4.3 Service Firewall

I mentioned in the Secure Message pattern that messages travel in "no-man's land". You can use the Secure Message or the Secured Infrastructure patterns helps deal with protecting the messages while traveling that space - but what do we do if the sender is malicious? An attacker send us a malicious messages (e.g. with a virus as a SOAP attachment) and it wouldn't really help us that we manage to ensure that malicious message got to us intact and securely.

4.3.1 The Problem

To illustrate the type of attacks a malicious sender can cause let's look at one of them a little closer. Figure 4.4 below an XML Denial of Service (XDoS) attack. In this type of attack a malicious sender attaches a lot of digital signatures to a message. Parsers that aren't ready for this type of attack examine each of these signatures causing the service to slow down under the load.

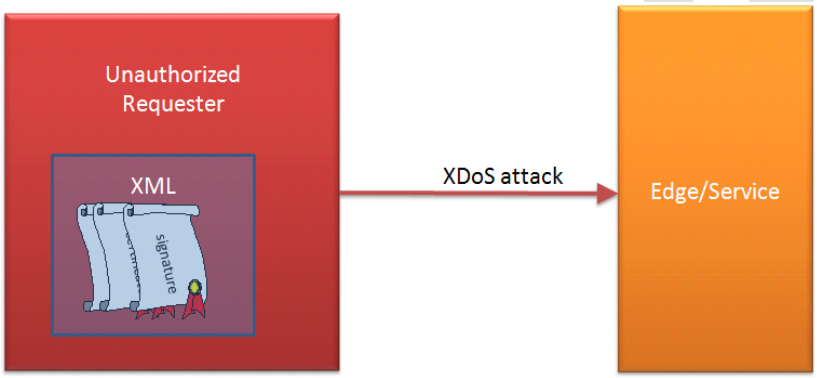

Figure 4.4: Illustration of a XDoS attack. A malicious sender prepares an XML that looks valid but is loaded with a lot of digital signatures. An unsuspecting parser will try to verify each of these signatures while hogging down CPU cycles which can result in unavailability of the service.

Attacks using incoming messages as demonstrated in Figure 4.4 are one type of threat we need to handle, another related type of threats or problems has to do with outgoing messages. Here we need to make sure that private or classified information does not leak outside of the service. In this type of scenarios we want to find a way to make sure they hold only information allowed in the contract flows out of the service.

> **How can you protect a service against detect malicious incoming messages and prevent information disclosure on outgoing messages?**

One option for dealing with malicious senders is to apply the Secured Infrastructure pattern (mentioned earlier in this chapter) and require certificates for authorizing clients. This means that clients that do not have a certificate will not be allowed to contact the systems. One problem with this approach is that it is only good when the number of service consumers is controlled and not open for the general

public (e.g. exposed on the internet). Another limitation of the certificate approach is that it doesn't handle attacks by insiders since they are authorized to access the system.

Another option is to incorporate the security logic that screens malicious content as part of the business logic. There are several problems with this approach the problem with this approach is that you get code duplication as there are many threats that are common to all services. Another problem is a that you the business logic is tainted with the security logic which makes it harder to write and maintain.

The better option is to externalize the security to another component. Let's look at this option more closely.

4.3.2The Solution

SOA messages are application level components – however. the notion of messages is not new or unique to SOA. We (the computer industry) already have a lot of experience with messages on a lower level of the OSI stack - the network level, specifically TCP packers and UDP datagrams. TCP and UDP have few similarities with SOA messages, the interesting ones, for the purpose of this pattern are the threats they face. Since the threats are similar maybe we can also use the same solutions we've found to work for TCP and use them for our SOA messages.

Implement the Service Firewall pattern and intercept incoming and outgoing messages and inspect them in a dedicated software component or hardware.

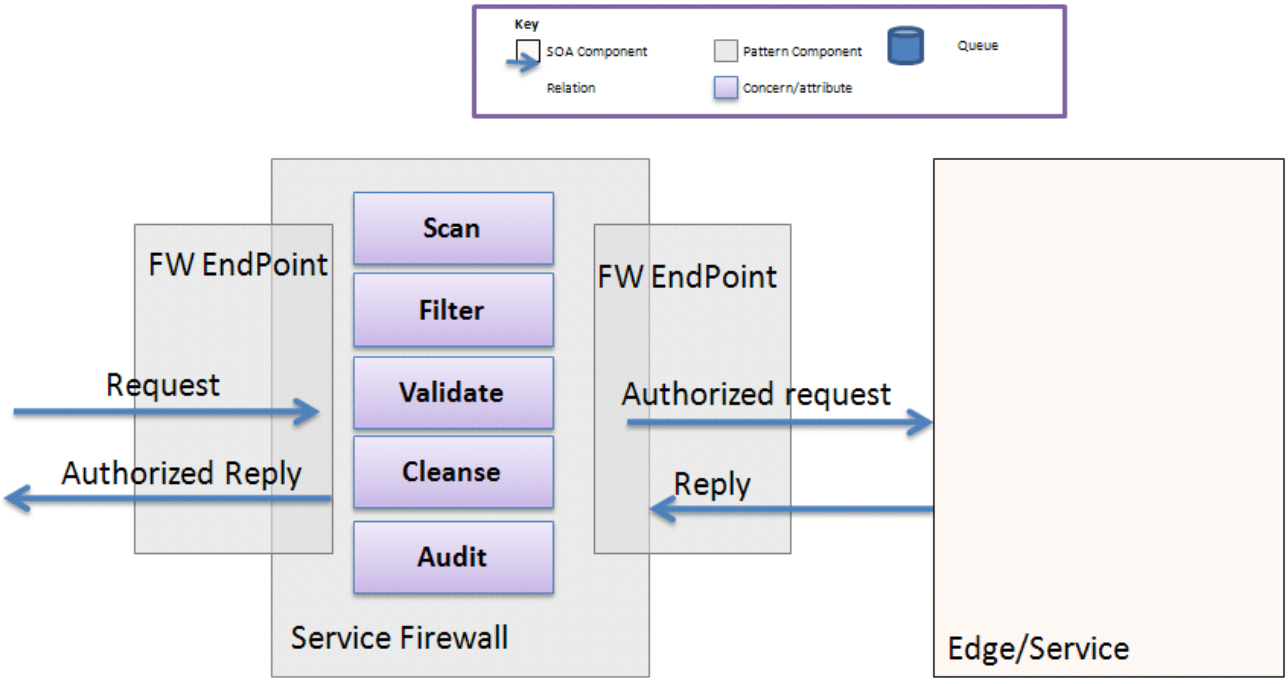

Figure 4.5 The Service Firewall Pattern. The Service Firewall sits between the outside world and the actual service (or Edge). The Service Firewall scans, validate and audit both incoming and outgoing messages. Once a message is identified as problematic it can either be filtered out or cleansed.

The Service Firewall pattern is an application of the Edge Component pattern (see Chapter 2). Figure 4.5 above illustrates how the Service Firewall operates. First it intercepts each incoming and outgoing message and inspects it. Once intercepted the Service firewall can scan the message for malicious content such as viruses or XDOS attacks as mentioned in the sample scenario. Additionally, the Service Firewall can validate messages by making sure they conform to the contract, verifying property types and sizes etc. When a message is identified as problematic the Service Firewall can audit and log the message and then decide whether to filter it out or cleanse the problematic content and let it through.

The Service Firewall acts as a first line of defense for the service. As illustrated in Figure 4.5 below, when a request arrives at the firewall it is scanned and verified, requests that were authorized are then routed to the real service (or another Edge Component)

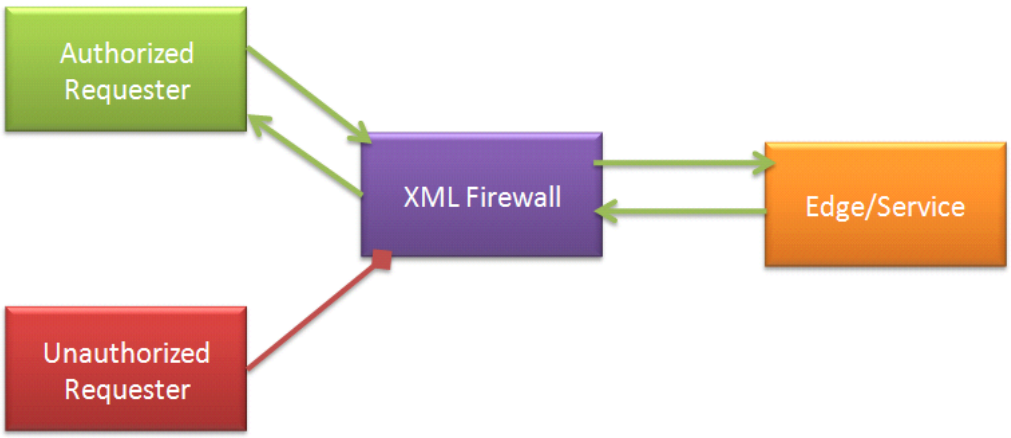

Figure 4.5 When a request arrives at a Service Firewall (an XML firewall in this illustration) it is screened for validity, for instance they firewall can check that an XML matches the predefined XSD. Authorized requests get through and unauthorized requests are rejected.

The idea behind a Service firewall is simple, the implementation, however is a little more complicated since there is a lot of functionality that has to be implemented to get each of the roles (scan, validate, filter etc.). On top of that you need a way to make sure the Service Firewall gets all the messages to be able to work

4.3.3 Technology Mapping

As mentioned in the patterns structure in Chapter 1, the technology mapping section takes a brief look at what does it mean to implement the pattern using existing technologies as well as mention instances where current technologies implement the pattern.

The simplest way to implement the Service Firewall pattern is to create a designated Edge Component, deploy it on the DMZ. Deploy the real service behind a regular firewall and

Implementing the Service Firewall pattern without using a regular firewall is a little more problematic as an attacker can just call the endpoints that is used by the actual service and bypass the Service Firewall altogether. In these situations you can rely on the interception capabilities of the technology you use. For

instance, Figure 4.6 below shows the relevant extension points offered by Windows Communications Foundation for intercepting incoming messages.

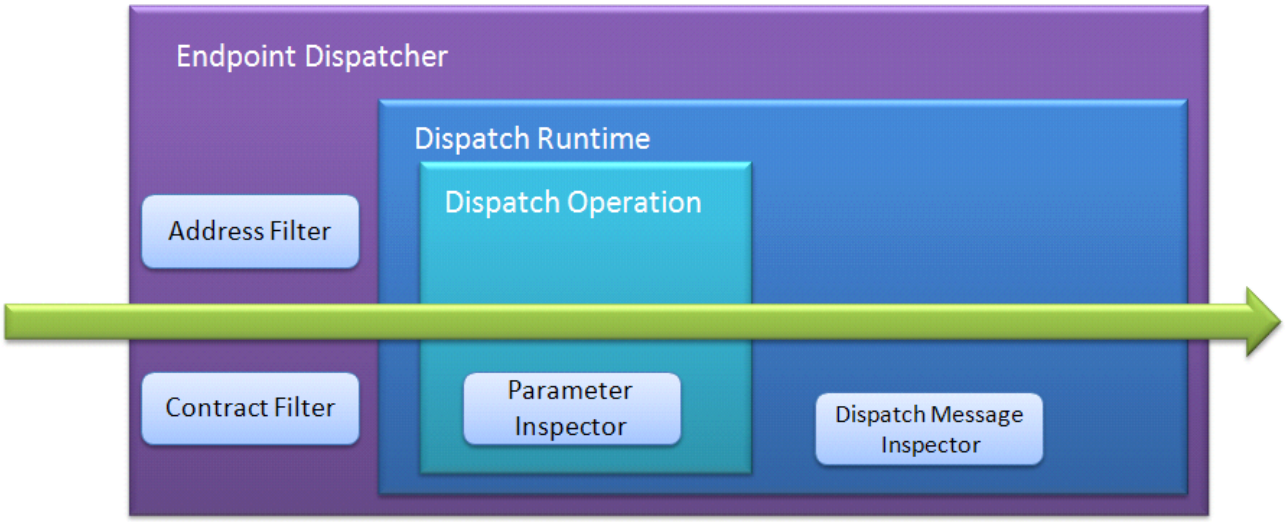

Figure 4.6. WCF supports a few dozens of extension points to control the way a message is handled when in enters or leave the service. You can use four of these extension points (depicted above) to implement the different roles defined in the Service Firewall pattern.

As illustrated in Figure 4.6 there are four relevant extension point (out of the few dozens of extension points supported by WCF) where you use classes to perform the various roles of the Service Firewall pattern. You can have classes that and verify addresses, verify contract, inspect the messages and inspect parameters – both for incoming and outgoing messages. You can then use code like in Code listing 4.1 to add these classes as interceptors when a message passes in or out of the service.

```
Public void SetInterceptors(ServiceDescription serviceDescription,
ServiceHostBase serviceHostBase)
   {
      foreach (ChannelDispatcher ChannelDisp in
serviceHostBase.ChannelDispatchers)
       { foreach (EndpointDispatcher EndPointDisp in ChannelDisp.Endpoints)
         { EndPointDisp.DispatchRuntime.MessageInspectors.Add(new
         MyMessageInspector()); foreach (DispatchOperation op in
         epDisp.DispatchRuntime.Operations)
          op.ParameterInspectors.Add(new MyParameterInspector());
         }
    }
}
```

Code Listing 4.1. a WCF code snippet to add interceptors to incoming and outgoing messages handled by WCF. You can use these interceptors to implement the Service Firewall pattern and validate, scan etc. any message that passes in or out.

Another implementation option for the Service Firewall pattern is using hardware or embedded appliances. For instance, there are several companies like Layer7, IBM, Vordel and a few others that produce XML firewall appliances. The advantage of using XML appliances is that you can deploy them along with your other firewalls in the DMZ and have them serve as the first line of defense. Another

advantage is that these are platforms optimized for XML handling so the performance impact of the appliances is lower vs. self coded solution. One disadvantage of using hardware XML firewalls is setup costs (tens of thousands per unit) another disadvantage is increased maintenance complexity which comes both from managing an additional hardware type and more so from the double management of your SOA contract (both in the service and in the appliance).

Whether you use an appliance or implement the Service Firewall pattern in code it can really boost the security of your services by helping prevent threats like denial of service attacks or even just save some validation efforts for the service itself

4.3.4 Quality Attributes

As noted in the beginning of this chapter, the quality attributes section for security patterns discusses the threats that the pattern helps prevent.

The Service Firewall pattern is a very versatile pattern and it can be made to handle many types of threats. Table 4.4 below shows the threat categories and the actions that an implementation of the Service Firewall pattern can take to protect against threats in this category.

Threat	**Actions**
Tampering	Changing the content of request or a reaction
	Validating that messages are not mal-formed
Information Disclosure	Scanning outgoing messages for protected content
	Restricting reply addresses to a closed groups
Denial of Service	Preventing XDoS attacks by examining XMLs before validating each singnature
	Blocking known attackers
	Restricting Requestors addresses to a closed group
	Scanning attachments for viruses
Elevation of Privilege	Examining an incoming message for injection attacks
	Examining an incoming message for buffer overruns by validating contract and sizes of elements

Table 4.4 Threat categories and actions. List of the action that implementations of the Service Firewall can take and the threats these actions mitigate.

In Addition to the specifics of the threats that the Service Firewall pattern helps mitigate, we can also look at it from the wider scope of quality attributes. Like most of the patterns in chapter 4, the Service Firewall pattern is a security pattern. It is interesting to note that unlike most other security patterns it is relatively easy to add it on towards the end of a project. You should note however that it is not a completely free ride and for instance you still have to measure its impact on system performance, it can add an overhead in regards to contract maintenance etc.