Bringing Security Home: A process for developing secure and usable systems

Ivan Flechais
Department of Computer Science
University College London
Gower Street
UK - London WC1E 6BT
(+44)207 679 3642
i.flechais@cs.ucl.ac.uk

M. Angela Sasse
Department of Computer Science
University College London
Gower Street
UK - London WC1E 6BT
(+44)207 679 7212
a.sasse@cs.ucl.ac.uk

Stephen M. V. Hailes
Department of Computer Science
University College London
Gower Street
UK - London WC1E 6BT
(+44)207 679 3432
s.hailes@cs.ucl.ac.uk

ABSTRACT

The aim of this paper is to provide better support for the development of secure systems. We argue that current development practice suffers from two key problems:

1. Security requirements tend to be kept separate from other system requirements, and not integrated into any overall strategy.
2. The impact of security measures on users and the operational cost of these measures on a day-to-day basis are usually not considered.

Our new paradigm is the full integration of security and usability concerns into the software development process, thus enabling developers to build secure systems that work in the real world. We present AEGIS, a secure software engineering method which integrates asset identification, risk and threat analysis and context of use, bound together through the use of UML, and report its application to case studies on Grid projects. An additional benefit of the method is that the involvement of stakeholders in the high-level security analysis improves their understanding of security, and increases their motivation to comply with policies.

1. INTRODUCTION

"Effective security is at odds with convenience" [14]. This statement reflects a common point of view among security experts and software providers. The effectiveness of a security mechanism, however, depends on both users and technology "doing the right thing". The usability of security mechanisms is not just a question of improving interfaces to security tools, but designing security to work with the real-world tasks users perform, and within the physical and social context of that interaction [18].

Recent research on usability and security has focussed on user problems and needs (e.g. [6], [20], [21]). There is compelling evidence that system developers deserve at least as much attention. According to CERT [1], the number of security vulnerabilities in systems is increasing rapidly (from 2437 in 2001 to 4129 in 2002). A recent survey [4] of similar products from different providers found that the least secure product carried a 6 times higher business risk than the most secure one, highlighting the fact that the security quality of a product can vary drastically depending on who designed and implemented it.

It is self-evident that developers play a key role in the provision of usable and effective security. But to make the right decisions during the design and implementation process, developers need a development method that helps them to identify and represent security and usability requirements in the design from the outset. Such a method must be lightweight, compatible with notations and tools already in use, and lead to secure systems that work in practice. To answer this need, we have developed AEGIS (Appropriate and Effective Guidance for Information Security), a secure software engineering method that integrates security requirements elicitation, risk analysis and context of use, bound together through the use of UML.

In section 2, we discuss in detail what type of support software developers need to build secure systems. In section 3, we present the detailed stages of AEGIS, and in section 4 we report on case studies where AEGIS has been applied.

2. ISSUES IN DEVELOPING SECURE SOFTWARE

Since the advent of the software engineering process, developers have been required to balance a number of requirements in building systems (e.g. functionality, efficiency, time-to-market, modularity, scalability, extensibility). Over the past few years, the rapid evolution of wide area networked systems has created additional security concerns. Recent research on usability of security points out that systems must be designed to make it easy for intended users to *"do the right thing"* when it comes to security [12]. The number and complexity of issues that developers of secure systems have to consider has increased such that many find it difficult to cope. Following good software engineering practices is, in many cases, not enough.

Building secure systems necessitates:

1. Following a systematic process of software engineering.

New Security Paradigms Workshop 2003 Ascona Switzerland
© 2004 ACM 1-58113-880-6/04/04....$5.00

2. Carrying out a risk assessment on which to base security decisions [10], [19].
3. Up-to-date knowledge of security threats and countermeasures.
4. Devising security mechanisms that are effective in the real world, i.e. that are usable by the intended users in their specific context of use [18].

The problem is that existing design methods for secure systems do not address all of these goals and do not provide enough support for the developers to realistically achieve them.

One method that does address all principles is the one by Abrams [5], which aims to integrate security engineering into the evolutionary acquisition process. The method follows a prototyping and pragmatic risk-based security design approach. It also relies on regular input from various stakeholders such as users and developers, as well as an understanding of the context in which the system operates.

Although we agree with the principles of this approach (contextual information about the system, integration into a software engineering strategy, risk-based security decisions), we believe that it fails to provide sufficient support for developers.

1. There is no integration with the system design documents: the results are presented in a security specification in a separate document, and in a different notation. This is more than a mere inconvenience: having a separate specification document means security requirements are usually "out of sight" when design decisions are made.
2. The context in which the system operates is not part of the documentation used throughout the design process. The system context is reviewed at the beginning of each new iteration of the prototype, as opposed to being visible throughout the whole process.
3. Although the design process provides a placeholder for risk-driven security design, there is no particular guidance as to what factors to take into account when making such decisions – especially social and cultural factors.

We introduce a method that builds on [5], but which can support developers in building effective and usable security throughout the design process, and is fully integrated with the existing software engineering tools and notations.

Appropriate and Effective Guidance in Information Security (AEGIS) uses context regeneration (based on contextual design [9]) and risk analysis as tools to assist developers in representing and addressing security and usability requirements in system design. By involving stakeholders in the high-level risk analysis and selection of countermeasures, their understanding of the need for security countermeasures, and their motivation to contribute to security are likely to be improved [10], [19].

Finally, by using UML, AEGIS provides a uniform basis on which to discuss and bind the separate areas of usability, risk management and technical design.

Grid computing

We are currently applying AEGIS to the analysis and design of a number of Grid projects. The purpose of Grids, such as Seti@Home [3], is to use the Internet as an infrastructure for distributed computing. Computing power, storage or results can all be shared across Grids, lowering the cost of research. In areas of research that require very large investment (physics, medicine, astronomy, etc.), the advantages of sharing data and resources are very attractive. Whereas current computing power can only be upgraded through the purchase of expensive machinery, Grids allow completely different concepts of operation to be supported, such as the remote use of another institution's specialised facilities (e.g. supercomputers, a specific observatory, a specialised laboratory, etc.).

This has led to a number of projects being started to investigate and create the necessary technology to make Grids a reality. Because of the nature of Grids and the number of different environments they aim to operate in, however, there exist a large number of threats, many of which are not considered in standard security analyses. This makes the need for security in these projects paramount to the future success of Grids.

3. AEGIS

AEGIS is a software engineering method for creating secure systems based on security requirements identification through asset modelling, risk analysis and context of use.

Based on the spiral model of software development – as seen in Figure 1 (inspired from [19] and [11]) – AEGIS integrates security and usability with the prevailing modelling

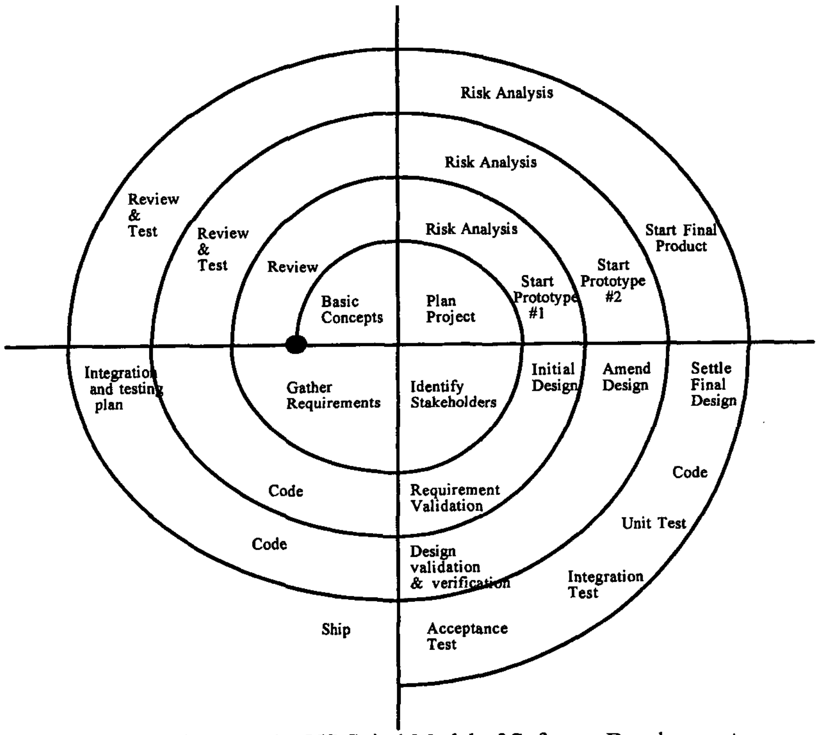

Figure 1. AEGIS Spiral Model of Software Development

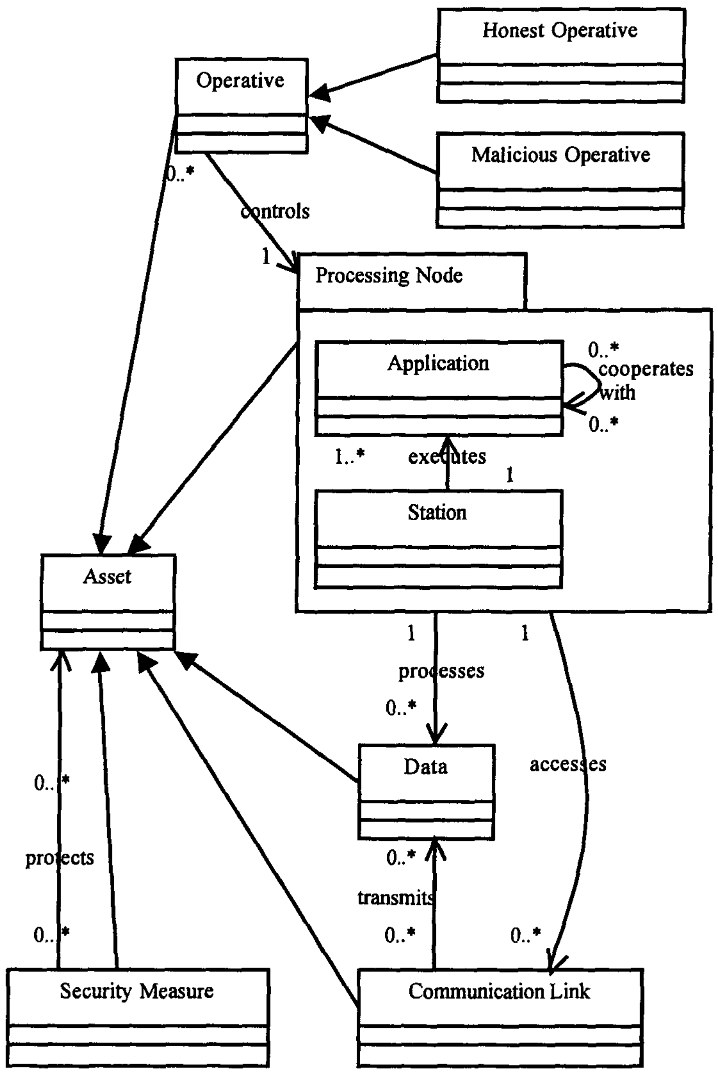

Figure 2. Relationship Diagram of a System's Assets

technique UML. This ensures that usability, thanks to contextual regeneration (inspired from the same technique that Zurko et al. [22] used to successfully design a secure and usable authorisation system), and security are visible throughout the process.

Evidence from the case studies we have conducted so far suggests that AEGIS can take place over a series of four design sessions between developers and stakeholders. Depending on the level of security needed and experience of the developers, security experts should be included to assist with identification of threats and selection/design of countermeasures.

As part of our ongoing review of AEGIS, we are envisaging more detailed support in the form of checklists/FAQs to address known security pitfalls, and to help identify appropriate security mechanisms for specific contexts of use. In the long term, links to appropriate security patterns [2] should also be added.

3.1 Participants

AEGIS is conducted with three different types of participants:

1. Facilitators
2. Stakeholders (owners, developers, users)
3. Security Experts

Facilitators are in charge of conducting AEGIS. They are necessary to keep the design sessions on track and to elicit and document the security requirements.

Stakeholders consist of developers, users and owners. It is important to have a variety of stakeholders (i.e. owners/management and all groups of users should be represented), although for practical purposes the number of participants in the meetings is best kept to 5-6. The reason for involving both owners and users is to ensure that:

1. all contexts in which the system is used are represented, and
2. owners and users become aware of each others' goals and needs.

Today, many systems are built to minimise the need for geographic closeness in cooperation – Grid systems being an example. Whilst these systems can offer many benefits, communication between different stakeholders is limited to occasional meetings. In the absence of day-to-day communication, the number of implicit assumptions made – e.g. what others are trying to achieve, and how they work – increase. Another prominent phenomenon we have encountered is what social psychologists call *diffusion of responsibility*: the notion that it is tempting to assume that someone else will take care of a particular problem [13]. To counter these tendencies, better education [20] and motivation [20] are key factors; getting stakeholders together provides the basis for improving the motivation to behave securely, and the knowledge of how to do this.

Security Experts must be involved if neither Facilitators nor stakeholders have any technical security knowledge. Expert knowledge is best used, however, in the Risk analysis and security design phase.

3.2 Identifying Assets and Security Requirements

The foundation of AEGIS is to base every security decision on knowledge of the assets in the system. Inspired by the work of Herrmann et al. [15], we use UML syntax to model the system, its assets, threats and security controls. Figure 2 shows a relationship diagram of the assets in a system.

During the first design session, the facilitators help stakeholders build a model of the system, representing various assets and their relationships.

Facilitators ask participants to state the raison d'être of the system: who is involved, what is to be achieved, and how; anything that contributes to achieving the goal is represented as an asset. Facilitators must pay particular attention to ensuring that the *context* in which people are interacting with the system is represented. This includes the physical and cultural environment, the particular roles that people must assume and the tasks they must carry out [9].

Using the model of the assets, security requirements are then gathered from the stakeholders through scenarios where particular properties of the security of an asset are compromised. For example, a requirement for the integrity of a database can be elicited by asking what would happen if the database were corrupted or intentionally (maliciously or not) modified. It is important to record these scenarios for future use and checking. This can be done by modelling them as *abuse cases* [16] – use cases of undesirable events.

For example, Figure 4 shows a model generated in a case study.

3.3 Risk Analysis and Security Design

The second design session focuses on clarifying the asset model of the system and the security requirements. Dependencies between the assets of the system must also be identified.

Based on the information gathered in the asset model and the security requirements, the third session is spent identifying the risks, vulnerabilities and threats to the system, and the fourth session selects or designs the appropriate countermeasures. Figure 3 shows the process of risk analysis and security design.

For the risk analysis and security design part of the process, it is important to ensure that expert knowledge is available in order to identify risks and countermeasures. AEGIS recommends using a lightweight risk analysis method that allows the rapid assessment of human and technical risks and threats, and focuses on building the system. It is possible, however, to employ more time-consuming, exhaustive and quantitative methods should it be appropriate for the project.

1. Determine *vulnerabilities*

A *vulnerability* is an area which is susceptible to undesirable action. There are many kinds of vulnerabilities, which can be broadly divided into two categories: technical vulnerabilities and social vulnerabilities. Technical vulnerabilities can include buffer overflows, protocol timing attacks, message replays, unsecured access points and so on. Social vulnerabilities consists of people making mistakes on security administration (forgetting to backup their files, not rescinding access privileges, leaving computers unlocked, etc.), deliberately trying to subvert the system for malicious purposes (more commonly called social engineering. e.g. convincing an administrator to reset a user's password by impersonating the user, getting a user to reveal their password by impersonating the administrator, activating the fire alarms and physically accessing a computer in the confusion, etc. [17]). More information about social vulnerabilities and a technique for modelling them can be found in [12]. This uses a model adapted from the domain of industrial safety, and distinguishes between active failures (at the operator level) and latent failures (weaknesses in the system). A security breach is a result of the combination of active and latent failures. Active failures are categorised into:

- slips (attention failures)
- lapses (memory failures)
- mistakes (rule or knowledge failures) – intended actions that lead to unintended results
- violations – actions that intentionally breach the security of the system

Both technical and social vulnerabilities should be considered equally.

2. Assess *cost and likelihood of attack in context*

This step is necessary to establish how damaging an attack on the asset (utilising the vulnerability) will be, and how likely it is to happen in the context of use.

John Adams states that *'risk is subjective. It is a word that refers to a future that exists only in the imagination'* [8]. He also shows that any risk compensation affects the risk being compensated for and that subsequent behaviours can create different risks [7]. Adams illustrates this with evidence that seat-belt legislation has reduced the number of injuries in car passengers, but has increased the number of injuries to pedestrians. This is because seat belts provide the driver with an added sense of safety and their behaviour becomes less risk averse as a result. Assessing risk is therefore a complex endeavour which, as Blakely et al. [10] state would benefit from adopting a structure which allowed the sharing of information.

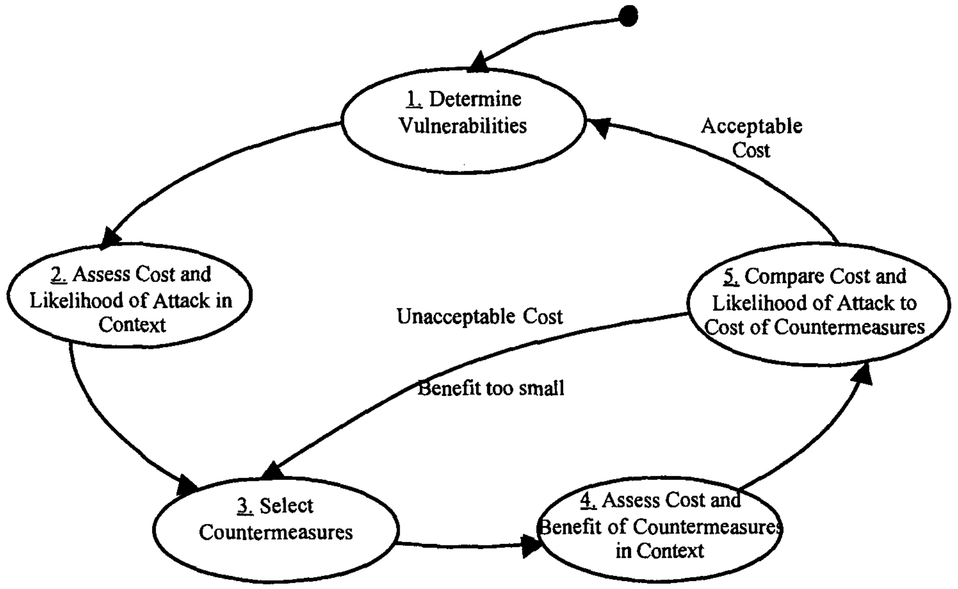
1. Determine Vulnerabilities
Acceptable Cost
2. Assess Cost and Likelihood of Attack in Context
Unacceptable Cost
5. Compare Cost and Likelihood of Attack to Cost of Countermeasures
Benefit too small
3. Select Countermeasures
4. Assess Cost and Benefit of Countermeasures in Context

Figure3. Risk Analysis and Secure Design Process

Quantitatively evaluating risks and damages, such as the ALE (Annualised Loss Estimate – a product of the probability of the risk occurring and the financial damage it would incur [10]), allows an easily used and shared measure for risk and damages. Another example of a widely used quantitative risk measurement is the security metric accompanying CERT vulnerability disclosures [1], which is based on a number of factors including the impact of the vulnerability being exploited, the ease with which it can be exploited, the number of systems at risk, etc.

One problem with this is that only easily financially estimated assets can make use of this. Non-tangible assets such as reputation, goodwill, staff morale, etc. cannot be assigned a meaningful quantitative financial cost, and this does not take account of non-financially motivated attackers

Furthermore, the usefulness of sharing quantitative ratings (such as the CERT security metric) – thereby reusing some of the acquired knowledge in the field – is currently badly affected by their lack of contextual information. Without this information, it is impossible to know whether the value has any use in a given environment. By modelling context as well as risk we hope to provide a starting point to the meaningful sharing of risk knowledge in computer security.

We currently use qualitative ratings as a means of ascertaining the importance of a particular security requirement because the relative importance of different assets is often sufficient to make decisions in research projects such as the Grid ones. These generally take the form of 'high', 'low' or 'medium' as ratings of importance. Particularly important ratings are generally labelled as 'essential'. In other application areas, such as the financial sector, quantitative ratings can also be added.

In this step, it is important to seek accurate knowledge in order to achieve an informed decision and both quantitative and qualitative measurements should be used where most appropriate. Since risk is ultimately subjective, a consensus should be reached with security experts and stakeholders, based on available information – which can include existing risk assessments, field experience, numbers of past incidents, environment of the asset, dependencies between assets, etc.

When determining the cost of a potential attack, one method of assessing this is to run through the security requirement document and confirm the ratings of importance. This information can then be correlated with other sources, such as legal requirements, know replacement costs (for replaceable assets), industry standards and brand impact so as to gather a good picture of the cost of an attack. This process is also useful to validate the initial security requirements and any changes should be reflected in the requirement documentation.

3. Select *countermeasures*

This section is the first stage of an iterative process of identifying the most cost-effective countermeasures.

Once assets and the risks they face have been identified, the next step is to determine how to address these risks. From the information gathered thus far, a clear picture should emerge as to which parts of the system are most at risk, either due to a very high likelihood of attack, or due to the estimated crippling cost of a successful attack. Attention must be given to the most likely and damaging risks first.

For example, all other factors being equal, it would be more important to secure a salary database residing on an Internet connected workstation (seen as high risk) than it would the same database on an unconnected workstation (with a lower risk from the internet). This does not mean that no attention should be paid to the second salary database, because although it has a lower risk, it still holds very valuable data.

Expert advice should be used in order to identify as quickly as possible the most likely countermeasures. It can be proposed to employ:

1. no countermeasure
2. deterrence, prevention, detection and reaction to attacks,
3. transfer of liability and responsibility (through insurance or third party intervention).

Returning to our example, in order to secure the high-risk salary database some countermeasures might include disconnecting the workstation from the Internet and locking it in a room to which only two people have the key (prevention). Other alternatives might be to install access control and intrusion detection mechanisms allowing the audit of whoever accessed the machine (detection), making misuse a punitive offence (deterrence and reaction), allowing only a limited number of MAC addresses to connect to the machine (prevention), getting a third party to secure the database and maintain it's security (transfer of liability), etc.

4. *Cost-benefit assessment* in context

This next stage in the countermeasure selection process determines what the cost of the proposed countermeasures will be, and weighs it against the benefits that they bring.

Cost of countermeasures in context

Cost in this section not only addresses financial issues, but also refers to the effort a user will expend in deploying the countermeasures. The context refers to the environment in which the attack can occur and in which the countermeasures are deployed. It is very important for the facilitator to gather information from the users in order to identify the projected costs associated with a particular countermeasure. Scenarios and use cases can again be used to document this activity.

For example, if a system forces a user to change his password whilst he is simultaneously being urged to achieve a production task for which he needs the system, the cost will be very high both in terms of loss of productivity and in frustration of the user.

Benefit of countermeasures

Benefit in this section refers to whether the controls actually reduce the risk, as well as establishing whether they provide any advantages to the user. It is important to put the control in context with the other security controls as well as the rest of the system. Taking the previous example, the benefit of forcing a password change may not be particularly evident in the face of the potential problems. It may be that a different or additional countermeasure would be more beneficial. A different countermeasure - such as a physical authentication token - or an additional countermeasure - such as user training in selecting passwords - would provide additional benefits to the user, at the cost of greater financial expenditure and the potential creation of different risks (such as having the token stolen).

5. Compare *cost* and *likelihood of attack* against *cost of countermeasures* in context

This is the final stage in the countermeasure selection process, where the actual decision to adopt a countermeasure is made depending on its benefits versus its cost.

Owners in the project should be involved at this stage – these include owners and developers. This is to establish whether the vulnerability poses sufficient risk and potential damage to justify the cost of the countermeasures.

Thanks to the information gathered so far about the various countermeasures proposed, a clear picture should be evolving as to the impact of a particular countermeasure. If the cost proves to be unacceptable, or the risk still too great, the process of selecting countermeasures (step 3) should be started again. Otherwise, time and money permitting, a new cycle (step 1) should be started to conduct a new determination of the vulnerabilities taking the new countermeasures into account. If no further controls have been added, the assessment is over.

The final output of the risk analysis and security design phase is a design document detailing the architecture of the system together with all the countermeasures which have been agreed upon (including training and staff motivation as well as technical measures), the necessary user behaviour these countermeasures depend on, and the workload this adds to users. In addition to this, the documentation generated in this process can be built upon and used to support future iterations.

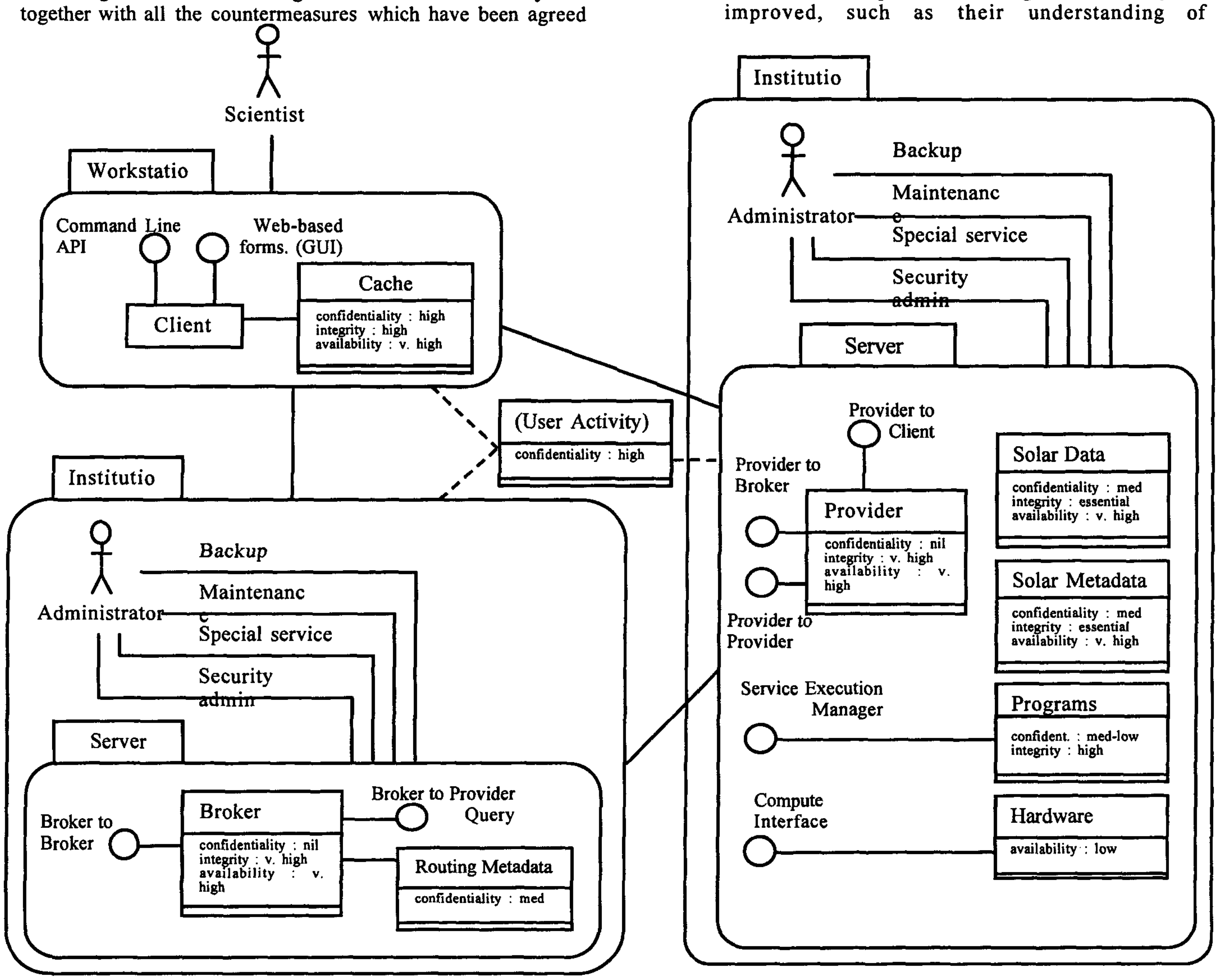

Figure 4. EGSO Asset and Security Requirement Model

4. CASE STUDIES

One of the projects that we have documented as a case study is EGSO (European Grid of Solar Observations).

The purpose of EGSO is to provide a Grid making the solar observations of a number of different observatories and institutions available to customers.

We evaluated AEGIS by looking at:

1. whether developers are aware what workload the design imposes on users
2. whether developers' knowledge of security is improved, such as their understanding of vulnerabilities, threats, risks and how to address them
3. whether developers' and users' awareness of, and motivation to, apply security have increased

4.1 Background

We started our case study by conducting three meetings with stakeholders in order to determine what the aims and requirements of the project were, and also to establish the current state of security in the project.

We then arranged a series of design sessions with up to three stakeholders (two developers and one user/manager) and applied the AEGIS method.

The initial review uncovered that thanks to the presence of very competent software engineers in the project, a high standard of practice was being applied to EGSO. This could be seen in documented use cases, requirements validation, user interface design and UML system design. The need for security had been acknowledged and some use cases, albeit in vague terms, described the need for some security mechanisms (e.g. the need for *'direct access to satellite data in near real-time, perhaps only with necessary authorisation'*).

4.2 AEGIS

In the beginning of the process, a number of previously undocumented security needs emerged, such as *'users want their results to be protected'* and data providers need to protect their resources from being swamped and attacked.

We also uncovered that *'no one is in charge of security'*. It was also stated that security had not been considered in depth because the project was *'still in (the) early stages (of) going from requirements to design'*. A final comment justified a lack of concern for security by insisting that functionality was much more important at this point in time, and that security would be addressed later.

Evidence of *diffusion of responsibility* with respect to security was also present. Assumptions were made that other people or technologies would take care of some security aspects. For example, if digital certificates were to be used, the middleware would *'take care of the PKI'* (Public Key Infrastructure). Another example, witnessed to a greater extent in other Grid projects, was the assumption that the technical support of the institutions hosting the projects would take care of their security. What happened in reality is that many institutions isolated Grid projects from their internal network, but did not make any further efforts to protect the projects.

4.2.1 Asset Identification

As facilitators, we started by focussing on identifying the major assets of EGSO. We asked our participants to draw a model of EGSO, and because of the distributed nature of GRID applications, we asked for a model that would represent every different kind of asset, without worrying about modelling the multiplicity.

The natural inclination was to draw the system isolated from its environment, and we encouraged the participants to describe where people were involved in the system and in what kinds of environments various different parts of the system existed. The wide range of possible environments for EGSO users led us to refrain from modelling too much detail, although the commonalities of the rest of the system were identified.

4.2.2 Security Requirements Elicitation

Once the main assets of the system had been modelled, we set about identifying security requirements. We started by defining the concepts of *confidentiality*, *integrity* and *availability* (for other projects, different concepts might be applicable as well, such as dependability, accountability, non-repudiation, etc.). We then looked at specific assets and asked the participants to rate them (qualitatively) according to these three terms. More specifically, we asked them to evaluate what the impact would be on the system should a specific type of attack occur.

For example, this is how we rated the solar data asset:

Availability: What would happen if users were unable to access this information? The system needs to be *'robust within reason'*. Identifying levels of acceptability was *'not something that's been clearly defined.'* Availability was therefore rated as being a *'very high'* requirement.

Integrity: How important is it for the information held at the providers to be what users and providers expect it to be? *'If there was no data, there would be no system'*. Similarly, if the data was modified in any way so as to mislead, this would be unacceptable. The Integrity requirement was therefore rated as being *'essential'*.

Confidentiality: Does the Solar Data have to be kept secret from anyone? *'Some providers may want to restrict the access to the data for a period of time'*, but *'they may not want to use EGSO for that type of data'*. The requirement for confidentiality was rated as *'medium'*.

This proved to be useful for three reasons:

1. Participants had to look systematically at their system and identify a wide range of security requirements for every part of the system (many people tend to forget that requirements other than confidentiality are also important).
2. It allowed the explicit description of implicit assumptions, which in turn uncovered problems.
3. The final outcome, although it consisted of qualitative ratings, allowed the easy identification of the most important assets in the system

The full asset model, complete with the identified security requirements can be seen in Figure 4.

4.2.3 Risk Analysis

Although the risk analysis is not complete, we started by identifying the various dependencies between the assets of EGSO. This highlighted, for example, that the availability of the solar data (rated as very high) was completely dependent on a wide range of factors such as provider administrators, broker administrators, routing, hardware operation, network links and their traffic.

Prior to carrying out the AEGIS analysis with EGSO, there had been a debate about whether or not to use digital certificates. The perceived cost and complexity of employing certification (based on little more than word-of-mouth) was driving the discussion, but the full consequences of either path of action had not been analysed.

Before even starting the risk analysis, a strong desire to avoid having to use digital certificates was voiced, illustrating the fact that accurate knowledge in this area is paramount.

During this process, we identified that some users were going to require a privileged access in order to be able to run resource-consuming jobs. This conflicted with the stated desire to avoid having to employ a robust version of access control and authentication. It soon became apparent that ruling out certification at this stage would be premature and could possibly lead to a greater workload on developers and more complex system.

We anticipate that the rest of the risk analysis will identify a number of vulnerabilities, mainly in areas of availability of services and integrity of data. We have already provided a number of scenarios in which the data that was assumed to be public could be modified to suit a particular attacker, or where user software running on provider hardware could be used to attack the system.

4.2.4 Security Design

Whilst the security design sessions are incomplete, the identification of the dependencies in the beginning of the risk analysis highlighted the total dependency on system administrators and prompted the need for specifying their duties. This in turn led to some discussions about the stated need for a low cost buy-in from observatories wishing to participate in the project, balanced against the current design requirement for their administrators to actively carry out various security tasks.

Other areas were also identified where policies would have to be detailed, such as the expansion to different providers, data update and integrity control, and acceptable use.

4.3 User issues in security decisions

The need to document specific administrative policies has stemmed from explicitly stating the implied behaviours, duties and skill levels expected of the administrators of the system. This analysis has highlighted the need to detail the duties of the administrators in order to provide ground for both guidance and security.

Issues that will be raised include the problems users can have with key management if the need for certification arises, the need to clarify the specifics of tasks that administrators must perform and conflicts that may occur if there is no provision for prioritising administrator tasks (backup, maintenance, security) and production tasks (special service). We will also highlight the need for a security culture in which secure behaviour is encouraged, possibly through the use of incentives and punishment for transgression.

4.4 Developer knowledge of security

Some statements uncovered during the design sessions illustrated a confusion and misunderstanding over what securing the project entailed. For example, backup needs and procedures were initially seen as an archival problem that should be solved by individual providers, even though EGSO was intended (among other things) to be a reliable means of access to the data.

Other evidence of a better understanding of security can be taken from comments such as how this approach has raised a number of issues that had never been contemplated, such as the need for EGSO to trust providers to behave in the expected way as much as the need for providers to trust EGSO. Also, in the words of one participant (and paraphrasing an American politician), it was *'converting the unknown unknowns to known unknowns'*.

The process also seems to have changed the attitude of the stakeholders from an initially held optimistic outlook on security, to a more searching and deterministic attitude.

Furthermore, developers are happy to use the process and some have even found it to be useful in gathering functional requirements and understanding the system.

4.5 Motivation to apply security

In this case study, even without our involvement, the motivation to apply security existed – what was missing was a systematic analysis and plan for implementing it. There was isolated evidence of some initial reluctance by some participants of EGSO to get involved because of the need to pursue functionality, but this quickly disappeared as soon as we started.

Since our involvement, some of the points and suggestions that were made have prompted changes in the design and increased the resolve that security is a necessary step.

5. SUMMARY

These are the initial results for AEGIS and we are currently in the process of gathering more detailed results and transcripts from a number of other case studies.

From the evidence gathered so far, AEGIS has proved to be approachable, engaging and simple to use. Through the application of AEGIS, EGSO also identified a number of problems and instituted a number of key changes:

- No one was explicitly in charge of ensuring the project was secure
- Little work was done to approach security systematically
- There was little coordination between the project and the institutions that run the project regarding security
- We identified and modelled main assets
- We identified and documented security requirements
- We identified many areas which forced the project to look at their implicit assumptions.
- We identified the need to document policy for a large number of areas: backup, data update and integrity checking, administrator duties, expansion of EGSO to other providers and acceptable use

There is evidence that this process also improved the developers' and researchers' knowledge about security. We also believe that the inclusion of contextual information has highlighted the need to document and regulate specific duties of human personnel in the system that other security methodologies would have overlooked in favour of technical issues.

6. CONCLUSIONS

Although research in the usability of security is ongoing, we believe there is a need to address the problems developers face when building secure systems. They require help to overcome the complexity of applying good security and designing usable systems at the same time.

In response to this, we have presented AEGIS, a lightweight approach improving the usability of a secure development method as well as providing security decision makers with increased awareness of user context.

Although this method is not necessarily as comprehensive in its technical coverage of security when compared to other methodologies, we believe it is the first to actively involve user information in security decisions.

Our case studies have shown that this method is well received, useful and approachable, having in many cases resulted in a more comprehensive and structured approach to security. We are currently expanding the number of projects we are working with as a result. As part of our review of AEGIS, we intend to extend and refine our methods in order to provide more extensive support for the risk analysis and security design phases.

We envisage future work to involve identifying common security requirements and linking them to the appropriate security design patterns [2] as well as improving tool support.

7. ACKNOWLEDGEMENTS

We gratefully acknowledge the contributions of the NSPW 2003 participants and NSPW reviewers, who helped to improve the earlier version of the paper. Ivan Flechais is funded through an EPSRC CASE studentship with BT Labs.

8. REFERENCES

[1] *CERT.* http://www.cert.org

[2] *Security Patterns.* http://www.securitypatterns.org/

[3] *Seti@home.* http://setiathome.ssl.berkeley.edu

[4] @stake. *The Security of Applications: Not All Are Created Equal.* http://www.atstake.com. 2002.

[5] Abrams, M. D. *Security Engineering in an Evolutionary Acquisition Environment.* New Security Paradigms Workshop 1998.

[6] Adams, A. & Sasse, M. A. *Users Are Not The Enemy.* Communications of the ACM 1999. Vol. 42, No. 12 December

[7] Adams, J. *Risk.* 1995. UCL Press.

[8] Adams, J. & Thompson, M. *Taking account of societal concerns about risk: framing the problem.* Health and Safety Executive. Research Report 035 2002. http://www.geog.ucl.ac.uk/~jadams/publish.htm

[9] Beyer, H. & Holtzblatt, K. *Contextual Design : Defining Customer-Centered Systems.* 1998. Morgan Kaufmann Publishers, Inc.

[10] Blakley, B., McDermott, E., & Geer, D. *Information Security is Information Risk Management.* New Security Paradigms Workshop 2001. pp 97-104.

[11] Boehm, B. W. *A spiral model of software development and Enhancement.* IEEE Computer 1988. 21(5) , pp 61-72.

[12] Brostoff, S. & Sasse, M. A. *Safe and Sound: a safety-critical approach to security design.* New Security Paradigms Workshop 2001.

[13] Darley, J. M. & Latané, B. *Norms and normative behaviour: field studies of social interdependence.* Altruism and Helping Behaviour. 1970. New York: Academic Press. J.Macauley & L.Berkowitz (eds).

[14] Grygus, A. *2003 And Beyond.* http://www.aaxnet.com/editor/edit029.html. 2003.

[15] Herrmann, P. & Krumm, H. *Object-Oriented Security Analysis and Modeling.* Proceedings of the 9th International Conference on Telecommunication Systems - Modeling and Analysis 2001. pp 21-32.

[16] McDermott, J. P. & Fox, C. *Using Abuse Case Models for Security Requirements Analysis.* Proceedings of the 15th Annual Computer Security Applications Conference (ACSAC'99), Phoenix 1999. pp 55-67. IEEE Computer Society Press.

[17] Mitnick, K. D. & Simon, W. L. *The Art of Deception: Controlling the Human Element of Security.* 2002. Wiley Publishing Inc.

[18] Sasse, M. A., Brostoff, S., & Weirich, D. *Transforming the 'weakest link': a human-computer interaction approach to usable and effective security.* BT Technical Journal 2001. 19 , pp 122-131.

[19] Viega, J. & McGraw, G. *Building Secure Software.* 2002. Addison-Wesley.

[20] Weirich, D. & Sasse, M. A. *Pretty Good Persuasion: A first step towards effective password security in the real world.* New Security Paradigms Workshop 2001.

[21] Whitten, A. & Tygar, J. D. *Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0.* Proceedings of the 8th USENIX Security Symposium, August 1999, Washington 1999.

[22] Zurko, M. E., Simon, R., & Sanfilippo, T. *A User Centered, Modular Authorization Service Built on an RBAC Foundation .* IEEE 1999.