Computer Security Principles: Reducing attack vectors with Reverse Proxies

Christophe Debruyne

2020-06-30

Outcomes

- Understand the of reverse proxies in securing networked systems
- Understand the advantages and disadvantages of reverse proxies
- Understand the examples provided with this presentation (link below)

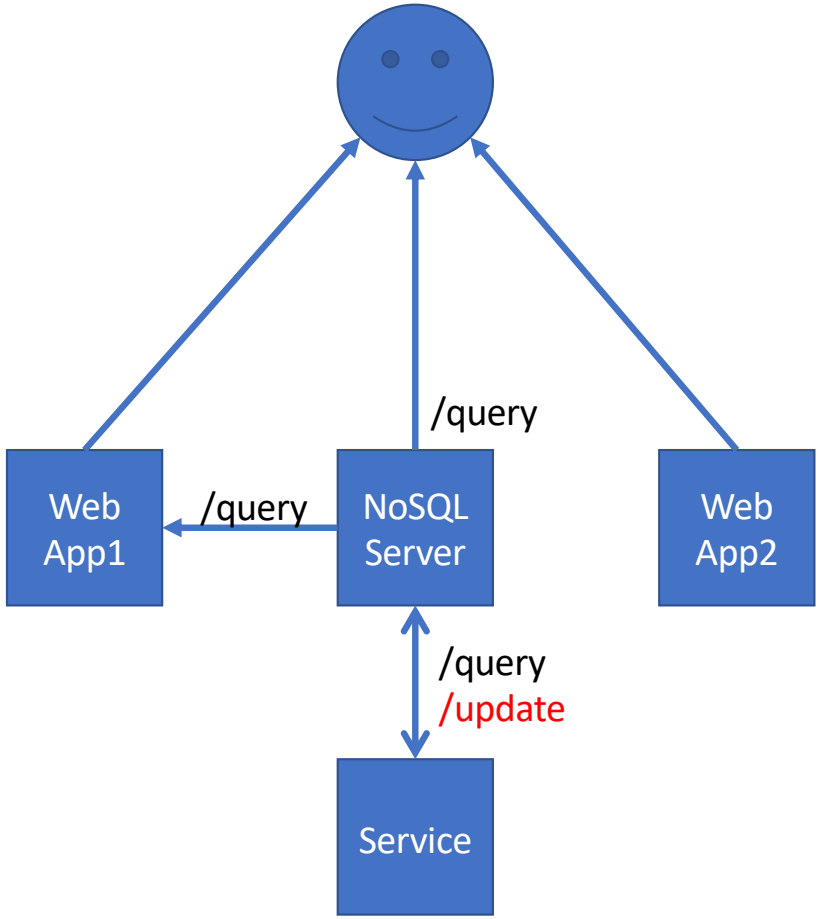
Examples are based on Apache2

With additional reading, you should be able to

- Be able to apply, tailor, and extend the examples with Apache2
- Be able to extrapolate the examples using Nginx

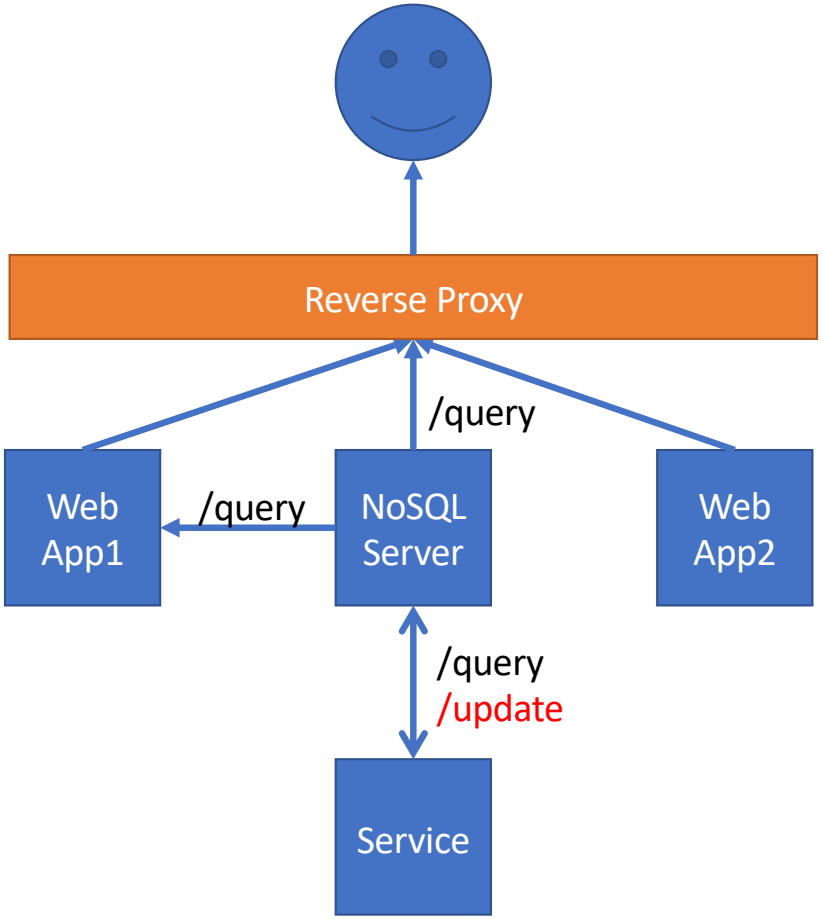
Motivation

- Providing multiple services to multiple agents (human and computer-based)
- Services may run on different (virtual) machines, operating systems, etc. and can interact with each other
- Each machine (and service) becomes an *attack vector* and together form a large *attack surface*.



Motivation

- *Reverse proxies* allow a server (usually a dedicated machine) to act as a single access point
- You can think of it as a façade controlling how certain requests are delegated
- Reducing outward facing attack vectors, and thus the attack surface

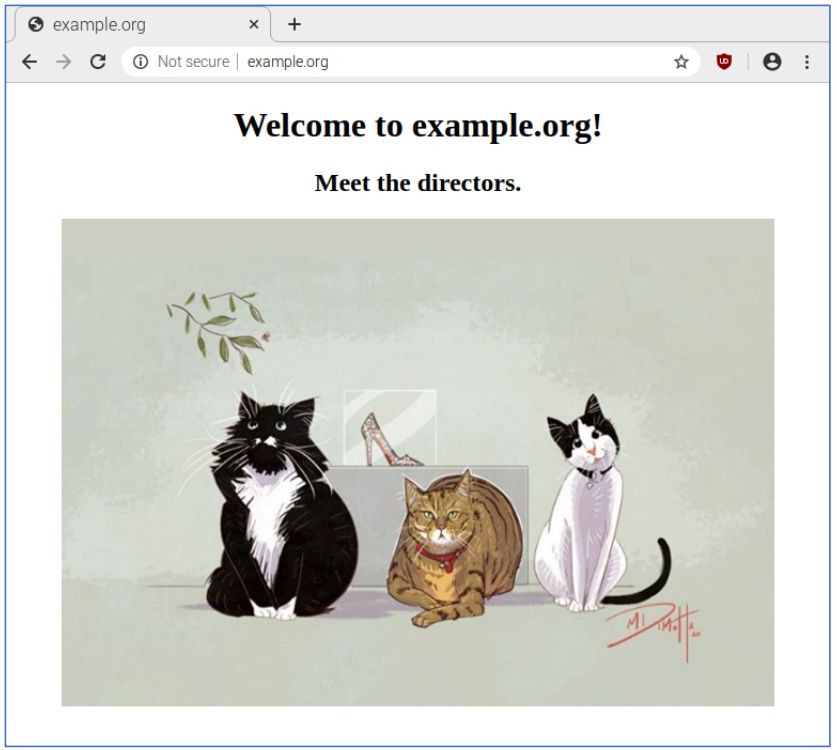


Scenario

- Our Apache2 server, with IP address *192.168.0.33*, is listening to port *80*. The IP address is tied to the domain *example.org*.
- We have an application, *system1*, listening on *192.168.0.16*:*5000*. Configure system1.example.org so that all requests are delegated to that application.
- We have an application, *system2*, listening on *192.168.0.16*:*5050*. System2 has a control panel (/control). Delegate all requests, except those to the control panel.

Approach

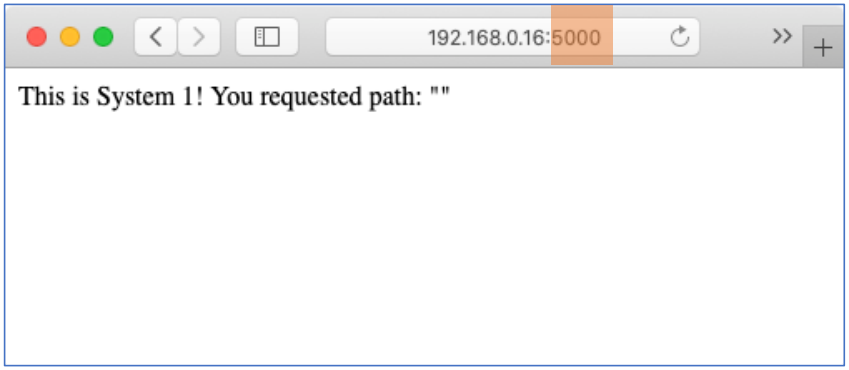
- First ensure that the domains www.example.org, system1.example.org, and system2.example.org are tied to the server running Apache2 by editing /etc/hosts (on *nix)
- Alternatively, you can avail of dnsmasq

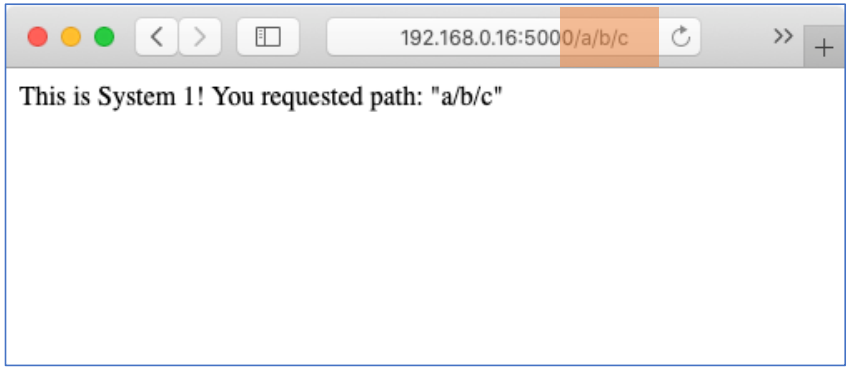


System1

```
from flask import Flask
app = Flask(__name__)

# Catch all the paths in System 1
@app.route('/', defaults = { 'path': '' })
@app.route('/<path:path>')
def System1(path):
    return f'This is System 1! \
    You requested path: "{path}"'
```





/etc/apache2/sites-enabled/001-system1.conf

```
<VirtualHost *:80>
  ServerName system1.example.org
  ProxyPreserveHost On
  ProxyPass "/" "http://192.168.0.16:5000/"
  ProxyPassReverse "/" "http://192.168.0.16:5000/"
</VirtualHost>
```

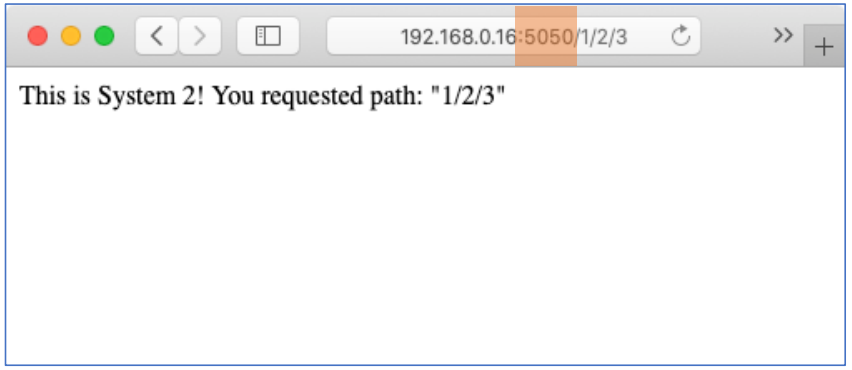
Restart the Apach2 Server, and…

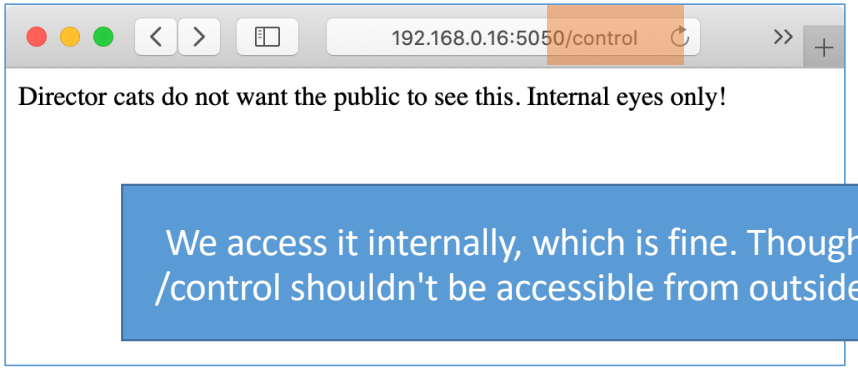
System2

```
from flask import Flask
app = Flask(__name__)

# All paths except for /control are caught by
# the System2 function.
@app.route('/control', defaults = { 'path': '' })
@app.route('/control/<path:path>')
def control(path):
    return 'Director cats do not want the \
    public to see this. Internal eyes only!'

@app.route('/', defaults = { 'path': '' })
@app.route('/<path:path>')
def System2(path):
    return f'This is System 2! \
    You requested path: "{path}"'
```



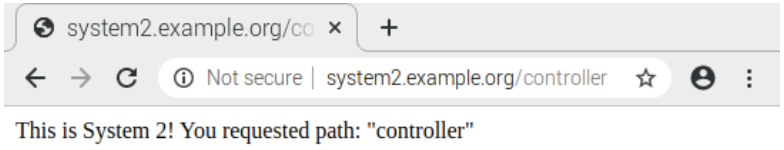


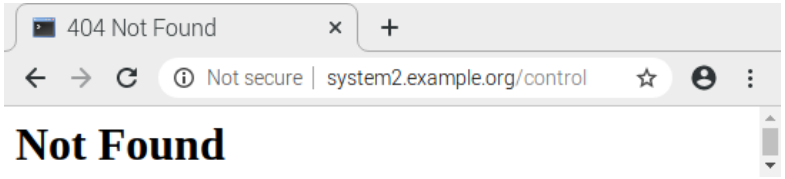
We access it internally, which is fine. Though /control shouldn't be accessible from outside.

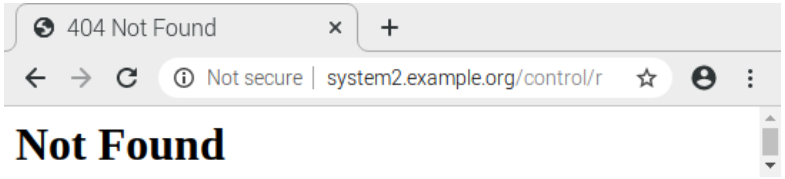
/etc/apache2/sites-enabled/002-system2.conf

```
<VirtualHost *:80>
    ServerName system2.example.org
    ProxyPreserveHost On
    ProxyPassMatch  ^\/control(/(.)*)?$  !
    ProxyPass  "/"  "http://192.168.0.16:5050/"
    ProxyPassReverse  "/"  "http://192.168.0.16:5050/"
</VirtualHost>
```

- The regular expression matches /control or anything that starts with /control/
- The ! Indicates that those URLs are ignored.







Reverse Proxies

Advantages/Benefits

- Single point of access
- Simplifies access control
- Hides details of the backend
- Users unaware of changes in backend
- Can be "nested"
  - a proxy server for the university delegating to proxy servers per department
- load balancing not failover (not covered)

Disadvantages/Risks

- Securing the proxy server
- Single point of failure
  - Provide safeguards for attacks
- Accidentally creating a forward proxy by setting proxy requests on.
  - Forward proxies can be used by clients to bypass firewalls.
- Overhead (translations, rewrite rules, checks, etc.)

Concluding remarks

- You hopefully understand what reverse proxies are and how they can be used to secure aspects of an organization's networked system.

- Not covered in this lecture are
  - reverse proxies between HTTPS and HTTP, which is straightforward
  - *rewrite rules*, which allow for greater control on how requests are passed onto other services, including content types.

- Reverse proxies are not only useful for reducing the attack surface, they also help
  - simplify complex systems (*separation of concerns*)
  - provide solutions for *load balancing*
  - *…*

References

- For Apache2: https://httpd.apache.org/docs/2.4/mod/mod_proxy.html
- For Nginx: https://docs.nginx.com/nginx/admin-guide/web-server/reverse-proxy/

- GitHub Repository with running example: https://github.com/chrdebru/reverse-proxy-tutorial
- Amore elaborate example (setting up a Linked Data Frontend): https://github.com/chrdebru/linked-data-frontend-tutorial