Making Safe Software Secure

Odd Nordland
SINTEF ICT
Trondheim, Norway

Abstract

Safety and security are traditionally treated separately, particular with respect to software. The increasing demand for remote access to computer controlled industrial systems requires merging the two fields. Some possible methods are described and possibilities for improving both safety and security related standards are presented.

1 Introduction

Nowadays it is almost unthinkable to control a complex technical process without using computers. They are used in nuclear power stations, chemical plants, rail and air traffic control and a vast number of industrial applications. In the past decades, extensive experience has been gathered in developing software that can perform the control tasks in a safe and reliable way; indeed, it is usually considered safer to let a computer perform a complex control task, rather than relying on fallible humans: the computer is faster, doesn't get tired, never forgets or overlooks anything and doesn't get distracted by its surroundings.

Unfortunately, the increasing reliance on computer systems for technical control makes us increasingly dependent on the computer's software, which in turn makes us vulnerable if the software gets modified in an undesirable way. Indeed, simply misusing otherwise correct software can become a threat.

The internet has shown how easy and effective it can be to connect to remote computers. Salespeople can access their company's data bases and place orders without having to return from their business trip first and home banking is widespread. On the other hand, the internet has also brought us the problems of hackers, viruses and spyware.

There is now a growing demand to use remote access not only to monitor but also to control processes in e.g. oil platforms, albeit not necessarily by the internet. But however it is done, remote access opens the door for hackers, virus programmers or terrorists who want to interfere with the process control; a possibility they didn't have earlier. So now it is not enough to have safe software in industrial control; it has to be secure too.

2 Safety and security

The terms safety and security are often mixed and there is a multitude of definitions for each of them. Meine van der Meulen (van der Meulen 2000)

collected four different definitions of "safety" and six different definitions of "security" from various standards, guidelines etc. Some standards even operate with multiple definitions, an indication that even the authors of the standard could not reach an agreement. The Internet lists a larger number of definitions for security than for safety, so there is apparently a greater consensus on the meaning of the word safety than on the meaning of the word security. However, a glance at those definitions reveals a considerable overlap, and very often the terms are even considered to be synonymous!

The problem is not specific to the English language. The Germanic languages have one word for both safety and security and a certain degree of difficulty in expressing the differences: it is widespread use the English words to indicate which meaning is intended, but that certainly doesn't contribute to clarification, more the opposite.

The problem is that a definition should be concise, complete and comprehensible. For the complexity of the concepts behind safety and security, this is almost impossible to achieve. The result is that somebody will always find an aspect that isn't covered by the definition. So rather than trying to produce yet another set of definitions, we shall use a somewhat more verbatim explanation of what is meant here.

We start by considering a technical system that we want to control. This means that the system can have an effect on its environment (i.e. the rest of the world), but it can also be affected by its environment. If the effects are undesired, we will try to prevent them from occurring.

The word "undesired" is important here: the effect of a bomb is to blow up a building, which is usually undesirable, particularly if you happen to be in that building. But it's but not undesired: the undesired effect is blowing up the wrong building!

The inability of the system to affect its environment in an undesired way is usually referred to as safety; the inability of the environment to affect the system in an undesired way is usually called security. Alternatively, safety and security are used to express attributes that are based on measures that aim at preventing the system from having an undesired effect on its environment respectively vice versa.

Now these explanations are intentionally kept very generic. Particularly with respect to security there is a tendency to limit the concept to e.g. protection against terrorism or other malicious acts. This is a too narrow view: an earthquake is not a malicious act, but it can certainly damage or destroy power plants, buildings or roads. Constructing buildings or roads to survive an earthquake is in fact a typical security measure, because it is protecting the building or road from being damaged by its environment. But it's also a safety measure, because by protecting a building from being damaged by an earthquake, we are also protecting the inhabitants of that building from being harmed by the building.

At this stage we see that safety and security are strongly coupled. Nancy Leveson (Leveson 1995, p.182) argued for keeping safety and security segregated, stating "*Although there are some commonalities and interactions, safety is a distinct quality and should be treated separately from other qualities, or the tradeoffs, which are often required, will be hidden from view.*" As we will see in the next section, the connection between safety and security has become so much

stronger that segregating the two can have the effect that mutual aspects can get completely overlooked, because neither side feels responsible.

3 Scenarios

Part of the problem is that it is possible to misuse a perfectly correct function of a technical system to achieve a totally undesired effect. A typical example is a Denial-of-Service attack against an internet service provider. The perfectly legitimate function of transferring data, e.g. by e-mail, is misused to flood the service provider's machines so that they are unable to provide their services to genuine customers.

Now such attacks will cause some economical damage to the service provider, and probably also to some of his customers, but human life will not usually be endangered. But there are other possibilities.

3.1 Rail Traffic Control

Nowadays railway networks make extensive use of computer systems to control and monitor rail traffic. A central element of such systems is the network of interconnected computers that control traffic and trackside equipment at stations and junctions. Trains that run on such networks are equipped with on-board computer systems that send data to, and receive and process data from, the interlocking computers. The information sent to the train includes data about the maximum permissible speed, the distance to the next signal, its status and so on. The on-board computer transmits data back to the interlocking computers, such as the train number, the train's position, speed and travelling direction etc.

The data transfer is achieved by radio signals: there are transponders, so called "balises", placed between the tracks at strategic positions. The trains have a transceiver on board, and when they pass over a balise they transmit their data to it. The energy they transmit is sufficient to activate the circuits of the transponder, which then transmits its data back to the train.

In such a situation, falsified data can have a catastrophic effect. If the train's on-board computer can be tricked into allowing maximum speed in a sharp curve or an area where construction works are going on, the result could be as terrible as the Eschede accident in Germany in 1998.

A lot of effort is invested in making the software capable of recognising and either correcting, or at least ignoring erroneous data, but the complexity of the communication mechanisms is usually considered sufficient to prevent intentional generation of bogus messages. However, terrorists or other criminals have plenty of money and technical facilities available, and an insider with sufficient knowledge of the system could possibly succeed in creating bogus messages with the aid of a sufficiently determined criminal organisation.

3.2 Remote maintenance

For systems in a hostile, or at least difficult to access environment, there is a growing demand to perform not only control, but also maintenance activities using remote control. An example is the oil platforms in the North Sea. It is an expensive

exercise to send maintenance staff out to each and every platform in order to update the software running on the computers. How much easier would it be to send the latest software update by radio link and do a remote installation? It would be faster too, since all platforms could be upgraded simultaneously.

But if the software gets modified during transmission, the result could be that once safe software becomes unsafe or even dangerous. We could end up with a complete loss of control with catastrophic effects for the people on board the platforms and for the environment. So preventing falsification of transmitted software becomes just as important as making sure that the transmitted software is safe.

3.3 Telemedicine

In sparsely populated areas, such as the far North of Norway, doctors are few and far between. Those that are available are, of necessity, general practitioners. For patients who need specialist treatment, the distance to the nearest specialist can be prohibitive. A possible solution is to use a specialist, located far away in a big hospital, as an online supervisor for the general practitioner. A video connection is needed, so that the specialist can see what his colleague is doing, and the data that is typically monitored electronically (such as pulse rate, blood pressure etc.) can be transmitted directly to the specialist. He, in turn, may remotely control some of the equipment, such as infusion pumps for example.

If the data that is exchanged gets tampered with, the remotely controlled equipment might start executing completely wrong treatment, possibly with a fatal effect. So the safe data, which is also part of the software on such equipment, must be protected.

4 Methods

The above scenarios are simple examples that illustrate that safe software must itself be protected. In the past, enormous efforts have been made to ensure that software that controls a technical process is safe. Now it's time to make it secure.

Security is an attribute that should be built in to the software from the outset. Many of the methods that have been developed in order to achieve safety can be adapted to security. Line et al. (Line 2006) discussed the similarities and differences between safety and security methods and argue "*... the analysis techniques and tools that are used to determine and classify hazards can easily be adapted to threats, and the techniques for analyzing threats can be adapted to hazards...*" Saglietti (Saglietti 2005) has also pointed out the potential in adapting safety techniques to incorporate security and states "*... it is felt that this question is becoming increasingly important and should be handled in a unified, systematic way by means of an extended fault tree analysis, capable of integrating security incidents as sub-events possibly leading to safety-related failures.*"

There is still a lot of work to be done before we get unified analysis techniques, and we cannot afford to wait for them. There is safe software in use today that needs to be protected in an ever changing and increasingly networked environment.

And that protection can be achieved by a combination of hardware, software and administrative procedures. Some possibilities are sketched below.

4.1 Hardware

Software can be protected by hardware! Obviously, segregating the machine from any form of network will also protect the software running on it. Where this is possible it should be done. Modifications to the software will then necessitate a data transfer by some form of removable data carrier, which in turn requires a conscious and controllable human activity. Falsification of the software will still be possible, but at a much higher risk for the person doing it. In addition, administrative procedures can contribute to preventing such acts of sabotage.

The use of "dongles" to prevent software piracy was never particularly popular, because it was and is inconvenient for the users and not least because a dongle cannot be downloaded: the hardware has to be delivered to the customer and physically connected to the computer that the software is running on.

For commercial off-the-shelf software that is intended for a mass market, dongles are probably more of a drawback than a benefit. But for dedicated safety critical software that is not subject to mass distribution, the technology could provide a high degree of protection against falsification of the executable code in a flexible and reliable way. The fact that a dongle is removable means that it can easily be replaced when the software is updated; it can also be used to make sure that only compatible versions of different programs are installed on the machine.

4.2 Software

The software security community has developed a multitude of technologies to protect data, albeit not necessarily with the protection of safe software in mind. But methods that have been developed to protect data integrity will also protect code integrity, so the task is not really new.

Line et al. (Line 2006) argue "*The techniques used in software safety have been around for quite some time and are well established and tested. Some of these techniques may be useful also for security people who may thus benefit from the experiences of the safety community. On the other hand, there are also security techniques that will become significant for the safety community. For example, in the near future we will see more use of open communication networks for remote control of industrial and transportation applications. When vitally important commands are transmitted through such open networks, security techniques such as encryption and authorization control will become indispensable for safety. Security techniques will have to become an integral part of safety thinking.*"

One challenge can be the timing aspects. Safety related software must often react rapidly and cannot afford to be slowed down by lengthy, time consuming verification, authentication or decryption procedures. A possible solution is to encapsulate the safety critical software in a secure shell.

Jaatun et al. (Jaatun 2007) describe an "onion model" to protect safety critical software in an offshore application. Basically, the system uses several layers of software surrounding the safety critical kernel; each layer has specific access options so that the further in a layer is, the more strongly it is protected. The safe

kernel can then perform its tasks without having to include security related activities.

4.3 Administrative procedures

Administrative procedures are often overlooked as a safety measure, whilst they are often considered for security. Typically, users will be required to use passwords that are changed regularly and are sufficiently long to make them difficult to crack. Requiring controlled authorisation should be a must for modifying safety critical software, but the fact is that it is not usually done. At best the computer should be put into maintenance mode or switched offline before the software is modified, but the technician who has to do the job does not normally need a password to start. And certainly not a set of passwords, depending on the kind of maintenance he has to do.

Even with a password controlled access, the technician will assume that the data he has been given is correct. One way of ensuring that the new data is correct could be to get it installed twice by two different technicians at two different times. If the two installations are not identical, the data has been modified.

5 Assessment

For both safety and security there are certification schemes based on independent third party assessments. However, the schemes are substantially different: for safety the standards identify requirements to be fulfilled in order to achieve a defined safety integrity level; for security the standards describe assessment methods to be applied in order to achieve a certain level of confidence, but they don't define the security requirements to be fulfilled.

5.1 Safety assessments

Depending on the application area there are often sector specific safety standards with sector specific safety requirements, but as far as software is concerned, IEC 61508 can be regarded as the "mother" of the safety standards. Indeed, IEC 61508 explicitly states that it "*...provides general requirements for ... safety-related systems where no application sector standards exist...*" One consequence of this is that many sectors, for example offshore, don't even try to develop sector specific standards and simply apply IEC 61508 directly. In those cases where sector specific standards have been developed, such as the railways, they at least claim to be a sector specific implementation of IEC 61508, albeit without providing any form for evidence! Nevertheless, the requirements from IEC 61508 for software can be used as a suitable example.

It should be noted that IEC 61508 also explicitly states that it "*... does not cover the precautions that may be necessary to prevent unauthorised persons damaging, and/or otherwise adversely affecting, the functional safety of ... safety-related systems.*" In other words, security is explicitly excluded.

The standard uses the concept of Safety Integrity Levels ("SIL") which was first introduced by the British Defence Standard 00-56 (MOD 1996). In spite of being over twenty years old, the concept is still poorly understood. The important

thing to understand is that SILs are related to safety critical functions, not to the components or equipment that are used to implement those functions. In particular, the failure rates that are used to define the different safety integrity levels refer to failures of the safety critical function, not to failure rates of the equipment.

Let us take a very simple example: a motor car has a hydraulic and a mechanical braking system. A failure of either the hydraulic or the mechanical braking system alone does not mean one has lost the braking function, so the failure rate of the braking function will be lower than the individual failure rates of either the hydraulic or the mechanical braking system.

For software the concept becomes somewhat more complicated. Software may possibly be the sole way of implementing a safety critical function, and it is virtually impossible to determine a failure rate for software. Ideally software - if it is perfectly correct - can never fail. But of course software is seldom, if ever, perfectly correct, so it can fail. Ultimately, the probability of the software containing faults that can lead to a failure is determined by the techniques and methods that have been applied when the software was being developed and throughout its entire life cycle. The standard identifies a variety of techniques and measures to be applied during the software life cycle, and it is then the task of an independent assessor to determine if and to what extent this has been done.

This process is independent of the safety integrity level, since the SIL is regarded as a property of the system being assessed, not of the assessment procedure. It is therefore not correct to say “a system is certified to SIL 3”; it IS a SIL 3 system in that it must fulfil the safety integrity requirements that are defined for SIL 3. The assessor can certify that the corresponding requirements either are or are not fulfilled, but a SIL 3 system cannot become a SIL 4 system without being modified, in which case the whole safety certification process has to be repeated.

5.2 Security assessments

For security, and in particular software security, criteria for assessing the software have been defined in the so-called Common Criteria. Together with the Common Evaluation Methodology they form a toolset for evaluation and certification of both software and hardware with respect to security.

The Common Criteria consist of three parts: part 1 is an introduction giving an overview of the standard, part 2 contains a listing of possible security functions and part 3 defines the Evaluation Assurance Levels (EAL). Although seven different EALs are defined, there are only guidelines on how to evaluate according to EALs 1 to 4, because the international community has not yet agreed upon the process and techniques for evaluation according to the higher EALs.

There are attempts to define some kind of correspondence between SILs and EALs. This simply reveals how poorly the concepts are understood. EALs say something about the degree of confidence one can have in the results of an assessment, and it is no problem to “upgrade” a system’s EAL by simply re-assessing it. That hasn’t made the system more secure, it has only increased our confidence in its security. Whether a SIL 4 system that is certified to EAL 1 is better than a SIL 3 system that is certified to EAL 4 is currently not determinable.

6 Future work

There is a lot to do before we can claim that safe software is secure. Certainly existing safety related software can and should be given improved protection, and the foregoing text has given some examples of how this can be done. It is important to note that making already existing safe software secure does not have to compromise the safe functionality of that software in any way. And when suitable standards have been agreed upon, it will be possible to certify that safe software is also secure.

As mentioned earlier, there are both safety and security standards for software. The safety standards define safety integrity requirements, but leave it entirely up to the assessor to determine how to evaluate whether those requirements are adequately fulfilled or not. For security the opposite is the case: the standards define how to evaluate a system, but they don't define "security integrity levels". Clearly, there is room for synergy here!

As a first step, the safety community should look at the security standards and agree on assessment methods for safety. There are already several guidelines, such as (HSE 1992, MOD 1996, Pygott et al. 1999 or HSE 2003), so the work has, in fact, been begun. Now the community has to agree on a set of common evaluation criteria for safety related software.

At the same time, security levels should be defined for safety related software. Clearly, we will have different demands for the security of e.g. banking software and simple internet browsers, so there is already a degree of implicit categorisation. Here again, the international community must agree on how to assign security functions to well defined categories and how to allocate a security category for a given application.

When those two tasks have been done, it will be possible to start merging the safety and security standards. The result could still be two standards, but this time one standard would be defining safety and security requirements while the other would be defining assessment methods and criteria. And then the speakers of Germanic languages will be able to retain their single word for safety and security.

7 References

HSE (1992). Safety Assessment Principles for Nuclear Plants. HSE Books, Sudbury, 1992.

HSE (2003). Safety Case Assessment Manual. HSE Books, Sudbury, 2003.

Jaatun M G, Grøtan T O, Line M B (2007). Secure Safety: Good Practice for Secure Remote Access to Critical Safety Systems In Offshore Installations. Private communication to be published later.

Leveson N (1995). Safeware: system safety and computers. Addison-Wesley, Reading, Massachusetts, 1995

Line M B, Nordland O, Røstad L, Tøndel I A (2006). Safety vs. Security? In: Stamatelatos M G and Blackman H S (Eds.) Proceedings of the Eighth International Conference on Probabilistic Safety Assessment and management - PSAM 8. ASME Press, New York, 2006, ../pdfs/track-10/PSAM-0148.pdf

Ministry of Defence (1996). Defence Standard 00-56: Safety Management Requirements for Defence Systems. Her Majesty's Stationary Office, London, 1996

Pygott C, Furze R, Thompson I and Kelly C (1999). WP5 Final Report, Safety Case Assessment Approach for Air Traffic Management (ATM). ATM System Criticality Raises Issues in Balancing Actors' Responsibility (ARIBA). DERA, 1999

Saglietti F (2005). Assessing Software Safety and Security for Critical Infrastructures. European CIIP Newsletter 2005 Vol. 1 No. 2, pp 30-33

van der Meulen M J P (2000). Definitions for Hardware and Software Safety Engineers. Springer Verlag, London, 2000

http://www.springer.com/978-1-84800-099-5

Improvements in System Safety
Proceedings of the Sixteenth Safety-critical Systems Symposium, Bristol, UK, 5-7 February 2008
Redmill, F.; Anderson, T. (Eds.)
2008, X, 266 p. 42 illus., Softcover
ISBN: 978-1-84800-099-5