Security Design Patterns

Part 1 v1.4

Sasha Romanosky

November 12, 2001

Table of Contents

1.0 Overview

1.1 Context

A comprehensive security strategy first requires a high level recognition of overall Security Principles. They are simple statements, generally prepared by a Chief Information Officer (or Chief Security Officer) that addresses general security concerns. E.g. monitor all activity, audit your practices, promote security awareness, etc.

Next, Security Policies are created. These are the realization of Security Principles. Prepared by security professionals, Security Policies are meant to address security issues when implementing business requirements. E.g. use out of band communication when responding to an incident alert, employ secure coding techniques, implement a central log server, etc.

Finally, Security Procedures are identified. This is an itemized, quantifiable list that identifies specific hardware, tools and tasks. E.g. disable telnet and ftp on all hosts – replace with ssh and scp, validate html form data on both client and server, change default application passwords, etc.

This essay is not meant to replace any of these documents, but to supplement all three. That is, once general policies are defined, security patterns can assist in identifying and formulating all security practices that are relevant to your environment.

These patterns are essentially security best practices presented in a template format. This format, we feel, will assist the reader in identifying and understanding existing patterns, and enable the rapid development and documentation of new best practices.

1.2 History of Security Design Patterns

Design patterns were first introduced as a way of identifying and presenting solutions to reoccurring problems in object oriented programming. Joseph Yoder and Jeffrey Barcalow [1] were one of the first to adapt this approach to information security. Here, we attempt to build upon this list by introducing eight patterns. The format was adopted from the object oriented design pattern template developed by the Group of Four [2], [3], Appendix A. Of course, no experience with OO programming is required to enjoy these patterns.

The Yoder and Barcalow paper presented the following patterns:

- **Single Access Point**: Providing a security module and a way to log into the system.
- **Check Point**: Organizing security checks and their repercussions.
- **Roles**: Organizing users with similar security privileges.
- **Session**: Localizing global information in a multi-user environment.
- **Full View with Errors**: Provide a full view to users, showing exceptions when needed.
- **Limited View**: Allowing users to only see what they have access to.
- **Secure Access Layer**: Integrating application security with low-level security.

These are a good start, but when we consider the issues that arise when securing a networked application there are others that will apply.

1.3 A summary of patterns discussed in this essay

In this essay we present the following security patterns:

- **Authoritative Source of Data**: Recognizing the correct source of data.
- **Layered Security**: Configuring multiple security checkpoints.
- **Risk Assessment and Management**: Understanding the relative value of information and protecting it accordingly.
- **3rd Party Communication**: Understanding the risks of third party relationships.
- **The Security Provider**: Leveraging the power of a common security service across multiple applications.
- **White Hats, Hack Thyself**: Testing your own security by trying to defeat it.
- **Fail Securely:** Designing systems to fail in a secure manner.
- **Low Hanging Fruit:** Taking care of the “quick wins”.

1.4 How to use these patterns

The patterns described in this essay (along with the ones already published) represent a collection of security best practices. Naturally, depending on one’s environment and goal, some may apply and others may not.

The intent is for the reader to review all patterns and identify those that are relevant to their environment; the implementation of which may define or refine an existing security policy.

Note that the scope of these patterns should not be restricted to software applications alone. For example, Check Point, Single Access Point and Layered Security all apply to network security just as well.

1.5 A Quick Look at How to Use Security Patterns

Let’s assume you have an existing ebusiness site. You have gone through initial due diligence to secure the application, servers, and network. What else can be done and where do you start?

Let’s review the patterns you may already have used:

Session: You know basically who your users are and what they’re accessing. You may have targeted web content and individual login accounts for specialized information.

Layered Security: Your ISP has (assured you they’ve) protected your network with ACLs on their (shared) switch or firewall. Your servers are patched as of two months ago and run minimal services. Your 3rd party applications don’t use their default passwords and don’t run as root.

Risk Assessment and Management: Your clickstream and web logs aren’t encrypted, but customer credit card information exists encrypted in the database. Sensitive corporate information sits on a file server on a separate subnet, behind a firewall.

3rd Party Communication: On a scheduled basis, you exchange information with a business partner. The files are sent cleartext over ftp.

Not bad, but what else can be done? Let’s go through the patterns…

Authoritative Source of Data:

- Are the applications processing the proper data? That is, are they using values from a trusted database or do they originate from a potentially fraudulent source?
- Is the trusted source still valid? Has there been a migration of data or data ownership?

Layered Security:

- How does the firewall restrict access to the corporate firewall? Have these ACLs been revisited lately?
- How are vpn, home DSL users secured?
- Has there been a network or application breach of security? Would you really know if there was? How?

Risk Assessment and Management:

- Do you have managerial support for a company privacy policy? Have you written and kept it up do date? Have the employees read and agreed to it?
- Can you locate all of the sensitive corporate documents? Can you locate those responsible for them – the data owners? Are the documents stored and transferred securely?
- Have you recently performed a vulnerability and risk assessment of your network and applications? Have you addressed the results?

3rd Party Communication:

- Other than cleartext ftp, how is access controlled? Are the passwords ever changed? Is the data sanitized before being processed?
- Are you are actively monitoring your network and have learned to detect anomalous behavior like burst traffic, forged packets or unused protocols?
- Are your business partners adequately segregated from one another? Are you sufficiently protected from them? Could one business partner potentially use your network to attack another partner? How do you know?

The Security Provider:

- Do you provide access via web, ftp or other applications to business partners? If so, is the access control managed centrally? Does the current method scale? Does it need to?
- Do your business applications provide adequate authentication and authorization? Would you benefit from having these services abstracted out to a single system? Could it then be leveraged by other applications and managed centrally?
- Is there a sufficient level of delegated admin?

White Hats, Hack Thyself:

- Are you aware of all known vulnerabilities in you environment?
- How seriously does management take security? Would this change if you sent them their password, or those of your customers?
- How does management view the risk of attack (in financial terms)? Have they tried to quantify the risk?

- Are you prepared (or even able) to take the appropriate legal action in the event of an incident?

1.6 Additional patterns to be discussed

Given that there are many more patterns to be discussed, this essay presents only a limited number. The following are additional patterns to be discussed in a follow-up paper.

- **Distributed Trust**: Distributing trust amongst multiple entities.
- **Least Privileges:** Granting the minimum access necessary to perform any given task, for a minimum amount of time.
- **Role Based Access Control (RBAC):** Abstraction of users from the resources they're attempting to access.
- **Data Privacy, Integrity, Authentication:** Protecting data from eavesdroppers, theft and manipulation.
- **Data Sanitization:** Removal of expired, duplicate and unnecessary data, finding owners, normalizing at times, legalization almost always (i.e. no shared versions of licensed code).
- **Feel the Network:** Learning to recognize load and activity patterns in your environment. Establishing a datum for the purpose of identifying anomalies.

2.0 Authoritative Source of Data

2.1 Alias

System of Record

Trusted Source

2.2 Motivation

Patient heath records are nowadays becoming accessible over public networks. Granted, every packet may be strongly encrypted, with guaranteed privacy, authentication and integrity. While the networked applications may be built securely and provide high availability, this is of little comfort, however, if this highly protected information is outdated or incorrect. Understanding the authoritative source of data means recognizing where your data is coming from and knowing to what extent you can trust the validity of such information.

2.3 Problem

If an application or user blindly accepts data from any source then it is at risk of processing potentially outdated or fraudulent data. Therefore, an application needs to recognize which, of possibly many sources, is the single authority for data.

Are you assured the data you're using is the cleanest and most accurate? In other words, is the data coming from a legitimate source or from an unknown party?

2.4 Forces

- Enterprises with multiple business units fail to recognize which, of many possible data stores, is the proper authority for information. E.g. a local database, corporate HR, managed outsourced provider, etc.
- Web applications store confidential information inside http cookies without properly protecting the contents from theft, modification or impersonation.
- A news wire receives a report of the resignation of several board members of a company. Several employees are also allegedly involved in an internal computer attack. The news wire mistakenly publishes the counterfeit report, causing the company's value to plummet.
- Web applications process (hidden) form values without verifying their integrity.

2.5 Solution

Never make assumptions about the validity of unverified data or its origin.

Business applications are designed to accept, process and (optionally) return information. They may accept data from end users, static repositories or other applications; in real- time, delayed, or by batch processing. Regardless of the origin, type, or purpose, there should be meaningful validation at each step.

In most cases, determining the authoritative source of data will lie with the owner of the business process. They, rather than information security or IT groups, will understand the purpose of data in a larger context. Information security and IT, however, should still advise the business owner on the volatility and integrity of the data source(s) under consideration.

2.6 Example

- Some application servers recognize when an html form value has been changed. They hash the names and values of hidden form fields before they are served to the client and compare the hash when the form is posted back.
- Etailer applications retrieve pricing, discounts from the application's database and never rely on hidden values passed along in form submissions.
- Enterprise applications need to agree on a primary source for employee information and ensure duplicate or expired data has been purged.

2.7 Consequences

✓ Integrity of data is maintained

✓ The risk of processing and propagating fraudulent (poisoned) data is reduced.

✗ Increased time to implement new processes as multiple data sources may be consolidated into one.

2.8 Related Patterns

Session

Layered Security

3.0 Layered Security

3.1 Alias

Tiered/Distributed Security

Defense in Depth

Belt and Suspenders

3.2 Motivation

Networked applications are susceptible to many forms of attack that may target the network, host or application layer and the communication between them. Naturally, the overall security of a system is greatly improved when each one of these layers are identified, protected, and audited for possible weakness.

Moreover, attacks may originate internally or externally. Basing security rules on the premise of “internal users are good” and “external users are bad” is fundamentally flawed (read insider threat) and difficult to manage. With increased use of external business communication channels, it therefore becomes much more difficult to identify which users or sessions are “internal” and which are “external”.

3.3 Problem

Networked applications and the environment within which they operate are vulnerable at many layers and from all directions. Each device, data object, session, file and process is a potential target and needs to be identified and secured.

3.4 Forces

- Once standalone applications are suddenly now networked and unprepared to withstand network attacks.
- Protection of any one of network, server or application is not sufficient to adequately protect the data within an enterprise.
- While one or many components of a system may be protected, it truly is only as secure as the weakest link.
- If a single devices or application fails or is misconfigured it could potentially expose all private resources.

3.5 Solution

Employ security measures at all layers of a networked application and throughout its operating environment.

Router ACLs, address translation and intrusion detection systems protect the network layer. OS hardening, thoughtful application installation and configuration protect the host and the applications that run on it. Practicing secure coding techniques protect all of the above. Finally, proper baselining and monitoring methodologies protect all these layers on an ongoing basis.

Don’t ignore insider threat. Intrusions and attacks can originate from the inside just as they can from the outside. Authentication, authorization, antivirus software, and intrusion detection systems should protect resources from both sides of the corporate boundary.

3.6 Example

- Firewalls provide ingress/egress packet and protocol filtering.
- Application servers and 3rd party services authenticate users over SSL.
- Applications validate form data by length, bounds and type.
- All network and application activity is monitored and logged for analysis.

3.7 Consequences

✓ Greatly improved overall security

✓ Reduced exposure to attack if one security measure should be subverted or misconfigured

✓ Continuously validates security efforts

✗ More complex security architecture

3.8 Related Patterns

Risk Assessment and Management

4.0 Risk Assessment and Management

4.1 Alias

Risk Analysis

4.2 Motivation

Not all information requires the same degree of protection. Patient records, web log files, military tactics, and hourly weather reports all have varying degrees of sensitivity. The proper security of all of this information requires risk analysis. Naturally, if the risk is high, the effort to protect the data should be great. If the risk is low, the protection should be low.

Similarly, hardware and software throughout the enterprise will require varying degrees of hardening. The security requirements of a front-end application server are different than those of an internal development machine. A front-line firewall is secured differently than a QA router.

It is worth noting that this could be considered a catch-all pattern. Since security is all about risk management, every resource (file, servlet, object, datastore, application, server, etc.) warrants risk assessment.

4.3 Problem

Whenever information needs to be transferred, stored or manipulated, the privacy and integrity of that data needs to be reasonably assured. Hardware and software require protection from misconfiguration, neglect and attack. Underprotection of any of these could drive a company to bankruptcy (or legal battle) and overprotection is a waste of resources.

4.4 Forces

- Time and money improperly allocated to protecting resources.
- Risk incorrectly assessed, or not assessed at all.

4.5 Solution

Identifying and assessing risk is the first step to better security. Risk is proportional to the following three variables: threat, vulnerability and cost(value). That is,

`Risk = Threat * Vulnerability * Cost` `Eq. 1`, [4] where;

`Threat` is the frequency of attempts or successes,

`Vulnerability` is the likelihood of success, and

`Cost` is the total cost of a successful breach by this mechanism. `Cost` also accounts for the value of the resource or information being protected.

`Eq. 1` also implies that if any one of these variables is zero, the risk will also be zero. Identifying a practical example of this is left as an exercise to the reader.

Learn to recognize what is valuable and to whom. Different attackers will have different motives and will therefore target different resources. Youth hackers, generally, are motivated by publicity or mischief and seek to deface web pages or spread malware. Professional criminals are motivated by financial reward and may seek to steal credit card numbers or specialized information (secret recipes, blueprints, etc.). Terrorists care little for web page defacement but more for infrastructure denial of service and mass destruction.

4.6 Example

- Production web and application servers are severely hardened, kept up to date with patches and actively monitored.
- QA and development machines have a reduced (from default) set of services running but may be behind on patch updates.
- Customer credit cards are strongly protected and stored encrypted (or not stored at all).
- Hourly weather feeds are not stored or transferred securely.
- Press releases, while hopefully authenticated, need not be encrypted.

4.7 Consequences

✓ Only the appropriate amount of effort is spent to protect data

✓ A better understanding is gained of the profiles of attackers and the value of data they seek.

✓ Recognition of ownership and accountability of data within the organization

4.8 Related Patterns

Layered Security

5.0 3rd Party Communication

5.1 Alias

Virtual Enterprise Network

5.2 Motivation

Enterprises often partner with third parties to support their business model. These may include application and managed service providers, business partners, vendors, and even satellite offices. As part of this relationship, access must be granted to allow potentially sensitive data to travel between the organizations. Without attention to the security of that data and the methods of transfer, one or both organizations may be at risk. Not only is there risk of data theft and manipulation, but also the risk of allowing other organizations to access your resources.

You may trust the partner with whom you entered into a relationship, but you may not trust their contractors, application vendors, networks or firewall configuration. A breach in their network may lead to a breach in yours. E.g. May 30th, 2001, an OSDN break-in that allowed an attacker to jump from Sourceforge to a server of the Apache Software Foundation.

5.3 Problem

Two companies in a business relationship may trust each other, but to what degree? Specifically, when two businesses exchange information, users and/or applications will require access to privileged resources. How can access be granted while at the same time protecting both organizations? Additionally, how can this be managed in such a way that is neither overly complex nor dangerously simplistic?

5.4 Forces

- Companies need to be assured that private information is adequately protected when traveling over a public or private network.
- Applications that communicate with business partners become vulnerable not only to attack from that partner but also from attacks from users who defeat the partners' security.
- Accountability is difficult to assure without a proper security policy signed by all parties involved.
- Security procedures become difficult to manage when both business partners do not share the same security requirements and considerations.

5.5 Solution

Begin by identifying appropriate channels of communication and information exchange. This includes all protocols and any hardware devices that will be used. E. g. an ipsec vpn, https, ssh, or ftp. Next, define the authorized access points. For IP connectivity, this implies defining where connections will be originating and where they are destined. This helps restrict access based on source and destination host. Next, identify all users that require privileged access. Assign usernames and passwords via out-of-band communication.

Additional security will be achieved if all 3rd party traffic can be separated from one another. Switched networks, separate subnets and individual hosts are examples of reasonable practices.

Provide technical and emergency points of contacts and define any fall back procedures. This information becomes critical in the event of system failure and steadfast business deadlines.

Once the risks have been identified and security measures defined, both parties should signoff on these policies. They must commit but be flexible to modify them should the risk or business requirements change.

Both parties should be willing to provide audit and compliancy reports proving adherence to the policy. Under some circumstance, a personnel security audit may be required.

Finally, once a business relationship has terminated, swiftly revoke all access by the partner to your network and applications.

5.6 Example

- Web based extranet access will be available only over SSL. Username and password will be provided via OOB communication or encrypted email.
- Adequate password hygiene will be maintained.
- Users will not share accounts nor escalate their privileges by using another person's account. SUDO will be provided where necessary.
- File transfer will take place on a scheduled basis via ftp. Chroot environments will be configured and files will be pgp encrypted and stored in a write only directory.
- Activity logs will be distributed on an appropriately scheduled basis. Each party is requested to confirm all activity.

5.7 Consequences

✗ Additional security configurations and policies to manage

✓ Properly managed expectations with respect to security precautions and procedures

✓ Auditable activity for both parties

✓ Secured third party communications enables new business partnerships and alliances

5.8 Related Patterns

Layered Security

Risk Assessment and Management

Fail Securely

6.0 The Security Provider

6.1 Alias

Single Sign On

6.2 Motivation

An enterprise application may be comprised of a number of software and hardware components with each potentially performing its own authentication, authorization, or encryption. While some of these components may implement open or standards-based APIs, others may use closed or unknown technology or simply lack functionality altogether. By abstracting security services from individual applications, an organization is able to centralize the management and functionality of the protocols and policies governing authentication and authorization services.

6.3 Problem

When disparate applications seek to provide their own security services, privacy, synchronization and management of data becomes unnecessarily complex. Moreover, applications may not provide the security features or strength required, risking the overall integrity of the data. These applications may be communicating securely or they may be using weak or inappropriately vulnerable methods. Without a common security infrastructure, the management becomes unnecessarily difficult and risks the security of the entire environment.

6.4 Forces

- Desire to use a single service to provide management and auditing for a common set of security services for all enterprise applications.
- Desire to use stronger, or more flexible security features in applications.
- Desire to provide integrity and consistency of data for authentication and authorization.

6.5 Solution

A Security Provider is a central service to which are directed all authentication and authorization requests. Applications such as email, web, corporate applications and others, would communicate directly with the Security Provider. The Security Provider then communicates with a user or policy store to evaluate a user’s credentials and privileges.

A Security Provider has the following properties:

- Authoritative source for user verification (authentication)
- Authoritative source for role assignment and policy enforcement (authorization)
- Provides centralized (and possibly delegated) management of security policies
- Provides consolidated reporting and auditing facilities
- Implements secured connections to possibly separate user and policy data stores
- Defines appropriate type and strength of technology for information protection (encryption) between itself and requesting applications
- May provide single sign on (SSO) facilities across applications
- May provide single sign on facilities across organizations or satellite offices

6.6 Example

- BEA's WebLogic Server can abstract authentication requests to an external user store, affording integration with a Security Provider.
- Entrust and other vendors provide single sign on applications that centralizes user credentials and authorization policies.
- Netegrity's Siteminder can effectively create a single sign on across multiple disparate applications by brokering trust back to the user's "home" authentication service.

6.7 Consequences

✓ Efficient user and data management due to centralized user store

✓ Common set of technologies and standards used for all security services

✓ Transparent session for end users across applications and potentially across participating organizations

✕ Applications need to be configured (or reconfigured) to utilize this common authentication service.

6.8 Related Patterns

Layered Security

Authoritative Source of Data

Roles

7.0 White hats, Hack Thyself

7.1 Alias

Fire Drill

7.2 Motivation

Policies and information security documentation will ultimately fail unless they are understood, practiced, and revised. Once an organization has developed reasonable security measures, the implementation must be verified. Testing security by applying gray hat techniques against your own systems can be quite revealing. The goal is not to crash systems, but to test the behavior and response of your network, application and staff.

7.3 Problem

How can you be assured of the true security of your systems without real-world testing?

7.4 Forces

- After-the-fact discovery of misconfigured security tools or measures.
- Response personnel ill prepared for incident handling.

- Uncertainty of how devices will respond to targeted attacks.

7.5 Solution

Under a controlled, but non-trivial circumstance, plan and execute an attack. You have the option of targeting various parts of your environment:

- Social Engineering (aka Semantic Attack): Attempt to acquire passwords or privileged information from employees by impersonating a manager, office administrator, or operations staff.
- Server: Test backups by randomly deleting (or “misplacing”) a file or directory. Use Crack, John the Ripper or L0ftCrack to determine weak user or application passwords.
- Network, Personnel: Perform a TCP SYN flood against a web, mail, or ldap server. Note this does not need to be an externally facing server. An “internally” facing attack may, indeed, be more educational.
- Application Code: Attempt some of the popular application exploits; buffer overflow, misconfigurations, cookie poisoning, parameter tampering, replay attack. Check for meaningful log messages and abnormal application behavior.
- Passive attacks: Sniffing the wire for cleartext passwords or other confidential information. Perform a TCP and UDP port scan.
- Active attack: Penetration or reconnaissance attack from the outside in.
- Save the viruses, trojans, worms and other malware for isolated testing environments. Be certain to cleanly wipe the infected machines afterwards.

Perform the attacks on an ongoing basis and be sure to record the results. This will be valuable when determining the effectiveness of the tests and the organization’s overall security.

To protect the integrity of the tests, ensure they are performed with limited staff knowledge; you don’t want to spoil the surprise. It is also wise to wait for an appropriate time when there is available staff and there are no corporate emergencies.

Be very careful with these tests; you do not want to permanently damage any system, application or reputation. And of course, this should only be performed against your own environment and not against your customer or business partner.

7.6 Example

- Sanctum’s AppScan has the ability to automate and document controlled web-based intrusion attempts.
- nCircle actively monitors networks and hosts for new activity and vulnerabilities and responds accordingly.

7.7 Consequences

✓ Opportunity to bring controlled security testing into the QA cycle.

✓ Repeatedly testing security measures provides a measurable audit trail of improvement.

✓ Using attacker tools educates security professionals on methods of attack and defense.

✓ Social engineering attacks raise security awareness for all employees.

✓ Helps quantify cost of attempted and successful intrusions to upper management.

✗ Risk of actual damage.

7.8 Related Patterns

Risk Assessment and Management

Fail Securely

8.0 Fail Securely

8.1 Alias

Fail Safely

Fail-safe defaults

8.2 Motivation

Networks, hosts and applications should default to secure operation. That is, in the event of failure or misconfiguration they should not reveal more information than necessary with regard to

- error messages (for efficient debugging purposes)
- the application configuration (directory, version/patch levels)
- the operating environment (network addressing, OS version/patch levels)

As well, they should not allow transactions or processes to continue

- with more privileges than normal
- with more access than normal
- without proper validation of input parameters and output results
- bypassing any monitoring or logging facilities

8.3 Problem

In the event of a failure or misconfiguration of an application or network device, would the result be a more, or less secure environment? That is, would the consequence result in a user performing a given operation unprotected; or a device passing unauthorized information?

8.4 Forces

- Failure of a system without proper error handling may result in a user gaining additional privileges or access.
- During a failure, improper (or complete lack of) fail-safe measures may result in a denial of service condition.

- The silent failure of a security measure (application monitoring tool, IDS, etc.) would prevent administrators from recognizing malicious or anomalous activity.

8.5 Solution

When processing input of any kind, if a problem is detected, fail safely and stop processing the request. Log (and optionally alarm) the incident. Failure to validate or continue could result in any number of unwanted conditions, including a crashed or compromised system, escalated privileges or a denial of service.

Configure systems such that they, by default, prevent all access. Then, selectively add privileges for users, hosts or protocols.

Design critical systems for high availability. This may include the following:

- Hot-swappable hardware (disk, cpu, memory),
- Redundant servers and network devices (email servers, routers, firewalls), and
- Clustered and fail-over applications (web, application and database servers)

8.6 Examples

- Employ the premise of “deny all” and only allow specific protocols, host or users
- Provide system lockouts on consecutive bad login attempts.
- If an application encounters an error while processing a transaction, trap and return the errors and exit cleanly.
- When dealing with sensitive information requiring encryption, if the encryption fails, return an error and ensure all temporary cleartext is securely wiped from disk and memory.
- Create a high-availability environment with redundant or failover components.

8.7 Consequences

✓ System failures are logged and alarmed.

✓ A misconfiguration or software bug does not suddenly expose all resources.

✓ A single device or application failure does not lead to a denial of service.

✗ Extra cost and effort is required to support a redundant and fail-safe enterprise.

8.8 Related Patterns

Low Hanging Fruit

9.0 Low Hanging Fruit

9.1 Alias

Quick wins

9.2 Motivation

New installations of operating systems, applications and hardware are rarely secure by default. Often, they are configured to be as “useable” as possible by enabling most or all services and defaulting to trivial or no authentication mechanisms. Lacking the most current patches, this all results in a very insecure configuration.

Under pressure to bring this into production, there may not be the opportunity to properly secure it. Since the risk of activation may be significant, however, something must still be done. Low hanging fruit are simple fixes that can be implemented quickly and will greatly improve the overall security.

9.3 Problem

Good security is a cycle that requires intelligent planning, careful implementation and meaningful testing. Unfortunately, administrators, developers and managers may not have the time or opportunity to properly complete this cycle. Therefore, taking advantage of the quick wins may be the only opportunity to establish reasonable security.

“Some security now is better than perfect security never.” [5]

9.4 Forces

- Administrators or developers may not have the time to implement perfect security. That is, business or external forces may require that a system be made immediately accessible without undergoing proper hardening.
- The skills required to properly secure applications might not be immediately available.
- An adequate testing environment for new tools and procedures may not be available.

9.5 Solution

Do not to attempt to redesign the environment or reinstall applications. At this stage, the goal is to apply these basic steps to remove obvious vulnerabilities (and gain valuable awareness) of the systems and environment. Each fix (just as with the examples listed below) should be fairly simple to address and execute. The goal is to be able to plug as many holes as quickly as possible.

9.6 Examples

- Patch the software. Patch the hardware.
- Prevent all but essential processes from running on startup.
- Log all network and application activity. Monitor these logs.
- Learn to recognize normal behavior and what may be malicious activity. Pay attention to the activity patterns in your environment (protocols, traffic profiles, most active/ least active users).
- Promote employee awareness programs, perhaps as a weekly security bulletin or message of the day.
- Change the default password when applications are first installed; you don’t need to make it undefeatable for now, just different than the default.
- Run applications as lesser-privileged users (in chroot jails, for example).

- Enable sufficient application error handling and data checking.
- Employ basic authentication on private web directories.
- Vendors will often recommend minimal configuration changes to their products to prevent trivial attacks against default installations. Be sure to follow them!
- Standardize installations of similar machines, with scripting or ghosting. Be sure to patch these source images.
- Remove expired user accounts.
- Remove or disable all unused or “temporary” access or authorization privileges.
- Configure centralized logging (aka a log server).
- Configure TCPWrappers to deny all but specific hosts, and log both failed and successful connections.
- Be aware of vulnerabilities by signing up for industry and vendor mailing lists.
- Replace cleartext protocols with secure alternatives (ssh, https, etc).
- In the absence of proper backup facilities, use tar and custom scripts to backup information.

9.7 Consequences

✓ Some of the most effective security measures can be accomplished with these simple steps.

✓ Servers begin operation with an acceptable, minimum level of protection.

✓ Applications are not left exposed to trivial attacks and vulnerabilities.

✓ Basic troubleshooting and auditing trails are enabled.

9.8 Related Patterns

Layered Security

Risk Assessment and Management

10.0 Appendix A – Pattern Template

Alias: Other well-known names for the pattern, if any.

Motivation: A scenario that illustrates a design problem. The scenario will help you understand the more abstract description of the pattern that follows.

Problem: Describes the problem to be solved.

Forces: Forces determine why a problem is difficult. Describe the forces influencing the problem and solution.

Solution: The solution should solve the problem stated in the problem section. A good solution has enough detail so the designer knows what to do yet general enough to address a broad context.

Examples: Concrete examples that illustrate the application of the pattern.

Consequences: How does the pattern support its objectives?

Related Patterns: What design patterns are closely related to this one?

11.0 References

[1] Architectural Patterns for Enabling Application Security, http://citeseer.nj.nec.com/yoder98architectural.html.

[2] Group of Four design patterns: The template for these patterns were adopted from the template used by the Gang of Four at http://www.hillside.net/patterns/Writing/GOFtempl.html

[3] Pattern Checklist: A checklist of for defining a pattern can be found at http://www.hillside.net/patterns/Writing/Check.html.

- Describes a single kind of problem.
- Describes the context in which the problem occurs.
- Describes the forces leading to the solution.
- Describes at least one actual instance of use.
- Describes or refers to other patterns that it relies upon.

[4] Risk equation, Peter Tippett, executive publisher, Information Security magazine.

[5] Bruce Schneier

[6] "Security Manager Initiates Friendly Fire", http://www.computerworld.com/cwi/story/0,1199,NAV47_STO59330,00.html

[7] http://www.ibiblio.org/pub/Linux/docs/HOWTO/Secure-Programs-HOWTO

[8] SP 800-27, "Engineering Principles for Information Technology Security (A Baseline for Achieving Security)", June 2001, http://csrc.nist.gov/publications/nistpubs/800-27/sp800-27.pdf

12.0 About the Author

Sasha Romanosky is currently a Senior Security Engineer at a major financial institution and lives in San Francisco. He has a Bachelor of Science in Electrical and Computer Engineering from the University of Calgary, Canada and has been working with computer and Internet technologies for over 6 years. His passion is Internet security.

He can be reached at sasha_romanosky@yahoo.com.