

INTERNET “PIZZO”

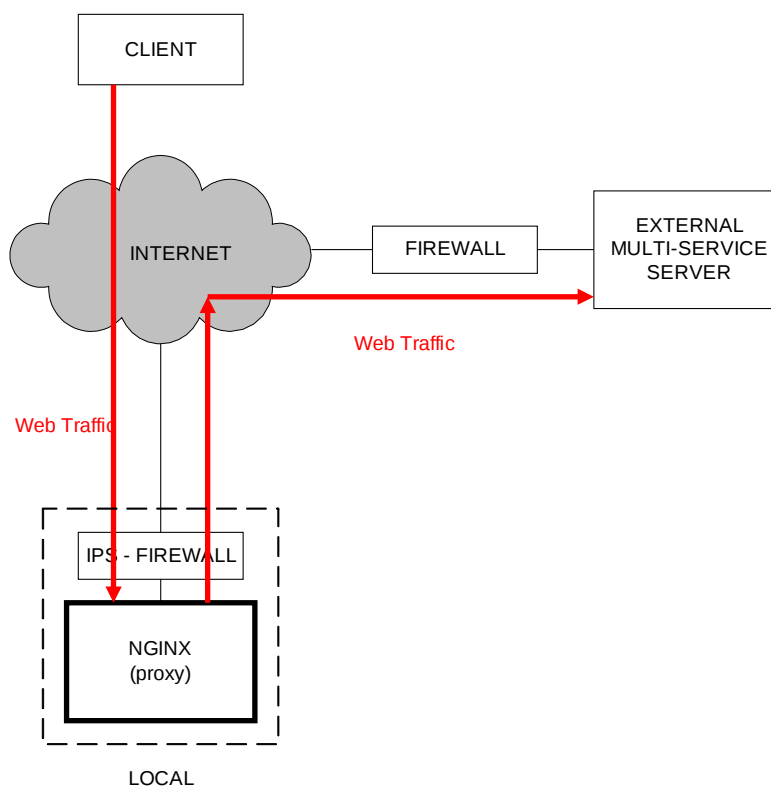
Pierpaolo Palazzoli, Brescia, Italy

Fabio Mostarda, Brescia, Italy

Claudia Ghelfi, Brescia, Italy

With the expression “Internet Pizzo” we may reference a system which is able to protect Internet services remotely; the Italian term “pizzo” (as “protection”) is hereby used in an ironic and mocking form, far distant from its literal meaning, which refers to the fee that mobsters require to assure “protection”. “Remote protection” is the peculiar and innovative feature of this system, i.e. the ability to protect services that are not physically close to the security infrastructure.

We will show that, thanks to a “NGINX” proxy and featuring proper IPS/firewall protection, it’s feasible to extend a security perimeter beyond geographical boundaries; the following diagram shows the logic structure of this architecture:



The logical flow of the system is:

1. We force DNS to redirect all internet traffic related to the customer service to the NGINX proxy.
2. This traffic is filtered by a firewall and by an IPS, and then it's redirected to the original external customer server using a secondary address.
3. Of course, it's suggested that the customer server is allowed to receive traffic from the NGINX proxy server only.

Every service request is thus separated in two different segments:

- o From the end-user to the NGINX proxy
- o From the trusted NGINX proxy to the customer server

Using this approach, only trusted and filtered traffic is serviced to the external server; likewise, every response from the external server to the client follows the same path in the opposite direction.

NGINX

Let's now have a closer look to NGINX; its features include http server, reverse proxy and IMAP/POP3 proxy: all those services can be delivered even by a single instance of this application.

NGINX main strengths are stability, a rich feature set, ease of configuration and a remarkably low system resource consumption; it is widely used in the following roles:

- o As an Apache substitute, featuring:
 - Support of heavy user-session loads, static files, index files and autoindexing.
 - Fast uncached reverse proxying and very easy load-balancing.
 - Remote FastCGI support and load-balancing.
 - Modular architecture, featuring gzipping, byte ranges, chunked responses, XSLT and SSI. Multiple SSI includes in the same page can be parallelly processed (in case they're handles by proxy or FastCGI servers)
 - SSL and TLS SNI support.
 - Ease of reconfiguration.
 - Online reconfiguration and upgrade, without service downtime.
 - IP-based client access control and basic http authentication.
 - Band limiting features.
 - Ability to limit the number of simultaneous sessions or of requests coming from specific source addresses.
- o As a proxy server, leveraging NGINX load-balancing capabilities.

- o As a proxy mail server, featuring:
 - IMAP/POP3 redirection, featuring an external http authentication server.
 - External http server authentication, featuring a redirected link to an internal SMTP server.
 - Authentication methods:
 - POP3: USER/PASS, APOP, AUTH LOGIN PLAIN CRAM-MD5;
 - IMAP: LOGIN, AUTH LOGIN PLAIN CRAM-MD5;
 - SMTP: AUTH LOGIN PLAIN CRAM-MD5;
 - SSL support.
 - STARTTLS e STLS support.

NGINX supports the following Operating Systems:

- FreeBSD 3.x, 4.x, 5.x, 6.x i386; FreeBSD 5.x, 6.x amd64;
- Linux 2.2, 2.4, 2.6 i386; Linux 2.6 amd64;
- Solaris 8 i386; Solaris 9 i386 and sun4u; Solaris 10 i386;
- MacOS X (10.4) PPC;

It can be freely downloaded at <http://sysoev.ru/nginx/download.html> and requires the pcre, openssl, zlib libraries in order to install.

APPLICATION

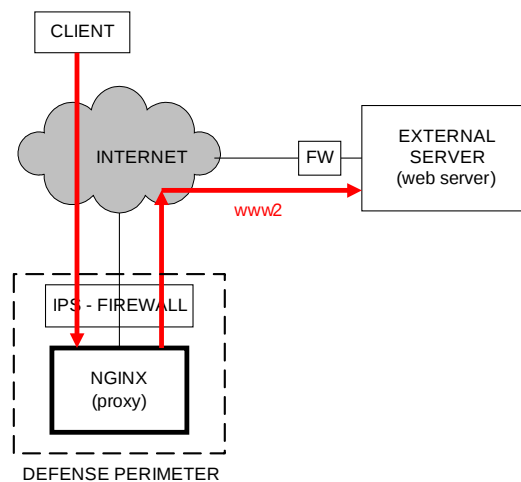
In our setup, NGINX is used as reverse proxy. By forcing the user accesses from the internet to flow through NGINX, it is feasible to protect the customer services without any need to install a protective architecture close to the server, which is actually hosting the services.

According to the nature of the services to be protected, there could be a number of applications, from web sites protection to email infrastructure remote shielding.

WEB PROTECTION

As a sample case study, we'll analyze the security infrastructure for a web site (i.e. www.ilpizzodiinternet.it).

First of all, to remotely defend a web publishing system leveraging our setup, it's mandatory to have some configurations (detailed in the „prerequisite“ section) in order to build the following achitecture:



Once we set up the operational environment, it will suffice to correctly configure NGINX in order to have the proper service forwarding, thus allowing the traffic to flow through the defense perimeter.

This perimeter, numbering the NGINX server, IPS and firewall, acts as a gateway which all traffic is forced to flow through, enabling centralized packet inspection and security policing.

Additionally, other NGINX features can be offered to the customer, i.e. load balancing.

Prerequisites

In order to get the setup working, it's mandatory that:

- o All traffic to the web server is redirected to the defense perimeter (NGINX); thus, it's necessary to modify the DNS configuration file for the website, adding the record: `www IN A "ip nginx"`
- o We have now to setup the link between the NGINX server and the customer equipment; we'll use another DNS record, like `www2`. We'll thus add in the DNS server: `www2 IN A "external server ip"`
- o Finally, we suggest to enforce the external customer server to accept connections from the NGINX server only; this can be easily accomplished with a firewall.

Remarkably, the URL that will appear within the end user browser will be the „public“ one, and not the private one linking the NGINX server to the customer server (i.e. „www2“).

In fact, this setup is totally transparent to the end user experience.

NGINX Configuration

NGINX has to be configured in order to channel the traffic towards the customer web site. The configuration file is located at `/etc/nginx/nginx.conf` and it has to be edited following the guidelines that are shown below; the bold lines are the one requiring editing.

```
user www-data;
worker_processes 1;

error_log /var/log/nginx/error.log;
pid /var/run/nginx.pid;

events {
    worker_connections 1024;
}

http {
    include /etc/nginx/mime.types;
    default_type application/octet-stream;

    access_log /var/log/nginx/access.log;

    server {
        listen 80;
        server_name www.ilpizzodinternet.it; #sito da proteggere
        location / {
            proxy_pass http://www2.ilpizzodinternet.it; #sito a cui reindirizzare il traffico
        }
    }

    sendfile on;
    #tcp_nopush on;

    #keepalive_timeout 0;
    keepalive_timeout 65;
    tcp_nodelay on;

    gzip on;

    include /etc/nginx/conf.d/*.conf;
    include /etc/nginx/sites-enabled/*;
}
```

ALTERNATIVE SETUP

This architecture can be also implemented using mod_proxy for Apache instead of NGINX; in order to setup this second method, it will suffice to insert in the Apache configuration file (httpd.conf) the following lines:

```
<VirtualHost www.ilpizzodinternet.org>  
  ServerAdmin admin@ilpizzodinternet.org  
  ProxyPass / http://www2.ilpizzodinternet.org/  
  ProxyPassReverse / http://www2.ilpizzodinternet.org/  
  ServerAlias www.ilpizzodinternet.org www.ilpizzodinternet.net  
    www.ilpizzodinternet.com  
  ServerName www.ilpizzodinternet.org  
  ErrorLog logs/www.ilpizzodinternet.org_error_log  
  CustomLog logs/www.ilpizzodinternet.org_log combined  
</VirtualHost>
```