Dealing with Security: A Multiple Case Study on Software Architects

Vidya Lakshminarayanan, WenQian Liu✶, Charles L Chen, Dewayne E Perry

Empirical Software Engineering Lab (ESEL)
Electrical and Computer Engineering
The University of Texas at Austin, Austin TX
{vidya,clchen,perry}@ece.utexas.edu

✶Software Engineering
Department of Computer Science
University of Toronto, Canada
wl@cs.toronto.edu

Abstract

While security has long been a significant issue in military systems, the spread of the internet has stimulated a growing interest in, and increasing demand for, secure systems. Understanding how architects manage security requirements in practice is a necessary first step in providing repeatable processes using effective techniques, methods and architectural structures. We present the results of multiple case studies of practicing security architects: key aspects in security requirements, critical issues in managing security requirements, essential characteristics of security architects, the relationship of security with evolution, and problem solving by security architects. We conclude with a lessons learned, and a discussion of related and future research.

1 Introduction

Security has long been a major issue in military and defense systems. Making sure that only the right people get access to information, that plans do not land in the wrong hands, and that communication channels are not compromised are among the top priorities for national defense. More recently, the internet boom has exacerbated the problem. By connecting everyone with everyone else, the internet has greatly enhanced our ability to exchange information, but it has also opened more doors for attackers. With the growing concern over malicious attacks compromising data integrity and privacy, security in software systems has become an increasingly important topic and has led to increased software engineering research [1][6][7][10]. The increasing demand and importance of security requirements in systems engineering has created a relatively new engineering discipline called 'security engineering'.

Our research goal is to understand how software architects view and manage security requirements and architectural designs for secure systems in practice. Understanding how architects manage requirements gives us a solid foundation on which to develop techniques, methods, processes, and tools to aid architects in managing requirements and transforming them into architectures. We take an empirical based approach and use an interview based case study methodology to carry out our investigations. Case studies are a specific empirical research method used to gain a deep understanding of a particular phenomenon in its real life context. As such, they are characterized by analytical generalizations, rather than statistical generalizations – i.e., they are not to be understood in terms of samples, but in terms of analyses and comparison of cases [3]. In [8], we describe our process of defining the case study from preparation to the evidence chain and evidence trail.

The interviews we conducted are semi-structured and we used a pre-designed questionnaire with an open-ended set of questions. We follow a conversation-based approach rather than a question and answer form. The questionnaire was divided into four parts: part 1 focuses on the problem domain of the architect, etc; part 2 captures how architects elicit, view and manage security requirements in practice; part 3 focuses on the architect's views on the meaning of architecture and how requirements are transformed into architectures and implemented; and part 4 is concerned about how architecture affects, and is affected by, evolution.

Our study of security architects has taken place in the context of a larger study of software architects in general [8] in which a number of our subjects were either security architects by title or were involved in security issues as part of their architectural practice. One of the security architects has been working in computer security and data privacy for the last 15 years. Another architect has been a security architect for the last 10 years and his job entails both product architecture and solutions architecture. A third has been primarily involved for the last three years in building security models in software for the auto industry and had previous experience working at a major internet search engine/web portal. In addition to our general architecture interviews, we also conducted follow up interviews with these security architects to collect more detailed data for further analysis. To support this part of our study, we specialized the questionnaire to focus more directly on the security aspects of requirements and architecture (see the Appendix).

Our research has shown that security is often compromised by circumventing security mechanisms within the architecture. These flaws in the design of security critical systems may become visible only after years of use. Due to the rapidly increasing severity of software security threats, it is imperative that security concerns be addressed in the early stages of the software development lifecycle. Security issues must be addressed both in requirements and architecture with bounded investments in time and costs.

In this paper, we describe how practicing architects view and manage security requirements and architectural designs for secure systems. Furthermore, we delineate what characteristics and skills security architects should have to successfully manage and implement security requirements. We believe understanding current practice is a necessary step in providing the foundation for repeatable processes using effective techniques, methods and architectural structures to satisfy security requirements.

2 Security and Software Architecture – Multiple Case Study

In this section, we provide our insights into security issues based on the data collected from the semi-structured interviews with security architects. We present selected interview data that reflect how these architects view security and address related issues in practice.

We present our data and analysis in five parts. First, we describe the basic perspectives on security requirements that caught our attention during the study. Next, we discuss how the architects deal with security issues and manage security requirements in the process of architectural design. Then, we illustrate the critical characteristics that security architects possess in common, particularly the skills required for doing the job effectively. Then, we discuss the relationship between security and evolution. And finally, we provide some observations on how security architects solve problems.

2.1 Basic Perspectives

We summarize our interview data in the following aspects: problem characteristics, maturity and stability of the domain, and special issues of concern.

2.1.1 Problem Characteristics

Our subjects suggest that security issues in software systems typically surface in four domains: communication, data access/exchange, operating system, and cryptography. However, specific security requirements of a system are usually determined with respect to the business context and user preferences. The definitions of security can vary from 'a guard at every physical door' to comprehensive data confidentiality, integrity and availability requirements.

The interview data uncovered three levels in security related problems. The basic level concerns with authentication and protection of discrete resources against unauthorized access. Solutions at this level are well established and widely applied.

The second level is the protection of confidentiality in the presence of inference. Solutions for these problems are difficult but can often be found.

"[T]he canonical example … is protecting confidentiality of information in a relational database… because a relational database is basically an engine for developing lots and lots of aliases for the same information. [W]hen you get a new reference that you haven't seen before, it is difficult to tell whether that reference applies to information that you have already protected in some way… therefore, it is difficult to apply the correct policy. So, inference is known to be a hard problem [in] preventing unauthorized inference from a string of queries to a database."

The third level is on intellectual property related issues. General solutions are considered impossible, and mitigation strategies are used instead.

2.1.2 Maturity and Stability

One subject indicates that security as a domain is immature and unstable. The most notable evidence for him is the following:

"You read the newspaper. There is no possibility I am going to be out of job anytime soon. By my definition, it means that it's not a mature domain."

The reason for this instability may be because there are no commonly accepted metrics and much of what is done is based on intuition and experience.

Another subject argues that part of the secure domain is mature and has well-established solutions despite the other parts of that domain which are still immature.

"Security is a huge topic, and there [are] a lot of things which are immature in security. Like Federation [Identity Management] is very, very immature. … There's both maturity and immaturity within the space. … Things like the encryption algorithm, pretty mature; you know how to do it. [For] user-name protection, figure out what level of protection you need [and] what are you protecting against, you have… different protocols [to solve them and] each one has pros and cons."

Some fundamentally flawed ideas can succeed in the marketplace, and that is another indication of the domain's immaturity level.

"Things like electronic wallets make it easy for the merchant, but they are fundamentally a bad idea because it allows you to easily give private information to people you don't really know who don't need that information… Although [it is] very successful from the popular money making standpoint."

2.1.3 Special Issues

Composition is particularly difficult in engineering secure systems because emergent properties can cause serious problems while integrating two or more components. It is possible for individual components to possess specific security features, but the combination of these components may violate the desired level of security.

"[It is unfortunately the case that lots of security problems [s] do not compose in a mathematical sense. If X has security property 1, and Y has security [property] 1, [then] X+Y does not [necessarily] have security property 1."

"The problem is that if I got this '-ility' and I have got five things I can do about it, if I pick one of these mechanisms either it is going to be inconsistent with one of the mechanisms for this other –ility... or it may open things up, for example if I am doing testability here and I am adding test interface and things like that to make it more observable and controllable, that's exactly what security doesn't want. ... So how to pick the right mechanism is not widely known in industry."

One of our subjects suggests that a framework approach does not work well for building secure systems due to an undesired amount of generality. Specifically, he believes that although it is theoretically possible to have a formally defined framework that provides both the required specificity in security and the necessary abstraction for general applications, practically, it is impossible.

Another issue is that security awareness and understanding is low among the public which leads to the infeasible requirements and difficulty in justifying the security levels of systems. However, damage caused by security holes in existing systems can help to justify security needs in new systems.

In the next section, we present how our subjects deal with security issues in architecting software systems.

2.2 Dealing with Security Issues

We present our data on managing security requirements from three perspectives: establishing security requirements, prioritizing security requirements and architecting security requirements.

2.2.1 Establishing Security Requirements

Some key characteristics of security are fundamentally different from other requirements. Performance, for example, is a requirement that can be tuned incrementally in a variety of ways (typically to increase performance). Security, on the other hand, is not something that can be adjusted in the same way.

Reliability and safety are typically implemented with the expectation that there will be random component failures and accidents. In contrast, security is implemented with the expectation that failures are (most likely) caused intentionally by capable and motivated adversaries.

Hence, it is important to capture the malicious intentions, motivations and capabilities of attackers in the security domain. Threat models are used for these considerations. Threat modeling is a security analysis methodology that can be used to identify risks and guide subsequent design, coding, and testing decisions.

In general, threat modeling involves decomposing the system, identifying its key assets or components, and specifying and categorizing the threats to each asset or component.

"In security the primary problem is the existence of a capable and motivated adversary who wants the system to fail. This [property] makes security architecture different from any other discipline."

"Security architecture is fundamentally based on the idea of threat models. You have to start off with the model of the threats you are trying to defend against, and if the threat model incorporates the possibility of physical attacks, then you have to pay attention to physical attacks… In fact, threat analysis do include an element of characterizing adversaries in terms of capability, motivation, and desired outcome"

One subject explained the importance of threat models by pointing out the difficulties encountered when people try to analyze security requirements with use case modeling.

"[U]se cases tend to be more functional than quality oriented which drives you to only have the one kind of requirement but not the other kind… But then the other thing is that they tend to concentrate too much on 'This is what the system shall do' and the actors are the normal people interacting with the system. And they therefore ignore the single most important actor in that kind of situation: the attacker…"

Another subject also commented that security problems cannot be solved by ontology-based approaches since those are generally very inadequate. A way to approach it is through generalization over past attacks and experiences.

"Ontology is the enemy for security. Because as soon as [you] put together the ontology, by definition that defines everything there is and therefore everything else is unthinkable… Unthinkable stuff [is] really bad."

"[T]he way security people learn how to think about things… is by studying past failures. … You look at the collection of successful attacks on past systems [and] make sure that none of those work on the current system. [T]hen… try to identify patterns… and abstract types of things... that are not specifically the same… but have some of the same ideas. And then you try all of those. [I]f you really hit a dead end and want to break the system, you take somebody who doesn't have any assumptions. [They will be] able to enter into the whole thing because [they will not try to think like the designers]. Sometimes it is very important to be able to do that when you are designing security systems."

Sometimes the requirements and the problems are not presented in the right form. In such cases, it is necessary to discover the shape of the problem and identify the correct form of the requirements in order to proceed.

"If … the customer is putting a twist [on the problem] we try to shift the customer or the requirements to the right direction, saying, 'maybe you should think this way or maybe you can do the same thing by an alternate way'. Because customers are set in their ways and they don't want to change, so they want to do what they have been doing. … Education helps [at] certain times. … People seem to have [a] narrow focus sometimes [and] sometimes you have to broaden them."

"What [is the] business problem [that] you are trying to solve? Don't come to me with 'we have to upload this spreadsheet'. [Tell me] what are you trying to solve; what are [you] trying to do. And when [we] don't do that [we] just end up in a rat hole."

There can be situations where it is not possible to accommodate all the requirements at the same time. In such cases, architects do the best they can by assessing the pros and cons.

"When a requirement is outright impossible, we say that's impossible. … And sometimes we're told to do it anyway. Digital Rights Management is the perfect example… it's just demonstrably the case that you can't do Digital Rights Management to meet a set of requirements that people in the entertainment industry want. … Nevertheless, we enable our systems for DRM and build DRM mechanisms anyway. Because people say they want them. … It filters out a number of dumb attackers. The smart attackers get in and copy things anyway."

"There's not a luxury to do everything that we want to do or everything which is ideal. You have to go with the requirements, go with the political nature, the business requirements, the funding aspects. … So you do a pros and cons and decide what is the best..."

On top of the intellectual aspects, the physical aspects of security also play an important role. It is important for the architect to look at the entire system and not just a particular set of technologies.

"You really can't afford not to pay attention to physical aspects of things. It's sort of like designing the pressure vessel of a submarine; it only has to leak in one place for you to have trouble. And if that is in the physical infrastructure then that's just as bad a problem as if you have screwed up some conceptual thing. [You] have to pay attention to every aspect of how you might attack a system."

"Then there [are the] physical [aspects we need to pay attention to]. How's our data safe? … What happens if a tornado hits? How secure is that data in any kind of disaster? ... That's at the macro level. Then you get into the application, and they're very sensitive about different [roles] – people that are designing ad-drawings don't need to be looking at the financial information. … [S]ecurity … cuts across every type of object in the system."

"[Y]ou have things that you do in hardware for security, in the software for security … in the data for security, but you also have to deal with physical security, you have to deal with the security of your staff and the people who are interacting with your systems. So the more you get into this, the more you realize it's a larger, more complex issue, and just looking at one tiny little piece of the problem leads you to a false sense of security that you've handled it when you haven't."

2.2.2 Prioritizing Security Requirements

Having established a set of requirements, two situations often arise: (i) there are conflicting requirements, and (ii) the cost of building a system that satisfies all the requirements is too high. Hence, there is a need to prioritize the requirements to establish which are critical and which are subordinate.

Priorities could be based on the likelihood of the risk, cost/benefit analysis, and specific areas of concern for the stakeholder. It requires the understanding of the capabilities, resources, motivation, risk tolerance and level of access of the likely adversaries. Such an understanding helps to elicit hidden requirements and to provide a strong defense and effective recovery mechanism with a modest cost.

The actual prioritizations are typically done through negotiation. For functional requirements, choices can be made through prioritization of features.

"[Y]ou get a good feeling for the weight of the requirement… You can generally tell, by the discussion, what's important to them especially if you push back on something. [If] it's really important to them, you'll start getting the messages, the body language, [that they are] not comfortable…"

It is not practical, and usually impossible, to achieve complete security. Not only is it expensive, it is unrealistic because there are always some aspects not anticipated. Vulnerabilities can be found in even carefully designed products. New attacks are constantly invented.

Instead, security levels are defined with respect to specific goals; they can either be achieved or not. Thus, security requirements have to take precedence without giving any concessions. Aside from setting the level of security that is acceptable, there is not much to be adjustable. In other words, acceptable risk mitigation is attainable even though security is not achievable at large.

"The attack succeeds or it fails. So [security] is a difficult property to subject to engineering tradeoffs. But you can… decide in advance that the system has to impose some specified work factor on the adversary and have that as a design goal."

"You can't... really continuously tune your level of security. And this of course pisses off all the other designers in the organization because they are all sitting around saying, 'Well, you know we can tradeoff a few clock cycles here for a better user interface here or something like that' and the security guy is just sitting in the room and everybody else says, 'So what do you have to offer?' And the security guy says, 'Nothing, you have to do it my way.' "

However, some factors can trump security, as in the case of legal issues.

"We had a certain product and it failed because of the… legal ramification of issuing a certificate. … [I]f I am issuing a certificate to [you] then I am accountable for it if [you do] any fraud with that certificate. … There is a liability [issue] associated with that. So we spend tons of money on the product, and it was failed."

2.2.3 Architecting Security Requirements

All our subjects expressed that it is important to consider all requirements in support of security requirements while building secure systems. We believe it is because security requirements are highly integrated with other requirements, and must be considered from the very beginning of the development cycle.

We observe that there is a slight disagreement on how our subjects categorize the supporting requirements[1]. For example, some of them categorize performance to be functional while others categorize it to be non-functional. Whether functional or non-functional, it is agreed that a set of supporting requirements is needed in building successful secure systems. These include performance, scalability, interoperability, availability, manageability and maintainability.

"Typically in security the functional requirements mostly have to do with interoperability mechanism and with manageability. So it's a functional requirement that my VPN client has to be able to talk this bizarre protocol that is spoken by the mutant VPN server."

"[A] system administrator [needs] to [be able to] update the access of everybody in department X by running a script overnight. [This implies] there's got to be an API level interface for the security management system and it's got to have certain kinds of authentication and authorization functions so that we can run it safely."

"[P]erformance is frequently a functional requirement; you are not allowed to slow down."

In building secure systems, both functional and non-functional requirements play a critical role in all phases.

"The functional aspects are something like the core aspects of the product… Non-functional are performance and scalability... [W]e try to give functional requirements more importance because that's what is seen [and] marketed. … But non-functional requirements are worked in with that because what's the point in releasing a product if it doesn't scale beyond 100 users, or… doesn't perform. So it goes down [to] all phases. Whether it is architecture [or] design, you combine those two things at all points of time and work towards a cohesive architecture."

"I don't really feel there's that big a difference between non-functional and functional. It is just a requirement and somehow you've got to accommodate it."

Security requirements are defined relative to specific goals capturing known vulnerabilities. These goals must be accounted for in designing the system structure. Given that security is embedded in the system structure, it cannot be altered easily.

"[Y]ou can decide in advance that the system has to impose some specified work factor on the adversary and have that as a design goal. [Then,] you basically have to hit that mark or do better. You can't… really continuously tune your level of security."

"We have a great deal of flexibility to adjust and replace mechanisms. [For example] we can add stronger cryptography… on the wire protocols... What it's much less easy to do is to change the basic structure of the system in a way that has an impact on security. Sometimes we end up having to do that, and it's a lot of work."

Unfortunately, security requirements are often done separately from the system requirements. Typically, system requirements are done first and security is added as an afterthought. This often leads to significant changes to the architecture.

"One of the number one problems that I often see, and especially true in security and safety, is you have got a security team over here and safety team over here [that] never talk to the requirements people [and] rarely talk to the architectural people, at least upfront. ... You have the requirements team doing their requirements, they don't understand these guys and these guys haven't fed their stuff into here. … And so the actual real honest requirements end up in the requirements spec, which drives the architecture. And then later on, what happens is these guys come in here and say, 'You forgot about us'. … And then they try to slather it on the outside. Well you can't add some of these major things to a pre-existing architecture by just adding it on. Now that doesn't necessarily mean that it had to be there from scratch. What it does mean is you have to have some significant changes to the architecture. … Which is why it is so critical to make sure that all of the -ilities are thought of up front, and all of the quality requirements are fed into the requirements spec."

However, one of our subjects does not believe in the importance of security being an integral part of the design rather than being added in as an after thought. According to him, it only means that the security should be the first priority requirement and all the other requirements should change in order to accommodate security. And this is important only when designing control mechanisms for nuclear missiles and not when designing commercial operating systems.

"Now there are lots of systems which have been built with security being added which are out there functioning in the world today. The telephone network is a good example okay…. Well the fact that security is added on to Windows XP is not what makes it insecure. The thing that it makes it insecure is that it is huge and a mess".

Security goals must be designed with a far-seeing vision; the lack of that will lead to failures.

"[T]here was a cellular phone protocol that [depended] for its security on the assumption that bad guys

[1] We believe that the disagreement is due to the necessary reification from non-functional requirements to functional structures.

can't put up a tower … [T]hat's [definitely] not a good assumption… [T]hat protocol does not exist in that form anymore as it turns out it's not that hard to put up something that looks to a cell phone handset as if it were a tower."

Even though security is an integral part of the system, we must be able to address the issues of modularity and externalization. Security needs to be configurable for different security levels, and it must be replaceable depending on the context without breaking the system.

"[It is ideal that] in the production environment with full on security, various layers of security can be turned off [and] the system still functions. [Also,] you can layer more and more security if you want."

Security requirements often restrict the choices of other requirements. There is an obvious tradeoff between security and performance because extra operations are required in more secure systems.

"[It] tends to be the case that security trades off against performance, … because as you harden the interfaces of the components and isolate it more and more, you make it more difficult to cross the boundary between the non-secure portion of the system… and the part of the system that enforces security."

In summary, we have seen how architects establish, prioritize and architect security requirements in practice. We observe that (i) architects must explicitly give considerations to security issues at the requirements level; (ii) security concerns should not be an add-on but an integral part of the requirements; (iii) emergent security properties should be taken into account explicitly in terms of what should be protected, from whom, and for how long.

2.3 Characteristics of Architects

During the interviews, our subjects discussed at length the characteristics required of architects in designing highly secure systems. We begin with the general characteristics of architects, then narrow down to particular skills required of security architects.

It is noted that breadth is one of the most important characteristics. Architects must be generalists so that they understand all the parts of the system and do not focus on just a single aspect.

"Breadth is very important, to not be just focused only on security, but being able to know the other aspects of software engineering, whether it is performance, ... hardware, … application [or] whatever it is."

"Generally, security people are generalists rather than specialists. They have [to] understand a lot about different parts of the system and how they work, [and] enough about each of those parts of the system so that they can figure out [where] vulnerabilities [can surface]."

"The idea that you can be a software architect and know nothing about hardware or the rest of the system I think is a complete misnomer… Safety, Security, Reliability, Robustness, Availability, all of those are system characteristics, not software characteristics. [So an architect has] to look at the hardware, the software, and the data components as well as procedural components and… the human beings that are involved."

Some of the essential personality traits of an architect are persistence and persuasiveness. However, if these fail to influence the team, the architect may need to act the role of a benevolent dictator to push the progress forward. In general, an architect needs to possess strong technical, people, leadership and communication skills.

"[G]ood technical background is the key. Good people, good leadership skills are very important, because you are leading a team … a set of people to believ[e] what you think is right, You got to be able to convince. … If you are a dictator, that's bad. You got to be a team player … have some level of leadership skills and be able to listen. Because if you don't listen, then you will be going to your tunnel vision and do what you think is right, as opposed to what is required for the job."

"[An architect needs to be a] politician, diplomat, nursery attendant, business liaison. You have to be [a] benevolent dictator. [You] have to be technically savvy, but more so, sound. ... I don't think you need to know the latest version of the latest spec [but] good sound design principles and… learn [quickly]."

There is some debate over whether good architects are 'born-to-be' or grow out of extensive training and coaching. One subject suggests that one can become a good architect through mentoring, but basic talent is still necessary.

"Generally… majority of the security people are born, but then after that they have to be trained. So it's a select from a population that have the right characteristics"

However, another subject suggested that they can be trained and need not to be born with such skills.

"I think anybody can do anything in life if you work hard. That's my fundamental belief. Having said that… Yes, some people just don't get it. … [T]ypically when we try to grow somebody, the biggest problem we face is they are very focused in what they know, and they are not easy to learn the rest of the concepts

To manage security requirements effectively, architects must be able to adopt the mentality of the attackers.

"The most important qualification to be a security architect is [being] able to think like the bad guys. … If you do not have an element of ... malice [or] at least an appreciation of the beauty of malice … you are just going to fail."

Typically, people react to new attacks with one of three attitudes: (i) reluctant and annoyed, (ii) excited and motivated, (iii) prohibitive and dismissive. Only those who welcome the challenge, as in the second case, are appropriate for a security architect.

"[The first says] 'that is really annoying, I can't get my job done'. They are fine, they are probably not dangerous; you could use them to test things or something. [The next says] 'oh that is really neat! I wonder how he did that'. Those will likely be good security people. [The last says] 'Well you know, nobody should be allowed to do that'. They have to be kept far away from security. They have totally the wrong attitude, they don't get the problem, [and] they will never be able to think that way."

2.4 Security and Evolution

Evolution is generally driven by changing requirements, usually the result of demands for new functionality, in most software systems.

However, in security, evolution is often driven by changing contexts while the requirements have remained the same. As one security architect pointed out, some of the security problems in Windows is a result of people using Windows in an environment that it was not initially designed for.

"[Windows] was designed to run a personal computer. ... Nobody knew at the time Windows 3.1 was designed that it was even going to be networked. ... Windows was designed to operate as an isolated personal computer and have a nice interface and be friendly and everything. And it was designed for other good characteristics that continue to haunt it from a security point of view today. So for example it was designed almost from the very beginning to accommodate people with vision and hearing defects, and what it meant was you could stick a device driver straight into the brain of the operating system that would allow you to operate a Braille device or allow you to operate something that wasn't a keyboard. And therefore, there is like no way to assure that you are actually talking to the real user."

Sometimes the context change can be caused by something as simple as an increase in computing power. Encryption algorithms are a good example.

"When we make a product we assume that algorithms are good for a certain duration of time. ... Most of the crypto algorithms have been there since the 70's, right? Now at that time the computing power was not even a millionth of what we have right now. Your desktop is probably more powerful than mainframes at that time."

2.5 Problem Solving

Traditionally, activities are described as a set of well-defined goals and plans that are determined a priori. This is known as a plan-based approach. In her book, Suchman offers an alternative theory, that of situated actions [5]. She suggests that "*every course of action depends in essential ways upon its material and social circumstances*". That is human cognition and subsequent action (in this case, problem solving) is an emergent property of the moment-by-moment interaction of an individual with the physical and social environment. However, this theory does not imply that plans cannot or do not exist.

In our study, we find an analogous duality in our architects in their responses to unanticipated problems. On the one hand we find architects who fall back on organizations' well-defined processes when so confronted. Indeed the majority of the subjects interviewed when asked about certain kinds of abnormal situations, referred to "falling back on the defined process" as the means by which these situations would be handled. A smaller set of architects, on the other hand, exhibited a "situated action" response in which what they would do was dependent on the situation and their experience. In this latter case, experience, not process, formed the basis of their architect response in solving these problems.

3 Validity Issues

In this section, we discuss three validity issues that are important to empirical studies [4]: construct [2], internal and external validity.

There are two perspectives that contribute to the construct validity in this case study. One is on the coverage of the questionnaire, and the other is on the abstractions employed. The goal in designing the questionnaire is to be both thorough and broad.

The general questionnaire was initially drafted by one author based on an initial brainstorming session. It then underwent a number of reviews by each author. Reviews were carried out among the authors after each interview session and revisions were applied whenever necessary. While the questionnaire is not focused specifically on security, all of the quotes that we have used in this paper were taken from parts of the interviews and are focused on security.

The follow on security interviews were based on questionnaire based on the general one, but tailored specifically for security architects. The development, review and evolution cycle followed was that we used for the general one.

Semi-structured interviews may suffer from the problem of leading our subjects. This may lead to internal validity issues making the data collected less objective than it should be. However, we know where this occurs and can mitigate that problem by being careful in using the results in these contexts. Moreover, we have all interviews transcribed, and when we spot that there is leading, we will use other data instead or note the context of the subjects' comments.

Two of our security architects are from the same international organization. We recognize that there may be some unintentional bias introduced by a shared company culture that may lead to external validity issues. However, these subjects are from different levels of the corporate hierarchy. Moreover, this work is ongoing, and we plan to choose subjects that are more diverse in the future.

4 Conclusions

We summarize lessons learned from the interview data as well as our own remedies as follows:

Lesson 1: *Building secure systems and managing security requirements effectively depends on established software engineering principles and practices. Though it is not always achieved in practice, we believe well-engineered systems should be the foundation for achieving security goals.*

The literature has shown similar evidence. In his keynote speech, Wolf pointed out that "*Security engineering is a technical field dependant upon methods, tools, and models for requirement analysis, design analysis and implementation analysis*" and concluded that security engineering really is good software engineering [10].

Lesson 2: *Security is a critical domain that requires highly specialized treatment. It relates to a system's complexity and connectivity, and thus, touches all aspects of engineering. The pros and cons of various security strategies must be weighed during system architecting and planning activities.*

The software engineering research community is starting to take notice of the security domain and its unique domain properties. As a result, new techniques, methods and technologies are emerging. One noticeable contribution is the anti-goal models introduced by van Lamsweerde et al. in capturing malicious obstacles [6][7].

Lesson 3: *Good security begins with an awareness of security requirements and implementation of security features in the architecture of the system. To achieve this, security must be included in the design goals right from the beginning. It should be treated as a required property that must be an integral part of the system since it is neither tunable nor imposable later on.*

Empirical studies are needed to determine which practices are most effective. However, very little empirical proof exists for many technical practices used today for producing secure software. In [9], the authors present an empirical view on security engineering practices. The results are based on the observations made by three information security practitioners. They describe that different application domains have different security needs which should be frequently updated because the world is changing and the old security architectures would no longer work in the new environments. This need to evolve security architectures in order to keep up with changing contexts mirrors what we observed in our research and discussed in section 2.4.

Lesson 4: *Understanding security problems is an ongoing challenge. Current security problems are different from the past or the future. It is important that architects understand different threat models and continuously update on the newest solutions to prevent new attacks.*

As Wolf pointed out, "*software threat analysis is a young art*" and existing models do not adequately support the analysis needed [10]. There is much work to be done in the security domain.

Lesson 5: *There is no universal definition for the term security architecture. In the absence of agreement, the first thing a security architect usually does is to describe what relationships are secure and what are not.*

In general, the security architecture must (i) facilitate proper and efficient security identification, authentication and authorization in response to the access and use of information resources; (ii) provide a modular approach to authentication, authorization and accountability; (iii) ensure security requirements and associated risks are adequately evaluated when preparing to the support different needs of an organization; and (iv) be flexible enough to support integration of new technologies while maintaining appropriate security protection. The evidence in our case study supports this position.

In conclusion, we believe security specialists should employ established software engineering principles and practices to their advantage, and software engineers must recognize the unique aspects of the security domain and continue to provide and to apply appropriate methods to attain a higher level of software security. It is important to raise awareness not only among the users but also among the administrative staff about the importance of security and security architectures.

Several issues have surfaced in our case study, which require further research: (i) how security architects should be involved in requirements elicitation and negotiation; (ii) what frequently occurring problem and requirement patterns can be; (iii) what specific modeling tools/methods are needed for capturing security requirements; (iv) what evaluation techniques are required to assess security levels in architecture; and (v) how conflicts between security requirements should be resolved. In cases where more evidence is required, we will either follow up with the current subjects or conduct new interviews.

About the Author

Vidya Lakshminarayanan is a MS/PhD student in Electrical & Computer Engineering at University of Texas at Austin. Since summer 2004 she has spent her time in working on the project on “Transforming Requirements Specifications into Architectural Prescriptions”. Currently, she is working as a co-op engineer at Advanced Micro Devices to gain experience and a better understanding of the actual practice.

WenQian (Wendy) Liu is a Ph.D. candidate in Computer Science at the University of Toronto and an IBM CAS Student. She received her Hon. B.Sc. in Computer Science and Mathematics and M.Sc. in Computer Science both from the University of Toronto. Her primary research interests are software architectural design and requirements engineering.

Charles L. Chen is a MS/PhD student in Electrical and Computer Engineering at the University of Texas at Austin. He obtained his BS degree from the Department of Electrical and Computer Engineering from the same institution in 2006.His research interests includes software architecture, software evolution, computer security, and computer networks.

Dewayne E Perry is Professor and Motorola Regents in Software Engineering in Electrical & Computer Engineering at The University of Texas at Austin. His research interests include software architecture, software evolution and empirical studies in software engineering. He is a member of ACM and IEEE Computer Society

Acknowledgements

We thank all of our anonymous interviewees for their participation and contribution. This research was supported in part by NSF CISE Grant CCR-0306613 and IBM CAS Fellowship.

5 References

[1] R. Anderson. *“Security Engineering - A Guide to Building Dependable Distributed Systems”*. John Wiley & Sons, Inc. 2001.

[2] D. E. Perry. “An Empirical Approach to Design Metrics and Judgments”. In *New Vision for Software Design and Production Workshop*. Vandebilt University, Dec 2001.

[3] Robert K. Yin, *“Case Study Research: Design and Methods”*, 3/e. Thousand Oaks, CA: Sage Publications, 2002.

[4] R. Rosenthal and R. L. Rosnow. *“Essentials of Behavioral Research: Methods and Data Analysis”*. McGraw Hill, second edition, 1991.

[5] L. A. Suchman. “*Plans and Situated Actions*”. Cambridge: Cambridge University Press, 1987.

[6] A. van Lamsweerde. “Elaborating Security Requirements by Construction of Intentional Anti-Models”. *In Proc. ICSE’04: 26th International Conference on Software Engineering, 2004.*

[7] A. van Lamsweerde et al. “From System Goals to Intruder Anti-Goals: Attack Generation and Resolution for Security Requirements Engineering”. In Proc. *Requirements for High Assurance Systems Workshop (RHAS'03), 2003.*

[8] W. Liu, C. L. Chen, V.Lakshminarayanan, D.E. Perry, “A Design for Evidence-based Software Architecture Research”, *Workshop on REBSE'2005*, ICSE May 2005.

[9] R.B. Vaughn, R. Henning, K. Fox. “An Empirical Study of Industrial Security-Engineering Practices”. In *Proc. Journal of Systems and Software, April 2002*.

[10] A. L. Wolf. “Is Security Engineering Really Just Good Software Engineering?” In *Proc. of the Foundations of Software Engineering, Keynote speech. 2004*.

Appendix

Interview Questionnaire

1. Goals

To develop a deep understanding of how architects view, manage and architect security requirements in practice
- What they think the key aspects of security domain are
-How do they establish, prioritize and architect security requirements
- How do they transform these requirements into security architecture
- How they relate security requirements to general requirements, detailed design and implementation.

2. Privacy issues

- Anonymity (of both architects and company) is guaranteed, unless explicit permission is provided by the interviewee and/or his/her company.
- Access to the recording data is strictly limited to researchers involved in the project and no one else.
- Company confidentiality will be maintained on architecting process issues if requested.
- Company confidentiality will be maintained on IP issues, unless explicit permission is provided by the interviewee and/or his/her company.

3. Definitions

- Within the context of the system (or subsystem), architectural design is one that is the most abstract depiction of the system that enables reasoning about critical requirements and constrains all subsequent refinements. [Clements et al 2001]
- A good architecture is one that can be built given the current available resources and uses less resource during maintenance and easily maintainable.
- Architectural drift refers to an evolved system that is insensitive to the original architecture.
- Architectural erosion refers to an evolved system that has taken away the load bearing structures from and/or violates the architecture.

4. Questionnaire

- Note that the questions below may have overlaps, please feel free to repeat your answer if appropriate.
- The order of the questions can be altered according to the architect’s preference.
Questions may also be skipped.

A. Problem Domain

A1. Describe the problem domain(s) that you have been working in
- General domain
- Specific domain examples where security was a critical concern
- Were there any non-software aspects of the system that played a role in security in this case?
- Is security the primary focus of the system?
- Do you know of any successful projects with very poor security? What are the various reasons?
- Do you know of any unsuccessful projects with very good security? What are the various reasons?

A2. What are the key aspects of the security domain?
- General goals of security
- Problem characteristics
- Can you give some examples of kinds of problems that you encounter?
- Are there any techniques for recognizing these different kinds of problems? How do you react to problems that are completely new?

- What kinds of effects do security issues have on requirements and architecture?
- Maturity vs. immaturity
  Mature – established business process; well-defined theory, processing and expected behaviors; well understood objects
- Stable vs. unstable
  Degree or constancy of change during and after development
- Sources of difficulties and obstacles

B. Requirements

This section is to capture how security architects elicit, view and manage security requirements in practice

B1. What specific modeling tools/methods are needed for capturing security requirements? Do you transform requirements into a specific security level goal? And if so, how?

B2. Security is not the only requirement in software development. What are the other supporting functional and non-functional requirements? How do you prioritize these requirements? In case of conflicts with other requirements, how do you handle security requirements? Are there conflicts between security requirements? If there are, how are they handled?

B3. How is security different from other non-functional requirements?

B4. Should the security architects be involved in requirements elicitation and negotiation?

B5. How do you handle security requirements that are very complex or costly? What about the requirements those are outright impossible?

C. Architecture, Implementation and Design

This section is to capture the architect's general understanding on the meaning of security architecture, how requirements are transformed into security architecture and how they are implemented and designed in practice.

C1. How do you define security architecture?
- How detailed should the architecture be?
- How is the architecture communicated to the stakeholders?
- What are the driving forces of the architecture?
- What should the architectural representation include?

C2. What are the critical characteristics of a security architect?
- Can these architects be trained or must they be born with such skills?
- Other than these critical characteristics, what are some general characteristics of security architects?

C3. Are there any specific rules of thumb for developing secure software?

C4. What factors make security hard to architect/implement?

C5. How do you recognize which parts of the software need to be secure?

C6. Is threat analysis/threat modeling important? How is it defined and practiced? Does this help to minimize the vulnerability?

C7. How does the problem structure affect your architecture?

C8. In your opinion, are there any relationships between the problem structure and a good architecture? What would it be?
- Are the requirements sufficient to give you a good understanding of the problem structure? Or do you need additional domain knowledge to understand the problem structure?
- Do you use that understanding of the problem structure in structuring the architecture?

C9. Are there standard architectural structures that you use for security? Architectural transformations? Patterns?
- How and when do you apply any of these?
- Is there a problem with composition of security components? Is so, how do you overcome it?

C10. How do you evaluate the architecture? Are there any tools/methods? What are your criteria?
- What evaluation techniques are needed to assess security levels in the architecture?

D. Evolution

D1. How do new requirements affect the architecture after the system is built? Especially security requirements.

D2. How do you handle continuous requirements change?

D3. How does the architecture evolve during the system's lifetime in response to changing requirements?
- How do you deal with architectural drift or erosion?

D4. What measures do you take while designing the architecture to minimize the impact of change?
- How do you identify and understand the various effects of requirements changes on the architecture?
- What about the effects of architectural changes on design and implementation?
- Has maintaining architectures been a useful task?

D5. How do you reuse security architecture?
- How do you make architecture reusable?
- Are you usually concerned with reusability while designing?
- Do you find common parts that you can reuse?

D6. Should security be an integral part of the system? What are the difficulties encountered if security is added on as an afterthought?

D7.How do you anticipate future security requirements, both in terms of requirements that have not been requested yet and in terms of brand new attacks? How much room do you leave in the design for the requirement changes?

E. Comments

E1. What do you think of the questions?
- Are they relevant or interesting?
- Do they help you to think differently about architecture than you did before?
- Do you find this interview a useful exercise? - Why and how?
- What were your expectations of the interview?
- What values are you looking for from this exercise?

E2. Do you recommend any other security architects who might have an interest in this work?
- Names and contact info.

E3. Any feedback, future thoughts, suggestions and contacts, e-mail: vidya@ece.utexas.edu, perry@ece.utexas.edu.