A Case Study of Architecting Security Requirements in Practice: Initial Analysis

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 following initial results of multiple cases of practicing security architects: key aspects in security requirements, essential characteristics of security architects and critical issues in managing security requirements. We conclude with 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][4][5][6].

This preliminary 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 several 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. It is important to address security issues within both requirements and architecture with bounded investments in time and costs.

Our general research goal is to understand how architects view, manage and architect requirements in practice. We take an empirical based approach and use an interview based case study methodology to carry out our investigations. This study involves a series of carefully designed semi-structured interviews. 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.

Four of the ten architects we have interviewed classified themselves either as security architects or as architects heavily involved with security problems. Thus, we have analyzed these interviews in depth, distilling critical comments and perceptions about how security requirements are managed and architected in practice.

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

The rest of the paper is organized as follows. In section 2, we present the results of our study in three parts: (i) delineating key aspects in security, (ii) presenting the characteristics of security architects and (iii) discussing the critical issues in managing security requirements. In addition, we briefly discuss validity issues in our case study. In section 3, we summarize our findings, draw our conclusions, relate our conclusions to current work and indicate areas of future research.

2 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 how they architect such requirements in practice. To preserve the anonymity of our subjects, we will use *A*, *B*, *C* and *D* instead of their real names in all the quotes. Subject *A* is a security architect who has been working in computer security and data privacy for the last 15 years. Subject *B* has been a security architect for the last 10 years and his job entails both product architecture and solutions architecture. Subject *C* has been primarily involved for the last three years in building security models in software for the auto industry. Subject *D* has been a systems architect for the last 15 years and has worked on systems where security was the primary concern.

We present our data and analysis in three parts. First, we describe the key aspects in security. Next, we illustrate some of the critical characteristics of security architects, particularly the skills required for managing security requirements effectively in addressing these key issues. Last, we present how our subjects architect security requirements in practice.

Attributions are provided in the form of quotations or summaries whenever the subject has a remark on the topic. The lack of such indicates that no relevant discussion was found in the interview data. Further, our editorial additions to the quotations for both grammar and clarification are enclosed in square brackets.

2.1 Key Aspects in Security

We summarize our interview data with respect to the following aspects of the security domain: problem characteristics, maturity and stability, and sources of difficulties and obstacles.

Problem Characteristics

Subject *A* suggested that security issues have typically surfaced in three areas of software engineering: communication, operating system and cryptography. The specific security requirements of a particular installation can only be determined after careful consideration of the business context and user preferences. These definitions vary from 'a guard at every physical door' to comprehensive data confidentiality, integrity and availability requirements.

Furthermore, he distinguished three types of security problems. The first type is authentication and protection of discrete resources, for which solutions are well established.

"[A] fairly large collection of security problems, including authenticating principles and protecting discrete resources against unauthorized access and protecting content of communications at least for some time interval, [have] well-established solutions that are often fairly easy to apply and that work reasonably well." [Subject A]

The second type is the protection of confidentiality in the presence of inference, for which solutions are difficult.

"[T]he canonical example of a difficult security problem 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 - preventing unauthorized inference from a string of queries to a database." [Subject A]

The third type is intellectual property related issues, for which solutions are impossible.

"[T]here is a third set of problems for which we keep trying to solve, but for which solutions are actually probably impossible. I include Digital Rights Management among this set of problems. Not everyone agrees with me." [Subject A]

Subject *B* also agreed that encryption algorithms have well-established and easy-to-apply solutions. In addition, he pointed out that privacy in transit (i.e. protecting content away from the source) is a difficult issue.

"[An] FBI personnel going to CIA… should be able to read a document, but the moment [he] leaves CIA, [he] should not read the document. [This] idea... is privacy in transit. Because usually… we enforce privacy close to the source, but once [this is taken] away from you, you can't really enforce it [anymore]. [W]e do spend a lot of time thinking about [such] problem[s]." [Subject B]

Maturity and Stability

According to subject *A*, the security domain as a whole is immature and unstable.

"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."[Subject A]

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.

"[I]t is an unstable domain specifically because the architectural artifacts that we have designed for security were designed with a set of assumptions in mind which are no longer true of real computer systems. So the architecture is not well matched to the real world."[Subject A]

While subject *B* did not remark on the domain in general, he indicated that many aspects within security are immature. On the other hand, he also pointed out there are some mature aspects within security that have well-established solutions.

"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." [Subject B]

"[T]here's both maturity and immaturity within the space. … [T]hings 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."[Subject B]

Sources of Difficulties

Composition is particularly difficult in engineering secure systems because emergent properties can cause serious problems when putting two or more components together. It is possible to have individual components that possess certain security features but the combination of these components may not provide the same desired level of security.

"[I]t is unfortunately the case that lots of security problem[s] do not compose in a mathematical sense. [I]f X has security property 1, and Y has security [property] 1, [then] X+Y does not [necessarily] have security property 1."[Subject A]

"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." [Subject D]

Subject *A* suggested that a framework approach does not work well for this problem since it is often underspecified - an undesirable property when building secure systems. He indicated that it is theoretically possible to have a precisely defined set of frameworks that are specific enough for security but abstract enough for general applications; however, practically it does not exist.

Awareness and understanding of security issues is low among people in general. According to subject *A*, this leads to difficulties in getting people to appreciate the feasibility (as in the case of Digital Rights Management) and justification of various levels of security. We believe that awareness promotion programs and user training can help improve this situation.

"[C]ost justifying security has always been a nightmare… So you can come up with a perfectly good architecture and everybody says, 'Ok, let's build one tenth of it.' " [Subject A]

In fact, security is sometimes so poorly understood that ideas that are fundamentally bad can still succeed in the marketplace.

"There's large number of products on the market with unsuccessful security architectures." [Subject A]

For example, "*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." [Subject D]*

However, some signs show that people are starting to pay more attention to security.

"[Security] could actually make or break the deal."[Subject C]

"It's getting easier to justify it now… on the basis of reputation damage… [I]t's way easier to get security projects justified after your website has been defaced." [Subject A]

In order to address these key issues, we are interested in finding out what skills are required of software architects, especially those who deal with security issues. In the next section, we will present our subjects' opinions on the critical characteristics of architects.

2.2 Characteristics of Architects

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

Breadth is the most important characteristic. Architects must be generalists so that they understand all the different parts of the system and do not only focus on 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." [Subject B]

"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]." [Subject A]

"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." [Subject D]

Some of the essential personality traits of an architect are persistence and persuasiveness. If this is not enough to convince the team, an architect needs to have the ability to take on the role of a benevolent dictator as well. In other words, 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."[Subject B]

"[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]." [Subject C]

There is some debate about whether good architects are 'born' or whether people can become good architects through training and coaching. Subject *A* suggested that a good architect can be mentored, but the person being mentored has to have some amount of raw talent for being an architect.

"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" [Subject A]

However, subject *B* 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." [Subject B]

To be effective in managing security requirements 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." [Subject A]

Subject *A* described three attitudes people may have in response to a new attack. However, only one is 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." [Subject A]

2.3 Critical Issues

So far, we have discussed key aspects in security and presented critical characteristics of security architects. We now present our data on managing security requirements from three perspectives: establishing security requirements, prioritizing security requirements and architecting security requirements.

Establishing Security Requirements

Security has a fundamental difference from all the other requirements, such as reliability, safety and performance. In the latter, we usually expect to have random component failures and accidents. In the former however, failures are often caused intentionally by capable and motivated adversaries. Therefore, 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. Overall, threat modeling involves decomposing an application to identify its key assets and then identifying 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." [Subject A]

"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." [Subject A]

Subject D 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…" [Subject D]

Subject *A* also commented that security problems cannot be solved by ontology-based approaches since those are generally very inadequate. He suggested that 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." [Subject A]

"[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." [Subject A]

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." [Subject B]

"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." [Subject C]

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." [Subject A]

"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..." [Subject B]

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." [Subject A]

"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." [Subject C]

"[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." [Subject D]

Security must extend beyond simply the software aspects of the system. For example, *"Kevin Mitnick should not be able to talk the operator into giving up a password. [This] is a genuine requirement." [Subject A]*

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 key and which are subordinate. Priorities could be based on the likelihood of the risk will becoming reality, cost/benefit analysis, areas of particular concern for the stakeholders etc. This requires understanding the likely adversaries in terms of their capabilities, resources, motivation, risk tolerance and level of access. Only through this understanding, it is possible to derive requirements that provide the strongest defense and recovery mechanisms at an affordable cost. Deciding on how to prioritize requirements is usually done through negotiations. 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…" [Subject C]

It is not practical, and usually impossible, to achieve 100% security. Not only is it too expensive, it is unachievable because not all weaknesses and attacks can be anticipated. Vulnerabilities can be found in even carefully designed products, and new attacks are continually being discovered. However, security levels are defined with respect to specific goals; they are either 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 about security requirements that can be adjusted. In other words acceptable risk mitigation is attainable even though security is not achievable in the 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." [Subject A]

"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.' " [Subject A]

However, sometimes other 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." [Subject B]

Architecting Security Requirements

All our subjects expressed the opinion that it is important to consider other requirements in support of security requirements while building secure systems. We believe this is because security requirements, unlike functional requirements, must be considered at every iteration of the development cycle.

We observed that there is a slight disagreement on how the subjects categorize these 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." [Subject A]

"[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." [Subject A]

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

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." [Subject B]

"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." [Subject C]

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." [Subject A]

"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." [Subject A]

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." [Subject D]

Security goals must be designed with a farseeing 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 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." [Subject A]

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

Even though security is an integral part of the system, we must be able to address the issues of modularity and externalizability. 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." [Subject C]

"[In] the first release of J2EE [security] was totally enclosed within the architecture, meaning it was not open… So every vendor did security in their own way because that was not specified by the standard… we realized that this was no good. So rather than implementing something ad hoc for the moment, we said… it would be nice to externalize the security so that anybody can plug into [it]." [Subject B]

Subject *B* pointed out that this decision helped their company in two ways: (i) they were able to integrate with several other products and (ii) when the standards came out they only had to replace their API with the standard API, unlike the other vendors who were struggling to dissociate security from their application server.

"If you design with some pretty standard rules up front, it makes things a lot easier moving on." [Subject C]

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." [Subject A]

In short, we have seen how architects establish, prioritize and architect security requirements in practice. In general, we observe that at the requirements level, the architects must consider security explicitly. Security requirements should not be an "add on"; and one should take into account emergent characteristics of security, including explicit coverage of what should be protected, from whom and for how long.

2.4 Validity Issues

Case studies are a specific empirical research method to gain a deep understanding of a particular phenomenon in its real life context. As such, it is characterized by analytic generalization, not statistical generalization, i.e. it is not understood in terms of samples, but in terms of analysis and comparison of cases.

We address three validity issues in our case study that are critical in empirical studies [3]: 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 questionnaire was initially drafted by one author based on brainstorming. It then underwent a number of reviews by each author. Reviews were carried out among the authors after each interview session where 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.

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.

3 Discussion

From the data collected, we observe that 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.

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. 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.

Understanding security problems is an ongoing challenge. Today's security problems are different from yesterday's, and tomorrow's problems will be different from today's.

It is important that architects understand different threat models and continually learn about new solutions to prevent new attacks. It is also evident that there is no universal definition for the term *security architecture*. The first job of a security architect is to describe which kinds of relationships are and are not secure. Security architecture in general provides a framework and a foundation to enable secure communication, protect information resources and ensure that new methods for delivering services are secure.

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.

The literature also has supporting evidence for our position. 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 just good software engineering [4]. 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. *"In the context of security engineering, standard obstacle analysis appears too limited for handling malicious obstacles"* [5][6]. Nevertheless, as Wolf pointed out, *"Software threat analysis is a young art"* and existing models do not adequately support the analysis needed [4]. There is much work to be done in the security domain.

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 [7], 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. 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. They finally conclude by stating, "*security engineering is a systems engineering skill*", and its most fundamental policy is that it is based on common sense. From the above discussion, it is evident that the current software engineering literature is congruent with our findings.

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.

Several issues have surfaced in our case study, which require further research: (i) how architects should be involved in requirements elicitation and negotiation; (ii) how frequently occurring problem shapes and requirement forms can be identified; (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) if there are conflicts between security requirements, how they should be resolved. In cases where more evidence is required, we will either follow up with the current subjects or conduct new interviews.

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.

4 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 Judgements". In *New Vision for Software Design and Production Workshop*. Vandebilt University, Dec 2001.

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

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

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

[6] 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.

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