A Secure Software Engineering Framework to vaccinate operational software

Daniel Fischer[1] and Mehran Sarkarati[2]
European Space Agency
European Space Operations Centre
Darmstadt, Germany

Thomas Michelbach[3]
Accenture
Kronberg, Germany

ESA is developing, deploying, and operating a wide variety of mission data systems for the command & control of spacecraft, as well as for the exploitation and dissemination of space related services to end users. These systems, once completely isolated, are getting more and more inter-connected and use global communication channels such as the Internet to communicate between each other and with end users. In recent years, the threats resulting from cyber-attacks and hacker activities have been increasing throughout the world and software systems have been a particular target of such attacks. In this paper, we describe a secure software engineering framework that has been developed by ESA to enhance the robustness and security of operational mission data systems and is currently further improved. The framework reduces the cost and effort overhead induced by enhanced software engineering security through automation and exploitation of commonalities where possible. It also supports software developers directly by providing them a secure software development toolbox. We are convinced that this secure software engineering framework, though automated as far as possible, is a mandatory addition to traditional software engineering. It will help to vaccinate operational software against the growing number of cyber threats.

I. Introduction

ESA is developing, deploying, and operating a wide variety of mission data systems for the command & control of spacecraft, as well as for the exploitation and dissemination of spacecraft supported services to end users and stakeholders. These mission data systems are based on a variety of different software engineering technologies, from distributed SOA-based systems to traditional, monolithic legacy systems. With the increasing presence of wide area networks, distributed sites and the need for near-real-time product delivery to users everywhere in the world, interconnectivity using global communication networks such as the Internet is ubiquitous. This process makes data systems more effective and reduces development and maintenance costs.

However, this process also changed substantially the threat situation with respect to mission data systems. Previously, data systems were not seen as a major vulnerability since they were protected by strong network and physical layer security. This paradigm however is not valid anymore in an interconnected world. While firewalls can still substantially increase network security, they are in many cases not capable of identifying and preventing application layer attacks. In fact the application layer has turned out to be the major target of the newest generation of hacker attacks. These attacks exploit flaws and imperfections in software products such as unprotected buffers and subsequently break into the system. In fact, the more complex a data system is, the more likely it is that it contains flaws that can be exploited. This trend is very visible in the Internet, where critical software components such as operating systems and browsers are continuously targets of successful application layer attacks. Furthermore, using malware such as worms, viruses, and Trojan horses, such attacks are very easy to automate. The

[1] Data Systems Manager, HSO-GD, Daniel.Fischer@esa.int
[2] Data Systems Manager, HSO-GD, Mehran.Sarkarati@esa.int
[3] Senior Technology Architect, Accenture Innovation, Thomas.M.Michelbach@accenture.com

current software engineering strategy for mission data systems, which is derived from ECSS software engineering standards, only addresses this threat through the testing step, which is insufficient.

In fact, to be able to counter this increasing threat, the robustness and security quality of the operational data systems is currently being increased. An important piece in this puzzle is the development of a Generic Application Security Framework (GASF) that introduces new development steps to be applied during software development as well as mandating an information risk assessment that is specific to the data systems underlying technology. In the remainder of this paper, we describe the GASF in detail and report on the status of its development and the first use cases.

A. Contribution and Technology Spin-In

In this paper, we describe in detail the GSAF and its enhanced software development lifecycle, from requirements definition phase to operations phase. In particular, we focus our description on the reduction of the development overhead induced by security requirements. Furthermore, we analyze different types of software, based on different technologies, and what impact they have on the set of threats and vulnerabilities that need to be addressed during development.

The GSAF has been designed based on a number of processes and methodologies that are well accepted within the computer science community since a long time. Commercial business and industry are facing security issues on a day-to-day basis and thus have already invested a lot in the protection of their software assets. They have the necessary processes and methodologies already in place. Our main contribution is the spin-in, and tailoring where necessary, of these processes to the domain of mission data-systems, which is quite a different environment.

B. The Generic Application Security Framework

The GSAF is composed from several core components that we describe in greater detail in this paper.

At the heart of the GASF is the Secure Software Development Lifecycle (SSDLC). It extends the traditional ESA Software Development Lifecycle (SDLC), derived from the European ECSS E-40C [3] standard, with additional steps, procedures, and deliverables for each step of the lifecycle. A basic element, which re-occurs in each of the major steps of the SSDLC is the information risk analysis, which is part of the on-going security assessment. The SSDLC initially starts with a high-level initial risk analysis, leading to the identification of initial risks that need to be mitigated. The information risk analysis is then repeated in the successive steps of the SSDLC, adding more detail in each subsequent iteration. In the final iteration that occurs during the software implementation & testing phase, a detailed understanding of the risks that are either countered or accepted is obtained.

The SSDLC naturally imposes additional effort for the software development process. This is particularly true when developing web services, data persisting applications, or web based application clients that share common characteristics. In times of limited mission budgets, this could quickly evolve into a potential showstopper for security control implementation. However, this additional development effort can be significantly reduced by using a number of security patterns, which have been developed as part of the application security framework to support the developers. Once the initial information risk assessment is performed, the developers can choose from a number of pre-defined application security patterns that contain appropriate security requirements for countermeasures to mitigate the initially defined risks. As a result, subsequent risk analysis steps will only have to be executed for new or special functions that are not shared among the web services.

Furthermore, the GASF provides tools and methodologies for adequate generation of security requirements, implementation, testing & validation of applications for mission data systems based on several technologies. Currently, web applications and web services are supported, however the currently on-going evolution will then also support C++ and Fortran based legacy applications. To achieve this, well-known and widely accepted standards and guidelines (de-facto standards) such as ISO 27001 [4], SQUARE [5] , and OWASP [6] have been used as the basis for specification of the subject methodology, tailored where necessary, and integrated into the application security framework. Finally, the GASF can also deal with classification requirements for certain pieces of information that need to be processed by the system.

C. Organization

The remainder of this paper is organized as follows. In Section II, we review the software engineering technologies that the GASF is addressing. We also highlight the associated security challenges. In Section IV, we describe our approach to Application Security Assessment. Section V contains our main contribution and describes the Secure Software Development Lifecycle (SSLC) that lies at the heart of the GASF. In Section VI, we describe the core component of the SSDLC, the information risk assessment approach, in more detail. Section VII analyses the operational impact of using the GASF and finally Section VIII outlines the next steps and concludes this paper.

II. Supported Software Engineering Technologies Review

In this section, we review the software engineering technologies that are or will be supported by the GASF. We also highlight the differences between these technologies and what it means for the GASF. Currently, two sets of technologies are considered:

- SOA-based data systems (i.e. web applications and web services)
- Monolithic and distributed C++ based data systems

Support for additional technologies such as Fortran will be added to the GASF once its currently ongoing evolution is completed.

A. SOA-based Mission Data Systems

Some of the newer ESA missions and programs, such as the Space Situational Awareness (SSA) Initiative require a highly-distributed system that is service-oriented[11]. Service-Oriented Architecture (SOA) [2]-based mission data systems are the best suited technology to address these needs.

In the context of the SSA Initiative, ESA has prepared the implementation of a Common SSA Integration Framework (COSIF). The COSIF shall enable the integration of existing SSA assets as well as deployment of new heterogeneous SSA applications, forming a system-of-systems. It shall serve as the backbone for all SSA mission data systems. The specific SSA services will be developed and integrated as enterprise applications, mainly web service components, on the top of COSIF.

SOA-based architectures are by their nature distributed and consist of interconnected services. Physical isolation of communication channels is a significant challenge in such systems and often proves infeasible. For civilian systems, the use of publicly available communication networks such as the Internet has proven to be the most practical and efficient approach.

It has been recognized, that from an information security point of view, an interconnected SOA architecture is much more vulnerable than an individual monolithic system and securing such a system is a significant challenge [10]. The main reason is that many of its components and services are exposed to public communication channels and become a potential target for threat sources such as hackers. It is no longer possible to protect the whole system-of-systems by relying only a perimeter defence strategy such as firewalls. Instead, the individual components of the SOA-based architecture that implements the system-of-systems and their interactions have to be protected.

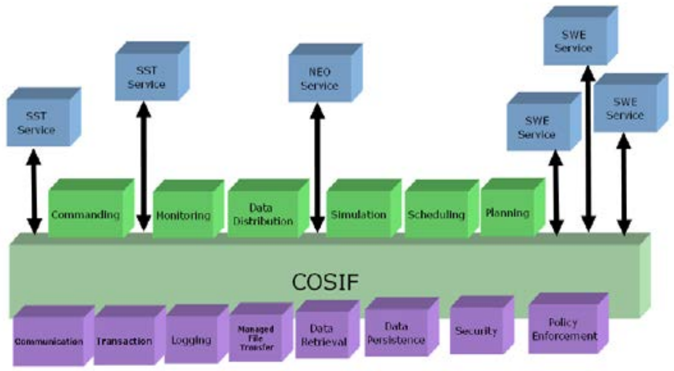


Figure 1. COSIF Service-Oriented Architecture.

While strong perimeter defences for each component of the architecture are possible, this approach is very cumbersome and not really flexible.

B. Legacy Data Systems

While new missions and programs such as the SSA Initiative more and more use technologies such as SOA-based architectures, legacy applications such as control systems and other support systems are more monolithic and based on a completely different software technology. Many of these systems need to process huge amounts of data in near real-time and thus need to be as performant as possible. For this reason, they are constructed using system-close programming languages such as C++ or Fortran.

From security point of view, the challenges are completely different than for SOA-based architectures. System-close programming languages allow direct access to memory resources and other system-level functions that are not available for software engineers of higher level programming languages such as Java are used (which is usually the case for SOA-based systems). In fact, the majority of the threats concerning legacy data systems is not related to communication channels and messaging problems, but rather on the improper use of commands that directly manipulate system resources (the buffer overflow is a classic example).

III. Development of the Generic Application Security Framework

Before we turn to the description of the GASF, we describe how we created and implemented its first version, which is supporting web applications and web services. The approach that has been chosen is a selection of simple steps, which will facilitate the implementation of the end-to-end methodology. The steps in principle are the same also for legacy applications, however the input sources and methodologies vary.

The GASF was developed using following principle:

- Gather security relevant input;
- Define security domains;
- Identify security capabilities;
- Derive security controls/requirements;
- Provide templates and tools.

In the following, these principles are addressed in more detail.

A. Gather Security Relevant Input

The GASF for web applications and web services is built upon requirements that have been gathered through the analysis of relevant and well accepted industry standards such as ISO 27002:2005 [7] and industrial de-factor standards like SQUARE [5], OWASP [6], BSI [8] and other sources. In this way, the framework is compliant with international standards and can easily be validated or further prepared for a possible certification.

B. Define Security Domains

The security requirements gathered in step A were used to define three security domains that represent categories of specific security controls relevant for mission data systems. Each of these service domains must be executed across the information technology layers to support an end-to-end security environment.

The security domains are:

- Communications and operations management (Maps to ISO 27001 A.10 domain);
- Access control (Maps to ISO 27001 A.11 domain);
- Information system acquisition, development and maintenance (Maps to ISO 27001 A.12 domain)

Each of the above domains is refined in the ISO 27001 [4] standard and lists different security controls, in order to mitigate possible threats and vulnerabilities the application infrastructure might have. The security domains are the highest level of abstraction and will be used to map the needed capabilities for each specific application.

C. Identify Security Capabilities

In order to satisfy the requirements of the security domains that have been identified above, technical and procedural security capabilities (mapping to security functional and security assurance requirements) have been defined in the security capability list.

Each capability detailed in the reference architecture provides a layer of security protection that is intended to reduce risk and increase security for the application infrastructure. Examples for such capabilities include authentication techniques, access control, secure communication, cryptographic algorithms, security testing, etc. The application security framework currently contains a full catalogue of capabilities that are applicable for SOA-based mission data systems. The same catalogue for C++ based legacy implementations is currently under development and will be implemented into the GASF.

D. Derive Security Controls / Requirements

Based upon the documentation and best practices gathered during the input phase, a detailed list of information security requirements were developed to serve as a baseline for the application security framework. These requirements were derived from vulnerabilities and threats, which can be mapped to the security domains and capabilities that are mentioned above.

Furthermore, the gathering of security requirements is driven by the Security Quality Requirements Engineering (SQUARE) methodology [5], which consists of eight steps:

1. Agree on definitions;
2. Identify security goals;
3. Develop artefacts to support security requirements definition;
4. Perform risk assessment;
5. Select elicitation techniques;
6. Elicit security requirements;

7. Categorize requirements as to level (system, software, etc.) and whether they are requirements or other kinds of constraints;
8. Prioritize requirements;
9. Requirements inspection.

The validation of these requirements will be mapped to the security controls of each domain. Thus, traceability to the original high level decisions is guaranteed.

E. Provide Tools

In addition to the security domains, capabilities and requirements that are defined in this document, tools are provided to aid in the implementation of these concepts. The GASF toolset will be hooked into the organization and even more to the software development lifecycle in order to provide tools to address security concerns.

IV. Application Security Assessment

Before the development of an application can be initiated, its security needs must be assessed. This process will be executed by information security experts in close collaboration with the application development team. It fill first categorize the application to be developed in terms of confidentiality, integrity, and availability (CIA) levels. This categorization can result from Agency security directives and/or an initial risk assessment.

The categorization drives the discovery of security capabilities to tackle the different types of controls that need to be implemented. A public application that will never process any sensitive information and can do that at any point in time, might have very low confidentiality and availability requirements and therefore will implement less security controls, than an application that handles very sensitive personal and system data and needs to be available around the clock (24/7).

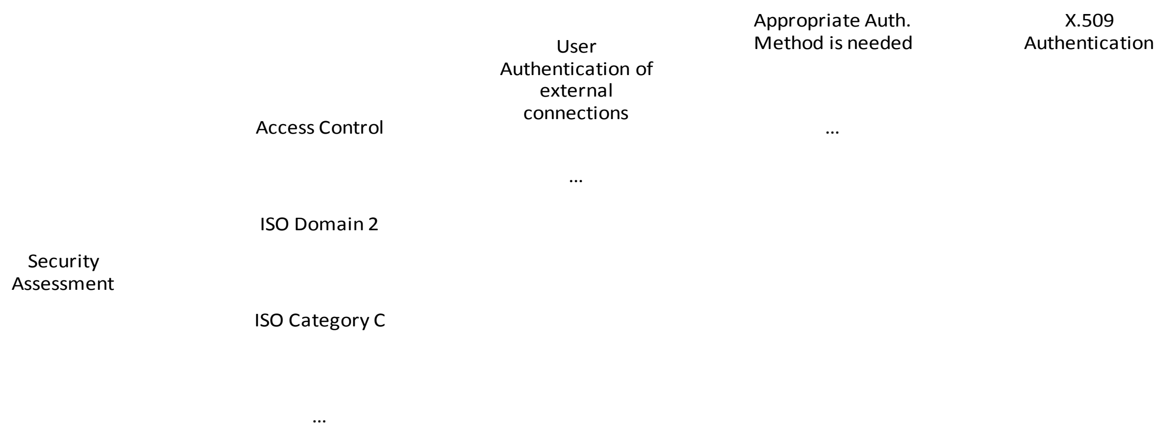


Figure 2. Structure of Security Assessment.

A. Approach to assessment

Different organizations, standardization bodies and companies have defined standards and procedure to approach security concerns and assessments. The security assessment defined as part of the GASF re-uses industry standards and performs a tailoring to the ESA needs based on the different categorizations. Figure 2 sketches the structure of the applicable assessment.

ISO 27001, a family of standards for Information Security Management Systems (ISMS), was used as high level map to group and categorize security controls. This standard is well-known and deals with information systems. It can increase the abstraction level depending on the audience analyzing the security assessment.

In the example of Figure 2, Access Control is a so-called ISO domain, which specifies in one of its categories the usage of user authentication for external connections. This category is a recommendation that appropriate authentication method is needed in other to control security that is why this recommendation is also called security control.

There are many types of authentication methods; in this example X.509 authentication is mentioned. This is considered a strong authentication mechanism through certificates, which could be used in the case of sensitive applications.

B. Repository of Threats and Vulnerabilities

The repository of threats and vulnerabilities lists the typical problems an application might face from a security point of view.

Of course, this repository is specific to the domain of application being designed. As mentioned above, the current version of the GASF supports web services and web applications and the evolution which is currently under development will add legacy applications as a new class of applications.

In our example, the authentication capability might face a brute force attack. Authentication based on X.509 with a minimum key length will minimize the effect of a brute force attack, due to the authentication technique and key length that increases the amount of iterations needed to crack a password in comparison to plain text passwords. In the other hand if the application is only secure by username and password, the attack might be extended to a dictionary attack. In this case a correct password management policy can minimize the effects of the attack, by increasing the amount of possibilities with the introduction of special characters and case sensitiveness.

Each threat and vulnerability can be mapped to a security capability/control in a similar way.

V. The Secure Software Development Lifecycle

The Secure Software Development Lifecycle (SSDLC) is at the heart of the GASF. This section provides a description of the SSDLC phases as well as a detailed breakdown of the responsibilities of the project team and the security team for each delivery phase of the SSDLC. Each phase of the SSDLC contains recommended security tasks for the project team as illustrated in Figure 3. These phases are the same no matter which kind of application is being developed. Each work product from the standard software development lifecycle from Figure 3 is mapped to the deliverables foreseen in the SSDLC.

A. Requirements and Architecture Phase

We describe the responsibilities, deliverables, and detailed activities that occur in the Requirements and Architecture phase of the SSDLC.

High-Level Architecture Review: A high-level architectural review of the system must be conducted in order to understand the different components of the infrastructure. The results, along with the security requirements, lead to the selection of the appropriate security controls for a given system. Developing a security data flow also assists in the identification of the infrastructure associated with the application where security controls are required. A new iteration might be needed if security requirements influence the component decision.

Categorization of Data Sensitivity: Project teams must identify and categorize the data that the system will handle according to the information identification and categorization process. The model walks through each step in the data categorization process and provides an inventory of security controls that could be implemented for each level of sensitivity identified. This step can be extended to also deal with data item classifications but must then comply with the organizations security directives for data classification.

Define high level security approach. The major security capabilities that will be developed to support the system are identified here according to the previously identified security controls.

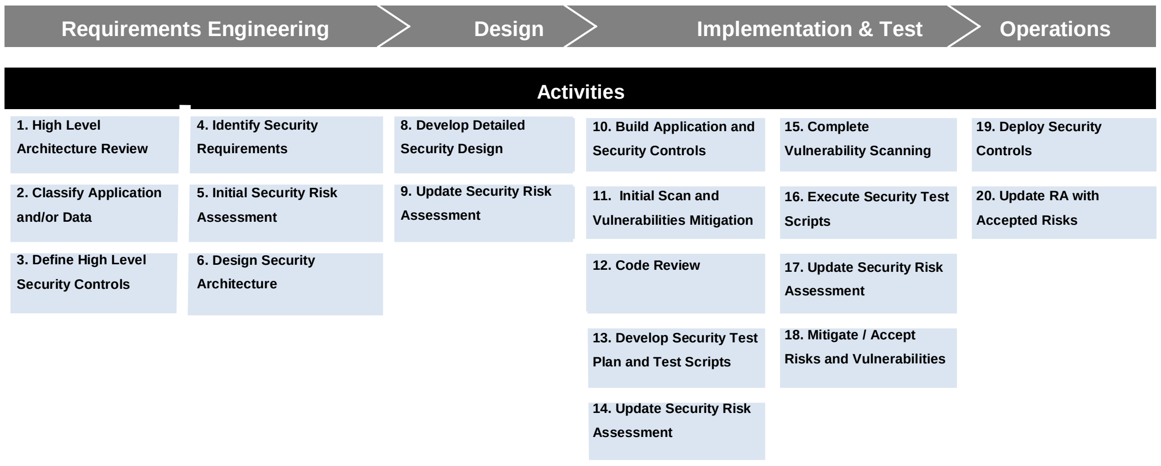

Requirements Engineering
Design
Implementation & Test
Operations
Activities
1. High Level Architecture Review
2. Classify Application and/or Data
3. Define High Level Security Controls
4. Identify Security Requirements
5. Initial Security Risk Assessment
6. Design Security Architecture
8. Develop Detailed Security Design
9. Update Security Risk Assessment
10. Build Application and Security Controls
11. Initial Scan and Vulnerabilities Mitigation
12. Code Review
13. Develop Security Test Plan and Test Scripts
14. Update Security Risk Assessment
15. Complete Vulnerability Scanning
16. Execute Security Test Scripts
17. Update Security Risk Assessment
18. Mitigate / Accept Risks and Vulnerabilities
19. Deploy Security Controls
20. Update RA with Accepted Risks

Figure 3. Secure Software Development Lifecycle (SSDLC).

Initial security risk assessment. To aid in the process of identifying system specific security risks and their sources, a Security Risk Assessment must be performed. The purpose of this assessment is to identify potential security risks the system may introduce into the production environment, and to support the development of a system security strategy that can be leveraged to help address those risks going forward. The assessment process includes the creation and subsequent analysis of the data flow diagram and mapping of the system architecture to security risks. These risks should be prioritized by severity. Risk severity is a function of the potential impact of the vulnerability if it were exploited, multiplied by the potential likelihood of the vulnerability being exploited or other more comprehensive risk ranking system such as OWASP's [6] Risk Rating methodology.

Design Security Architecture. The conceptual security design provides a high-level design summary of proposed security controls based on the requirements developed in the previous tasks. The conceptual security design may address features and functions of each security control with information such as operating system, services, data, applications, etc. Specific details for each security control are not required during the conceptual security design phase. The intent is to provide a high-level overview of features and functions of the key components and how each may support the overall goals of the project.

B. Design Phase

In the design phase, a detailed security design is derived and the information risk assessment updated. The security architecture detailed design is a specification for security technology components comprised of selected techniques and services. The design may be adapted to meet quality requirements, technical constraints, and performance requirements. During this process, the design may be refined until it satisfies the technical requirements. The detailed design also includes the specification of the environmental constraints of the application. This is important since it affects later applicable security policies or certification attempts. As part of the detailed security design, a revisit of the security requirements is also foreseen. The security operations detailed design provides additional design details when required, in order to meet quality and performance requirements.

Subsequently, based on the detailed designs of the technology and operations components, the Security Risk Assessment may have to be updated. A change in the Security Assessment may also drive a change in the detailed designs. One typical update to the Security Assessment at this stage would be a fulfillment of a security control, which would show the progress of the security assessment during the project execution.

C. Implementation Phase

The main activity during the implementation phase is the building of the application. However, several other related activities are foreseen by the SSDLC.

Build Application and Security Controls. The developer teams will need to complete the following steps when building security technology components:

- Build detailed technical component configurations;
- Prepare test data for unit and integration tests;
- Execute unit and integration tests;
- Fix defects.

A major activity at this stage is also the development of security operations processes and procedures for issue escalation or breach notification procedures.

Develop security test plan. A security test plan is required in order to validate the security of a deliverable and to test the controls defined in the detailed design which map back to the requirements. The test plan should cover testing security functional and security assurance requirements. It shall be subject to a formal review. In case a security certification is foreseen for the application, the security test plan will be a central element in the evaluation process.

Execute application scanning. Vulnerability testing is a leading-practice means of determining whether a system has exploitable weaknesses. These vulnerabilities may exist because of errors in configuration or deficiencies in application code. The Security Team will coordinate with the Project Team to obtain test data, credentials etc. required to conduct this activity. The scan is executed in three levels – application scan, code scan, and platform scan.

Update Security Risk Assessment. The Security Risk Assessment document is be published again with updated risks. Furthermore, now that the application is built, are more detailed risk analysis can be conducted and previously unidentified risks may be found.

D. Test Phase

Following the implementation phase, the test phase represents a central element in the SSDLC.

Security control testing. The security control testing process involves incorporating testing criteria for security controls into system, performance, integration, and acceptance tests. In this stage, the environment should be prepared for security control testing and the test scripts should be executed. Defects should be identified, documented and fixed. Verification is complete after all test cycles have been executed and defects have been fixed. Security control testing involves a high level of coordination with the Security team. This testing can be time and resource intensive since it involves extensive collaboration with teams, and it is recommended that delivery estimates include ample time for this activity.

Complete application scanning. Completing the security accreditation process involves completing application and platform vulnerability scanning tests using approved application and platform scanning tools in addition to completing all other application-specific accreditation requirements. It is at this stage absolutely crucial that the security team acts completely independent from the system development team. Otherwise, results may be unintentionally falsified and potential certification goals can be threatened.

E. Operations Phase

During the deployment phase, a transition team must be mobilized to support the operations team to ensure optimal support, resolve issues, and document resolutions. This team is a temporary proxy for the production team and will manage the transition to production. It is important to involve the production support team in pre-production activities to prepare for a smooth transition.

F. Summary of SSDLC Activities

Summarizing the required additional activities of the SSDLC, it is clear that a substantial amount of additional effort is required for secure web application engineering. In the past, this has led to reluctance from stakeholders to implement all stages of the SSDLC. As described in Section VII, our generic security patterns help to reduce this overhead to an acceptable level.

VI. Information Risk Assessment

The Information Risk Assessment lies at the core of the SSDLC. As highlighted in the previous section, the information security risk assessment has to be iterated in every step of the SSDLC. Here, we describe the information security risk assessment in greater detail.

A secure information model, which is a direct result of the information risk assessment, allows security practitioners to determine the degree to which specific types of information must be protected. The intent of the model is to appropriately protect information assets based on a balanced view of their value, the risk to the enterprise, and the cost of protection. This model follows a seven step process to identify information assets, identify security risks, and identify and apply required security controls. A high level description of this process is described in the following. It should be noted that this process is the same, no matter which specific risk assessment methodology is used (e.g. NIST, ISO 27000, EBIOS etc).

1. Identify and categorize data - Identify and categorize data by their business usage and then categorize the data using an information categorization process defined in this model. To help identifying the various types of data handled by the system, a system data identification questionnaire is to be created and implemented by the project team. The three characteristics of data security – confidentiality, integrity and availability – must be considered when identifying risks to data and the controls needed to mitigate those risks. For each characteristic and data, a rating on a previously agreed scale will be created following the evaluation of the questionnaires. This procedure will provide a very good image on which data in the system required which level of protection.

2. Identify application and infrastructure components – In this step, the application systems that will process, transport, and store the data that has been identified in step 1 are identified. To achieve this, two additional questionnaires need to be prepared and completed. The application identification questionnaire describes the components of the application and links them to the previously identified data types. The infrastructure identification questionnaire describes the environment in which the application is to be deployed.

3. Develop a security data flow – The purpose of the security data flow is to diagram the interactions of the data, application, and infrastructure elements of the system. Understanding the flow of data will assist in both the risk and control identification processes. Developing a security data flow assists in the identification of applications and infrastructure associated with the application where security controls are required. Development of the security data flow diagram is an iterative process. As system components, risks, and controls are identified, the diagram will be updated to reflect this information.

4. Identify Security Risks – Identify risks to the data and systems that were identified in the previous steps. Risks are assigned a rating for impact (possible outcome if risk is realized) and probability (chance that risk will be realized). The overall rating for the risk is given by the product of the values assigned to impact and probability. It is important to identify, prioritize, and address weaknesses in a system that could potentially lead to the compromise of the confidentiality, integrity, or availability of the data that is managed by the system. Security risk is a calculated measurement that is used to prioritize response to threats to the confidentiality, integrity, or availability of an information asset. The required information is extracted from information documented in the previous steps of the process and an understanding of common information security exploits. Risks are assigned a rating for impact (possible outcome if risk is realized) and probability (chance that risk will be realized). The overall risk measurement is calculated by multiplying the values assigned to impact and probability.

5. Identify Required Controls - The data elements that have been identified through this process are then analyzed based on classification and risk of exposure to determine which types of security controls must be applied to them. Security controls are process and technology elements that are used to minimize the risks of threats to an information asset or organization. The security controls selected for a given project are typically selected based on the value of the asset, the associated cost to implement the controls for protecting the valued asset, and the risk of security threats related to that asset. Managers, including the security expert for a given project, will determine the value and work closely with the project team to calculate the trade-off costs of various security control options.

VII. Operational Issues of the GASF

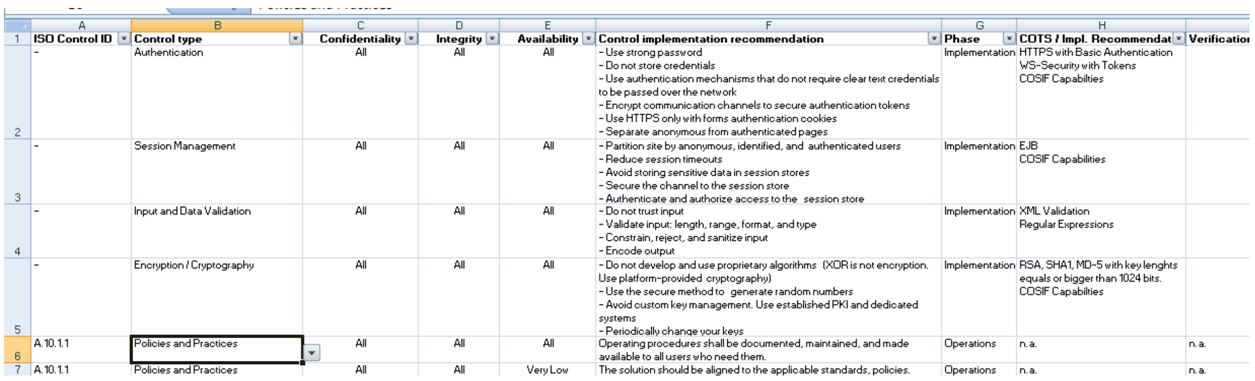
In this section, we address operational issues related to the GASF. We discuss security patterns, tools and methodologies, followed by some considerations regarding security certification.

A. Security Patters, Tools, and Methodologies

As mentioned earlier, the introduction of the GASF, and in particular the SSDLC, implies a significant overhead on the application development process. We now address the tools and patterns that are collected within the application security framework in order to reduce that overhead.

Applications from either of the two domains supported by the GASF (web applications/ web services and legacy applications) share a large number of commonalities, fixed security patterns can be defined for and mapped to each possible information risk assessment result. The GASF toolset maintains a catalogue of security controls for each domain and the development team can immediately assign appropriate security controls once the initial risk assessment has been completed. Thus, the information risk assessment process is dramatically shortened and can be implemented much more efficient. Figure 4 shows a screenshot from the toolset.

Security patterns exist for all kinds of security controls needed for applications within the SSA system-of-systems environment such as access control, authentication, or secure communication.



	A	B	C	D	E	F	G	H	
1	ISO Control ID	Control type	Confidentiality	Integrity	Availability	Control implementation recommendation	Phase	COTS / Impl. Recommendat	Verificatio
2	-	Authentication	All	All	All	- Use strong password - Do not store credentials - Use authentication mechanisms that do not require clear text credentials to be passed over the network - Encrypt communication channels to secure authentication tokens - Use HTTPS only with forms authentication cookies - Separate anonymous from authenticated pages	Implementation	HTTPS with Basic Authentication WS-Security with Tokens COSIF Capabilties	
3	-	Session Management	All	All	All	- Partition site by anonymous, identified, and authenticated users - Reduce session timeouts - Avoid storing sensitive data in session stores - Secure the channel to the session store - Authenticate and authorize access to the session store	Implementation	EJB COSIF Capabilities	
4	-	Input and Data Validation	All	All	All	- Do not trust input - Validate input: length, range, format, and type - Constrain, reject, and sanitize input - Encode output	Implementation	XML Validation Regular Expressions	
5	-	Encryption / Cryptography	All	All	All	- Do not develop and use proprietary algorithms (XOR is not encryption. Use platform-provided cryptography) - Use the secure method to generate random numbers - Avoid custom key management. Use established PKI and dedicated systems - Periodically change your keys	Implementation	RSA, SHA1, MD-5 with key lenghts equals or bigger than 1024 bits. COSIF Capabilties	
6	A.10.1.1	Policies and Practices	All	All	All	Operating procedures shall be documented, maintained, and made available to all users who need them.	Operations	n. a.	n. a.
7	A.10.1.1	Policies and Practices	All	All	Very Low	The solution should be aligned to the applicable standards, policies.	Operations	n. a.	n. a.

Figure 4. Typical screen from the GASF toolset.

Of course, some elements of the target application may implement special functionality that is not covered by the available security patterns. In this case, a reduced information risk assessment has to be executed. However, it has to be noted that this is only required for the additional functionality with respect to the basic application template. Once this has been done, the result is synthesized in a new security pattern and may be reused later for other applications.

Aside from tools that are directly related to the guidance of the developers for applying the SSDLC to new developments, a large variety of commercial and open-source tools and methodologies exist to support the SSDLC for both application domains. The GASF lists the most useful of these tools for each domain of applications.

Examples for such tools include, among others, automatic test tools, security requirements engineering tools, and code checkers. They introduce a significant amount of automation and thus can further reduce the secure development overhead.

B. Security Certification

Security Certification is a way to validate the security of the target application and to prove that all the imposed security functional and security assurance requirements have been met. Security certification is usually executed by an independent third party, in most cases accredited by a governmental entity.

International standards such as the Common Criteria [9] exist that are used to certify information technology products. Depending on the level of certification, a constant evaluation of the application throughout its development lifecycle is required. The SSDLC provides the necessary foundations to execute such an evaluation.

VIII. Next Steps and Conclusions

The development of the Generic Application Security Framework is completed for the web services and web applications domain while work is going on for the legacy application domain. The web application/web service specific GASF has been deployed operationally in a number of pilot software development projects to assess its suitability in practical use. The gained feedback and lessons learned will be incorporated in the evolution of the GASF. The currently ongoing evolution of the GASF will also revise and improve the available toolset and in particular extend it to also be able to address legacy applications. Once both domains are fully supported, the GASF can be used to improve the security and robustness of most mission data system applications.

To summarize, in this paper, we have presented a generic application security framework for mission data systems that is capable of increasing significantly the robustness and security of the applications. While initially only foreseen for use within the system-of-systems environment of the European SSA Initiative, the framework is also used in other areas. The framework provides a set of tools and templates that aim at reducing the effort related to the additional steps introduced in the SSDLC. Thus, the framework helps developers with efficient secure application development. We are confident that the application security framework will drastically improve the security and robustness mission data systems for the, while at the same time keep the related additional effort minimal.

References

[1] P.G. Carlock, S.C. Decker, and R.C. Fenton. Agency level systems engineering for systems of systems. Systems and Information Technology Review Journal, Spring/Summer:99–109, 1999.

[2] T. Earl, Service-Oriented Architecture: Concepts, Technology & Design, Prentice Hall/ Pearson PT, August 2005

[3] European Cooperation for Space Standardization, Software, ECSS-E-ST-40 C, Issue 3, March 2009

[4] ISO/IEC 27001:2005, Information technology -- Security techniques -- Information security management systems – Requirements, October 2008

[5] N. R. Mead, E. Hough, T. Stehney II, Security Quality Requirements Engineering, Technical Report CMU/SEI-2005-TR-009, November 2005

[6] The Open Web Application Security Project (OWASP) – http://www.owasp.org

[7] ISO/IEC 27002, Information technology - Security techniques - Code of practice for information security management, July 2007

[8] Bundesamt für Sicherheit in der Informationstechnik – http://www.bsi.de

[9] Common Criteria for Information Technology Security, July 2009, Version 3.1, Revision 3

[10] J. Epstein, S. Matsumoto, G. McGraw, Software security and SOA: danger!, Will Robinson, IEEE Security and Privacy, pp 80-83, vol 4, Iss, 3, February 2006

[11] M. Sarkarati, M. Spada, "Leveraging Advanced Software Technologies for Implementing the European Space Situational Awareness Ground Data Systems", SpaceOps 2010