Conteúdo

1. Information Obscurity

1.1. Intent

All systems are potentially liable to attack, whether from internal or external sources. If the information held by a system is sensitive, it should be protected. Part of this protection can take the form of obscuring the data itself, probably through some form of encryption, and obscuring information about the environment surrounding the data.

1.2. Example

A typical Internet technology system will use a combination of Web and application servers, together with a COMMON PERSISTENT STORE [1], usually in the form of a common database, in which application data is stored. All these parts of the system will be protected from external attack by a firewall and possibly a DEMILITARIZED ZONE. However, this is no guarantee of security—what if the attacker breaches these external measures, or if an attack is internal to the organization?

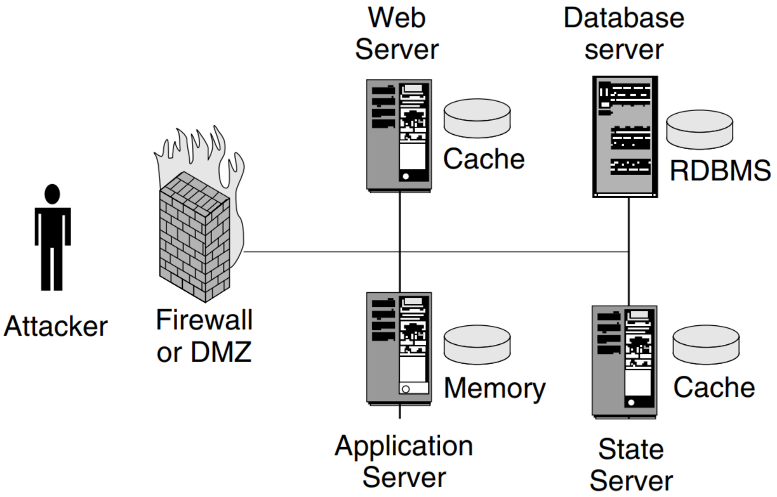

Figure 1: Protection using a firewall or DMZ

The system will gather user information, such as credit card details, and store this in the database. The user information in the database is an obvious target for any attacker who wishes to steal or alter such information. Hence extra security measures may be put in place for the database. However, user information may also be retained temporarily by other parts of the system, in memory, in a cache, or in session state server, as shown in figure 1.

Application data can be protected by encrypting it, but such encryption is comparatively slow. Widespread use of encryption for all data in the system will impact system performance. Even then, there is no guarantee of security, as the system must have access to the keys required to decrypt the data when it is needed by the application. This means that such keys are also vulnerable to attack. If the intruder can find and identify the encryption keys used for particular purposes, then all benefit from the encryption is lost. This can be addressed by designating one server to hold and distribute the keys. This server can then be specially protected. However, if an intruder can obtain credentials to access this server, then it too may be compromised, hence anywhere the application has access to such credentials (or equivalent privilege must also be protected).

1.3. Context

An APPLICATION SERVER ARCHITECTURE [1] has been adopted to deliver Internet technology application servers together with a COMMON PERSISTENT STORE [1]. The business logic and dynamic Web content generation of the application resides on application servers, while all static content is provided by Web servers that also act as a PROTECTION REVERSE PROXY or an INTEGRATION REVERSE PROXY for the dynamic Web content. The application gathers information on users and holds this in its database. The application is protected from external attack by a DEMILITARIZED ZONE.

1.4. Problem

How do we ensure that sensitive data gathered and stored by our system is protected from unauthorized access?

The solution to this problem must resolve the following forces:

- Much application data is non-sensitive, but the data that is sensitive needs to be protected in parts of the system that are vulnerable to attack. The degree of protection should be commensurate with the sensitivity of the data, and the data must still be readily accessible by the system itself.
- Encryption and decryption are comparatively slow and expensive in resource terms and so should be avoided unless necessary.
- To encrypt and decrypt information you need the appropriate encryption key. However, you must then guard this encryption key from unauthorized access.

1.5. Solution

Grade the information held by the system for sensitivity. Obscure the more sensitive items of data using an encryption mechanism in situations in which it might be exposed to attack, while leaving the bulk of the application data unencrypted. Take appropriate measures to protect the encryption artifacts, such as encryption keys, from direct attack.

1.6. Structure

INFORMATION OBSCURITY requires the following elements:

- Encryption keys, to encrypt and decrypt sensitive data.
- A key storage mechanism, to store and possibly distribute the keys. This could be anything from a system registry to an off-the-shelf key management server.
- An encryption mechanism, used with the encryption key to obscure data.
- An application component or components obtain and use encryption keys to secure application data in various parts of the system.
- A protected location, a place to store encryption artefacts used by the system. This location should itself be defended by obscurity and/or other defense mechanisms.

The following relationships govern the encryption of data to obscure its contents:

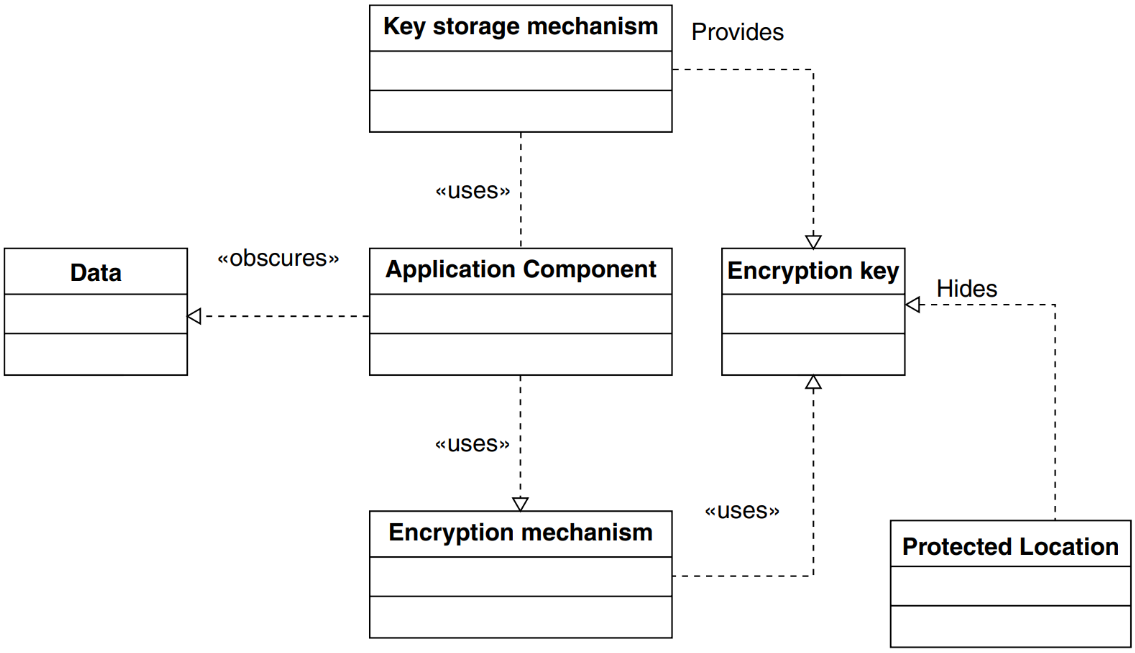

Figure 2: INFORMATION OBSCURITY structure

The application component uses an encryption mechanism, seeded with an appropriate key, to obscure the data it uses.

1.7. Dynamics

1.8. Implementation

Only part of the data held needs to be obscured, as only part of it is sensitive. The first task is therefore to categorize the data held and used by the system. This process of identification and classification is a form of SECURITY NEEDS IDENTIFICATION FOR ENTERPRISE ASSETS, in this case based on considerations such as:

- The impact should that data be accessed by an unauthorized third party, for the user, for the company, and for the relationship between the two: for example, a list of HIV-infected patients on a medical system.
- The incentive for a third party to find this data: for example, credit card details.
- The accessibility of the place where the data is stored: for example, in a cache file on disk.
- Whether this data can be used to compromise further data.: for example, an encryption key.
- The data protection rules governing the specific type of data.

The last point should be well noted, as in many countries there are legal requirements for organizations to take due care in the management and protection of information gathered from customers and clients. Failure to conform to the appropriate set of rules will not only be insecure, but also illegal.

Because part of the sensitivity assessment is based on the location of the data in the system, and hence its exposure to attackers, this audit should be repeated whenever the system architecture changes in a major way as the system evolves—for example, the introduction of a DEMILITARIZED ZONE. Ideally you should make the decisions about the sensitivity of the data independently of the decision about the obscurity mechanism to be applied. If you find that you have lots of data that needs to be obscured to a high level, then the project's sponsors should be persuaded to make the budget available to do this.

Most sensitive user data will still be stored in a database. Small amounts of information can be encrypted and stored in character- or byte-based fields, while larger amounts of ciphertext would be stored as BLOBs (binary large objects). Whether you store your encrypted data in the database or on the file system, you will need metadata to describe it in order to identify the user with which it is associated, used as a primary key in the database, or the file name on the file system. For custom software elements you can use the encryption APIs provided in the Java and .NET world to manipulate encrypted data, although you need to be aware of some limitations built into cryptographic products exported from the USA, which limit key lengths for 'foreign' implementations. Alternatively, you can buy third-party cryptographic libraries from many places that achieve the same purpose.

If any part of your system is not enabled for encrypted data, you may need to build a custom adapter. One way of reducing the need for obscurity is to increase the number of the strength of the 'locks' through which a cracker must pass to be able to access the data. You might find that it is easier in overall terms to implement a stronger DEMILITARIZED ZONE and use less encryption within the internal network than to make many parts of your application encryption aware.

One thing to remember here is that INFORMATION OBSCURITY, when applied to data, is concerned with the protection of information inside the application. Once it moves outside the application, or even onto the network between elements in the application, this data is still potentially vulnerable. For this reason, you often find INFORMATION OBSCURITY used in combination with SECURE CHANNELS, so that data is protected both inside the system and in transit.

As noted earlier, it is not just user data that needs to be protected, but also the configuration information used by the application. To be flexible, information used by the application for its own purposes is often held externally, for example in configuration files. However, some of this configuration information is in itself sensitive information. An example of such application configuration data would be an encryption key. The application needs the encryption key to access encrypted user data, but you do not want an intruder to obtain it easily. Information-based security artefacts such as encryption keys are particularly sensitive, as they can just be stolen—copied—without your knowledge if you don't spot the intrusion.

To secure this type of data, you could secure your external configuration file from unauthorized access. In addition, you could use an obscured name to identify the key in the configuration file. This makes it more difficult for an attacker to identify their target information. If you are still not happy with the level of security—for example, the file could be accessed over the network if the system is configured incorrectly— you could move the sensitive data into a location that is only accessible to local principals, such as the Windows system registry. Alternatively, you can embed the information in a binary artefact such as a compiled class or resource component to make it more difficult to retrieve. In a late-bound environment such as Java or .NET, you might even want to obfuscate your bytecode or intermediate language to make it even less obvious which bit of data is the key.

Obviously, most of the considerations for the encryption key relate to the strength of the ‘lock’ protecting it. In the case of other sensitive information, such as a database connection string containing credentials for the database, encryption can be used to obscure the contents of the string to help prevent the discovery and use of the embedded credentials. This encrypted information can then be placed in a suitable location, as discussed above for encryption keys.

Once you have decided what is to be encrypted, you need to consider the impact on the rest of the system. The main issue with encryption is speed. Encryption and decryption on general purpose computer systems requires resource-intensive cryptographic algorithms to be run using the standard processor and memory. Although these resources are suitable for general application server usage, they are quite slow compared to what you would ideally want for cryptographic purposes. If you only require a small amount of cryptographic processing, this is usually acceptable. However, the more cryptographic processing you require, the more impact is caused by running it on sub-optimal hardware.

One solution would be to upgrade all the systems to have faster processors, for example, more on-board cache and faster memory. However, this would increase the cost of each system noticeably. The alternative is to buy dedicated hardware that performs the encryption and decryption. Depending on the level and type of encryption required, this would probably be cheaper than upgrading the processor and memory. It would almost certainly be faster.

One final aspect to consider is infrastructure security, as application security can be undermined by an insecurely configured infrastructure. To address this, INFORMATION OBSCURITY can also be used to help to improve the security of the infrastructure. Some parts of the system already use obscurity, for example when storing passwords. However, this can be undermined if a suitable password policy is not enforced. Other steps can be taken to make a system less vulnerable to attack, such as using obscure host names rather than, say, ‘dataserver,’ ‘kerberos1’ or ‘keymanager.’

1.9. Example Resolved

All public data, such as catalog information held in caches and in memory on the Web servers, is held in plain text. However, any credit card details are held in encrypted form. The only place in the system where such details appear in plaintext is in memory on the application server as it is delivering this information to the credit card processing agency.

After weighing the possible consequences of data disclosure against the risk of intrusion, it is considered that the system contains other data worth encrypting explicitly—customer information. The passwords used by customers for personalization are encrypted anyway by the personalization and customization engine, but their personal details however are not. The encryption used for the customer information is not too strong, as we don’t want to impact system performance too much. The main intention is to make it difficult for any intruder to break this encryption casually.

One point to note is that there is a single encryption key used for all customer information, not one per customer. There is little benefit (and much complexity) in the use of multiple keys, as the intention is that the application server software is authorized to view this data, as it has the decryption key. The authentication and authorization of each customer is a separate matter —see KNOWN PARTNERS.

1.10. Consequences

The following benefits may be expected from applying this pattern:

- Security is improved by data obscurity because, even in the event of an attack during which the attacker gains access to the file system, system memory and application database, sensitive data is not usable by the attacker.
- The impact on system performance is minimized, because only a small percentage of the application and system data is typically encrypted in order to deliver a reasonable level of security.
- Security is also improved by configuration obscurity, because any attacker will find it more difficult to obtain the information, they need to crack the system.

The following potential liabilities may arise from applying this pattern:

- Performance is impacted if an obscurity mechanism is introduced, due to the processing overhead associated with the mechanism. This is particularly true of complex encryption algorithms with long key lengths.
- Manageability is impacted, as additional configuration will be needed for any encryption mechanism, such as key management.
- Components that use obscured data may need to encrypt and decrypt that data themselves, so adding to the cost and effort of developing them.
- Cost is probably increased, as the extra requirements for encryption may require either additional general capability to support software encryption, or dedicated encryption hardware. You may also need to buy additional encryption software, depending on what comes with your existing platforms and tools.

1.11. Known Uses

Web application components that cache sensitive data on the Web server will obscure the data in those caches. User authentication mechanisms apply INFORMATION OBSCURITY to the data they need to maintain by only encrypting the user’s password.

1.12. See Also

1.13. References

[1] Dyson, P., & Longshaw, A. (2004). *Architecting enterprise solutions: patterns for high-capability internet-based systems*. John Wiley & Sons.

1.14. Source

Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., & Sommerlad, P. (2006). Security Patterns: Integrating Security and Systems Engineering (1st ed.). Wiley.

2. Intrusion Detection Requirements

2.1. Intent

An intrusion detection system (IDS) must satisfy a set of requirements for both the service and the quality of service. IDS is a security service that automates the monitoring of events occurring in a computer system or network and analyzes these events for any indication of security violations. While each situation that calls for intrusion detection is unique, there are common generic requirements that apply to all intrusion detection situations. This pattern provides a common generic set of intrusion detection requirements. The pattern also helps you to apply the general requirements to your specific situation and helps you to determine the relative importance of conflicting requirements.

2.2. Example

The museum's research department has a network that they use for messaging and collaboration with various universities around the world. Among the information exchanged and stored are details about the location of various natural gemstone mines. Samuel the museum system engineer wants the museum immediately to detect unauthorized and successful attempts to gain access to the network and to any hosts that contain sensitive information. Once alerted, Samuel would like information that can be used to hold accountable the individual(s) that have breached their perimeter. In addition, Samuel would like to have information recorded and available on unsuccessful attempts to gain access. Samuel understands that trade-offs are involved, because stopping intruders and capturing information about attempted intrusions can require significant resources that degrade system performance, and which may make legitimate access more difficult. Privacy considerations are also a constraint on intrusion detection efforts. Samuel needs to identify requirements for an IDS service that will help the museum achieve the goals while balancing the constraints.

2.3. Context

Accounting requirements and their relative importance are understood. The requirements might have been selected by applying SECURITY ACCOUNTING REQUIREMENTS. The planned uses of IDS are understood.

2.4. Problem

IDS is a security service that automates the monitoring of events occurring in a computer system or network. It analyzes these events for any indication of security violations. You need a clear set of requirements to ensure that the intrusion detection strategy employed actually satisfies the needs of the organization or system. Requirements for intrusion detection often conflict with each other, and trade-offs among them are often necessary. The conflict stated in the example is that the need to detect intrusion must be balanced with resource and privacy constraints.

What types of information are appropriate or required for an IDS to analyze? How can you determine a balanced set of specific requirements for an IDS service, and their relative importance?

The process of selecting and prioritizing intrusion detection requirements needs to balance the following forces:

- Applying intrusion detection increases the likelihood of achieving the desired security properties, especially accountability and integrity.
- Applying intrusion detection has associated costs, such as software, additional processing time and resources, and risks, such as privacy violations.
- Intrusion detection errors can result in two different types of problems. First, if an intrusion occurs that violates security, and the IDS service does not detect it or prevent it, then damage can occur, and it might not be discovered until a later time. Second, if no intrusion occurs but the IDS incorrectly believes an intrusion has occurred, then resources are wasted trying to respond to a problem that does not exist.

2.5. Solution

Specify a set of intrusion detection requirements for a specific domain such as a system or network and determine the relative importance of each requirement. The solution has two aspects: a requirements process and a common set of generic requirements.

2.5.1. Requirements Specification and Prioritization Process

The requirements process is typically performed by a system requirements engineer in conjunction with an enterprise architect and includes several activities. An important first step is explicitly to define the domain for which IDS requirements are to be specified, such as a specific system or facility. Factors that affect specialization and importance of requirements are also defined, such as organization constraints. IDS requirements for the target domain are then specified, using the generic requirements provided below. The final activity is to define the relative importance of the specified requirements.

2.5.2. Generic Requirements Description

The following are general requirements that drive the design of an IDS Service:

- Detect intrusion events. An IDS service must detect intrusion attempts. This information is used to determine organization vulnerabilities. By its very need to provide immediate information, IDS services will only be able to provide information about security events as it is received. While some IDS services can provide a degree of correlation between events, there is an inherent time delay before such information can be reported.
- Report on successful intrusions and thwarted intrusion events. Reported information includes actor identities and any distinguishing characteristics of the events. The information should also include, but not be limited to: the location of the actor, software or hardware used in the attack, discussion of whether or not any elements of the attack were detected in advance, and the responses that ensued.

- Provide countermeasures against intrusions. An IDS service has the responsibility to try to thwart intrusion attempts. An IDS service will need to perform some event correlation so that it will be able to recognize attack patterns and warn security officers and system administrators. Compiling user profiles based on behavior patterns can also help to recognize and thwart attacks. If reasonable, the IDS service should be permitted to shut down avenues of access when attack patterns indicate that an attack is beginning to happen. In some cases, the known presence of an IDS may in itself deter actors from engaging in malicious activity.
- Support the capability for repeated examination of information derived from an event. The IDS service needs to provide the security events and information it detects to the normal audit trail and logging mechanisms for capture and storage for the longer term.
- Perform its service when needed. The IDS service will itself require protection. An IDS needs to be available to provide its services when the tracking of events is absolutely important. During operation the IDS should be aware of events that could cause significant damage to the organization, and the IDS service needs to be able to continue functioning during those high-impact events.
- Provide reliable and accurate information. Malicious actors should not be able to tamper with information the IDS service obtains or generates: the IDS should protect its own information as far as possible. Decision makers will need to judge how well the IDS information is protected from malicious actors. This requirement is essential to support confidentiality and integrity.

An additional set of requirements applies to all service requirements patterns. Instead of duplicating the discussion of the same set in each requirements pattern, they are simply listed here, because they do need to be considered in each requirements pattern. The requirements are: minimize time and effort to use, minimize mismatch with user characteristics, risks to user safety, costs of per-user set-up, costs of maintenance, management, and overhead, and changes needed to existing system infrastructure. Further discussion of each of these cross-cutting requirements, including implementation factors, is given in I&A REQUIREMENTS.

2.6. Structure

2.7. Dynamics

2.8. Implementation

This section first provides more detail on the process that was summarized in the Solution section, then discusses factors in determining relative importance of requirements.

2.8.1. Process Guidelines

The requirements process is typically performed by a system requirements engineer in conjunction with an enterprise architect, and includes several steps:

1. Establish the domain for which the intrusion detection service is needed. Ensure that the domain has been identified and scoped. Typical intrusion detection domains include [1]:
    a. Trespass: gaining unauthorized physical access to sensitive data by circumventing a system's protections
    b. Penetration: gaining unauthorized logical access to sensitive data by circumventing a system's protections
    c. Reverse engineering: acquiring sensitive data by disassembling and analyzing the design of a system component
    d. Cryptanalysis: transforming encrypted data into plaintext without having prior knowledge of encryption parameters or processes

    Other constraints may also bound the domain.
2. Specify a set of factors that affect specialization and importance of requirements. The factors include use of IDS, intrusion detection needs, response needs, organization constraints, and priorities. You can find a general candidate set of factors below.
3. Specify the intrusion detection requirements for the target domain. To do this, specialize the set of generic requirements given in the Solution section.
4. Define the relative importance of specific requirements. You can find more details on the association of factors and requirements below

2.8.2. Factors in Determining Relative Importance

Table 1 reiterates the generic requirements described in the Solution section, along with factors for judging their relative importance to the organization. For each requirement, positive and negative impacts of the factors on importance or priority of the requirement are also provided.

Generic Requirement	Factor	Resulting Priority
Detect intrusion events	Potential intrusions could give access to highly sensitive or valuable assets or could cause significant damage.	High
	Intrusions would not cause significant loss or damage, or the loss is covered by insurance.	Low
Report on successful intrusions and thwarted intrusion events	Strong need to assess quality of IDS and patterns of intrusion attempts.	High
	Information needed only for insurance claims.	Medium
Provide countermeasures against intrusions	Potential intrusions could cause loss of or damage to highly valuable assets that could not be replaced or repaired.	High
	Assets could easily be replaced or repaired.	Low
Support the capability for repeated examination of information derived from an event	IDS is the only accounting service deployed and understanding of patterns that emerge over time is needed.	High
	An audit trail and logging service is deployed, or the primary need for IDS is to detect and thwart current attacks.	Low
Perform its service when needed	Potential intrusions could give access to highly sensitive or valuable assets or could cause significant damage.	High

	Intrusions would not cause significant loss or damage, or the loss is covered by insurance.	Low
Provide reliable and accurate information	Strong need to assess quality of IDS and patterns of intrusion attempts.	High
	Information needed only for insurance claims.	Medium

Table 1: Intrusion detection system service requirements factors

2.9. Example Resolved

Samuel the museum systems engineer defines the museum research network as an IDS domain. Table 2 shows the requirements ratings Samuel has specified for this domain.

Requirement	**Museum Priority and Concern**
Detect intrusion events	HIGH – The museum wants immediately to detect unauthorized and successful attempts to gain access to the network and to any hosts that contain sensitive information.
Report on successful intrusions and thwarted intrusion events	HIGH – Once alerted, the decision makers would like information that can be used to hold the individual(s) that have breached their perimeter accountable.
Provide countermeasures against intrusions	MEDIUM – The museum wants to thwart intrusions, but for this domain, the benefit-to-cost ratio for this capability is less than detection and reporting.
Support the capability for repeated examination of information derived from an event	LOW – The museum is most interested in current attacks rather than long-term analysis.
Perform its service when needed	HIGH – The problem statement clearly states that the museum needs the IDS to capture malicious activity. The museum must have confidence that the IDS can perform this task.
Provide reliable and accurate information	MEDIUM – Information on malicious actors is important but protecting other tracking information is only moderately important.

Table 2: Resolution of example problem for IDS requirements

2.10. Consequences

The following benefits may be expected from applying this pattern:

- It facilitates conscious selection of IDS requirements, so that decisions about selecting IDS mechanisms have a clear basis, rather than occurring in a vacuum.
- It promotes explicit analysis of trade-offs that encourages balancing and prioritizing of conflicting requirements and forces. This includes balancing the need for accountability with the need for privacy. This helps to avoid stronger than necessary IDS mechanisms that would generate excessive false warnings or cost too much, and at the same time it helps to avoid a weaker than necessary IDS that makes it easy for malicious actors to penetrate.
- It results in documentation of IDS requirements that communicates to all interested parties and is useful in determining the adequacy of accounting services such as IDS.

- The explicit requirements resulting from the pattern foster a clear connection of requirements to audit and intrusion policies: this also encourages organizations to make their accounting policies more explicit.

The following potential liabilities may arise from applying this pattern:

- It requires an investment of resources to apply the pattern, including time to analyze domains and IDS needs. In some cases, the cost of applying the pattern may exceed its benefits.
- It poses a danger of possibly violating privacy rights if extensive actor data is captured and analyzed. You can mitigate this by capturing and analyzing the minimum amount of data, and by working closely with your legal department.
- The formal selection process may be too long and costly and produce too much overhead. You can mitigate this in the same ways as noted above.
- Specific circumstances might not be covered by generic IDS requirements. You can mitigate this by adding specific requirements and including them in the trade-offs.
- Documentation of requirements implies that they must be maintained as they change over time. You can mitigate this by keeping the requirements in a form that is easy to update, integrated with other system documentation.
- Perception of IDS requirements can differ throughout an organization or in a particular domain. This may make it difficult to reach agreement on the relative priorities of requirements. On the other hand, bringing such disagreements to the surface may be a benefit of the pattern, because then they can be properly discussed and resolved.

2.11. Known Uses

The general IDS requirements and the process of specifying IDS requirements described in this pattern are widely known, but are generally used informally, as opposed to being codified or published. The requirements as stated here represent a consolidation of MITRE Corporation’s experience in working with multiple customers over several decades. However, some publications on intrusion detection and IDS requirements exist. Examples are:

- [2] discusses intrusion detection as one of the primary safeguards.
- [3] is an international standard that defines evaluation criteria for information technology security. It includes criteria that address IDS requirements, although the discussion is tangential and in the context of audit and system monitoring activities.
- [4] discusses requirements for IDS message exchange in the context of the Internet.
- [5] discusses requirements for wireless IDS.
- [6] discusses criteria for organization-wide IDS products.

2.12. See Also

AUDIT TRAILS AND LOGGING REQUIREMENTS describes requirements for capturing and storing information that could be passed from an intrusion detection system.

2.13. References

[1] Shirey, R. (2000, May). Internet Security Glossary. Network Working Group. *RFC 2828*. http://www.faqs.org/rfcs/rfc2828.html

[2] “Technical Report 13335-4:2000 Information Technology - Guidelines for the Management of IT Security - Part 4: Selection of Safeguards” International Organization for Standardization, 2000-03-01

[3] International Organization for Standardization. (1999). Common Criteria for Information Technology Security Evaluation (Version 2.1). CCIMB-99- 031. ISO/IEC JTC 1 adopted CC 2.0 with minor modifications in June 1999 as ISO/IEC 15408, Version 2.1

[4] Intrusion Detection Working Group. (2002, October 22). Intrusion Detection Message Exchange Requirements. http://www.ietf.org/internet-drafts/draft-ietf-idwg-requirements-10.txt

[5] Farshchi, J. (2003). Wireless intrusion detection systems. *http://www.securityfocus.com/infocus/1742*

[6] Liesen, D. (2002). Requirements for Enterprise-Wide Scaling Intrusion Detection Products, A Criteria Catalog for IT Executives, IDS Users and Vendors.

2.14. Source

Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., & Sommerlad, P. (2006). Security Patterns: Integrating Security and Systems Engineering (1st ed.). Wiley.

3. Known Partners

3.1. Intent

An organization conducting e-commerce, offering services, or publishing information using Web technologies must make their service easily accessible to their users. However, if these interactions are commercially sensitive or of a high value, we want to ensure that the users with whom we are interacting are who we think they are, and the users themselves want to be sure that our system is what they think it is. By introducing a system of KNOWN PARTNERS, identified uniquely in a way that can be authenticated, we can be sure of who is interacting with our system. We can also prove to users that we are who they think we are.

3.2. Example

A commercial Internet system offers two Web-technology interfaces: one for the general public and the other for business partners. The business partner interface allows the users to place orders for goods, often with a value that runs to many tens of thousands of dollars. Once the order is placed with the Web-technology system, it is sent to the corporate ordering facility. This initiates a number of supply-chain-management functions, culminating in the goods being shipped to the business partner along with an invoice for the goods.

If we allowed anyone to access this system anonymously, we would run the risk that, either maliciously or accidentally, orders would be placed by users not authorized to do so. This could result in goods being shipped in error, invoices being issued incorrectly, and business partners claiming that orders shipped to them were never placed by them.

Equally, users will be less willing to use the system and to submit information such as credit details and user information for an order, if there is a chance that someone is ‘spoofing’ the system, for example offering something that looks like our system, but is in fact an operation set up to collect information that can be used to commit fraud.

3.3. Context

An APPLICATION SERVER ARCHITECTURE [1] has been adopted to deliver an Internet technology application. The business logic and dynamic Web content generation of the application resides on application servers, while all static content is provided by Web servers that also act as reverse proxies (see PROTECTION REVERSE PROXY, INTEGRATION REVERSE PROXY, and FRONT DOOR) for the dynamic Web content. The application provides commercially sensitive or high value services to a restricted set of users.

3.4. Problem

We want to provide a system that allows us to collaborate with an organization either as a customer or as a business partner. How can we validate the identity of an organization so that we can be sure they are who we think they are, and they can be sure that we are who we say we are?

Solving this problem requires you to resolve the following forces:

- We want to make the system as easy to access as possible to encourage business: this is probably one of the reasons we chose to offer the system via Web technologies in the first place. However, we need to balance accessibility against the need to identify and authenticate users, and to protect users from anyone who is trying to spoof our system.
- Lightweight security mechanisms such as username and password combinations are typically one-way: they identify the user to the system, but not vice-versa. We could adopt a lightweight approach, but these types of mechanisms are relatively easy to break, and the user is often required to provide information that is valuable to anyone that has gone to the trouble of setting up a spoof system.
- The cost of an extensive security solution will be high, but the cost of invalid system use may also be high in terms of theft and loss of customer confidence. If the potential rewards from the attack are high in terms of financial gain or publicity, the risk of such an attack will be higher. The scope, and hence cost, of any countermeasure must be commensurate with the level of perceived threat and the potential cost of the fraud.

3.5. Solution

Ensure that access to system functionality and data is restricted to known partners who must authenticate themselves in a secure manner. This ‘secure manner’ should involve some form of two-way exchange such that the user is identified to the system and the system is shown to be what the user thinks it is. In effect, the user and the system are both identifying each other as KNOWN PARTNERS with whom they want to interact.

3.6. Structure

This pattern requires the following elements:

- System identity. The system has an identity that verifies to the user that the system is what they think it is.
- User identity. The user has an identity that verifies to the system that the user is who it thinks it is. This identity can be passed through the system to provide non-repudiation of interaction: that is, the user cannot claim that an interaction was performed by someone else, and that they should not be responsible for the consequences of the interaction, because the interaction is effectively ‘signed’ with their identity.
- User identity verification service, a service either provided by the system or by a trusted external agency that verifies that any user identity submitted to the system is valid.
- SECURE CHANNELS—identities are usually exchanged via a secure channel, as well as any further interactions between the user and the system.

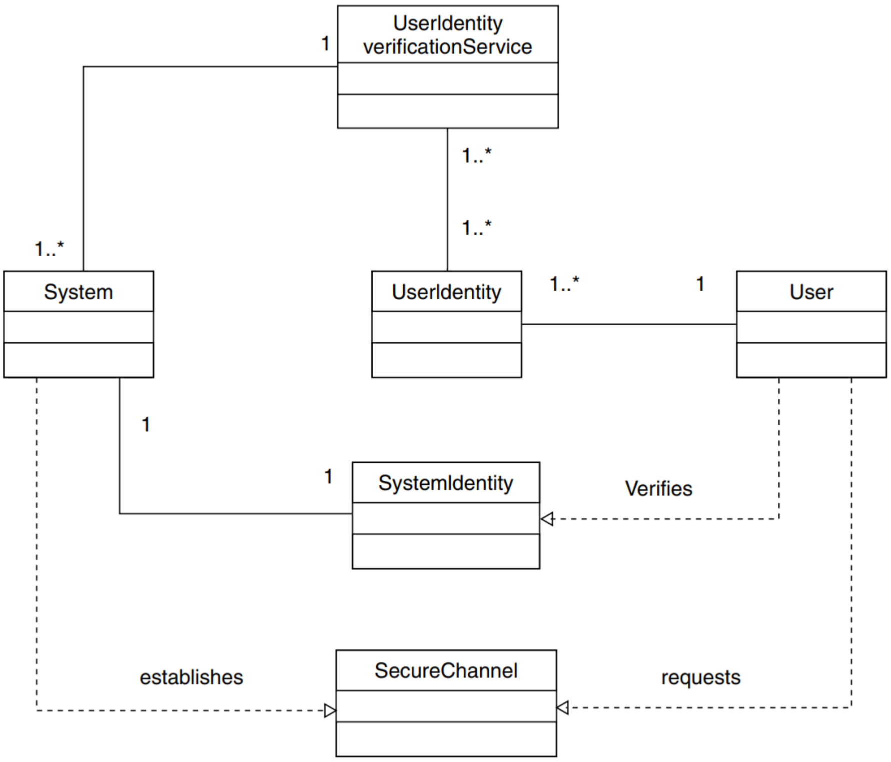

Figure 3: The structure of KNOWN PARTNERS

3.7. Dynamics

The first scenario (Figure 4) shows a successful interaction between the user and system. The user wants to send the message Message to the system which requires access to restricted functionality. First, both the User and the System need to establish that they are each KNOWN PARTNERS.

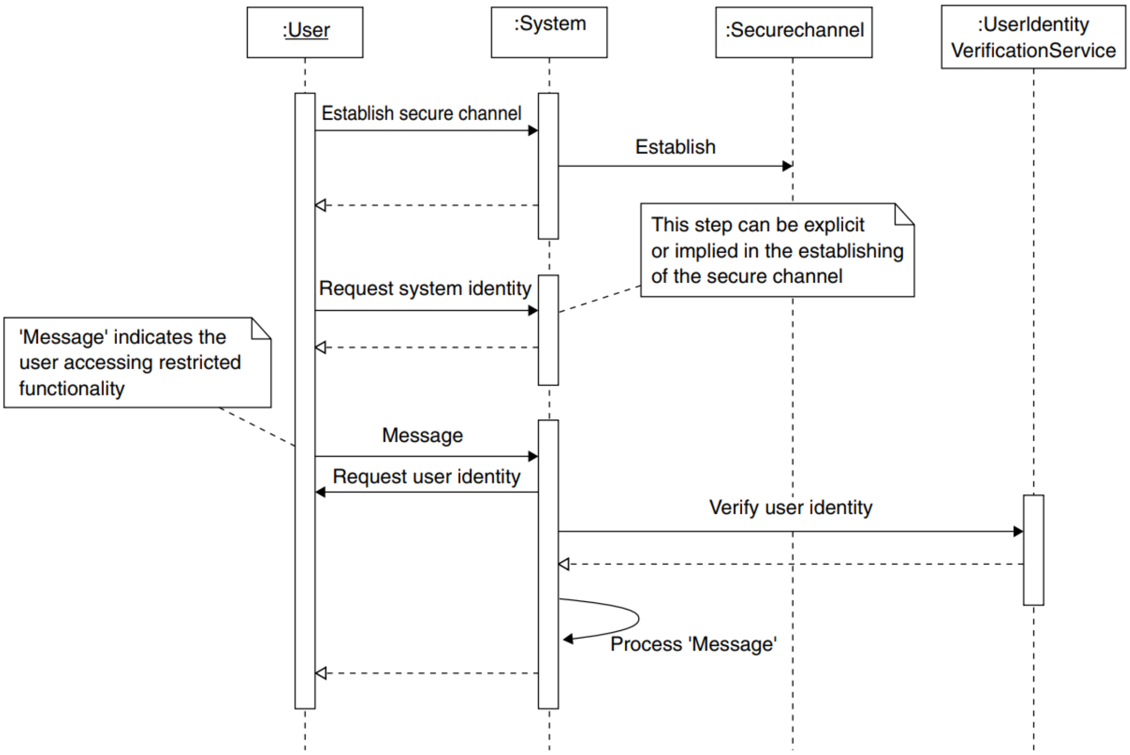
:User
:System
:Securechannel
:UserIdentity VerificationService
Establish secure channel
Establish
This step can be explicit or implied in the establishing of the secure channel
Request system identity
'Message' indicates the user accessing restricted functionality
Message
Request user identity
Verify user identity
Process 'Message'

Figure 4: Successful interaction between KNOWN PARTNERS

The second scenario shows an invalid client identity being detected and blocked by the system. See figure 5.

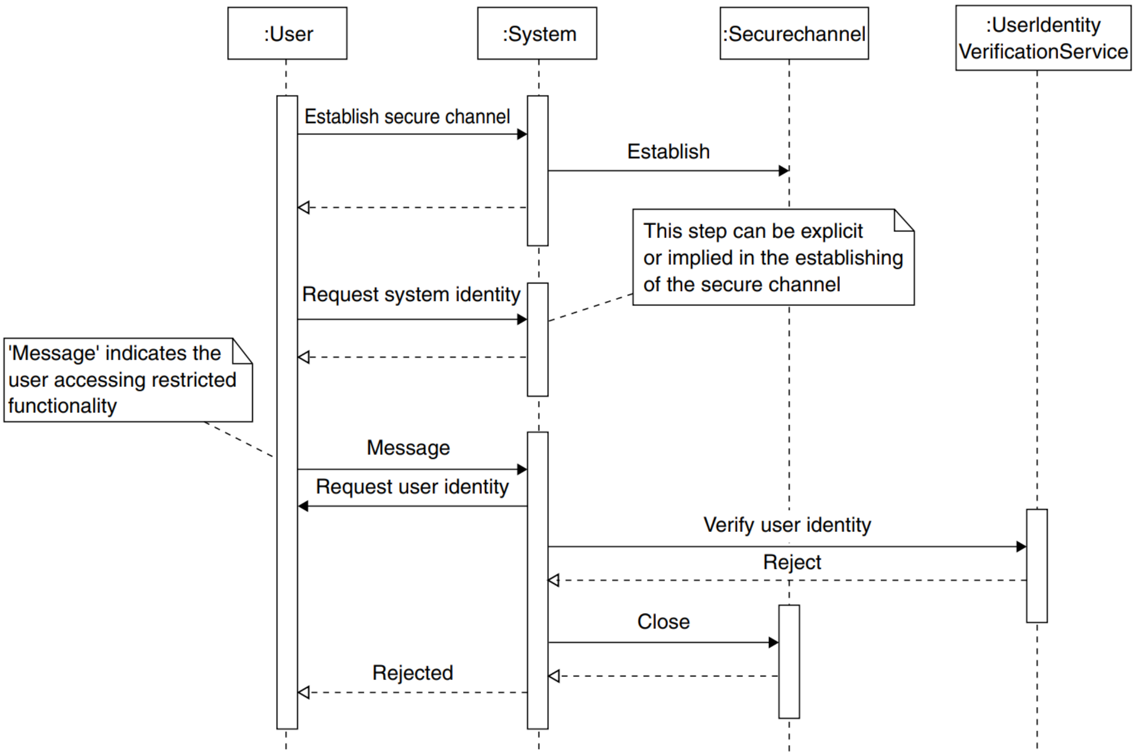
:User
:System
:Securechannel
:UserIdentity VerificationService
Establish secure channel
Establish
This step can be explicit or implied in the establishing of the secure channel
Request system identity
'Message' indicates the user accessing restricted functionality
Message
Request user identity
Verify user identity
Reject
Close
Rejected

Figure 5: Rejecting an invalid client

3.8. Implementation

One of the commonest implementations of KNOWN PARTNERS is to use digital certificates for both the system and client identities. In this case the system provider obtains a certificate from a known certification authority (CA). This certificate has the domain name of the system embedded in it. When the user first connects to the system, the system provides the certificate, while the browser verifies that it is correct for the domain name and has been authorized by a known CA. This prevents spoofing, as a spoof organization should not be able to obtain a CA-authorized certificate and, if they do, it will not be tied to the domain name of the system provider.

To access any restricted functionality, the user also needs to obtain a CA-authorized certificate. This certificate is installed directly into the browser, where it is secure from tampering—although the machine on which browser is installed may not itself be secure. When the user's browser has verified that the system provider's certificate is valid, the user must then provide their certificate to the system. The user identity verification service then checks that the user certificate is also valid—that is, it is CA-authorized and has not been revoked or expired by the CA. If the certificate is valid, the user is given access to the restricted functionality.

To ensure non-repudiation, it is not uncommon for the system to require that the user certificate is passed to the system for every interaction or culmination of interaction, such as the confirmation of order placement. This means that the certificate's details can be stored with the results of the interaction (or passed to back-end systems). As long as the system provider can demonstrate that there is no way within the system for one user's certificate details to be replaced by another, it is very hard for the user to contend that the interaction was not carried out by them, and that they therefore should not be liable for its consequences.

How the client obtains their certificate is dictated by the level of security required by the system provider. One option is for the system provider to act as their own CA: they provide the certificate to the user and maintain the set of valid user certificates. Another option is to partner with a recognized CA and outsource the verification of user identity, issuing of certificates, and maintenance of the revocation list to them. CAs will offer different levels of user identity verification, from a simple check of online identity through to a face-to-face identity verification.

3.9. Example Resolved

The commercial organization implements a certificate-based KNOWN PARTNERS mechanism. It obtains a certificate from a recognized CA which it uses to set up an SSL-based SECURE CHANNELS. All access to restricted functionality must take place over that SECURE CHANNELS.

The organization decides to act as its own CA because it already has a lot of face-to-face interaction with its business partners. Each business partner that requires access to the on-line functionality is issued an individual certificate signed by the organization. When the user accesses the restricted functionality, they are required to provide the certificate, which the system then checks against its own revocation list.

At the culmination of an interaction such as the confirmation of order placement, the individual user ID embedded in the certificate is passed with the order details to the corporate ordering facility.

3.10. Consequences

The following benefits may be expected from applying this pattern:

- Security is improved, because the system can be sure that any user accessing the system is who it thinks they are.
- User confidence is improved because they can be sure they are not accessing a ‘spoof’ system.

The following potential liabilities may arise from applying this pattern:

- Performance is slightly impacted, because exchanging and verifying system and user identities introduces overhead in processing a user’s request.
- Availability is potentially impacted, because the user identity verification service becomes a single point of failure for access to restricted functionality.
- Manageability is impacted, because system and user identities must be actively managed to maintain the required level of security.
- KNOWN PARTNERS is significantly more expensive to implement and maintain than a lightweight mechanism based on passwords.

3.11. Variants

Multi-part user identity. The use of digital certificates actually ties the interaction to a browser on a machine rather than to an individual user. This is advantageous if we want to allow multiple users to act on behalf of a business partner and we don’t care which individual but is a liability if we want to identify individual users. A common variant of certificate-based user identification is the addition of a password or PIN individual to each user, that must be supplied at the same time as the certificate. Multi-part user identities are also useful in the case of machine theft, as possession of the certificate alone is not sufficient to access the restricted functionality of the system.

Hardware token. Rather than using certificates for user identification, a hardware ‘token’ is issued to each user. The token usually provides a key that changes frequently and must be provided to the system on log in—either the key is displayed, and the user types it in, or the hardware token is physically connected to the machine and provides the key automatically. Hardware-token based systems also frequently use a multi-part user identity, as theft of the token is usually easier than theft of the client machine, and less readily noticed by the user.

3.12. Known Uses

KNOWN PARTNERS mechanisms are becoming increasingly common for commercially sensitive or high-value online interactions. The authors have worked with several companies that implement a certificate-based KNOWN PARTNERS scheme to provide access to ‘extranet systems’ as well as internal resources such as document and code repositories. The UK government also uses a certificate-based scheme for its ‘government gateway’ (http://www.gateway.gov.uk/), which provides access to functionality such as on-line filing of business tax returns.

3.13. See Also

3.14. References

[1] Dyson, P., & Longshaw, A. (2004). *Architecting enterprise solutions: patterns for high-capability internet-based systems*. John Wiley & Sons.

3.15. Source

Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., & Sommerlad, P. (2006). Security Patterns: Integrating Security and Systems Engineering (1st ed.). Wiley.

4. Non-Repudiation Requirements

4.1. Intent

A non-repudiation service must satisfy a set of requirements for both the service and the quality of service. The function of non-repudiation is to capture and maintain evidence so that the participants of a transaction or interaction cannot deny having participated in that activity. While each situation that calls for non-repudiation is unique, there are common generic requirements that apply to all non-repudiation situations. This pattern provides a common generic set of non-repudiation requirements. The pattern also helps you to apply the general requirements to your specific situation and helps you to determine the relative importance of conflicting requirements.

4.2. Example

The museum seeks to increase the publicity of its new wing for gemstones. To do this, the museum seeks to have many exotic gems on display for the grand opening. The Crown Jewels of England are scheduled to be a part of the display. Manuela the museum manager would like to have a high degree of confidence that the receipt of the jewels by the museum and the release of the jewels after the opening are protected. Samuel the museum system engineer needs to specify the requirements for nonrepudiation and the relative importance of those requirements, as a means of driving and evaluating a non-repudiation service that will support events such as this grand opening. How can Samuel define such a set of requirements?

4.3. Context

Accounting requirements and their relative importance are understood, for example, from applying SECURITY ACCOUNTING REQUIREMENTS. The planned uses of non-repudiation are understood. A common transaction type is the sending and receiving of materials such as merchandise or contracts. Non-repudiation is used to prevent the receiver from denying that they received the materials when in fact they did receive them. Sometimes non-repudiation is used to prevent the sender from claiming that they sent the materials when in fact they did not send them.

4.4. Problem

Non-repudiation is a security service that captures and maintains evidence so that the participants of a transaction or interaction cannot deny having participated in that activity. The need is to identify the common requirements that drive the design of this service. The model in figure 6 places non-repudiation in the context of the identity of the participants, in terms of the activity that it collects, and the facts and evidence that it provides. This ensures that the participants cannot deny having engaged in the activity. Non-repudiation needs information about the event that will disallow the participants from denying their participation. If the participants are allowed to deny their involvement in the activity, then the integrity of the activity will be jeopardized, and other participants may suffer negative consequences. For example, if a purchaser receives a book that they ordered from Amazon.com, and then denies receiving it, Amazon may need to send another copy of the book, which is a financial loss to them.

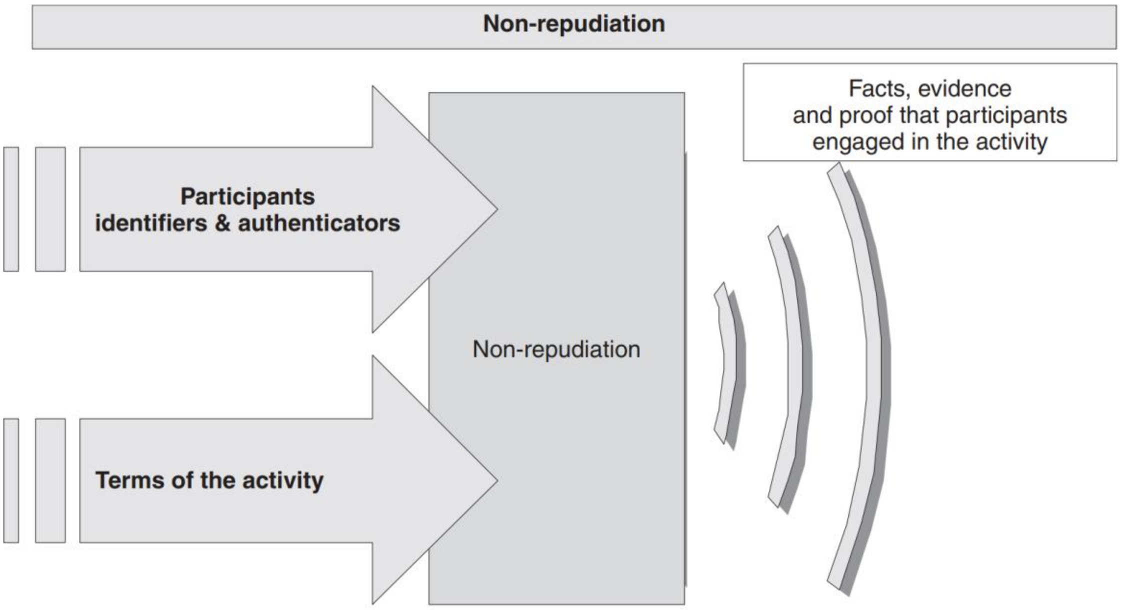
Non-repudiation
Facts, evidence and proof that participants engaged in the activity
Participants identifiers & authenticators
Non-repudiation
Terms of the activity

Figure 6: Common model for non-repudiation

How can specific requirements for a non-repudiation service, and their relative importance, be determined?

The process of selecting and prioritizing non-repudiation requirements needs to balance the following forces:

- You can use non-repudiation to help achieve the desired security properties, especially integrity and accountability.
- Obtaining evidence that a person or organization participated in a transaction can have significant benefits in cases in which they deny participation, including favorable resolution of both economic and legal disputes.
- Applying non-repudiation has associated costs, including the time and resources required for continuously capturing identifiers and authenticators and explicitly defining the terms of an event. This is counter to the organization goal of minimizing total costs.
- High need for non-repudiation often involves intrusive or inconvenient constraints on participants.
- There may be legal constraints that mandate that participants have access to the facts and evidence of their activities.
- The elements of the non-repudiation service need protection if the service is to perform its function.

4.5. Solution

Specify a set of non-repudiation requirements for a specific domain such as a system or organization and determine the relative importance of each requirement. The solution has two aspects: a requirements process and a common set of generic requirements.

4.5.1. Requirements Specification and Prioritization Process

A system requirements engineer, in conjunction with an enterprise architect, typically perform the requirements process. An important first step is explicitly to define the domain for which non-repudiation requirements are being specified, such as a specific system or facility. You also define factors such as organization constraints that affect specialization and importance of requirements. You then specify nonrepudiation requirements for the target domain, using the generic requirements provided below. The final activity is to define the relative importance of the specified requirements.

4.5.2. Generic Non-repudiation Requirements

The following is a general set of requirements appropriate to non-repudiation services. An engineer will need to consider each of these and determine its priority based on criteria specific to the target domain, as well as on broader organization constraints. Additional requirements may be added to this list to address system-unique characteristics. Some of the general requirements represent non-repudiation functional requirements. The remaining requirements represent non-repudiation nonfunctional requirements, including requirements for security of the non-repudiation service.

- Provide information that an actor took specified actions in an activity or event. Non-repudiation needs to have the ability to form strong links between the participants who engage in an activity and the activity itself. The evidence and facts that are derived from capturing information about the event need to be explicit and detailed enough to help assign accountability. This requirement has increased priority when the events are of high importance. 'High' importance may mean critical to business functions or operations, providing legal or financial evidence, or otherwise significant.
- Provide identifiers, authenticators and the terms of an event when requested. Non-repudiation should examine any legal or external considerations regarding the gathering of information about participants of an event. There may be consequences for the organization if laws are not followed regarding the collection of identifiers and authenticators.
- Minimize the time it takes participants to provide their identifiers and authenticators. You need to consider that if events require non-repudiation and the events must happen for other business reasons, participants in these events should not be discouraged from joining due to complexities associated with identifiers and authenticators.
- Protect all non-repudiation information associated with an event. The confidentiality, integrity, and availability of facts and evidence need to be maintained. Due to the need to help to assign accountability, it is imperative that the information gathered by the non-repudiation service be uncorrupted. The better the non-repudiation service can maintain and provide a degree of confidence about the protection of the information, the more the service user can rely on the information that it provides. Non-repudiation also needs to verify that the information that it collects is not forged or misrepresented.

An additional set of requirements applies to all service requirements patterns. Instead of duplicating the discussion of the same set in each requirements pattern, they are simply listed here, because they do need to be considered in each requirements pattern. The requirements are: minimize mismatch with user characteristics, risks to user safety, costs of per-user set-up, costs of maintenance, management, and overhead, and changes needed to existing system infrastructure. Further discussion of each of these cross-cutting requirements, including implementation factors, is given in I&A REQUIREMENTS.

4.6. Structure

4.7. Dynamics

4.8. Implementation

This section provides more detail about the process that was summarized in the Solution section. The requirements process typically includes these steps:

1. Establish the domain for which the non-repudiation service is needed. Ensure that the domain has been identified and scoped: typical non-repudiation domains include categories of transactions or interactions. For example, transactions at a company's public Web portal may be a different domain from transactions involving contracts with suppliers. Other constraints or distinctions may bound the domain as well, such as separating transactions that occur outside the organization from internal transactions.
2. Specify a set of factors that affect the specialization and importance of requirements. Factors can include uses of non-repudiation, non-repudiation needs, organization constraints, and priorities.
3. Specify non-repudiation requirements for the target domain. Specialize the set of generic requirements given above.
4. Define the relative importance of specific requirements. Priority is increased when the transactions or their consequences are of high importance. 'High' importance may mean critical to business functions or operations, providing legal or financial evidence, or otherwise significant.

4.9. Example Resolved

Samuel the museum system engineer defines the domain for non-repudiation to be transactions in which the museum lends or borrows gems of high value. Borrowing the Crown Jewels for an exhibit is an example of a transaction in this domain. To ensure protection of the reception and dispatch of the Crown Jewels, the museum defines specific non-repudiation requirements. Table 3 shows the specific requirements and relates them to the general requirements defined in the Solution section.

Although many aspects of this exchange will be time-consuming, it will also provide a very high degree of confidence that the parties exchanged the Crown Jewels and that the Crown Jewels were returned in the same condition as that in which they were received.

General Requirement	Specific Requirement for this Transaction
Provide information that an actor took specified actions in an activity or event	Capture, store, and record the receipt and return of the Crown Jewels by videotaping the event or having it done with witnesses from both the sender and the receiver.
Provide identifiers, authenticators and the terms of an event when requested	• Identify and authenticate the individual(s) from whom the Crown Jewels should be received and to whom they should be given after the opening. • Explicitly outline the terms of the exchange and have all participants provide an authenticated signature. • Provide copies of this agreement to the sender and the receiver.
Minimize the time it takes participants to provide their identifiers and authenticators	Prepare everything, including videotaping preparations and writing down the agreement, to make it as efficient and unobtrusive as possible. Document the process and have standard forms available for use in similar transactions.
Protect all non-repudiation information associated with an event	Store this agreement, the signatures, and the videotape in a secure location.

Table 3: Museum specific requirements for non-repudiation

4.10. Consequences

The following benefits may be expected from applying this pattern:

- It facilitates conscious selection of non-repudiation requirements, so that decisions about selecting non-repudiation mechanisms have a clear basis, rather than occurring in a vacuum.
- It promotes explicit analysis of trade-offs that encourages balancing and prioritizing of conflicting requirements. This helps to avoid stronger than necessary non-repudiation which places increased burden on the parties to a transaction, and at the same time it helps to avoid weaker than necessary non-repudiation, which would make it easy to deny participation.
- It results in documentation of non-repudiation requirements which communicates to all interested parties and also provides information for security audits.
- The pattern fosters a clear connection of non-repudiation requirements to security accounting policies. This also encourages organizations to make their policies more explicit.

The following potential liabilities may arise from applying this pattern:

- It requires an investment of resources to apply the pattern, including time to analyze domains and non-repudiation needs. In some cases, the cost of applying the pattern may exceed its benefits.
- It poses a danger of over-engineering and complexity creep if stakeholders are offered too many options. You can mitigate this by using the requirements only as guidelines for analysis, or by selecting those parts of the pattern that give the most help.

- The formal selection process may be too long and costly and produce too much overhead. You can mitigate this in the same ways as noted above.
- Specific circumstances might not be covered by generic non-repudiation requirements. You can mitigate this by adding specific requirements and including them in the trade-offs.
- Documentation of requirements implies that they must be maintained as they change over time. You can mitigate this by keeping the requirements in a form that is easy to update, integrated with other system documentation.

4.11. Known Uses

The general non-repudiation requirements and the process of specifying non-repudiation requirements described in this pattern are widely known, but are generally used informally, as opposed to being codified or published. The requirements as stated in this pattern represent a consolidation of MITRE Corporation's experience in working with multiple customers over several decades. However, some publications on non-repudiation requirements exist.

- [1] discusses non-repudiation as one of the primary safeguards, in the context of integrity.
- [2] is an international standard that defines evaluation criteria for information technology security. It includes non-repudiation requirements in the context of communication.
- [3] is an international standard on non-repudiation.
- [4] discusses non-repudiation protocol guidelines and stresses the need to match protocols with requirements.
- [5] discusses requirements for non-repudiation in the context of the Internet.
- [Gindin01] discusses technical requirements for non-repudiation, in contrast with legal requirements.

4.12. See Also

4.13. References

[1] "Technical Report ISO13335-4 Information Technology - Guidelines for the Management of IT Security - Part 4: Selection of Safeguards" International Organization for Standardization, First Edition, 2000

[2] International Organization for Standardization. (1999). Common Criteria for Information Technology Security Evaluation (Version 2.1). CCIMB-99- 031. ISO/IEC JTC 1 adopted CC 2.0 with minor modifications in June 1999 as ISO/IEC 15408, Version 2.1

[3] "ISO/IEC 13888:1997 Information Technology – Security Techniques – Non-repudiation" International Organization for Standardization, 1997.

[4] Louridas, P. (2000). Some guidelines for non-repudiation protocols. *ACM SIGCOMM Computer Communication Review*, *30*(5), 29-38.

[5] IETF. (1999, September). Internet Draft on Non-Repudiation Requirements.

[6] Gindin, T. (2001, January). Requirements for Technical Non-Repudiation. http://www.ietf.org/proceedings/00dec/slides/PKIX-1/

4.14. Source

Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., & Sommerlad, P. (2006). Security Patterns: Integrating Security and Systems Engineering (1st ed.). Wiley.

5. Packet Filter Firewall

5.1. Intent

Some of the hosts in other networks may try to attack the local network through their IP-level payloads. These payloads may include viruses or application-specific attacks. We need to identify and block those hosts. A packet filter firewall filters incoming and outgoing network traffic in a computer system based on packet inspection at the IP level.

5.2. Example

Our system has been attacked recently by a variety of hackers, including somebody who penetrated our operating system and stole our clients' credit card numbers. Our employees are wasting time at work by looking at inappropriate sites on the Internet. If we continue like this, we will soon be out of business.

5.3. Context

Computer systems on a local network connected to the Internet and to other networks with different levels of trust. A host in a local network receives and sends traffic to other networks. This traffic has several layers or levels. The most basic level is the IP level, made up of packets consisting of headers and bodies (payloads). The headers include the source and destination addresses as well as other routing information, while the bodies include the message payloads.

5.4. Problem

Some of the hosts on other networks may try to attack the local network through their IP-level payloads. These payloads may include viruses or application-specific attacks. How can we identify and block those hosts?

The solution to this problem must resolve the following forces:

- We need to communicate with other networks, so isolating our network is not an option. However, we do not want to take a high risk for doing so.
- The protection mechanism should be able to precisely reflect the security policies of the organization. A too coarse defense may not be useful.
- Any protection mechanism should be transparent to the users. Users should not need to perform special actions to be secure.
- The cost and overhead of the protection mechanism should be relatively low or the system may become too expensive to run.
- Network administrators deploy and configure a variety of protection mechanisms; hence it is important to have a clear model of what is being protected.
- The attacks are constantly changing; hence it should be easy to make changes to the configuration of the protection mechanism.
- It may be necessary to log input and/or output requests for auditing and defense purposes.

5.5. Solution

A PACKET FILTER FIREWALL intercepts all traffic coming and going from a port P and inspects its packets (see figure 7). Those coming from or going to mistrusted addresses are rejected. The mistrusted addresses are determined from a set of rules that implement the security policies of the organization. A client from another network can only access the Local Host if a rule exists authorizing traffic from its address. Specific rules may indicate an address or a range of addresses. Rules may be positive (allow traffic from some address) or negative (block traffic from some address). Most commercial products order these rules for efficiency in checking. Additionally, if a request is not satisfied by any of the explicit rules, then a default rule is applied.

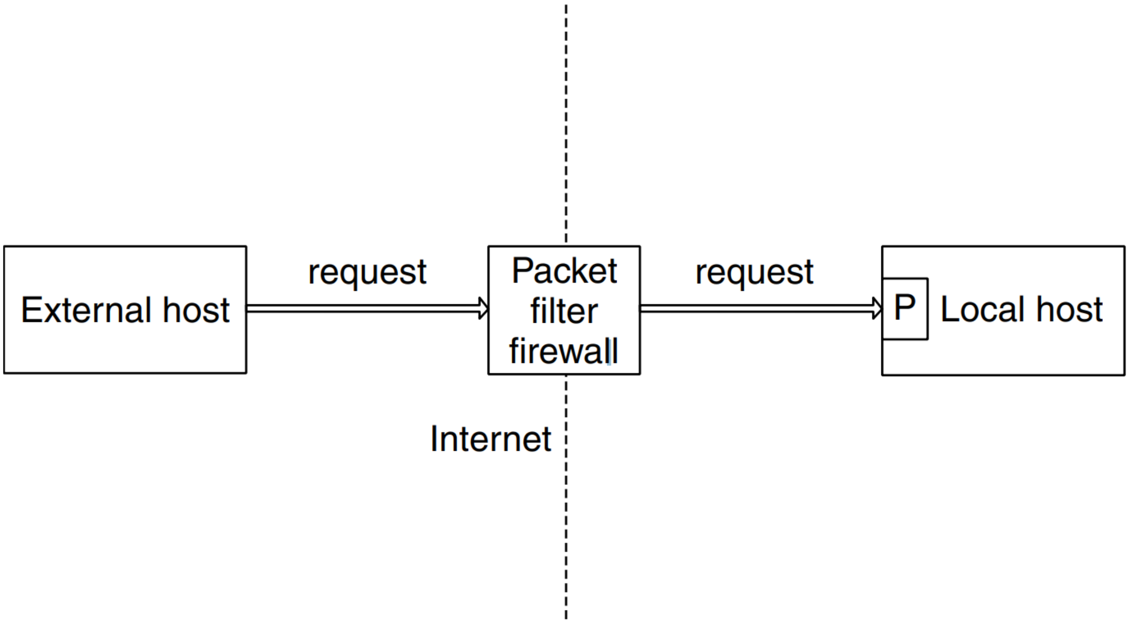

Figure 7: The concept of the packet filter firewall

5.6. Structure

Figure 8 shows an external host requesting access to a local host (a server) through a packet filter firewall. The organization policies are embodied in the objects of class Rule collected by the RuleBase. The RuleBase includes data structures and operations to manage rules in a convenient way. The rules in this set are ordered and can be explicit or default.

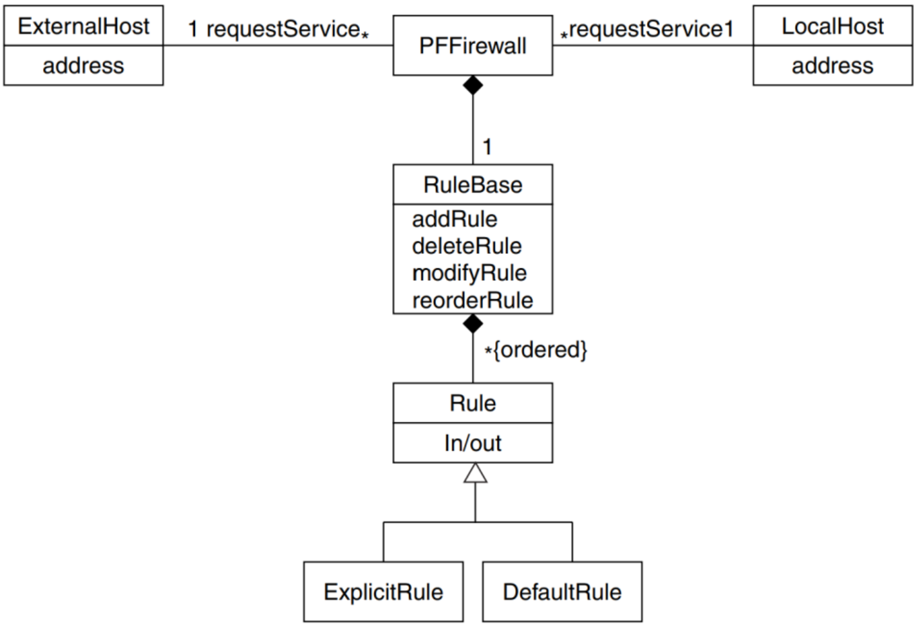

Figure 8: Class diagram for PACKET FILTER FIREWALL

5.7. Dynamics

We describe the dynamic aspects of the PACKET FILTER FIREWALL using a sequence diagram for one of its basic use cases. There is a symmetric use case, filtering an outgoing request, which we omit for briefness. We also omit use cases for adding, removing, or reordering rules because they are straightforward. See figure 9.

Use Case: Filtering a Client's Request

Summary A host in a remote network wants access to a local host to either transfer or retrieve information. The access request is made through the firewall, which according to its set of rules determines whether to accept or deny the request—that is, it filters the access request.

Actors A host on an external network (client).

Precondition An existing set of rules to filter the request must be in place in the firewall.

Description

1. An external host requests access to the local host.
2. A firewall filters the request according to a set of ordered rules. If none of the explicit rules in the rule set allows or denies the request, a default rule is used for deciding.
3. If the request is accepted, the firewall allows access to the local host.

Alternate Flow

- The request is denied.

Postcondition The firewall has accepted the access of a trustworthy client to the local host.

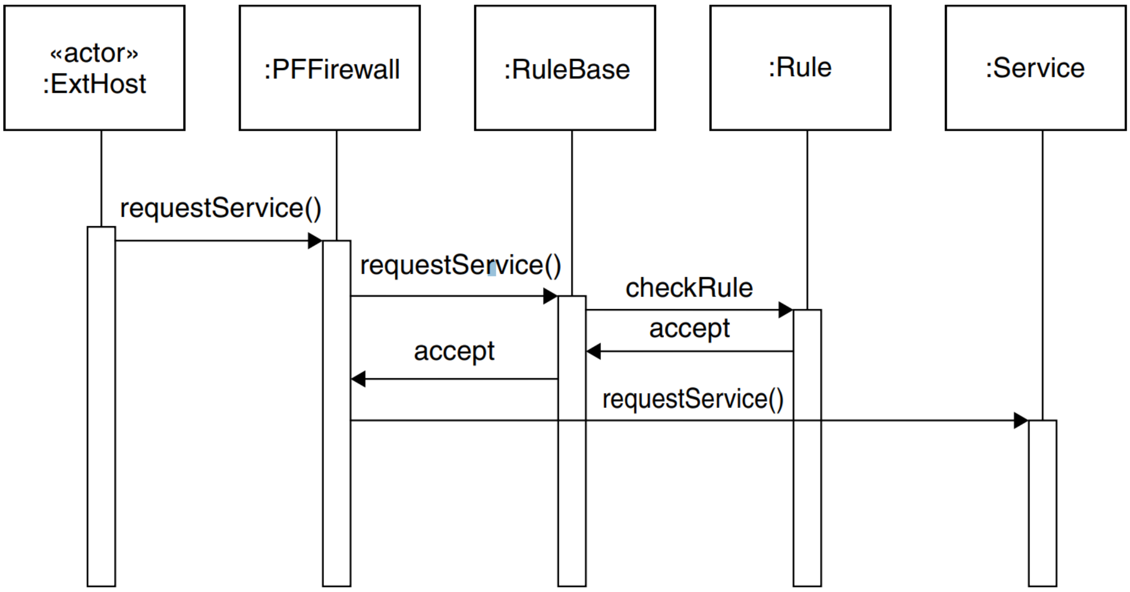

Figure 9: Sequence diagram for filtering a client's request

5.8. Implementation

1. Define an organization policy about network access, classifying sites according to our trust in them.
2. Convert this policy into a set of access rules. This can be done manually, which may be complex for large systems. An alternative is using an appropriate commercial product, such as Solsoft [1].
3. Note that the idea of a single point of access is virtual: there may be several physical firewalls deployed at different places. This means that it is necessary to install firewalls at all external boundaries, such as routers or gateways.
4. Write the rules in each firewall. Again, products such as Solsoft and others automatically propagate the rules to each registered firewall.
5. Configure the corresponding firewalls according to standard architectures. A common deployment architecture is the DEMILITARIZED ZONE (DMZ).

5.9. Example Resolved

We were able to trace the addresses of our attackers and we installed a firewall to block requests from those addresses from reaching our system. We also made a list of addresses of inappropriate sites and blocked access to them from the hosts in our network. All this reduced the number of attacks and helped control the behavior of some employees.

5.10. Consequences

The following benefits may be expected from applying this pattern:

- A firewall transparently filters all the traffic that passes through it, thus lowering the risk of communicating with potentially hostile networks.
- It is possible to express the organization's filtering policies through its filtering rules, with different levels of protection for different parts of the network.

- It is easy to update the rule set to counter new threats.
- Because it intercepts all requests, a firewall allows systematic logging of incoming and outgoing messages. Because of this, a firewall facilitates the detection of possible attacks and helps to hold local users responsible for their actions when interacting with external networks.
- Its low cost enabled it to be included as part of many operating systems and simple network devices such as routers.
- It offers good performance, only needing to look at the headers of IP packets rather than the complete packet.
- It can be combined with intrusion detection systems (IDS) for greater effectiveness. In this case, the IDS can tell the firewall to block suspicious traffic.

The following potential liabilities may arise from applying this pattern:

- The firewall's effectiveness and speed may be limited due to its rule set (order of precedence). Addition of new rules may interfere with existing rules in the rule set, so a careful approach should be taken in adding and updating access rules.
- The firewall can only enforce security policies on traffic that goes through the firewall. This means that one must make changes to the network to ensure that there are no other paths into its hosts.
- An IP-level firewall cannot stop attacks coming through the higher levels of the network. For example, a hacker could put malicious commands or data in header data not used for routing, or in the payload.
- Each packet is analyzed independently, which means that it is necessary to analyze every packet. This may reduce performance.
- A packet filter cannot recognize forged addresses (IP spoofing) because it only examines the header of the IP packet. This can be corrected (at some extra cost) using link layer filtering, in which each IP address is correlated to its hardware address [2].

5.11. Known Uses

This model corresponds to an architecture that is seen in commercial firewall products, such as ARGuE (Advanced Research Guard for Experimentation), which is based on Network Associates' Gauntlet Firewall [3], OpenBSD Packet Filtering Firewall [4], which is the basic firewall architecture for the Berkeley Software Distribution system, and the Linux Firewall [5], which is the basic firewall architecture used with the Linux operating system. PACKET FILTER FIREWALL is used as an underlying architecture for other types of firewalls that include more advanced features.

5.12. See Also

AUTHORIZATION defines the standard security model for PACKET FILTER FIREWALL. This pattern is also a special case of SINGLE ACCESS POINT and is the basis for other, more complex, types of firewalls. DEMILITARIZED ZONE (DMZ) defines a way to configure this pattern in a network. This pattern can also be combined with STATEFUL FIREWALL.

5.13. References

[1] Solsoft. (n.d.). Inc: http://www.solsoft.com

[2] Frantzen, M., Kerschbaum, F., Schultz, E. E., & Fahmy, S. (2001). A framework for understanding vulnerabilities in firewalls using a dataflow model of firewall internals. *Computers & Security*, *20*(3), 263-270.

[3] Epstein, J. (1999, December). Architecture and concepts of the ARGuE guard. In *Proceedings 15th Annual Computer Security Applications Conference (ACSAC'99)* (pp. 45-54). IEEE.

[4] Rustad, R. E. (2002, November). Guide to OpenBSD Packet Filtering Firewalls. Kuro5hin article. http://www.kuro5hin.org/story/2002/11/23/14927/477

[5] Ziegler, R. L., & Constantine, C. B. (2002, March). *Linux firewalls:* Packet Filtering. Sams Publishing.

5.14. Source

Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., & Sommerlad, P. (2006). Security Patterns: Integrating Security and Systems Engineering (1st ed.). Wiley.

6. Protection Reverse Proxy

6.1. Intent

Putting a Web server or an application server directly on the Internet gives attackers direct access to any vulnerabilities of the underlying platform (application, Web server, libraries, operating system). However, to provide a useful service to Internet users, access to your server is required. A packet filter firewall shields your server from attacks at the network level. In addition, a PROTECTION REVERSE PROXY protects the server software at the level of the application protocol.

6.2. Example

You are running your Web site using a major software vendor's Web server software. Your Web site uses this vendor's proprietary extensions to implement dynamic content for your visitors, and you have invested heavily in your Web site's software. Your server is protected by a PACKET FILTER FIREWALL.

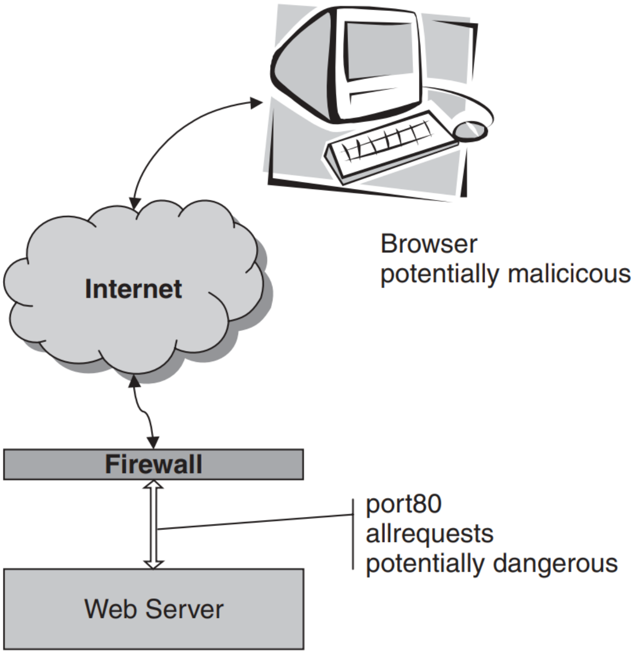

Figure 10: Protection using a PACKET FILTER FIREWALL

You must open this firewall to allow access to the public port (80) of your Web server. Attacks from the Internet that exploit vulnerabilities of your server software frequently burden your system administrator with patch installation. Switching to another vendor's Web server is not possible, because of the existing investment in the Web server platform, its content, and your own software extensions. In addition, with every new patch you install, you run the risk of destabilizing your configuration so that your system and software extensions cease to work. How can you escape the dilemma and keeping your Web site up without compromising its security?

6.3. Context

Any kind of service accessible through the Internet or a through another potentially hostile network environment. Usually, the access protocol is HTTP or HTTPS.

6.4. Problem

Even if you install a simple PACKET FILTER FIREWALL your Web server can remain vulnerable to attacks that exploit weaknesses in its protocol implementation. How can you protect your Web server infrastructure in the light of its potential vulnerability to attacks using its protocol?

The solution to this problem must resolve the following forces:

- A simple packet filter firewall is not enough to protect your Web server, because access to its protocol (for example port 80) must be provided to the Internet.
- Attack scenarios often employ extra-long or extra crafted request parameters to exploit buffer overflows. Most firewalls work at the network packet level and cannot detect attacks using such invalid requests.
- Hardening your Web server might be beyond your capabilities, for example because it comes as a black box from your vendor, or because it is too complex.
- Installing patches to your Web server platform helps avoid exploitation of known vulnerabilities. But with each patch, you risk your system extensions ceasing to work. You need to re-run your integration tests at each patch level and might need to keep your extensions up to date with each patch level. It might even be impossible to upgrade your Web server in a timely manner because the extensions aren’t ready.
- Switching to another Web server software by a different source is also potentially expensive, risky and time consuming. A new Web server might have fewer vulnerabilities, but you are less familiar with it. In addition, it might also require you to adapt your own system extensions.
- You cannot know about vulnerabilities that might be detected in the future.

6.5. Solution

Change your network topology to use a PROTECTION REVERSE PROXY that shields your real Web server. Configure this reverse proxy to filter all requests, so that only (mostly) harmless requests will reach the real Web server. Two PACKET FILTER FIREWALLs ensure that no external network traffic reaches the real Web server. The resulting network topology provides a DEMILITARIZED ZONE containing only the reverse proxy machine, and a secured server zone containing the Web server.

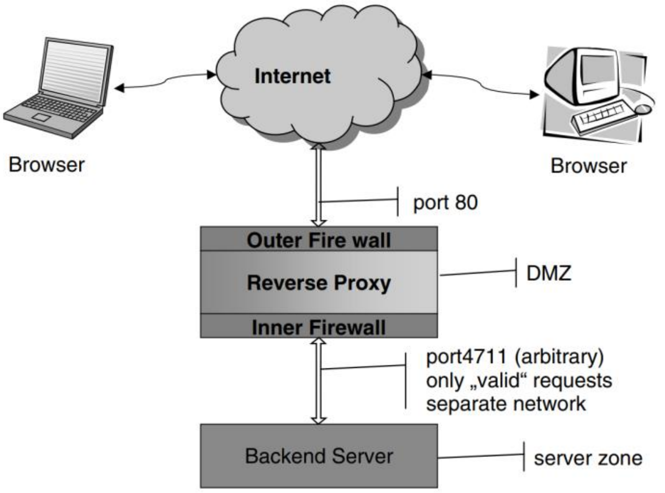

Figure 11: Protection using a PROTECTION REVERSE PROXY

Although this solution discusses only Web servers, it applies to other protocols like FTP, IMAP, and SMTP as well. A PROTECTION REVERSE PROXY for FTP, for example, might scan files for viruses or executable content and prohibit upload of such files, or limit the available FTP commands and prohibit third-party host data connections, which are allowed by the FTP standard.

6.6. Structure

The class diagram for this pattern is shown below:

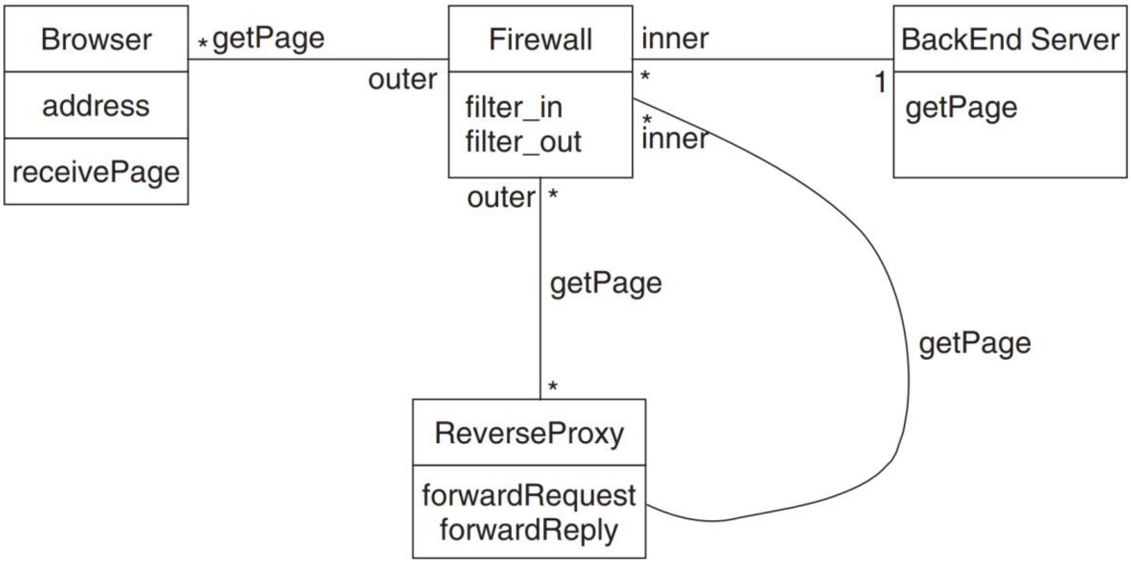

Figure 12: Class diagram for PROTECTION REVERSE PROXY

6.7. Dynamics

The first scenario shows how a valid request is checked and passed on by the protection reverse proxy. The inner and outer firewall components are assumed to be transparent in that case and thus are not shown. Post processing a back-end server’s reply is optional, but can be used to adjust protocol header fields, for example. Note that the access log is only written after the reply was sent, to improve the responsiveness of the system.

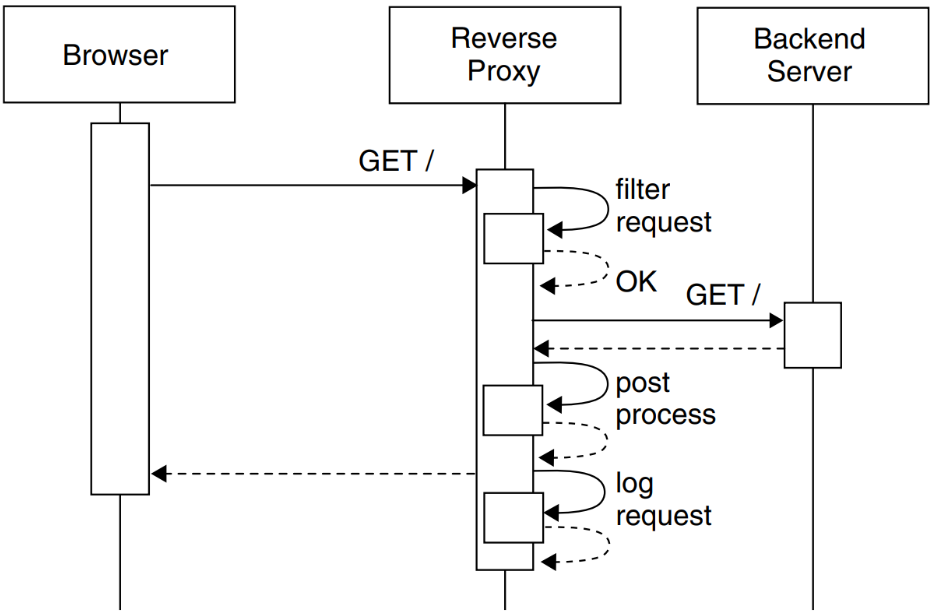

Figure 13: Allowing a client request in PROTECTION REVERSE PROXY

The second scenario demonstrates the blocking mechanism of the protection reverse proxy by ignoring an invalid and thus potentially malicious request. Nevertheless, even though the browser will not get an answer, the attempt will be logged. According to the official hypertext transfer protocol, the reverse proxy should return an error code (typically '403 forbidden' or '404 not found'). It depends on your security policy whether you give an error reply, or silently ignore the attempt and close the connection at the Protection Reverse Proxy. See figure 16.

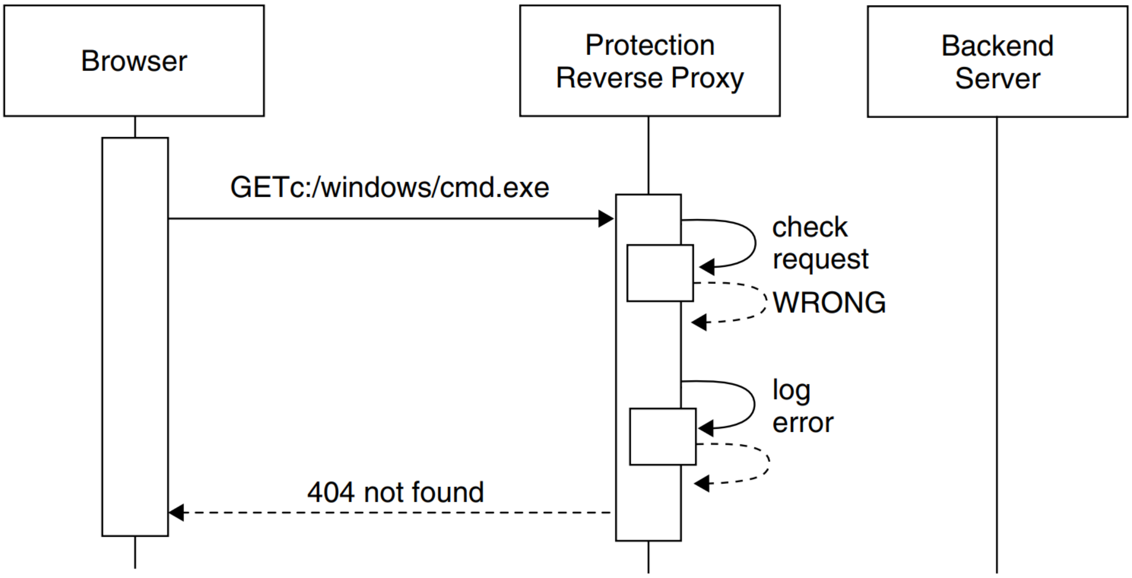

Figure 14: Refusing a client request in PROTECTION REVERSE PROXY

6.8. Implementation

To implement this pattern, several tasks need to be carried out:

1. Plan your firewall and network configuration. Even if the firewall update is done after every other part is in place, it is good to start with a plan, so that configuration of the other components can rely on the firewall plan. Often the concrete configuration needs to consider more than just one protocol, and some explicit ‘holes’ in your firewall may be needed. Find out what protocol your reverse proxy solution needs to support. Typically, only HTTP (port 80) is needed, but you might want to allow other protocols through your reverse proxy as well.
2. Select a reverse proxy platform. You might create your own reverse proxy, for example by configuring the Apache Web server with mod_rewrite and mod_proxy modules—several vendors offer professional reverse proxy solutions—or you might need to implement your own reverse proxy, for example because you are using a special protocol not supported by other solutions.

   Showing the implementation details of your own reverse proxy server software is beyond the scope of this pattern. Nevertheless, there are cases in which you might not trust a solution provider, or not have one and rely on your own skills.

   When selecting a vendor or source for your protection reverse proxy, you should opt for a simple and proven solution. For example, by using Apache you risk all the Apache Web server vulnerabilities being present in your protection reverse proxy. On the other hand, the Apache Web server is deployed so often that most vulnerabilities and countermeasures are known.
3. Configure your back-end Web server(s). The Web content should rely on relative path names and not use its internal name or IP address to refer to itself. Otherwise, links might not work, because the browser can no longer access the machine it is running on directly.
4. Configure your protection reverse proxy. For the security to work you need to define which requests should be mapped to your back-end Web server and define what should happen if invalid requests occur. For example, you might want to log requests that were denied by the reverse proxy. There are two approaches to request filtering: ‘blacklists’ and ‘whitelists.’
   a. A blacklist filter only blocks requests that its list of malicious requests knows of but passes all others. Blacklist filters are easier to deploy, but riskier. They are often used by ‘higher-level’ firewalls.
   b. A whitelist filter is more restrictive and only lists allowed requests. It needs to be configured with detailed knowledge of the back-end server and allowed URLs. A whitelist filter needs to be adapted every time your backend server changes significantly in its URL space. Nevertheless, it is the better choice for a PROTECTION REVERSE PROXY.

   If your back-end server relies on redirects or other mechanisms that use its host address and you cannot change that, you need to configure your reverse proxy to modify server responses accordingly.
5. Deploy everything. Initial deployment, setting up firewalls, network and routers, host IP addresses, and so on requires good planning. If you have a system up and running already, this re-configuration might mean some service interruption. Nevertheless, later changes to the topology need only consider the reverse proxy and eventually the inner firewall.

6.9. Example Resolved

Following these implementation guidelines, we are able to protect our vulnerable Web server with a PROTECTION REVERSE PROXY.

6.10. Consequences

The following benefits may be expected from applying this pattern:

- Attackers can no longer exploit vulnerabilities of the back-end server directly. Even when the back-end server is compromised, the firewalls hinder further spreading of Internet worms and so on by blocking outgoing requests from the back-end server.
- Even with known vulnerabilities, you might be able to keep your Web server configuration stable, because the PROTECTION REVERSE PROXY, with its request filtering capability, can prohibit exploitation of the Web server's vulnerabilities.
- Easier patch administration. Only one machine remains connected to the Internet directly, needs to be monitored for potential vulnerabilities, and have existing patches applied. However, you cannot blindly trust your PROTECTION REVERSE PROXY. A back-end server still needs to be configured with your brain switched on, to avoid exploitation of vulnerabilities with 'allowed' requests.
- More benefits apply when combined with more functionality—see INTEGRATION REVERSE PROXY and FRONT DOOR.

The following potential liabilities may arise from applying this pattern:

- Blacklist filtering can give you a false sense of security. Like patches, blacklists can only be constructed after a vulnerability is known.
- Whitelist filtering can be fragile when back-end servers change. Adding functionality, or rearranging content structure on the back-end Web server, can imply additional work to re-configure the whitelist filter of the PROTECTION REVERSE PROXY.
- Latency. A reverse proxy adds latency to the communication, not only because of the additional network traffic, but also because of the filtering and validation of requests.
- Some loss of transparency: some restrictions are imposed on the back-end servers. However, these are typically good practice anyway, such as relative paths in URLs. Nevertheless, the back-end servers no longer see the communication end partner directly at the network level. The protocol may therefore need to provide a means of identifying the original communication end point (which HTTP allows).
- An additional point of failure. If the reverse proxy stops working, access to your Web site is impossible. Any additional component that can fail increases the overall risk of system failure. To reduce this risk, you can provide a hot or cold stand-by installation with hardware or software fail-over switches.
- Hardware, software, and configuration overhead. PROTECTION REVERSE PROXY requires you to configure an additional packet filter firewall, as well as another machine to run the reverse proxy on.

6.11. Variants

INTEGRATION REVERSE PROXY and FRONT DOOR can (and should) be combined in their function with PROTECTION REVERSE PROXY, and thus vary this pattern by adding functionality.

6.12. Known Uses

PROTECTION REVERSE PROXY are popular. Some organizations in the financial industry have as a guideline to use a reverse proxy for every protocol provided over the Internet (with some exceptions, such as DNS). They can thus ensure that a vulnerable server is never directly accessible from the 'wild.'

Vendors of security infrastructure provide PROTECTION REVERSE PROXIES as part of their broader infrastructure. Examples of such infrastructures are Bull Evidian's Access Master and PortalXpert [1] as well as IBM's Tivoli Access Manager [2].

6.13. See Also

PROTECTION REVERSE PROXY is a special implementation of SINGLE ACCESS POINT.

In conjunction with regular firewalls, PROTECTION REVERSE PROXY builds on the principle of 'defence in depth'—see [3].

6.14. References

[1] Evidian. (n.d.). Product Description and White Papers. http://www.evidian.com

[2] IBM. (n.d.). Enterprise Security Architecture using IBM Tivoli Security Solutions. http://www.redbooks.ibm.com/pubs/pdfs/redbooks/sg246014.pdf

[3] Schneier, B. (2003). *Beyond fear: Thinking sensibly about security in an uncertain world* (Vol. 10). New York: Copernicus books.

6.15. Source

Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., & Sommerlad, P. (2006). Security Patterns: Integrating Security and Systems Engineering (1st ed.). Wiley.

7. Proxy-Based Firewall

7.1. Intent

A proxy-based firewall inspects, and filters incoming and outgoing network traffic based on the type of application service to be accessed or performing the access. This pattern interposes a proxy between the request and the access and applies controls through this proxy. This is usually done in addition to the normal filtering based on addresses.

7.2. Example

After we started using a PACKET FILTER FIREWALL most of our problems were reduced. However, some of the messages sent from sites we don't consider suspicious contain malicious payloads, because hackers were spoofing trusted addresses. These payloads sometimes contained incorrect commands or the wrong type and length of parameters. Our PACKET FILTER FIREWALL cannot stop these attacks because it doesn't look at the message payload, and as a result we are experiencing new problems. It is also hard to block every malicious site.

7.3. Context

Computer systems on a local network connected to the Internet and to other networks, where a higher level of security than the one provided by packet filters is needed. Specifically, we want to control attacks at the application layer of the network protocol. Incorrect commands or parameters can produce buffer overflows and other conditions that can be exploited for attacks. In some cases, we might also want to authenticate the client to avoid spoofing. Outgoing flows (to malicious sites) can also be damaging in this environment.

7.4. Problem

PACKET FILTER FIREWALL only inspects the network addresses when deciding whether to allow access for a request. We can only block supposedly malicious sites. It is hard to know about all of those sites, and we need further defenses. Also, how do we protect our network from potential attacks that might be embedded within the data segment of the packets?

The solution to this problem must resolve the following forces:

- We need to let external networks access our services and local users access external sites. Isolation is not acceptable.
- There are a variety of application services in a system, for example mail, file transfer, and others. Hackers can plan specific attacks against them, and we need to be prepared for a variety of attacks.
- Network administrators deploy and configure a variety of protection mechanisms, so it is important to have a clear model of what is being protected and what types of attacks are possible.
- The protection mechanism should be able to precisely reflect the security policies of the organization.

- The types of attacks are constantly changing, so it should be easy to make changes to the configuration of the protection mechanisms.
- It may be necessary to log requests for auditing and defense purposes.

7.5. Solution

Make the client interact only with a proxy of the service requested, which in turn communicates with the protected service (see figure 17). The client can only receive service from the server if an application proxy exists for the requested service. Each application proxy has its own access rules pre-defined by the administrator that may be used to authenticate, inspect, change, and filter the incoming (or outgoing) messages.

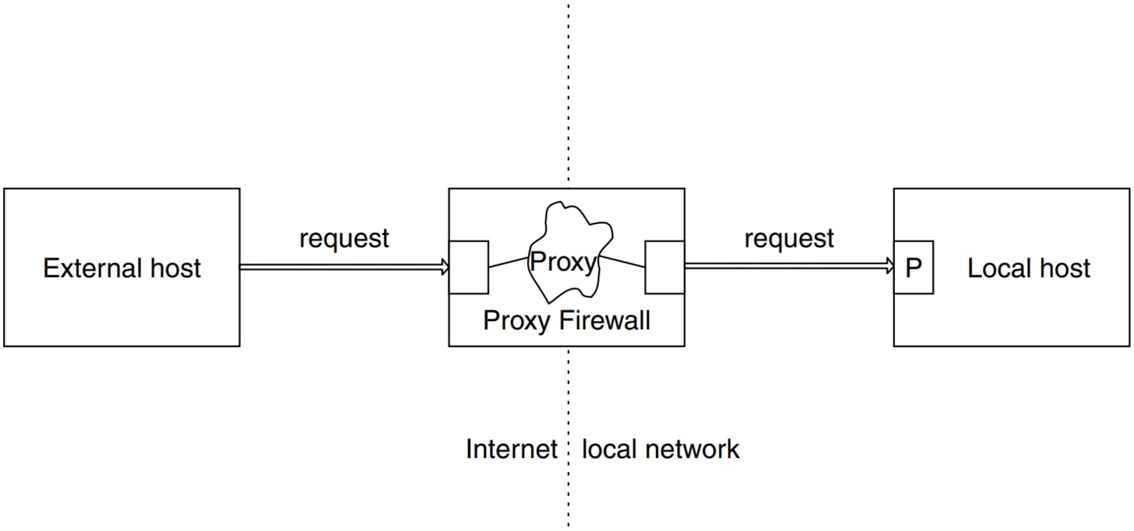

Figure 15: The concept of the proxy-based firewall

7.6. Structure

The figure below shows the class diagram for this pattern. We show here only the proxy aspects: the classes shown in figure 18 can be part of this firewall or can be provided separately. This firewall contains Proxies, which in turn contain Rules, collected in a RuleBase. All the hosts of a local network share the firewall. Each local host provides a set of services. The rules may now specify specific constraints for the use the available services.

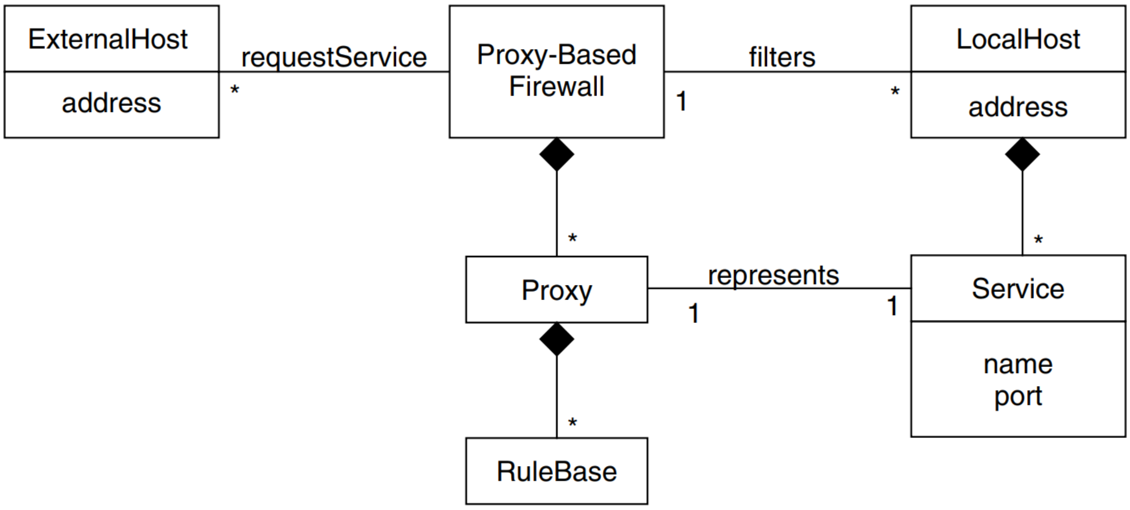

Figure 16: Class diagram for PACKET FILTER FIREWALL

7.7. Dynamics

We illustrate a use case for filtering requests for services. See the sequence diagram on figure 19.

Use Case: Providing Service to a Client

Summary An external client wants access to a service from a local host. The access request is made through the firewall, which according to its application proxies and their rules determines whether to deny or accept the request.

Actors External client.

Precondition None.

Description An external network requests a service to the PROXY-BASED FIREWALL. The firewall filters the request according to its application proxies and their access rules. If none of the rules in the rule set are satisfied, then a default rule is used to filter the request. If the request is accepted, the client is allowed to access the service through the proxy.

Alternate Flows If the service request is not supported by the PROXY-BASED FIREWALL, or the firewall considers the client untrustworthy, the firewall will block the access.

Postcondition The firewall has accepted the service request from a trustworthy client to the local host.

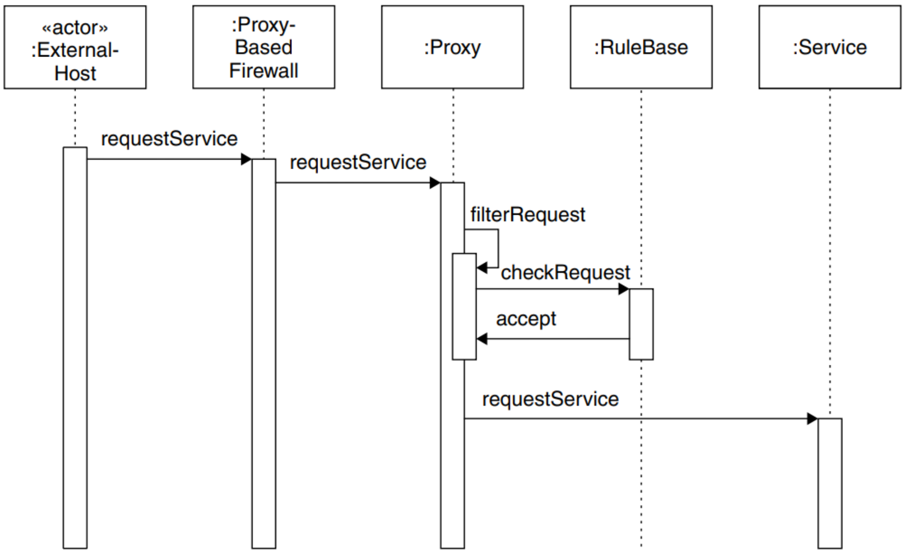

Figure 17: Sequence diagram for filtering service requests.

7.8. Implementation

1. According to organization policies, define which services will be made available to clients of the network.
2. Write, reuse, or buy a proxy for each service and assign a location or address to it.
3. Define who can have what type of access to which service and other restrictions on their use.
4. Implement these constraints in the rule base.
5. Consider configurations such as PROTECTION REVERSE PROXY, INTEGRATION REVERSE PROXY or a combination with a PACKET FILTER FIREWALL in a distributed configuration [1].

7.9. Example Resolved

We bought a PROXY-BASED FIREWALL and now every request for a service is authenticated and checked. We can verify that the requests are authentic and filter out some payload attacks, for example, a wrong command for a service, wrong type parameters in the service call, and so on.

7.10. Consequences

The following benefits may be expected from applying this pattern:

- The firewall inspects and filters all access requests based on predefined application proxies that are transparent to the users of the services. In some cases, it may even modify a request—for example, doing network address translation.
- It is possible to express the organization’s filtering policies through its application proxies and their rules.
- The implementation details of the local host can be hidden from the external clients. This also improves security.

- A firewall permits systematic logging and tracking of all service requests going through it. This facilitates the detection of possible attacks and helps hold local users responsible of their actions.
- It provides a higher level of security than packet filters because it inspects the complete packet including the headers and data segments. This global view may control attacks in the payload and attacks based on the structure and size of the packets.

The following potential liabilities may arise from applying this pattern:

- Possible implementation costs due to the need for specialized proxies. The proxies also need to be configured correctly. On the other hand, proxies already exist for common services.
- Performance overhead due to the need for inspection of the data segment of packets and maybe additional checking.
- Increased complexity of the firewall. A PROXY-BASED FIREWALL may require a change in applications and/or the user's interaction with the system. This is not necessary, however, in a well-designed system.

7.11. Known Uses

Some specific firewall products that use application proxies are Pipex Security Firewalls [2] and InterGate Firewall. The SOCKS Protocol from IETF, although not intended as a firewall, uses a similar principle [3]. Postfix filters act as proxy and packet filter firewalls [4].

7.12. See Also

This pattern uses the PROXY pattern from [5]. It can be combined with PACKET FILTER FIREWALL and STATEFUL FIREWALL.

7.13. References

[1] CyberGuard Corporation. (2003, January). Defense in Depth: Applying a Fortified Firewall Strategy. white paper

[2] Pipex. (n.d.). Pipex Firewall Solutions. http://www.security.pipex.net/about.shtml

[3] IETF. (n.d.). SOCKS Protocol. http://www.socks.permeo.com/

[4] Hafiz, M., Johnson, R. E. (2005). The Security Architecture of qmail and Postfix. https://netfiles.uiuc.edu/mhafiz/www/ResearchandPublications/SecurityArchitectureofqmailAndPostfix.htm

[5] Gamma, E., Helm, R., Johnson, R., Johnson, R. E., & Vlissides, J. (1995). *Design patterns: elements of reusable object-oriented software*. Pearson Deutschland GmbH.

7.14. Source

Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., & Sommerlad, P. (2006). Security Patterns: Integrating Security and Systems Engineering (1st ed.). Wiley.

8. Risk Determination

8.1. Intent

Risk determination is the final stage of a risk-assessment process, and incorporates the results from an asset valuation, a threat assessment, and a vulnerability assessment. Using the input of these patterns, the enterprise is able to evaluate and prioritize the risks to its assets.

8.2. Example

The museum has identified the following assets as part of its risk assessment:

- Information asset types
    - Museum employee data
    - Museum financial/insurance data, partner financial data
    - Museum contractual data and business planning
    - Museum research and associated data
    - Museum advertisements and other public data
    - Museum database of collections information
- Physical assets
    - Museum building
    - Museum staff
    - Museum collections and exhibits
    - Museum transport vehicles

It has also completed the three major steps in a risk assessment, as defined by ASSET VALUATION, THREAT ASSESSMENT, and VULNERABILITY ASSESSMENT. It must now assimilate this information, evaluate the overall risk, and present the results.

8.3. Context

An enterprise has defined the assets to be included in a risk assessment and has evaluated the importance of those assets in an asset valuation table. As well, it has performed a threat assessment and vulnerability assessment and collected unique combinations of threats and vulnerabilities in a threat-vulnerability table.

8.4. Problem

Once the work has been done to determine an asset’s worth and assess the threats and vulnerabilities that affect it, its overall risk needs to be determined. Without a formal method for determining risk, how can one be assured that effort expended in protecting an asset is too high or too low?

How does an enterprise evaluate the risks posed to its assets?

An enterprise must resolve the following forces:

- The results of the risk assessment must be understood by the executive team if they are to address risk in the enterprise effectively.
- Determination of risk is directly related to asset value, threat likelihood, and vulnerability severity.
- Conducting a risk assessment requires resources such as time, people, and project funding, as well as a commitment to follow up the results.
- Quantitative risk measures imply greater precision and are therefore preferred over qualitative indicators, but only if the quantitative scores are based on adequate measurements: false precision in risk levels is misleading.

8.5. Solution

Systematically determine the risk that is posed to each enterprise asset. This process involves the following four steps:

1. Collect results from ASSET VALUATION, THREAT ASSESSMENT and VULNERABILITY ASSESSMENT. Recall that the previous stages of the risk assessment are the asset valuation, threat assessment and vulnerability assessment. Apply those patterns and collect the following:
    a. The asset valuation table: this table shows the overall value of enterprise assets.
    b. The threat-vulnerability table: this table is a catalog of threats and their associated vulnerabilities. Each threat includes a likelihood rating, and each vulnerability includes a severity rating.
2. Associate threat-vulnerability pairs with assets. Using the threat-vulnerability table, identify all threat-vulnerability pairs that pose a direct risk to each asset separately.
3. Evaluate risk. Evaluate a risk equation using the numerical values for asset valuation, threat likelihood and vulnerability severity. The result will represent the final risk posed to each asset.
4. Present the results. Sort the results in order of decreasing risk. Use qualitative terms, a color scale or other scale system (as appropriate) to display the results.

8.6. Structure

8.7. Dynamics

The allowable sequence for performing RISK DETERMINATION is shown in figure 20:

- First, collect the asset valuation and threat-vulnerability tables from ASSET VALUATION and VULNERABILITY ASSESSMENT, respectively.
- Use a risk equation to calculate the risk posed to each asset.
- Finally, sort and present the results in descending order.

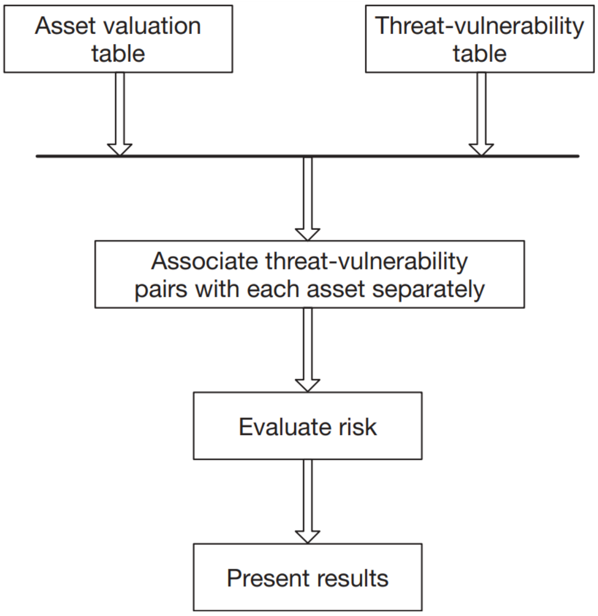
Asset valuation table
Threat-vulnerability table
Associate threat-vulnerability pairs with each asset separately
Evaluate risk
Present results

Figure 18: Risk determination sequence constraints

8.8. Implementation

The implementation of the process for risk determination is described below.

1. Collect results from ASSET VALUATION, THREAT ASSESSMENT and VULNERABILITY ASSESSMENT. Apply these three patterns and collect the asset valuation and threat-vulnerability tables.
2. Associate threat-vulnerability pairs with assets. In both THREAT ASSESSMENT and VULNERABILITY ASSESSMENT, we grouped assets by either physical or information type, rather than individually. At this stage of RISK DETERMINATION, we now need to consider the threat-vulnerability pairs for each asset separately.
   The threat-vulnerability table lists all threat actions and their corresponding vulnerabilities. Each of these pairs may pose a risk to one or more informational or physical assets. Therefore, identify all the threat-vulnerability pairs that affect each asset directly. The condition of 'affecting directly' is important, because to associate all threat-vulnerability pairs for every asset would lead to identical and, ultimately, meaningless results. However, a single threat-vulnerability pair may certainly affect multiple assets directly.
3. Evaluate risk. Regardless of the actual equation or method used to evaluate risk, it must consider the following properties:
   a. The more vulnerabilities that exist in an asset and the systems that enable access to it, the greater the risk.
   b. The more severe the vulnerabilities, the greater the risk.
   c. The greater number of threats that could exploit a vulnerability, the greater the risk.
   d. The more likely the threats, the greater the risk.
   e. The more valuable an asset, the greater the risk.
   f. The risk to an asset is zero if no threats or vulnerabilities exist for that asset.

   Any number of equations could be used to calculate a risk value, including those presented in the Variants and Known Uses sections. For the purposes of this pattern, we will use the following equation for each asset included in the scope of the risk assessment:

Risk(A) = SUM[Threat * Vulnerability](A) * Asset Value(A)

This can be read as, ‘the risk to asset ‘A’ is the sum of all unique combinations of threat likelihood, multiplied by the vulnerability severity, multiplied by the asset value.’

4. Present the results. Present the results in order of descending risk. The greatest risk will have the highest numerical value, whereas the lowest risk will have the lowest numerical value. All values will be greater than zero, and the numbers will most certainly vary from one risk assessment to another.
   If necessary, the raw numerical values can be presented in a table. However, a more intuitive effect can be achieved by using qualitative terms, consistent with those used throughout the risk assessment pattern set. First, on a scale of 1 (representing the lowest possible risk value) to the highest risk value, create 6 equal ranges, labeled as: Negligible, Low, Medium, High, Very high and Extreme. Then group each asset according to its qualitative value.
   a. Understanding and presenting the results. The importance of sorting and clearly presenting the results to a senior management team cannot be overemphasized. It is their task to interpret the results and develop plans to mitigate, transfer or accept the risk, often as part of an overall risk management strategy. Generally, this senior management team will only be interested in the risk values relative to other assets, so the actual value itself is not important. An exception to this is when the results from one assessment are compared with those from another assessment, perhaps from previous years. A declining value, for example, would demonstrate a reduction in risk, either due to fewer or less likely threats, more effective security controls, or declining asset value.
   b. Qualitative versus quantitative risk determination. Although the final results can be given in numerical terms, RISK DETERMINATION (as with ASSET VALUATION, THREAT ASSESSMENT and VULNERABILITY ASSESSMENT) is very much a qualitative process. The values used in these patterns reflect the relative numerical values, rather than objective, quantifiable numbers.

8.9. Example Resolved

Using the asset valuation table and threat-vulnerability table as input to RISK DETERMINATION, the museum has evaluated and prioritized the risks to its assets. The complete results of the risk equation for three museum assets are presented below, and the remaining results are summarized in Table 9.

8.9.1. Evaluation of Risk Equation

1. Risk evaluation for museum building. From the threat-vulnerability table of VULNERABILITY ASSESSMENT, the museum has identified three threat-vulnerability pairs that affect the museum building, as shown in Table 4.
   ASSET VALUATION identified the museum building as having a value of 6. The risk equation can therefore be written as follows:

Risk = (3*6 + 3*5 + 3*4) * 6
Risk = (18 + 15 + 12) * 6
Risk = (45) * 6
Risk (museum building) = 270

Threat Action (Frequency)	Vulnerability (Severity)
Natural	
Museum fire (3)	Failure of fire alarm system (6)
	Failure of fire suppression system (5)
Fatigue of support fixtures, building structural failure (3)	Lack of regularly scheduled inspections (4)

Table 4: Threat-vulnerability pairs for museum building

2. Risk evaluation for museum collections and exhibits. The museum collections and exhibits asset has an asset value of 6, with the threat-vulnerability pairs as shown in Table 5.

Risk (33 + 12 + 16 + 12 + 12 + 8 + 20 + 12 + 6) * 6
Risk = 131 * 6
Risk (museum collections and exhibits) = 786

Threat Action (Frequency)	Vulnerability (Severity)
Natural	
Museum fire (3)	Failure of fire alarm system (6)
	Failure of fire suppression system (5)
Fatigue of support fixtures, building structural failure (3)	Lack of regularly scheduled inspections (4)
Failure of monitoring and alarm systems (4)	Lack of regularly scheduled inspections (4)
Professional criminals	
Theft of museum collections and exhibits (2)	Lack of regular alarm testing procedures (3)
	Lack of adequate storage and protection of physical assets (3)
Physical attack against employees (3)	Lack of security training for employees (4)
Employees	
Accidental damage to museum collections and exhibits (4)	Carelessness of employees when handling/cleaning exhibits (2)
Theft of museum collections and exhibits (2)	Lack of regular alarm testing procedures (3)
	Lack of adequate storage and protection of physical assets (3)
	Susceptibility of employees to bribery (4)
Misconfiguration of monitoring and alarm systems (4)	Lack of regular alarm testing procedures (3)
Museum patrons	
Accidental damage to museum collections and exhibits (3)	Carelessness of museum patrons when viewing exhibits (2)

Table 5: Threat-vulnerability pairs for museum collections and exhibits

3. Risk evaluation for museum employee data. See Table 6.

$$Risk = (12 + 12 + 21 + 15 + 10 + 9) * 5$$
$$Risk = 79 * 5$$
$$Risk\ (museum\ employee\ data) = 395$$

Threat Action (Frequency)	**Vulnerability (Severity)**
Natural	
Electrical spike in computer room (3)	Lack of surge protection, uninterruptible power system (UPS) (4)
Loss of electronic documents (3)	Incomplete or corrupt data backups (4)
Professional criminals	
Theft of information assets (3)	Susceptibility of employees to bribery (3)
	Lack of proper physical controls for document storage (locks, safe) (4)
Employees	
Unauthorized access of informational assets (5)	Weak information security controls enabling unauthorized access (3)
Data entry errors (5)	Lack of data validation during form input (2)
Leaking confidential information (3)	Exposure of information assets (3)

Table 6: Threat-vulnerability pairs for museum employee data

4. Complete results. Risk values have been calculated for the remaining assets and are presented in Table 7.

Asset	**Risk Value**
Museum collections and exhibits	786
Museum employee data	395
Museum staff	342
Museum financial/insurance data, partner financial data	316
Museum building	270
Museum contractual data and business planning	232
Museum database of collections information	232
Museum research and associated data	147
Museum transport vehicles	120
Museum advertisements and other public data	98

Table 7: Prioritized risks for museum assets

8.9.2. Presentation of results

6 equal ranges (from 1 to 786) have been created, as shown in Table 8, and the final qualitative results are presented in Table 9.

Rating	**Range**
Extreme	656–786
Very high	525–655
High	394–524
Medium	263–393
Low	132–262
Negligible	1–131

Table 8: Qualitative risk translation

Asset	**Risk**
Museum collections and exhibits	Extreme
Museum employee data	High
Museum staff	Medium
Museum financial/insurance data, partner financial data	Medium
Museum building	Medium
Museum contractual data and business planning	Low
Museum database of collections information	Low
Museum research and associated data	Low
Museum transport vehicles	Negligible
Museum advertisements and other public data	Negligible

Table 9: Qualitative risks for museum assets

8.10. Consequences

This pattern has the following benefits:

- The enterprise is now able to identify and address the risks posed to its assets, as part of a risk mitigation effort.
- The qualitative results provided are much easier to calculate, prioritize and interpret.
- The results can be archived and used to track the progress of asset risk among consecutive risk assessments.

As well as the following liabilities:

- The risk equation may not account for all the properties of the relationship between threat, vulnerability, and asset value.
- The results are based on the completeness and subjectivity of ASSET VALUATION, THREAT ASSESSMENT and VULNERABILITY ASSESSMENT, and therefore cannot be objectively verified or guaranteed.
- Because of the various methods for calculating an actual risk value, an enterprise may find it difficult to identify the particular equation that meets their risk assessment needs.

8.11. Variants

An alternative formula for risk determination is provided by [1]:

Risk = Probability * Damage Potential

in which both the probability and damage potential variables are represented numerically as values from 1 to 10, giving a minimum and maximum risk value of 1 and 100 respectively. To achieve qualitative results, ‘low’ represents any risk from 1 to 33, ‘medium’ represents risks from 34 to 66, and ‘high’ represents risks from 67 to 100. Note that because this method is threat- based, it gives the risk of a particular threat, as opposed to the risk posed to an asset.

Appendix E of [2] provides a number of examples of the use of matrices to evaluate risk, in which each example places emphasis differently. One example offers an asset-based evaluation, whereas another assesses the risk of given threats. While these examples recognize the inherent relationship between threats and vulnerabilities, they do not provide a formal way of accounting for them.

8.12. Known Uses

[3] uses a 3x3 matrix made up of threat likelihood and threat impact. Qualitative values of threat likelihood (high, medium, low) are converted numerically to ratings of 1.0, 0.5, and 0.1 respectively. Qualitative values of threat impact (high, medium, low) are converted numerically to ratings of 100, 50, and 10 respectively. Risk is then computed by multiplying the threat likelihood by threat impact for each identified threat-vulnerability. The resulting value represents the 'degree or level to which an IT system, facility or procedure might be exposed if a given vulnerability were exercised.' Note that while this method is clear and straightforward, it does not provide an overall risk rating to a given asset, but simply the risk of a single threat-vulnerability pair.

[4] describes an Annual Loss Exposure (ALE)—an equation that provides a quantitative method for calculating loss. The ALE is calculated from the value of an asset (A) multiplied by the likelihood of a threat occurrence (L) as follows: The likelihood value used is calculated from a multiplier table in which an occurrence of once a day is 365, once a month is 12, once a year is 1, once every 5 years is 1/5, and so on.

8.13. See Also

[1] Meier, J. D. (2003). *Improving web application security: threats and countermeasures*. Microsoft press.

[2] "Technical Report ISO13335-3 Part 3: Techniques for the Management of IT Security" International Organization for Standardization, American National Standards Institute, 2002

[3] Stoneburner, G., Goguen, A., & Feringa, A. (2002). Risk management guide for information technology systems. *Nist special publication*, *800*(30), 800-30.

[4] Peltier, T. R. (2005). *Information security risk analysis*. CRC press.

8.14. References

8.15. Source

Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., & Sommerlad, P. (2006). Security Patterns: Integrating Security and Systems Engineering (1st ed.). Wiley.

9. Role Rights Definition

9.1. Intent

'Least privilege' is a fundamental principle for secure systems. Roles can directly support the least privilege principle, but a systematic approach to assigning only the required rights to each role is required. This pattern provides a precise way, based on use cases, of assigning rights to roles to implement a least-privilege policy.

9.2. Example

Multitronics is a company that sells on-line digital media such as video, sounds, or images. They have been advised that for security reasons they should use a ROLEBASED ACCESS CONTROL approach, in which they can apply a least-privilege policy. For this they need to first identify the roles required to perform the business functions. In this system a manager administers the items on sale, deciding what is to be sold, at what prices, and so on. He can also order items for future sale. Subscribers register and create accounts so that they can purchase copies of digital items and download them to a mobile device such as a cellular phone. Subscribers can also reserve items not yet in stock. A salesperson maintains a catalog of items for sale and bills the subscribers for their purchases. To apply the required policy, we need a systematic way to assign rights to these roles.

9.3. Context

Applications composed of a variety of roles in which it is not easy to assign proper rights to the roles.

9.4. Problem

The ROLE-BASED ACCESS CONTROL model is used now in many systems. However, the different component frameworks (.NET, J2EE) provide support only to define roles and to write authorization rules, and do not say anything about where the rights come from. It is not easy for system designers or for administrators to define the required roles and their corresponding rights.

How can we assign appropriate rights to the roles when we want to implement a least privilege policy?

The solution to this problem must resolve the following forces:

- Roles correspond to functional tasks in an organization, and we need to assign to these tasks sufficient rights to perform their work.
- Rights should be assigned according to the need-to-know (least privilege) principle, in which each role gets only the rights required to perform their duties.
- New roles appear and some roles may not be needed any more: changes to roles and their rights should be easy to perform.
- The assignment of rights should be independent of the system implementation.

9.5. Solution

Define the use cases of the system. The design of object-oriented systems always starts this way, but even systems that use other methodologies often define use cases as part of the requirements stage. As use cases define the interactions of actors with the system, we can interpret actors as roles. The roles that appear in a use case must be authorized for all the operations initiated by the role, or the role could not perform its functions. If we collect all the operations performed by a role over all use cases, they define the necessary rights for this role. To make this approach more detailed and systematic, we should build a use case diagram that displays all the use cases for the system, and sequence diagrams that show the interactions of roles with the system for each use case.

The figure below shows a generic sequence diagram indicating that actor role1 must use operations op1, op2, … opN to interact with the system. This means that role1 should be given the rights to apply these operations to the system.

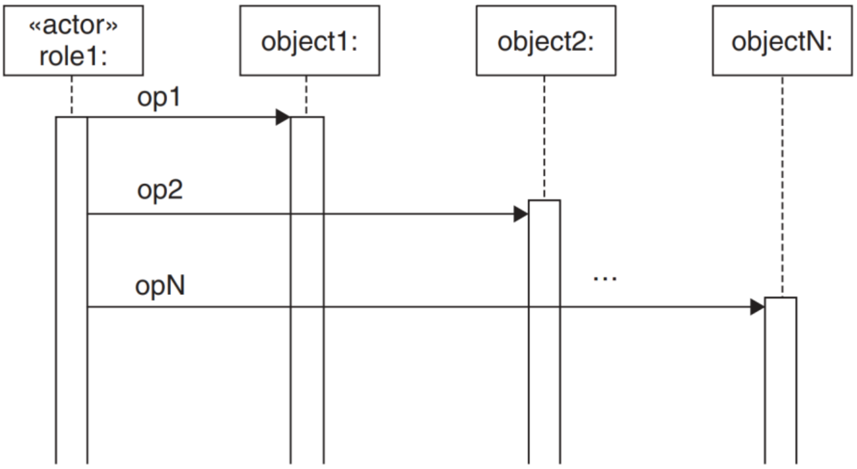

Figure 19: Generic sequence diagram to obtain rights for a role

9.6. Structure

9.7. Dynamics

9.8. Implementation

Consider the following steps in order to implement the solution:

1. Start by building a use case diagram to display all the use cases of the system. The actors in this diagram correspond to roles and we can capture all the required roles in this way.
2. Build sequence diagrams for each use case. There is a sequence diagram for the main flow and a few more diagrams for alternate flows [1].

3. Analyze all the sequence diagrams to see what operations the actors (roles) need to apply to interact with the system. These operations correspond to the role rights. In fact, these rights could be generated automatically from the use cases—tools such as Rational Rose can keep track of use cases, and they could be extended to generate the required authorization rules. One can also find all this information in the textual descriptions of the use cases, but it is harder to see the interactions, the sequence diagrams make the interactions more explicit.
4. From the use case exceptions, the administrator implements the actions needed for security violations.
5. Addition or deletion of authorization rules is only necessary if a use case is added or deleted, or some of the actions of a use case are changed.

In a centralized system, authorizations could be enforced at the user interface, while in a distributed system, authorization could be enforced in a centralized system component such as the application server. Object-oriented systems use approaches based on model-view separation, for example the MVC or PAC architectures [2]. These two models separate the conceptual model objects—a digital item in our example— from user interfaces that can observe and modify these conceptual objects. The user views should be defined based on use cases [3], and it is clear that they should be the only way to interact with the system. The user views should have access to the set of authorization rules to allow or deny access to the conceptual objects in the system.

Sequences of use cases can be used to define a workflow that requires a specific set of authorizations for different roles. For example, a digital item can only be added by the vendor, released by the administrator, purchased, and downloaded by the subscriber, in that order. This complete workflow could be authorized as a unit.

9.9. Example Resolved

Figure 22 below shows a use case diagram for the Multitronics on-line digital item vending system, including the roles defined earlier.

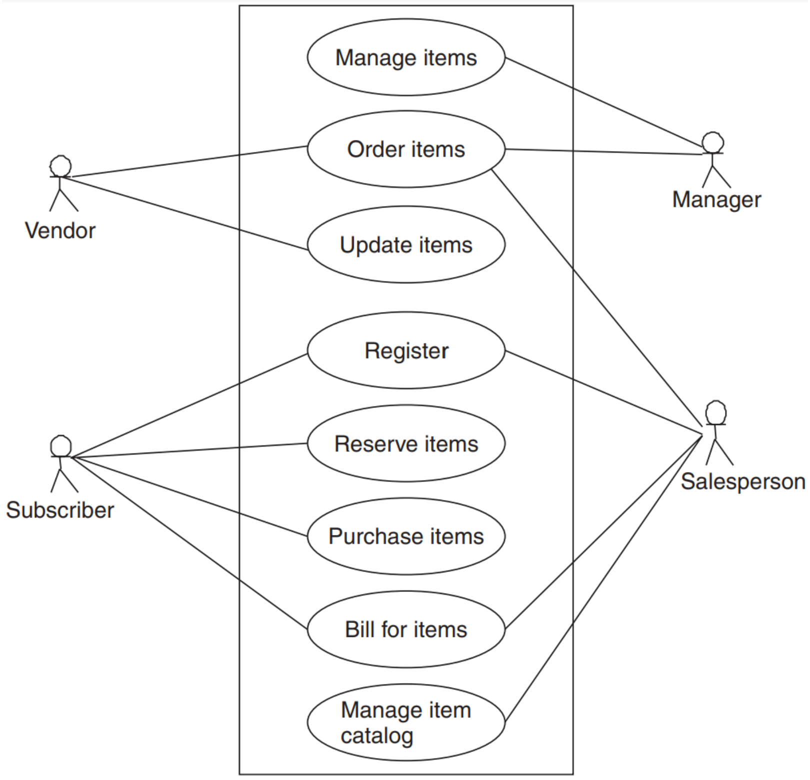

Figure 20: Use case diagram for a digital item management system

A subscriber participates in four use cases. Any user, once authenticated, has the right to register, but only registered users have the right to reserve and purchase items. These are all the rights needed for a subscriber in this system. A salesperson registers users in the system, bills users for their purchases, and maintains the catalog of products. He can also order new items from vendors according to customers' requests. A manager manages items and approves ordered items. Vendors have the right to upload the ordered items.

However, this is not the whole story. As indicated in the solution, a use case may include several actions that may be performed by different roles. To capture all the required rights, we need to look into the details of the use case or its corresponding sequence diagram. The figure below shows a sequence diagram to order an item. We can see that the salesperson initiates this use case and needs a right to order items. The manager has the right to approve the purchase, after which the salesperson has the right to send the order to the vendor.

The sequence diagram on figure 23 shows the purchase of a digital item by a registered user.

From the two use cases shown, we can deduce that a salesperson role needs a right to order items, a manager role needs a right to approve orders, a registered subscriber role has the right to purchase and download an item. Sequence diagrams for the remaining use cases would provide the complete set of rights for all the roles.

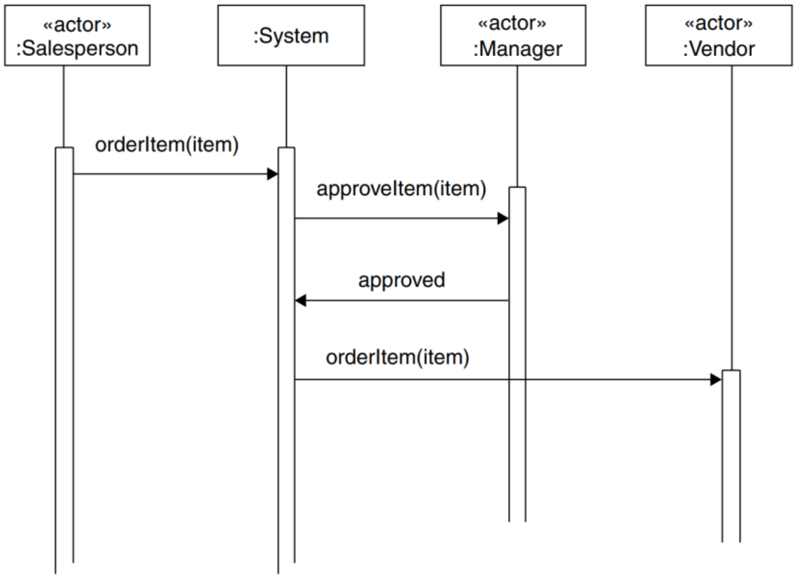

Figure 21: A sequence diagram for the use case Ordering a digital item

9.10. Consequences

The following benefits may be expected from applying this pattern:

- Because roles correspond to functional tasks, their rights are defined according to the needs of the tasks.
- If these are the only rights given to the tasks, we have implemented a least privilege policy.
- Since all the use cases define all the interactions with the system, all the necessary rights can be generated in this way.
- A new use case just defines new rights that can be easily added to the existing set of rights.
- The approach is independent of the actual system implementation. Only the actor's commands to the system need to be authorized, not the internal object accesses triggered by these commands. As long as the external view of the system does not change, there is no need to change authorization rules when the implementation changes. This is consistent with the information-hiding property of object-oriented systems.

The following potential liability may arise from applying this pattern:

- Building use cases requires specialized expertise, which may not be available in the organization.

9.11. Known Uses

Every complex object-oriented application using the RBAC model needs to define rights for its roles. Databases, for example Oracle, support user roles. Most modern frameworks, for example .NET and J2EE, support roles. Modern operating systems, for example Trusted Solaris 7 and higher versions, also support roles. ROLE RIGHTS DEFINITION indicates how to define rights for the roles in those systems.

9.12. See Also

ROLE-BASED ACCESS CONTROL defines the structure of the security system, using classes such as User, Role, and others. ROLE RIGHTS DEFINITION complements it, by providing a way to define the specific rights needed in a particular system.

9.13. References

[1] Larman, C. (2004). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development.

[2] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., & Stal, M. (1996). *Pattern-Oriented Software Architecture: A System of Patterns.* John wiley & sons.

[3] Losavio, F., & Matteo, A. (1997). Use case and multiagent models for object-oriented design of user interfaces. *Journal of Object-Oriented Programming*, *10*(2), 30.

9.14. Source

Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., & Sommerlad, P. (2006). Security Patterns: Integrating Security and Systems Engineering (1st ed.). Wiley.

10. Secure Channels

10.1. Intent

Messages passing across any public network—particularly the Internet—can be intercepted. The information contained in such messages is thus potentially available to an eavesdropper. For sensitive communication across a public network, create encrypted SECURE CHANNELS to ensure that data remains confidential in transit.

10.2. Example

A typical Internet-based application will exchange a variety of information with its users. Some of this information about people, products and services will be sensitive in nature. Typical examples include credit card numbers when making on-line purchases or bookings, or product plans and shipment schedules exchanged between business partners.

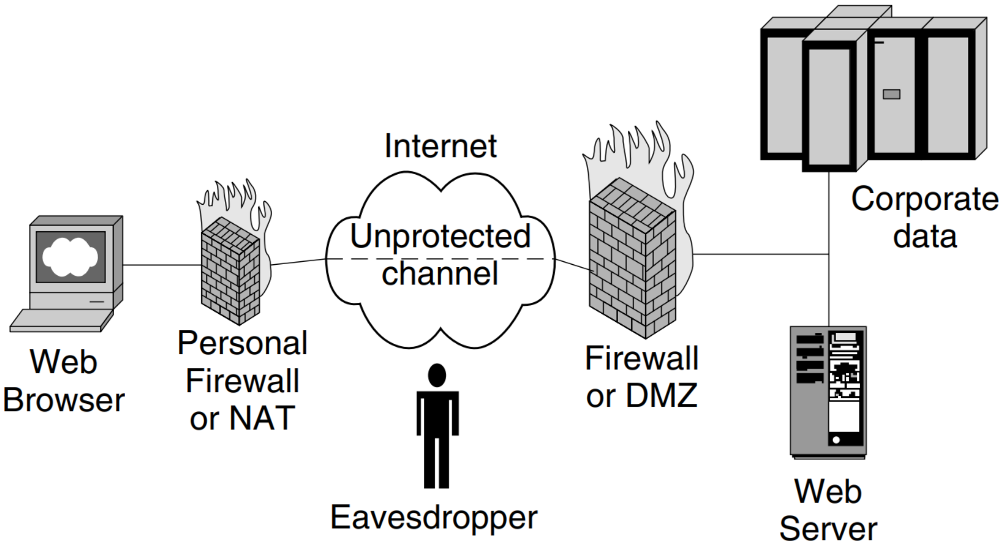

Figure 22: An unprotected channel

It is relatively straightforward to secure data on the client and the server. The client and server can be protected by different firewall mechanisms, in the case of the server maybe even a fully-fledged DEMILITARIZED ZONE, to make it difficult for an attacker to penetrate the systems and gain access to the data they hold. If the data is of a particularly sensitive nature, such as credit card numbers, it may even be stored in encrypted form following INFORMATION OBSCURITY. However, data on the Internet itself has no protection from intruders, and straightforward encryption mechanisms that can be used in a managed environment cannot be applied between two unrelated machines across the Internet. Because of this, data is passed across an unprotected channel, as shown in figure 24.

A private channel could be set up between client and server, but this would rely on a private networking mechanism, which defeats the object of delivering services cheaply and conveniently across a public network such as the Internet.

10.3. Context

The system delivers functionality and information to clients across the public Internet through one or more Web servers. Larger systems may use multiple Web servers and multiple application servers to deliver this functionality, all protected by a DEMILITARIZED ZONE. The application must exchange data with the client. A percentage of this data will be sensitive in nature.

10.4. Problem

How do we ensure that data being passed across public or semi-public space is secure in transit? The solution to this problem must resolve the following forces:

- Much application data is non-sensitive, but data that is sensitive needs protection when it is made available outside the system's defense mechanisms.
- Encrypting data is a significant overhead on system performance.
- It is easier to provide encryption solutions with known partners, but many customers of the system cannot or will not install specific software or hardware for this purpose.

10.5. Solution

Create secure channels for sensitive data that obscure the data in transit. Exchange information between client and server to allow them to set up encrypted communication between themselves. Reduce the associated overhead on the system by using ordinary communication channels for non-sensitive data.

10.6. Structure

SECURE CHANNELS requires the following elements:

- A Web server, which provides access to the application's functionality and information. The Web server software could be one of many common types such as Apache or Internet Information Services, but it must support the encryption and key exchange mechanisms being employed.
- A client browser, the universal client used to access the system. The browser can be any generic browser such as Internet Explorer or Netscape, but it must support the encryption and key exchange mechanisms being employed.
- A shared encryption key. To exchange encrypted data in a secure but efficient manner, the client browser and Web server must share a secret value. This usually takes the form of a symmetrical encryption key that can be used to both encrypt and decrypt data.
- A key exchange mechanism such as an agreed protocol that the client browser and Web server use to exchange the shared encryption key securely.
- An encryption mechanism. The client browser and Web server use an agreed encryption mechanism that is applied to sensitive data. Armed with the shared encryption key and an algorithm to implement this encryption mechanism, both client and server can encrypt and decrypt confidential data.

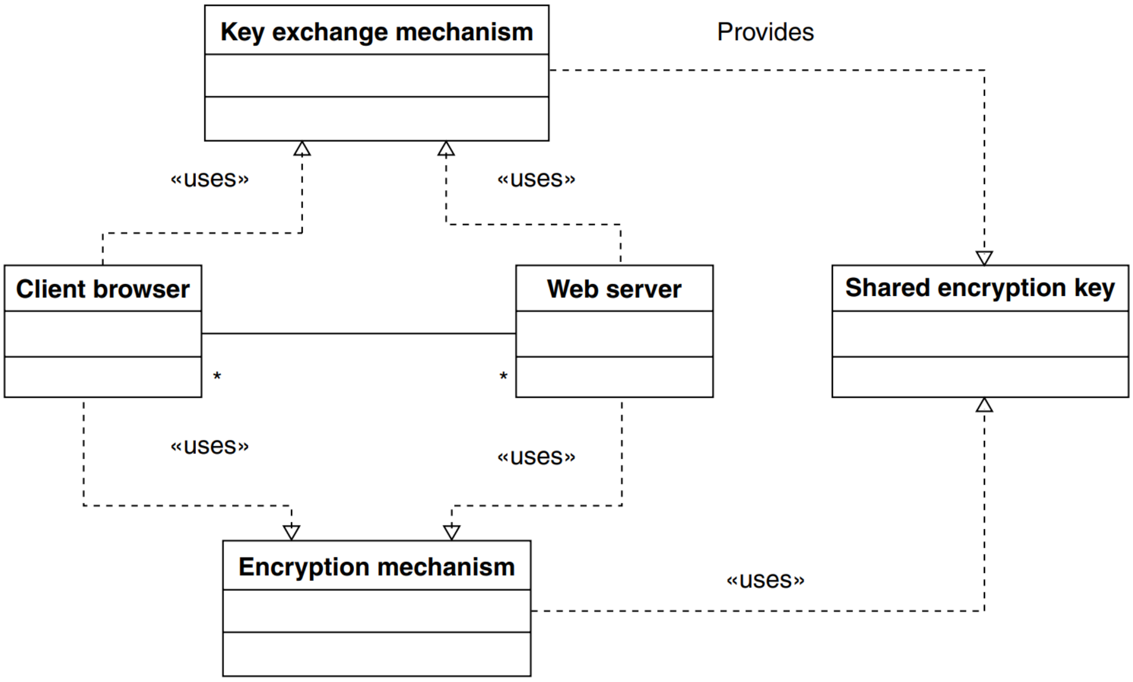

Figure 23: Composition of elements for SECURE CHANNELS

10.7. Dynamics

Most of the time the client and server exchange information over a normal, unencrypted channel. When they wish to exchange sensitive data, they set up a secure channel using encryption. This section focuses on such an encrypted exchange. In a typical encrypted exchange, two parties, Alice, and Bob, wish to exchange information. They can use a shared, symmetric encryption key to pass information privately across a public space, such as the Internet. Eve, the eavesdropper trying to intercept the message, cannot decrypt the message even if she captures it, because she does not possess the shared key. This is the basic privacy mechanism used by Secure Sockets Layer (SSL), the most prevalent secure channel mechanism on the Internet.

In Internet terms, the roles of Bob and Alice are played by the Web server and the browser or other client software. However, this presents us with a problem: usually the client and server do not know each other well enough to have established a shared key. The shared key cannot just be built into the Web server and browser software, otherwise everyone could decrypt everybody else's messages. Each SSL session uses a unique shared key—hence it is also called the 'session key' in SSL. What we need is a way for the Web server and the client to exchange the session key to be used.

The exchange of session keys is based on the use of a digital certificate. The owner of the Web server must obtain a server certificate that associates a given public key with the server's DNS name, for example www.securitypatterns.org. Once the server certificate has been installed on the Web server, a client requests a secure channel by accessing a resource using a URL that starts with 'https:' rather than 'http:' This request causes the server to send the client its digital certificate to prove its identity, and to provide the client with its public key. The client then checks the digital certificate to make sure that it is issued by a trusted third party and that it matches the DNS name with which it is accessing the server.

If the certificate looks valid, the client generates a symmetrical encryption key (the session key) and uses the public key in the certificate to encrypt it. The server uses its private key to decrypt the session key, then starts using the session key to exchange encrypted messages with the client. This exchange has achieved two things: the client now believes that the server is genuine, and the client and server now have a shared, secret key with which to exchange private messages—in this case, the contents of HTTP POST requests. This key exchange can be extended to allow the client to authenticate itself with the server, which is important for KNOWN PARTNERS, but is not essential for most SECURE CHANNELS across the Internet. The essence of the key exchange is shown in the figure below, with Bob in the role of the Web server. For a more detailed description of how the whole exchange works, see [1]. See figure 26.

You may at this point be wondering why we do not just use the public/private key pairs to encrypt the data passing back and forth. The answer is that the symmetrical session key and its associated algorithm are respectively shorter and quicker to run than those for public/private key encryption. Most machines do not currently have the necessary resources to encrypt the amount of data passing between a Web client and server in an appropriate time using public key cryptography. This is a trade-off between performance and security.

Because the session key is shorter and its algorithm simpler, ciphertext based on it is easier to crack than ciphertext based on public key cryptography. If the same session key is used all the time between a client and server, it becomes increasingly possible to work out the shared key using statistical analysis of the messages being passed based on the number of times particular words appear in the language in which the messages are written. To counter this, the session key is changed on a regular basis using a mechanism similar to the initial exchange of the session key described earlier.

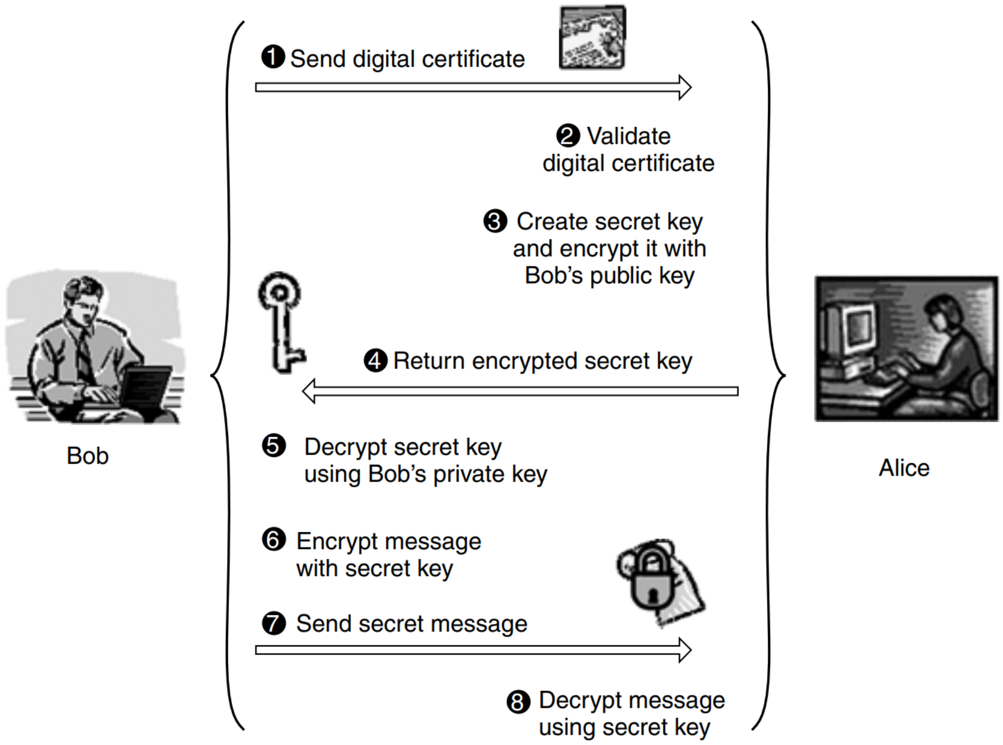
❶ Send digital certificate
❷ Validate digital certificate
❸ Create secret key and encrypt it with Bob's public key
❹ Return encrypted secret key
❺ Decrypt secret key using Bob's private key
❻ Encrypt message with secret key
❼ Send secret message
❽ Decrypt message using secret key
Bob
Alice

Figure 24: Key exchange for SECURE CHANNELS

10.8. Implementation

Because much information provided by the system is non-sensitive, it can be distributed by normal Internet mechanisms such as HTTP. When the system needs to exchange sensitive data with a user, a SECURE CHANNEL is set up specifically for that exchange. The most common mechanism for creating SECURE CHANNELS across the Internet is the Secure Sockets Layer (SSL). SSL capabilities are built into all major current Web browsers, and also into popular development platforms such as J2EE and .NET. Any application that wishes to use these capabilities merely needs to obtain a server certificate for SSL that can then authenticate the server to the client and can be used as the basis for secure session key exchange.

One issue to consider is that the increased security delivered by SECURE CHANNELS may conflict with other desired non-functional characteristics. One obvious conflict is between the use of SSL and performance. In theory we could use SSL for all exchanges between client and server—in practice, this imposes too great an overhead on the exchange of non-sensitive information. Another less obvious conflict is between SSL and the application of LOAD BALANCED ELEMENTS [2] to the Web servers to improve availability and scalability. When load balancing is combined with SECURE CHANNELS it presents a problem, because if the client were to be routed to a different server than the one that began the SSL session, the new server would not possess the session key for that SSL exchange. One solution here is to 'pin' a particular client to a particular server for the duration of its SSL exchange, a technique that is sometimes termed 'server-affinity.' However, this impacts on the availability and scalability of the solution.

To use load balancing and SECURE CHANNELS in combination, it is best to use load balancing hardware that understands secure channels and that can itself participate in the secure channel on behalf of the server. This solution avoids issues of server-affinity, but does open up a further security gap, as unencrypted information is exchanged between the load balancer and the servers. To address this problem, we introduce a totally new set of Web servers and load balancers. Any traffic that enters the outermost switch will either arrive on port 80, the default HTTP port, or port 443, the default HTTPS port. Traffic on port 80 is switched to a standard Web server via the standard load balancer. Traffic on port 443, however, can only go to a secure Web server via a secure load balancer. The packets passed between the secure load balancer and the selected secure Web server are still unencrypted, but we now have the opportunity to put in place additional security measures to harden the channel from secure load balancer to secure Web server. This effectively extends the secure channel from the browser right down to the secure Web server.

The use of SSL between the client and the Web server is fairly standard and is the obvious place to apply SECURE CHANNELS: this applies for both B2C e-commerce and B2B2 e-commerce. However, this is not the only place that such security should be considered. Even if a site has been protected as described in DEMILITARIZED ZONE, it may be possible for an attacker to penetrate one of the routers, or even a Web server on the DMZ. From this vantage point they can potentially monitor traffic within the DMZ as it passes between the Web servers and the application servers. If this traffic is not encrypted, it is then available to the eavesdropper. To avoid this possibility, you can set up a virtual private network (VPN) between the Web servers and the application servers. This VPN makes sure that data is encrypted as it passes through the DMZ, the firewall and the internal router.

10.9. Example Resolved

The Internet application uses SSL between browser and Web server to create a SECURE CHANNEL. Such channels are used to protect application data in transmission in different scenarios:

- Passing payment information between client and server
- Viewing of order status by customers
- Logging in by customers
- Changing of details by customers

For a high-availability system, load balancing content switches would be used to process SSL so that the SSL session is between the client browser and the load balancer (as opposed to between the client and the server). Although a VPN using IPSec would provide peer-to-peer security between the Web and application servers, this would be an unnecessary overhead for most applications given the sensitivity of the information passed back and forth.

10.10. Consequences

The following benefits may be expected from applying this pattern:

- Security is improved, because even in the event of an attack that captures data in transit, the data is not usable by the attacker.
- Common implementations of SECURE CHANNELS are built into most Internet software.
- Key exchange allows previously unknown partners to conduct confidential conversations.
- The mechanism does not impact the exchange of non-sensitive data, because it is only used when sensitive data is to be exchanged.

The following potential liabilities may arise from applying this pattern:

- Performance is impacted by the processing overhead associated with the encryption mechanism.
- Scalability is potentially impacted if the encryption mechanism caused server affinity, which would undermine effective load balancing.
- Availability is potentially impacted if the encryption mechanism caused server affinity, which would undermine effective fail-over.
- Cost is increased and maintenance overhead is added, because you must obtain and maintain one or more server certificates for your SECURE CHANNELS. Also, you may need to increase the hardware specification of your Web servers or buy dedicated encryption hardware to mitigate the associated performance overhead.

10.11. Variants

Asynchronous Secure Channel. So far, this pattern has discussed synchronous secure channels between clients and Web servers. However, there are other ways to implement secure channels. One option is to use an asynchronous messaging system and to encrypt the contents of the messages. Asynchronous operation gives us better performance and availability characteristics, at the expense of the additional processing that is required to correlate messages and to recover from failure.

In terms of Internet technology, we can use MIME-encoded e-mail messages with encrypted payloads as a secure, asynchronous channel. Alternatively, we can use encrypted XML/SOAP messages, as defined in the WS-Security specification [3]. These messages can be delivered synchronously (HTTP), pseudo-asynchronously (one-way HTTP message), or asynchronously (e-mail). While the use of asynchronous messaging is generally useful, you may well need to write more custom code to support this unless you find some good products to help you out.

10.12. Known Uses

SECURE CHANNELS is implemented in all mainstream Web browsers (Internet Explorer, Netscape, Mozilla, Opera, Firefox) and Web servers (IIS, Apache) through the provision of SSL functionality. Support for SSL is also included in development platforms such as .NET and J2EE. There are other, less commonly used variations on SECURE CHANNELS, such as IPSec, TLS and various VPN protocols.

10.13. See Also

10.14. References

[1] R. Anderson: Security Engineering – A Guide to Building Dependable Distributed Systems, John Wiley & Sons, 2001

[2] Dyson, P., Longshaw, A. (2004). Architecting Enterprise Solutions: Patterns for High-Capability Internet-Based Systems. John Wiley & Sons

[3] OASIS Open. (2005). OASIS Standards and Other Approved Works. http://www.oasis-open.org/specs/index.php#wssv1.0

10.15. Source

Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., & Sommerlad, P. (2006). Security Patterns: Integrating Security and Systems Engineering (1st ed.). Wiley.

11. Security Accounting Requirements

11.1. Intent

A security accounting service must satisfy a set of requirements for both the service and the quality of service. The function of security accounting is to track security related actions or events, such as damage to property, attempts at unauthorized database access, or transmission of a computer virus, and provide information about those actions. While each situation that calls for security accounting is unique, there are common generic requirements that apply to all security accounting situations. This pattern provides a common generic set of security accounting requirements. The pattern also helps you apply the general requirements to your specific situation and helps you to determine the relative importance of conflicting requirements.

11.2. Example

Gemstones within a museum are objects used in archeological research. They are also cleaned, transported, and handled by several authorized personnel. The museum is interested in protecting museum assets from theft, damage, or any mishandling. The museum is serious about assigning responsibility for any asset compromise or attempts to compromise assets. The museum needs to identify the requirements for the key components of a security accounting service that will help them protect their valuable gems and help assign responsibility for attempts to compromise their assets.

Based on the results of applying ENTERPRISE SECURITY SERVICES, Samuel the museum system engineer understands that the museum needs accountability of actions and events when the gems are transported or handled, and accountability of actions on the information about the gems, which is stored in a database. The museum needs to be able to assign responsibility for any asset compromise or attempts to compromise assets. For example, the museum needs to know who is responsible for transporting a gem. When information about a gem, such as its current location or its recorded carat weight, is entered or modified, the museum needs to know who made the addition or change. But Samuel also understands that the need to track and account for these actions and events must be balanced with the need for privacy and ease of operations. Therefore, Samuel needs to specify a balanced set of requirements for security accounting and the relative importance of those requirements, as a means of driving and evaluating an appropriate security accounting service for the museum. How can Samuel define such a set of requirements?

11.3. Context

The planned uses of security accounting are understood, for example, from applying ENTERPRISE SECURITY SERVICES. Asset types with a need for security accounting services are known, and the general types of actors that are to be held accountable are known. Actor types can include humans, software, business or automated processes, or information systems. Actors can be internal to an organization, such as an employee, or external, such as a supplier or customer. Asset types include both physical and information assets. The degree of confidence needed for the security accounting services by general asset types is known in relative terms. For example, a museum needs a very high degree of confidence in knowing who broke into the museum and stole a valuable gem, but it needs a lower degree of confidence in knowing who defaced the outside of the museum building.

11.4. Problem

Security accounting is an activity that takes in the detectable data from an event and provides some security-relevant information about that event to a human. A basic accounting sequence is completed when security-relevant information associated with an event of interest is provided to the accounting user. You need a clear set of requirements to ensure that the strategy employed for a security accounting system actually satisfies the needs of the organization or system. Requirements for security accounting often conflict with each other, and trade-offs between them are often necessary. You need to prioritize these requirements to determine under what circumstances you should put more emphasis on one requirement over another.

How can you determine specific requirements for a security accounting service, and their relative importance?

Below are examples of different security accounting use situations that define different security accounting needs for an organization. Many other security accounting service use scenarios are possible.

1. Security accounting is used to establish how well financial assets are being protected over a five-year period. The organization suspects that authorized access to the records is being used to misdirect funds, so security accounting is employed to help identify any perpetrators.
2. Security accounting is used to search for any intrusions into the organization's network. Security accounting monitors network traffic and compares that information to authorized traffic. Security accounting issues an alert if there is activity that is unwanted or unexpected.
3. Security accounting is used to establish a documented trail of evidence for global, very large, financial transactions. Security accounting must capture transaction terms, and the identities of parties that engage in such transactions. The terms and party identities must be accessible for review and reported to decision makers. There is a risk of large financial loss, and therefore the security accounting service must be as accurate as possible.

The process of selecting and prioritizing accounting requirements needs to balance the following forces:

- You can use security accounting to help achieve desired security properties, especially accountability
- Applying accounting has many associated costs (support personnel, software, additional processing time, and so on) that are counter to the organization goal of minimizing total costs
- Collecting extensive relevant raw accounting data increases the likelihood of achieving accountability
- Collecting extensive raw accounting data increases the risk of violating privacy laws, or of abusing such data, or of damaging the reputation of the collector

- The range of time in which accounting may be needed for an event is very broad, ranging from near-real-time to years after the event
- Types of events for which accounting is needed may include repeatable, consistent events, as well as ad-hoc events
- Applying accounting adds complexity to the administration processes, which is counter to the organization goal of minimizing and simplifying administrative and maintenance processes
- Accounting needs to interface with other security services (for example, access control, I&A), thereby increasing the complexity of the software, which is counter to the organization engineering goal to maximize service independence
- Supporting multiple types of accounting policies across an organization increases complexity, which is counter to reducing overall costs
- The elements of the security accounting service need protection if the service is to perform its function

11.5. Solution

Specify a set of accounting requirements for a specific domain such as a system or organization and determine the relative importance of each requirement. The solution has two aspects: a requirements process and a common set of generic requirements.

11.5.1. Requirements Specification and Prioritization Process

A system requirements engineer, in conjunction with an enterprise architect, typically perform the requirements process. An important first step is explicitly to define the domain for which you are specifying security accounting requirements, such as a specific system or facility. You also define factors that affect specialization and importance of requirements, such as organization constraints. Then you specify security accounting requirements for the target domain, using the generic requirements provided below. The final activity is to define the relative importance of the specified requirements.

11.5.2. Generic Requirements Description

Security accounting is a security service that involves the capturing, storage, reviewing and reporting of security-relevant information from an event. The following is a general set of requirements appropriate to security accounting services.

- Provide information about specific events. A security accounting service must allow information to be obtained about events that are undesirable or harmful to the organization. The time of day, day, month, and year are all pieces of information that should be included in details of an event. The details provided by security accounting can either be given as they are captured or made available for scrutiny at a later date. This information will be used to help protect assets by allowing an accounting user to determine what the event was, who was involved, when and where the event happened, why and how the event happened, and how an asset was affected by an event. It also allows actions to be taken to preserve the confidentiality, integrity and availability of an asset based on the type of event.

- Provide information about who engages in activities. This requirement is essential for accounting for user actions. The security accounting service should allow information to be obtained that can be used to establish links between user activity and some event. Security accounting needs to allow its users to determine who the actors are who engage in a malicious or undesired event, and a description of their activities at the time the event was captured. This information will be used to help assign responsibility to an actor for the event and its consequences.
- Provide a degree of confidence that its service will function when needed. This requirement is essential to support security availability. Security accounting needs to be able to provide its services during times when the tracking of events is absolutely important. During operation the security accounting service should be aware of events that could cause significant damage to the organization, and it needs to be able to continue functioning during those high-impact events. Whether the information needed from security accounting is in real-time or nonreal time, security accounting is required to be ready to perform its function.
- Provide a degree of confidence that the information it provides is accurate. This requirement is essential to support integrity. Security accounting should provide information about the accuracy of the data it provides to a user. This information gives decision makers insight into the trustworthiness of the security accounting information.

An additional set of requirements applies to all service requirements patterns. Instead of duplicating the discussion of the same set in each requirements pattern, they are simply listed here, because they do need to be considered in each requirements pattern. The requirements are: minimize time and effort to use, minimize mismatch with user characteristics, risks to user safety, costs of per-user set-up, costs of maintenance, management, and overhead, and changes needed to existing system infrastructure. Further discussion of each of these cross-cutting requirements, including implementation factors, is given in I&A REQUIREMENTS.

The remainder of this pattern focuses on the access control-specific requirements identified and discussed above.

11.6. Structure

11.7. Dynamics

11.8. Implementation

This section first provides more detail about the process summarized in the Solution section, then discusses factors in determining the relative importance of requirements.

11.8.1. Process Guidelines

The requirements process typically includes these steps:

1. Establish the domain for which the accounting service is needed. Ensure that the domain has been identified and scoped. Typical security accounting domains include information system, physical facility, network, portal, or entire organization. The domain consists of at least three parts: a defined scope of actors, a defined scope of assets, and a defined scope or set of events that involve actions on those assets. Note that other terms are also used in place of actor, asset, action. For example, [1] uses subject, object, and operation, respectively. Other constraints may also bound the domain—for example, the accounting requirements for real-time service may differ from those for multi-year service. These might represent two distinct domains.
2. Specify a set of factors that affect the specialization and importance of requirements. The factors include uses of accounting, accounting needs, organization constraints, and priorities. You can find a general candidate set of factors below.
3. Specify accounting requirements for the target accounting domain. To do this, specialize the set of generic requirements given in the Solution section.
4. Define the relative importance of specific requirements.

11.8.2. Requirement Priority Factors and Impacts

Table 10 reiterates the generic requirements described in the Solution section, along with factors for judging their relative importance to the organization. For each requirement, positive and negative impacts of the factors on importance or priority of the requirement are also provided.

Generic Requirement	**Factor**	**Resulting Priority**
Provide information about events (what, when, where why and how)	Required by law or other mandate outside of the organization, or events involve highly sensitive or valuable assets.	High
	Internal organization concern rather than external mandate, or events involve assets of medium value.	Medium
	Only prevention approach used, not detection or response, or events involve low value assets.	Low
Provide information about who engages in activities (who)	Assigning responsibility is a high priority, because it is required by law, or events involve highly sensitive or valuable assets.	High
	Accountability is an organization concern and not a legal or external mandate, or events involve assets of medium value, or losses are covered by insurance, or fall within the boundaries of acceptable risk.	Medium
	No action will be taken against individuals, or events involve low value assets.	Low
Provide a degree of confidence that the service will function when needed	The need for accountability is high, and security accounting is the only source of this information.	High
	The need for accountability is moderate, or alternative sources of accounting information are available.	Medium
Provide a degree of confidence that the	The need for accountability is high, or security accounting information must be provided to an outside organization.	High

information the service provides is accurate	The need for accountability is moderate, and only required inside the organization.	Medium

Table 10: Accounting service requirements importance factors

11.9. Example Resolved

Samuel the museum systems engineer defines several domains because the importance of accounting requirements varies for different asset types. The domains include high value gemstones, the database system that records information about gems, and the physical facilities that house the gem exhibits. Table 11 shows the requirements ratings Samuel has specified for the high-value gems domain. Not surprisingly, all security accounting requirements are rated High for this domain.

Requirement	**Museum Requirement Rating**
Provide information about events (what, when, where why and how)	HIGH – The museum decision makers want to track all activities and events regarding high value gems across the organization.
Provide information about who engages in activities (who)	HIGH – The museum decision makers want information that can hold people responsible for malicious activities regarding high value gemstones.
Provide a degree of confidence that its service will function when needed	HIGH – The museum would like to have a high level of certainty that accounting will perform its function, and specifically requires a 0.9999 availability rating.
Provide a degree of confidence that the information it provides is accurate	HIGH – The museum would like to have certainty that it can rely on the information that security accounting provides.

Table 11: Museum requirements for security accounting service

11.10. Consequences

The following benefits may be expected from applying this pattern:

- It facilitates conscious selection of security accounting requirements, so that decisions about selecting security accounting mechanisms have a clear basis rather than occurring in a vacuum.
- It promotes explicit analysis of trade-offs that encourages balancing and prioritizing of conflicting requirements and forces. This includes balancing the need for accountability with the need for privacy. This helps to avoid stronger than necessary security accounting mechanisms that would make it difficult for valid users, and at the same time it helps to avoid weaker than necessary security accounting that makes it easy for unauthorized actors to avoid.
- It results in documentation of security accounting requirements which communicates to all interested parties and is useful in determining the adequacy of accounting services such as audits.
- The explicit requirements resulting from the pattern foster a clear connection of requirements to security accounting policies: this also encourages organizations to make their accounting policies more explicit.

The following potential liabilities may arise from applying this pattern:

- It requires an investment of resources to apply the pattern, including time to analyze domains and security accounting needs. In some cases, the cost of applying the pattern may exceed its benefits.
- It poses a danger of over-engineering and complexity creep if stakeholders are offered too many options. You can mitigate this by using the requirements only as guidelines for analysis, or by selecting parts of the pattern that give the most help.
- The formal selection process may be too long and costly and produce too much overhead. You can mitigate this in the same ways as noted above.
- Specific circumstances might not be covered by generic security accounting requirements. You can mitigate this by adding specific requirements and including them in the trade-offs.
- Documentation of requirements implies that they must be maintained as they change over time. You can mitigate this by keeping the requirements in a form that is easy to update, integrated with other system documentation.
- Perception of security accounting requirements can differ throughout an organization or in a particular domain. This may make it difficult to reach agreement on the relative priorities of requirements. On the other hand, bringing such disagreements to the surface may be a benefit of the pattern, because then they can be properly discussed and resolved.

11.11. Known Uses

The general accounting requirements and the process of specifying accounting requirements described in this pattern are widely known, but are generally used informally, as opposed to being codified or published. The requirements as stated here represent a consolidation of MITRE Corporation’s experience in working with multiple customers over several decades. However, some publications on accounting requirements also exist.

For example, the Common Criteria [1] is an international standard that defines evaluation criteria for information technology security. It includes some discussion of accounting requirements, especially in the context of the potential conflict between accounting and privacy, or in some cases between accounting and availability. An example of the latter is specifying the required action when an audit trail is full: should you make the associated asset unavailable, or should you retain availability of the asset and allow collection of accounting data to lapse?

11.12. See Also

After applying this pattern, the next step typically is to apply AUDIT REQUIREMENTS, AUDIT TRAILS AND LOGGING REQUIREMENTS, INTRUSION DETECTION REQUIREMENTS, or NON-REPUDIATION REQUIREMENTS.

11.13. References

[1] International Organization for Standardization. (1999). Common Criteria for Information Technology Security Evaluation (Version 2.1). CCIMB-99- 031. ISO/IEC JTC 1 adopted CC 2.0 with minor modifications in June 1999 as ISO/IEC 15408, Version 2.1

11.14. Source

Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., & Sommerlad, P. (2006). Security Patterns: Integrating Security and Systems Engineering (1st ed.). Wiley.

12. Security Needs Identification for Enterprise Assets

12.1. Intent

This is the root pattern for all enterprise security concerns. It helps resolve the issue of whether security is really needed and, if it is, what properties of security should be applied for a particular enterprise. Security properties considered include confidentiality, integrity, availability, and accountability.

12.2. Example

A new wing of a museum of gemstones is to be opened. The museum has significant previous experience handling gems, and theft is a large enough risk for the museum to want protection from the unauthorized removal of any gems. The museum also has information about the collections, and employee information, that should be protected from damage or deletion, and in some cases should be kept confidential. How can the museum determine the assets that need security protection, and which types of protection?

12.3. Context

An enterprise considers security as a significant non-functional requirement. Key business factors and assets of the enterprise are understood.

12.4. Problem

An enterprise that considers security to be important must plan for appropriate security in accordance with the overall enterprise business plans. An enterprise may need to address legacy security plans and policies for the enterprise or develop completely new ones. The same will apply to any information technology (IT) systems that are major assets of the enterprise. For the IT systems, the enterprise may need to adopt an existing security architecture or specify a new target architecture. To determine the most appropriate security to select and implement, the enterprise must establish its validated security needs.

How can realistic enterprise security needs be explicitly identified?

The resolution of this problem is strongly intertwined with the environment of the enterprise, which consists of the following forces:

- The enterprise needs to comply with laws and regulations, such as privacy laws
- It needs to handle sensitive information in a way that protects confidentiality
- It must comply with its own existing policy, especially any security policies
- It needs to provide sufficient protection for mission-critical business assets
- It must ensure that the security employed has minimum potential impact on business efficiency and efficacy—that is, it does not protect more than is necessary
- It must know when undesired events occur
- It must be able to recover from undesired events

- Overall costs need to be minimized

12.5. Solution

Systematically and explicitly identify the types of business assets that need protection and determine the types of protection they need. This activity is typically performed by an enterprise architect or strategic planner, and includes five steps:

1. Identify the business assets of the enterprise:
    a. Information or data assets such as personnel and financial data
    b. Physical assets such as personnel and buildings
2. Identify business factors that influence the security protection needs of assets, both external and internal to the enterprise:
    a. Laws and regulations, such as privacy laws: see [1,2,3,4]
    b. Enterprise partner relationships
    c. Enterprise mission, goals, and objectives
    d. Desire for strong enterprise financial health
    e. Business processes, such as accounting and ordering processes
    f. Sensitive business events, such as the monthly payroll processing
    g. Locations at which business processes and events occur
3. Determine which assets relate to which business factors. Examples include:
    a. A privacy law may apply to employee data
    b. Certain physical asset types may exist only at certain business locations
    c. Selected financial data may need to be shared with an enterprise partner
4. Identify what types of security may be needed: see [5,6,7,8]. Our recommended set is:
    a. Protection against inadvertent or unauthorized disclosure: confidentiality
    b. Protection against inadvertent or unauthorized modification: integrity
    c. Making business assets available for authorized use: availability
    d. Attribution of responsibility for actions: accountability

    Confidentiality, integrity, and availability are the core properties of security literature. Accountability is also important, but it has a different context. Confidentiality, integrity, and availability are attributes of an asset, while accountability is not. When someone is specifying security properties of enterprise assets, it is important to identify who is responsible for security related activities, and that is where accountability comes in.
5. Based on the business factors, determine for each asset type which types of security are needed. The desire for security must be balanced against the resources required to achieve security in making this determination. More details about the association of common types of assets, types of security needed, and business factors are provided in the Implementation section below.

    These steps can be applied in a linear fashion, as listed, but other alternatives are also possible. The Dynamics section discusses allowable sequences.

12.6. Structure

Using the UML class diagram notation, the general relationships among assets, business factors, and security properties are illustrated in the figure below. A security need is an association between an asset type and a security property: each asset type needs a security property. A given asset type may need any number of security properties (0, 1, or multiple), while a given security property may be needed by any number of asset types. Business factors influence security needs. The security properties are listed in the figure, as well as common asset types and business factors:

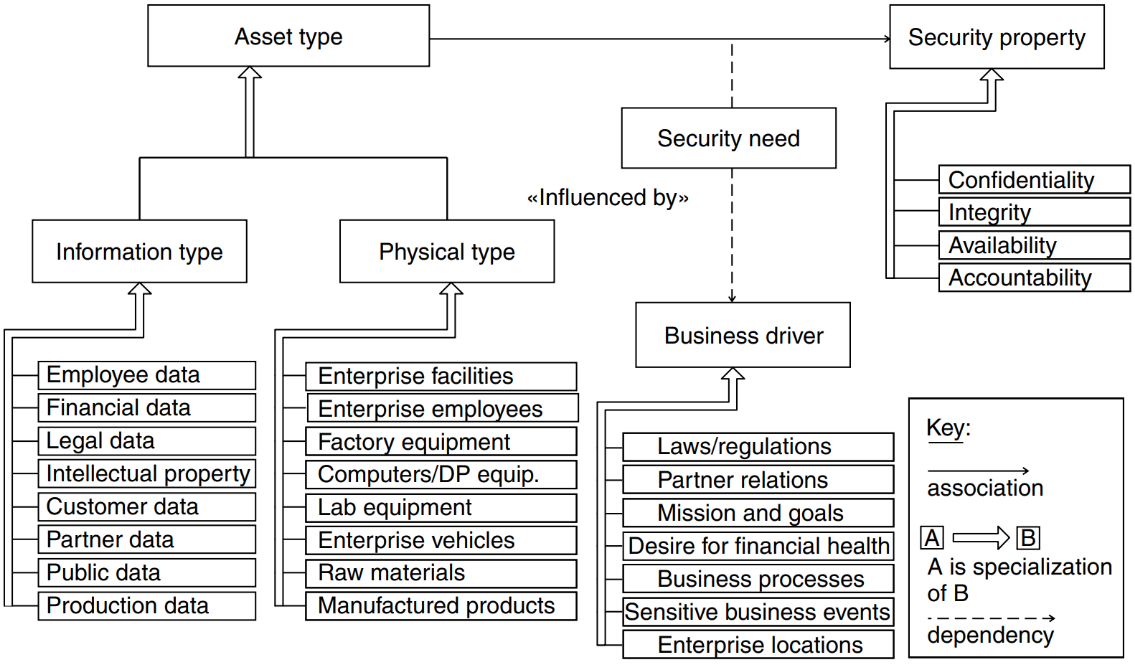

Figure 25: Security needs solution structure

12.7. Dynamics

Allowable sequences for performing the solution steps are shown in figure 28. Identifying assets and identifying business factors are essentially independent activities and can therefore be performed in parallel. However, both activities must be performed prior to determining relationships between assets and business factors. There is often some iteration among these three steps. Defining the set of security properties is also independent and can be performed in parallel with the first three steps.

Defining properties can also be trivial, providing that the enterprise planner agrees with our suggested set of properties: confidentiality, integrity, availability, and accountability. Some enterprises may want to focus on a subset of these, or add related properties such as privacy, safety, or reliability. Several references discuss this issue further: see [5,6,7,8]. In any case, both defining properties and defining relationships must be performed prior to the last step, determining asset security needs.

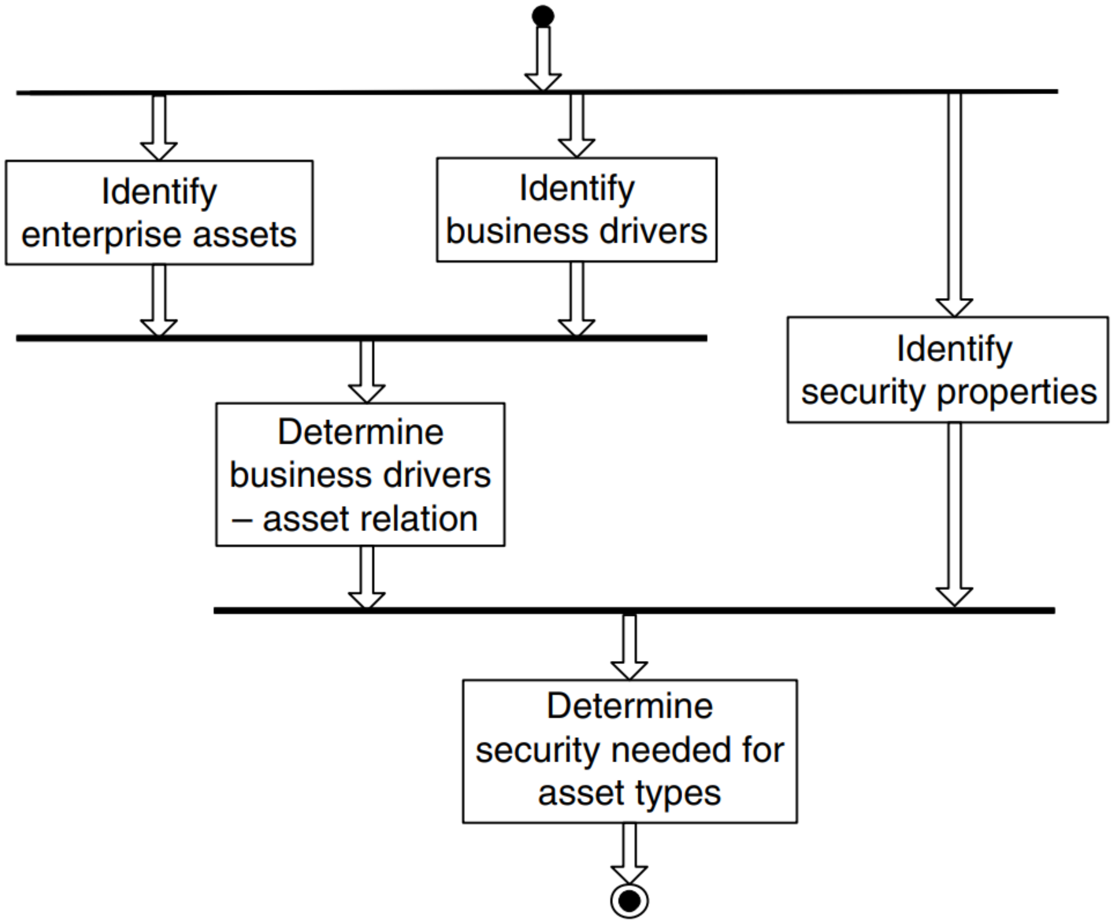
Identify enterprise assets
Identify business drivers
Identify security properties
Determine business drivers – asset relation
Determine security needed for asset types

Figure 26: Security needs solution sequence constraints

12.8. Implementation

Business factors tend to present conflicting forces regarding security. Some, such as laws and regulations, the sensitivity of certain assets, and the desire to be viewed as a secure enterprise, encourage a high level of security. Others, such as cost constraints, the need for financial health, and the desire to be viewed as open and accessible, encourage a minimum degree of security. The result of this trade-off is that assets need to be differentiated according to their importance to the enterprise.

An investment in security is needed for critical assets, while a greater degree of risk may be accepted for non-critical assets. Critical assets typically are those whose loss or damage would cause significant harm to the enterprise, such as assets whose protection is required by law, strategic plans and other assets related to competitive advantage, irreplaceable items, the reputation of the enterprise, or assets whose loss would entail significant cost impact. Non-critical assets are those whose loss or damage would cause little or no harm to the enterprise, such as easily replaceable items, or information that could be divulged with little or no effect.

In addition to criticality of asset, the types of security needed can also vary by type of asset. Confidentiality and integrity typically apply to data. Integrity and availability apply to physical assets as well. Availability applies to services and may also apply to data. Accountability applies to actions taken on assets. To some degree confidentiality conflicts with availability—the more available an asset is, the less confidential it tends to be.

In some cases, one asset or type of asset may require all types of security protection. A software program is an example:

- It may be proprietary, in which case it requires confidentiality
- It needs to be protected against unauthorized change, and it thus requires integrity
- It must be accessible for authorized users, and it thus requires availability
- Any changes made to it must be known and attributed, and it thus requires accountability.

The following tables identify typical asset categories that need protection, the type of security needed to protect the assets, the business factors that influence the need, and some explanatory discussion. The tables provide common examples from an enterprise perspective, but they should not be construed as addressing all possible asset types. Table 12 lists and discusses protection of information assets, while Table 13 lists and discusses protection of physical assets.

In using the above tables, it is important to understand, first, that the information is generated from an overall enterprise perspective, and second, that specific combinations may vary from those in the tables for a given enterprise.

An example will illustrate both of these points. Table 12 indicates that personnel data needs availability, while financial data does not. The reasoning is that, in a typical enterprise, availability of finance information is not needed outside the finance department and the senior officers, while availability of personnel data is needed by multiple parts of the enterprise, such as human resources, finance, training, and security. Clearly the finance department needs availability of financial data, but this pattern is an enterprise-level pattern, and across the typical enterprise availability of financial data is not a significant issue. In addition, this table is only representative of common associations. There may be variations for specific enterprises—each will have its own business processes that may differ.

Asset Type	**Protection Needed**	**Business Factors**	**Discussion**
Personnel data (including payroll)	Confidentiality, integrity, availability, and accountability	• Privacy laws • Competition issues	Privacy law will require that personnel private information be treated confidentially. Enterprise staff will need assurance that only human resource staff can modify their information. The data will need to be available to human resource staff as needed, and to financial staff to support payroll. Changes to personnel data must be accountable within the enterprise.
Financial data (enterprise financial data)	Confidentiality, integrity, accountability	• Reporting requirements of tax collection agency • Competition issues	Financial laws and the regulations of government agencies must be upheld in the enterprise or legal repercussions will ensue. Such laws and regulations will require that the financial data be protected from unauthorized modifications and that when modifications occur, there is a

		• Nature of the enterprise (public, private, or stockheld)	clear record of accountability in the enterprise. No enterprise willingly provides its financial data to its competition; the confidentiality of this information must be protected.
Legal data (for example, contracts and information on legal proceedings)	Confidentiality, integrity, accountability	• Law • Competition issues	An enterprise will need to provide confidentiality under contract law that may also require confidentiality of information related to participants in the contract. The modification of such contracts should be restricted to authorized and knowledgeable personnel and there should be a clear record of accountability in the enterprise.
Intellectual property (data and processes)	Confidentiality, integrity, availability	• Partially dependent on the nature of the enterprise (public, private, stock-held) • Some competition issues	While some intellectual information (for example, advertisements) will be for the public, others, such as sensitive business processes, will not. Sensitive intellectual property may need restricted access. At the same time, if the business process contains design specifications, it may also need to be highly available within the enterprise.
Customer and business partner data (including personal and financial data and intellectual property)	Confidentiality, integrity, accountability	• Competitive issues • Service issues if a public company	Enterprise privacy information may be contained in this data. If competitors are aware of the relationships with customers and business partners, they can cause an enterprise to lose its competitive edge. Access to all customer and partner data should be accounted for to ensure that it is not altered in unauthorized ways, and that access to the data is restricted.
Public data (product/service information, advertisements, public enterprise information)	Integrity, availability	• Service issues	Unauthorized modification of the data could result in loss of enterprise reputation and/or business share. When such public information is made unavailable, a denial-of-service situation arises.

Table 12: Common information asset categories and protections

Asset Type	Protection Needed	Business Factors	Discussion

Buildings	Integrity, availability	Critical business processes	An enterprise needs to protect the buildings that provide a work environment for the enterprise from unauthorized modifications or destruction. By doing so, they also promote the availability of the buildings for the enterprise.
Employees	Availability, accountability	Critical business employees and processes	An enterprise needs to provide environments that are safe for personnel to ensure the availability of critical personnel. In part they accomplish protecting personnel by establishing accountability for employees.
Raw materials/ durable goods/ manufactured products	Integrity, availability	Need to minimize the cost of doing business	Raw materials and durable goods need to be available for use in business processes as required. The enterprise needs to be able to assure its client base that manufactured products will be available as required. Damage, theft, or destruction of raw materials/durable goods will make them unavailable to support business processes. Likewise, damage, theft, or destruction of products will make them unsalable to clients.

Table 13: Common physical asset categories and protections

12.9. Example Resolved

This example solves the problem identified as the problem example described earlier. The museum enterprise identifies the following asset types and business factors:

- Information asset types
    - Museum employee data
    - Museum financial/insurance data, partner financial data
    - Museum contractual data and business planning
    - Museum research and associated data
    - Museum advertisements and other public data
    - Museum database of collection information
- Physical assets
    - Museum building
    - Museum staff
    - Museum collections and exhibits
    - Museum transport vehicles
- External business factors
    - Insurance policy constraints
    - International laws and agreements relative to on-loan materials
    - Privacy laws
    - Museum charter
    - Goals and strategies relative to exhibits
    - Loan of materials and accessions (acquisitions)

    - Requirements or constraints of organizations that loan materials
- Internal business factors
    - Tracking of exhibit items/cataloguing
    - Item data, including location and value (both a factor and an asset)
    - Exhibit planning, including loan agreements, transport and installation plans and schedules, legal contracting with exhibitors
    - Accession (acquisition) planning, via purchase or loan or gift
    - Legal data (acquisition)
    - Cost constraints, including funds available for acquisition, personnel, and patron data (including donation amounts), financial data (how much depends on charter: public, private, semi-public, and so on), and cost of security
    - Intellectual property, such as studies and research data, statistics, and papers
    - Public information, including hours and current exhibit schedules (near term) as well as brochure and exhibit publications
    - Building plans
    - Importance of enterprise reputation for security
    - Importance of enterprise reputation for accessibility
    - Sensitive business events, including accession of new items, asset transport to alternate locations, cleaning/caretaking of assets, and special temporary accession for on-loan exhibits

The planner generates a scope statement listing all the above information. The scope statement will be presented to and refined with the museum director. Together they will work to generate an asset protection list such as that shown in Table 14.

Asset Type	**Required Security Properties**	**Business Factor**
Museum employee data	Confidentiality (HR, management, individual), Integrity (HR, individual only), Availability (HR and management), Accountability (changes in HR)	• Privacy law • Enterprise/employee relations
Museum financial/ insurance data, partner financial data	Confidentiality, Integrity, Basic accounting	• Contractual obligations • Financial reporting laws
Museum contractual data and business planning	Confidentiality, Integrity, Basic accounting	• Museum/partner relationships • Protect acquisition and transport plans and strategies • Protect scheduling data • Insurance policy constraints
Museum research and associated data	Confidentiality (restricted to narrow group)	• Museum charter requirements • Intellectual property • Enterprise/employee relations • Enterprise/public reputation
Museum advertisements and other public data	Integrity	• Enterprise/public reputation • Museum charter requirements • Partner reputations for loan exhibits

Museum building	Integrity, Accountability (for any change)	• Insurance policy constraints • Enterprise/employee relations • Enterprise/public relations
Museum staff	Availability (safety)	• Enterprise/employee relations • Laws • Enterprise/public reputation
Museum collections and exhibits	Integrity, Availability, Accounting	• Insurance policy constraints • Enterprise/partner relations • Costs

Table 14: Establishing security properties for the museum

12.10. Consequences

SECURITY NEEDS IDENTIFICATION FOR ENTERPRISE ASSETS has the following benefits:

- It facilitates making balanced and informed decisions about enterprise security needs, by making the competing forces and business factors explicit. The tradeoffs in these factors cause a clear distinction to be made between critical and non-critical assets. The result is increased likelihood that security properties will be applied where needed. That is, protection needs will be explicitly designated for the most critical assets.
- An additional beneficial result of applying this pattern is that traceability of business asset protection needs back to the relevant business factors is produced and is available for additional use. This information offers a useful rationale to support the evolution of security needs over time. It can also be used, as indicated above, as a basis for more detailed protection requirements in ENTERPRISE SECURITY SERVICES.

SECURITY NEEDS IDENTIFICATION FOR ENTERPRISE ASSETS also suffers from the following liabilities:

- Applying this pattern does not come free of charge. It requires an investment of resources, including the time of people who have intimate knowledge of enterprise assets and business factors. While the benefits of applying the pattern are expected to exceed these costs, it is possible for an enterprise to assign people to this task who have less than adequate knowledge of enterprise assets and business factors, and thus to obtain results that are inaccurate or not useful.
- It is also possible for an enterprise to produce good results from this pattern, but then fail to make use of the results in succeeding patterns.

In both cases, the cost of applying this pattern can exceed the benefits.

12.11. Known Uses

Identification of enterprise assets and their security needs is best practice but is often done informally or as part of security risk analysis. A few examples that illustrate concepts in this pattern—and in some cases were sources for the guidance in the pattern—are briefly discussed here.

The Systems Security Engineering Capability Maturity Model (SSE CMM) [6] defines capability levels of a security engineering process, associated with risk assessment. It has elements in common with this pattern:

- It addresses security across the scope of the enterprise
- It addresses coordination of security needs driven from external entities, including laws, policies, standards
- The assess impact process includes identifying and characterizing enterprise assets and the need for confidentiality, integrity, availability, accountability, authenticity, or reliability.

PriceWaterhouseCoopers has a process for designing an enterprise security framework that incorporates many of the elements of this pattern [9]. It tailors a security process based on the business requirements and factors of an enterprise. It includes asset inventory collection and information classification.

Mint Business Solutions has defined an approach to security in the context of best practice in enterprise information management [10]. Within this broad framework, they base their security approach on the ISO standard 17799. This standard identifies a set of controls that include:

- Organization of assets and resources, with relation to managing information security
- Asset classification and control, so that they may be identified and protected
- Information security policy
- Compliance with any criminal and civil law, statutory, regulatory, or contractual obligations, and any other security requirement

These controls correspond to the identification of assets and business factors for security addressed in this pattern.

Other standards and practices, including ISO 13335 Part 3 [11], SANS Institute [12], and Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) [13], discuss asset identification as part of the overall risk analysis process. OCTAVE is an asset-driven evaluation approach that requires an analysis team to identify information-related assets such as information and systems that are important to the organization.

12.12. See Also

After applying this pattern, the next step typically is to apply a set of risk-assessment patterns to further calibrate the security needs of each asset type to determine more specific security requirements. Risk patterns help assist in deciding how much protection is needed for each business asset type. While SECURITY NEEDS IDENTIFICATION FOR ENTERPRISE ASSETS is somewhat internally focused, the risk assessment patterns include both internal and external considerations.

Following risk assessment, the next step is to assess enterprise security approaches that meet the combined security needs and requirements from this pattern and from the risk assessment.

12.13. References

[1] Department of Justice. (2004, May). Overview of the Privacy Act of 1974, US DoJ, Washington, DC. http://www.usdoj.gov/04foia/04_7_1.html

[2] European Union. Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the Protection of Individuals with Regard to the Processing of Personal Data and on the Free Movement of Such Data, Official Journal of the European Communities, No. L 281, 23rd November 1995, pp. 31–50

[3] European Union. Directive 2002/58/EC of the European Parliament and of the Council of 12 July 2002 Concerning the Processing of Personal Data and the Protection of Privacy in the Electronic Communications Sector (Directive on privacy and Electronic Communications), Official Journal of the European Communities, No. L 201, 31 July 2002, pp. 37–47

[4] U.S. Congress. FOIA – The Freedom of Information Act, 5 U.S.C. § 552, As Amended by Public Law No. 104-231, 110 Stat. 3048, U.S. Congress, Washington, DC, 1996

[5] International Organization for Standardization. (1999). Common Criteria for Information Technology Security Evaluation (Version 2.1). CCIMB-99- 031. ISO/IEC JTC 1 adopted CC 2.0 with minor modifications in June 1999 as ISO/IEC 15408, Version 2.1

[6] Carnegie Melon University (CMU). (2003, June). Systems Security Engineering Capability Maturity Model, Model Description Document, Version 3.0. http://www.sse-cmm.org

[7] DiDuro, J., Crosslin, R., Dennie, D., Jung, P., & Louden, C. (2002). *A Practical Approach to Integrating Information Security into Federal Enterprise Architecture*. LOGISTICS MANAGEMENT INST MCLEAN VA.

[8] National Security Agency (NSA). (2002, September). Information Assurance Technical Framework (IATF), Release 3.1. Fort Meade. MD. http://www.iatf.net

[9] PriceWaterhouseCoopers. (2001). Designing a Secure Framework for the Competitive Enterprise.

[10] R. Leming. (2003). Best Practice in Information System Management. Mint Business Solutions. http://www.mintsolutions.co.uk/pages.asp?p=43

[11] “Technical Report ISO13335-3 Part 3: Techniques for the Management of IT Security” International Organization for Standardization, American National Standards Institute, 2002

[12] Bayne, J., SANS Institute. (2002). An Overview of Threat and Risk Assessment, Version 1.2f. http://www.sans.org/rr/paper.php?id=76

[13] Alberts, C. J., Dorofee, A. J. (2001, December). OCTAVE Criteria, Version 2.0. http://www.sei.cmu.edu/pub/documents/01.reports/pdf/01tr016.pdf

12.14. Source

Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., & Sommerlad, P. (2006). Security Patterns: Integrating Security and Systems Engineering (1st ed.). Wiley.

13. Security Session

13.1. Intent

Verifying a user's identity and access rights for every system function can be tedious. To keep track of who is using the functions and their corresponding access rights, systems establish a security session after a user has logged in successfully. A unique reference to the session object is made available, instead of passing all access rights or re-authenticating a user repeatedly. Queries regarding a user's security properties are delegated to the attached session object via the session reference.

13.2. Example

Our medieval city is concerned about foreigners entering through its gate. Merchants are welcome, but burglars and thieves shouldn't be let in by the guards. Peasants looking for work in one of the city's workshops are allowed in depending on the demand of the guilds. On the other hand, once a person has entered the city, it is hard for the city dwellers to tell who that person is if they are not a well-known city dweller. Even the night watchman patrolling the city's streets has a hard time knowing how to deal with a stranger. A merchant should be welcomed and protected, while someone else lingering in the streets at night might need to be dealt with.

The problem of city inhabitants and the night watchman is that they do not have equivalent resources to the guards at the city gate, to interrogate and investigate people and check their identity. In addition, it would be annoying to visitors that are welcome if they had to answer the same questions over and over again, such as who they are, where they are going, and what their business is in the city.

The mayor summons the city council to discover a way to keep the city secure while making the city a welcoming as possible for merchants and other guests. Another requirement that comes up at the council meeting is that the city officials would also like to know when a visitor has left.

13.3. Context

Your system is shared by multiple users and system components need a way to share (security) data associated with a user. For example, you have already applied CHECK POINT.

13.4. Problem

Systems shared by multiple users, either via terminals or via a network, have become commonplace. Instead of single-user non-networked systems like the—now almost extinct—DOS PCs, shared or networked systems need to account for a user's actions and ensure users only have access to areas for which they have privileges. A user therefore needs to be identified and authenticated by the system. In addition, shared resources require controlling access to them.

Different components acting on behalf of a user might need to know which user is activating them and what the user’s permissions are. Having every individual component or program within the system identifying, authenticating, and authorizing users is annoying to both users and developers. In addition, system components might call each other or work together, and thus need a way to share information about the user without compromising this global data to other users.

For example, when buying from a Web-based on-line store, you want to put items in a shopping cart that is associated with you. Later, the check-out process requires you to approve your credit card information and delivery address. The underlying protocol (HTTP) does not provide a context for multi-step interaction because it is stateless. Accordingly, the on-line store’s software needs to associate every click you make in your browser with your identity, your shopping basket contents, and your billing information. In addition, you want the system to forget your credentials after the transaction is complete, either by an explicit sign-off mechanism or by a time-out after no interaction by you, thereby ensuring that forgetting to sign off will not compromise your private data such as your credit card account number.

How do you provide easy access for system components to the security properties and other values related with the current user, without requiring them to identify and authenticate every time they interact with the system or an individual component?

The solution to this problem must resolve the following forces:

- You need to provide access to global values shared by different system components. These values also need to be distinguished for individual users. Simple global variables will therefore not work.
- Such values might change during a user’s interaction and might be different between several activity periods of the user’s session—for example, the contents of the shopping cart in the on-line store.
- Different components or applications within your system can be interested in different values and might want to change them or define new ones.
- Passing all shared values around the system for a given user can cause too much overhead and result in bloated interfaces.
- Asking a user for I&A over and over again is annoying, so the system needs a means of associating an action automatically with a user that was previously authenticated.
- After a long period of inactivity, the system needs to re-authenticate a user to prevent misuse and overhead. In other words, the system should automatically sign off inactive users.

13.5. Solution

Introduce a session object that holds all user-relevant shared data. Security information related to the user, especially, is kept in the session object. In addition to the session object that holds the values, the system needs to associate every action a user makes with this session. This can be either implicitly, such as associating every action coming over the user’s connection with the session, or explicitly with an identifier like a session cookie that is sent to a user’s browser by an on-line store site.

A system’s CHECK POINT is the usual place to instantiate the session object and set up its initial values. For systems with access control, the session object can be used to obtain access permissions at sign-on and cache them, to avoid multiple queries to an external database.

In addition, the session object can provide a scratch-pad area to allow different system components to share arbitrary data about the user between different actions within a log-on period. For example, classic mainframe systems provide a so-called ‘terminal control block’ that can be used by different interactive transaction programs to share data, such as the last values entered in a form. This allows otherwise independent transaction programs to be chained easily on behalf of the user. Web applications use cookies to share data for a given user.

Often a MANAGER [1] is used to keep track of active session objects and controls their life cycle. This MANAGER can also be used to provide the mapping of external session identifiers, such as those stored in a session cookie, to the session object and its data. Furthermore, it can collect obsolete sessions that were abandoned by their users.

13.6. Structure

The following diagram shows the component relationships assuming that there is a MANAGER. The CHECK POINT uses the MANAGER to associate a Session object with the user. Later, the components accessed by the user rely on the associated Session object to access the user’s access rights and further information. The Manager class uses the timestamp to keep track of stale session objects and forces the user to re-authenticate if a session is either used for too long without authentication, or if it has not been used for a longer time.

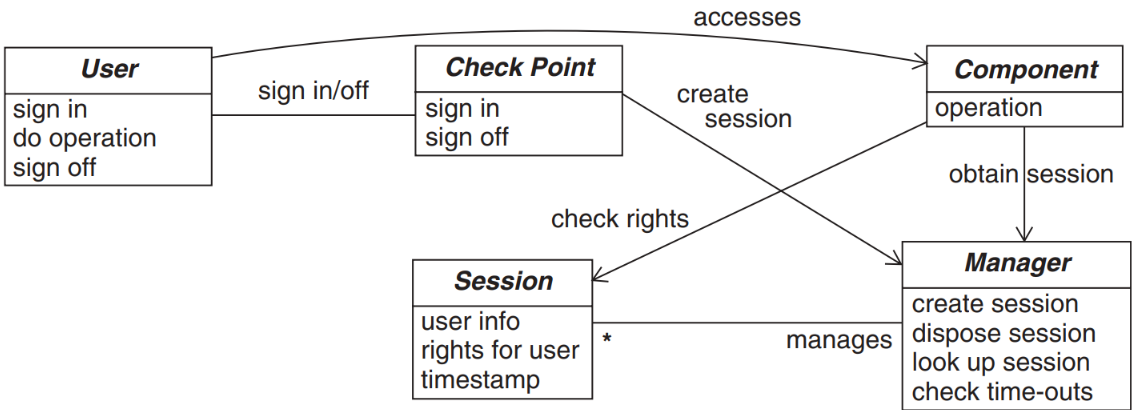

Figure 27: Security Session structure

13.7. Dynamics

The following scenario shows a simplified interaction, in which a user logs into the system via the CHECK POINT. The CHECK POINT uses the MANAGER to obtain a new Session object for the user. The manager does not return the object directly, but instead returns an external session identifier to be used by the user or CHECK POINT for later reference. This scenario assumes that the user explicitly provides his session identifier instead, as would occur with a Web application’s session cookie. Other systems can provide an implicit association of a user with his session object—this is not shown here.

Later on, the user interacts with a system component, providing his session identifier for reference. The system component authorizes the user by asking the MANAGER for the underlying session object and checking the user's data stored there.

When the user logs off at the CHECK POINT, the MANAGER deletes the Session object belonging to the user, invalidating the corresponding session identifier, which no longer can be used. See figure 30.

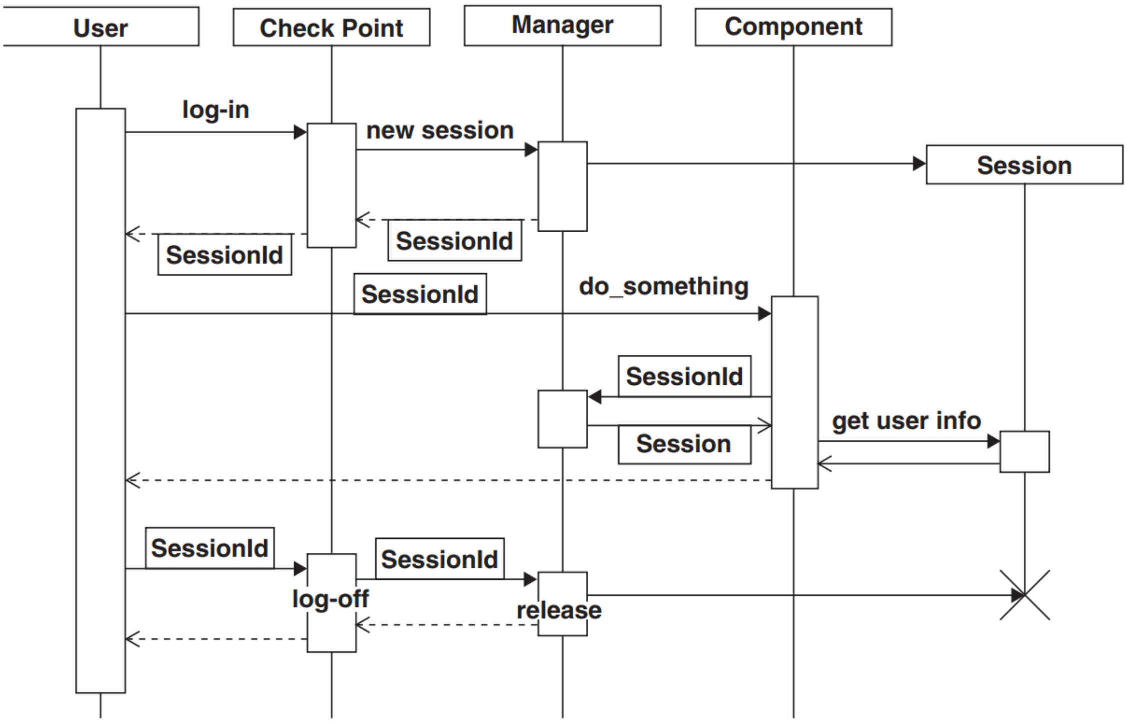

Figure 28: Security Session sequence diagram

13.8. Implementation

To implement SECURITY SESSION several tasks are required:

1. Create a session object to hold all (security) variables associated with the user that may be needed by other components. Typical information kept in the session object are the user's identification, their access rights, the user's role (see ROLE-BASED ACCESS CONTROL), and other system- or application specific data, such as a shopping cart's content. In addition, you should add a timestamp when the user logged in successfully, and a timestamp of the user's last activity. For a flexible solution you might use a data container like PROPERTY LIST [2] or an ANYTHING [3] to keep track of varying data without changing code. Web applications might opt to keep the session data in a cookie in encrypted form. Even when just storing the session identifier in a cookie or URL to keep track of users, such Web applications must ensure those identifiers are not easily guessable, to limit the risk of session hijacking.
2. Introduce a MANAGER and unique session identifiers to keep track of active session objects. If a user is only allowed to log in once, you might use a user's identification as the session identifier—otherwise a synthesized identifier is sufficient. A publicly accessible session identifier must be protected against fraud, which in many cases disallows the user's identifier from use directly as their session identifier. Apply

MANAGER [1] or RESOURCE LIFECYCLE MANAGER [4] as a reference for implementing the MANAGER. The MANAGER provides an interface for other system components to access a session object corresponding to its identifier.

3. Define session time-out semantics. Lingering unused session objects carry risk, not only for security reasons, but also for memory management. The MANAGER should periodically check for inactive sessions and release them. If this inactivity time out is short, it effectively prohibits misuse of a session by an unidentified user. On the other hand, if it is shorter than the typical transaction time of a user, such a session time-out gets annoying.
4. Define re-authentication time-out semantics. In security-sensitive environments, the MANAGER should also enforce re-authentication at the CHECK POINT for long-lived active sessions to protect a user's session from misuse and the user from forgetting his password. Appropriate values for such time-outs depend heavily on the given domain and use profile. For example, Yahoo! uses cookies that live for about five years to identify a user. However, from time to time, and whenever accessing sensitive data, a user needs to reauthenticate. In a system in which access rights management is separate, this re-authentication also provides a means of updating a user's access rights that are cached in the session object.
5. Allow a user to log on and log off at the check point. Even though it seems trivial, you shouldn't forget to provide the mechanism that allows a user to establish the security session and to allow them to cancel a session of their own will. This actively allows the user to care about security, which can be an important security measure. During log-in the MANAGER creates and initializes the session object with the user's access rights and other relevant data.

13.9. Example Resolved

Our medieval city council comes up with the concept of a day pass. This day pass is issued by the gate guards to every foreign visitor entering the city and needs to be returned when leaving the city. To distinguish the more desirable guests from less liked ones, the passes are color coded: peasants looking for work get a green pass, the private visitors of city inhabitants a white one, while merchants receive a bronze pass that they can keep for later visits.

A visitor is obliged to show his pass to everybody asking for it. Since the city is small enough for all citizens to know each other, the city does not need to issue passports for its inhabitants.

13.10. Consequences

The following benefits may be expected from applying this pattern:

- The session object provides a single, well-defined place to keep user and security-related data.
- Instead of passing different values around, the system can pass the single session object around for a user.
- Extending the session object to hold new data is straightforward and can be done without impacting unrelated system components.
- The system can use the session object to cache access permissions, thus improving performance.

- It is easy to externalize a session object's identifier when no implicit association between a user and a session object can be achieved, such as with a Web application.
- Checking the associations between sessions and users allows detection of multiple simultaneous uses of the same user credentials, which can be a security compromise, for example, if a user's password is used by several people.

The following potential liabilities may arise from applying this pattern:

- Developers thinking in terms of global variables, such as those the session objects provide, can imply badly structured programs and uncoordinated or hidden coupling of system components.
- Keeping too many too large session objects around can limit system performance. Special means for collecting session garbage, for example session timeouts, might need to be implemented if users cannot be coerced to log off, or if a single user can initiate multiple sessions simultaneously.
- In a distributed system, session identifiers might be forged by attackers and thus lead to security compromises. Careful design of non-guessable and nonenumerable session identifiers is therefore a must. However, providing such session identifiers must be automatic, or at least easier for a user than providing their original credentials.
- If all session data is sent to the user's browser in a cookie instead, the cookie needs to be encrypted and signed to avoid security compromises of the session data. Again, authenticating the cookie sent by a user must be easier for the user than providing their original I&A information.
- System components that initially do not need the session object might still keep a reference to it, since components instantiated or called might require it.
- Retro-fitting session objects to a (badly designed) system relying on SINGLETONS or global variables can be difficult.

13.11. Known Uses

Netscape invented cookies as a means of keeping track of a user's session via the otherwise stateless HTTP protocol [5]. A user's browser automatically returns a cookie to the originating Web server, effectively passing the session object without the user needing to care about it. Cookies are the means of session tracking for Web applications.

Operating systems such as Unix or Windows use an implicit session object associated with each process in the system. This session object is copied or inherited when a new process is created by its parent process, and only privileged processes such as Unix' log-in program are allowed to set the corresponding session data. For example, in Unix this session object holds the user id and group id of the process owner, among other data. These two imply the corresponding permissions of the process.

Many classic Internet protocols such as FTP and Telnet, as well as many database systems like Oracle or MYSQL, use the TCP/IP connection between a client and a server as an implicit session mechanism. Each session is thus represented by an individual TCP connection. The termination of the connection also terminates the session. Operating systems and the 16-bit port numbers used in IPv4 place a hard limit on the number of usable sessions.

The open-source implementation of SSL (Secure Sockets Layer), openSSL, uses a session id to avoid the expensive re-negotiation of certificates, encryption algorithms, and encryption keys for connections re-established between the same client and server. Some Web security systems use this SSL session mechanism instead of cookies to associate a security session with a user for HTTPS.

13.12. See Also

CHECK POINT typically relies on SECURITY SESSION to provide sign-on functionality for users. If a check point protects multiple systems and those share a single user session, it can provide effective single sign-on for users.

The session object plays the role of an ENCAPSULATED CONTEXT [Kell03] holding several parameters related to the user and their access rights. It is passed through the system as a single parameter, and components of the system can access the encapsulated data via the session object. The ENCAPSULATED CONTEXT avoids wide parameter lists for methods, and ripple effects on changing interfaces when additional user or session-related data is needed.

INTEGRATION REVERSE PROXY and FRONT DOOR rely on SECURITY SESSION to keep track of Web users. They implement it via cookies, SSL session ids, or by encoding the session identifier into URLs.

13.13. References

[1] Sommerlad, P. (1997). Manager, Addison-Wesley

[2] Joseph, B. F., & Yoder, J. (1998). Metadata and Active Object-Models. In *Dept. of Computer Science, Washington University Department of Computer Science*.

[3] Sommerlad, P., & Ruedi, M. (1998, July). Do-it-yourself Reflection. In *EuroPLoP* (pp. 337-366).

[4] Kircher, M., & Jain, P. (2004). Pattern-Oriented Software Architecture: Patterns for Resource Management.

[5] Kristol, D., & Montulli, L. (1997). RFC2109: HTTP state management mechanism.

13.14. Source

Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., & Sommerlad, P. (2006). Security Patterns: Integrating Security and Systems Engineering (1st ed.). Wiley.

14. Single Access Point

14.1. Intent

If you need to provide external access to a system, but want to protect it from misuse or damage, define a single access point that grants or denies entry to the system after checking the client requiring access. The single access point is easy to apply, defines a clear entry point to the system, and can be assessed when implementing the desired security policy.

14.2. Example

Consider a small medieval village. It consists of a group of houses in close proximity. While there is little economic prosperity, there is little value in protection from burglars and little interest to robbery. Security for people means that each man protects his own belongings. Each man spends time building weapons and training in their use so that he can defend his family.

Somehow economy prospers (we do not speculate why and how), more and more people move to the village, and more and more value is accumulated within the village. However, since the people need more time for their prospering businesses, they have less time to spend practicing defense. Their level of protection becomes lower, while the threat from burglars grows. In addition, visitors must convince every individual shopkeeper that they are valuable customers instead of thieves before they can actually conduct business with them. The village dwellers wonder how they can simplify protection, so that everybody no longer needs to deal with it many times a day.

14.3. Context

You need to provide access to a system for external clients. You need to ensure the system is not misused or damaged by such clients.

14.4. Problem

Whenever a system is used by an external client such as a user, the system’s integrity is in danger. Often such systems require some security property, like protection from misuse or damage. One means is to check every interaction with an external client to determine whether it is authorized. When the system has a non-trivial inner structure and consists of multiple parts or subsystems, an external interaction of the system can result in many different interactions of the client with the individual parts of the system. Checking each of these sub-interactions is required to protect all the parts, and thus the whole system. First, implementing all these checks can be a burden: second, if the same information has to be presented over and over, these checks can hinder performance and annoy a user: third, assessing the correct implementation of the overall security policy is hard, because of its complexity.

In addition, it is a good practice to have a clearly defined entry point to a system, as you are used to with the main entrance to a building. Such a prominent and well-known entry point makes using the system easier, because we do not need to spend time searching for the entrance.

The solution to this problem must resolve the following forces:

- You need to provide access to a system to make it usable.
- In a complex interconnected world, no system is an island.
- Most systems exhibit a non-trivial structure and are constructed from sub-systems that also need protection.
- Many entry points to a system reduce security because the additional complexity makes it easier to bypass controls.
- Multiple entry points can have duplicate code for the same kind of checking.
- Repeated checks annoy clients or slow down the system.
- Uniform access to a system can lowers its usability if different situations really require different means of access—for example, entering the Windows log-in password on a tablet PC is annoying if no actual keyboard is present on which to type it.
- Uniform access to a system is easier to control.

14.5. Solution

Define a single access point for clients using the system. At this access point you can check the legitimacy of the client according to your defined policy. Once clients passed the access point, they are free to use the system from that point on.

Protect the rest of the system's boundary, so that no circumvention of the single access point is possible. The inhabitants of our medieval village build a wall to serve as such a passive boundary protection: a computer operating system denies all activity from anonymous users not logged in.

Make the single access point prominent, so that it is easy to find and absolutely obvious where to enter the system. Nothing is more annoying than circling the city wall looking for a gate. A computer operating system usually shows a log-in window or prompt when no user is currently logged in.

If auditing is required, the single access point can record which clients entered the system and when. It might also record the termination of a client's use of the system.

This pattern applies to many levels of abstraction and technology. It further might apply within a more complex system to the system itself as well as its subsystems, which in turn can have additional single access points.

The above description is very generic: here is a list of some concrete examples:

- Function entry point with precondition check (for example in Eiffel)
- Windows operating system log-in screen
- Firewall patterns
- A security guard in an office building or military base

14.6. Structure

The single access point can be represented by the following UML diagram.

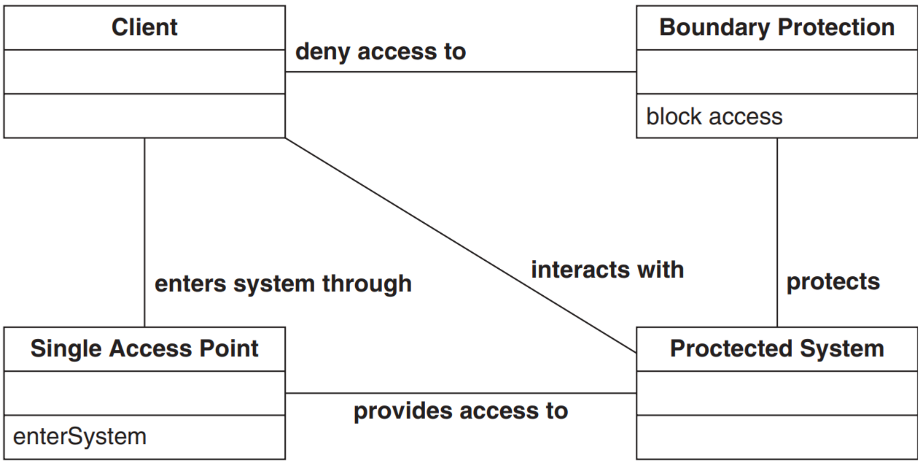

Figure 29: Single Access Point structure

However, it is more intuitive to presented it as shown in the accompanying sketch, since it is hard to show the boundary protection of the protected system. Boundary protection is essential to make the single access point efficient in checking clients and hindering intruders to access the system.

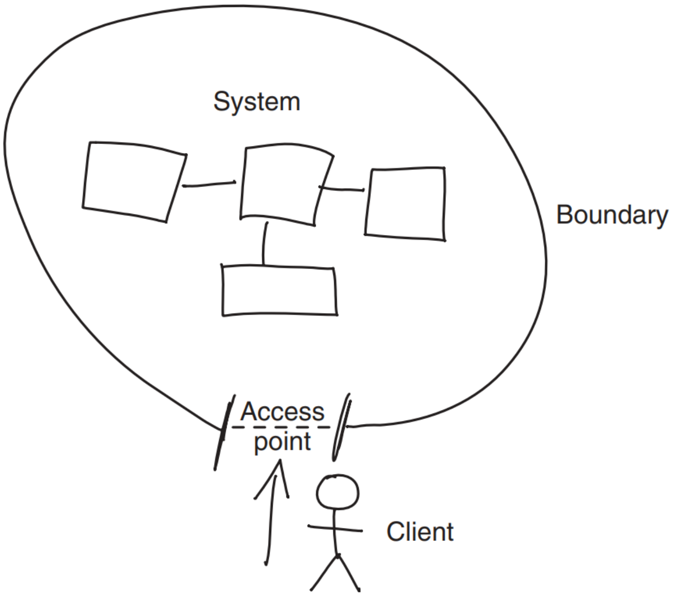

Figure 30: Single Access Point sketch

14.7. Dynamics

The sequence diagram illustrates a regular scenario of a client entering the system. The client logs in at the single access point and then uses the protected system. The passive protection given by the boundary (the city wall) cannot be shown here.

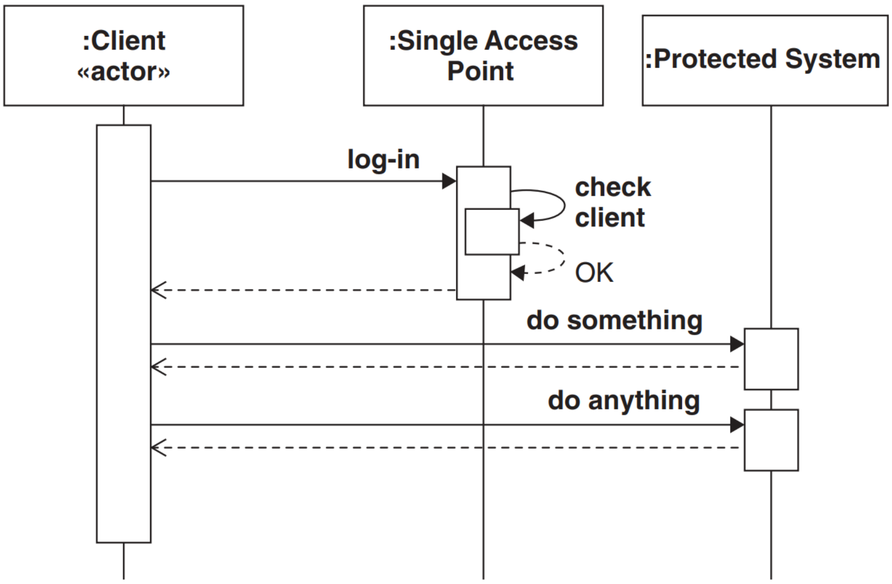

Figure 31: Single Access Session sequence diagram

14.8. Implementation

To implement the SINGLE ACCESS POINT, several tasks are required:

1. Define your security policy for the system at hand. Before you start securing your system, you should know what you secure and why. Apply the patterns from this book to obtain the security requirements for the system to be protected. The security policy must contain the trust relationship between the internal subsystems. All of them need to trust the single access point and also each other. Even if such trust can be established, it might be wise to apply the Defense in Depth security principle for extra sensitive subsystems.
2. Define a prominent or well-known position for the single access point or make it transparent for its legitimate users. Christopher Alexander's MAIN ENTRANCE [1] gives some guideline about where to place your main entrance, which SINGLE ACCESS POINT definitely is. He writes, 'Therefore: place the main entrance of the building at a point where it can be seen immediately from the main avenues of approach and give it a bold, visible shape which stands out in front of the building.' Microsoft Window's classic log-in screen with its 'Press ALT-CTRL-DEL to log in' is definitely not following the essence of that rule.

   Another option is to make the single access point invisible, but impossible to circumvent. This makes the access transparent for clients, but nevertheless allows the system to be protected. The firewall patterns in this book are examples where the single access point is made transparent for legitimate uses but impassable for intruders.
3. Optionally implement the entry check at the single access point. If your system's security policy requires authentication and authorization, the single access point can play a major role in implementing it easily. A typical software system will provide a log-in window for the user to provide an identifier and password. If both match with corresponding stored values, the user is allowed to use the system. CHECK POINT shows how to make this checking flexible.

If you apply CHECK POINT, you can also associate a SECURITY SESSION, or a so-called ‘day pass,’ with the client. Every client showing such a day pass is automatically trusted within the system once past the single access point. The single access point will initialize parameters and variables within the client’s session to valid values on which the system can rely. In simple cases the ‘day pass’ can be implicit, by letting the client enter the system and trusting the boundary protection to hinder intruders.

4. Implement the system initialization at the single access point. Some protected systems need to be initialized corresponding to their user before they can be used. For example, the Unix log-in program initializes the user’s process with their user and group identities, thus enforcing correct authorization later on. It also presets environment variables, as predefined by the system, and executes an initialization script before the user can start working with the system. SECURITY SESSION shows details of how to identify the user throughout their use of the system and keep their related data in a convenient place.
5. Protect the boundary of your system. The single access point can only be effective if you provide a closed perimeter to your system. Especially, you need to look for potential ‘back doors’ that have been left open. It is in the sense of SINGLE ACCESS POINT not to have these. For example, when you set up a firewall, you ensure that only those ports that are actually needed are open: all other network connections are disabled.

The boundary protection can be physical, like a city wall, or built into the system itself, such as operating systems not allowing anonymous users to start processes other than via the log-in program.

14.9. Example Resolved

The village people build a wall around their dwellings, effectively making it a walled town with a gate. The wall hinders burglar’s access to the town, while the gate allows customers and townspeople to enter and leave the town. A single guard at the gate is now able to protect the whole town. The townspeople acquire more time for business by paying the trusted guard.

Despite their successful city wall and city gate protecting their enlarged village, our medieval folk still have some problems with theft from their open houses. They therefore apply the security principle of Defense in Depth and re-apply SINGLE ACCESS POINT at their individual houses. Each house gets a front door that can be locked, thus protecting its inhabitants, but still allowing them and their visitors in and out. The existing stone walls of their houses already provide good boundary protection, especially because, being medieval, their windows are made from iron bars instead of glass.

14.10. Consequences

The following benefits may be expected from applying this pattern:

- It provides a single place to go for entering the system, a clearly defined entrance for users of the system, and a single place to set up the system or application properly.

- It provides a single place to guard your system: you only need to trust your gate guards at the single access point within your system. However, applying Defense in Depth might be required to improve security further.
- The inner structure of system is simpler because repeated authorization checks are avoided. The system trusts the single access point.
- No redundant authorization checks are required: once the access point is passed, the system trusts the client.
- It applies to many levels of abstraction.

The following potential liabilities may arise from applying this pattern:

- Having just a single access point may make the system cumbersome to use, or even completely unusable. For a medieval city, if you arrive from the wrong direction, you have to walk right round the city just to reach its gate.
- You need to trust your gatekeeper and your city wall. However, it might be easier to check the single access point instead of multiple ones. Nevertheless, the boundary protection still can be a weak point of your system.
- The single access point might need to check the client on entrance more thoroughly than is required in the concrete situation, thereby annoying the client, or slowing down entrance unacceptably.
- In a complex system, several single access points might be required for subsystems.
- The single access point might become a single point of failure. If the single access point breaks down, the system might become unusable, or its security compromised.

14.11. Known Uses

Many operating systems, such as Mac OS, Microsoft Windows, and Unix, require a user to log into the system before it can be used. All provide either a dedicated login program or a prominent log-in window for the user to provide their identity and password. The boundary protection is built into the operating system by not allowing programs to be run by unauthorized or anonymous users.

Other patterns in this book, such as the firewall patterns and PROTECTION REVERSE PROXY, provide examples of effective single access points, in which the clients are not always users, but can be network traffic that needs entry to the protected system.

14.12. See Also

CHECK POINT and SECURITY SESSION both provide details of how to implement access control based upon SINGLE ACCESS POINT in a flexible and effective way.

14.13. References

[1] Alexander, C. (1977). *A pattern language: towns, buildings, construction*. Oxford university press.

14.14. Source

Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., & Sommerlad, P. (2006). Security Patterns: Integrating Security and Systems Engineering (1st ed.). Wiley.

15. Stateful Firewall

15.1. Intent

A stateful firewall filters incoming and outgoing network traffic in a computer system based on state information derived from past communications. State information generally describes whether the incoming packet is part of a new connection, or a continuing communication whose connection was approved previously. In other words, states describe a context for each packet.

15.2. Example

We have been able to contain many attacks with PACKET FILTER FIREWALL and PROXY-BASED FIREWALL. However, we are still plagued with distributed denial of service attacks that prevent customers from reaching our site. We also have performance problems for high-speed streams. In addition, a more sophisticated group of hackers is attacking us, sending us viruses whose bodies are assembled from parts included in message data and commands.

15.3. Context

Computer systems on a local network connected to the Internet and to other external networks. A higher level of network security is needed than static packet or proxy filtering. A PACKET FILTER FIREWALL only inspects the address of the packet, without the knowledge of previous communications of the same network. Similarly, a PROXY-BASED FIREWALL filters based on proxy restrictions for each packet. The knowledge of whether a connection is a new connection or an established connection is important for improved security: in particular, denial of service attacks could be identified more conveniently if we knew the relationship between packets [1].

15.4. Problem

How can we correlate incoming packets? This correlation may be useful to see if they include portions of commands or data needed for attacks, or to avoid redundant checks and improve performance.

The solution to this problem must resolve the following forces:

- Network administrators deploy and configure a variety of firewalls, so it is important to have a clear model of what packet correlations are required to be inspected and filtered, and what level of stateful inspection is desired. Otherwise, configuration errors and extra overhead may result.
- The configuration of the firewalls must reflect the organization’s security policies, otherwise it would be difficult to decide on what to filter and what stateful features to include.
- What is being inspected and filtered is constantly changing, so it should be easy to make changes to the configuration of the firewall.
- It may be necessary to log client requests for auditing and defense purposes.

15.5. Solution

Keep a list or table (a dynamic rule set) with the connections that have been opened and correlate the type of messages received or sent. This gives the option of not inspecting the packets of a well-established connection.

15.6. Structure

The figure below shows the Stateful Firewall class as including a StateTable class that describes the existing network connections. The new client (an external host) can only access our local network if a rule exists for authorizing traffic from its address. In addition, if it is a continuing communication from the same client, access is allowed based on whether a corresponding entry is in the StateTable. Each association link between the client and local network is therefore controlled by a Rule and/or an entry in the StateTable. The Stateful Firewall includes a set of access rules defined for the organization or local network according to its policies. If a particular request is not satisfied by any of the explicit rules, then the default rule is applied. For every new connection, an entry is made into to StateTable.

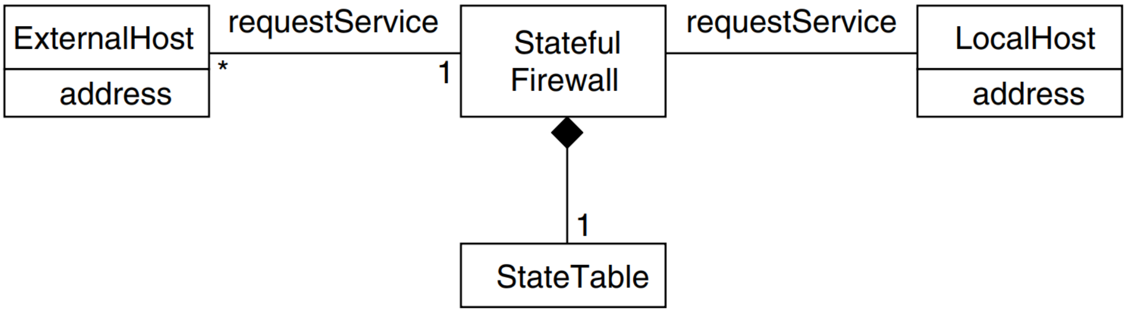

Figure 32: Class diagram for STATEFUL FIREWALL

15.7. Dynamics

In figure 35 the dynamic aspects of the STATEFUL FIREWALL (417) are described by a sequence diagram that corresponds to the basic use case of filtering a client's request using states.

Use Case:	Filtering a Client's Request
Summary:	A remote network requests access to the local network to either transfer or retrieve information. The access request is made through the firewall, which according to a state table, and if necessary, a set of rules, determines whether to accept or deny the request.
Actors:	External client.
Precondition:	The state table contains the list of previously established connections or connection attempts. If the state table does not allow a request, rules must be consulted as in PACKET FILTER FIREWALL.
Description:	1. An external network requests access to the local network. 2. A firewall filters the request according to a state table. If the connection exists in the state table, the request is accepted without further inspection. 3. If the connection does not exist in the state table, the request may be filtered based on a set of rules, assuming a packet firewall is part of the combination—see Variants below. If none

	of the rules are satisfied, then the default rule is used to filter the request, as in PACKET FILTER FIREWALL. 4. If the request is accepted, the firewall allows access to the local network.
Alternate Flows:	If the request if denied, the firewall rejects the access request by the external network to the local network.
Postcondition:	The firewall has filtered the access of a client to the local network.

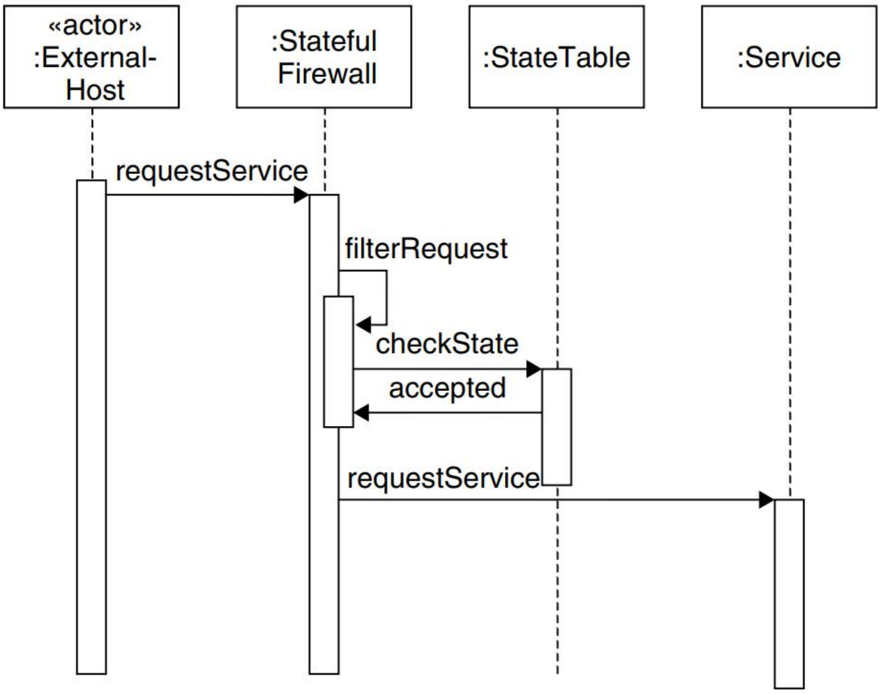

Figure 33: Sequence diagram for providing service to a client via a state table

15.8. Implementation

1. Make a list of the types of attacks we want to prevent.
2. Set the state tables to correlate packets according to these attacks.

15.9. Example Resolved

Typical denial of service attacks starts by sending a connection request not followed by an establishment of the connection after its acknowledgement. Our state table keeps a list of all open connections, and if the connections are not established within a given time period, we just cancel them. We also have a catalog of virus patterns, and we can make our firewall inspect sequences of messages to detect these attacks.

15.10. Consequences

The following benefits may be expected from applying this pattern:

- It is relatively easy to set up the state table once we know what attacks we are expecting.

- It has a low implementation cost, as it requires only a state table.
- If offers good performance. It only needs to look at packet headers for new connections. For existing connections, it looks only at the state table.
- It can enhance the security of the other types of firewalls by adding information from different levels about correlated packets.
- New attacks only require more ways to correlate packets.
- It allows connection-based logging of traffic. This may be useful for detecting patterns of attack that can be used by intrusion detection systems.

The following potential liabilities may arise from applying this pattern:

- The state table may fill and allow some attacks that take advantage of this fact [2].
- Attack patterns must be defined and coded so they can be recognized.

15.11. Variants

A STATEFUL PACKET FILTER FIREWALL (shown in figure 36) combines addressbased filtering with state information, that is, it filters based on the address of the packet and the information in the state table.

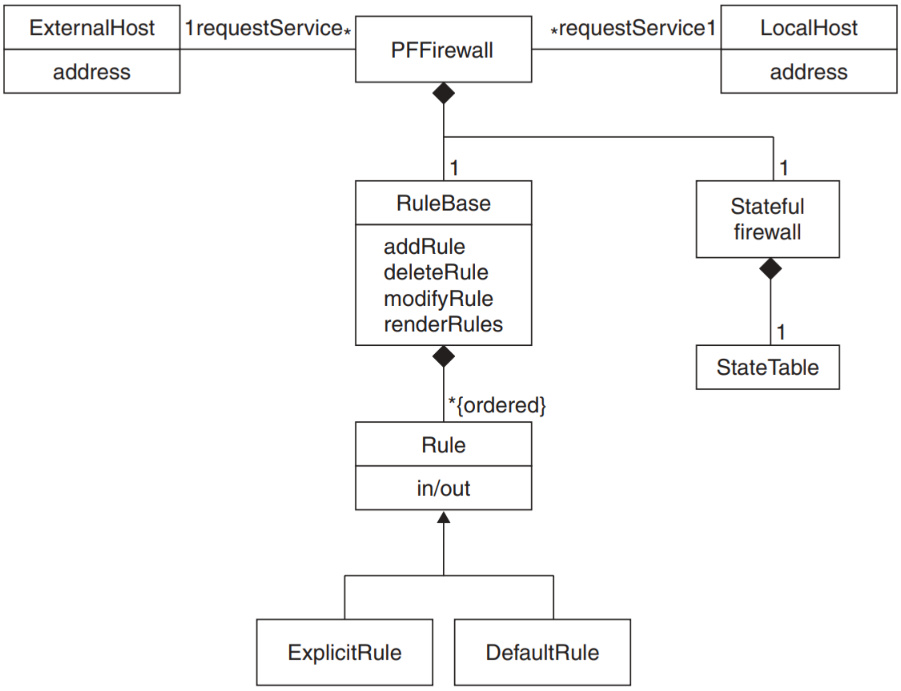

Figure 34: Stateful Firewall combined with a packet filter firewall

The STATEFUL PROXY-BASED FIREWALL (not illustrated) inspects, and filters incoming and outgoing network traffic based on the type of network application they are accessing, and the state of the communication between the networks.

15.12. Known Uses

This pattern can be found in commercial firewall products from organizations such as Software Technologies [3], Check Point Technologies, and CyberGuard [4]. Some specific firewall products that use stateful application proxies are Pipex Security Firewalls [5] and InterGate Firewall [6].

15.13. See Also

This firewall is usually combined with one or both of the previous types of firewalls, PACKET FILTER FIREWALL and PROXY-BASED FIREWALL.

15.14. References

[1] Noureldien, N. A., & Osman, I. M. (2000, September). A stateful inspection module architecture. In *2000 TENCON Proceedings. Intelligent Systems and Technologies for the New Millennium (Cat. No. 00CH37119)* (Vol. 2, pp. 259-265). IEEE.

[2] Frantzen, M., Kerschbaum, F., Schultz, E. E., & Fahmy, S. (2001). A framework for understanding vulnerabilities in firewalls using a dataflow model of firewall internals. *Computers & Security*, *20*(3), 263-270.

[3] Software Technologies. (2003). http://www.sofaware.com/

[4] Henry, P. (2001, April). An Examination of Firewall Architectures. CyberGuard Corporation White Paper. http://www.cyberguard.com

[5] Pipex. (n.d.). Pipex Firewall Solutions. http://www.security.pipex.net/about.shtml

[6] Vicomsoft. (n.d.). InterGate Firewalls. http://www.vicomsoft.com/

15.15. Source

Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., & Sommerlad, P. (2006). Security Patterns: Integrating Security and Systems Engineering (1st ed.). Wiley.

16. Threat Assessment

16.1. Intent

Threats are the likelihood of, or potential for, hazardous events occurring. They can affect any asset or object on which an enterprise places value. An enterprise threat assessment identifies the threats posed to the enterprise's assets and determines the likelihood or frequency of their occurrence.

16.2. Example

The museum has begun a risk assessment and identified the following assets to be in scope:

- Information asset types
  - Museum employee data
  - Museum financial/insurance data, partner financial data
  - Museum contractual data and business planning
  - Museum research and associated data
  - Museum advertisements and other public data
  - Museum database of collections information
- Physical Assets
  - Museum building
  - Museum staff
  - Museum collections and exhibits
  - Museum transport vehicles

The museum has also identified the major security needs for these assets using SECURITY NEEDS IDENTIFICATION FOR ENTERPRISE ASSETS and must now determine the threats to those assets.

16.3. Context

An enterprise has defined the assets to be included in a risk assessment and must now identify the events that could cause harm to those assets.

16.4. Problem

Enterprise assets face a barrage of attacks and hazardous events from all directions. Without effectively acknowledging the origins and frequency of these threats, an enterprise may never recognize the extent to which their assets are at risk.

How can an enterprise identify harmful events and determine the likelihood of their occurrence?

An enterprise must resolve the following forces:

- It must identify only those threats that have the potential for causing damage

- The type of business in which an enterprise is engaged will strongly affect the potential threat sources it will face
- The enterprise would like to develop a standardized way of identifying threats and assessing their likelihood, to be consistent with subsequent threat assessments
- The solution should address all assets included in the scope of a risk assessment, including informational and physical assets and, ideally, should be able to address vulnerabilities in non-IT systems

16.5. Solution

Systematically and explicitly identify and assess the threats against an enterprise and determine the types of protection they need. This activity is typically performed by an enterprise architect or strategic planner, and includes the following steps:

1. Identifying threats. Identify major threat sources that could potentially impact the assets defined by the scope of the risk assessment and trace their threat actions and consequences.
2. Building a threat table. Build a threat table by grouping threats first by asset type, then threat source.
3. Creating a likelihood scale. Create a scale for rating the frequency of attempted events, or likelihood for events occurring. This scale will represent the expected rate of occurrence of a given hazardous (natural or accidental) event, or an attack attempt.
4. Rating each threat. Rate each threat according to the likelihood scale and update the threat table to reflect this rating.

16.6. Structure

16.7. Dynamics

First, identify threats to the assets define by the scope of the risk assessment and build a threat table. A threat likelihood scale can also be developed in parallel. Finally, using the severity scale, rate each threat and update the threat table. The allowable sequence for performing a threat assessment is shown in the figure below.

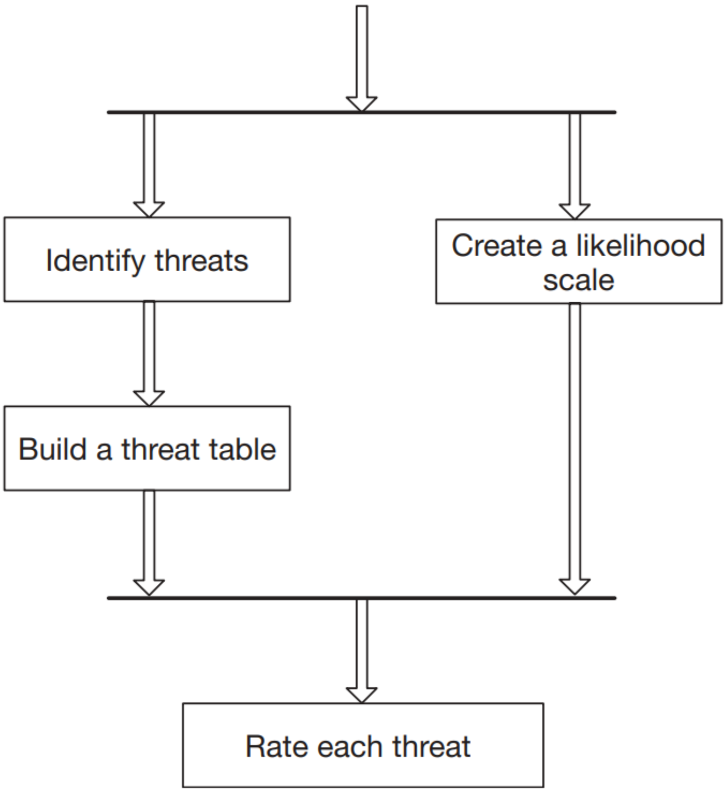

Figure 35: Threat assessment sequence constraints

16.8. Implementation

The implementation of the process for assessing threats is described below.

16.8.1. Identifying Threats

Identifying threats. A threat consists of three parts: the threat source, action, and consequence.

1. The threat source is that which initiates an attack or causes an event: a youth, an employee, or a fire, for example.
2. The threat action is the specific method by which an attack or event is carried out. An e-mail worm, a careless command entry, or water short-circuiting a motherboard are examples of threat actions.
3. The threat consequence is the security violation that results from the successful realization of the harmful event. Disruption of service, exposure of data, or destruction of hardware are some examples of the consequences that may occur.

Commonly, the threat source and threat action are grouped together and referred to as simply a 'threat.'

When determining threats, it is only necessary to consider those that are relevant to the assets, as defined by the scope of the risk assessment. Similarly, when an infrastructure is modified, such as when new applications are installed or new communication paths created, this threat landscape will change. Guidelines for defining the threat landscape include the following:

1. Specific environmental threats can quickly be removed given geographical or geological situations. For example, earthquakes cannot occur where there is no tectonic collision, tsunamis can only reach so far inland—although a flood can certainly occur in a building with a water supply.
2. Threats that have no measurable chance of occurring within the life expectancy of an asset can be eliminated. Forms of material decay or deterioration, or astronomical hazards—while all being possible—have such a low frequency of occurrence that they can realistically be ignored.
3. Threats can only target vulnerabilities. If a system isn't vulnerable to an exploit, then there is no threat, and consequently no risk. Consider a network environment consisting solely of Unix machines. Attacks launched against that network exploiting a buffer overflow on a Microsoft IIS Web server will obviously be ineffective, and thus the threat landscape should not include these threats.
4. Alterations to the management of data or other enterprise assets will alter the threat landscape. An attack that was previously not possible may now exist. For example, providing remote VPN access to employees now exposes an enterprise to residential-based threats.

16.8.1.1. Threat Sources

Sources of threats can be natural or human in origin. Natural threat sources are environmental forces frequently referred to as 'Acts of God.' Examples of natural threat sources include the following: tsunami, earthquakes, wind, snow, or rainstorms.

Human threat sources can be deliberate (attacks) or accidental (errors). Deliberate human threat sources are attackers and are differentiated by their motives, capabilities, and the assets they target. Those who seek to deliberately cause harm include the following:

- Hackers. They are generally motivated by mischief or grandstanding and may only seek publicity or notoriety among their peers. They employ simple tools, often precompiled using point-and-click interfaces created by others. While the tools may not be sophisticated, the results can range from the minor annoyance of a defaced Web page to major damage caused by the mass dissemination of malware such as worms or viruses.
- Professional criminals. They are motivated by financial reward and thus may steal credit card numbers, personal health information (PHI) or specialized documents such as corporate trade secrets, blueprints, or recipes, and offer them for sale to competitors, the corporation from which they were stolen, or the individuals themselves. Their techniques are more advanced than the youth hacker, both in organizational structure, technological skills, and attack execution.
- Terrorists. They care little for Web page defacement, but more for infrastructure disruption and destruction. Their methods can be crude but highly effective. The targets can be civilian, diplomatic, or military personnel in addition to public infrastructure systems such as power generation and distribution, water processing, telecommunications, financial/banking, emergency services, or transportation systems. Terrorist groups are often well funded and highly organized.

- Internal threat. Current or past employees who are angry or disgruntled are motivated by revenge or anger. They know the assets and the defenses of an organization, and can destroy data or interrupt services, posing a serious threat to any enterprise.

Accidental threat sources are actors who inadvertently cause damage or compromise the security posture of an asset. They may be employees or customers who are careless, inattentive, or poorly trained. This form of threat source might also be an application that is simply performing as programmed and mistakenly compromises a system.

16.8.1.2. Threat actions

Threat actions are the actual events that exploit the weakness of a system. They are the methods used by attackers to gain control of assets, they are the naturally occurring events that cause damage to systems, and they are the mistakes made by negligent users. They fall into the following categories:

- Natural. This includes extremes or fluctuations in temperature, causing metal fatigue or structural distress, electrical failures, surges, spikes or brownouts, fires, as well as natural disasters such as lightning strikes, earthquakes, uncontrolled flow of water into buildings or rooms through. rain, floods, inundation, storms, or hurricanes.
- Human deliberate. An example might be an attacker masquerading as a system administrator, using social engineering techniques to gather personal information about users, or an employee planting a logic bomb in a system, scheduled to erase critical system files.
- Human accidental. For example, a data center employee who inadvertently stumbles and jerks the power cord from a production server, or an employee transferring a file from a personal laptop to a corporate desktop, unaware that the file is infected with a virus.

16.8.1.3. Threat consequences

The realization of a threat can results in the violation of one or more of the security properties defined throughout this book: confidentiality, integrity, availability, or accountability. Regardless of the source or action of the threat, the consequences will be one of disclosure, deception, disruption, or usurpation.

16.8.2. Building a threat table.

Grouping by asset type becomes useful when the final risk to each asset is determined. Further grouping by threat source ensures that one does not overlook the fact that the same threat action can be initiated by different sources, each with a corresponding, and possibly different, frequency. For example, theft can occur from both a professional criminal and an employee. However, the frequency of theft from employees may be significantly higher than that of a criminal.

The threat consequence is included for each threat action and provides supporting clarification of the possible outcome of an incident.

It is possible that the threat table will be updated after the completion of the vulnerability assessment. Given the tight relationship between threats and vulnerabilities, identification of vulnerabilities can lead to the discovery of new threats that were previously not considered.

16.8.3. Creating a likelihood scale

While difficult to determine in precise quantitative terms, qualitative values can be used, and numeric estimates can be correlated. As an example, Table 15 shows a modified version of the probability levels given by [1].

Note that this table does not represent the only way to categorize event frequencies. Other threat assessments methodologies exist that define their own scale and they are equally valid. The important point is that an enterprise uses the same scale year after year, to provide consistent results between assessments.

Rating	**Likelihood**	**Description**
6	Extreme	The threat action is continually occurring
5	Very high	The threat action occurs very often
4	High	The threat action regularly happens
3	Medium	The threat action occurs infrequently
2	Low	This threat action rarely takes place
1	Negligible	The occurrence of this threat action is extremely unlikely within a human lifetime

Table 15: Event likelihood

16.8.4. Rating each threat

Each threat will have a certain likelihood or frequency of occurrence, and as expected, some will transpire more often than others, based on specific factors. Note that this is not the frequency of successful violations in which damage has occurred, but an event or attack that could cause damage.

To estimate or predict the frequency of a threat, it is necessary to consider many issues. Factors that affect the likelihood of a natural threat include the following:

- Proximity to dangerous chemical or petroleum factories. A few additional miles from an industrial incident may make the difference between a precautionary evacuation of a facility and human fatalities.
- The possibility of extreme weather patterns and fluctuations such as heat, wind, rain. While internal temperatures can be controlled to a certain degree, external temperatures can overload the control systems, affecting both human and mechanical systems. Specific geographical locations will naturally be more prone to such fluctuations and extremes.
- The state of the operating facilities with regard to structural integrity, fire suppression, and other emergency response systems. Older, less sturdy buildings may require constant refurbishment, resulting in disruption of service.

Factors that affect the likelihood of a deliberate human threat include:

- The time since a vulnerability has been publicly known. The longer since the vulnerability has been discovered, the greater the number of attackers that will be aware of it, and the more opportunity there will be to catalog, research and develop tools to exploit it.

- Whether or not a working exploit is available for the vulnerability. Graphical user and command-line interface exploits certainly have a much greater chance of being used than ones that require custom development such as coding. Having precompiled or point-and-click code reduces the knowledge level required to launch an attack: suddenly, one does not need detailed knowledge of the vulnerability in order to exploit it.
- The frequency of attack attempts. The more frequent the number of attack attempts, the greater the chance of a successful attack.
- The potential reward offered to an attacker. Hacker challenges and monetary reward increase the chances of an attack.
- The asset value. High-wealth businesses and assets attract more attention than those of lesser value, and therefore offer more incentive for compromise. Attackers will therefore not generally target systems that contain no value or provide no reward. There are two exceptions to this, however: either an attacker targets the system out of curiosity or simply to prove that it can be done, or an attacker breaches a useless system only to provide a launching point to another system of value (for example, compromising a home computer in order to penetrate a corporate network).
- The perceived difficulty of realizing a successful attack. If the asset is known to be heavily protected and the chances of reward low, the fewer will be the number of attempts.
- Public visibility and sentiment towards the business. Organizations that are viewed as having an unpopular affiliation, or that act inappropriately, may incur more attacks as a result.
- Employee morale. Low employee morale frustrates employees and can cause malicious or vengeful retaliation. It can also simply cause indifference to quality and service. Either way, low morale increases the potential for accidental or deliberate threat actions.
- Past prosecutions. If an organization is known for seeking retribution and prosecution of crimes, attackers will seek easier or less risky targets.

Factors that affect the likelihood of an accidental human threat include:

- The availability of skilled employees. If unqualified personnel are required to manage sensitive or complex systems, the opportunity for errors due to ignorance or mistakes increases greatly.
- Security measures. Administrative controls, such as user awareness and emergency training, educates users on policies and procedures, making them less susceptible to social engineering attacks and more aware of information security requirements.
- The frequency of changes to systems, including patches, upgrades, and other modifications. The more frequently changes are made, the more potential there will be for mistakes or corruption due to new configurations.

Arguably the most reliable method for determining the frequency of future events is historical data. Naturally occurring events are often recorded by educational and governmental organizations for study. Commercial and governmental references exist that record information security attacks. Relevant data can also be collected from the enterprise's own systems. Some examples of useful sources include the following:

- Historical almanacs (in the cases of natural disasters).
- News archives including federal services. For example http://www.fema.gov/
- Information security newsletters and Web sites. For example, http://www.securityfocus.com, CERT, Symantec, FedCIRC and SANS.
- Current and archived intrusion detection, incident response and application system log files.
- Previous threat assessment documents, if available, may also contain particularly relevant information.

16.9. Example Resolved

From SECURITY NEEDS IDENTIFICATION FOR ENTERPRISE ASSETS, the museum has identified its informational and physical assets:

- Information Asset Types
  - Museum employee data
  - Museum financial/insurance data, partner financial data
  - Museum contractual data and business planning
  - Museum research and associated data
  - Museum advertisements and other public data
  - Museum database of collections information
- Physical Assets
  - Museum building
  - Museum staff
  - Museum collections and exhibits
  - Museum transport vehicles

After use of THREAT ASSESSMENT, the museum has identified a brief list of threats to information and physical assets, as shown in the threat Tables 16 and 17, respectively

Threat Action (Frequency)	**Threat Consequence**
Natural	
Electrical spike in computer room (3)	Incapacitation, corruption of informational assets
Loss of electronic documents (3)	Incapacitation of informational assets
Professional criminals	
Theft of information assets (3)	Misappropriation, incapacitation, misuse, exposure, corruption of informational assets
Employees	
Unauthorized access to informational assets (5)	Exposure, falsification, incapacitation, misappropriation of informational assets
Data entry errors (5)	Corruption of information assets
Leaking confidential information (3)	Exposure of information assets

Table 16: Threats to information assets

Threat Action (Frequency)	**Threat Consequence**
Natural	
Museum fire (3)	Incapacitation of physical assets
Fatigue of support fixtures, building structural failure (3)	Incapacitation of physical assets

Failure of monitoring and alarming systems (4)	Intrusion, misappropriation of physical assets
Professional criminals	
Theft of museum collections and exhibits (2)	Misappropriation of museum collections and exhibits
Physical attack against employees (3)	Incapacitation of employees
Employees	
Accidental damage to museum collections and exhibits (4)	Incapacitation of museum collections and exhibits
Accidental damage to vehicles (4)	Incapacitation of museum collections and exhibits
Theft of museum collections and exhibits (2)	Misappropriation of museum collections and exhibits
Misconfiguration of monitoring and alarm systems (4)	Incapacitation, obstruction of monitoring and alarm systems
Museum patrons	
Accidental damage to museum collections and exhibits (3)	Incapacitation of museum collections and exhibits

Table 17: Threats to physical assets

16.10. Consequences

This pattern has the following benefits:

- The solution provides the enterprise with an understanding of the factors that increase both the existence and the frequency of harmful events.
- It identifies the consequences incurred should a given threat be realized.
- The threat assessment is a major component of the risk assessment pattern set that will prioritize and ultimately result in a more secure organization.

It also has the following liabilities:

- Accurate historical data may not be available, preventing the enterprise from acquiring useful threat frequency data.
- The effort required to conceive of all possible threats can be too time consuming for an enterprise. Constraints may therefore have to be placed on the completeness of the threat landscape.

16.11. Known Uses

Threat assessment is, for example, defined in the ISO Technical Report 13335-3 [2]. This definition of the process focuses on three tasks: identification of threat sources, the threat target, and the threat likelihood. It identifies that determining the likelihood should consider the threat frequency, the threat motive, and geographical factors such as proximity to industrial factories. This technical report differentiates the threat likelihood simply as high, medium, and low. The actual determination and definition are left to the implementer of the threat-assessment process.

NIST also describes a complete risk management process whose first step is a risk assessment [3]. Steps 3.2 and 3.5 in this process are dedicated to the identification of threats and determination of their likelihood. This publication also uses a likelihood scale of high, medium, and low. In making the determination of the likelihood of a threat, this scale also incorporates the existing controls and their capability to neutralize the threat. NIST also separates the identification of threats and the likelihood of their realization into two separate processes.

In her publication Security Engineering and Information Assurance, Debra Herrmann describes the need for a complete information security process to identify threats, their type, source, and likelihood [1].

Microsoft describes a threat and countermeasures pattern that offers alternative methods for identifying and assessing threats through 'Threat Modeling' [4]. The authors use a method called STRIDE that categorizes threats based on the 'goals and purposes of the attacks.' The categories that make up the acronym are: spoofing, tampering, repudiation, information disclosure, denial of service and elevation of privileges.

16.12. See Also

16.13. References

[1] Herrmann, D. S. (2002). Security Engineering and Information Assurance. Auerbach Publications

[2] "Technical Report ISO13335-3 Part 3: Techniques for the Management of IT Security" International Organization for Standardization, American National Standards Institute, 2002

[3] Stoneburner, G., Goguen, A., & Feringa, A. (2002). Risk management guide for information technology systems. *Nist special publication*, *800*(30), 800-30.

[4] Meier, J. D. (2003). *Improving web application security: threats and countermeasures*. Microsoft press.

16.14. Source

Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., & Sommerlad, P. (2006). Security Patterns: Integrating Security and Systems Engineering (1st ed.). Wiley.

17. Vulnerability Assessment

17.1. Intent

A vulnerability is a weakness that could be exploited by a threat, causing the violation of an asset's security property. Conducting an enterprise vulnerability assessment helps to identify the weaknesses of the enterprise's assets and the systems that enable access to them and evaluates the severity if a vulnerability were to be exploited.

17.2. Example

The museum has begun a risk assessment and identified the following assets to be in scope:

- Information asset types
    - Museum employee data
    - Museum financial/insurance data, partner financial data
    - Museum contractual data and business planning
    - Museum research and associated data
    - Museum advertisements and other public data
    - Museum database of collections information
- Physical assets
    - Museum building
    - Museum staff
    - Museum collections and exhibits
    - Museum transport vehicles

The museum has also identified the potential threats to those assets and must now determine vulnerabilities that can compromise those needs.

17.3. Context

An enterprise has defined the assets to be included in a risk assessment, and has identified potential threats, for example through applying THREAT ASSESSMENT. It must now identify the vulnerabilities that can be exploited by those threats.

17.4. Problem

Enterprise assets and the controls protecting them may be fully secure, or may have numerous weaknesses, some of which may never be exploited, and some of which may be exploited every day. Without proper cataloguing of these vulnerabilities, an enterprise might never recognize the extent of the weaknesses of their assets.

How can an enterprise identify vulnerabilities to its assets and determine the severity of those vulnerabilities?

An enterprise must resolve the following forces:

- It might have experience with a single tool or method for discovering weaknesses but may not be aware of other techniques that can reveal other, potentially critical, vulnerabilities.
- It need only identify vulnerabilities for which threats exist, and therefore the enterprise must be able to determine if a given vulnerability has an associated threat.
- It would like to develop a standardized way of identifying vulnerabilities and assessing their severity, in order to be consistent with subsequent vulnerability assessments.
- The solution should address all assets included in the scope of a risk assessment, including informational and physical assets, and, ideally, should be able to address vulnerabilities in non-IT systems.

17.5. Solution

Systematically identify and rate probable vulnerabilities of the enterprise assets. This process involves the following five steps:

1. Collect threat information. Collect information on threats. For example, if THREAT ASSESSMENT has been used, appropriate threat information is available from the resulting threat table.
2. Identify vulnerabilities. Using the threat table, identify the vulnerabilities of the assets and the systems protecting them defined in the scope of the risk assessment.
3. Build a threat-vulnerability table. Extend the threat table by associating each vulnerability with a threat action.
4. Create a severity scale. Create a scale for rating the severity of vulnerabilities. This scale will represent the degree to which an asset is susceptible to a vulnerability, and the potential impact should the vulnerability be exploited.
5. Rate each vulnerability. Rate each vulnerability according to the severity scale and update the threat vulnerability table to reflect this rating.

17.6. Structure

17.7. Dynamics

The allowable sequence for performing the vulnerability assessment process is shown in the figure.

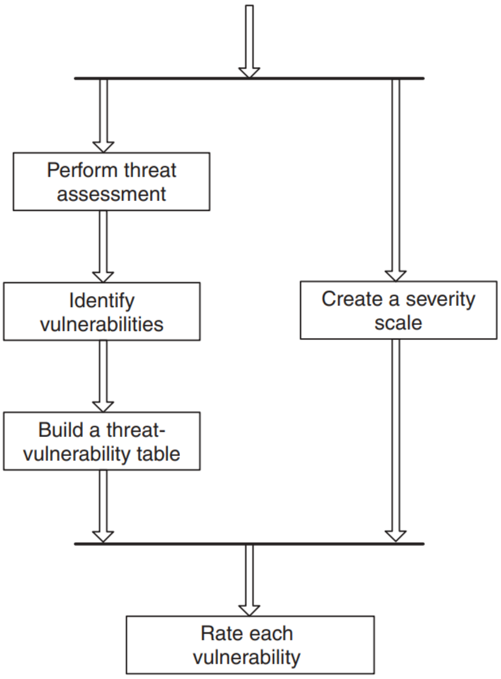

Figure 36: Vulnerability assessment sequence constraints

First collect appropriate threat information. Then, using the methods outlined, identify all vulnerabilities, and associate them with threats in the threat table, creating the threat-vulnerability table. A vulnerability severity scale can be developed at any time. Finally, using this scale, rate each vulnerability.

17.8. Implementation

The implementation of the process for assessing vulnerabilities is described below.

17.8.1. Collect threat information.

Threat information should include a list of events that could cause harm to assets and provide context for the vulnerabilities.

17.8.2. Identify vulnerabilities.

Use any of the following methods to identify vulnerabilities exploitable by the threats in the threat table.

17.8.2.1. System characteristics.

[1] describes four main causes of system vulnerabilities, and while it focuses on software applications, the causes can be generalized to help identify weaknesses in non-IT systems.

- Those that can be caused by dependency failure. Rarely, if ever, does an application does not interact with other applications or systems to perform its function. These interactions may be with database tables, shared system libraries, network services or devices, or operating system resources. The behavior of the application in the event of a failure or unavailability of these dependencies is a prime target for attack. For example, how would an application respond when a security library could not be loaded? Does it bypass all security calls, log an error, and continue, or halt all operation and alert operators? How does an application respond to low system resources such as low disk or memory conditions?

- Those that can be caused by unanticipated data input. The absence of data input validation is a very common mistake. It can also be the most damaging because the results can range from denial of service to complete subversion of the system through full administrator access. Buffer overflows and SQL injection are two of the most prevalent examples of this class of attack.
- Those that can be caused by design vulnerabilities. As the size and complexity of an application grows, it becomes more difficult to identify and validate the flow and integrity of data. The potential for exploiting design flaws therefore increases. Such design flaws may include use of cleartext protocols where encrypted ones are necessary, acquiring escalated privileges by circumventing access or authorization controls, assumptions made by designers or developers regarding the use of or operation of the application, and jumping outside the bounds (and constraints) of the system to perform unauthorized tasks or operations.
- Those that can be caused by implementation vulnerabilities. A most secure design can still lead to a substantial vulnerability if the implementation is faulty. This provides another reason why scale and complexity are impediments to secure systems. The larger and more intricate a design, the more opportunities there will be for implementation errors.

An example is software that deals with sensitive information. It must ensure that while processing the information, it is not temporarily copied to either disk or memory unprotected. Replay attacks, ‘bait and switch,’ and ‘man in the middle’ are all examples of implementation vulnerabilities: a replay attack is a form of network attack in which a valid data transmission is maliciously or fraudulently repeated or delayed, while a bait and switch is a form of fraud in which the fraudster lures in customers by advertising goods at an unprofitably low price, then reveals to potential customers that the advertised goods are not available, but that a substitute is. A ‘man in the middle attack’ (MITM) is an attack in which an attacker is able to read, insert and modify messages between two parties without either party knowing that the link between them has been compromised.

17.8.2.2. Development life cycle.

[2] recognizes that vulnerability identification is dependent on the nature of an IT system and its phase in a development life cycle. Differentiation is made between systems being designed, systems being implemented, and systems in production.

For systems or applications that have not yet been designed, ‘the search for vulnerabilities should focus on the organization’s security policies, planned security procedures, system requirement definitions, and the vendors’ or developers’ security product analyses.’ At this stage, these design documents and product specifications are all that are available for security review.

For systems that are in the process of being implemented, vulnerability identification ‘should be expanded to include more specific information, such as the planned security features described in the security design documentation and the results of system certification test and evaluation.' This is where security auditing tools can first be used to test applications before they are released into production. These tools generally perform signature-based checks to test for known weaknesses.

Finally, for systems that are already in production, vulnerability identification 'should include an analysis of the IT system security features and the security controls, technical and procedural, used to protect the system.' For these systems, security auditing tools as well as penetration tests will identify weaknesses most effectively, although not necessarily safely. They function by directly testing for the presence of known exposures, as opposed to theorizing their existence based on documentation, policy or a countermeasure that is supposed to block an attack. Penetration tests can be a complex effort requiring the cooperation of many departments, including information security, operations, and application development. As mentioned, these tests can be most effective because they test the weakness of the IT system as well as the countermeasures that are (or should be) in place.

17.8.2.3. Other

Specialized techniques for identifying vulnerabilities in IT systems include the following:

- Vulnerability scanning. Vulnerability scanning is the act of running automated tools or procedural tests on networks and applications in order to detect or confirm the presence of vulnerabilities [3].
- Penetration tests. Penetration testing is a procedure that attempts to circumvent, disable, or otherwise defeat the security controls of a system using any available tool or technique.
- Vulnerability catalogs. These catalogs provide a list of vulnerabilities for specific applications and configurations. Examples include:
    - CERT Knowledgebase: http://www.cert.org/kb
    - CVE database: http://www.cve.mitre.org
    - NIST ICAT: http://icat.nist.gov
- Open-Source Vulnerability Database: http://www.osvdb.org/
- Vendor advisories and patch lists. Commercial vendors and Open-Source developers will often provide vulnerability advisories for their products.
- Information security forums and mailing lists. These lists provide a popular discussion and distribution forum for security vulnerabilities. For example:
    - Bugtraq: http://www.securityfocus.com/
    - SANS: http://www.sans.org
    - RISKS: http://catless.ncl.ac.uk/Risks/
    - Dartmouth College Institute for Security Technology Studies: http://news.ists.dartmouth.edu/

17.8.3. Build a threat-vulnerability table

Extend the threat table by pairing vulnerabilities with threats, creating a threat vulnerability table. Recall that threats are grouped by threat source (natural, hacker, criminal, and so on), accommodating situations in which the same threat action (for example theft) originates from multiple sources (for example employees and criminals).

This table format enforces the restriction that it is only necessary to consider vulnerabilities for which threats exist. If a vulnerability is found to have no associated threat, either remove the vulnerability from consideration, or update the threat table to include the threat.

To determine whether a vulnerability has an associated threat, ask yourself if there is any way that the security properties (confidentiality, integrity, availability, and accountability) of an asset could be compromised as a result of the weakness. Importantly, this does not answer the question of who caused the compromise, or how it occurred, but simply whether it could occur.

17.8.4. Create a severity scale

Create a severity rating scale by first defining a rank, then assign a meaning and description to each rank. An example is shown in Table 18. The rating represents the degree to which the asset is susceptible to the vulnerability, and the potential impact should the vulnerability be exploited. Note that the range and description are at the discretion of the enterprise—it can change the range, severity term, and description as appropriate. The important consideration is that the table remain constant throughout the risk assessment and across the enterprise.

Rating	Severity	Description
6	Extreme	1. The vulnerability is trivially exploitable and commonly found, or 2. Major loss of life and destruction of systems would occur
5	Very high	1. The vulnerability is easily exploitable and found in most systems, or 2. Some loss of life and major destruction of systems would occur
4	High	1. Exploiting the vulnerability would be a challenge but it exposes many systems, or 2. Human physical injury, some destruction of systems would occur
3	Medium	1. The vulnerability is difficult to exploit, and exposes some systems, or 2. Significant disruption of service and compromise of confidentiality, availability, or integrity of assets would occur
2	Low	1. The vulnerability would be very difficult to exploit, with no real gain, or 2. Slight disruption of service or mild compromise of security properties would occur
1	Negligible	1. This is a theoretical vulnerability only exploitable with massive infrastructure or computing power, or 2. Minor distraction to business processes and no compromise of security properties would occur

Table 18: Vulnerability severity scale

17.8.5. Rate each vulnerability

Rate the severity of each vulnerability according to the considerations listed below.

1. General factors:
   a. The number of threats that can be realized as a result of a given vulnerability being exploited. Also, the number of systems affected by the vulnerability. If a

single vulnerability provides the opportunity for many threats to be realized (and perhaps many subsequent vulnerabilities to be exploited), then the severity should be reflective of this.

   b. The prevalence of the systems affected by the vulnerability. Some vulnerabilities may impact uncommon or infrequently used applications or enterprise resources, while others may impact a ubiquitous Internet service or physical infrastructure.
   c. Whether or not the weakness exists in default configurations or installations.
   d. Whether there are any preconditions that need to exist before the vulnerability can be exploited, such as the compromise of other systems or security controls.
   e. Whether the affected asset is responsible for monitoring or protecting other assets.
   f. Whether the attacker needs to lure victims to a hostile server in order to exploit a vulnerability.

2. Existing security controls. Security measures that are already in place significantly affect both an enterprise's susceptibility (resistance) to a vulnerability, and the severity of damage it causes.
   a. Preventative controls. These controls are employed to inhibit attacks and prevent harmful events from reaching their destination. Firewalls, anti-virus scanners, code reviews, encryption techniques, fences, door locks, and so on are all forms of preventative controls.
   b. Detective controls. Detective controls are employed to discover attacks. By the time these controls are used, an attack or event has already occurred. These controls must be capable of reacting very quickly to prevent loss or damage. Technical examples are intrusion detection systems (IDS)—either host-based or network-based, audit trails, and so on. Physical detective controls would include tripwires, and IR or motion sensors. An administrative control would be a policy dictating mandatory job rotation and job vacation.
   c. Corrective controls. A potentially harmful event has occurred, and the detective controls have recognized it. Now the corrective controls are employed to mitigate the impact or loss due to the event. Intrusion prevention systems (IPS), and auto-restore features (as found in Windows XP, for example) are some examples of technical corrective controls. Physical controls would include doors or gates that lock automatically, trapping any intruders, fire suppressant systems, and security alarms.
   d. Recovery controls. These controls are designed to recover from the loss or damage incurred by the event. Backups, disaster recovery (DR) and business continuity plans (BCP) are examples of recovery controls.

Note that deterrent controls are not included, as they assist in reducing the threat or probability of an incident.

17.9. Example Resolved

After applying a sequence of THREAT ASSESSMENT (providing the threat action frequencies) and VULNERABILITY ASSESSMENT patterns, the museum has identified the vulnerabilities to information and physical assets shown in Tables 19 and 20 respectively. The threat action frequency values of both tables are taken from THREAT ASSESSMENT.

Threat Action (Frequency)	**Vulnerability (Severity)**
Natural	
Electrical spike in computer room (3)	Lack of surge protection, uninterruptible power system (UPS) (4)
Loss of electronic documents (3)	Incomplete or corrupt data backups (4)
Professional criminals	
Theft of information assets (3)	Susceptibility of employees to bribery (3) Lack of proper physical controls for document storage (locks, safe) (4)
Employees	
Unauthorized access to informational assets (5)	Weak information security controls enabling unauthorized access (3)
Data entry errors (5)	Lack of data validation during form input (2)
Leaking confidential information (3)	Exposure of information assets (3)

Table 19: Threat-Vulnerabilities table for information assets

Threat Action (Frequency)	**Vulnerability (Severity)**
Natural	
Museum fire (3)	Failure of fire alarm system (6) Failure of fire suppression system (5)
Fatigue of support fixtures, building structural failure (3)	Lack of regularly scheduled inspections (4)
Failure of monitoring and alarm systems (4)	Lack of regularly scheduled inspections (4)
Professional criminals	
Theft of museum collections and exhibits (2)	Lack of regular alarm testing procedures (3) Lack of adequate storage and protection of physical assets (3)
Physical attack against employees (3)	Lack of security training for employees (4)
Employees	
Accidental damage to museum collections and exhibits (4)	Carelessness of employees when handling/cleaning exhibits (2)
Accidental damage to vehicles (4)	Carelessness of employees while driving vehicles (2) Lack of regularly scheduled maintenance checks (4) Lack of adequate employee background checks (4)
Theft of museum collections and exhibits (2)	Lack of regular alarm testing procedures (3) Lack of adequate storage and protection of physical assets (3) Susceptibility of employees to bribery (4)
Misconfiguration of monitoring and alarm systems (4)	Lack of regular alarm testing procedures (3)
Museum patrons	
Accidental damage to museum collections and exhibits (3)	Carelessness of museum patrons when viewing exhibits (2)

Table 20: Threat-vulnerability table for physical assets

17.10. Consequences

This pattern has the following benefits:

- An enterprise obtains a list of all vulnerabilities that could impact their systems, some which may have been previously unknown.
- The enterprise is able to rank the vulnerabilities according to severity and potential impact.
- An enterprise is able to recognize which vulnerabilities can be discounted where there are no accompanying threats.

It also has the following liabilities:

- A thorough vulnerability scan involves the coordination of many departments and may be difficult to initiate if these departments are not in cooperation.
- This pattern cannot be used in isolation to patch or eliminate vulnerabilities. The results of VULNERABILITY ASSESSMENT should be returned to the RISK DETERMINATION pattern, where the final risk can be determined, and an appropriate control implemented.

17.11. Variants

The SANS Institute and the CERT Coordination Center at Carnegie Mellon are two renowned information security centers. They provide vulnerability lists and databases of common vulnerabilities. [4] uses a purely quantitative scale of 0 to 180 to rank the severity of a vulnerability., whereas [5] uses the following qualitative scheme:

- Critical vulnerabilities are those where essentially all planets align in favor of the attacker. These vulnerabilities typically affect default installations of very widely deployed software, result in root compromise of servers or infrastructure devices, and the information required for exploitation (such as example exploit code) is widely available to attackers.
- High vulnerabilities are usually issues that have the potential to become critical but have one or a few mitigating factors that make exploitation less attractive to attackers.
- Moderate vulnerabilities are those where the scales are slightly tipped in favor of the potential victim. Exploits that require an attacker to reside on the same local network as their victim, or only affect non-standard configurations or obscure applications, are likely to be rated moderate.
- Low vulnerabilities usually do not affect most administrators, and exploitation is largely unattractive to attackers. Often these issues require the attacker to have some level of access to a target already, require elaborate specialized attack scenarios, and only result in limited damage to a target.

[2] uses the following definitions for vulnerability severity:

- High. Exercise of the vulnerability (1) may result in the highly costly loss of major tangible assets or resources, (2) may significantly violate, harm, or impede an organization’s mission, reputation, or interest, or (3) may result in human death or serious injury.

- Medium. Exercise of the vulnerability, (1) may result in the costly loss of tangible assets or resources, (2) may violate, harm, or impede an organization's mission, reputation, or interest, or (3) may result in human injury.
- Low. Exercise of the vulnerability (1) may result in the loss of some tangible assets or resources, (2) may noticeably affect an organization's mission, reputation, or interests.

The Common Vulnerability Scoring System [6] is an open framework that can be used by any security or application vendor to determine the overall severity posed by a vulnerability. Three categories of metrics are scored and combine to produce a final score.

- The base metric represents the properties of a vulnerability that do not change over time, such as access complexity, access vector, degree to which the vulnerability compromises the confidentiality, integrity and availability of the system, and requirement for authentication to the system.
- The temporal metric measures the properties that do change over time, such as the existence of an official patch or functional exploit code, and the level of effort to remedy the vulnerability.
- The environmental metric measures the properties of a vulnerability that are representative of users' IT environment, such as prevalence of the affected system and overall potential loss.

17.12. Known Uses

A vulnerability assessment is a key component of all widely accepted risk assessments, including those from [2,7,8], and others. While they differ slightly in their approach, the purposes and overall goals are consistent.

17.13. See Also

17.14. References

[1] Whittaker, J. A. (2002). How to break software.

[2] Stoneburner, G., Goguen, A., & Feringa, A. (2002). Risk management guide for information technology systems. *Nist special publication*, *800*(30), 800-30.

[3] Wack, J., Tracy, M., & Souppaya, M. (2003). Guideline on network security testing. *Nist special publication*, *800*(42), 13-14.

[4] CERT Coordination Center. (2004). Carnegie Mellon University. https://www.kb.cert.org/vuls/help/fieldhelp/

[5] SANS Institute. (2004). The CVA Process. http://www.sans.org/newsletters/cva/#process

[6] Common Vulnerability Scoring System SIG. (2005). FIRST — Forum of Incident Response and Security Teams. https://www.first.org/cvss/

[7] "Technical Report ISO13335-3 Part 3: Techniques for the Management of IT Security" International Organization for Standardization, American National Standards Institute, 2002

[8] Peltier, T. R. (2001). Information Security Risk Analysis, Auerbach Publications, 2001

17.15. Source

Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., & Sommerlad, P. (2006). Security Patterns: Integrating Security and Systems Engineering (1st ed.). Wiley.