Bridging the safety-security software gap

C. W. Axelrod
Delta Risk LLC, USA

Abstract

Software security and safety engineers live in different and often separate worlds. The former professionals worry about protecting information-processing systems and data from attacks. The latter are concerned with potential harm inflicted by malfunctioning or failed industrial control systems (ICSs).

Some researchers, such as Joseph Weiss, have addressed the need to have security built into industrial control systems, including safety systems. Weiss attributes the general lack of security for ICSs to a "hole ... in academia" since "security is taught in computer science departments, whereas control systems are taught in various engineering departments." Others have expressed concern about who might be liable when information and control systems, are combined, as in autonomous (driverless) vehicles.

However, many issues relating to combined security and safety systems are much broader and more critical than the above. In this paper, which is based on his recent book, Axelrod takes a holistic view of the consequences of integrating security-critical information systems and networks with safety-critical control systems, such as those systems related to avionics, electricity grids, nuclear power plants, weapons systems, and the like. It is not sufficient to train software engineers about securing control systems. It is also necessary that security professionals gain a greater understanding of the control systems to which their information systems are increasingly being connected. This two-way exchange of ideas and approaches is crucial for ensuring that systems, which combine both security-critical and safety-critical components, meet standards and certification requirements.

Keywords: cybersecurity, safe software systems, cyber-physical systems, systems of systems, industrial-control systems, systems engineering, software engineering, computational systems.

1 Introduction

Software engineers, considering security or safety to be within their purview, come from markedly different backgrounds. Those dealing primarily with cybersecurity worry about protecting information-processing systems and data from externally-generated and internally-originated attacks. Software professionals, with responsibility for software safety, are mostly concerned with the potential level of harm inflicted by malfunctions or failures of industrial control systems (ICSs).

In today's world of complex cyber-physical systems and systems of systems, there is a rapidly increasing need to somehow combine skills and knowledge in both the software security and safety arenas. Weiss [1] attributes much of the divergence in viewpoint to the teaching of cybersecurity in computer science departments and of control-safety in "various engineering departments." This author was an electrical engineering undergraduate student at the University of Glasgow in the 1960s and subsequently a doctoral candidate at Cornell University. Both information systems and control systems courses were included in the various curricula. Granted, computer security was at its very early stages at the time. Nevertheless, while we see increasing specialization, the need to obtain a broader systems education still holds and various academic institutions have begun to offer courses and degrees in software system assurance and other cybersecurity topics. For example, this author contributed to a curriculum in software assurance [3].

In his recent book, Axelrod [2] investigates the differences between software systems safety and security and he suggests how software engineers might become aware of, and capable in, the knowledge and skills needed to ensure that software systems are both safe and secure. This paper focuses on these areas.

2 Definitions

To a significant degree, attempts to differentiate between software system security and safety have been stymied by a lack of consistency of definitions and understanding within the systems engineering field. Progress has been further hampered by the fact that newer fields, such as software engineering, have not yet fully adopted the techniques and discipline of their more mature parents.

In Figure 1, we illustrate a structure and hierarchy for systems engineering. The diagram shows systems engineering to consist of a number of elements including people, processes and technology. For our purposes we look at technology as comprising hardware and software, and within software we concentrate on so-called non-functional characteristics, which include security and safety, each of which has its own branch of engineering. Engineering is made up of management and assurance activities. The shaded squares illustrate how the diagram might be used to come up with appropriate terminology, as in software security engineering. While all interconnecting arrows have not been included in the diagram for the sake of clarity, the hierarchy can still be used for a term such as hardware safety engineering, for example. It should be noted that if one does not use such a hierarchy, then one might end up with a term such as security software engineering, which has a completely different meaning from software security engineering.

To be clear, the area that we are addressing here is the safety and security of software systems, including software and/or firmware embedded in physical equipment. This difference between information-processing systems and embedded software is a source of confusion, particularly when it comes to cyber-physical systems.

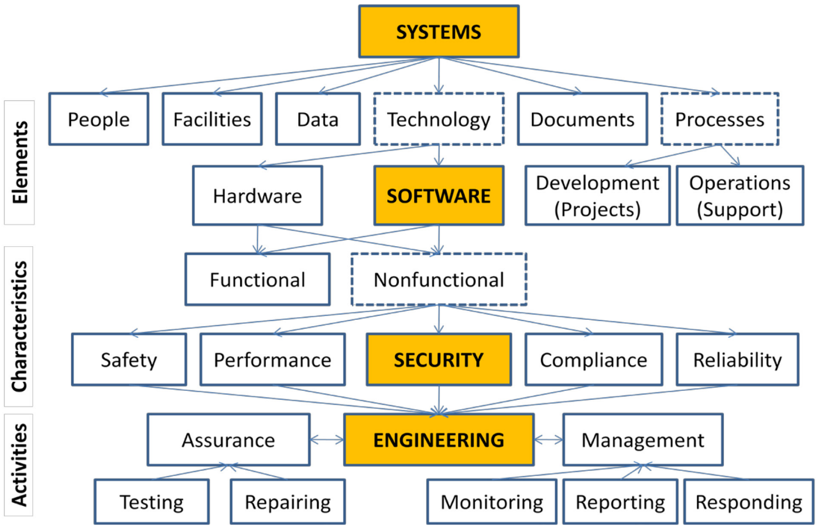

Figure 1: Structure and hierarchies of systems engineering, (adapted from: C. W. Axelrod, *Engineering Safe and Secure Software Systems*, © 2013 Artech House).

One definition of the difference between cyber and physical components of cyber-physical systems is illustrated in Figure 2.

As shown in Figure 2, there are essentially two types of software system within this broader definition of cyber-physical systems, namely, data-processing software, which usually runs on general-purpose platforms, and computational software, which controls and manages physical systems, which is often specific to a particular platform. The latter is termed "embedded software." The original definition of cyber-physical systems, as given by the National Science Foundation [4], covers only the software and equipment on the right-hand side of Figure 2. Recently, as distributed and network information processing systems, shown on the left-hand side of Figure 2, are increasingly interconnected to industrial-control systems. Such systems of systems, which include the smart grid, are being called cyber-physical systems. It is the latter, which is represented by all of Figure 2, which we consider here. It should be noted that computer hardware is also physical but is differentiated from ICS equipment for the purposes of this paper.

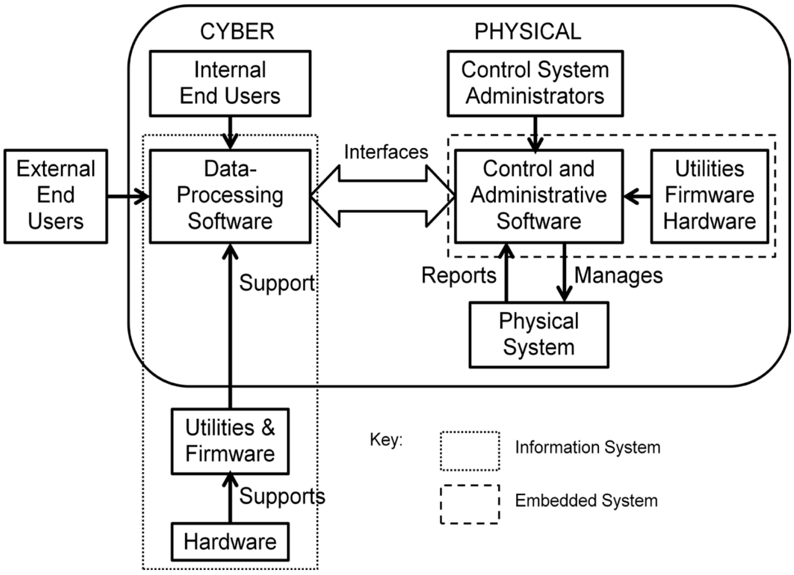

Figure 2: Cyber and physical components of cyber-physical systems, (source: C. W. Axelrod, "Managing the Risks of Cyber-Physical Systems", *IEEE LISAT Conference*, Farmingdale, New York, May 2013, IEEE).

2.1 Safety vs. security

The words "safety" and "security" are often used interchangeably. This can be confusing, particularly when one is trying to distinguish between the different cultures and objectives of software engineers working on cyber-physical systems. Somewhat restricted, but useful, definitions of safety-critical and security-critical software systems from Boehm [5] are as follows:

- Safety: The system must not harm the world
- Security: The world must not harm the system

This difference in perspective is a major contributor to the scism between software safety and security engineers, as discussed below. The goal of this paper is to suggest how engineers from each of these two silos might improve communications betweeen the groups and share knowledge, processes, skills and tools with each other.

3 The security perspective

As mentioned above, information security professionals are generally focussed on the protection of data-processing systems and communicatons networks against external attacks and nefarious insider activities, any of which might lead to misuse, damage or destruction to the processing systems and/or the compromise of applications and data in order to steal sensitive information with a view to fraud, identity theft, and other crimes. The common mantra of information security engineers is C-I-A (confidentiality, integrity, availability). In practice, most attention is paid to confidentiality, which includes security, privacy and secrecy, as well as to incident response.

Systems and data protection is achieved mainly through prevention, deterrence and avoidance. Security incident response usually involves monitoring, detection, reporting, response, recovery and reconstitution.

As a result, there has been a veritable proliferation of detection and prevention tools, such as firewalls, intrusion detection systems (IDSs) and intrusion prevention systems (IPSs) and, to a lesser degree, deployment of identity and access management (IAM) systems and computer forensics techniques. There appears to be growing interest in the security of applications, which is purportedly the vector commonly used by hackers in the majority of successful attacks and data breaches. This has resulted in the creation of such approaches as the "build security in" method of software development. As an example, DHS (the U.S. Department of Homeland Security) has developed a website devoted to software assurance at https://buildsecurityin.us-cert.gov/. The focus of the build-security-in approach is on invoking secure architectures, implementing secure application design, coding applications based on security principles (e.g., avoiding applications security risks, eliminating common vulnerabilities and weaknesses), as well as testing during the development lifecycle (static testing, code reviews) and in operation (dynamic testing). Axelrod, among others, has suggested more comprehensive security testing [6] and improved instrumentation within applications to identify anomalous user and system behavior [7].

Traditionally,very little, if any, attention has been paid by information security professionals to potential physical harm that might result from a software system malfunction or failure.This is likely due in large part to different backgrounds, education and training noted by Weise [1].

4 The safety perspective

Safety engineers, on the other hand, pay most attention to hazard risks related to harm that a malfunctioning or failed system might do to humans and/or the environment, particularly the type of environmental damage that could impact human beings physically as well as financially.

As a consequence, the emphasis of software engineers working with safety-critical systems is on the testing of various states that might result from a malfunction or failure of the system. In the realm of safety-critical software systems, the systems themselves are assigned to categories that relate to the degree of harm that a malfunctoning or failed system might inflict. As an example, we see in Table 1 a list of typical aircraft systems and the failure conditions, per the RTCA/DO-178C standard, to which they are assigned. See http://en.wikipedia.org/wiki/DO-178C, for an overview of the standard.

Table 1: RTCA/DO-178C standard applied to aircraft certification.

System	Type of System	Level A (Catastrophic)	Level B (Hazardous)	Level C (Major)	Level D (Minor)
Flight control	Control	X			
Cockpit display and controls	Control	X			
Flight management	Control	X			
Brakes and ground guidance	Control		X		
Centralized alarms management	Information			X	
Cabin management	Information				X
Onboard communications	Information				X

As it can be seen from the table, malfunctions and failures of systems related to the control of aircraft would be mostly catastrophic whereas information systems relating to cabin management and onboard communications are relatively less hazardous should they malfunction or fail. As a consequence, the safety standards required for certification are generally much more stringent than they are for information systems. This readily explains the concentration on safety issues by engineers who create, test and implement such systems. However, as aviation systems become more interconnected and accessible from public networks, so the risk of system compromise increases with the potential for hackers to tunnel through information systems into control systems and wreak havoc. Although some companies, which design and develop aviation systems, have asserted that their applications are isolated and therefore secure, there is no guarantee that this situation, if indeed true, will continue to exist, or even that software-development companies, which develop control software, are fully aware of present and potential dangers. The author discusses this issue at http://www.bloginfosec.com/2013/04/22/hacking-avionics-systems/, which was posted on April 22, 2013.

5 Collaboration and communication

In Figure 3, we see that, while information systems and control systems are usually subject to different threats and exploits and the consequences of breaches, malfunctions and failures vary for different system categories, when these systems are integrated into cyber-physical systems, such systems are exposed to the full range of threats, exploits and consequences.

This suggests that teams of engineers, who come up with requirements, designs, coding practices, testing scenarios, and the like, should include both cybersecurity and safety engineers as active participants. Risks assessments need to include analyses of both threats and hazards. Program code should follow secure coding practices and static and dynamic testing should be implemented. The impact on the environment, were a system to malfunction or fail, needs to be assessed and systems must be certified accordingly.

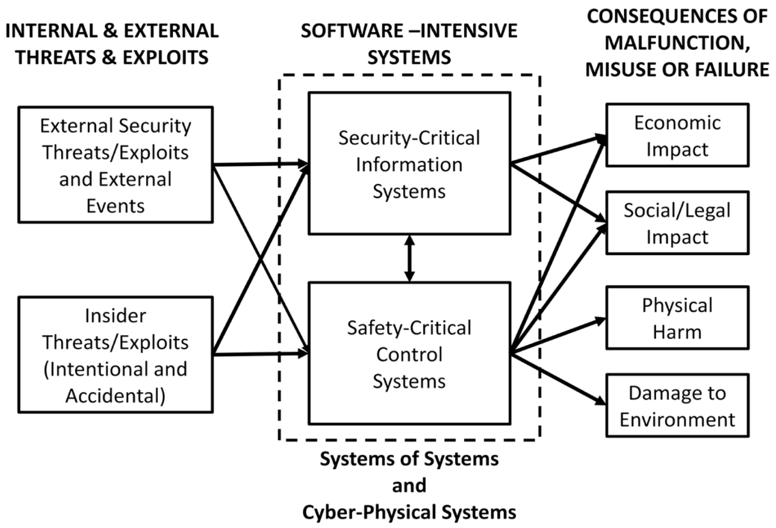
INTERNAL & EXTERNAL THREATS & EXPLOITS
SOFTWARE –INTENSIVE SYSTEMS
CONSEQUENCES OF MALFUNCTION, MISUSE OR FAILURE
External Security Threats/Exploits and External Events
Insider Threats/Exploits (Intentional and Accidental)
Security-Critical Information Systems
Safety-Critical Control Systems
Economic Impact
Social/Legal Impact
Physical Harm
Damage to Environment
Systems of Systems and Cyber-Physical Systems

Figure 3: Consequences of malfunction, misuse or failure of critical systems, (Source: C.W. Axelrod, *Engineering Safe and Secure Software Systems*, © 2013 Artech House).

In general*, aspects of software assurance for both security and safet*y need to be combined in an overall assessment of risks of exploitation of, and damage from, systems and the best-of-breed processes and tools need to be carried over from each silo to the other. Details of the security and safety aspects of the software system development lifecycle and the trading of knowledge, skills and tools between software safety and security subject-matter experts are provided in Axelrod [8].

6 Conclusion

We have seen that there exists a considerable gap between the orientation and focus of those with responsibility for the security and safety of software systems brought about by different education and training and reward systems that favor a concentration by professionals and researchers on one aspect or the other. This situation might be considered to be acceptable when information systems and control systems occupy separate and distinct domains. However, we are seeing rapidly escalating complexity and risk as software-intensive systems, which were previously separated, are interconnected to form cyber-physical systems, which become subject to the risks of both security-critical and safety-critical systems. In order to mitigate all of these risks, cybersecurity professionals and software safety engineers must form collaborative alliances to ensure that software systems meet the superset of requirements for the resulting systems of systems.

References

[1] Weiss, J., Protecting Industrial Control Systems from Electronic Threats, Momentum Press: New York, p. ix, 2010.

[2] Axelrod, C.W., Engineering Safe and Secure Software Systems, Artech House: Norwood, MA, 2012.

[3] Carnegie Mellon University Software Engineering Institute (CMU SEI), Software Assurance Curriculum Project: Volume I: Master of Software Assurance Reference Curriculum, Technical Report CMU/SEI-2010-TR-005, ESC-TR-2010-005, CMU SEI, 2010. Available at http://www.sei.cmu.edu/library/abstracts/reports/10tr005.cfm.

[4] National Science Foundation (NSF), Cyber-Physical System (CPS), Program Solicitation NSF 10-515, 2010, www.nsf.gov/pubs/2010/nsf10515/nsf10515.htm.

[5] Boehm, B.W., Characteristics of Software Quality, North-Holland: New York, 1978.

[6] Axelrod, C.W., "The Need for Functional Security Testing," CrossTalk, 24(2), pp. 17-21, 2011.

[7] Axelrod, C.W., "Creating Data from Applications for Detecting Stealth Attacks," CrossTalk, 24(5), pp. 19-24, 2011.

[8] Axelrod, C.W., "Applying Lessons from Safety-Critical Systems to Security-Critical Software," 2011 IEEE LISAT (Long Island Systems, Applications and Technology) Conference, Farmingdale, NY, May 2011, http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5784222&queryText%3DAxelrod+C.W.