Global Information Assurance Certification Paper

The Role of the Security Analyst in the Systems Development Life Cycle

Brad Gray, MBA
GIAC Security Essentials Certification (GSEC Practical Assignment)
January 12, 2005.
Option 1

Abstract:

This document discusses security considerations during each phase of a generic development life cycle. The generic phases used for this document are Planning, Analysis, Design, Implementation, and Support. Before expounding on the security considerations encapsulated in each phase, each major section of this document briefly discusses the spirit of the life cycle's phase, highlights the responsibilities of a security analyst during that phase, and often compares the similarities and differences between the developer and the security analyst's roles. In the end, a broad group of security topics are addressed. These include the differences between program and issue-specific policies, CIA (confidentiality, integrity, and availability) and Risk Assessment, different levels of Security Strategies, levels of an Application Security Management Plan and how to manage vulnerability assessments in application testing, and other best practices.

Introduction

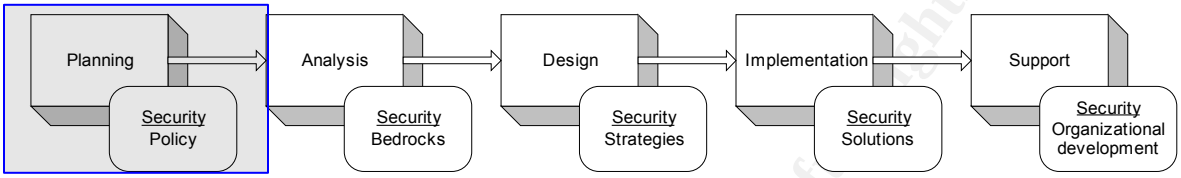
The catalyst for this document stems from real-life experiences while providing security services to application developers. In light, this document may serve as a guide to developers who would like to understand the security analyst role, developers who wish to incorporate the security analyst skill set into the developer skill set, or to those who lack a general understanding of the relationship between data security and the System's Development Life Cycle (SDLC). This document serves as high level guide to expose the relationship between requisite security considerations and the development life cycle.

This paper will proceed in a very logical manner to describe how a sequential development life cycle increases in depth as security is applied. Each major portion of the paper will address a phase of the system development lifecycle. In the end, this document will describe a repeatable process where enterprise security policies allow business requirements to be met through the execution of appropriate security strategies and solutions. How the SDLC couples with data security is reflected in diagram 1.0, on page 2.

1.0 Planning

The generic "phases" used for this document are Planning, Analysis, Design, Implementation, and Support.[1] These five phases comprise a generic system development life cycle (SDLC). These are clearly reflected in diagram 1.0 below.

Diagram 1.0



In addition, to the five generic life-cycle phases, diagram 1.0 also has generic security pieces associated with each phase. These are Policy, Bedrocks, Strategies, Solutions, and Organizational Development. It is important to understand how security couples with the SDLC because future sections of this paper will elaborate on these relationships.

Section 1.0 of this document focuses on the planning phase of the SDLC. The planning phase in any development methodology begins with a "first-cut" at gathering business requirements. This may be done through meetings with business partners, discussing job responsibilities, or interviewing customers. This is the place where the members of the development team understand the spirit of the effort which they need to accomplish.

Again, this document is meant to be a reference for understanding how data security considerations can be made during the system's development life cycle. For the developer, the planning phase generally offers an opportunity to understand the project's scope, what resources are available, and the project's timeline.

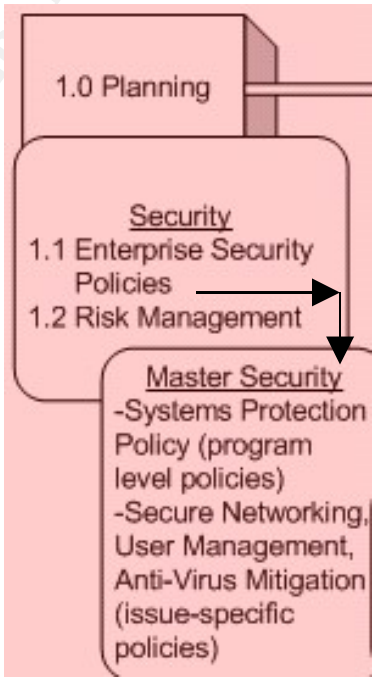
For the developer responsible for security, or the independent security analyst, this is where the business requirements and risk management first shake hands. Therefore, in the Planning phase there are two elements the security analyst has to consider for the organization. These elements include the enterprise security policies and risk. These elements are discussed in section 1.1 and 1.2 of this paper. After these subsections of the Planning phase are explained, the Analysis phase will be discussed.

[1] Whitten and Bentley 1998, 9-11

1.1 Enterprise Security Policies

"A policy is a guideline or directive that indicates a conscience decision to follow a path towards a specific objective. Often a policy may institute, empower resources, or direct action by providing procedures or actions to be carried out."[2] The organization's enterprise security policy is the first element the security analyst must understand. While it is not critical to understand every constituent policy of the enterprise security policy at this time, it is critical to understand the high-level security policies. SANS calls the organization's high-level security policies "Program Policy(s)". The program policy "sets the overall tone of an organization's security approach…can provide direction for compliance with industry standards…as well as with applicable law and government regulations."[3]

Diagram 2.0



Practical experience reflects the enterprise's security policy is made up of seven to ten program "master" policies. These master program policies reflect the most important areas where security must be applied to maintain or maximize business functions. These master policies generally encompass multiple issue-specific policies, system-specific policies, local policies, and other security procedures. For instance, a company may have a "Systems Protection" enterprise security (program) policy, which encompasses several other (issue-specific) policies like "Secure Networking", "Development Life Cycle", or "Antivirus Protection". The relationship between the Planning phase, security considerations, and master level security Policies is reflected in diagram 2.0. The relationship between master level security policies and other security levels is clarified in section 3.0, on pages 7, 8, and 9.

1.2 Risk

The second element in the planning phase the security analyst must understand and consider, is the element of risk. Since during this planning phase is the first time a security analyst becomes to understand the system's objectives, this is where risk can begun to be understood. From these early discussions, the security analyst will begin to be able to gauge whether the effort will expose (or be exposed) to a high, medium, or low level of risk. While the risk analysis

[2] Sans Defense-In-Depth 1.2, 73
[3] Sans Defense-In-Depth 1.2, 74

process will be much more detailed in the following Analysis phase, in the Planning phase the process is a simple approximation. This approximation is expressed in the equation:

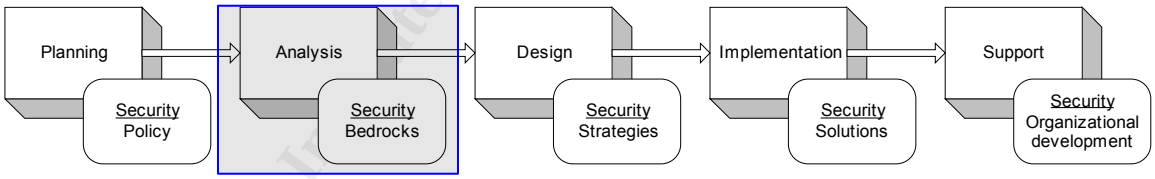
$$\text{Risk}_{\text{(to effort, organization, or object)}} = \text{Threat} \times \text{Vulnerability}_{\text{(of the threat)}} \times \text{Impact}_{\text{(asset value)}}$$

Let us use an example to illustrate. What is the perceived Risk of a person breaking into the author's house? This can be answered by looking at the equation's components. What is the threat? The threat is the author lives in terrible neighborhood, petty theft is common, and the front door is made of particle board. What is the vulnerability? The author often leaves the door unlocked. So by answering these questions, the reader can determine the risk to the author's house is fairly high. However, the closer any variable is to zero, the less the overall risk. For instance, what is the value of the assets (Impact) in the author's house? Well, counting the silverware, it is about a hundred dollars (very near zero). This brings the overall risk down to nearly zero.

Being able to assess the amount of risk with the aforementioned equation allows the security analyst to estimate the cost of security for the effort. For instance, the analyst may be able to estimate how many hours will be needed to deploy the effort securely, how much security resources may cost, or how much it may cost if security is ignored or the risks are simply accepted.[4] This is what can often be determined in the planning phase, in regard to risk.

2.0 Analysis

Diagram 3.0



In the second step of the SDLC, one finds the Analysis phase. While this phase generally involves acquiring and defining the system requirements, the security analyst's role requires going a step farther. In this phase the analyst must understand the tenets of Confidentiality, Integrity, and Availability (CIA). In addition, the analyst must formally assess and document the risk the system may expose or the risk the system may expose for the organization. These concepts are described in section 2.1 and 2.2.

2.1 Confidentiality, Integrity, and Availability

[4] Sans Defense-In-Depth 1.2, 123-124

The first element in the Analysis phase that must be understood by the security analyst is comprised of three sub-components. However, these sub-components are generally considered the "bedrock of security" in the security analyst world.

In the final paragraph discussing risk, the author describes the ability to estimate risk as an important precursor to determining risk management alternatives. In the case a risk reduction approach is chosen, the analyst must understand the tenets of Confidentiality, Integrity, and Availability.

2.1.1 Confidentiality

The spirit of the confidentiality tenet revolves around protecting a system, and ultimately the organization, from exposing information to unauthorized individuals. This may be information ranging from business strategies, competitive secrets, personal information about employees or customers, or other sensitive information. This is done by ensuring every system user can be identified. This cannot be underscored enough. "A crucial aspect of confidentiality is user identification and authentication. Positive identification of each user is essential to ensuring the effectiveness of policies that specify who is allowed access to which data items."[5]

2.1.2 Integrity

Once the tenet of Confidentiality is identified, the tenet of Integrity may be addressed. Understanding this tenet, allows the security analyst to emphasize the importance of accurate data. Without accurate data, the system at best cannot provide direction for its business partners. At worst, the system may lead to business partners or the organization to make poor decisions. Therefore, "Integrity is the protection of system data from intentional or accidental unauthorized changes".[6] While identification and authentication were paramount in respect to confidentiality, Integrity focuses on what the authenticated principle is authorized to do.

2.1.3 Availability

The final tenet in the CIA model is Availability. This tenet is present to remind the security analyst that no system adds value when users cannot access its resources. A system must be able to ensure it can deliver information to business partners on demand.

2.2 Risk Assessment

[5] Handbook of Information Security Management: Access Control. http://www.cccure.org/Documents/HISM/019-021.html

[6] Handbook of Information Security Management: Access Control. http://www.cccure.org/Documents/HISM/021-023.html

The second element in the Analysis phase builds on the CIA tenets by assessing the business's risk via a formal risk assessment methodology. There are multiple risk assessment methodologies in the industry. However, a common thread exists between them. In as much, all of the methodologies force an analyst to review a system's business requirements. Then assess the risk to the system's business partners if the confidentiality, integrity, or availability of the system's information is threatened. Being able to assess the system in this manner is imperative because it allows the level of risk to map to the appropriate technical solutions. These technical solutions are what allow the organization to manage the risk. For an excellent overview on risk assessment and the different methodologies available, refer to Vishal Visintine's, GSEC practical, An Introduction to Information Risk Assessment.[7] While it is not this document's intention to discuss individual risk assessment methodologies, this document does intend to discuss the importance for an organization to have a defined methodology for all analysts to follow during the Analysis phase. A standardized approach to assessing the system's risk, will allow for more consistent mappings to security solutions. The more repeatable and consistent the solution's mappings become, the less it costs to build security into the system.

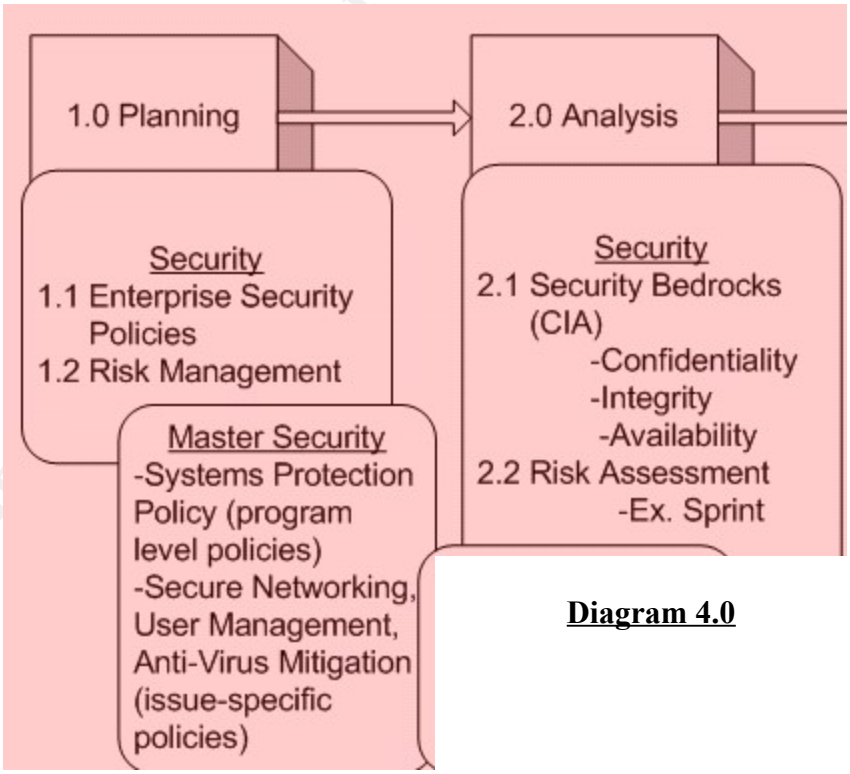


Diagram 4.0

Finally, a security analyst cannot follow a risk assessment methodology unless he or she is fairly well versed in multiple technical areas. The security analyst skills would likely include networking and the OSI model, firewall technologies, intrusion detection systems, cryptography, antivirus, operating systems (including scripting), of course troubleshooting, general security practices (physical and virtual), application security, and the common vulnerabilities and attacks that

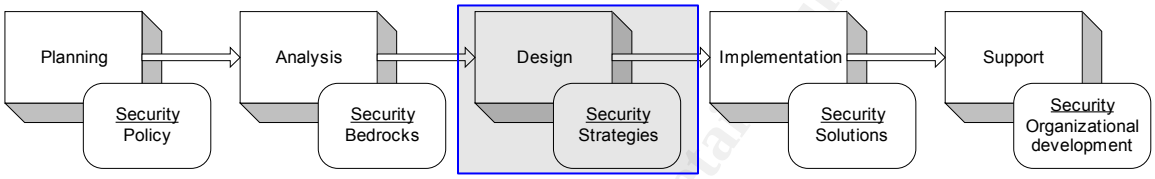
[7] An Introduction to Information Risk Assessment. 2003 http://www.sans.org/rr/whitepapers/auditing/1204.php

are int the environment (for example, those found atop the Sans Top 20 [8] or the OWASP Top 10 [9]). In closing, the security analysts overall technical skills must be strong enough to assess risk and understand the organization's security strategies, which are an integral part of the Design phase of the SDLC. Without these competencies, it will be very difficult to map technical solutions to the system's security needs.

Diagram 4.0, on page 6, summarizes the progression this paper has made in discussing the security considerations in the Planning and Analysis phases.

3.0 Design

Diagram 5.0



In general, the Design phase focuses on technical requirements, versus the Planning or Analysis phase where the focus is more on business requirements. For the security analyst, this is obviously the place where the organization's security strategies begin to get mapped to the effort's business requirements. Therefore, the security analyst must understand how the organization's "grand" security strategies service both the enterprise security policies and the business requirements. It is very important to understand the connection between security strategies at the "master" level and the security strategies at the "grand" level. In short, the grand level strategies that exist allow the execution of the security policies at the master level. The next few paragraphs will further exemplify this relationship and diagram 6.0, on page 8, will help the reader visualize the security considerations through the design phase.

3.1 Strategies

Let us review the security analyst position in the Planning Phase. In that phase, the analyst had to understand the Enterprise Policies, or the "program policies". These were really the seven to ten polices the organization needs to follow to be compliant with organization's overall goals. These program level policies also delimited some "issue-specific" policies. These policies were then serviced by security strategies which help further define the technical requirements needed satisfy any effort's business requirements.
A quick illustration will clarify the relationships between the phases, policies, and strategies. For example, let us assume during the Planning Phase, the analyst identified one of the program policies as the "System Protection Policy". In

[8] Sans Top 20 vulnerabilities. http://www.sans.org/top20/
[9] OWASP Top Ten Project. http://www.owasp.org/documentation/topten.html

accordance with best practices, the content of the policy has necessary prerequisites like purpose, background, scope, and responsibility.[10] In short, this high level program policy states the organization will provide a secure end to end computing environment for its employees, customers, and business partners. The organization will do so, by following "Secure Networking", "User Management", and "Anti-Virus Mitigation" issue-specific policies.

Now let us look at the security strategies that will service these policies. For instance, let us assume those in security understand the tenets of CIA and have developed a user management "Authentication and Authorization" security strategy. In addition, those in security recognize the need for defense in depth, trust and trusted and zones, and continuous availability. Therefore a "Perimeter Protection" strategy has been developed.

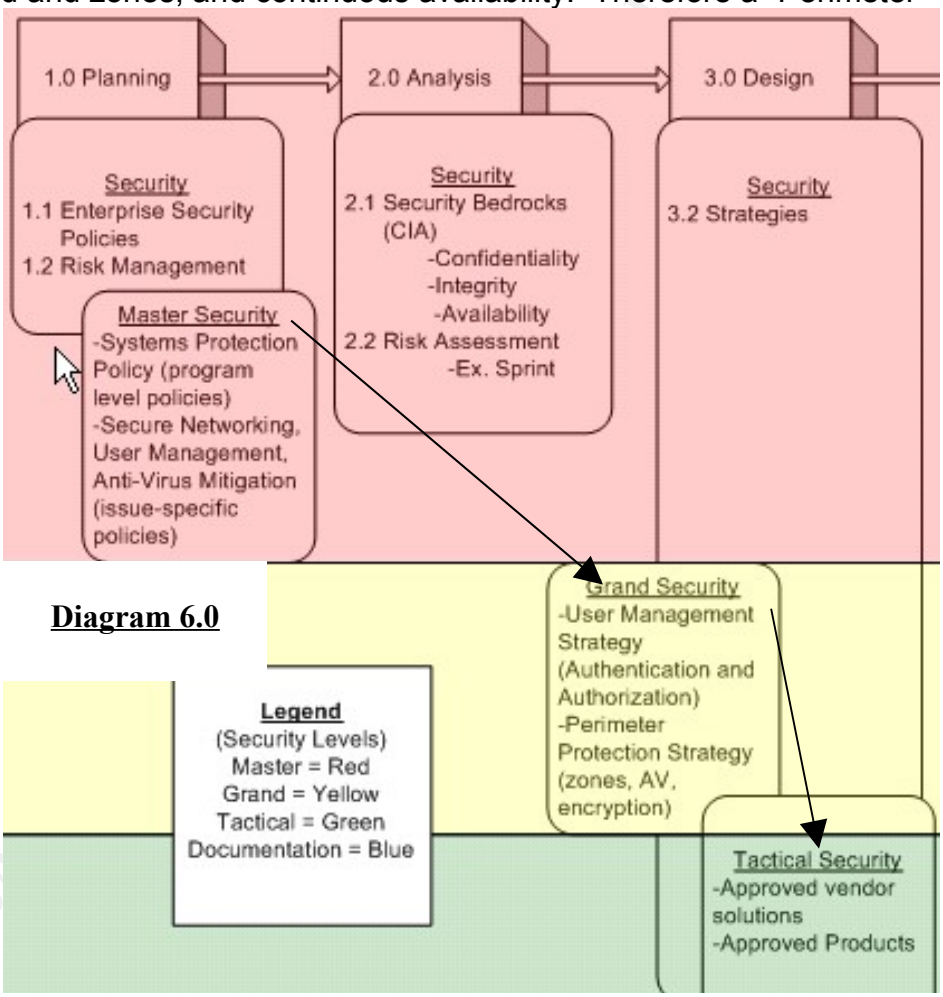


Diagram 6.0

The "grand" security strategies should also provide direction, as did the master security (program level) policies. For instance, the perimeter protection strategy should answer the question, "How will the strategy provide security?". In this case, the perimeter protection strategy outlines the use of networking devices to route and filter traffic, firewalls to maintain zone integrity, anti-virus software to protect from malicious code and executables, and encryption to protect sensitive data in transit or at rest. Furthermore, since this is the Design phase and the analyst needs to identify technical requirements, the perimeter protection strategy must provide approved "tactical" security solutions

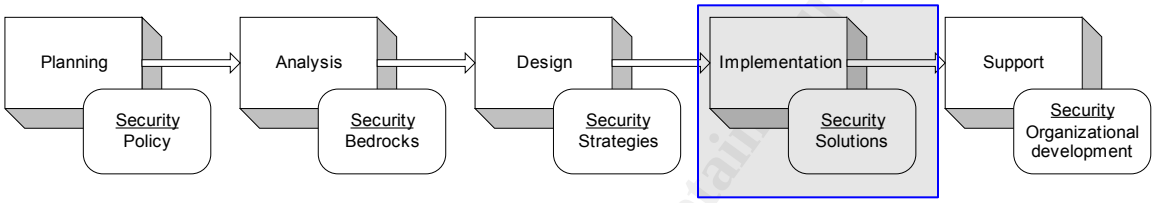
[10] Sans Defense-In-Depth 1.2, 75-76.

that can be deployed in a physical environment (example, vendor x firewall or vendor y antivirus). The "master" enterprise level security policies, "grand" security strategies, and "tactical" security solutions are mapped against the SDLC in 6.0 diagram on page 8.

4.0 **Implementation**

The next phase in the SDLC is the implementation phase, diagram 7.0. For both the developer and the developer/security analyst, this consists of building the system's individual components and testing the system's functionality.

Diagram 7.0



4.1 Deployed Solutions

While the developer and system analysts are deploying a vendor product or home-grown solution, the security analyst is often deploying or managing the infrastructure that will support the end solution. For example, while a developer is updating code on an application server, the security analyst may be building user or application identity's and groups, enabling SSL, deploying antivirus, or configuring firewalls.

The other significant difference between the developer and security analyst roles in the implementation phase, is in the way testing is done. Of course, both test for functionality. However, the security analyst generally needs to go a step further and test for vulnerabilities or exposures. Obviously, the more important the system the more vulnerability testing one would do. However, to limit the scope of this document and because there are several excellent papers in the Sans reading room on vulnerability and penetration testing as it relates to the networking and platform arenas, let us delimit the following paragraphs to discussing some of the best practices when evaluating and testing application security.

4.2 Application Security Charge

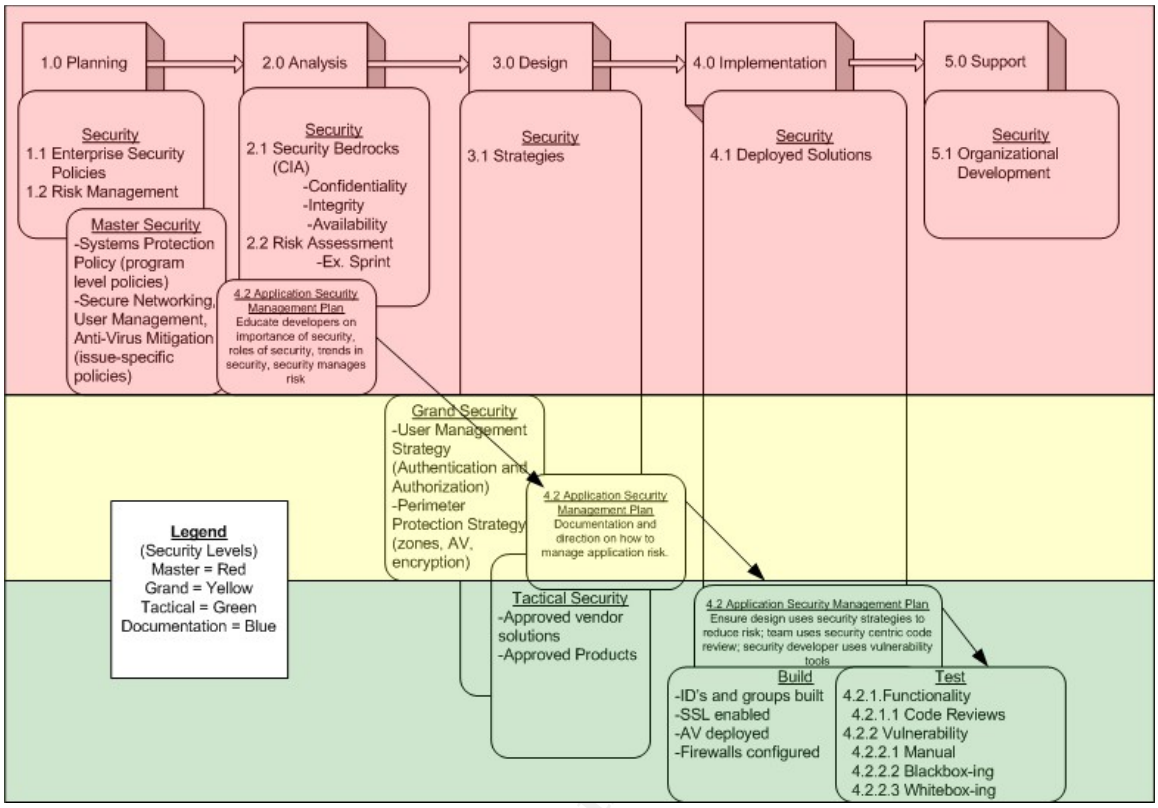
First, this document argues there should be an entity (either a single person or group, depending on the size of organization) that develops an application security program. While this group interfaces with the architectural groups and complies with security strategies and security best practices, this group is not a direction setter for enterprise security polices or security strategies. Instead, it is a group that is accountable for the development of an application security management plan within the SDLC, the broader information security context, and the business. An excellent example of this is Microsoft's Application Assurance Program. [11]

The Application Security Management Plan (ASMP) should also utilize a master, grand, and tactical framework. At the master level, executioners of the plan should focus on system education, ensuring everyone understands the importance of security in the enterprise and business, how security professionals identify risk, and what trends in security are emerging. At the grand level, the executioners of the application management plan need to ensure that all development teams understand how security fits into the SDLC and how security hopes to address known vulnerabilities. In addition, at the grand level, security assessments need to be performed and security risks need to be addressed within a risk management approach. This means providing the application security documentation and guidance for developers and security analysts to follow in order to ensure applications are secure in respect to the amount of risk they inherit. Finally, the application security management plan (ASMP) should have a tactical level. This is where security testing procedures are documented and outlined to address the most critical application vulnerabilities. At the very least, secure code reviews should be conducted. In addition, based on the organization's policies and the risk inherent in the system or application being built, vulnerability testing may also be needed. The reader can see the relationship to the ASMP and the separate SDLC phases in diagram 8.0.

Since this document is about security in the SDLC and not the application security management plan (ASMP), the master and grand level of the ASMP will not be discussed in detail. However, the tactical level of the plan is critical in the Implementation phase of the SDLC. It is important here because this is where testing for the effort occurs. Thus the testing that occurs in the tactical level of the ASMP is the application testing that should occur in the Implementation phase. With that acknowledged, this document can continue discussing the essential practices in application security testing. As discussed earlier, functionality is the paramount requirement. Next secure coding should be ensured via code reviews and application layer vulnerabilities should be discovered via vulnerability assessments. The reader can see the relationship to the ASMP and the separate SDLC phases in diagram 8.0.

8.0 Diagram

[11] Application Security Best Practices at Microsoft. 2003. http://www.microsoft.com/technet/itsolutions/msit/security/appsecbp.mspx#EDAA



4.2.1 Functionality Testing

Not a lot of detail needs to be articulated with regards to functionality testing. Security functionality testing should ensure the system or application meets the original business and technical requirements. In respect to application security, this usually means the system can be accessed by those entities that have been authenticated and authorized. However, a big piece of functionality testing should be code reviews. It is not enough that a solution simply works, it needs to be functioning as a result of deliberately following security guidance. This often means following secure coding guidelines. Therefore, code reviews can be encompassed within testing procedures.

4.2.1.1 Code Reviews

Code reviews are an activity that has been around since bad code has been written. Their popularity has only risen as the need for more efficient and reusable code has saturated the development environment. However, a code review or code walk-through should also be performed with security in mind.[12] Secure coding practices will undoubtedly play a large role in quality coding in the future. In addition, code reviews should be performed to validate compliance with security related best practices. Best practice reviews would consider OWASP's security guidelines and other topics covered in OWASP's, Guide to

[12] Trends in Web Application Security. 2004. http://www.securityfocus.com/infocus/1809,

Building Secure Web Applications.[13] By reviewing and understanding these principles and guidelines, the risk of many of the top application security vulnerabilities can be avoided. In the end, "One of the most effective ways of detecting application-level buffer overflows is via peer-level code reviews and inspections." Testing Web Security, chapter 6, p164[14]

4.2.2 Vulnerability Testing

Up to this point in the testing portion of the implementation phase, participants have tested functionality and used a code review group that ensured compliance with security policies. The functionality piece validated that the system met business requirements and authorized users had access, while the code review group validated secure code would not allow for common application related vulnerabilities. Now the penetration testing will go a step further by performing manual testing, blackbox testing, and whitebox testing.

4.2.2.1 Manual Testing

It is possible that while your system does allow access to authorized users, it may also allow access to users who should not have access. Therefore, it is important to perform a manual end-to-end review of the system. For instance, let us assume you have written a security script that enables security on a directory for group xxx. However, because of inheritance, that same directory has allowed other authorized users to access files. These types of logic errors can lead the developer or security analyst to believing a resource is secure when it is not. Regardless, it is imperative the access control assumptions created in the design phase are tested as the system is put into a production environment. Obviously, the more risk involved with the system, the more manual testing that would be performed. For instance, if sensitive data may be in transit, "sniffers" may be used to manually collect data to ensure sensitive data is not seen in plain text on the wire. Other procedures may include performing some sort of manual checkout of all security pieces. For instance, was required SSL enabled, was antivirus installed, were all of the patches placed on the machine, etc.

4.2.2.2 Blackbox testing

Two other techniques of application level testing are blackbox and whitebox testing. To oversimplify, blackbox testing is a functional testing approach which attempts to measure a system's effectiveness from an outside-in, or user, perspective.[15] However, from a security perspective, this approach is similar to how a system would be compromised by a user of the system via the application.

---

[13] OWASP Guide to Building Secure Web Applications and Web Services, The Open Web Application Security Project. http://umn.dl.sourceforge.net/sourceforge/owasp/OWASPGuideV1.1.pdf

[14] Splain 2002, 164

[15] Splain 2002, 5

A simple Blackbox test may start with "mirroring" the website. Black Widow is a commercial product that performs this act of "spider-ing" and downloading (http://www.networkingfiles.com/Download/BlackWidow.htm) a target's website. This is a great tool when an organization has fairly good security in place to protect internet facing applications. For instance, if the target has good defense in defense-in-depth mechanisms in place, attacking the home-grown business applications maybe the lowest hanging fruit. By being able to download the website to a local machine, the blackbox tester can search for weaknesses offline. The tool also makes it possible for the tester to dissect the content into smaller pieces and perhaps host part of the site in a lab. Thus the tester is able to emulate the client-server relationship that would typically exist between the user and the code.

Regardless if the web application is downloaded to a tester's environment, the simplest of blackbox penetration testing routines should have input validation as its core component.[16] Since the application has already been tested for functional input, it needs to tested for small amounts of invalid input. Obviously, as the application's risk grows more input validation needs to be performed via server-side mechanisms versus client-side mechanisms. In addition, the application should sanitize input to ensure that special characters and all other data inputs are accepted or rejected based on specific rules. Finally, the application needs to be able to handle large amounts of invalid input.[17] Spidynamics has a commercial tool for application testing. In addition, a portion of the spidynamics toolkit can be downloaded (https://www.spidynamics.com/products/listing/toolkit/WITools sdownload.html) for free for use against testing an application on the local host. Also, there is a nice whitepaper to get one started on using the tool's "fuzzing" capabilities.[18]

4.2.2.3 Whitebox testing

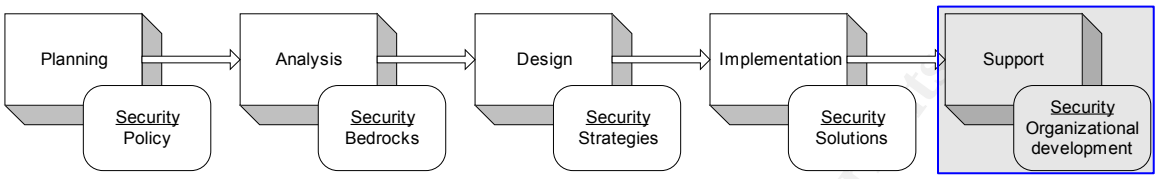
The second testing technique, "whitebox" testing, would be performed by someone who has an intimate knowledge with the design of the application. This person understands the business logic, the application's code, and often the technical environment in which both operate. While a portion of whitebox testing might occur during a code review, the primary whitebox testing should occur during the penetration testing phase. Whitebox testing should occur here because the penetration testing activities in this phase go the furthest in validating the quality of the code reviews and the application's ability to avoid being compromised. While whitebox testing does require access and review of source code, there are multiple automated code scanning tools available to help

[16] http://www.securityfocus.com/infocus/1704
[17] Splain 2002, 165-170
[18] Spidynamics Application Security: White Paper. http://www.spidynamics.com/support/whitepapers/index.html

speed up the testing. For an excellent overview on source code analysis tools, refer to Thien La's, GSEC practical, Secure Software Development and Code Analysis Tools.[19]

5.0 Support

9.0 Diagram



The final phase in the SDLC, is the Support phase (see diagram 9.0). This phase is concerned with the execution of two activities. The first activity the security analyst must perform is related to organizational development. The second is concerned with documentation.

5.1 Organizational Development

Organizational development is about practicing continuous improvement. It is especially important for the organization to continue learning and improving. If a security analyst found a flaw in an application or one of the approved vendor solutions, then it would be imperative the analyst provide that feedback to the necessary individuals or groups who recommended or support the product. This feedback to business partners on a solution's success provides value anyone deploying the solution in the future.

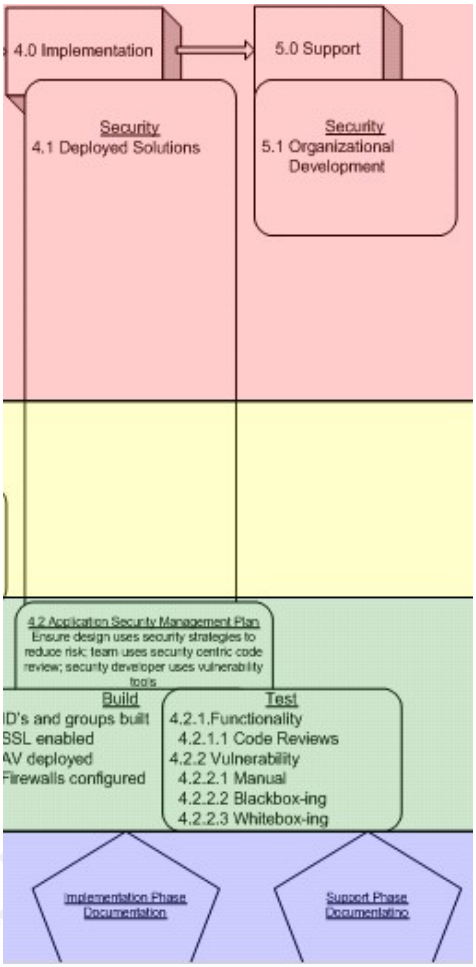
5.2 Documentation

Diagram 10.0 has been provided on this page to aid in this discussion concerning documentation. When referring to the diagram, the reader will notice each phase has a vertical arrow pointing to the phase's corresponding documentation container. This reflects the importance of documenting each phase's activities. Also, when referring to the diagram, the reader will see a horizontal arrow pointing to the "Implementation Phase Documentation" container and to the "Support Phase Documentation" container. This reflects how documentation from a previous phase must often be included in a following phase and ultimately the "Support Phase Documentation. It is important that any critical information be included in the final "Support Phase Documentation".

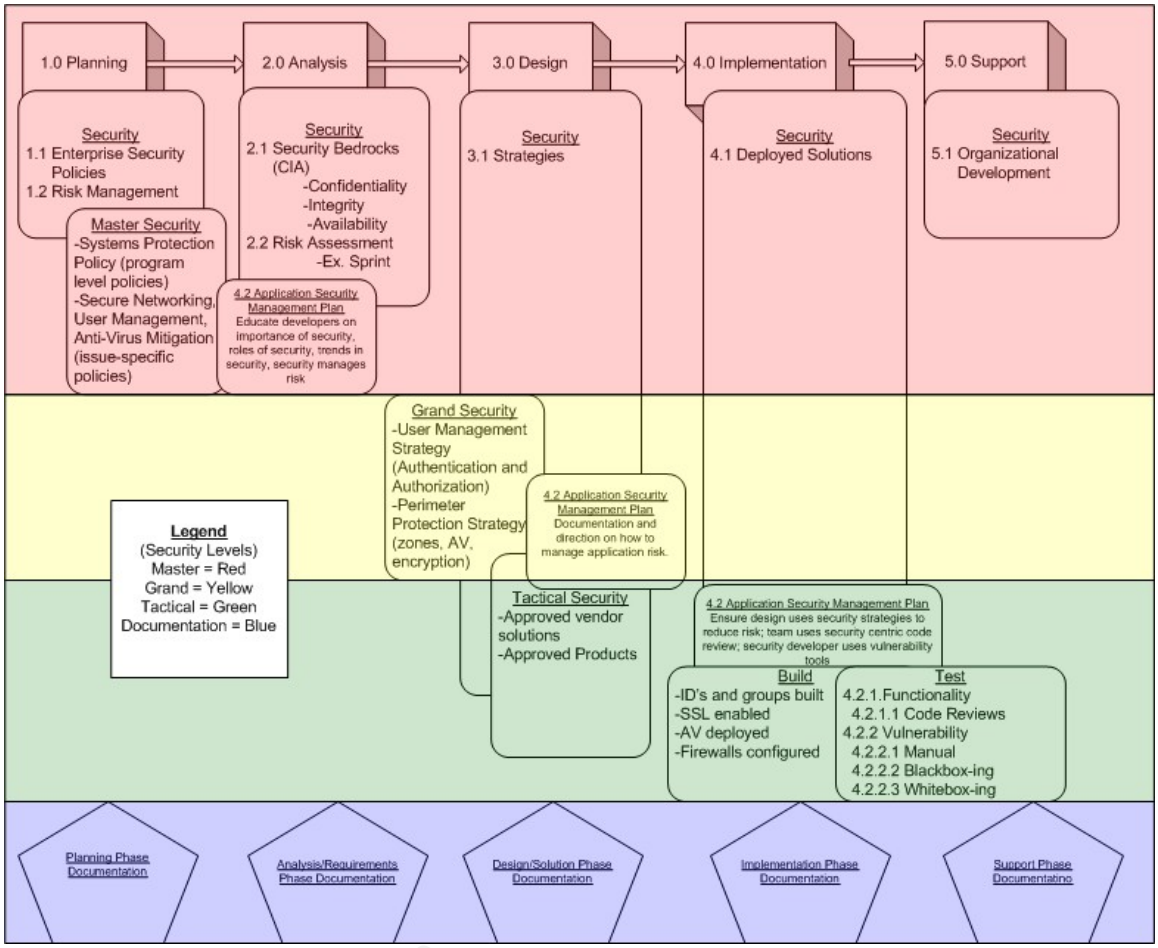
[19] Secure Software Development and Code Analysis Tools. 2002. http://www.sans.org/rr/whitepapers/securecode/389.php

10.0 Diagram

With the Support phase activities incorporated into the SDLC, the reader has a complete view of how the differences between program and issue-specific policies, CIA (confidentiality, integrity, and availability) and Risk Assessment, level of Security Strategies, levels of an Application Security Management Plan and how to manage vulnerability assessments in application testing. This complete picture is reflected on the following page in diagram 11.0.



11.0 Diagram



6.0 Summary

In the beginning, a very simple linear diagram was shown that provided a visual image as to the form of the SDLC (see diagram 1.0). Then the Planning phase, Analysis Phase, and the Design Phase were discussed. In light, the enterprise level security policies and security strategies were mapped against their SDLC phase and in accordance with conceptual, logical, and physical level (see diagram 2.0). Afterwards, the Implementation Phase and the Support Phase were described. Then in the preceding diagram (diagram 11.0), the entire lifecycle is shown. This diagram includes the build and test, physical-layer, activities of the Implementation phase, the Support Phase, and the documentation related activities in each phase. In addition, the diagram has incorporated placeholders to represent how the Application Security Management Plan (ASMP) would fit into the SDLC. In the end, this composite demonstrates a repeatable process where enterprise security policies allow business requirements to be met through the execution of appropriate security strategies and solutions.

7.0 References (http://www.nwmissouri.edu/library/citing/mla.htm#authors)

1. Whitten, Jeffrey L., and Lonnie D. Bentley. Systems Analysis and Design 4th Edition. Indianapolis: Mcgraw-Hill. 1998.

2. *Sans Defense-In-Depth 1.2.* 73. Sans Security Essentials. 2004.

3. *Sans Defense-In-Depth 1.2.* 74. Sans Security Essentials. 2004.

4. *Sans Defense-In-Depth 1.2.* 123-124. Sans Security Essentials. 2004.

5. Krause, M., and Harold F. Tipton. Handbook of Information Security Management: Access Control. n.d. Auerbach Publications. 6 Dec. 2004. http://www.cccure.org/Documents/HISM/019-021.html.

6. Krause, M., and Harold F. Tipton. Handbook of Information Security Management: Access Control. n.d. Auerbach Publications. 6 Dec. 2004. http://www.cccure.org/Documents/HISM/021-023.html.

7. Visintine, Visintine. An Introduction to Information Risk Assessment. 8 Aug. 2003. Sans Website. 6 Dec. 2004. http://www.sans.org/rr/whitepapers/auditing/1204.php.

8. Sans Top 20 vulnerabilities. n.d. Sans Website. 6 Dec. 2004. http://www.sans.org/top20/.

9. OWASP Top Ten Project. n.d. OWASP website. 6 Dec. 2004. http://www.owasp.org/documentation/topten.html.

10. *Sans Defense-In-Depth 1.2.* 75-76. Sans Security Essentials. 2004.

11. Application Security Best Practices at Microsoft. 1 Jan. 2003. Microsoft Website. 6 Dec. 2004. http://www.microsoft.com/technet/itsolutions/msit/security/appsecbp.mspx#EDAA.

12. Kapil, Raina. Trends in Web Application Security. 27 Oct. 2004. Security Focus Website. 6 Dec. 2004. http://www.securityfocus.com/infocus/1809.

13. OWASP Guide to Building Secure Web Applications and Web Services, The Open Web Application Security Project. n.d. OWASP Website. 6 Dec. 2004. http://umn.dl.sourceforge.net/sourceforge/owasp/OWASPGuideV1.1.pdf

14. Splain, Steven. Testing Web Security. Indianapolis: Wiley. 2002.

15. Splain, Steven. Testing Web Security. Indianapolis: Wiley. 2002.

16. Melbourne, Judy. Penetration Testing for Web Applications. 16 June 2003. Security Focus Website. 6 Dec 2004. http://www.securityfocus.com/infocus/1704.

17. Splain, Steven. Testing Web Security. Indianapolis: Wiley. 2002.

18. Spidynamics Application Security: White Paper. n.d. Spidynamic Website. 6. Dec. 2004. http://www.spidynamics.com/support/whitepapers/index.html.

19. La, Thien. Secure Software Development and Code Analysis Tools. 30 Sept. 2002. Sans Website. 6.Dec. 2004. http://www.sans.org/rr/whitepapers/securecode/389.php.