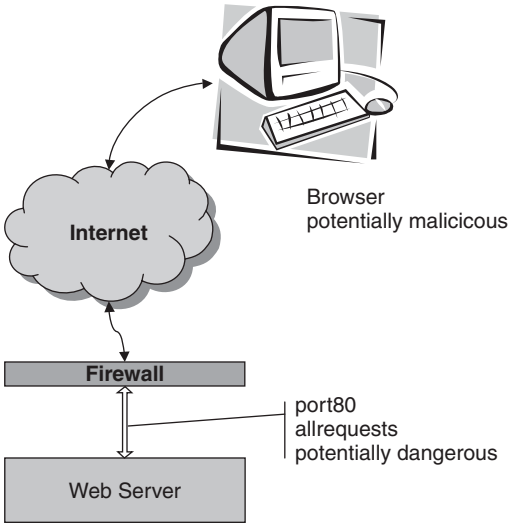
13.5 Protection Reverse Proxy

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 (457) protects the server software at the level of the application protocol.

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 (405).



Protection using a PACKET FILTER FIREWALL (405)

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?

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.

Problem

Even if you install a simple PACKET FILTER FIREWALL (405) 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?

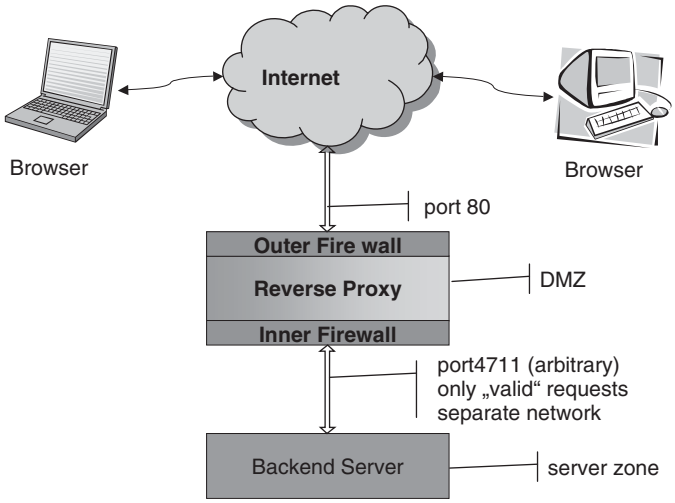
The solution to this problem must resolve the following forces:

- 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.

Solution

Change your network topology to use a PROTECTION REVERSE PROXY (457) 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 (405) ensure that no external network traffic reaches the real Web server.

The resulting network topology provides a DEMILITARIZED ZONE (449) containing only the reverse proxy machine, and a secured server zone containing the Web server.

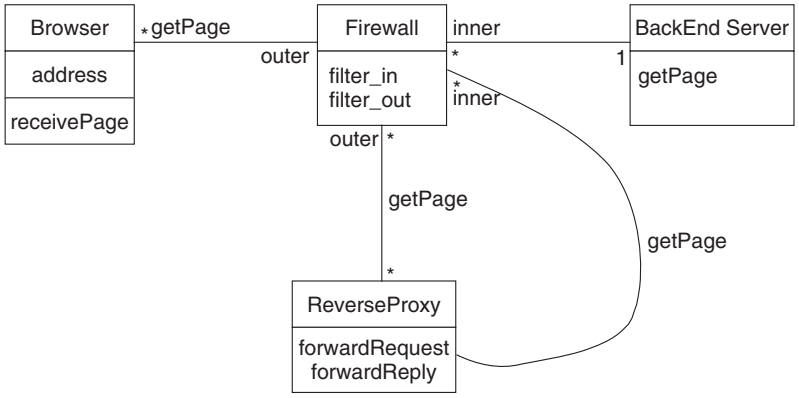


Protection using a PROTECTION REVERSE PROXY

Although this solution discusses only Web servers, it applies to other protocols like FTP, IMAP, and SMTP as well. A PROTECTION REVERSE PROXY (457) 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.

Structure

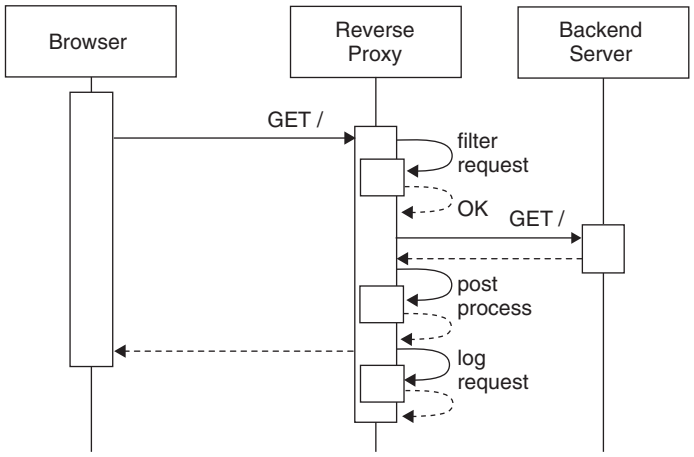
The class diagram for this pattern is shown below:



Class diagram for PROTECTION REVERSE PROXY

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.



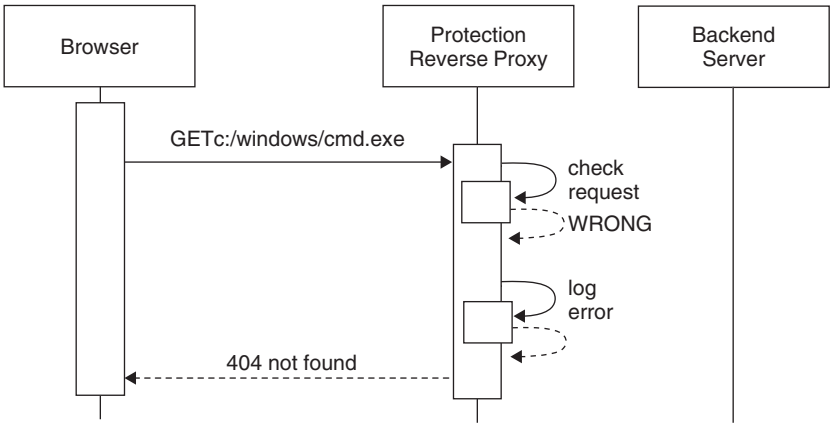
Allowing a client request in PROTECTION REVERSE PROXY

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 on page 461.

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



Refusing a client request in PROTECTION REVERSE PROXY

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: 'black lists' and 'white lists.'

- A *black list* filter only blocks requests that its list of malicious requests knows of, but passes all others. Black list filters are easier to deploy, but riskier. They are often used by 'higher-level' firewalls.
- A *white list* 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 white list filter needs to be adapted every time your back-end server changes significantly in its URL space. Nevertheless, it is the better choice for a PROTECTION REVERSE PROXY (457).

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.

Example Resolved

Following these implementation guidelines, we are able to protect our vulnerable Web server with a PROTECTION REVERSE PROXY (457).

Variants

INTEGRATION REVERSE PROXY (465) and FRONT DOOR (473) can (and should) be combined in their function with PROTECTION REVERSE PROXY (457), and thus vary this pattern by adding functionality.

Known Uses

PROTECTION REVERSE PROXY (457) 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 [Evidian] as well as IBM's Tivoli Access Manager [Tivoli].

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 (457), 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 (457). 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 (465) and FRONT DOOR (473).

The following potential liabilities may arise from applying this pattern:

- Black list filtering can give you a false sense of security. Like patches, black lists can only be constructed after a vulnerability is known.
- White list 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 white list filter of the PROTECTION REVERSE PROXY (457).
- 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 (457) requires you to configure an additional packet filter firewall, as well as another machine to run the reverse proxy on.

See Also

PROTECTION REVERSE PROXY (457) is a special implementation of SINGLE ACCESS POINT (279).

In conjunction with regular firewalls, PROTECTION REVERSE PROXY (457) builds on the principle of 'defence in depth'—see Section 15.1, *Security Principles and Security Patterns* and Schneier [Sch03b].