Information Security Antipatterns in Software Requirements Engineering

Miroslav Kis, *Ph.D., CISSP, Member IEEE*

miroslav.kis@bmo.com

Abstract

Requirements engineering is one of the key activities in the software development process. The rapid expansion of e-commerce and internet applications increases the need for adequate application security. Yet, conventional requirements engineering methodologies rarely mention information security aspects. The information security community, on the other hand, has developed system security requirements specification methodologies. These methodologies, from the software architect's point of view, are often hard to understand and too general to be applied. By following conventional methodologies and failing to thoroughly understand the security consequences, architects end up with inadequate application security. This paper presents two commonly observed cases - antipatterns. In the first case, an old and well-known (perimeter security) model is applied in a new context without analysis of the security requirements. In the second case, the impact of lacking data sensitivity classification and threat analyses is considered.

***Keywords**: Software Engineering, Information Security, Requirements Engineering*

1 Introduction

According to Greek mythology, Odysseus had to navigate his ship between Scylla, a sea monster who lived on the rocks of the Strait of Messina, and the whirlpool Charybdis that was on the other side of the strait. Scylla, a horrible doglike creature with six heads and twelve feet, seized sailors from passing ships and devoured them. Charybdis sucked in and spewed out huge amounts of water. The whirlpools this created would pull in any ship that happened to be nearby.

In today's e-business era, software architects, who need to develop secure software, have to navigate their projects through a strait between conventional requirements engineering methodologies and system security specification methodologies. Conventional methodologies barely even mention information security and offer little or no help (e.g.[1][2][3]). Methodologies developed by the information security specialists are often too general to be directly applied in software development (e.g.[4][5]). Furthermore, a substantial information security background is needed to understand them (e.g. [6]). That makes them unreadable and practically unusable for most software architects.

According to the legend, Odysseus saved his ship from being pulled by Charybdis. He lost, however, six of his sailors to Scylla. In real life, to save projects from being late and over budget, we avoid system security specifications methodologies. This paper is about the losses we incur by following conventional requirement methodologies and failing to understand security aspects of requirements engineering.

There are two main types of problems we face in everyday practice related to this. First, to secure an application without spending excessive time and effort, we are tempted to use some known solutions like putting up a firewall or using simple password authentication. Applying a pattern, a solution that has already been extensively used in practice, might seem to be a reasonable idea. In many cases, however, a solution applied without a thorough understanding of security requirements does not provide adequate protection within the specific context

The second issue is that we design the application failing to understand the real value of data we need to protect: we do not perform data sensitivity analysis. Furthermore, we do not analyze if an attacker has an interest to compromise the data processed by the application. The consequences of not performing data sensitivity and threat analyses are that the application security requirements cannot be properly defined, and the solution will not provide an adequate security. In most cases we will waste time and effort to protect unimportant data and, at the same time, fail to provide strong enough protection for the most important information.

We analyze these problems using an example of a payroll application. The application is deployed in both mainframe and intranet environments. The two antipatterns should help software architects and project managers to recognize and avoid some common, security related pitfalls.

Copyright ã 2002. Miroslav Kis. Permission is granted to copy for the PLoP 2002 conference. All other rights reserved.

2 Requirements Antipatterns

For each antipattern in this section we present the following elements. First, we present the description of a problem to be solved and some relevant background information. Next we analyze the context in which the problem usually arises and faulty beliefs that lead to the antipattern solution. Then we present the antipattern solution, analyze its security impact and give advice how to properly solve the initial problem. Finally the symptoms that can be helpful in diagnosing the antipattern are presented.

2.1 Perimeter security: the Maginot line of enterprise applications

2.1.1 Problem

The statement of the problem is pretty simple: we have to secure a typical n-tier enterprise application. Based on this problem statement, it is natural to ask the following question: if we are faced with a typical application, couldn't we simply apply a typical security solution? This antipattern shows why a well known, typical solution from the pre-internet time, fails when it is applied to a modern enterprise application.

2.1.2 Background

Before the internet era, an application would be deployed on a mainframe computer (see

Figure 1). Only a limited number of users and administrators would have physical access to the system terminals. Outside network connections were rare if they existed at all. The typical security solution was to:

- Use passwords to control user access to the system
- Use firewalls to restrict and guard network connections

The solution of using a simple password and a firewall was adequate at pre-intranet time, because one could make the following assumptions.

- Users access the mainframe using terminals.
- A separate wire is used to connect each terminal to the mainframe.
- Physical access to the terminals is limited to a small number of users and administrators.

2.1.3 Context

Yet today, when we analyze modern intranet infrastructure, the context has changed and the following assumptions are valid:

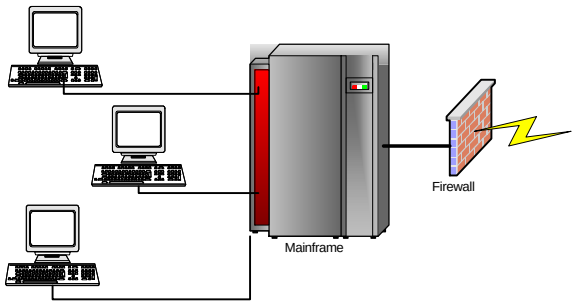

Figure 1 Mainframe connectivity diagram

- Users access the mainframe using intelligent terminals (desktops, laptops and workstations).
- All the intelligent terminals are connected to the mainframe over a local area network (LAN).
- Most of the company's employees have access to the LAN through their computers.
- The number of potential attackers has been increased from a small group of trusted users to almost all employees in the company.

2.1.4 Forces

The two main forces that influence the quality of the security solution are the above mentioned: time to market and the difficulty with applying general system's security theory in software development. Let's analyze in more details both of them.

First, consider the time to market factor. In order to keep competitive position on the market, companies are forced to come up with new versions of the products as soon as possible. In most cases, the functionality of the product is much more important than any other quality attribute of the product including information security. Additionally, security of the product is hard to evaluate and normally becomes an issue only if it is compromised. Under these circumstances project sponsors and managers are tempted to reduce time and effort devoted to security design.

Second, there is a big disconnection between software development theory and practice on one side and general system's security theory on the other. There are several reasons for that disconnection. One very important reason is that information security deals with human attackers that can exploit a wide range of vulnerabilities:

- Technical vulnerabilities of the system. One example of that would be when confidential data is stored in the clear text on an unprotected, shared network drive.

Copyright ã 2002. Miroslav Kis. Permission is granted to copy for the PLoP 2002 conference. All other rights reserved.

- Overlooked problems in development and maintenance processes. Implanting malicious code into the application caused by lack of change control would be an example of this type of vulnerabilities.
- Human factor related problems. An example of this type of vulnerability would be weak passwords.

Any theory that tries to model such a diverse environment inevitably has to be complex and therefore hard to learn and implement. Conversely, simple models can only cover certain aspects of the overall problem and they might not provide sufficient protection in a real life situation. For example one of the most famous models, Bell-LaPadula model [7], covers only confidentiality aspects of the security. Biba model [8] on the other hand covers integrity. Even the most complex, Common Criteria model [6], has a whole list of the security aspects that are not covered.

The next important factor is that significant background knowledge in security, system's theory, software architecture and, software development methodologies are needed in order to understand and apply these models. Educational background of a typical information security person is computer systems administration combined with network level security. Education of a typical software architect covers only very basic security topics. Both these professionals need substantial additional knowledge to be able to provide an adequate security solution for software applications.

2.1.5 Faulty beliefs

In essence, the perimeter protection model, described earlier, is still the dominant security architecture model. Since the perimeter solution would normally be applied once the application is deployed, the typical assumption architects make is that security is a plug-in feature added to the application once development is completed. Due to the fast growth of the internet, applications have moved quickly to the 'global network.' Yet, the understanding of new security challenges lags behind.

2.1.6 Antipattern solution

The antipattern solution applies perimeter security model to the modern enterprise application architecture (see Figure 2). The application logic may still be implemented on a mainframe computer as a legacy application or it can be implemented on a separate application server. In both cases the communication between the presentation and business logic tiers goes over the local area network (LAN).

2.1.7 Consequences

To better understand the security consequences of avoiding security requirements analysis and implementing the perimeter security concept in the wrong context, we will analyze an example of a payroll application initially implemented on a mainframe computer (Figure 1) and then used in an intranet environment (Figure 2).

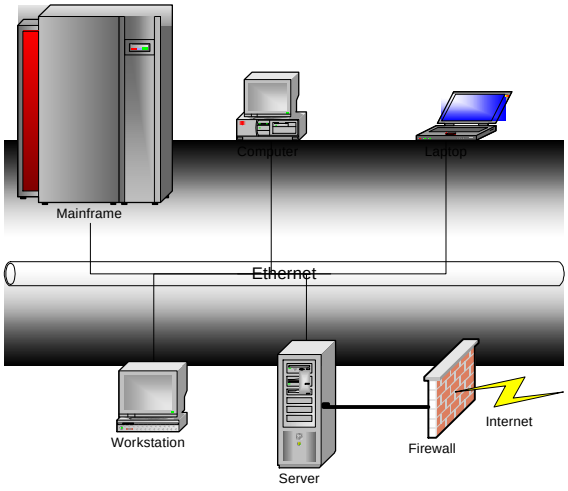

Figure 2 Intranet connectivity diagram

Any communication between users and the mainframe in the intranet environment (Figure 2) can be easily observed and altered by an attacker. By starting a simple 'sniffer' program on his computer, the attacker can monitor traffic on the LAN, including passwords sent in the clear. In the original scenario (Figure 1), it was a significantly more difficult task to achieve the same: physical access and a separate wiretap for each and every connection line were needed.

Firewalls provide only partial access control to the resources they are protecting. By checking the source and destination addresses of the packets, they can limit the number of computers that can access the mainframe. The source address of an IP packet can be, however, easily changed in any text editor. That way an attacker can spoof the packet original address. While some firewalls can do slightly more advanced analyses, they still cannot help at all in protecting data confidentiality, integrity and authenticity.

The perimeter security model proves, therefore, to be the Maginot line of the information security in internet / intranet era. The Maginot line was built in the 1930s, based on assumptions valid in the First World War. It failed badly to protect France from defeat in the beginning of the Second World War. In the same way, the perimeter security model, based on the assumptions valid in the

Copyright ã 2002. Miroslav Kis. Permission is granted to copy for the PLoP 2002 conference. All other rights reserved.

mainframe era, fails to protect an enterprise application from the new threats related to the internet / intranet environment. More specifically, the perimeter security model does not provide data confidentiality and cannot protect from an attack originating from the local area network within the company.

2.1.8 Symptoms

A good indication that this antipattern has impacted architects' decisions is that security requirements specification is postponed until the late phases of application development, and sometimes avoided altogether.

A question usually asked by project teams with mainframe development background: 'Why is that solution not acceptable when it was fine before?' also shows its presence.

2.1.9 Refactored solution

The problem is that without proper security requirements analysis we cannot be sure that the perimeter concept, applicable in one context, provides adequate protection within a new context. That analysis should be performed in every case since a solution that provides an adequate level of security in one context can be entirely useless in a seemingly similar context. Without thorough analysis we cannot determine if the contexts are similar to the extent that the same solution would be adequate for both cases.

The second point is that an information security solution, added after an application is developed, in the majority of cases is not effective. This assertion is valid for all quality attributes of an application and information security is not an exception to that rule. Security of an application is impacted by all aspects of the application. Security analysis and design should go hand in hand with the analysis, design and deployment of the application.

We should also mention the common misconception that encryption of all the traffic on the LAN could resolve all the security issues. Without going into the details of all the aspects of this misconception, for the purpose of this analysis it is sufficient to note that all the employees would still have to have access to that encrypted communication channel. It is obvious that it does not in any way protect from the internal attack.

And last but not least: the root cause of the problem we are facing is disconnection between software development theory and practice on one side and general system's security theory on the other. The question is how to bridge that gap?

The key piece of the solution would be to integrate general system security theory into the existing software development methodologies. System security should be presented there from the perspective of the people that design and implement the system: software architects and developers.

The second important factor is that both software developers and security assessors need to have knowledge of software architectures, development methodologies and information security methodologies. Otherwise it is very difficult to communicate between people who only know their side of the story, and cooperation is often disappointing since they do not talk the same language.

2.2 Security design without assessment of the business value of the data – Clausewitz syndrome

2.2.1 Problem

The problem we have to solve is the security of enterprise software application.

2.2.2 Background

The main conclusion of the previous pattern is that without thorough security requirements analysis we cannot determine if a solution provides adequate protection for the application. The next question we have to ask is what we mean by thorough analysis. At minimum, we have to determine the key elements of such an analysis. In this antipattern we analyze the significance of the business value of the data we are protecting.

Two key indicators that show the business value of the data are data sensitivity and threat analyses. The data sensitivity analysis describes the business impact if data is observed or altered by an unauthorized person as well as the impact of data being unavailable to the legitimate user. The threat analysis, on the other hand, is concerned with a likelihood of somebody attacking the system under consideration.

2.2.3 Context

The context in which this pattern occurs is requirements gathering phase of the software development process. Information security requirements are needed to design a solution that would protect the information processed by the application.

2.2.4 Forces

The same forces that influenced solution in the previous antipattern: time to market and problems with applying general system's security theory in software development, are present in this case too.

Copyright ã 2002. Miroslav Kis. Permission is granted to copy for the PLoP 2002 conference. All other rights reserved.

2.2.5 Faulty beliefs

While technology alone cannot solve the problem, in most cases technology is indeed a very important part of a solution. It has to be recognized, however, that technology is just a tool to implement a solution. The problems exist in the business domain and we have to understand these problems and find the solutions before we can use any tool.

Another faulty belief is that business customers and users do not know what they need related to information security. While business people might not know about the technology, any business person capable of staying in business is well aware of the value of the information in his possession. In most cases, the problem is how efficient are our techniques for eliciting that knowledge.

2.2.6 Antipattern solution

In the vast majority of cases, any kind of business analysis of information security requirements is skipped. Consequently, a uniform protection of all the resources in the application is implemented. The other possible case is that some security solutions that are perceived as 'strong' are arbitrarily used within the application (e.g. usage of a strong encryption algorithm without real understanding why).

2.2.7 Consequences

The lack of data sensitivity and threat analyses leads toward inadequate protection of the resources we have to protect: some sensitive data might not be protected well enough, while we might spend unnecessary effort and money to protect data that does not need strong protection.

Furthermore, the severity and magnitude of the business impact also help to set an upper limit on the scale of costs that might be acceptable to invest in protection.

2.2.8 Symptoms

Generally in cases like this, project team understands that security aspects should be addressed. They even do some security analysis, but the architectural solution is based solely on the technical analysis of the problem. The key indicator of the existence of this antipattern is that customer and users are not involved in requirements gathering process: the technical part of the project team defines the requirements.

The statements that you can often hear in the situation like that are:

- "We will encrypt everything"
- "Customer does not know what he needs"
- "We will use the latest version of the security product xyz"

2.2.9 Refactored solution

In order to understand how to estimate the business value of the data, let us go back to our payroll example from the previous antipattern. We will also show on that example how the lack of understanding of data sensitivity and possible threats can influence the security solution.

A high-level version of data sensitivity analysis would identify the following data groups[1]:

- Employee name, phone number, and address (I)
- Department and position (I)
- Salary amount (C,I)
- Social Security Number (C,I)

The first idea that comes to mind when we process payroll data is to protect everything in the same way since this is a payroll. Yet, as we take a closer look at the actual content, it is easy to see that employee Social Security Numbers are highly confidential, and can be used to steal a person's identity. The telephone number, on the other hand, does not have to be hidden, since it is usually publicly known information. It is obvious that not all the data has to be protected in the same way. A more detailed analysis shows the following:

- For the name, telephone number, address, department and position, our only problem is to make sure that no unauthorized changes are made.
- Individual position and the employee's department are not secret. Aggregated information about the whole company which shows organizational structure, however, is normally kept secret.
- Salary level is considered confidential, and should be protected from unauthorized change. The rules related to who can view or change this information can be pretty complicated, reflecting a company's business rules.
- The Social Security Number is here as the employee identifier. Access to this information should be strictly controlled.
- Most likely, availability of the whole system is critical the day before pay day, so that pay checks are issued on time. At any other time, if the system is not available it will not significantly impact any business activity.

[1] *Note: the letters in brackets show which security attributes of the data group are important to protect (C – confidentiality, I – integrity).*

Copyright ã 2002. Miroslav Kis. Permission is granted to copy for the PLoP 2002 conference. All other rights reserved.

Now we can better understand the significance of all the pieces of data processed by the application. The next question would be whether that is enough to design the system. And the answer is no, it is not!

The analyses we performed so far have actually determined only the damage that would happen in case an attacker gets hold of the data. We could, based on this analysis, proceed and do the design. The design, however, would not provide an adequate security solution. The reason for that is simple. We have not analyzed the threat: are there people that are interested in attacking the application?

Let us go back to our example to clarify how practically that analysis can be performed. Additionally, we will analyze two cases where this application is deployed. The first case is a small startup company that wanted to automate their payroll system so they do not have to employ an additional person to do that job. The second case is a large corporation that has thousands of employees. So let us do a brief threat analysis for the startup company:

- It is highly unlikely that somebody would try to alter telephone number, address, department and employee position files for a small company.
- The organizational structure of a small startup is usually quite simple, and can be easily guessed without using the payroll application.
- Some current employees and prospective candidates might be interested to know salaries.
- Misuse of someone's Social Security Number is a criminal act. In most cases, only criminals outside the company would be interested to obtain them.
- Even an unfair competitor would not try to make the payroll system of the startup company unavailable. No significant harm could be made, nor any gain for the competition.

Now, let us analyze the threats in the case of a large corporation:

- Disgruntled employees or an unfair competitor might want to portray the corporation as incompetent, which can influence customer's confidence. Delaying pay checks for a day by altering employees' personal information can cause a huge problem that can become publicly known. The same motive can be behind an action to make the application unavailable.
- The organizational structure of a large corporation might reflect their intention to develop a new product. The size of their R&D department may help their competition to understand it.
- Both the employees and competitors could be interested to know salaries for several reasons.
- As in the case of the small company, criminals outside of the corporation would be interested to obtain Social Security Numbers.

From the analysis, it is obvious that in both cases salary levels should not be sent in the clear and Social Security Numbers should be protected with stronger encryption. Apart from these common elements, the other components of the solutions will be significantly different.

The large corporation would make sure the system is available whenever it is needed, so they would have one or more redundant servers. Aggregated information about the corporation and salary ranges should also be specially protected in the large corporation.

The small company would most likely define manual procedure to write checks to all the employees in case the application is down.

In reality we would take into consideration some more details. The role of the example above, however, is to show how the security solution for the same application can be dramatically different, depending on the business impact. This conclusion takes us back to our initial statement that avoiding the data sensitivity and threat analyses can impact a company because of either overspending (e.g. small company unnecessarily installs three servers) or lost customer confidence (e.g. prematurely released information of a new product development etc.).

3 Conclusion

Application security is a difficult problem to solve. In the past, only software architects engaged in military application development had to learn complex security methodologies. The rapid expansion of e-commerce and internet applications increases the need for an adequate application security for practically all the enterprise applications. The software architects of enterprise applications are faced with a difficult choice.

The first option is to make a significant effort to understand and implement complex security models. The time needed to do it can excessively increase time to market and impact the development company's business.

The second option is to implement a solution failing to understand even the security requirements for the application. This approach, on a short term, might give an impression that the application security problems are solved. Yet the two antipatterns presented in this paper clearly show the negative impact of such an approach on company's business and security.

Copyright ã 2002. Miroslav Kis. Permission is granted to copy for the PLoP 2002 conference. All other rights reserved.

The first antipattern shows that security cannot be treated as a feature to be added once the application development is completed. The main reason for the misconception is that the perimeter security model, predominant in pre-internet time, would have been applied after the application development was completed. The perimeter security model proves, however, to be a Maginot line of information security in internet / intranet era. Perimeter security model fails to protect an enterprise application from the new threats related to the internet / intranet environment. More specifically, the perimeter security model does not provide data confidentiality or integrity. In addition to that, it cannot protect from an attack originating from the local area network within the company.

The second antipattern presented in the paper shows the impact of the data sensitivity and threat analyses on the security solution. The common misconception is that an adequate security solution for the application can be developed without thorough understanding of the business environment. The analysis, however, shows that the lack of data sensitivity and threat analyses leads to inadequate protection: some sensitive data might not be protected well enough, while we might spend unnecessary effort and time to protect data that does not need strong protection.

The famous German theoretician of war Carl von Clausewitz concluded that war is not just about the fighting and arms, but also about the politics. Similarly, the conclusion of this paper, and a common denominator for both antipatterns presented, is that we have to recognize and implement in everyday software development practice that application security is not just about firewalls and passwords. Application security is much more about the business context within which the application is implemented.

Acknowledgements

The author would like to thank his colleagues and friends from Information Security Bank of Montreal for their support while writing this paper. Martin Green, Milena Jelich, Daniel Jones, Vivek Khindria and Marla Nystrom-Smith have reviewed several versions of the paper. Long discussions with all of them helped me to better understand the security problems analyzed in this paper and also to improve the clarity of presentation.

Special thanks to Alejandra Garrido, the Plop 2002 reviewer of the paper. Her excellent comments inspired me to do more research and significantly improve the quality of the paper.

References

[1] Michael Jackson, 'Problem Frames and Methods: Structuring and Analyzing Software Development Problems', Addison Wesley Professional, 2000

[2] Daryl Kulak, Eamonn Guiney, Erin Lavkulich, 'Use Cases: Requirements in Context', Addison-Wesley Pub Comp., 2000

[3] John Wordsworth, 'Software Development With Z: A Practical Approach to Formal Methods in Software Engineering', Addison-Wesley Pub Comp., July 1992

[4] Dieter Gollmann, 'Computer Security', John Wiley & Son Ltd, 1999

[5] Charles P. Pfleeger, 'Security in Computing', Prentice Hall PTR, 2nd edition, 1996

[6] 'Common Criteria' Version 2.1 / ISO IS 15408, http://csrc.nist.gov/cc/ccv20/ccv2list.htm

[7] D. Bell and L. LaPadula. 'Secure computer systems: Mathematical foundations and model' *MITRE Report*, MTR 2547 v2, Nov 1973

[8] K. Biba. 'Integrity considerations for secure computer systems' Technical Report 76-372, U. S. Air Force Electronic Systems Division, 1977

Miroslav Kis received his BS degree in Electronics and Telecommunications from the University of Belgrade in 1980. He received his MSc degree in 1985 and PhD in 1991 in Computer Science from the same university. He is a Certified Information Systems Security Professional and a member of IEEE Computer Society.

Dr Kis is a senior advisor for Bank of Montreal and manager of the Strategy and Technology group within the Information Security department. Before joining the Bank, he held a senior staff engineer position in Motorola Internet and Networking Group. There, he was the architect of Security Association Management (SAM), a Virtual Private Network protocol, implemented on Motorola Vanguard routers, before IPSEC was fully developed. He was also the architect of Vanguard ConfigWizard, an expert system for configuring networks.

Dr Kis is coauthor of BSDP, a speech recognition algorithm, and the architect of a speech recognition device developed based on that algorithm. His current research interests include system and software security methodologies, security aspects of software development processes, and biometrics authentication systems' theory and applications.

Copyright ã 2002. Miroslav Kis. Permission is granted to copy for the PLoP 2002 conference. All other rights reserved.