

Organizational Measures for Secure Software Development

Author:

EUROSEC GmbH Chiffriertechnik & Sicherheit

Tel: 06173 / 60850, www.eurosec.com

© EUROSEC GmbH Chiffriertechnik & Sicherheit, 2005

Contents

- Focus on non-technical tasks and issues that require management decisions and support, as well as thorough planning prior to beginning the development lifecycle
- Target group: Security officers, developers with management responsibilities, and everybody else who needs to ensure the security of a software product
- Describes workflows, risks, countermeasures, as well as specific tasks
- This presentation does not address technical issues like attack methods or specific vulnerabilities

Make or Buy Decisions

- Specific Security Modules should be developed by security experts; to buy such components can avoid a lot of security problems
 - Cryptographic Components
 - User Management
 - Session Management
 - Authorization/Authentication
 - Logging/Auditing
 - Input/Output Validation Filter Frameworks
 - etc.

The Usual Project Management Tasks

- Quality Management is a necessary precondition for good security
- Thus, for security projects, focus on projekt management tasks even more than usual
- Project delays imply insufficient time for final testing, which implies a higher security risk
- Whatever goes wrong (in term of security issues), it should be possible to track this down to an individual that is personally responsible

Define Responsibilities

- Besides the usual organizational roles, for each of the following issues there should be somebody who is responsible:
 - product in total (not only security)
 - product security
 - development in total
 - secure development
 - requirements specification (including security)
 - design specification (including security)
 - testing (modules as well as final product)
 - shipment issues
 - installation issues
 - documentation
 - quality (not security-specific)
 - patch management
 - bug reporting and –tracking
 - etc.

Provide Training Material for Administrators and End Users

- If administrators or end users cannot deal with the security options of your product, then you don't need such product security at all
- Plan for appropriate resources to provide training material for administrators and end users of your product
- Do not only think about handbooks, but also about online help in the product, internet help pages, forums, demo software, sample installations, checklists, step-by-step guides, training courses (online and/or face-to-face), etc.

Provide Security Training for Developers

- What you should know: even very good software developers can do a bad job with respect to security
- Thus, recruiting excellent staff is not sufficient to deal with security problems
- It is important to provide specific training courses for developers, dealing with the following issues:
 - how do hackers think
 - what are well-known attack methods, how do they work, how can one avoid them
 - demos, how to hack a system
 - best practices (concerning the development environment in question)

Provide Security Training for responsible Managers

- If the responsible management does not understand the importance and advantages (in terms of money) of early security measures during the development phase, there will be insufficient support and resources for such measures. Therefore,
 - demonstrate cost advantages of early measures as opposed to later patches
 - teach managers on which kind of security procedures exist during the different development phases (planning, testing, auditing, documenting, etc.)

Development Environment Decisions

Remark: most of the following will already be decided, prior to any security considerations; nevertheless, think about your options

- Programming Language (big differences with respect to security, e.g. Java versus C)
- Software Method (e.g. Waterfall Model, etc.)
- Compiler
- CASE Tools
- Operating System Platform
- Versioning Tool
- Development and Test Systems Security

Identify Government Regulations

- Depending on country, industrial sector, product focus etc., there might exist regulations that your product has to fulfill
- Some examples:
 - Data Privacy Laws (e.g. Germany)
 - FDA (US)
 - Sarbanes Oxley (US)
 - Usability for Disabled People (public sector contracts)
 - etc.

Product Security Documentation

- provide sufficient resources for product documentation, especially for security issues; involve people that are not part of the development team
- provide detailed information about security issues within your product manuals (installation, administration, end user, etc.)
- do not scatter security topics within your manuals, but provide one specific section comprising all relevant security details (this also holds for administrative interfaces)
- provide a secure out-of-the-box installation, then describe in your documentation how to „open“ the installation for additional functionality, - not the other way around

Internal Security Documentation

- Concerning security documentation, the whole development process should be covered:
 - requirements specification, design and architecture specification, source code, testing, etc.
- It should be well documented, who approved the individual development phases
- all unsolved issues should be documented (e.g. requirements that have not been implemented because of technical or cost limitations)
- known security problems should be documented; limited access should be given to such information

Rating and Approval Process

- For each phase in the development cycle, some technical inspections are necessary to ensure that all major requirements are fulfilled
- Thus, a formalized rating and approval process has to be defined
- Agree on critical lifecycle phases that need some form of inspection; provide checklists with rating criteria, responsible person, re-evaluation procedures in case of bad ratings, etc.

Code Check-In Procedures

- New or modified code fragments need to be checked in to the code base
- this requires not only an appropriate versioning tool, but also a check-in process that defines minimum requirements for such code:
 - compliance with existing (security) guidelines
 - responsible author
 - time and date of check-out and check-in
 - approval of a colleague or technical manager (optional)

Acceptance Criteria for Test Procedures

- During all phases in the development lifecycle, specific test procedures have to be performed (e.g. for design, architecture, code modules, complete application, installation process, etc.)
- Each test requires detailed acceptance criteria that need to be defined in advance, together with the management; it should be clear what happens if a test failed:
 - continue with the next step, and fix the problem later
 - stop and fix the problem, then go on to the next step
 - write a remark in the test protocol, but don't bother and go on
- Since a lot of tests will reveal existing problems, a practical re-evaluation process need to be installed

Improvement Process Definition

- Not only bugs and security problems imply necessary improvement
- It is also desirable to add useful security functionality to future versions of your software
- Thus, define an improvement process that may include the following:
 - ask customers about security features they miss
 - look for changing technology and resulting new threats
 - identify new attack scenarios that need countermeasures
 - track the competition and adopt their security features
 - ensure that your product meets (new?) government regulations

Backup and Protection of Development Data

- Most companies have well-established backup and recovery procedures for their standard user PCs
- Some companies, however, have special development areas and PCs that do not use the same security and backup mechanisms than the other default users
- Therefore, it is important for such areas to ensure that sufficient countermeasures against security threats (e.g. code theft) and loss of data have been implemented

Error Reporting Process

- All identified errors should be tracked in an appropriate database tool
- Each error should be rated with respect to its security implications (e.g. “not security relevant”, “security relevant”, “security highly critical”); this allows product security staff to track “their” bugs
- Also, include a rating with respect to the difficulty to fix the problem (e.g. a qualified guess: „less than 2 hours“, „less than 1 week“, etc.)
- Moreover, include information about the „blame factor“: who is to blame for this bug (at least a qualified guess)? E.g. own developers, externally bought code components, operating system, management, etc.
- It may happen (hopefully not) that more new security bugs will be reported than you can fix old ones; for this case you need an appropriate scheduling that does not completely ignore bugs of minor importance, since this might frustrate the reporting party (e.g. customers)

Patch Management Process

- In many scenarios it will not be possible to wait with security patches until another major release of your software will be installed
- Thus you need to prepare for a practical patch management process (how to inform all customers, how to get patches to the customer, how to ensure that the patch doesn't harm the system, how to install the patch in spite of time limitations, etc.)
- This is an important requirement that has to be considered prior to the design specification

Security Incident Warning Process

- New security incidents as well as potential threats related with using your software should be reported to customers, to your own management, and to your developers
- Of course it will depend on the target group how to „sell“ such news...
- Plan appropriate workflows and ways to reach your target groups
- Don't hide problems, but avoid to publish details that would help an attacker before you can fix the problem

Risk Assessment versus Baseline Security

- When preparing your security requirements specification, you have to decide whether you assume „standard“ security needs (e.g. a prepared list of baseline requirements), or whether you perform an individual risk assessment to identify the actual requirements depending on use scenarios
- Provide the following aids:
 - list of baseline security settings
 - checklist with risk assessment questions (different groups need to be involved to get the correct answers)
 - define different workflows (and assessment depths), depending on the actual security importance of the code that has to be developed

Customer Requirements Evaluation

- Still a lot of companies develop applications for customers without interviews to get their requirements
- We are aware of several cases where specific customers or industry sectors demanded non-standard security features that were by no means obvious
- Thus, prepare a process and corresponding checklists for interviewing your customer target groups

Test Procedures

- Assume that there is one team of developers, also responsible for testing, in a project with a strict final deadline and a test phase that follows after the coding phase; => this is designed to fail!
- Every developer will prefer coding as against testing
- Hence, coding will never (voluntarily) be ended in favor of testing
- Thus, design test processes that avoid this problem, either by separate testing staff or by defining test schedules that are mostly not at the end of the development phase

Liability Limitation

- Despite all technical countermeasures, there will always remain some risk that your software does not meet the requirements agreed before
- Thus, do not forget to include appropriate liability limitations into your contracts
- On the other side, if you licensed some code from external providers, ensure that their liability limits match your own contracts with your customers, since otherwise you may be held liable for mistakes that you do not take responsibility for

Insurance Coverage

- Even if you limit your liability for potential damage resulting from security deficits, you may still have some liability that implies a substantial business risk
- Therefore, it should be found out whether there exist sufficient insurance coverage that helps you in worst case scenarios
- Ask your lawyers, before you provide software to others without knowing all legal consequences

Source Code Escrow

- Suppose that somebody claims that your software contains a significant security threat, for example a hidden backdoor
- In such cases it might be helpful if you can provide evidence that this is not true
- If you deposit your code (sources, compiler and compiled code) at a trusted third party, you are able to prove whether such a reproach is justified or not

Necessary Certifications

- In some cases you may decide to get a security certificate for your product
- In case of Common Criteria Certificates, it is much simpler to provide the required paperwork if you know all certification requirements in advance and can adopt your documentation and processes accordingly
- If your certification demands formal evaluation methods, for example, this may not be possible if the coding already happened; instead, the formal process and evaluation tools starts in parallel to your specification work prior to coding
- Be aware of formal methods as being very costly and time consuming; moreover, in contrast to theory, practice shows that evaluated products are not necessarily more secure than other products and may still contain serious security deficits, for various reasons
- Thus, before starting with development, make your own decision concerning the pros and cons of the different security certificates available on the market

Conclusion/Recommendation

- A lot of security-specific tasks and processes need to be handled, at different stages of the product lifecycle
- Try to embed these tasks into your overall software development and product management, and try to avoid any incompatibilities with already existing workflows
- Ensure sufficient support and understanding from all people involved, since this is crucial for your success
- Never stop trying, start with so-called 80-percent-solutions and then improve them step by step

Appendix

3	4	5	6	7	8
0	0	0	0	0	0
3	4	5	6	7	8
		1			
		2			
2	3	3	3	3	3
-		4			
		5			
3	4	5	6	7	8
6	6	6	6	6	6
		7			
		8			
9	9	9	9	9	9
3	4	5	6	7	8
0	0	0	0	0	0
3	4	5	6	7	8
		1			
		2			
2	3	3	3	3	3
-		4			
		5			
3	4	5	6	7	8
6	6	6	6	6	6
		7			
		8			

Why this presentation from us? - our professional background:

- several project years‘ in research projects on secure software development, with partners like SAP, Deutsche Bank, Commerzbank, Universities, etc.
- large amount of application vulnerability checks for software companies, including reviews and coaching for developers
- Security requirements- and design specifications for large development projects
- Writing of company guidelines and checklists for secure development, mostly in banking and finance sector
- Reverse engineering and reviews of security mechanisms and crypto algorithms
- Implementation of security functions for customers

Final Remark

- the present work has been conducted within the so-called **secologic** research project (term: 2005+2006)
- for more details see www.secologic.org
- we are grateful to the German Ministry *Bundesministerium für Wirtschaft* for supporting this project
- we appreciate all suggestions and feedback with respect to our presentation slides and white papers

Copyright Remark

- This document has been prepared by EUROSEC and serves the purpose of conducting courses and seminars about secure software development (focus on web applications)
- We published these slides to support further activities in the development of better software applications
- These slides can be used for your own purposes/employees within your company, as long as you include an information about authorship by EUROSEC
- Commercial use by companies specialized in seminars and/or consulting, is not allowed without a separate agreement; please contact us: kontakt@eurosec.com