specific questions for probing the architecture. Figure 4 shows this security characterisation.

2) Security Analysis: Given the quality attributes identified from the scenarios from all stakeholders. Utility trees were created to elicit specific scenarios related to these quality attributes. The scenarios were prioritised with respect to their importance to the realisation of the client requirements. The security quality attribute of the architecture was described using the expression:

$$Q_s = F(D_f, E, D)$$

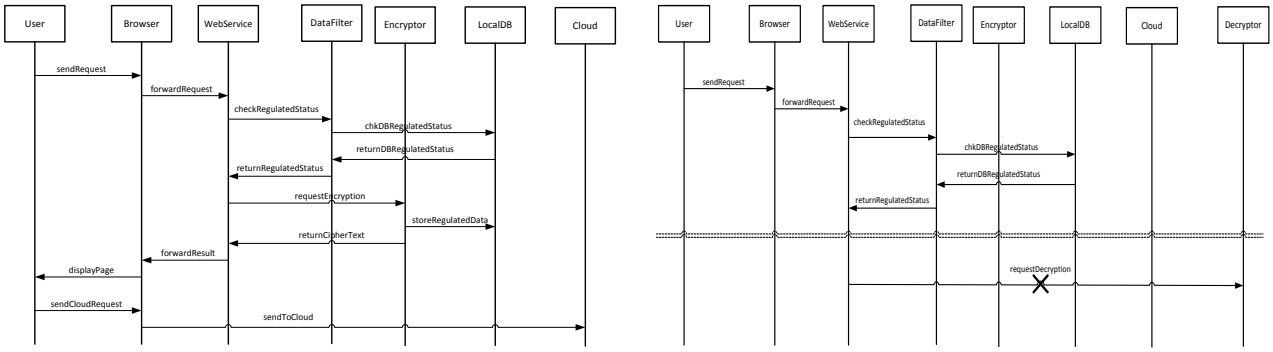
where Q_s refers to the security quality attribute, F is the function taking the security parameters, D_f is the data filter component, E represents the encryption component and D is the decryption component.

- *Data Filter Component, D_f:* This component was modelled as

  $$D_f = g(f)$$

  where the function g returns the query latency per field, f, to determine whether it is a sensitive field or not. Consider the case where the function $g = 5ms$ per data field. The performance impact is minimal when few data fields are involved, but this impact increases exponentially as the number of data fields increases. This problem may result in unintended security consequences such as buffer overflow which may be exploited by an attacker to execute malicious code.
- *Encryption component, E:* Recall that the parameter, s, in the encryption scheme is a unique identifier generated per encryption operation. The limit of such a unique identifier generator could represent a risk point for the architecture. This is important because any collision in the value of s will result in the same ciphertext for two separate data fields having equal values.
- *Decryption Component, D:* The following questions were used to probe this component: What is the complexity of the decryption function? During peak periods, what is the cost of the repository fetch operation for retrieving a user's key based on the value of s? What is the complexity of the verification function?

3) Risk, Sensitivity points and Tradeoff points: In this section some of the security-related tradeoff points discovered during the ATAM analysis are presented.



(a) Save Sensitive Data Scenario (b) Detected Implied Scenario

Figure 3. Architectural Component Interactions

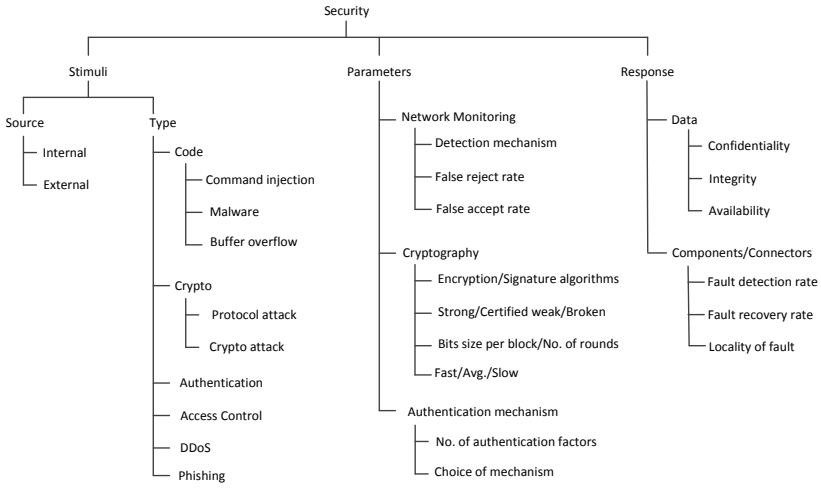


Figure 4. Security Characterisation

- *Security Vs Usability*: This tradeoff is evident from the choice of key management scheme. While centralised key management offers better usability by isolating the users from the burden of key generation, renewal and revocation. Decentralised key management offers better security but less usability.
- *Security Vs Maintainability*: Storing sensitive data in-house and non-sensitive data in the Cloud provides better assurance since sensitive data could not be easily compromised because they are under the control of the client. However, if a substantial amount of the client's data are sensitive, the maintenance overhead incurred by storing them in-house could outweigh the benefits of adopting CC for provisioning the service.
- *Security Vs Performance*: Using arguments similar to (2) above; the more data marked as sensitive, the more performance overhead incurred due to encryption/decryption of data. Also the computational complexity of the encryption scheme (symmetric or asymmetric) could further degrade performance.
- *Security Vs Locality of Data*: Replication of data across multiple sites introduces the risk of exposing the data to security attacks. Suppose a replica of the data repository is held in a country which does not enforce strong security policies (e.g. permitting encryption algorithms which are already broken or weak encryption keys). There is the risk of compromising the data from this replica, while the primary data repository may be in a more secure location.
- *Security Vs Availability*: The Security Manager is a potential single point of failure, hence a replica of the component is provided to ensure availability in the event of a distributed denial of service (DDoS) attack on the primary Security Manager. In addition, the use of a DDoS detection mechanism would serve to facilitate early detection of such attacks.

4) Findings: Some of the important findings from the ATAM analysis are: (i) The security of the Data Repository poses the highest risk to the security of the architecture. (ii) There could be a huge cost in terms of maintenance

and performance if a large proportion of the client data are sensitive. (iii) The Administration tool represents a single point of absolute trust in the architecture. Therefore, an additional layer of security is required to secure data in the presence of corrupt administrators. Two approaches are to: (a) Use a Trusted Platform Module (TPM) [13] thereby transferring trust to a tamper-resistant hardware. (b) Utilise a (k, n) *threshold scheme* (e.g [14]) to share trust among a number of trusted entities.

While some of the findings from the analysis may appear unsubtle, they were not obvious prior to the evaluation of the architecture using the ATMIS method.

B. Indirect Evaluation (Testing for Implied Scenarios)

This phase of ATMIS could be described as the enrichment to plain ATAM because it involves performing dynamic analysis of component interactions to reveal security risks resulting from concurrent behaviour. Detected implied scenarios are useful to architects since security risks that were not captured by human static analysis efforts may be revealed. We classified these implied scenarios into one of three categories: *high risk*, *medium risk* and *low risk*. High risk implied scenarios will necessarily require a refinement of the architecture to ensure that they are not exploited. Low risk implied scenarios may be ignored, but they are documented as part of the evaluation report. The treatment of medium risk implied scenarios is left to the judgment of the evaluation team. Steps 5(a) - 5(d) in section 3 could be used as a guide to making such decisions.

1) Analysis: The open-source LTSA tool [10] was used to carry out the Implied Scenario evaluation on behavioural models of the components interaction. Our experiments covered scenarios for: storing sensitive and non-sensitive data in the Cloud; viewing sensitive and non-sensitive data previously stored or processed in the Cloud; adding, deleting and updating cryptographic keys; and designating data fields as either sensitive or non-sensitive. As an example, the *SavingSensitiveData* scenario is shown in Figure 3a to illustrate how components interact in the system.

2) Findings: After modelling the interaction among the scenarios, one implied scenario was detected (see Figure 3b). This implied scenario corresponds to the case when the client makes a web service call to encrypt a sensitive field value but an attacker is able to manipulate the web service to perform a decryption operation instead. Such a vulnerability could be exploited to perform a *key replay attack* on sensitive fields. This attack could be achieved by first requesting an encryption of a sensitive field value and subsequently requesting a decryption with some previously compromised key.

The detected implied scenario is not trivial as this represents a security risk which could lead to further security scenarios when examined from a technical/business perspective; hence it was classified as high risk. Consider a use-case where a member of staff at Smart bank requests to secure the field representing the Net Loss of Smart bank for the month. This information could be abused by a competitor to reduce the reputation of Smart bank before its customers in the media, thereby leading to loss of business for Smart bank. As an instance of a possible attack path that could be derived from this implied scenario; a malicious software developer in possession of a prerecorded ciphertext value of the field may compromise the architecture as follows:

- Bank staff member sets out to encrypt a sensitive data field (E.g. Net Loss value).
- Web service queries Security Manager for sensitive status of this data field.
- Web service passes the request to the Security Manager.
- Security Manager returns result indicating that the data field is sensitive.
- Instead of calling the encryption component, the malicious web service consumer calls the decryption component and passes the prerecorded ciphertext to it as parameter value.
- The web service returns the actual value of the Net Loss in plaintext.

These types of attack are predominant in dynamic environments where run-time composition of components makes it impossible to fully account for unpredictable interactions of components until deployment time.

C. Post-Evaluation Activities

The implied scenario detected in the first phase of the evaluation was added to the top scenario list of the ATAM and the architecture was subsequently revised to address it. Further testing of components derived from the refined ATAM architecture using Implied Scenario technique revealed 2 additional implied scenarios which were classified as high security risks (we omit their description due to spatial constraints). In accordance with the iterative nature of our method, the architecture was refined twice in this case before the evaluation process was terminated. During each iteration of the process, Implied Scenario technique generated scenarios which were missed during the ATAM static analysis, while the expertise of the ATAM team was used to evaluate and prioritise security-related implied scenarios at the background. Architecture refinement decisions were made by the architects with guidance from the ATAM team. Table II shows a summary of the evaluation result. A scaled graph showing the time and overhead incurred using our method is shown in Figure 5.

Following the architectural evaluation, we have implemented a prototype of the fittest candidate architecture. The prototype has proved to be intact with respect to the detected and probed security scenarios. The time spent testing the prototype system for security requirements was shorter. Hence, the time spent evaluating architectures for security paid off significantly by detecting avoidable security

Table I
COMPARISON OF EVALUATION METHODS

Components \ Methods	ATAM	Implied Scenario	ATMIS
Maturity stage	Refinement	Development	Inception
Process support	Comprehensively covered	Coarse-grained description	Embedded in method description
Method's activities	9 activities in 2 phases	Message sequence chart analysis	ATAM & Implied Scenario activities
Method's goals	Sensitivity & Tradeoff analysis	Detect implied scenarios	Sensitivity & Tradeoff analysis
Quality attributes	Multiple attributes	Security attributes	Multiple attributes
Applicable project stage	After architecture design	After architecture design or implementation	After architecture design or implementation
Architectural description	Process, data flow, uses, physical & module views	Process & data flow	Process, data flow, uses, physical & module views
Evaluation approaches	Questioning & measuring	Tool-based analysis	Questioning, measuring & tool-based analysis
Tool support	Partially Available	Available	Available
Stakeholders involved	All major stakeholders	Architecture Designer	All major stakeholders

Table II
SUMMARY OF EVALUATION RESULT

Metrics \ Method	ATAM	ATMIS
No. of candidate architectures	4	4
No. of scenarios	25	29
Security-related scenarios	9	13
Security risk points	5	9
Tradeoff points	8	10
Stakeholders involved	All	All
No. of architecture refinements	0	2
Evaluation duration	3 days	4 days

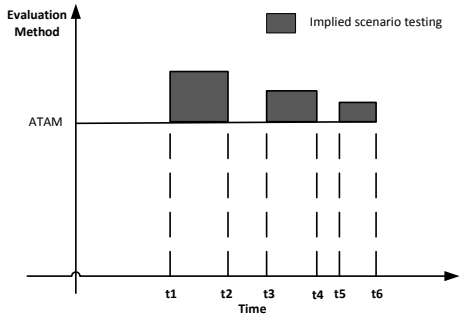


Figure 5. Effort Estimation

risks early in the development cycle. In addition, the results improved the architect's confidence in the fitness of the proposed architecture towards meeting client requirements.

VI. DISCUSSIONS

Using the framework proposed by [15], we compared ATMIS with plain ATAM and Implied Scenario technique as shown in Table I. The results indicated that ATMIS method is a more holistic approach to evaluating architectures for security. In particular, our method which is based on expert questioning, measurement and tool-based analysis offers a better coverage of scenarios than the ATAM, and a better expert analysis than Implied Scenario technique. At the moment, the maturity stage rating of our method (i.e. inception) makes it hard to arrive at strong claims about its generalisation. Consequently, more examples and case studies are required to further validate the ATMIS method.

From figure 5, it is clear that using ATMIS takes more time than using only ATAM. Therefore, a trade-off exists between the time required to perform the evaluation and the probing of the architecture to ensure that as many security scenarios as possible are accounted for. In an environment where thousands of components are involved, our method may require much more time to get results. The criticality of the architecture or application domain may guide how this trade-off will be addressed. In particular, the niche area of critical systems architecture where zero-tolerance for security failure is desirable may find the method very applicable.

A more principled approach may also be required to guide the decision-making process of whether to embark on architecture refinement or not after the detection of implied scenarios. As highlighted by the method, one approach to achieve this is to perform a cost estimation of the likely

impact of the security risk [16]. A possible path to consider is the idea of using Real Options [17] to precisely quantify security risks generated from implied scenario.

VII. Related Work

A model-driven framework called TREAT (Tracking of REquirements And Threats) was proposed by [18]. Their method blends together several architectural evaluation and software testing techniques including the ATAM. The objective of their approach was to match each stage of the software development process with the risk analysis method most suitable for it. They have considered *misuse cases* elicited by the architects as a mechanism for identifying security risks to the architecture. This approach suffers from the same weakness as the ATAM, since it is possible for the architects (human factor) to miss certain misuse cases which may pose security risks to the architecture.

Past experience reports using ATAM (E.g. [19]) have also motivated the need to improve the method based on limitations uncovered during its usage in various settings. Some attempts to achieve these involved combining the ATAM with other methods [20], [21]. However, most of these results have adhered to the static analysis theme underlying the ATAM. More importantly, they have provided insight into mechanisms for adapting scenario-based evaluation to architectures within different domains and industrial settings [22]–[24]. However, none of them have attempted to adopt a dynamic analysis approach to enrich the ATAM.

On the other hand, Implied Scenario is primarily a subject of academic research for testing architectures for security, reliability, concurrency and assessing the design of implemented systems [5], [25]. To the best of our knowledge, the use of Implied Scenario and associated tools (Acme and LTSA) in industrial settings has not been reported.

VIII. Conclusion

In this paper, we have motivated the need for architecture evaluation methods suitable for the dynamic unpredictable environments such as Cloud Computing. In particular, we have presented an evaluation method based on ATAM and Implied Scenario for evaluating the security quality attribute of architectures in this domain. The novelty of our work lies in the fact that we have been able to address weaknesses in a static analysis architecture evaluation method (ATAM) by enriching it with innovative ideas from a dynamic analysis method (Implied Scenario) to generate subtle scenarios which may lead to security attacks on the architecture if undetected. We have exemplified the approach with an industrial case study of an architecture interfacing the Cloud. The results indicated that our methodology found additional security scenarios beyond the plain ATAM, resulting in new risks and tradeoff points.

We are currently investigating the use of the method to evaluate architectures in various domains like identity management and enterprise web application. In the future, we seek to report more case studies to illustrate the repeatability of the method and further refine it to address emerging issues.

Acknowledgment

The authors would like to thank the software architects in Xactium Limited for their valuable contribution to the architecture evaluation process.

References

[1] M. Mowbray and S. Pearson, "A client-based privacy manager for cloud computing," in *COMSWARE '09: Proceedings of the Fourth International ICST Conference on COMmunication System softWAre and middlewaRE*. New York, NY, USA: ACM, 2009, pp. 1–8.

[2] Amazon Web Services, "Creating HIPAA-Compliant Medical Data Applications with Amazon Web Services," Amazon Web Services, Tech. Rep., April 2009.

[3] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," NIST, Information Technology Laboratory, Tech. Rep., 2009.

[4] R. Kazman, M. Klein, M. Barbacci, T. Longstaff, H. Lipson, and J. Carriere, "The architecture tradeoff analysis method," in *Proceedings of the Fourth IEEE International Conference on Engineering of Complex Computer Systems (ICECCS)*. Monterey, CA: IEEE Computer Society, 1998, pp. 68–78.

[5] S. Al-Azzani and R. Bahsoon, "Using implied scenarios in security testing," in *SESS '10: Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems*. New York, NY, USA: ACM, 2010, pp. 15–21.

[6] M. A. Babar and I. Gorton, "Comparison of scenario-based software architecture evaluation methods," in *APSEC '04: Proceedings of the 11th Asia-Pacific Software Engineering Conference*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 600–607.

[7] R. Kazman, M. Klein, and P. Clements, "ATAM: Method for Architecture Evaluation," Carnegie Mellon University, Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213, Tech. Rep., August 2000.

[8] R. Kazman, S. J. Carrière, and S. G. Woods, "Toward a discipline of scenario-based architectural engineering," *Ann. Softw. Eng.*, vol. 9, no. 1-4, pp. 5–33, 2000.

[9] S. Uchitel, J. Kramer, and J. Magee, "Detecting implied scenarios in message sequence chart specifications," in *ESEC/FSE-9: Proceedings of the 8th European software engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of software engineering*. New York, NY, USA: ACM, 2001, pp. 74–82.

[10] S. Uchitel, R. Chatley, J. Kramer, and J. Magee, "Ltsa-msc: tool support for behaviour model elaboration using implied scenarios," in *TACAS'03: Proceedings of the 9th international conference on Tools and algorithms for the construction and analysis of systems*. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 597–601.

[11] S. Al-Azzani and R. Bahsoon, "Semi-automated detection of architectural threats for security testing," in *ESEC/FSE Doctoral Symposium '09: Proceedings of the doctoral symposium for ESEC/FSE on Doctoral symposium*. New York, NY, USA: ACM, 2009, pp. 25–26.

[12] A. Raza, H. Abbas, L. Yngstrom, and A. Hemani, "Security characterization for evaluation of software architectures using atam," in *Information and Communication Technologies, 2009. ICICT '09. International Conference on*, aug. 2009, pp. 241 –246.

[13] E. Vila and P. Borovska, "Data protection utilizing trusted platform module," in *CompSysTech '08: Proceedings of the 9th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing*. New York, NY, USA: ACM, 2008, pp. V.13–1.

[14] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[15] M. A. Babar, L. Zhu, and R. Jeffery, "A framework for classifying and comparing software architecture evaluation methods," in *ASWEC '04: Proceedings of the 2004 Australian Software Engineering Conference*. Washington, DC, USA: IEEE Computer Society, 2004, p. 309.

[16] S. A. Butler, "Security attribute evaluation method: a cost-benefit approach," in *Proceedings of the 24th International Conference on Software Engineering*, ser. ICSE '02. New York, NY, USA: ACM, 2002, pp. 232–240. [Online]. Available: http://doi.acm.org/10.1145/581339.581370

[17] J. Li and X. Su, "Making cost effective security decision with real option thinking," in *Proceedings of the International Conference on Software Engineering Advances*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 14– . [Online]. Available: http://portal.acm.org/citation.cfm?id=1304609.1306415

[18] B. Malone and A. Siraj, "Tracking requirements and threats for secure software development," in *ACM-SE 46: Proceedings of the 46th Annual Southeast Regional Conference on XX*. New York, NY, USA: ACM, 2008, pp. 278–281.

[19] S. Ferber, P. Heidl, and P. Lutz, "Reviewing product line architectures: Experience report of atam in an automotive context," in *Software Product-Family Engineering*, ser. Lecture Notes in Computer Science, F. van der Linden, Ed. Springer Berlin / Heidelberg, 2002, vol. 2290, pp. 194–197.

[20] A. Erfanian and F. Shams Aliee, "An ontology-driven software architecture evaluation method," in *SHARK '08: Proceedings of the 3rd international workshop on Sharing and reusing architectural knowledge*. New York, NY, USA: ACM, 2008, pp. 79–86.

[21] P. Wallin, J. Froberg, and J. Axelsson, "Making decisions in integration of automotive software and electronics: A method based on atam and ahp," in *SEAS '07: Proceedings of the 4th International Workshop on Software Engineering for Automotive Systems*. Washington, DC, USA: IEEE Computer Society, 2007, p. 5.

[22] T. Kim, I. Y. Ko, S. W. Kang, and D. H. Lee, "Extending atam to assess product line architecture," in *Computer and Information Technology, 2008. CIT 2008. 8th IEEE International Conference on*, jul. 2008, pp. 790 –797.

[23] J. Lee, S. Kang, H. Chun, B. Park, and C. Lim, "Analysis of van-core system architecture- a case study of applying the atam," in *Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing, 2009. SNPD '09. 10th ACIS International Conference on*, may. 2009, pp. 358 –363.

[24] F. Olumofin and V. Misic, "Extending the atam architecture evaluation to product line architectures," in *Software Architecture, 2005. WICSA 2005. 5th Working IEEE/IFIP Conference on*, 2005, pp. 45 –56.

[25] G. N. Rodrigues, D. S. Rosenblum, and S. Uchitel, "Sensitivity analysis for a scenario-based reliability prediction model," in *WADS '05: Proceedings of the 2005 workshop on Architecting dependable systems*. New York, NY, USA: ACM, 2005, pp. 1–5.