Chapter 11

Security implications of structure

Jeremy Bryans and Budi Arief

University of Newcastle upon Tyne

1 Introduction

Computer security is an important issue in determining the dependability of computer systems. It becomes even more crucial when we talk about *computer-based systems (CBS)*, where we take into consideration the roles played by the human actors (or human components) involved in the system.

In this chapter, we begin to explore the security of complex CBS (sometimes called *socio-technical systems*). We do this by putting forward a common structuring abstraction for technical systems (that of *component-based systems*), then extending this abstraction to computer-based systems, in order to take into account the socio-technical structure of the system.

Section 2 introduces some basic notions of computer security largely developed within the technical domain, and in Section 2.2 we look at a well known model of how these systems are protected (the Swiss Cheese model [9]). In Section 3 we consider more closely the component-based architecture, and consider how well this architectural model copes with introducing people as components. The security implications of this architectural model are presented in Section 3.3, together with a new diagrammatic representation of the model, and an attempt to adapt Reason's Swiss Cheese model to socio-technical systems. A short discussion of socio-technical security policies is presented in Section 4, and we conclude in Section 5.

2 Security basics

Security is about protecting assets. In a computer system, these assets are things like information, processing power or hardware. In a computer-based system, this list must be extended to include more ethereal notions, such as trust.

Traditionally, the ways in which the assets of any system may be compromised are frequently grouped into three aspects: *confidentiality*, *integrity* and *availability*.

Confidentiality is broken if information is lost. Losing information has one significant difference from losing a physical artifact. Information can be lost if it is copied,

and if a copy is made, the rightful owner will not necessarily know that it has been lost. In the physical world items are protected by placing barriers in the way of would-be wrong-doers. Safes, walls and fences provide obstacles between the items to be protected and the outside world. In the electronic world these barriers are realised by things such as firewalls, password protected systems and encrypted files.

Integrity is broken if information is corrupted. This might mean that information is destroyed altogether, for example a wiped hard disk or a deleted file. It can also be more subtle. Information may have the correct form, but in fact be an inaccurate depiction of reality. Backing up information and keeping multiple copies are simple means of protecting integrity. This is achieved by restoring an uncorrupted version of the information.

Availability is compromised if access to information or services is lost. Protecting availability is therefore about protecting the channels through which this information or services are accessed and received, as well as ensuring that the processing power and data is present when the information or services are requested. A *denial of service* is an attack on availability, where the attacker may make so many false requests to a server that it is unable to process the true requests.

2.1 Typical security protection

In order to avoid security compromises, certain security measures are usually applied to systems. It is virtually impossible to have a "totally secure system" without sacrificing the usability of that system. For example, a stand alone computer that is not connected to any network and placed in a locked room might be secure, but it will not be useful if it is meant to serve other systems (or people) distributed over multiple locations.

Security measures are therefore employed to provide an acceptable level of protection, based on the purpose of the system and the perceived security threats that this system is going to face. Some of the most common security measures are:

- *Firewalls*
  A firewall acts as a "filter" at the boundary of the system. It is designed to let certain traffic through and prevent the rest of the traffic (be it benign or malicious) getting into the system. It can also perform address translation for networks so that the internal configuration details of a network are hidden from the outsider.
- *Intrusion Detection Systems*
  Intrusion detection is the process of monitoring the events occurring in a computer system or network, and by analysing these events, signs of security problems can be detected [3]. Due to the amount of traffic that computer networks carry these days, it is necessary to have tool support to analyse the data. This is the main purpose of an *Intrusion Detection System* (IDS). It records the stream of events on a network, analyses the data to find tell-tale signs of intrusion and reports to the system administrator who can take the appropriate action. Some IDS can even automatically perform emergency action when an intrusion is detected.
- *Passwords*
  A password is one of the most basic and most common protection mechanisms. It is usually the last line of defence to the system, therefore it is imperative to

have a strong password – i.e. a password that is not easy to guess or to crack. These days people tend to use multiple systems that are password protected, such as their desktop computer, internet banking accounts and web-based email accounts. This poses a threat as people cannot remember all their passwords; they might write them down or choose easy to remember but weak passwords. This problem is highlighted in [2]. Alternative versions of password protection exist, for example biometric passes (fingerprint or retina scan) or graphical passwords.

This is not an exhaustive list, there are many more measures that can be taken to ensure that the system in question remains secure. In most cases, several security measures are used to protect the system, as illustrated in Section 2.2. There is also a need to employ and enforce *security policies* in order to make these measures effective. More discussion on security policies can be found in Section 4.

2.2 Security layers and fault-tolerance

Security can be seen as an "all or nothing" property. Attackers must be kept from having any impact on the system whatsoever. In most cases, however, security protection is composed of several structured layers, protecting different levels of the system. So for example user-specific security such as a firewall is provided at the outermost boundary of a system, and the innermost layer of a system finds more general security mechanisms such as passwords and operating system checks. This is comparable to James Reason's Swiss Cheese model [9], where each layer provides protection from certain types of attacks but has weaknesses (represented as holes) against other types (see Fig. 1). Security breaches happen when holes on these layers are aligned, allowing attackers to penetrate every layer of protection.

Fault-tolerance is about error detection, containment and recovery. Error containment can be seen in structural terms, with potential sources of error within a system being contained by the (hardware or software) structures. Error containment is about not letting errors out. The same applies in reverse for security: it is about not letting malicious "errors" in.

These two approaches (all-or-nothing and fault-tolerance) lead to the development of quite different systems. Thinking about the two approaches in structural terms can help to understand the resultant systems.

Any physical thing protected by "all-or-nothing" security will have big, strong, obvious defences. A castle is an obvious example. To the serfs outside, the castle walls are unbreakable.

A fault-tolerance mindset leads to "absorbent" security. A physical thing protected by absorbent security will be surrounded by a number of layers of security. These will be designed to protect against different types of threats, so that an enemy finding it easy to break one layer will be likely to have difficulty in breaking another. Rather than defending the outer boundary of the system at all costs, an attack is "absorbed". Ideally, it is detected as it happens using an intrusion detection system, so that the potential damage is limited (possibly by restricting the access rights of the intruder) and any damage that has been done is identified and fixed.

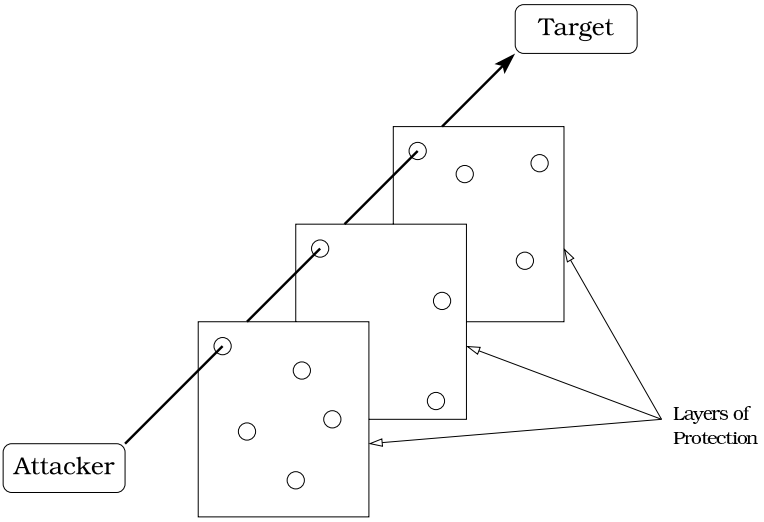

Fig. 1. Description of a successful attack within Reason's Swiss Cheese model

3 Structure and socio-technical security of CBS

Computer systems are built up of *components*, which may themselves be systems. In many cases, these components are *commercial-off-the-shelf (COTS)* products with their own identifiable function. They also have *boundaries*, and some means of interaction with surrounding components or systems [8].

In this section we will use this view of the structure of technical systems to examine the structure and the security of socio-technical systems. We begin with a brief recap of the technical perspective, then introduce humans as components, and examine the implications of this. We then consider the security implications for the socio-technical system.

3.1 Components and boundaries

As mentioned above, a component in a system has a particular *function* or *purpose*. For a fully automated component, this function could, for example, be expressed in terms of the communications that are allowed to flow across the boundary of that component, or in terms of the relationship between the input and the output of the component.

There is an implicit assumption in the above that a component always has the same function, even if it is a member of more than one system. Different systems built using a particular COTS component will contain replicas of the component, but in each case the same functionality is being used. Of course it is possible that in different systems, completely different subsets of the functionality of a component are being used, so much so that the two replicas may be behaving in completely different ways, but the underlying functionality is still the same.

When these components are combined into one system, the boundary of the resultant system encompasses all the boundaries of its components, which could lead to some interesting security issues, as discussed below. Security is about protecting boundaries around these components, both from having private information flowing from "inside" to "outside" and from attacker (could be outsider or insider) gaining full access (read, write, modify, delete) to the information.

The interconnected nature of these components makes it more difficult to secure the whole system. This is because it opens up the boundary to another level, where a breach in one component might lead to further breach in other components or even the whole system.

When we consider socio-technical systems, issues on boundaries still exist, but we now have to consider a very different form of component: people. This immediately leads to two new types of interface or boundary to consider: the *person–machine* boundary and the *person–person* boundary. We begin by considering people as components of a system.

3.2 People as components

When we consider people as components, the assumption that they have the same functionality or purpose breaks down. People have more than one purpose. Even within the same organization, one person can be a systems administrator, a sales executive and a CEO at the same time. These purposes do not readily succumb to being described as mathematical functions. Furthermore, people are able to change their behaviour (and indeed purpose) according to the situation (see Chapter 5), in a way that programmed components cannot.

A single over-arching purpose could be deduced by grouping these many purposes together, but they may conflict with each other, and no single obvious solution may be possible. It may be possible to arrange a set of purposes in order of priority, or to give explicit rules for every possible conflict situation, although each of these routes is fraught with difficulty. This flexibility is the basis for the dependability of many long-lasting socio-technical systems, but unanticipated behaviours can also lead to the most severe security and dependability breaches.

When we dig a little deeper, a more difficult problem arises. This has been hinted at earlier, and is the question of motive. A legitimate user within a socio-technical system may have many complex and even contradictory motives, and even a single motive may result in opposite behaviours. A motive as seemingly simple as wishing to keep their job might make an employee work hard to become expert in a given area but it might also make them reluctant to share their expertise with junior members of their team, for fear that they themselves become expendable.

People also tend to be creative: they find work-arounds to certain restrictions that were enforced to improve security. In most cases, improving security means more effort or less flexibility to people. They do not always like the idea, and if they can find a way to bypass this and make their life easier, they will do so.

person–machine boundary

Security at the person–machine interface is a small but growing area of research [1; 4; 10; 11].

In [4], the authors concentrate on the threat to security posed by the "legitimate user": one who is properly part of the socio-technical system. Beginning with Simon's concept of *bounded rationality* [12] the authors argue that a user must take a number of factors into account at any time. The authors go on to look at the trade-off the legitimate user makes between *usability* and security. Usability comes in a number of guises: remembering passwords, maintaining anti-virus protection and sharing files are all cited as usability examples.

If a user feels that a particular security-related activity is not worth the additional effort imposed, then it will not get done. This statement, however, opens many more questions than it answers. We have to define and measure "effort", and the cost of not performing the action, and then compare these to each other. Effort may be measurable in specific situations (such as in mouse-clicks or by time expended), and the security cost of not performing the action could perhaps be measured as a product of risk and consequence. Risk would be measured using some form of probabilistic measure. Consequence could be measured in terms of money lost (through, for example, downtime or repairing data loss). Forming a legitimate basis for comparison between these measures is by no means obvious. On top of this, humans are notoriously bad at making estimates of risk, so any effort to investigate this trade-off is fraught with difficulties.

person–person boundary

People interact with other people but such interactions tend to be less predictable than those of machine–machine or person–machine.

People have weaknesses that could be exploited through some psychological manipulation. We often hear about *social engineering* [5; 6] being used to breach security protection. Social engineering is the term used to describe breaking-in techniques that rely on weaknesses in the human components attached to the system (such as system administrators, operators, secretaries, etc.) instead of the vulnerabilities of the machine components (software bugs, configuration mismatch, etc.).

The aim of social engineering is to trick people into revealing passwords or other information that compromises a target system's security. For example, an attacker might phone the system's operator (who has the required information), posing as a field service technician or a fellow employee with an urgent access problem. By telling the operator a carefully-crafted and convincing story, the attacker manages to get the operator to reveal the desired information, which could be about the system's security vulnerabilities or even the password to get into the system.

There are various methods to perform social engineering (as described in [5]): false authority, impersonation, sympathy, personal stake, boosting egos, inconspicuous occupation, and reward. It is scary how effective social engineering can be, as illustrated in detail in [6].

The way people treat other people could also – albeit indirectly – lead to security problems. For example, an employee who is harassed or bullied by their colleagues at work, or even if they are just not happy with the work environment, might want to take

revenge against the whole organisation. This could manifest itself in a situation where this employee leaks out some sensitive information to outside parties, or they might cause havoc by deleting important information.

The examples given above show that it is unwise to ignore person–machine and person–person boundaries when we talk about socio-technical systems and their impacts on security. How these, along with machine–machine boundary fall into place in the overall system will be discussed in the next section.

3.3 Overall picture and security implications

The presence of humans in socio-technical systems affects the way information is accessed in those systems. We revisit the idea of "system boundary" mentioned in Section 3.1, where interfaces are placed on the boundary to allow interaction between the components inside the system and the outside world. Here we introduce the possibility of a "hole" (or even multiple holes) appearing on the boundary. A hole represents a point on the boundary where an illegitimate communication channel might appear. In terms of technical security, a hole could be a bug in the software that allows an authorised access to the information. In other words, a hole represents a security weakness point that could be exploited by an attacker.

Within the boundary of a system, there could be many components. These components can be machine or human components – or even sub-systems, each with their own (smaller) boundary, which has interfaces and possibly holes.

The overall picture can be seen in Fig. 2. Here, **A** represents a normal interaction channel. Access to the system and its internal components are allowed through predefined interfaces. **B** represents a situation where an attacker exploits a hole on the system boundary to make it appear that they have permission to access a component within the system. The hole could be a weak password that the attacker could easily guess or crack. **C** is similar to **B** but this time the attacker exploits holes on both system boundary and component (sub)boundary. An example could be a weak password coupled with a lack of security control in resources/network sharing. **D** depicts a scenario where an attacker uses an interface on the boundary to get into a human component, and then exploits the human weaknesses in order to gain illegal access to a machine component. This is usually what happens with a *social engineering* attack where the attacker uses the phone to dupe a human component (a system administrator, an operator, a secretary, etc.) into giving them access which might then seem to be legitimate from that point on. **E** shows the possibility of an "insider attack" where someone with a certain level of permission to the system gains access to other parts of the system to which they do not have access rights. **F** reminds us that there could be more holes in the system that have not been exploited yet.

Ideally, most – if not all – of a system's existing holes are identified and consequently some security measures such as firewalls, anti-virus software and security policies are applied to patch these holes. Unfortunately, this does not happen all the time, or in some cases, these measures are not deployed properly. Even worse, human components (e.g. system's users) might make some "adjustments" that render the security measures useless. For example, they might find that the firewall blocks certain legitimate traffic that is necessary for their work. As a consequence, they might disable

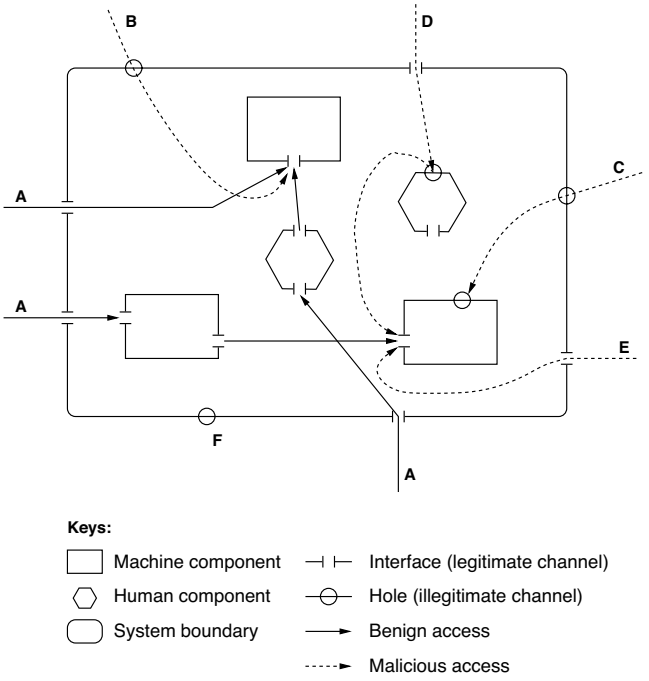

Fig. 2. Security boundary with interfaces and holes

the firewall completely to allow this traffic. This could happen especially if the users do not realise the significance of having a secure system or they do not know how to configure the security measures properly.

Another way to look at the impact on security when we have human components in a system is by revisiting the Swiss Cheese model (Fig. 1) described in Section 2.2. Taking the human factor into the equation, we can adapt this model slightly.

People can potentially improve system security. They are able to observe anomalous behaviours, which might be otherwise left un-noticed by a machine. One classic example is when Clifford Stoll detected a slight mismatch on the accounting report of the system he managed, which led to an investigation revealing security breaches that had been going on undetected for some time [13]. Unplugging the system from the network when some suspicious activity is detected is another example. This simple and crude method might not be advisable on certain systems, but nonetheless, it could prevent further damage to the system until proper recovery actions can be taken.

On the down side, humans could act as the weakest link when it comes to security. Human nature and tendency – for example their willingness to help others or their predictable behaviour under pressure – are often exploited by attackers through social engineering. When this happens, the consequences can be disastrous. An attacker could trick someone into giving them access to a system, hence rendering the rest of the security measures useless. Fig. 3 depicts a possible adapted version of the Swiss

Cheese model when human components are taken into consideration. In this example, an attacker uses social engineering to obtain the password of the target system. Once the password protection is compromised, the attacker can bypass the rest of the technological layers of protection, such as firewalls and intrusion detection systems. This example is comparable to point **D** in Fig. 2.

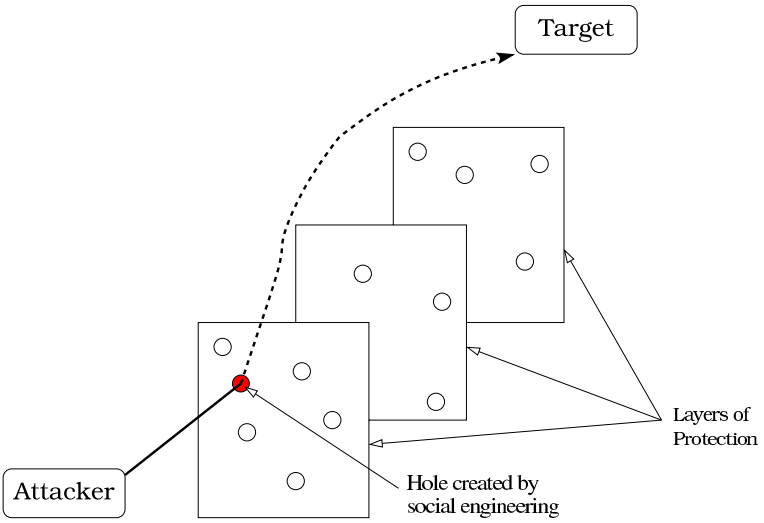

Fig. 3. Possible adapted version of the Swiss Cheese model

4 Socio-technical security policies

A security policy sets out the means by which an organisation hopes to secure its assets. It can be many different things, ranging from a vague wish-list through to a detailed set of rules.

According to [7], a good security policy should comprise both goals and rules. Goals capture the security requirements, such that a violation of the goals constitutes a failure. Security rules are lower level constraints on the behaviour of a system designed to ensure that the system is and stays secure. Further, the rules of a security policy should imply the goals of the security policy. Provided the rules are obeyed, no security failures should be able to occur.

To be effective, the rules of a security policy must take into account the nature of the system and components that it is securing, and must make reference to the particular threats that the system and its components face.

A security policy for a computer-based system must therefore take into account the nature of its components. The mechanical components of a computer system will do –

more or less – what they are programmed to do, but people are not so controllable. Security steps will be bypassed through both ignorance and expediency. There is seldom sufficient incentive for a user to take attention away from his or her primary job in order to pay attention to security. In [4], it is argued that the crucial issue is the perceived trade-off between effort and risk, and this agrees with our analysis here. Users bypass security either because their evaluation of this trade-off is wrong (they miscalculate the risk and effort involved) or because they make this evaluation and conclude that the security breach is "worth the risk".

The analysis above makes it seem as though users go through a fully deterministic, completely explicit cognitive process to arrive at their decision. Of course this is not always the case. Determining exactly how the users come to these conclusions is probably best explored experimentally.

From the point of view of people designing socio-technical policies, it seems to be important that legitimate users be able to come to an accurate measure of risk. Rather than give people a long list of simple rules to apply (as one would a computer component) they should be made to understand the reason for and the importance of the security measures they are being asked to implement. For example, users should understand what attackers might achieve if they learned a particular password. This will give users greater incentive to protect the password and foil social engineering attacks designed to extract this password. For people with the best interests of the wider system at heart, this should be sufficient.

Not everyone has the best interests of the wider system at heart, possibly including people within the organisation itself. To guard against a failure here, a policy should artificially inflate the risk to the individual of failing to keep to a policy. The social structure of the organisation in question may impose restrictions on this aspect of the security policy. It would be easier to achieve in a more regimented environment such as a military establishment, where people can be punished for failing to keep to the policy. In a less strict environment a similar effect could be achieved by for example, rewarding people who follow the policy correctly.

5 Conclusion

Human involvement in any system is unavoidable, and will critically influence the structure and security of the system, making it unpredictable and therefore hard to study. To understand how these socio-technical systems behave, we need to better understand the behaviour of people. This will lead to a better design of security measures in term of usability and effectiveness. As a result, the risk of human components bypassing or rendering the security measures useless through their careless actions could be reduced.

Another way to improve the security of computer-based systems is by making the human components aware of the importance of sound security practices and the havoc that security breaches could bring. It is very common – if not mandatory – for new employees to undergo safety training or induction. This could be extended to include *security induction*, where new employees are made aware of the organisation's security policies.

References

[1] Adams A, Sasse MA (1999). Users are not the enemy. Communications of the ACM, 42(12):40–46.
[2] Adams A, Sasse MA, Lunt P (1997). Making passwords secure and usable. In Proceedings of HCI'97 People and Computers XII, pages 1–19. Springer.
[3] Bace RG (2000). Intrusion Detection. Macmillan Technical Publishing.
[4] Besnard D, Arief B (2004). Computer security impaired by legitimate users. Computers & Security, 23(3):253–264.
[5] Hatch B, Lee J, Kurtz G (2001). Hacking Linux Exposed: Linux Security Secrets & Solutions. Osborne/McGraw-Hill.
[6] Mitnick K, Simon W (2002). The Art of Deception: Controlling the Human Element of Security. Wiley.
[7] Powell D, (Editors) RS (2003). Conceptual model and architecture of MAFTIA. Technical Report MAFTIA Deliverable D21, Project IST-1999-11583.
[8] Randell B (2004). Dependability, structure and infrastructure. Technical Report CS-TR 877, University of Newcastle.
[9] Reason J (1990). Human Error. Cambridge University Press.
[10] Reeder R, Maxion R (2004). Error analysis of a security-oriented user interface. Technical Report 872, Newcastle University Computing Science.
[11] Sasse MA, Brostoff S, Weirich D (2001). Transforming the weakest link - a human computer interaction approach to usable effective security. BT Technological Journal, 19(3):122–131.
[12] Simon HA (1957). Models of Man. Wiley, New York.
[13] Stoll C (1989). The Cuckoo's Egg. Doubleday.