FIREWALL PATTERNS

Markus Schumacher

Darmstadt University of Technology
Department of Computer Science
Wilhelminenstr. 7
64283 Darmstadt, Germany

markus@securitypatterns.org

Abstract

In this paper we provide three firewall patterns. The firewall pattern describes how access to internal networks can be restricted in general. It shows the basic problems and indicates a general solution. The other two patterns are more specific variants which are usually used in order to implement access control at the network border. This paper is also an experiment of understanding what it means to *specialize* security patterns and how we can cope with this presenting them in a useful and joyful way to the reader. The patterns themselves are part of an initiative to integrate known security-related patterns into an overall security pattern system.

Introduction

In this paper we present selected network security patterns. Some of them have initially been presented in a security book and at PLoP 2001 [8, 7]. Before we step into the patterns, we first give an overview of the patterns contained in this paper. We introduce an overall scenario which illustrates the common context for all patterns. Then we discuss the pattern template we used. The remainder of the paper contains three firewall security patterns.

Overview

In this paper we present three patterns which describe how networks can be protected with basic firewall techniques (see for example [2]). In particular, we present a generic FIREWALL pattern as well as the two more specific patterns PACKET FILTER and PROXY. More formally, we understand this relation between patterns as follows[1]:

> Let P_1 and P_2 be patterns. Then we say that P_1 specializes P_2 if the context and/or the problem (as well as the corresponding forces) of P_1 is more specific than the contextand/or problem of P_2.

This notion would mean, that we take a pattern and either add more specific facts or make certain statements more specific. By nature, this will also make the solution more specific. Presenting patterns in this way requires, however, a trade-off between readability and correctness (e.g. should we really repeat all more general elements and add the specific ones?). As such, this paper is also the author's experiment of understanding what it means to *specialize* patterns and how we can cope with this presenting useful *and* joyful to the reader.

Basically a firewall controls the traffic between two or more networks in different administrative domains. For the sake of simplicity, we just refer to an internal (the protected network) and an external network (where the bad guys are).

Figure 1 illustrates a scenario which represents the context of the patterns in this paper. Thereis an internal network with a set of hosts connected, e.g. the system I_1. The internal network is connected with a routing device to one or more external networks. Basically, the external network cannot be trusted. Thus it is important to distinguish regular packets from malicious connections originated from an attacker. Otherwise, the security of the internal systems cannot be guaranteed.

[1]A even more formal definition is given by the author [6].

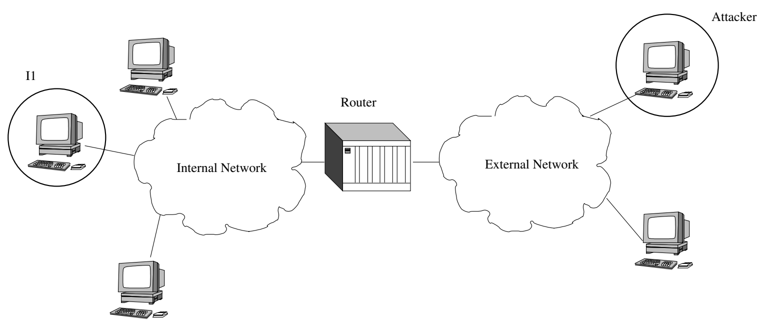

Figure 1: Transition between internal and external network.

Pattern Template

In contrast to the original work [8, 7], we present the patterns here in a more familiar form which is similar to Alexander's original template: Each pattern is divided into three parts which are separated by three diamond symbols (◊◊◊). Hereby, name and context create the *introductory part*, problem statement, forces and solution build the *central part* and the references to other patterns present the *closing part*.

Each pattern begins with a name (i.e.the heading of the section). The next few sentences are in a "you" form. They describe a context in which you may or may not find yourself. If you don't find yourself in such a context, the pattern probably isn't relevant for you. Then we provide a brief description of the problem in bold face (highlighting core elements of the pattern instead of giving them separate headings increases the readability). Afterwards, a number of forces (threats, attacks, etc.) that must be considered are discussed (now with a regular font face). The next section begins with a bold face "Therefore" and contains the core of the solution. This is followed by additional information about the pattern, how the forces are resolved (i.e. the consequences) and how to use or implement the pattern. Finally, we include references to related patterns.

Pattern Roadmap

In this paper we present only a subset of security patterns at the network layer. In particular, our focus is on firewall solutions. More patterns, e.g. dealing with INTRUSION DETECTION, VIRTUAL PRIVATE NETWORK, and further types of firewalls will complement the pattern landscape as illustrated in Figure 2.

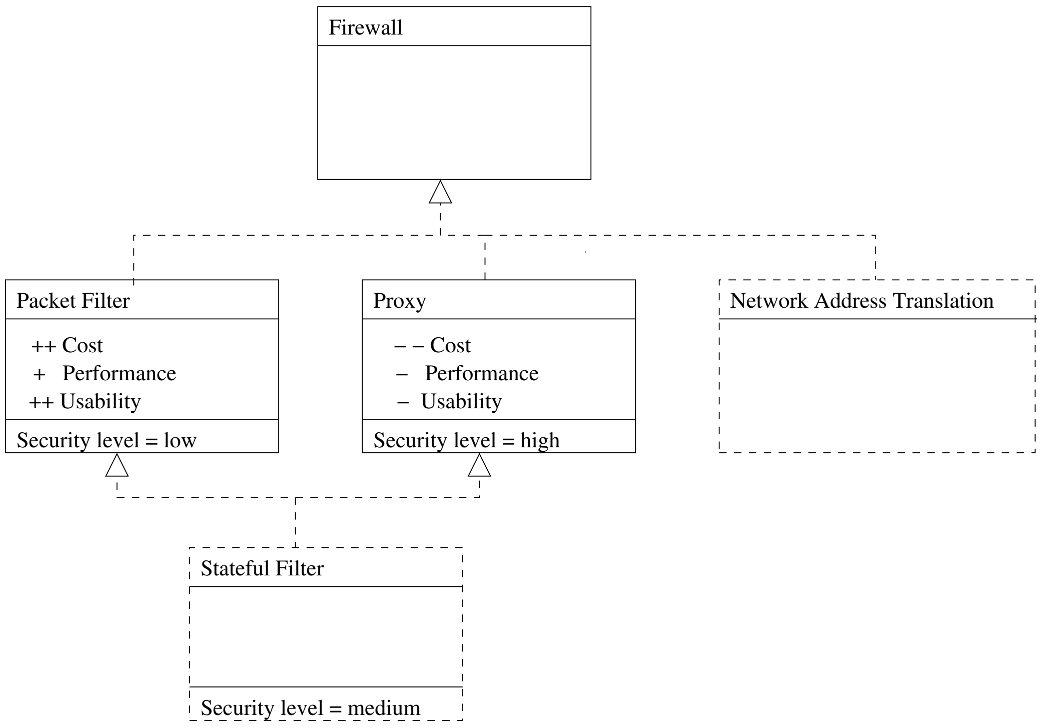

Figure 2: Firewall Pattern Landscape.

The diagram is a slight variant of a UML class diagram[2]. Each box indicates a pattern where a solid-line border indicates a pattern discussed in this document and a dashed-line box indicates a pattern that is identified but left for future work.

For the patterns discussed in the following we provided the dominant forces of the pattern inside the box. A '+' or '++' indicates that the pattern resolves the corresponding force in a good or optimal way. In turn, a '-' or '- -' indicates a rather bad resolution of the force[3]. For example, the pattern PACKET FILTER resolves the forces *cost*, *performance, and* usability in an optimal way. This is bought by a rather low level of security (compared to a PROXY). On the other hand, the pattern PROXY provides a high level of security that leads to losses of *performance* and *usability*. Note that a PACKET FILTER is usually cheaper than a PROXY, i.e. you have to pay the price for the higher security level.

The arrows between the boxes indicate the *specialize* relation between the patterns

[2]The analogy to UML breaks down sooner or later. For example, the pattern name echoes the solution and can often be about dynamic actions, while a class name tends to be a "thing", not an action.

[3]We haven't provided such quality trends of a pattern for the FIREWALL pattern due to its more generic nature.

as described before. For example, the PACKET FILTER *specializes* the generic FIREWALL pattern. That way a hierarchy of the patterns is established. The more generic patterns are intended for “decision makers” and the more concrete patterns should guide those who are responsible for the implementation of a security concept and have to meet certain boundary conditions (e.g. price and impact on usability).

Firewall

You operate a network that is connected to an external, untrusted network by means of a regular routing device. You can control the internal network but you have no influence over the external network. As such, both networks represent different administrative domains.

◊◊◊

How can you prevent that attackers can probe, access and misuse any system inside of the internal network?

It is unlikely that the access control facilities of all internal systems are activated and configured appropriately. In particular, out-of-the box installations offer standard services which can be misused by an attacker. Even if there are access restrictions it is unlikely that they are consistent, especially when more than one administrator is involved and there are no "global" guidelines.

Even worse, we can assume that most internal systems are not hardened: experience shows that patches aren't applied in time and that many, often unneeded services are running. This makes internal system also vulnerable to attacks at a rather low level.

Another basic threat is that the overall network topology is visible, i.e. an attacker can analyze possible targets without further burden.

Furthermore, it might happen that attacks can't even be detected as one cannot ensure that the audit facilities of the internal systems are activated and configured appropriately.

Therefore, you should restrict the ingoing and outgoing traffic at the border between the internal and the external network.

The regular router has to be replaced with a *firewall* system (or extended accordingly) that implements the following functions: analysis, access control, filtering and modification.

The firewall must be able to *analyze* the messages which are sent through it. The analysis itself can be conducted in several ways which differ in parameters such as technical implementation, quality, or granularity. For example, it is possible to analyze the semantics of specific protocol headers, the different states of a protocol, or the relationship between several parallel connections.

Based on the information gathered before, a firewall will be able to make an *access control decision*. First, the message has to be identified somehow, e.g. by the identity of the user, the process Id of the corresponding application or network addresses. With the given information the firewall can now decide whether a particular message is dangerous or not (hereby, a set of rules is usually evaluated). Possible *access control actions* are to grant, reject, discard, or modify messages.

A real world example for a firewall is the guard of a prison in a more oppressive country. The guard can examine all letters for and from the prisoners. He is basically in a position to decide whether a letter is passed or not.

The consequences of applying a firewall between two networks can be summarized as follows:

+ **Accountability.** As a firewall analyzes all messages which pass through it, rather fine-grained log information can be generated easily. Thus it is possible to detect possible attacks and to hold regular users responsible for their actions.

+ **Availability.** A firewall also helps to increase the availability of internal systems as the "attack surface" is made smaller significantly.

+ **Confidentiality/Integrity.** As there is an additional line of defense, the confidentiality and integrity of information hosted at the internal systems is increased, too.

- **Manageability.** A firewall is an additional, complex component of the network infrastructure. Thus, the efforts for managing the network are higher.

- **Usability.** Integrating a firewall often requires to change applications or the user's behavior. In either way, a firewall has an impact on the usability.

- **Performance.** The analysis and access control decision process consumes additional processing time. Thus, every firewall decreases the performance. Especially in high-bandwidth environments, this is an important issue.

- **Cost.** There will be additional costs for setting up and maintaining a firewall.

◊◊◊

More special variants of this pattern are PACKET FILTER and PROXY. The former is rather cheap and fast but provides security only at the Internet and Transport level. The latter is more expensive, more complex and slower but provides a high security level that covers all layers of the network stack.

In order to implement the firewall pattern, one of several available ACCESS CONTROL patterns have to be selected (e.g. [3]). It is also necessary to DEFINE IDENTIFICATION AND AUTHENTICATION DRIVERS [1] and to SELECT AN IDENTIFICATION AND AUTHENTICATION STRATEGY [4]. Furthermore, a firewall is some sort of a CHOKE POINT in a more general sense [9]. It should implement the security principle DEFENSE IN DEPTH and should FAIL SECURELY [5].

Packet Filter

You decided to restrict the ingoing and outgoing traffic at the border between the internal and the external network with a FIREWALL.

◊◊◊

How can you implement a firewall that is fast and doesn't decrease the usability too much?

An important issue is the performance between internal and external network. Any changes at the network border might have an impact on the available bandwidth.

Users don't want to bother to change their behavior or their applications in order to meet more or less abstract security needs. Usability is often the primary requirement that has to be satisfied.

Especially smaller companies cannot afford expensive security solutions. This includes both purchase and maintenance.

Therefore, you should restrict network traffic at rather low levels of the protocol stack.

According to a set of filter rules such a *packet filter* firewall can determine whether a particular packet is acceptable. Any packet that doesn't satisfy the security guidelines (implemented with the filter rules) will be rejected or discarded.

In particular, the packet filter analyzes the headers of the Internet and Transport layers in order to determine addresses, port numbers and flags of the given packet. That way, the firewall can , for instance, identify valid internal destination addresses and application port numbers. It is also basically possible to track the state of a particular connection. Possible access control actions are to grant, reject or discard packets.

Considering our example of a prison guard we can draw the analogy that the guard checks whether the prisoners are allowed to receive or send letters based on the postal address.

Beside the consequences of applying a generic firewall, the specific consequences of applying a packet filter firewall between two networks can be summarized as follows:

+ **Accountability.** A packet filter can log the details of processed packets as well as the access control decisions. For example, it can keep track of all packets which are rejected.

- **Manageability.** The filter rules can become complex very easily. Administrators have care about the order of the rule processing and ensure that the firewall performs its task as intended.

+ **Usability.** As packet filters operate at lower levels of the protocol stack, they are more or less transparent for users and applications.

+ **Performance.** Processing every packet decreases the performance. However, a packet filter is a rather fast firewall as it works at lower levels and can be implemented within the kernel. Nevertheless, Denial-of-Service attacks are still possible.

+ **Cost.** Many routers are shipped with packet filter functionality, i.e. the additional costs are minimal.

◊◊◊

This pattern specializes the FIREWALL pattern and is a variant to the PROXY pattern. The relations of the FIREWALL pattern are "inherited".

Proxy

You decided to restrict the ingoing and outgoing traffic at the border between the internal and the external network with a FIREWALL.

◊◊◊

How can you achieve the highest level of protection?

It is important to have a fine-grained view on the packets which travel from the inside to the outside network and vice versa. Otherwise you might miss important information for making an appropriate access control decision. In the worst case, you'll take the wrong access control decision.

When you want to reach such a security level, you are often willing to sacrifice performance and/or usability.

Therefore, you should virtually separate the networks and restrict network traffic at the application level of the protocol stack.

Such a *proxy firewall* is able to analyze application related information as well as any data that can be derived from the lower layers of the protocol stack. The packet moves up the protocol stack and is only passed to the other side of the network when the rules allow it, i.e. there is no direct connection between client and server. In addition to network addresses and application ports, specific elements of an application protocol can be taken into account when making an access control decision. For example, the user's account name can be determined.

As proxies split up network connections, additional security services can be integrated, e.g. authentication servers. A proxy can also modify certain parts of a message, e.g. replace particular (dangerous) commands. However, this only works in a straightforward way if the protocols are not encrypted.

Basically, a proxy requires to rebuild protocols for each application. As this can become complex, there are only proxies for the most popular applications available.

As proxies re-implement protocols, they also protect against implementation faults in the protocol stacks of the internal systems.

Our real-world prison guard would check the address of the letter. Furthermore, he would open the letter and search for offensive content (e.g. a prisoner tries to arrange further crimes from the inside). If necessary, the guard censors the letter or holds it back.

Beside the consequences of applying a generic firewall, the specific consequences of applying a proxy firewall between two networks can be summarized as follows:

+ **Accountability.** Proxies can log virtually any information contained in a packet at a fine-grained level.

- **Manageability.** In order to maintain a proxy firewall, administrators have to be familiar with the corresponding application protocols. The effort increases for each proxy that is used.

- **Usability.** Often it is required to change the user's behavior (login to a proxy first and then connect to internal/external systems) or to modify applications (configure which proxy should be used). Problems can occur, whenever several proxies are cascaded.

- **Performance.** As proxies analyze packets in a very detailed way, they are comparably slow firewall solutions. However, a proxy can also improve performance assuming it has caching capabilites.

- **Cost.** Proxies are more expensive than packet filters. It could happen that proxies have to be customized which increases the cost further.

◊◊◊

This pattern specializes the FIREWALL pattern and is a variant to the PACKET FILTER pattern. The relations of the FIREWALL pattern are "inherited".

Acknowledgments

I like to thank my shepherd Rick Dewar for his valuable comments and helpful suggestions for improvement. I also want to thank Utz Roedig for discussing early drafts of the patterns included in this document with me. I also want to thank all people of Workshop C and all the good girls and bad boys of EuroPLoP.

References

[1] Susan Chapin and Duane Hybertson. Define I&A Design Drivers. The MITRE Corporation, 2003.

[2] D. Brent Chapman and Elizabeth D. Zwicky. *Building Internet Firewalls*. O'Reilly & Associates, Inc., 1995.

[3] Eduardo B. Fernandez and Rouyi Pan. A Pattern Language for Security Models. Technical report, Florida Atlantic University, 2001. PLoP.

[4] Jody Heaney, Susan Chapin, Ann Reedy, Duane Hybertson, and Jr. Malcolm Kirwan. Select I&A Strategy. The MITRE Corporation, 2003.

[5] Sasha Romanosky. Enterprise Security Patterns. In *Proc. of EuroPLoP*, 2002.

[6] Markus Schumacher. *Security Engineering with Patterns*. PhD thesis, Darmstadt University of Technology, 2003.

[7] Markus Schumacher and Utz Roedig. Security Engineering with Patterns. In Proc. of the 8th Conference on Pattern Languages of Programming (PLoP), 2001.

[8] Markus Schumacher, Utz Rödig, and Marie-Luise Moschgath. *Hacker Contest - Sicherheitsprobleme, Lösungen, Beispiele*. Xpert.press. Springer Verlag, 2002. ISBN 3-540-41164-X.

[9] Joseph Yoder and Jeffrey Barcalow. Architectural Patterns for Enabling Application Security. *PLoP*, 1997.