Nebraska University Consortium on Information Assurance

NUCIA

LAMP Secure Web Hosting

A.J. Newmaster & Matt Payne

8/10/2005

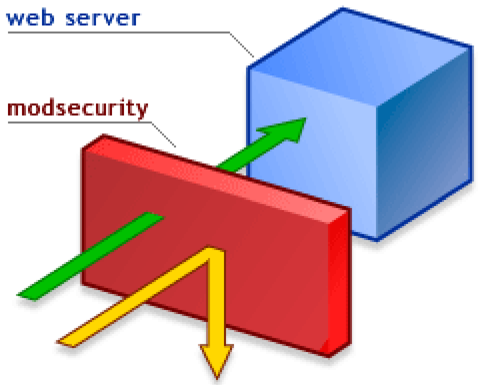
How do I lock down my server?

modsecurity

&

Jailkit

How can I use mod_security?



ModSecurity is an open source intrusion detection and prevention engine for web applications. Operating as an Apache Web server module, the purpose of ModSecurity is to increase web application security, protecting web applications from known and unknown attacks.

How can I use mod_security?

Introducing mod_security

Running public web applications may seem like playing Russian roulette. Although achieving robust security on the Web is possible in theory, there's always a weak link in real life. It only takes one slip of the code to allow attackers unrestricted access to your data. If you have a public web application of modest complexity running, chances are good that is has some kind of security problem.

How can I use mod_security?

Introducing mod_security

ModSecurity integrates with the web server, increasing your power to deal with web attacks. Some of its features worth mentioning are:

3. Request filtering
4. Anti-evasion techniques
5. Understanding of the HTTP protocol
6. POST payload analysis
7. Audit logging
8. HTTPS filtering

How can I use mod_security?

Introducing mod_security

1. Request filtering

incoming requests are analyzed as they come in, and before they get handled by the web server or other modules.

2. Anti-evasion techniques

paths and parameters are normalized before analysis takes place in order to fight evasion techniques.

How can I use mod_security?

Introducing mod_security

3. Understanding of the HTTP protocol

since the engine understands HTTP, it performs very specific and fine granulated filtering.

4. POST payload analysis

the engine will intercept the contents transmitted using the POST method, too.

How can I use mod_security?

Introducing mod_security

5. Audit logging

full details of every request (including POST) can be logged for later analysis.

6. HTTPS filtering

since the engine is embedded in the web server, it gets access to request data after decryption takes place.

How can I use mod_security?

Introducing mod_security

Let’s look at the following example URL:

http://www.samplesite.abc/login.php?username=admin';
DROP%20TABLE%20users--

If your application is vulnerable to SQL injection, invoking the URL above may very well delete all user data from your application.

Did you make regular database backups?

How can I use mod_security?

Introducing mod_security

Fortunately, the mod_security Apache module can protect you from this and other forms of web attacks.

To prevent the "drop table" SQL injection attack with mod_security, add the following to your Apache configuration:

```
SecFilter "drop[[:space:]]table"
```

How can I use mod_security?

Installing mod_security

http://www.modsecurity.org/download/

When installing from source you have two choices: to install the module into the web server itself, or to compile mod_security into a dynamic shared object (DSO).

How can I use mod_security?

Configuring mod_security

Turning filtering on and off

Filtering engine is turned off by default. To use it, you need to turn it on:

```
<IfModule mod_security.c>
    # Turn the filtering engine On or Off
    SecFilterEngine On
</IfModule>
```

How can I use mod_security?

Configuring mod_security

Turning filtering on and off

Supported parameter values for this parameter are:

–On – analyze every request

–Off – do nothing

–DynamicOnly

How can I use mod_security?

Configuring mod_security

Turning filtering on and off

DynamicOnly – analyze only requests generated dynamically at runtime. Using this option will prevent your web server from using precious CPU cycles on checking access to static files.

How can I use mod_security?

Logging mod_security

Standard Apache logging will not help much if you need to trace back steps of a particular user or an attacker. The problem is that only a very small subset of each request is written to a log file. This problem can be remedied with the audit logging feature of mod_security.

How can I use mod_security?

Logging mod_security

Use the following 2 directives:

```
SecAuditEngine On
SecAuditLog /var/log/modsecure/audit_log
```

How can I use mod_security?

Logging mod_security

The SecAuditEngine parameter accepts one of four values:

3. **On – log all requests**
4. **Off – do not log requests at all**
5. **RelevantOnly – only log relevant requests. Relevant requests are those requests that caused a filter match.**
6. **DynamicOrRelevant – log dynamically generated or relevant requests. A request is considered dynamic if its handler is not null.**

How can I use mod_security?

URL Encoding validation

Special characters need to be encoded before they can be transmitted in the URL. Any character can be replaced using the three character combination %XY, where XY is an hexadecimal character code. Hexadecimal numbers only allow letters A to F, but attackers sometimes use other letters in order to trick the decoding algorithm. Mod_security checks all supplied encodings in order to verify they are valid.

How can I use mod_security?

URL Encoding validation

Turn URL encoding validation on with the following line:

```
SecFilterCheckURLEncoding On
```

How can I use mod_security?

Unicode encoding validation

Unicode encoding validation is disabled by default. You should turn it on if your application or the underlying operating system accept/understand Unicode.

You can turn Unicode encoding on with the following line:

```
SecFilterCheckUnicodeEncoding On
```

How can I use mod_security?

Byte range check

You can force requests to consist only of bytes from a certain byte range. This can be useful to avoid stack overflow attacks (since they usually contain "random" binary content).

```
SecFilterForceByteRange 32 126
```

This however is causing problems, since it only allows characters from ascii code 32 to 162 (obviously).

How can I use mod_security?

Byte range check

```
Request: 62.131.150.160 - - [23/Sep/2004:14:46:08 +0200] "GET
/fo/curator.php?achternaam=&check%5Bmaatschapsnaam%5D=on&maatschapsnaam
=Dani%EBls+Dijkman+%26+Huisman+Advocaten&plaats=&submit=zoeken
HTTP/1.1" 407 492
```

Dani%EBls = Daniëls

character ë (ascii 137) is used in the request, and translated to hex EB, ascii 235.

Result, false positive.

How can I use mod_security?

Default action

Whenever a filter is matched against a request, an action (or a series of actions) is taken. Individual filters can each have their own actions but in practice you will want to define a set of actions for all filters. You can do this with the configuration directive SecFilterDefatultAction.

```
SecFilterDefaultAction "deny,log,status:500“
```

How can I use mod_security?

Allowing others to see mod_security

Normally, attackers won't be able to tell whether your web server is running mod_security or not. You can give yourself away by sending specific messages, or by using unusual HTTP codes (e.g. 406 - Not Acceptable “encoding”). If you want to stay hidden your best bet is to use HTTP 500, which stands for "Internal Server Error". Attackers that encounter such a response might think that your application has crashed.

How can I use mod_security?

Allowing others to see mod_security

One technique that often helps slow down and confuse attackers is the web server identity switch. Web servers typically send their identity with every HTTP response in the Server: header. Apache is particularly helpful here, not only sending its name and full version by default, but it also allows server modules to append their versions too.

How can I use mod_security?

Allowing others to see mod_security

Mod_security offers a directive that will mask the identity of you Apache web server:

```
SecServerSignature “ApacheCon 2004 Las Vegas“
```

You will need to set Server Tokens to Full in the httpd.conf file, for this mod_security directive to work.

How can I use mod_security?

Directory traversal

If your scripts are dealing with the file system then you need to pay attention to certain meta characters and constructs. For example, a character combination "../" in a path is a request to go up one directory level. In normal operation there is no need for this character combination to occur in requests and you can forbid them with the following filter:

```
SecFilter "\.\./"
```

How can I use mod_security?

Directory traversal

Audit log entry for the SecFilter "\.\./“ rule

```
Request: 213.136.105.146 - - [25/Sep/2004:11:47:34 +0200] "GET
/scripts/..%255c%255c../winnt/system32/cmd.exe?/c+dir" 405 0
Handler: (null)
----------------------------------------
GET /scripts/..%255c%255c../winnt/system32/cmd.exe?/c+dir
mod_security-message: Access denied with code 405. Pattern match "\.\./"
at THE_REQUEST.
mod_security-action: 405
```

How can I use mod_security?

Cross site scripting attacks

Cross site scripting attacks (XSS) occur when an attacker injects HTML and/or Javascript code into your Web pages and then that code gets executed by other users. This is usually done by adding HTML to places where you would not expect them. A successful XSS attack can result in the attacker obtaining the cookie of your session and gaining full access to the application!

```
SecFilter "<[[:space:]]*script“
SecFilter "<.+>"
```

How can I use mod_security?

Cross site scripting attacks

```
SecFilter "<[[:space:]]*script“
```

The above filter will protect only against Javascript injection with the "<script>" tag.

```
SecFilter "<.+>“
```

This second filter is more general, and disallows any HTML code in parameters.

How can I use mod_security?

Cross site scripting attacks

You need to be careful when applying filters like this since many application want HTML in parameters (e.g. CMS applications, forums, etc). You can do this with selective filtering. For example, you can have the second filter from above

```
SecFilter "<.+>“
```

as a general site wide rule, but later relax rules for a particular script with the following code:

How can I use mod_security?

Cross site scripting attacks

```
<Location /cms/article-update.php>
        SecFilterInheritance Off
        # other filters here ...
        SecFilterSelective "ARGS|!ARG_body" "<(.|\n)+>“
</Location>
```

How can I use mod_security?

Cross site scripting attacks

```
GET /phpinfo.php?SERVER_ADDR="><script>alert('test');</script> HTTP/1.1

Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, application/x-shockwave-flash, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/msword, */*
```

```
mod_security-message: Access denied with code 405.
Pattern match "<( |\n)*script" at THE_REQUEST
```

Vulnerable XSS site.

Vulnerable XSS site with mod_security.

How can I use mod_security?

SQL/database attacks

Most Web applications nowadays rely heavily on databases for data manipulation. Unless great care is taken to perform database access safely, an attacker can inject arbitrary SQL commands directly into the database. This can result in the attacker reading sensitive data, changing it, or even deleting it from the database altogether.

How can I use mod_security?

Redirect

The following rule sends Google back home by redirecting Googlebot somewhere else, based on the User-Agent header. It does not log rule matches.

```
SecFilter HTTP_USER_AGENT "Google"
nolog,redirect:http://www.google.com
```

How can I use mod_security?

Operating system command execution

Web applications are sometimes written to execute operating system commands to perform operations. A persistent attacker may find a hole in the concept, allowing him to execute arbitrary commands on the system.

A filter like this:

```
SecFilterSelective THE_REQUEST "bin/“
```

will detect attempts to execute binaries residing in various folders on a Unix-related operating system.

How can I use mod_security?

Supporting Snort rules

Snort classifies rules into web attacks and web activities. Web attack rules are converted to reject incoming requests, while web activity rules only log the activity into the error log.

Mod_Security

Daily routine

- Review logwatch output to see if anyone is trying to FTP or SSH into the system who should not have access. Block them out in your firewall.

- Review chkrootkit and rootkit hunter output to see if your system has been compromised with a rootkit.
- Review firewall logs and port scan logs to see if anyone is attacking your servers. Block them out in your firewall.

http://www.**chkrootkit.org/**

http://www.**rootkit.nl/** ←NOTICE!! .nl

Mod_Security

Daily routine

- Change your root password or any root equivalent password on a regular basis.
- Review your system log files on a regular basis looking for errors and suspicious activity.
- Regularly review new technologies that are available for helping with security.

Mod_Security

Conclusion

Mod_security is a powerful tool, but can be overwhelming at first. Start with some simple rules, and plan your rules ahead.

In a shared hosting environment it's difficult to apply system wide rules, since you have no control over the way your users and customers program.

In a closed environment, like a intranet, all users will need to adhere to rules and policies

Mod_Security

Acknowledgement

Mod_security is created and maintained by Ivan Ristic from England. He has currently written a book called “Apache Security” by O'Reilly.

Mod_security home page is located at:

http://www.modsecurity.org

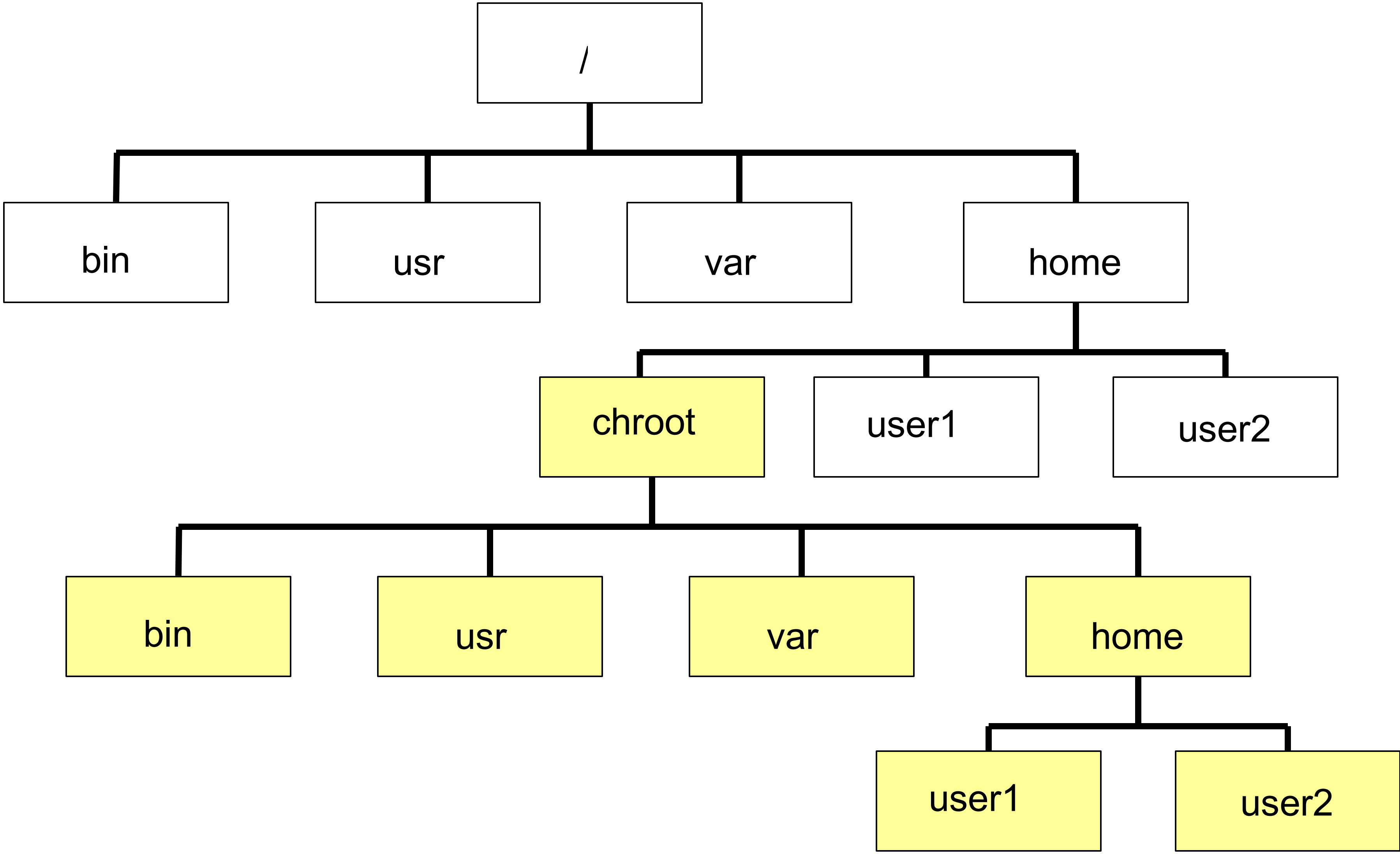
The previous slides have been borrowed from the Apachecon 2004 Conference in Las Vegas and edited for this presentation

Chroot Jails

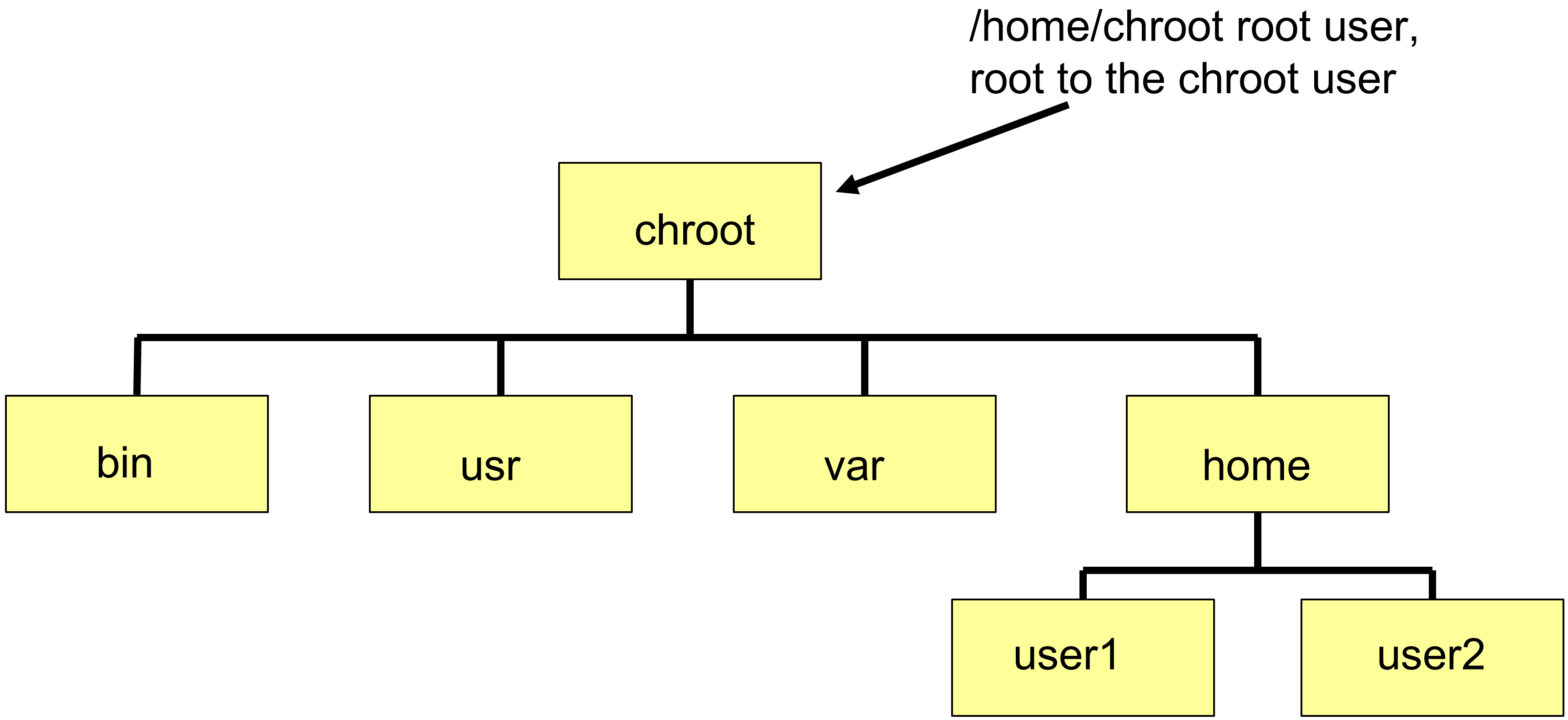
Putting Apache in a Chroot jail is quite time consuming and at times a tedious task (not to mention extremely boring!). The problem is that applications typically require shared libraries, and various other files and binaries to function properly. So, to make them function you must make copies of required files and make them available inside the jail. Mod_Security can make this a simple process by adding:

```
SecChrootDir /chroot/apache
```

In the configuration directory. If you don't want to do it this way, there is another tool that will help make chrooting easy…Jailkit. (But first an explanation on a chroot jail)


/
bin
usr
var
home
chroot
user1
user2
bin
usr
var
home
user1
user2

Chroot User Environment



Jailkit

“Jailkit is a set of utilities to limit user accounts to specific files using chroot() and or specific commands. Setting up a chroot shell, a shell limited to some specific command, or a daemon inside a chroot jail is a lot easier using these utilities.

Jailkit is often used on CVS servers (in a chroot and limited to cvs), sftp/scp servers (both in a chroot and limited to sftp/scp as well as not in a chroot but only limited to sftp/scp), and also on general servers with accounts where the shell accounts are in a chroot. Jailkit is furthermore used to jail daemon processes, for example apache servers, bzflag servers, squid proxy servers, etc.”

Jailkit

http://olivier.sessink.nl/jailkit/jk_lsh.8.html

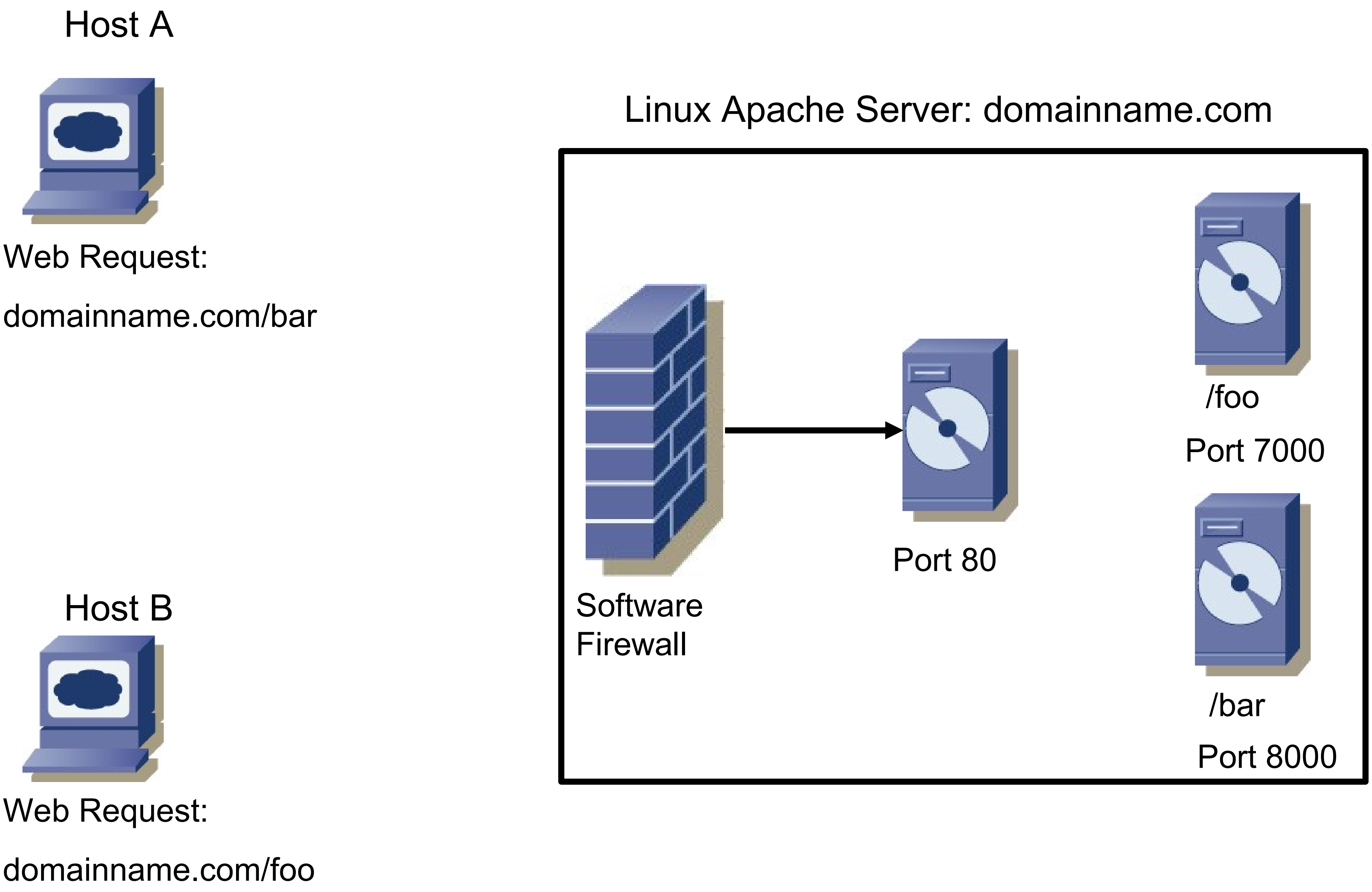
or

http://tinyurl.com/bgysq

A standard jail can be made in under 5 minutes with Jailkit.

Mod_proxy

- This module implements a proxy/cache for Apache. It implements proxying capability for FTP, CONNECT (for SSL), HTTP/0.9, and HTTP/1.0. The module can be configured to connect to other proxy modules for these and other protocols.



Host A

Web Request:

domainname.com/bar

Host B

Web Request:

domainname.com/foo

Linux Apache Server: domainname.com

Software Firewall

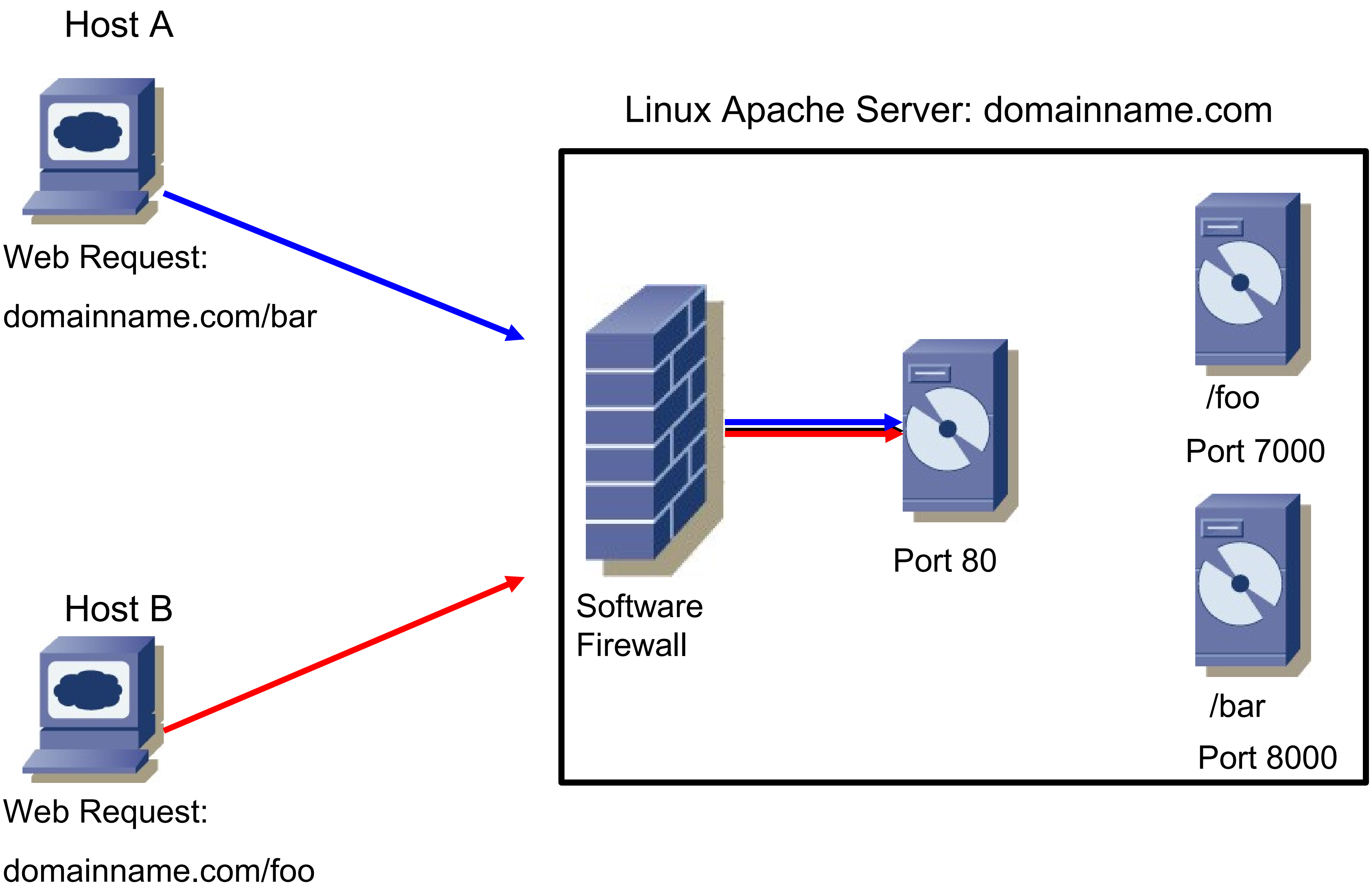
Port 80

/foo

Port 7000

/bar

Port 8000



Host A

Web Request:

domainname.com/bar

Host B

Web Request:

domainname.com/foo

Linux Apache Server: domainname.com

Software Firewall

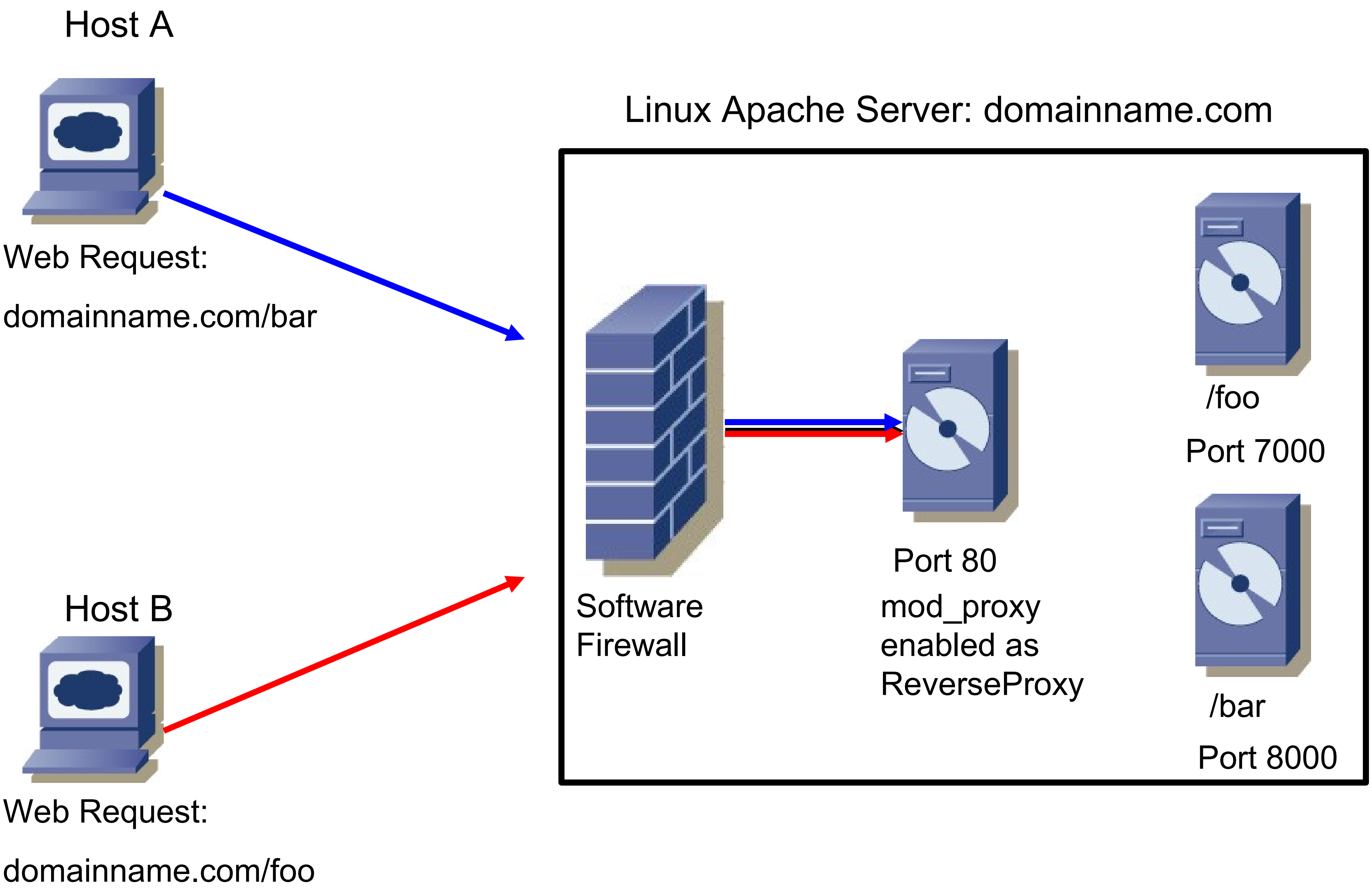
Port 80

/foo

Port 7000

/bar

Port 8000



Host A

Web Request:

domainname.com/bar

Host B

Web Request:

domainname.com/foo

Linux Apache Server: domainname.com

Software
Firewall

Port 80

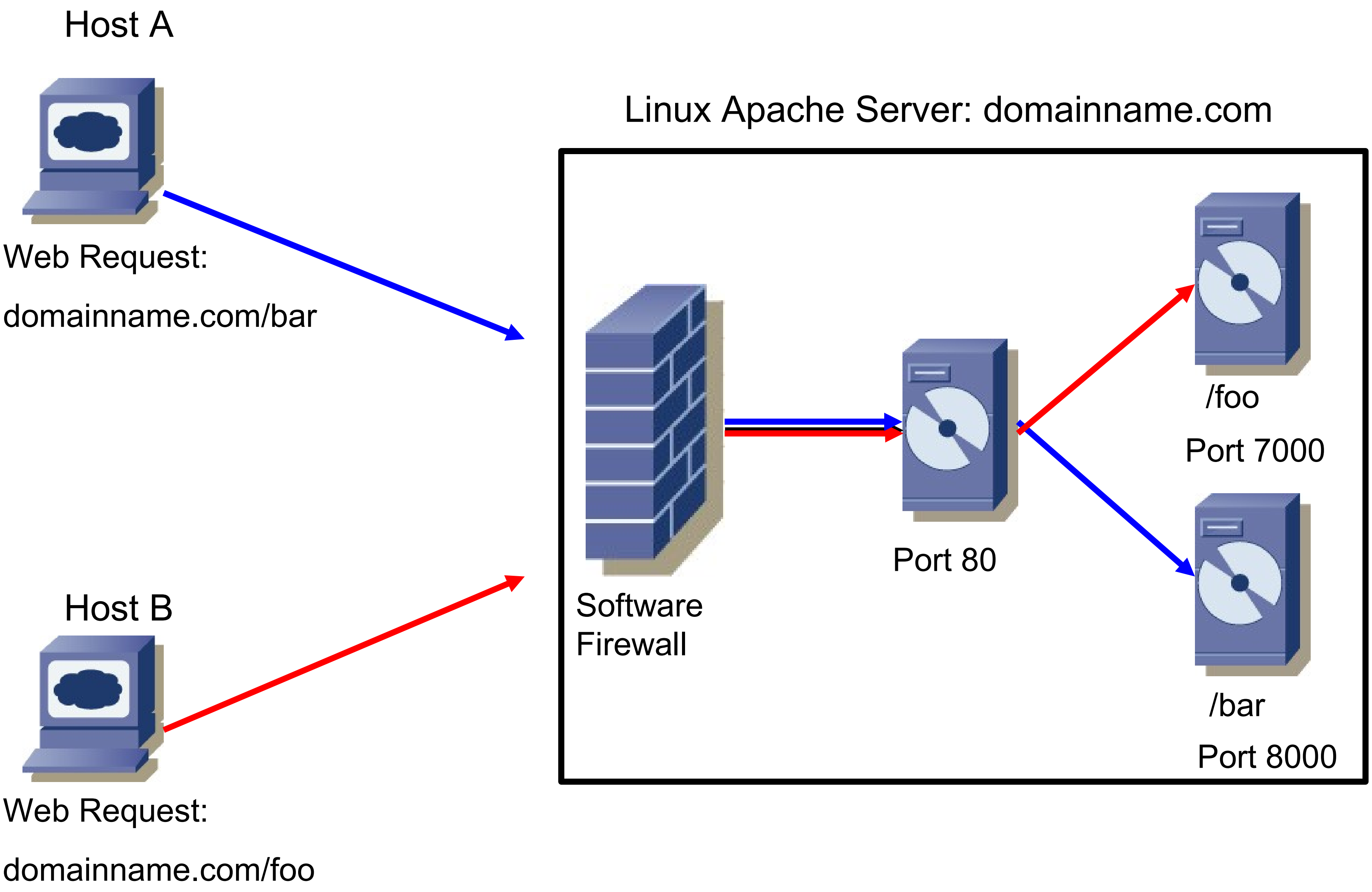
mod_proxy
enabled as
ReverseProxy

/foo

Port 7000

/bar

Port 8000



Host A

Web Request:

domainname.com/bar

Host B

Web Request:

domainname.com/foo

Linux Apache Server: domainname.com

Software Firewall

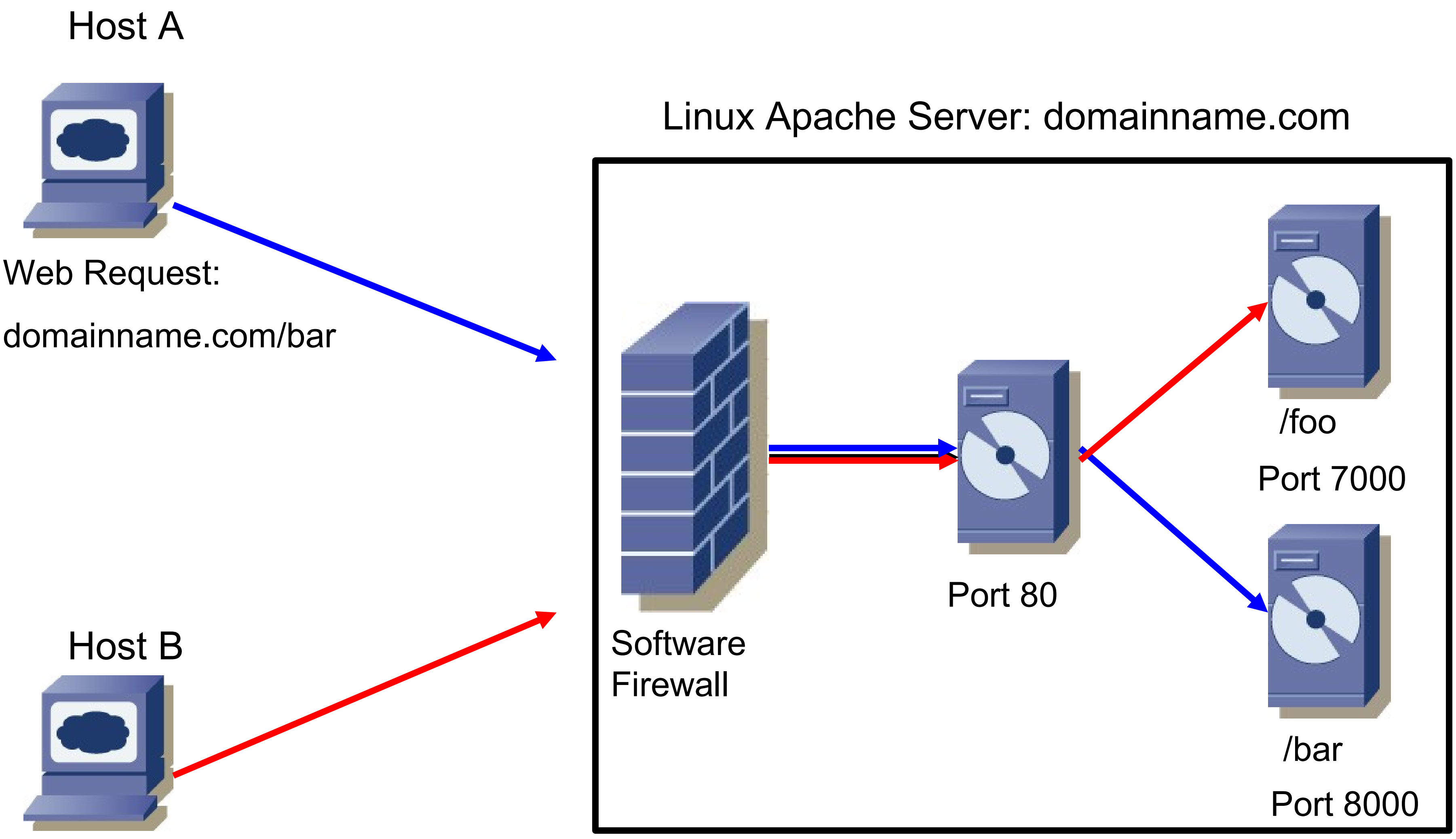
Port 80

/foo

Port 7000

/bar

Port 8000



Host A

Web Request:

domainname.com/bar

Linux Apache Server: domainname.com

Software Firewall

Port 80

/foo

Port 7000

/bar

Port 8000

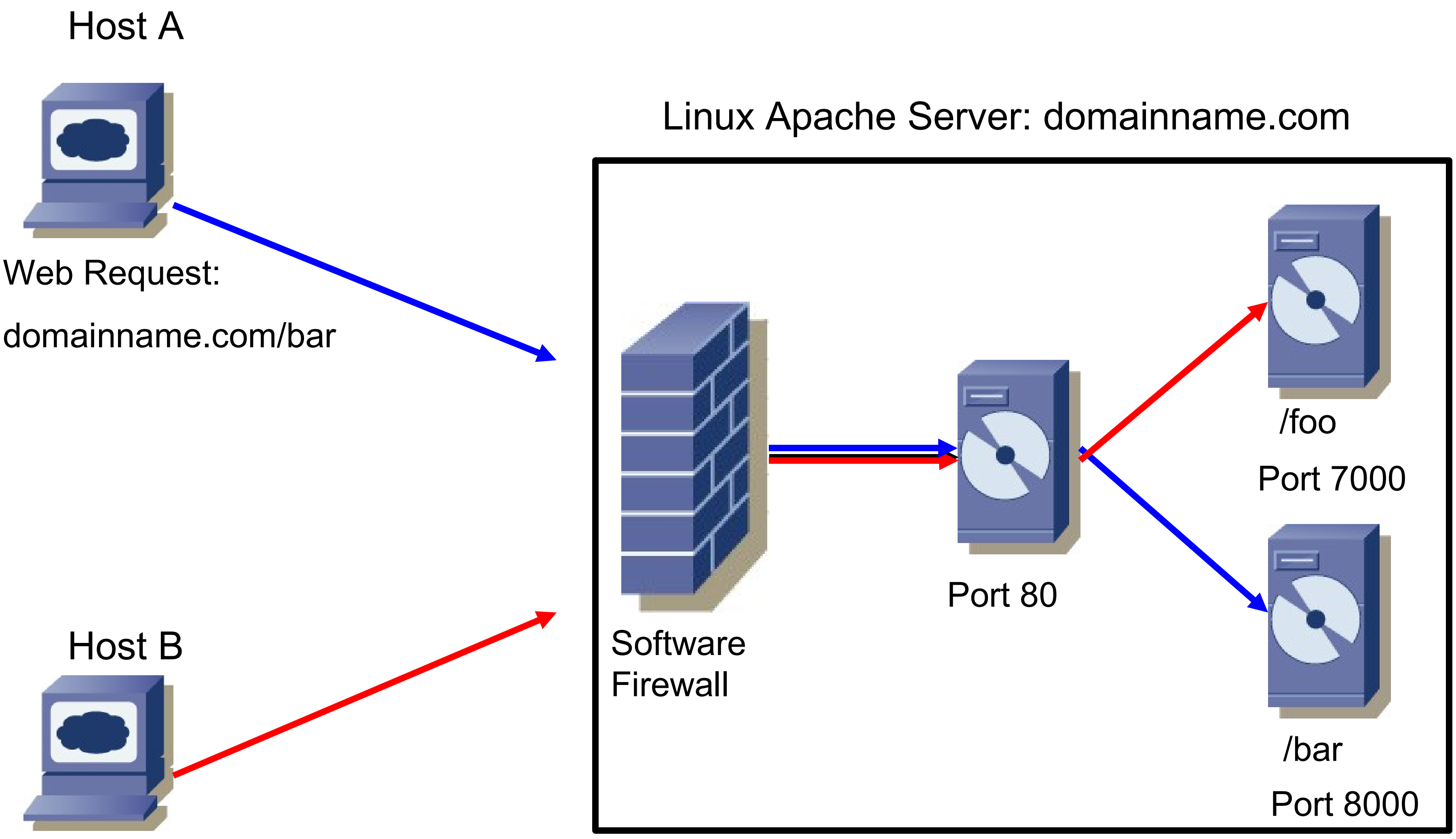
Host B

Web Request:

domainname.com/foo

Host A directed to /bar

Host B directed to /foo



Host A

Web Request:

domainname.com/bar

Linux Apache Server: domainname.com

Software Firewall

Port 80

/foo

Port 7000

/bar

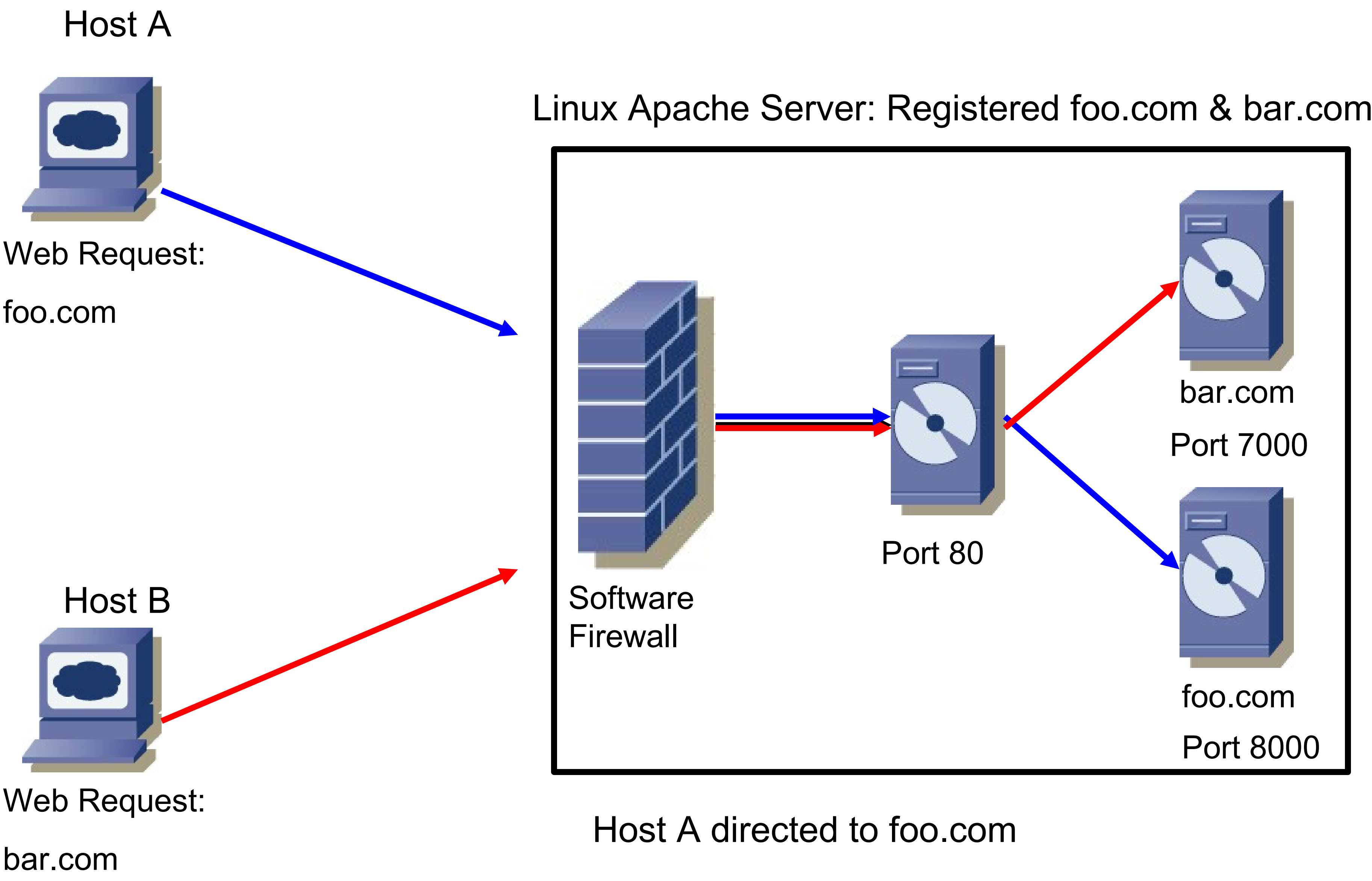
Port 8000

Host B

Web Request:

domainname.com/foo

Two domain names could be used for the apache server and directed towards the correct web server with mod_proxy.



Host A

Web Request:

foo.com

Host B

Web Request:

bar.com

Linux Apache Server: Registered foo.com & bar.com

Software Firewall

Port 80

bar.com

Port 7000

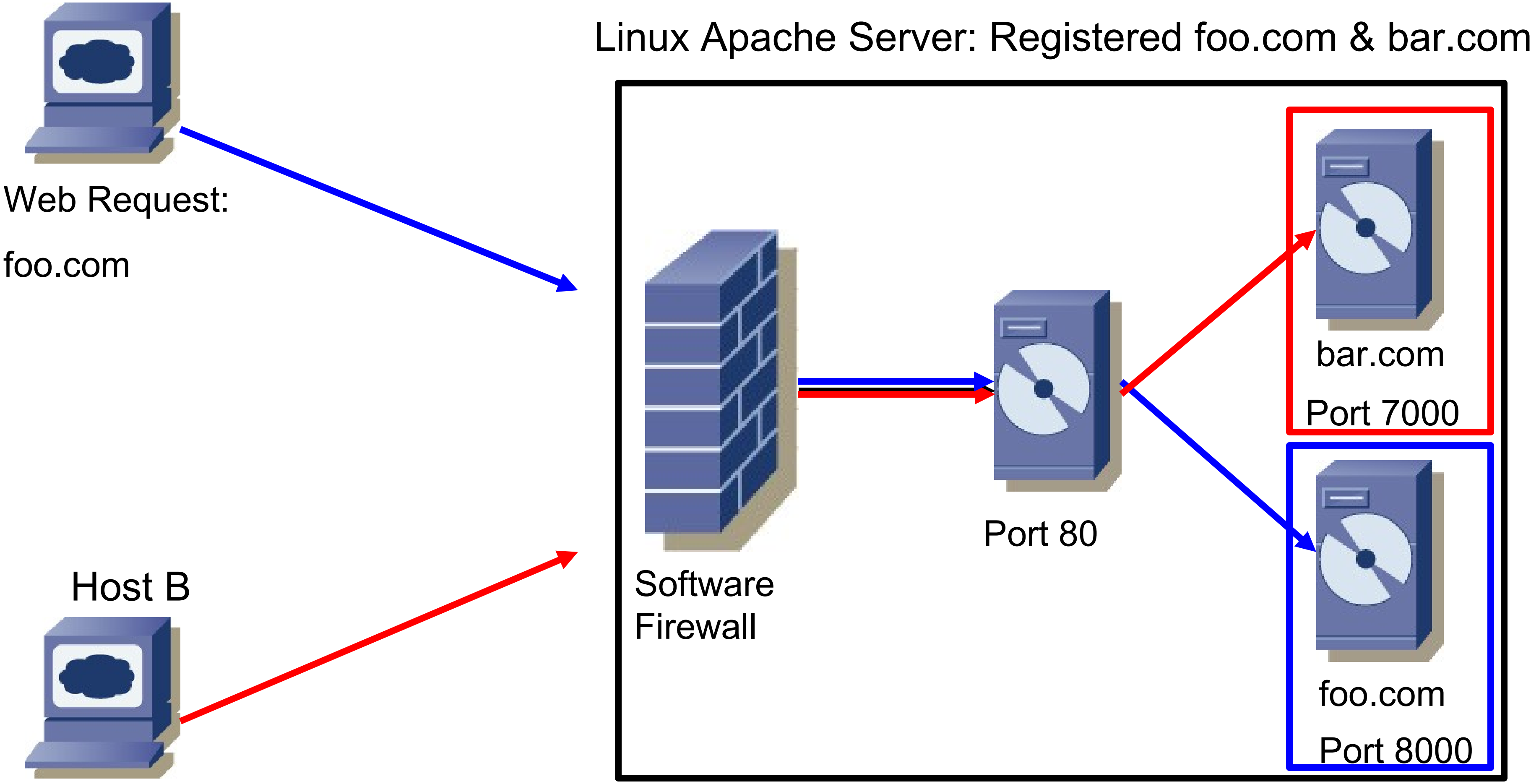
foo.com

Port 8000

Host A directed to foo.com

Host B directed to bar.com

Host A



Web Request:

foo.com

Linux Apache Server: Registered foo.com & bar.com

Software Firewall

Port 80

bar.com

Port 7000

foo.com

Port 8000

Host B

Web Request:

bar.com

Now lets think about both of the Apache servers in chroot jails.

bar.com
Port 7000

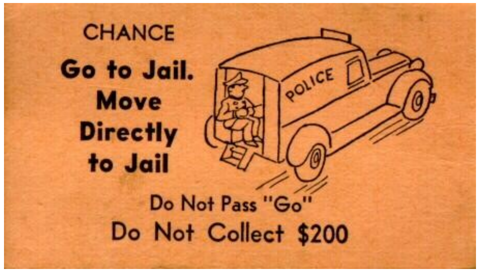
If malicious code is put into bar.com, it will be contained and foo.com will not be affected.

foo.com

Port 8000

The same goes for foo.com. Everything in it is contained.

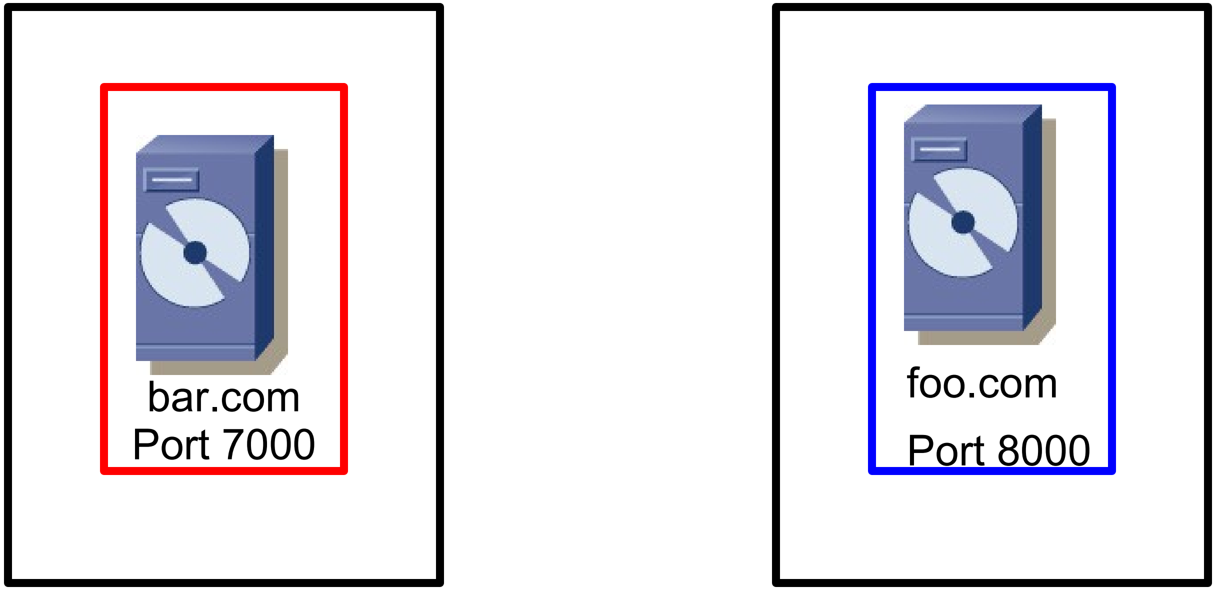
Also, if one of the servers is compromised from outside, they cannot get access to the other server or the system files, therefore, they basically pulled the wrong chance card…



CHANCE

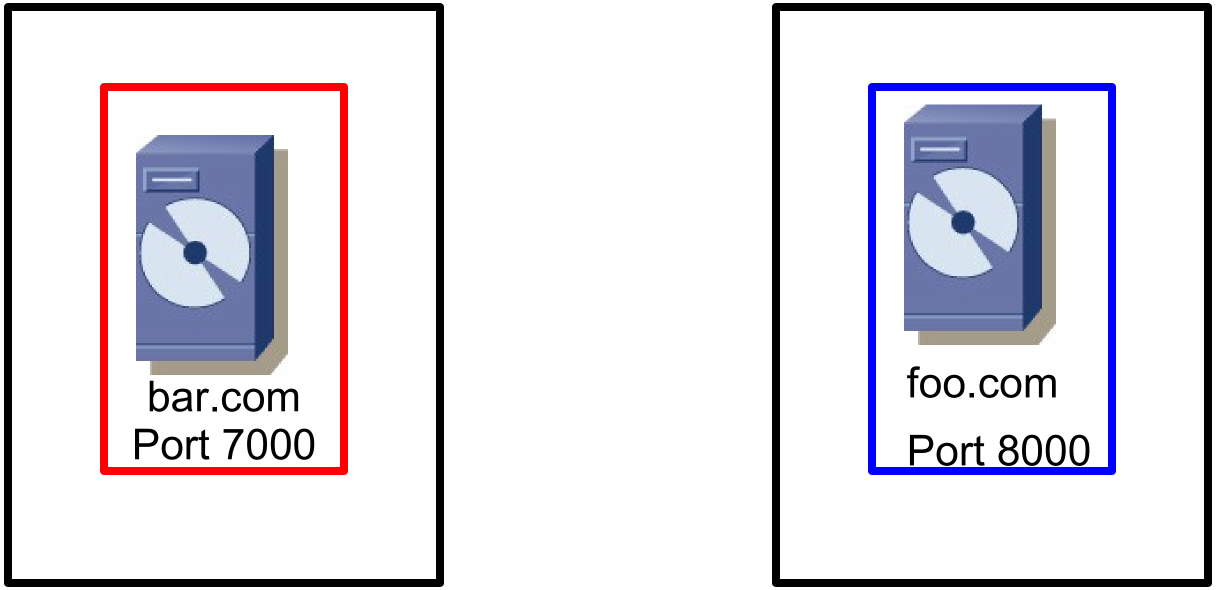
Go to Jail.
Move
Directly
to Jail

Do Not Pass "Go"
Do Not Collect $200



Lets go a little further and put both servers in an Encrypted File System, or EncFS, with FUSE (Filesystem in User Space)

With EncFS and FUSE, you are able to see the original file sizes, but that is all. Everything else, including the filenames, are encrypted.

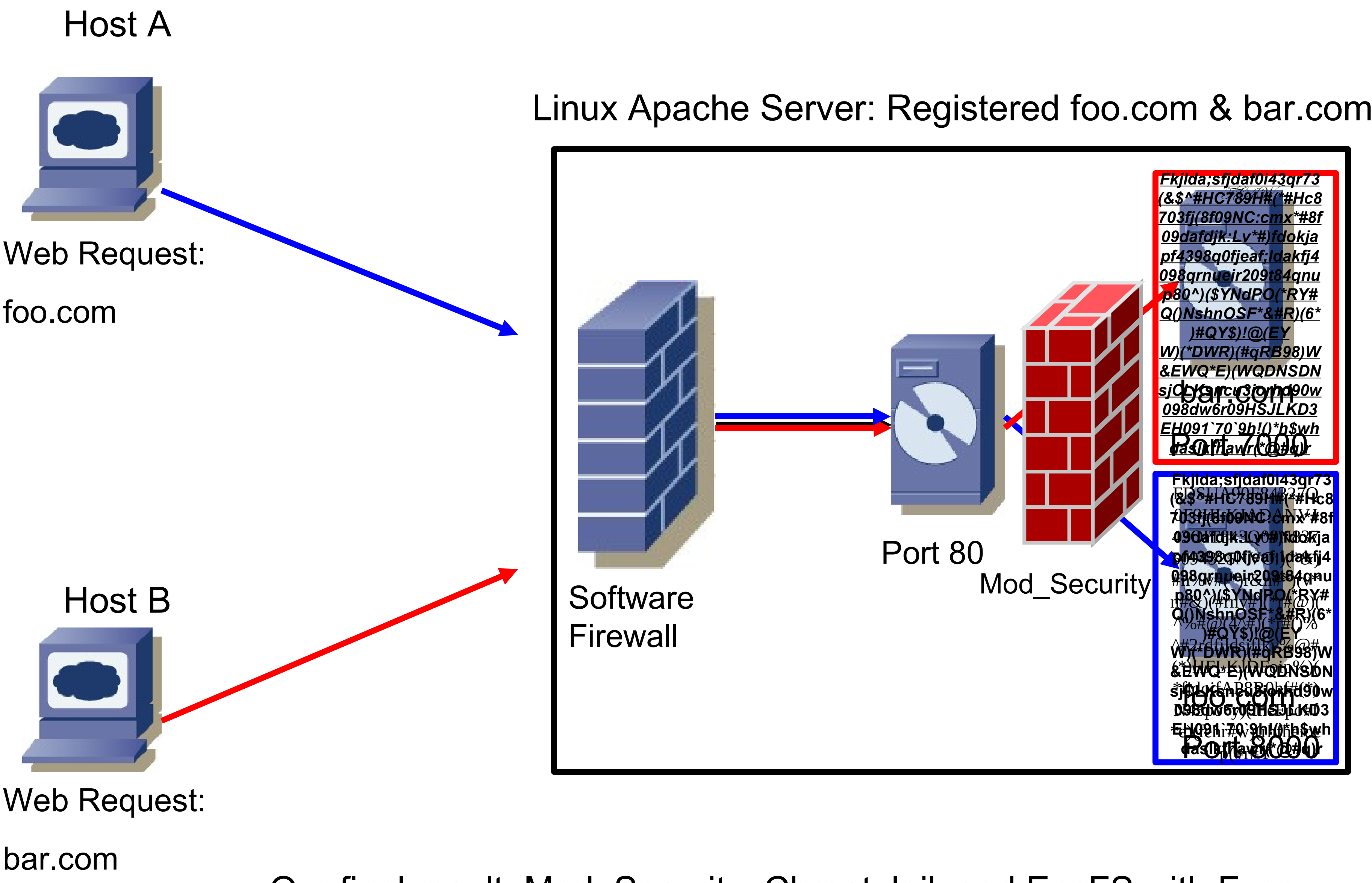


bar.com
Port 7000

foo.com
Port 8000

EncFS and Fuse have the ability decrypt one or both of the servers on-the-fly if you have the correct password.

Linux Apache Server: Registered foo.com & bar.com



Our final result: Mod_Security, Chroot Jail, and EncFS with Fuse.