Global Information Assurance Certification Paper

Application Security Architecture

Hari Simhadri

GSEC Practical Requirements(v1.4b) (August 2002)

Introduction

Software applications are developed with minimal security in mind. This happens most of the time because of various reasons:

- application analysts/architects mostly concentrate on the problem domain
- designers/developers concentrate on implementation details
- development teams lack security expertise
- It is difficult to justify a fulltime security professional on the team.

When deployment dates arrive, everyone realizes the need for protecting the application by applying the security layer to it. The notion of providing a logon screen to protect the application is immature by the current security standards, since:

- threats have multiplied and grown far more sophisticated, ranging from cyber-terrorists to industrial espionage
- application architectures have moved from centralized mainframes to distributed technologies
- Security technology has made a lot of progress

Enterprises which use software applications to run their businesses effectively have progressed from using mainframe technology all the way to web-based technology. Applications are becoming more decentralized and so is their administration. They are also moving to the Internet while making data and services available dynamically. At the same time, application fraud and attacks (insider and outsider) are increasing at an alarming rate. Most corporations assume that these threats are from outsiders and can be prevented by using network security products like firewalls, routers and intrusion prevention systems. These products are part of a comprehensive security strategy but do not focus on solving application security issues. Firewalls, for example, usually have port 80 open for use by web-based applications – the very port that a large number of application and system vulnerabilities take advantage of.

There is a definite need within enterprises to define architecture for application security. The architecture should work as a guideline for developing security in applications. Overall, the application security architecture should help the organization to:

- apply the security solutions to any application, no matter what technology it uses
- have proper security controls in place for an application
- protect the application from threats
- ease the burden of performing security administration
- help security assurance by providing proper data for incident handling (accountability)
- easily adopt to changing security infrastructures

This paper will attempt to define an application security architecture that will help an organization to develop secure applications.

Architecture talk

The term 'architecture' has various meanings. Traditionally, it means 'the art or science of building' or 'the method and style of building'.[1] When the meaning is extended to software applications it means the manner in which the components of a software system are organized and integrated. There is always some confusion between the definitions of architecture and design. A key distinction is that architecture is the description of the structure that can be applied to multiple situations while design is specific to a certain situation. Benefits of creating an architecture model include:

- providing guidelines for secure design and development
- defining a structure to provide secure layering functionality (reusability),
- Identifying a path towards a security goal
- Generating better ROI

Application software architecture has many definitions. One of my favorite definitions, by Garlan and Perry, is 'The structure of the components of a program/system, their interrelationships, and principles and guidelines governing their design and evolution over time'.[2]

Taking the above into consideration, we can define 'application security architecture' as the manner in which the security components of an application software system need to be constructed, so that they are:

- easy to use
- flexible to change
- reusable
- extendable
- interoperable

Application software trends

Application Software can be defined as software that constitutes any type of program that is tailored to satisfy real-world needs and requirements or is a program or a collection of program modules to deliver some business functionality. Application software may be developed in-house by the organization that uses it, custom-made by another consulting company, or purchased as an off-the-shelf package. For the latter two methods, it is especially important to:

- have all the technical documentation
- have access to source code (if possible)
- ensure that the provider uses industry standards
- ensure that the provider uses open API's for plugging in functionality

During the mainframe days or early client-server days, applications were often developed from scratch by one team. Applications have changed considerably from then with the emergence of distributed objects (reusability), n-tier architectures (layering) and enterprise application integration (interoperability).

Web based applications are quickly becoming a trend. A set of screens or web pages that a user navigates through may feel like an application to the user, but the functionality behind them may come from different systems, platforms and languages with purchased components and software hosted by business partners. Peer-to-Peer application architecture is gaining momentum with web services.

Business needs drive technology. During mainframe days, systems were required to automate business processes and reduce costs. Client/server technology introduced more functionality at the user level and helped increase productivity. The ability to use the Internet has changed the way businesses think. New breeds of applications such as customer relationship management, direct sales, maintenance/repair/operations, vendor managed inventory, and order tracking improved business processes. These applications extend beyond the perimeter of the corporation, requiring even more open ports in the firewall. Overall, the scope of applications has grown from residing on a single monolithic machine to deployment across multiple machines distributed in intranet, extranet and Internet environments. Applications are evolving from a client/server model to a network centric model.

Apart from the application architecture, two other key changes are types of users and communication means. Traditionally, applications had been used only by employees. Now, users of an application can be classified into various types:

- By Organization – employees, contractors, customers, partners
- By network – internal, external

There can be a mix of this classification, where an employee can be an external user, if he connects through a VPN. The same way a contractor can be an internal user if he is on the premises of the organization.

Various mechanisms are utilized for a user to communicate with an application as well as application layers to communicate with each other. Operating system inter-process communication mechanisms, like shared memory and semaphores to network protocol based means like TCP/IP, sockets, remote procedure calls, and distributed objects are being used.

Application Security Architecture

Everything in Information Security should start with a policy and so should application security. The security policy needs to be thoroughly applied to applications. Traditionally, it has always been applied to the network to protect the resources. Since applications are resources, there is more risk at this level and more need to protect it thoroughly. Two key components required to design, develop and deploy secure applications are:

- Application security development life cycle and guidelines
- Security infrastructure with interoperable components

Application security development life cycle and guidelines

Applications are designed and developed using various methodologies such as structured design methodology, object-oriented and extreme programming. Since it is difficult to develop a generic life cycle that fits all the needs, we need to develop a lifecycle with guidelines that might enable different methodologies to incorporate it in their flow.

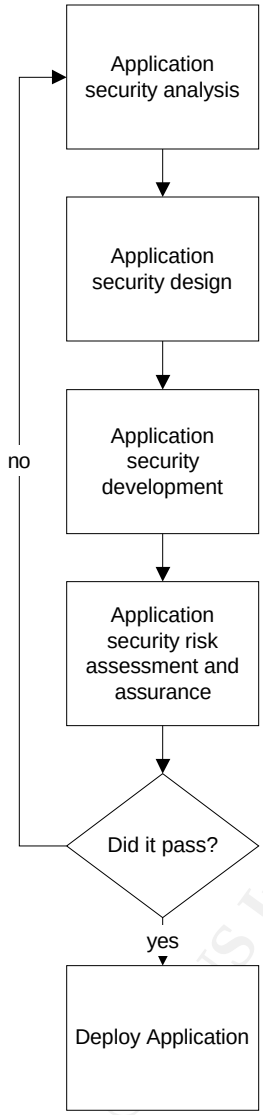

Fig 1: Application security development lifecycle

Application security analysis

An application, for security analysis purposes, can be defined as a “resource (data and business logic) repository” available to a “user” over a “communication channel”. The key security concepts that revolve around a user are identification, authentication and authorization. Often identification and authentication are used

interchangeably, but they are separate and distinct components. Identification is 'an act or process that presents an identifier to a system so that the system can recognize a system entity and distinguish it from other entities'.[3] Authentication is 'the process of verifying an identity claimed by or for a system entity'.[4] Applications typically implement these two aspects of security through a USERID and a password. Traditionally, users of an application within an enterprise were always employees. The Human Resource department would register an employee and assign an identity (employee id). Application developers would either use the HR database or load a sub-list of the employees requiring access into the application database as users. Employees would access the application by using their employee id and a password. Application trends suggest the range of users of an application can include employees, contractors, customers and partners. There needs to be a registration process to assign a unique identity to every user. This process needs to be defined very carefully, so that the application does not assign duplicate identifiers to the same user nor allow any unknown or unidentifiable user to register with the system. If there is a need for this process across multiple applications it is better to implement this functionality in a centralized fashion with common interfaces that can be utilized by multiple applications.

Authentication is used determine the legitimacy of a user who wants to access the application. Different levels of authentication can be used to protect the application based on the risk associated with the application. Sensitivity of the data, application functionality, application architecture (the more distributed it is, the more the risk), user base could be factors determining the risk. The three factors that can be used to authenticate a user are:

- something you know(password),
- something you are(a USERID or biometric) and
- something you have (badge, smartcard).

The cost of implementing a solution varies widely with the different authentication technology that is used. In some cases, where there is a need for strong authentication, a combination of these factors should be used.

Once users have validated themselves as legitimate, the application should be able to verify what resources they should be given access to. The security component for access control is **authorization**. 'An authorization is a right or a permission that is granted to a system entity to access a system resource'.[5] Typically, an application is developed to perform certain business function and users of that application might need to perform part of that function. Every user may not need access to the all the application resources including business logic and/or data. The resources need to be grouped based on the functional roles different users have. This way of assigning privileges to a user is called the roles-based access control (RBAC) method.

Once a user is authenticated and is authorized to perform a business function (transaction), there may be data/control transfer between different processes on different systems. A typical application is split up into multiple module/objects

which perform different logical operations to accomplish a complete task. For example, a mainframe application is monolithic in structure and is split into various modules which could run in the same process space to complete a task. An n-tier application could have multiple processes running on several machines to perform a task using inter-process communication mechanisms and/or communication protocols. In these scenarios, where several communication channels might be used, various network and system threats, including eavesdropping, could jeopardize this communication. If the data is corrupted or compromised, there will be integrity and confidentiality issues. In addition to protecting data in transit, protection of static data is required to provide integrity and confidentiality. For example, if the configuration files containing the connection and initialization attributes of an application were compromised, integrity and confidentiality of the application resources could be severely compromised.

Integrity is 'the property that data has not been changed, destroyed, or lost in an unauthorized or accidental manner'.[6] Incorporating integrity into an application will inform the recipient of data modification by unauthorized users during storage or transmittal. Various hash methods such as SHA-1 or MD5 can provide the integrity component.

Confidentiality is 'the property that information is not made available or disclosed to unauthorized individuals, entities, or processes [i.e., to any unauthorized system entity]'.[7] Confidentiality keeps the data secret from all but those who are authorized to see it. Various cryptographic methods can be used to provide confidentiality component. 'Cryptography is the art or science encompassing the principles and methods of transforming an intelligible message into one that is unintelligible and then retransforming that message back to its original form'. [8]

Encrypt (secret key) Decrypt (secret key)
plain text ----------→ cipher text ----------→ plain text

Plain text is the original intelligible message and Cipher text is the unintelligible message. Encryption transforms plain text to cipher text using a secret key and decryption transforms cipher text back to plain text using the secret key. The cryptography method using the same secret key to encrypt and decrypt the messages is called symmetric and the one using different keys is called asymmetric. Application data can be kept confidential using cryptography, if the secret key can be secured.

Auditing is another security feature that needs to be implemented in the application. Since no application can ever be completely invulnerable to threats, auditing must be in-place to provide data needed for incident response. The auditing feature should provide a trail of which user did what and when at any given point of the application lifetime.

The final component, **Non-repudiation** provides proof of existence of message. 'Non-repudiation with proof of origin provides the recipient of data with evidence

that proves the origin of the data, and thus protects the recipient against an attempt by the originator to falsely deny sending the data'.[9] 'Non-repudiation with proof of receipt, provides the originator of data with evidence that proves the data was received as addressed, and thus protects the originator against an attempt by the recipient to falsely deny receiving the data'.[10] The goal of a non-repudiation component is to collect, maintain, make available irrefutable evidence. For this component to work, all the previous five components must be established:

- All parties must be identified and authenticated
- All parties must be authorized
- Integrity of the transaction content must be intact
- Transaction information needs to be confidential to authorized users
- All transactions must be completely audited

Application Security Design

During the design phase of an application (for off the shelf purchase, it would be evaluation phase), security needs to be involved with the application team to assist them in implementing proper security technology and/or processes. The key structural needs within an application to incorporate proper security controls are:

- **Single Access Point:** User entry into an application should be though a single point. Backdoors are often created to perform administrative functions. Backdoors should be avoided and application entry should be restricted to a single point for all types of users, including administrators. Identification and authentication components need to be performed at this point. Access points to multiple applications can be consolidated to a single point, often called a portal. Also, technologies such as single sign-on are gaining popularity. Single sign-on enables a single access point into multiple applications by authenticating once. This feature also saves the user from having to remember multiple passwords which can provide enhanced security by eliminating a need to write down passwords in unsecured places (for example, sticky notes under a keyboard or on a monitor).

- **Session:** Users should not have to authenticate multiple times while they are traversing an application. Their current interaction with the application needs to be maintained by the application. This session must be unique and separately maintained for every user. For example, if a user is away from the application for sometime, the session should help the application from determining the user and their current state with the application. Also, authentication details like last active time can help the application determine whether it needs to authenticate the user again per the authentication policy.

- **Roles:** Users have different needs in an application. Users should be given different privileges (read, change, add, delete) to various application resources within an application. If implemented to only allow individual, user-by-user authorization, maintaining privileges would be an administrative nightmare. Typically, users are grouped into a certain role and privileges to required application resources will be given to that role. Once that is done, all the users who perform that role are assigned that role instead of each privilege individually. Instead of changing a privilege for every user, the administrator will just have to change it for that role. Check points have to be established at every resource level where a privilege to a user needs to be defined. These check points should query the access control data to make a decision for the user to access the resource.

- **Secure Access Layer:** Applications use various mechanisms to communicate with the user or other applications. Depending on the classification of the data that is exchanged or control that is transferred during the communication, the access mechanisms and access layer need to be secured. Confidentiality and integrity components need to be used to secure the communication, as well as static data. For example, SSL can be enabled on a web server using certificates to provide authentication of the application to the user, confidentiality between the application and the end user and integrity of the information transferred.

- **Audit:** components need to be placed in an application for tracking the actions being performed on the application resources. The placement of these components should help tracing any application event. Proper backup and recovery procedures need to be implemented for the audit output using retention guidelines. Also, the audit output should be properly formatted to help searches or statistic generation. The output should definitely contain the successful and unsuccessful attempts by a user accessing a resource at the single access point and all the checkpoints. Adding a timestamp, message code and message description will add value to the output.

- **Administration layer:** Easy to use administration functionality needs to be provided to the application administrators to maintain user identification attributes, authentication and authorization information. The administration functionality needs to be accessible to the administrators though the single entry point and proper authorization. Some features that would help application administrators are:
    - support for RBAC(creating roles and assigning them to users)
    - user administration(enable/disable users)
    - application scan facility to generate new access points when changes are made to existing applications

Application Security Development

Application security development should be given a lot of thought and care. An application security development guideline needs to be created, specifying various technologies and a coding style to eliminate vulnerabilities and help mitigate risk. Some of the application components that must be considered in developing the guideline are:

- A common vulnerability of application is buffer overflow. This threat can be mitigated by proper data type/bounds checks at the interface of any reusable component like a function/method or at any data input process.

- Authorization needs to be handled carefully. Just defining roles and assigning those to users will not completely secure authorization. Situations where multiple roles are assigned to the same user with different types of privilege on the same resource need to be considered.

- Application should always fail safely. It should never fail to an 'open' state.

- Error/exception handling mechanisms used should not display too many development details. Some coding scheme needs to be used which could be cross-referenced to an error description database.

- System configuration needs to be analyzed. Most of the systems come with all privileges open for the default. Never use the default configuration. Always use the least privilege model.

Application security risk assessment and assurance

Applications need to be assessed at the business level to ascertain the risk based on information compromise, unauthorized access and availability for determining the security level that needs to be assigned to them. After an application has been developed and functionally tested and before deploying it to production environment, we need to meticulously perform a security risk assessment and assurance test. This test will help ensure the total system is in compliance based on the security level assigned to it. These tests need to be mandated on all applications, newly developed or changing an existing application or a purchased product. Application risk checklists need to be developed to assure that proper security controls have been placed at the appropriate locations within the application. The checklist must be updated at a regular interval to accommodate newer technologies and threats. The checklist should contain all aspects of logical access for various security levels including:

- user identification (registration process)
- authentication (level, password strength, sign-on attempts, account lockout policies, helpdesk processes on unlocking, session tracking)
- authorization

- sensitive data handling (encryption, hashing)
- auditing features
- security administration

Apart from the application tests, any system software such as operating system, server software (web, application), and DBMS need to be assessed and patched to the latest security compliant level. Tools, products or processes need to be employed to standardize the methodology of security assessments.

Application contingency plans should be reviewed to make sure all the backup and recovery plans are up to date so that there is no disruption of service (availability).

Security infrastructure with interoperable components

Security needs to become an infrastructure service. The infrastructure needs to be interoperable with any application and be maintained by a team that can keep pace with the latest standards. Accomplishing the above and offering security as a centralized component can be a tedious and painful task. Eventually, the Enterprise will realize the fruits - cost savings and controlled environment. This centralized infrastructure should:

- provide Identity, authentication, authorization, confidentiality, integrity non-repudiation and audit components
- adhere to industry standards
- be easily manageable
- be scalable
- Provide the framework which needs to adapt to newer technologies with less effort (security is a race where the good guys always need to be in front of the bad guys).

To provide the above infrastructure, the security team needs to analyze the existing applications and define the requirements.

Enterprise application software(s) - security analysis

Security team needs to collect statistics on the various applications that are being used in the organization with an emphasis on identification, authentication and authorization. A sample template listing the data that should be collected could look like:

Characteristic	**Value**	**Example**
Application Name	EPay	Payroll Application
Number of users	300	
Application type	Client/server	Mainframe, Web based …

		…
Software Used	C, Oracle, PowerBuilder	Java, Sybase, HTML …
OS	Solaris/Windows 98	OS 390, VAX VMS
Identity source	HR database	Auto-registration process, IVR, Helpdesk
Identity used	Employee ID	SSN, Client certificate
Networks traversed by application data	Intranet	Internet, Extranet
Data Classification of data exposed by the application	Proprietary-restricted	Non-proprietary, Proprietary
User access	Custom Client	IE, Netscape Navigator, VT100, OS 390 terminal
…		

Defining these characteristics will help the security architecture team to:

- understand the functional, performance, cost and process requirements from a security perspective

- Develop solution(s) to support different types of applications with their security needs. Gaps should be filled by working with the application teams to provide custom solutions that can be reused for similar applications. For example the infrastructure may provide the security components in a certain language and the application might be using a different language. We might need to develop some integration software that would fill the gap.

Conclusion

Application security has always been a risk, but corporations have habitually ignored it, assuming firewalls will protect everything. With increases in the number of application attacks in spite of having layered perimeter security, corporations need to realize the threat and raise its priority. Here are the logical steps that will take us towards a secure application environment:

- A corporate security policy needs to be used and an application security specific document needs to be derived out of it.

- An application security lifecycle needs to be defined.

- Standards for systems and applications need to be established and the guidelines, processes and checklists need to be developed for supporting the application security lifecycle.

- Security teams need to work with application teams to incorporate this life cycle within the various development styles that are being used.

- Tools/products need to be acquired to perform risk assessment and assurance of the systems and applications that are being deployed.

- Applications need to be analyzed for vulnerabilities and a common, centralized security infrastructure needs to be established to accommodate their security needs.

- New applications need to use the security infrastructure from the start and older applications need to start migrating toward it.

- Update/upgrade the policy, guidelines, processes, checklists, infrastructure at regular intervals to accommodate needs and technology changes.

References

Christopher M. King, Curtis E. Dalton and T. Ertem Osmanoglu "Security Architecture: Design Deployment and Operations" Osborne/McGraw-Hill Copyright 2001

Jeff Forristal and Julie Traxler "Hack Proofing Your Web Applications" (Technical Edition) Syngress Publishing Copyright 2001

Ian Rathie "An Approach to Application Security" SANS January 30, 2002

www.owasp.org "Best Practices for Secure Development" http://www.owasp.org/whitepapers/best_prac_for_sec_dev4.pdf (1 JAN 2003)

Dan Blum "Securing the Virtual Enterprise Network" http://www.burtongroup.com (1 JAN 2003)

Garlan, David & Perry, Dewayne "Introduction to the Special Issue on Software Architecture. IEEE Transactions on Software Engineering 21, 4 (April 1995)." http://www.ieee.org/portal/index.jsp (1 JAN 2003)

Fred Maymir-Ducharme, P.C. Clements, Kurt Wallnau, Robert W. Krut, Jr. "The Unified Information Security (INFOSEC) Architecture" http://www.sei.cmu.edu/pub/documents/95.reports/pdf/tr015.95.pdf (1 JAN 2003)

Swanson, Guttman "Generally Accepted Principles and Practices for Securing Information Technology Systems." National Institute of Standards and

Technology. September 1996." http://csrc.nist.gov/publications/nistpubs/800-14/800-14.pdf (1JAN 2003)

Joseph Yoder& Jeffrey Barcalow "Architectural Patterns for Enabling Application Security" http://st-www.cs.uiuc.edu/users/hanmer/PLoP-97/Proceedings/yoder.pdf (1 JAN 2003)

RFC list Internet Security Glossary http://rfc-2828.rfc-list.org/rfc-2828-84.htm (1 JAN 2003)

Dictionary "Webster's Revised Unabridged Dictionary, © 1996, 1998 MICRA, Inc." http://www.dictionary.com (1 JAN 2003)

[1] http://dictionary.reference.com/search?q=architecture
[2] http://www.ieee.org/portal/index.jsp
[3] http://rfc-2828.rfc-list.org/rfc-2828-84.htm
[4] http://rfc-2828.rfc-list.org/rfc-2828-84.htm
[5] http://rfc-2828.rfc-list.org/rfc-2828-84.htm
[6] http://rfc-2828.rfc-list.org/rfc-2828-84.htm
[7] http://rfc-2828.rfc-list.org/rfc-2828-84.htm
[8] http://rfc-2828.rfc-list.org/rfc-2828-84.htm
[9] http://rfc-2828.rfc-list.org/rfc-2828-84.htm
[10] http://rfc-2828.rfc-list.org/rfc-2828-84.htm