Global Information Assurance Certification Paper

Secure Application Methodology for Business Applications

Clint Tarpley

GIAC Security Essentials Certification Practical Assignment Version 1.0

ABSTRACT

What is security? When I look up the word 'security' in a dictionary, it tells me security is freedom from risk, danger, doubt, anxiety, or fear[1]. Can a computing network be secure? Well, what does it mean to be 'secure'? Again I go back to the dictionary and to secure something is to make free from danger, attack, or risk of loss and being intercepted or listened to by unauthorized persons, reliable, dependable[2]. Can a computing network be free from danger, risk; be reliable, dependable; and provide expected operability? Maybe… however it is the job of a security analyst to provide the appropriate level of confidentiality, integrity, and availability on a computing network in order to satisfy business need.

A secure computing network is one that reduces risk and allows the business to maintain proper levels of interoperability. Aspects of a secure computing network include: user education, policies, operations, procedures, intrusion detection, response, and auditing, as well as the physical protection of hardware, data, and users. These are just a start to the layout of a secure computing network.

Due to the vast interoperability of systems on a network it is safe to say that a network is only as secure, as it's weakest link. The weak link that I will address in this paper is applications and the development of business applications residing on a secured computing network. A security-minded application development lifecycle will help minimize risk and keep an environment secure.

INTRODUCTION

In order for security to be effectively and timely applied to an application, security policies, procedures, and constraints must be considered throughout the application development lifecycle. Too often security becomes a concern at the end of a development lifecycle. This slows down deployment and creates a bad reputation for security processes and policies. Also applications that have not focused on security may forget to include security best practices making an entire computing network vulnerable to attack.

An example of a security best practice is input validation. When input is entered by an end-user an application should make sure it is the appropriate length, type, format, and range. Doing input validation is a common security best practice forgotten by many developers. Without input validation an application becomes vulnerable to many attacks. It only takes one successful attack and an entire network could be compromised. Prior to deployment, applications should comply with common policies or document why they have not complied, so the application can be monitored when alternative solutions become available.

Accidents happen; new vulnerabilities are discovered daily; business areas implement applications with known vulnerabilities. This makes the job of a security analyst very difficult and is why a secure computing environment cannot be without migrating

controls and processes minimizing the risk and exposure of an attack. Vulnerabilities and attacks can cause the loss or disclosure of business assets. Multiple layers of defense help to protect business assets and decrease the threat of financial loss, compromised trade secrets, damaged reputations, and decreased customer confidence.

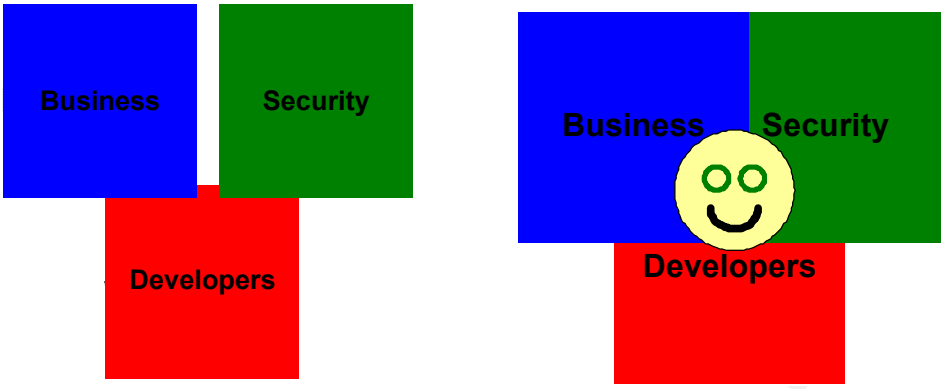
Government agencies are strong advocates for 'defense in depth' because it provides an overlapping protection mechanism against threats[3]. The nature of a computing network is very interactive and contains multiple systems that must all be secured. By overlapping protection mechanisms, a system is protected from the failure of one protection mechanism. Overlapping protection ensures that a system is protected even if one layer of defense has been penetrated. In order to implement 'defense in depth,' it is very important to address people, technology, and operations through user training and awareness. It is also very important for administrators and users of a defense in depth system to be able to identify when a breach has occurred. It is also important to fix the vulnerability in an efficient manner to reduce the exposure of more than one layer of defense.

A defense in depth system helps to limit risk of system interoperability and information sharing. Applications that provide interoperability and information sharing are targets for most attacks because they require elevated authority to perform work. Security must be involved in all stages of an application lifecycle. They must be involved in order to provide the appropriate level of confidentiality, integrity, and availability of a network where the applications reside. In turn, security helps satisfy business needs and provides the appropriate levels of protection for business assets by being involved in the application development life cycle.

BUSINESS ORIENTED APPLICATION DEVELOPMENT

Every company and every developer will approach application development differently. The goal of an organization should be to adopt a common development methodology in order to form consistency and better define security related events in an application development lifecycle. Consistency in application development allows for identical policies to be applied to applications, security minded developers, and business analysts focused on protecting business assets. Security and development cannot be considered separately or be considered as separate responsibilities. Developers and business analysts must understand each other and have the same goals. Security analysts and business analysts must define business assets and access to those assets. All three areas of knowledge must work together in order to effectively define a way to provide and protect business assets (See figure 1).

Figure 1: The best solution is achieved by sharing knowledge.



Requirements Gathering

Requirement gathering is typically the first step in application development and is an ideal location to inject security concepts into the heads of business partners and affiliated developers. At the beginning of a project, requirements are typically based on high-level business needs, which are recorded in sentence format and presented to a sponsoring area. When a sponsoring area approves the scope of a project, business needs are broken down into smaller units of work commonly referred to as business activities. A business activity is the first step in defining business assets that will be protected during application development.

After the first steps are taken and business activities are defined, actors should be specified for each activity. An actor is an individual or a collection of people that can take action on a business activity. The definition of actors is important because it will be used later in testing and deployment. It is important to remember when defining activities and actors, try to avoid defining every special case in an activity or abnormal actors for the activity. There will be plenty of time to map out special requirements later in design. Focus should remain on business functionality. By avoiding details during this activity, business models will be easier to understand and depict early signs of business assets and application roles.

The next step of requirements gathering is to further divide each business activity into use cases. Use cases are used to take the conceptual process of an activity and map the activity into smaller functional units of work. A use case will cover: expected business flow for each unit of work, actors that can take action, and possible exceptions during processing. At this phase of requirements, special cases around individual user access and administration should be documented. This phase is the first opportunity for a developer to see logical sequences of events making up a business activity. Along with understanding business activities, developers need to pay attention to exceptions and exits documented in the use cases. If an expected exception is not handled properly, the application may open up a vulnerability that could be exploited (e.g. exception exploit). Use cases documented in this phase of development should remain static and be used to develop test plans for the application later in development.

At this phase of application development, business functionality should be thoroughly designed. The next step prior to a developer beginning to write code is to understand and document security requirements around business activities and assets. Security requirement gathering should begin by considering the six foundational elements of security: authentication, authorization, event logging, confidentiality, integrity, and availability[4].

- Authentication is the process of uniquely identifying a user in an application.
- Authorization is the process of governing resources and actions that the authenticated user has permission to access.
- Event logging, or auditing, helps to guarantee non-repudiation of an activity.
- Confidentiality is keeping private or sensitive data protected from unauthorized users.
- Integrity guarantees that protected data has not been accidentally or deliberately modified.
- Availability is the process of making sure an application is available for use by all legitimate users.

These six elements make up the foundation of a secure application and each must be considered during requirement gathering in order to design a secure application.

The tricky part of defining security requirements is taking the six elements of a secure application and specifying constraints around business assets that must be fulfilled by the application. The first step of this process is specifying the data the application will be interacting with, based on activities and use cases previously defined. Once data is identified, it will need to be categorized into logical groupings. The logical groupings will be used to better understand what controls need to be applied during a transaction accessing this data. Government agencies are a prime example of businesses that base security controls of an application extensively on data classification[5]. The type of data being stored, the usage of the data, confidentiality, and data storage techniques should all be considered when classifying business data.

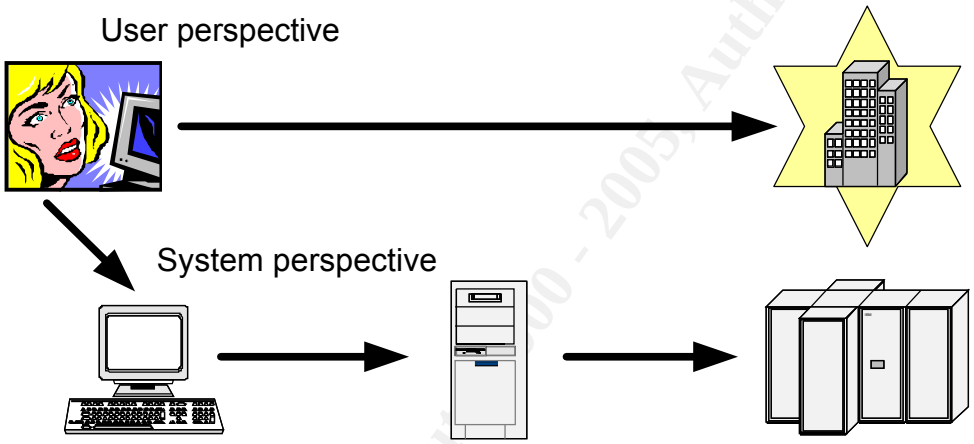
Risk Assessment

The next step of specifying what controls should be used to protect an application is to determine risk or threat of an attack. Threats and risks of an application are based on many different variables and cannot be complete without a proposed architecture to which the application will be deployed. Different architectures pose a greater threat than others based on known vulnerabilities of software, network services, and operating systems. System networks, data storage, server hardening, running services, and application configurations are just a few things to consider when defining risk for an architecture. When evaluating risk, not only should elements of risk be considered, but devices that eliminate risk on the proposed architecture should also be evaluated. (e. g. firewalls, routers, cryptographic services, authentication services, and authorization services.) For any measure of risk the proposed architecture, security vulnerabilities of the architecture, and security utilities available on the architecture must be considered. Once risk and protective controls have been identified, approving sponsors of the application should sign off on the risk assessment. This extra step is

important and will guarantee responsible parties of business assets are aware and accept the risk induced by making assets available through this application.

Data classification and risk assessment will cover confidentiality, availability, and integrity, but sometimes lacks detailed discussions around authentication and authorization. In order to define thorough authentication and authorization requirements; two approaches must be considered: system and user. When entering into an application, the server where the application resides will attempt to establish a trust relationship with the user. Once the user has been identified and trusted, the system will attempt to authorization the user to perform the requested action on the server. These two processes are considered system authentication and authorization. System authentication and authorization are typically defined through application configuration settings and Access Control Lists (ACL).

Figure 2 - User verse System controls



User-based Authentication and Authorization

User-level authentication and authorization happen last in the TCP-IP stack within the application layer. User-level authentication and authorization occur within the application and require a developer to write code in order to uniquely identify a user or govern access to business resources. These security requirements are typically left out of requirement gathering because they are very granular and depend on physical implementations of the application. User-level controls are most commonly discovered during development when system controls alone are not adequate in providing necessary levels of confidentiality and integrity. If low-level authorization requirements are gathered early during design, developers will have an opportunity to evaluate system-level security controls during development. Developers will also prepare the application for additional security constraints if necessary. However, developers and analysts typically do not have the foresight to see shortcomings of system-level authorization. Granular authorization requirements are typically discovered late in

development.

When developers identify granular authorization requirements late in development, they will do one of two things: scramble to security analysts for help which will probably cause modifications to code, or get creative with a unique solution to meet desired requirements. Quick-fix authorization solutions may help short-term needs of an application, but developers will continue to experience similar problems in sequential applications. Developers should focus on normalizing user-level controls to increase reusability across applications. In doing so, developers will decrease time to market and decrease administration burdens that are influenced by multiple hereditary application based user-level controls.

BUILDING USER-BASED SERVICES

When an analyst defines security architecture they will typically think of physical aspects of an application: server, router, firewall, file, packaged code, and an ACL. Defining user-level controls takes an abstract approach to security. Resources and actions in a business application are no longer physical objects but represent a logical unit of work. It is security-oriented in a way that the business can understand. The first step of user-level controls is defining business activities and use cases that were mentioned earlier in this document. The next step is to define application resources. Application resources are abstract business assets classified as sensitive and require an additional layer of defense. A user-level resource may be an auto policy, a report, or a payment option. It represents a business artifact the business understands and can relate to for ease of administration.

Too often application security becomes oriented around who can click a button on a screen. Buttons are not business assets if compromised and cause financial damage. The action and resource executed behind the button are true business assets. Business security requirements should reflect the business assets that need protection. This will allow the business to identify functionality of the application and provide proper administration of the business asset without making an assumption on the functionality of a button. The combination of an action and a resource may be referred to as a privilege, operation, or a claim.

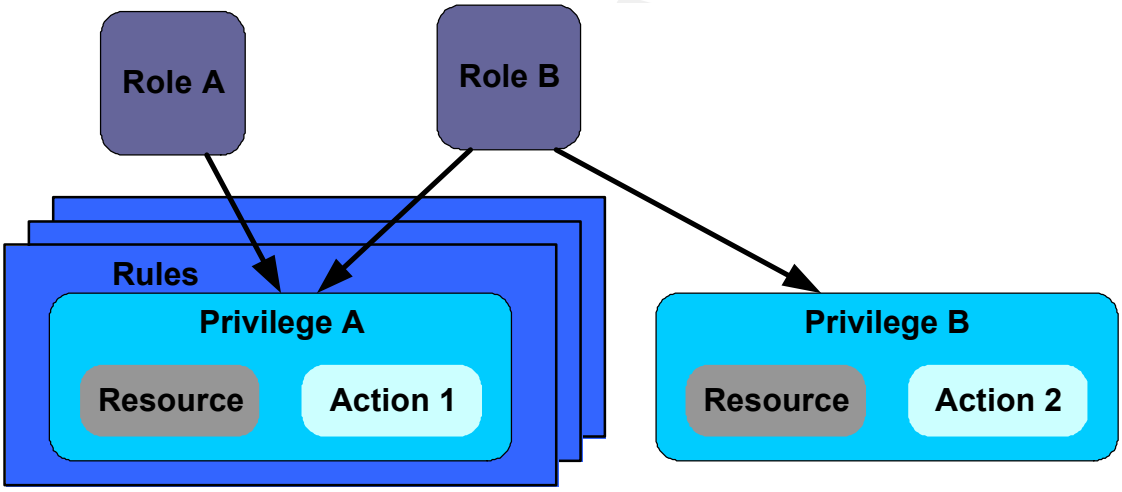
Defining Roles and Privileges

During the process of defining privileges for an application, business analysts should start to categorize privileges into groups based on functionality. Each logical group of functionality will relate to a low-level business activity. A logical group of privileges turn into work duties. Work is preformed by a collection of people, or an application role. Application roles are derived from business activities and should not be considered a collection of users, but a collection of privileges that a group of individuals need in order to perform work. The process of defining application roles can be difficult for analysts. Historically, roles have been work groups, teams, or individuals requesting access to resources. This model begins to break down when business analysts

accept ownership of application security and the assets used by the application. If the application security model is not a representation of the business it will be difficult to be appropriately maintained. Business-oriented security models help to reduce the risk of human error when administering security, because security becomes a better representation of the mental model of an analyst[6].

As mentioned earlier, an application role is a collection of privileges. A privilege is a specific action on a resource. User-level controls are designed with the business in mind so an abstract resource is a logical business artifact or a business event that requires special authorization. Special authorizations will require dynamic security control. As an example, the privilege 'withdrawal cash' will need to evaluate the current actor, current account balance, and the requested amount in order to grant or deny access to the privilege 'withdrawal cash'. Special run-time constraints applied to a privilege for granular security are called rules. Rules may contain environment constraints, current time and day, user attributes, transactional attributes, or a combination of all four. The definition of a rule should remain extensible and can be applied to a privilege that is contained by one-to-many roles.

Figure 3 - User-level security concepts



Designing a user-based authorization service can be difficult due to the unlimited number of variables required to make granular authorization decisions. A user-based authorization service should be extensible, scalable, and flexible. By making the service extensible, the service is better prepared for multiple underlying implementations. Extensibility will also help to eliminate the need of a developer to make multiple coding changes in applications when new security products become available and used. Scalability is important in order to handle multiple authorization requests in an efficient manner. Developers have become use to system-based authorization requests and will expect transaction durations not to be greatly impacted by using user-based authorization services. Reusable services, e. g. a user-based authorization service, should not change when an infrastructure change occurs. This is a developer pet peeve because it requires coding changes on behalf of every application using the service. Flexibility of the authorization service will help to

establish consistent interfaces independent of an implementation. A consistent interface definition is the key to a strong reusable service implementation. By decoupling the interface from an implementation specialized platform, dependent security controls can be used. This will decrease cost of supporting multiple controls without changing interface definitions on disparate platforms.

Administration of User-Based Services

Another element to consider when designing a user-based authorization service is administration. Administration interfaces must be easy to use, available for alternative user interaction designs, and scriptable for batch processing. By allowing multiple administrative options the business is given flexibility in the manner they wish to control business assets. It is very important to be consistent and secure when providing multiple administration options. A consistent implementation of the interfaces will guarantee events interacting with authorization data have been audited and protected as anticipated. Direct access to authorization data should not be allowed. If direct access is allowed, events acting on security data will not be captured consistently, and integrity of the system will be lost.

Customized administrative interfaces allow for independent representation of security administration based on business context. Business-oriented administration will help to reduce human error because security becomes a better representation of the mental model of the analyst. By reducing risk of human error, the security of the user-based authorization system is stronger.

System Integrated User-Based Services

The two forms of authentication and authorization mentioned in this document are system and user-based. Security administration is always considered a burden. Multiple security administration is a headache. Are both system and user-based services required, or is one or the other sufficient?

Both system and user-based services should be used to protect an application from attack. System services protect the front door of an application. Without protecting the front door, all users of the application are allowed in, and unauthorized individuals need to be filtered at runtime. Performance of the application will be impacted when user-based controls are not as fast as integrated system controls. Also, by protecting the front door of an application, unauthorized users are rejected prior to determining what functionality can be provided by the application. After access to an application is successful the application will execute under an application context instead of a user context. The context switch occurs within the application to increase data protection, performance, and the protection of sequential services used by the application. By securing data and services to application identities, thousands of user identities will not need to be maintained at the data layer. Authorizing user identities at the data layer of an application would increase exposure. The more identities authorized to a service, the more opportunity to compromise an identity. In particular only one or two support

personal would know the identity and password for an application, verses 100,000 users knowing their own identity and password. The risk of exposure greatly increases with user authorization to data services so application context switching should occur in an application.

However, application context switching is not always a good thing because applications will have to authenticate and authorize users within the application during a transaction. User-based services help applications maintain user credentials so they can be authorized to business functionality at run-time. System controls can handle static authorization decisions; but run-time decisions have to be made programmatically to user-based services. User-based services are best for granular authorization requests and help to prevent developers from hard coding security rules within code.

Hard coded security rules make administration difficult. When an exposure has been found in an application that is based from hard coded security rules, changes to the rule must be made in a fast and efficient manner with minimal impact to the application. The more time it takes to fix a known vulnerability, the more risk an application encounters. Modifications to fix vulnerabilities discovered in application code are not easy because of regression testing, bug tracking, and change management procedures. If authorization decisions are externalized from the business application, the external source can be modified without affecting the code using the security privilege. This will minimize the cost associated with fixing security-defects within the application.

Every application is different. Some applications require granular security others may not. If an application has a need for user-based authorization decisions, system authorizations cannot be ignored or forgotten. Administration of two authorization systems is very costly for an organization. Provisioning solutions should be created for user-based services to map fine-grain user policies into coarse system authorizations. By provisioning user-based services administration remains business oriented and unified in one central repository.

CONCLUSION

The job of a developer is to provide services that enable the availability of business assets to consumers. Levels of risk increase based on business needs to make assets more available to consumers. As business assets become more available levels of risk increase. It is the job of a security professional to provide the ability for a developer to minimize risk associated with making business assets available and to achieve a level of risk that has been deemed acceptable by business asset owners. Preparing detail business activities and designing application logic around security constraints help to achieve this goal.

Applications can be secured by using an ACL, an application server run-time, or by making programmatic calls to a security service. However, without a proper design

© SANS Institute 2000 - 2005 Author retains full rights.

and detailed security requirements, an application will not fulfill the needs of a business owner. The concept of security should be centralized. If not, it can lead to big administration headaches maintaining and debugging multiple security models. Common security services can be used to alleviate some of the pain caused by multiple security administration products. However, common services must be protected the same as a sensitive application. It should contain event logging for auditing and an administration module that can be used to protect the integrity of the data in a policy store. A common component should also provide integrity of a user session so that a user can be evaluated throughout the lifetime of a transaction without increasing risk on a system. User-based controls are not system-based controls, and the combination of both, provide defense in depth.

Processor speed, bus size, hard drives, and RAM continue to increase. It is expected that applications developed on these platforms be enhanced to use these expanding capabilities. With the explosion of application development, security for applications is difficult to manage. Security has to be expressed in an abstract sense to protect applications from changes induced by multiple security implementations. An abstract security model will also allow for the proper protection of business assets by business owners.

References:

Deputy Security of Defense. "Department of Defense Chief Information Officer Guidance and Policy Memorandum No. 6-8510." Guidance and Policy for Department of Defense Global Information Grid Information Assurance. June 16, 2000. p.3. http://www.defenselink.mil/nii/org/cio/doc/gigia061600.pdf.

Greenemeier, Larry. "Intel Unveils More Details About Next-Generation Processor." When the new processor, code-named Montecito, debuts in 2005, it will include 24 Mbytes of Level 3 cache memory, as well as two cores, each with multithreading capabilities. Nov. 14, 2003. http://www.informationweek.com/story/showArticle.jhtml?articleID=16100623.

Howard, Michael and David LeBlanc. Writing Secure Code 2nd Edition. Microsoft Press, 2003, p.4, 5, 51-68.

King, Christopher. "Intranet Application Security Checklist." 1997. http://www.infoseceng.com/intra.htm.

Meier, J.D. Improving Web Application Security, Threats and Countermeasures. V1.0. J.D. Meier, Alex Mackman, Srinath Vasireddy, Michael Dunner, Ray Escamilla, and Anandha Murukan. Microsoft Corp., 2003, p.9, 10, 689-694.

Pavlina, Steve. "Zero-Defect Software Development." 1999. http://www.dexterity.com/articles/zero-defect-software-development.htm.

Schneider, Bruce. Secrets and Lies, Digital Security in a Networked World. John Wiley & Sons, Inc., 2000, p.59-79, 367-388.

Smith, S.W. "Humans in the Loop, Human-Computer Interaction and Security". Security & Privacy. May/June 2003. Vol.1, No.3. p.75-79.

Yee, Ka-Ping. "User Interaction Design for Secure System." http://www.sims.berkeley.edu/~ping/sid/uidss.pdf

The American Heritage, Dictionary of the English Language, Fourth Edition. Houghton Mifflin Co., 2000, http://dictionary.reference.com/search?q=security.

The American Heritage, Dictionary of the English Language, Fourth Edition. Houghton Mifflin Co., 2000, http://dictionary.reference.com/search?q=secure

[1] http://dictionary.reference.com/search?q=security
[2] http://dictionary.reference.com/search?q=secure
[3] Deputy Security of Defense, p.3.
[4] Howard, p. 4-5.
[5] Schneider, p. 62-63.
[6] Smith, p. 75-79.