Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 22 (2013) 954 – 959

Procedia Computer Science

17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems - KES2013

How to develop Security Case by combining real life security experiences (evidence) with D-Case.

Vaise Patu[a], Shuichiro Yamamoto[a]

[a]*Nagoya University, Furo-cho Chikusa-ku, Nagoya City 〒464-8601, Japan*

Abstract

In this paper we present a new reasonable method for writing security cases. A security case is one form of assurance case as there are other forms such as safety case, reliability case, dependability case and so on. Assurance case is a technique to structure assurance requirements concerning a system. It contains argumentation structure and evidence to convince a stakeholder that a certain concern on whether it is safety, reliability or security is being assured by the implementation of proper and appropriate actions that counters the risks or vulnerabilities raised by the concerned stakeholder. A security case is somehow very unpopular if we compare it to a safety case. And therefore, it is safe to say that so far; safety cases are better versed than security cases. However, due to the ever-growing concerns of the area of Cyber-Security, it is extremely crucial to produce effective and accurate security cases for cyber-systems.

Keywords: Assurance Case; Security Case; Dependable System; Networking Security; Goal Structuring Notation; Risk Management

1. Introduction

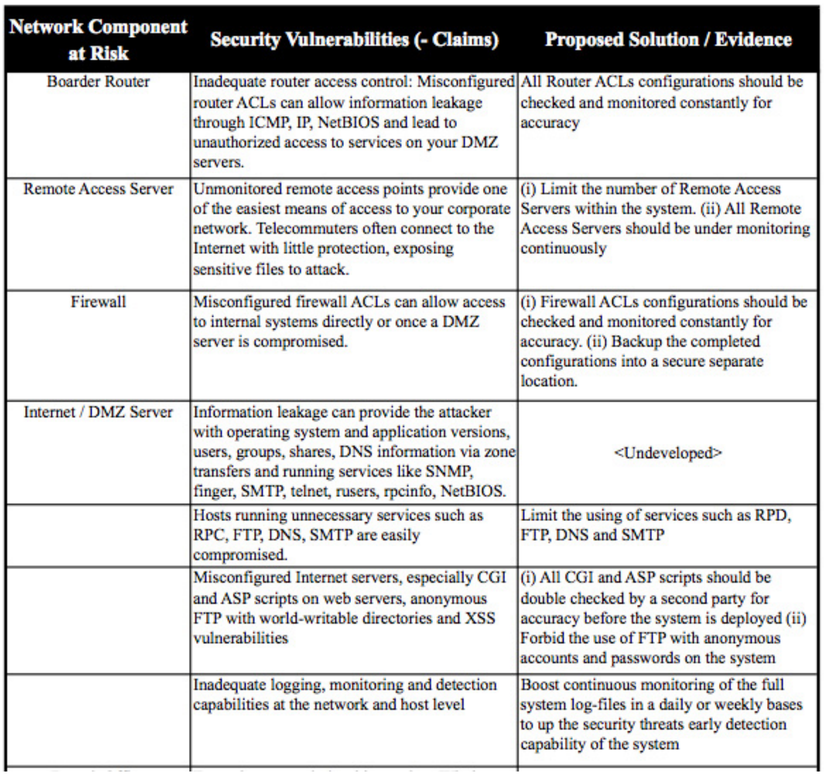
Security does not appear on the list of dependability attributes. Instead, the terms that are related to security are best defined in terms of integrity, confidentiality, and availability. The reason why security does not appear as an individual term is because it is very difficult to define what a secure system will mean because concerns in the security area are too diverse to deal with. For example, some systems are more concerned with unauthorized access to information while some are more concerned with denial-of-service attacks and so on. Nevertheless, our daily life reliance on information and software systems is ever increasing for the purpose of convenience, efficiency, and security. We all know that modern systems runs for long periods of time and are being constantly improved in service objectives and stakeholders' requirements under evolving technologies and changing regulations or standards. At the same time, these systems have become extremely complex and hence the high demand for dependable systems. And in order to achieve dependable systems, assurance cases are becoming indispensable.

Conventional technologies such as software processes and/or formal methods alone cannot guarantee the dependability of present-day complex systems anymore. Such complex systems are required to have proper written assurance cases in order for the system stakeholders to admit to its (complex system) dependability. There are many well-written documents on how to write safety cases and reliability cases but very few on security cases. The reason for this is not the concern of this paper but to propose a sensible method of how to write a security case by looking at previous and current systems with (1) similar functionalities (not necessarily an exact match), (2) running under similar environment/s and more important; (3) suspected of encountering similar risks. This paper is limited to networking and information systems such as our global networking e-learning system named KISSEL or Knowledge Integrated Server System for E-Learning.

2. Introducing our new method

From the current method, the decision of what aspect of the system (whether its safety, reliability or availability) to be assured via assurance case is usually directly or indirectly comes from its (the system under review) risk assessment results and report, system expert's previous experience (lessons leant) and so forth. However in our new method, since we are getiing all these risk assessment results and reports plus the lessons leant of from similar systems, we can actually use these gathered informations into our advantage. For example, in Table 1 below, we show a list of the most popular security vulnerabilities that most information sharing and infromation storage system have to overcome. And of this list of security vulnerabilities also known to us as claims, we built and derived from it a list of solutions evidence for the construction of a security case.

Table 1. List of the most common security vulnerabilities with possible solution to help by limiting the risk causal factors



Network Component at Risk	Security Vulnerabilities (- Claims)	Proposed Solution / Evidence
Boarder Router	Inadequate router access control: Misconfigured router ACLs can allow information leakage through ICMP, IP, NetBIOS and lead to unauthorized access to services on your DMZ servers.	All Router ACLs configurations should be checked and monitored constantly for accuracy
Remote Access Server	Unmonitored remote access points provide one of the easiest means of access to your corporate network. Telecommuters often connect to the Internet with little protection, exposing sensitive files to attack.	(i) Limit the number of Remote Access Servers within the system. (ii) All Remote Access Servers should be under monitoring continuously
Firewall	Misconfigured firewall ACLs can allow access to internal systems directly or once a DMZ server is compromised.	(i) Firewall ACLs configurations should be checked and monitored constantly for accuracy. (ii) Backup the completed configurations into a secure separate location.
Internet / DMZ Server	Information leakage can provide the attacker with operating system and application versions, users, groups, shares, DNS information via zone transfers and running services like SNMP, finger, SMTP, telnet, rusers, rpcinfo, NetBIOS.	<Undeveloped>
	Hosts running unnecessary services such as RPC, FTP, DNS, SMTP are easily compromised.	Limit the using of services such as RPD, FTP, DNS and SMTP
	Misconfigured Internet servers, especially CGI and ASP scripts on web servers, anonymous FTP with world-writable directories and XSS vulnerabilities	(i) All CGI and ASP scripts should be double checked by a second party for accuracy before the system is deployed (ii) Forbid the use of FTP with anonymous accounts and passwords on the system
	Inadequate logging, monitoring and detection capabilities at the network and host level	Boost continuous monitoring of the full system log-files in a daily or weekly bases to up the security threats early detection capability of the system

Branch Office	Excessive trust relationships such as Windows Domain Trusts, UNIX rhosts, and SSH files can provide attackers with unauthorized access to sensitive systems	<Undeveloped>
Workstation	Weak, easy to guess passwords at the workstation level can compromise your company's server	<Undeveloped>
	Unauthenticated services like X Windows allow users to capture remote keystrokes or workstation keystrokes after software is installed	<Undeveloped>
Internal LAN Server	Excessive file and directory access controls (Windows shares and UNIX NFS exports)	<Undeveloped>
	Software that is unpatched, outdated or left in default configurations, especially web servers are vulnerable to attacks	<Undeveloped>

To put things into perspective, it is more sensible if we show the strategy node of our security case diagram shown in Figure 1. From Table 1, we derived from it 7 sub-goals for our security case labelled Goal: G_2, G_3, G_4 up to Goal: G_7.

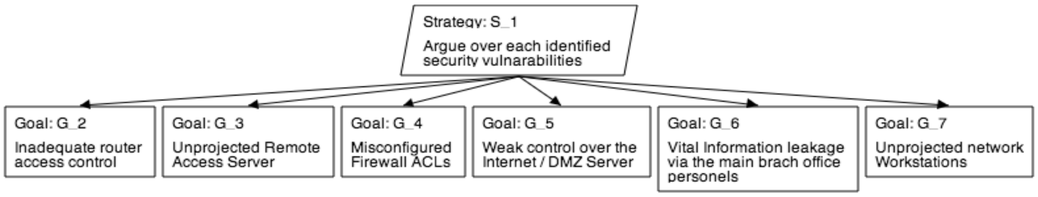

Strategy: S_1
Argue over each identified security vulnarabilities
Goal: G_2
Inadequate router access control
Goal: G_3
Unprojected Remote Access Server
Goal: G_4
Misconfigured Firewall ACLs
Goal: G_5
Weak control over the Internet / DMZ Server
Goal: G_6
Vital Information leakage via the main brach office personels
Goal: G_7
Unprojected network Workstations

Fig.1. Our 7 sub-goals derived from the list of vulnerabilities shown in Table 1.

This decomposition supports the strategy node Strategy: S_1 with an argument for security vulnerability.

2.1 *Essential documentations that is required to be inserted into the security case diagram*

During the development of the security assurance case, it is very important to understand what sort of documents one should use to show as evidence inside the security case and what documents is necessary to be inserted into the security case diagram. The following list is called Security Requirement Documentations List. In our new method, we used this list as a checklist of documentations that are required to be inserted into the "context" node and so as the solution node if found suited.

Table 2: Security Requirement Documentations List

Security Awareness:	Have you ensured that the corporate security policies and guidelines to which you are designing are the latest versions? Have you read them? Are you aware of all relevant computing security compliance and risk acceptance processes? (Interviewer should list all relevant policies and guidelines.)
Identification Authentication:	Diagram the process flow of how a user is identified to the application and how the application authenticates that the user is who they claim to be. Provide supporting documentation to the diagram explaining the flow from the user interface to the application/database server(s) and back to the user. Are you compliant with corporate policies on accounts, passwords, etc.?
Authorization:	Provide a process flow from beginning to end showing how a user requests access to the application, indicating the associated security controls and separation of duties. This should include how the request is approved by the appropriate data owner, how the user is placed into the appropriate access-level classification profile, how the user ID, password, and access is created and provided to the user. Also include how the user is informed of their responsibilities associated with using the application, given a copy of the access agreement, how to change password, who to call for help, etc.

Access Controls:	Document how the user IDs, passwords, and access profiles are added, changed, removed, and documented. The documentation should include who is responsible for these processes.
Sensitive Information Protection:	Provide documentation that identifies sensitive data requiring additional protection. Identify the data owners responsible for this data and the process to be used to protect storage, transmission, printing, and distribution of this data. Include how the password file/field is protected. How will users be prevented from viewing someone else's sensitive information? Are there agreements with outside parties (partners, suppliers, contractors, etc.) concerning the safeguarding of information? If so, what are the obligations?
Audit Trails and Audit Logs:	Identify and document group accounts required by the users or application support, including operating system group accounts. Identify and document individual accounts and/or roles that have super-user type privileges, what these privileges are, who has access to these accounts, how access to these accounts is controlled, tracked, and logged, and how password change and distribution are handled, including operating system accounts. Also identify audit logs. Who should read the audit logs; who should modify the audit logs; who can delete the audit logs, and how the audit logs are protected and stored. Is the user ID obscured in the audit trails?
External Access Considerations:	Will the application be used internally only? If not, are you compliant with corporate external access requirements?

3. Building the Security D-Case

In this section of the paper, we briefly explain the steps of how we build our security case structured diagram using the D-Case technique. Our security case diagram is derived from the information provided by the list (refer to Table 1) of the most common security vulnerabilities that are often encountered by corporate information sharing systems or networking systems. But before the building of the security dependability case, first we'd like to explain about what a dependability case is.

3.1. Building the Security Case

This new method was used and tested as an experimental research application to provide assurance to the security aspect of our networking E-learning system called KISSEL (Knowledge Integrated Server System for E-Learning). More work, experiment and tests (trail and error) are still needed in order to take our work into the next stage. But as all great ideas and great technologies of today, it all started from a simple vision.

The security domain is not new to networking and information sharing systems. Therefore, such systems have been associated with security factors for as long as networking and information systems existed. What is really new here is the urge to associate the networking system with assurance case that assured the networking system will behave as predicted, function well under pressure from future security vulnerabilities and attacks and so on.

Since we are building a security case, therefore our top goal or the main claim that we are going to argue about our system should be the "System is reasonably secure". To support this top goal or claim, the context nodes should provide whichever sort of environmental information about the system or any sort of attachment that helps to make the main argument truthful or convincing; like for example, documents like the system requirements which tells of what kind of security requirements that the system has in order to secure the system from unwanted attacks, or any kinds of security design architecture diagram if any, and so forth. Figure 2 is to provide a visualized view of what this paragraph is trying explain. Figure 2 shows only the top most part of our security case diagram.

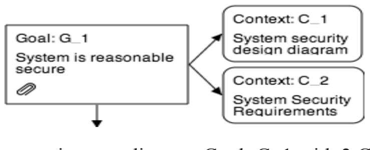

Goal: G_1
System is reasonable secure
Context: C_1
System security design diagram
Context: C_2
System Security Requirements

Fig.2. The main Goal of our security case diagram Goal: G_1 with 2 Context nodes C_1 and C_2

After the top goal is set with all the necessary contexts. What follows is the decomposition step or stage of the main argument or the ‘main goal’ into two or more sub-goals. However, prior to the decomposition stage, where the ‘sub-goal nodes’ comes into the picture, the node called ‘strategy’ is inserted between the main goal and the sub-goals. The strategy node should explain or give a sense of justification or reason to why we the security case builders decided to decompose the main goal into such and such number of sub-goals. In figure 6, the strategy links us to some of the identified security vulnerabilities displayed in table 1. In this example, we take 6 identified security vulnerabilities as sub-goals. And the strategy is to argue over each of the 6 identified security vulnerabilities. Figure 6 is to provide a visualized view of what this paragraph is trying explain.

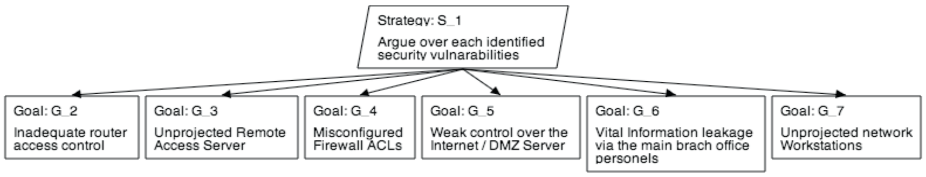

Strategy: S_1
Argue over each identified security vulnarabilities
Goal: G_2
Inadequate router access control
Goal: G_3
Unprojected Remote Access Server
Goal: G_4
Misconfigured Firewall ACLs
Goal: G_5
Weak control over the Internet / DMZ Server
Goal: G_6
Vital Information leakage via the main brach office personels
Goal: G_7
Unprojected network Workstations

Fig.3. Our 7 sub-goals derived from the list of vulnerabilities shown in Table 1

No matter what the assurance case or the D-Case is, whether it is a safety case, reliability case or a security case, they all follow a kind of pattern. For example, [9] security case patterns are claims-argument-evidence structures that can be reused in many different security cases. The security case method offers the opportunity for security and domain experts to organize security knowledge and mitigation strategies in the form of security case patterns. Such patterns can then be shared among the security community and other stakeholder communities and continually built upon, refined, and improved. A growing repository of security case patterns is a huge possibility for a variety of domains and operational contexts that not only would provide greater opportunities for reuse and standardization of assurance arguments, but also could allow the security community to associate an historical record of security performance and return on investment with particular security case patterns.

In the last measure to our security case, we would see that some of the sub-goals got themselves a straight forward evident or solution that satisfy the final objective in supporting of the main goal, while in some sub-goals, they have to be expanded more widely in order to get to the heart of where the evident truly exist. Figure 7 shows how sub-goals one, two and three are decomposed until the evident that satisfy all the objectives of the security case main goal.

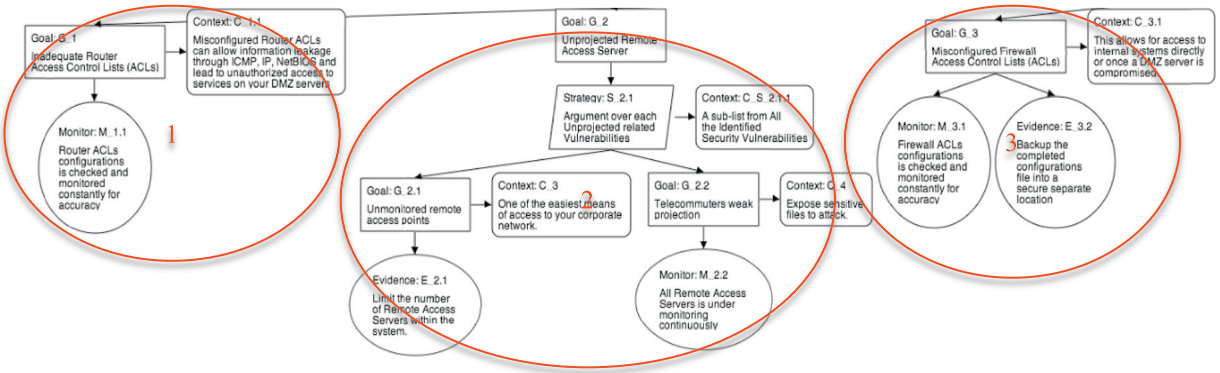

Goal: G_1
Inadequate Router Access Control Lists (ACLs)
Context: C_1.1
Misconfigured Router ACLs can allow information leakage through ICMP, IP, NetBIOS and lead to unauthorized access to services on your DMZ servers
Monitor: M_1.1
Router ACLs configurations is checked and monitored constantly for accuracy
1
Goal: G_2
Unprojected Remote Access Server
Strategy: S_2.1
Argument over each Unprojected related Vulnerabilities
Context: C_S_2.1.1
A sub-list from All the Identified Security Vulnerabilities
Goal: G_2.1
Unmonitored remote access points
Context: C_3
One of the easiest means of access to your corporate network.
2
Goal: G_2.2
Telecommuters weak projection
Context: C_4
Expose sensitive files to attack.
Evidence: E_2.1
Limit the number of Remote Access Servers within the system.
Monitor: M_2.2
All Remote Access Servers is under monitoring continuously
Goal: G_3
Misconfigured Firewall Access Control Lists (ACLs)
Context: C_3.1
This allows for access to internal systems directly or once a DMZ server is compromised
Monitor: M_3.1
Firewall ACLs configurations is checked and monitored constantly for accuracy
Evidence: E_3.2
Backup the completed configurations file into a secure separate location
3

Fig.4. Here is how we decomposed Goal: G_1 ~ Goal: G_3 and at the bottom is the solution nodes that we derived from our discussions which was then inserted in Table 1.

Our next figure; Figure 5 shows how we decomposed sub-goal Goal: G_4 into four more sub-goals of sub-goal G_4.1, sub-goal G_4.2, sub-goal G_4.3 and sub-goal G_4.4. Note that for sub-goal G_4.1, we added one property of D-Case called node = *undeveloped* which means this sub-goal G_4.1 is yet to be completed. Many factors could contribute to be the reason why a sub-goal is labelled undeveloped. For example, one of the factors could be that the evident or evidence provided to satisfy the objective of sub-goal G_4.1 is not well defined by both the stakeholders, system engineers and the security case builders.

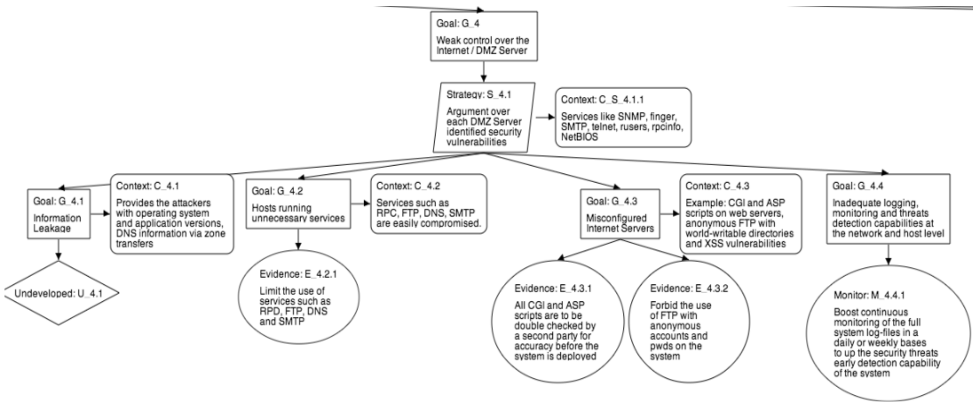

Goal: G_4
Weak control over the Internet / DMZ Server
Strategy: S_4.1
Argument over each DMZ Server identified security vulnerabilities
Context: C_S_4.1.1
Services like SNMP, finger, SMTP, telnet, rusers, rpcinfo, NetBIOS
Goal: G_4.1
Information Leakage
Context: C_4.1
Provides the attackers with operating system and application versions, DNS information via zone transfers
Undeveloped: U_4.1
Goal: G_4.2
Hosts running unnecessary services
Context: C_4.2
Services such as RPC, FTP, DNS, SMTP are easily compromised.
Evidence: E_4.2.1
Limit the use of services such as RPD, FTP, DNS and SMTP
Goal: G_4.3
Misconfigured Internet Servers
Context: C_4.3
Example: CGI and ASP scripts on web servers, anonymous FTP with world-writable directories and XSS vulnerabilities
Evidence: E_4.3.1
All CGI and ASP scripts are to be double checked by a second party for accuracy before the system is deployed
Evidence: E_4.3.2
Forbid the use of FTP with anonymous accounts and pwds on the system
Goal: G_4.4
Inadequate logging, monitoring and threats detection capabilities at the network and host level
Monitor: M_4.4.1
Boost continuous monitoring of the full system log-files in a daily or weekly bases to up the security threats early detection capability of the system

Fig.5. This figure shows how sub-goal 4 is decomposed into 4 more sub-goals of 4.1, 4.2, 4.3 and 4.4

4. Evaluation and Conclusion

We proposed a new and sensible method of how we should write security cases. This security case though may only be suited for information networking systems with small budgets. The current method is still working however it is too costly and therefore not suited for projects with small budgets. By assessing similar systems (similar in functionalities, running under similar environment and gathered previous encountered risks reports), we the security case developers could easily managed and discuss all the necessary actions prior to the development of the security case. This step helped to limit the cost of having experts to do all the preliminary jobs for you. We have a complete manual of security cases developed for our e-learning system. And so far, it has helped us to limit the threats we already faced and threats that are yet to be faced.

5. Acknowledgement

This paper presents an accomplishment of research and development of the research project titled "Dependable Operating Systems for Embedded Systems Aiming at Practical Applications" in the Research Area named Core Research for Evolutional Science and Technology (CREST), which is sponsored by Independent Administrative Agency - Japan Science and Technology Agency (JST).

6. Reference

[1] Ankrum, T. S. & Kromholz, A.H. "Structured assurance cases: three common standards," 99-108. Proceedings of the Ninth IEEE International Symposium on High-Assurance Systems Engineering (HASE'05), 2005.
[2] Avizienis, Algirdas, Laprie, Jean-Claude, & Randell, Brian. "Fundamental Concepts of Dependability." *Proceedings of the Third Information Survivability Workshop (ISW2000)*.
[3] Bloomfield, Robin & Littlewood, Bev. "Multi-legged Arguments: The Impact of Diversity Upon Confidence in Dependability Arguments." Proceedings of 2003 International Conference on Dependable Systems and Networks, San Francisco, California. IEEE Computer Society Press, 2003.
[4] Jackson, Daniel, Martyn Thomas, and Lynette I. Millett (Editors), Software for Dependable Systems: Sufficient Evidence? Committee on Certifiably Dependable Software Systems, Computer Science and Telecommunications Board, National Research Council, National Academies Press, ISBN:0-309-66738-0, (available at http://www.nap.edu/catalog/11923.html)
[5] Matsuno, Y., Takamura, H., & Ishikawa, Y. (2010). A Dependability Case Editor with Pattern Library. In Proc. IEEE HASE, pages 170-171.