[Optional] File Transfer and Web Caching

COSC301 Laboratory Manual

Old Lab

This lab hasn't been run recently, so no guarantees are made regarding the commands and their output.

In this lab, we shall be covering the File Transfer Protocol (FTP) briefly, before going on to the main part of the lab, which is Web Caching.

You will need to make some modifications to DNS. Add an alias for `ftp` that points to Server1 for both IPv4 and IPv6. There is no well-known hostname used for caches, although “cache” and “proxy” would be suitable choices, so add an alias for “cache” to point to Server1 as well; this will help prevent having to touch a lot of clients if we later decide to move the proxy cache.

Screenshot

Remember that names in DNS are generally always lowercase, although DNS is case-insensitive. Reload the DNS zones and test. Take a screenshot of your updated zone files and of your testing.

1. File Transfer Protocol (FTP)

Providing FTP services is common, but don’t provide anonymous FTP without being aware of the issues first. As we’ll see later in the course, it may be simple, but it’s not something to be taken lightly.

Depending on the particular FTP server implementation, default access policies will vary: some might allow only anonymous and deny users by default; some might allow users but deny anonymous by default… and I’m pretty sure the software we’re using has switched from one to the other in Ubuntu 10.04 LTS. Beware that FTP is a clear text protocol. There are cryptographic extensions to the FTP protocol (such as ftps)[1], but they are not widespread.

Under Debian-based systems vsftpd (Very Secure FTP Daemon) is often used. Install the package now, on Server1. Verify that the daemon is running, and listening on the FTP port.

Finding packages using apt-cache search

To find packages, you can use **apt-cache search `string`**. For example to search for the package that contains vsftpd, you might search use **apt-cache search vsftp**. In this case, you will find that one of the output lines says `vsftpd - The Very Secure FTP Daemon`.

Edit (as root) the vsFTPd configuration file `/etc/vsftpd.conf`. Make the following policy changes; you’ll have to figure out for yourself which options to change.

[1] Note: sftp is in no way similar to FTP, except as a user interface.

- Allow both users and guests (anonymous) to log in.
- Allow users (not guests) to write (upload) files etc. By default, the ability to write to anything is disabled.
- Set the banner text to “This is the localdomain FTP service.”
- Logs of what has been transferred must be enabled.
- Look at the manual page for vsftpd.conf(5), and find out more about the `dirmessage_enable` option.
- Find out where the anonymous files are to be found on the server. Have a look in `/usr/share/doc/vsftpd/README.Debian`
- When you have made the modifications and answered the questions, restart the FTP service.
- Create the file `/srv/ftp/.message` (note the dot) and put it in some text such as the following.

```
Currently there is nothing here, but if there were, you would
probably find a guide to the layout of the FTP server.
```

  While you are there, create a small file; the name doesn’t matter, you’ll use it for testing the retrieval and display of files.

```
$ sudo sh -c 'echo hello > /srv/ftp/test'
```

Restart vsFTPd, and try connecting to the FTP server from Client1; we’ll first try logging in as the anonymous (guest) user.

```
$ ftp ftp.localdomain
Connected to ftp.localdomain.
220 Welcome to the localdomain FTP service
Name (ftp.localdomain:mal): anonymous
331 Please specify the password.
Password: Your email address
230-Currently there is nothing here, but if there
230-were, you would probably find a guide to the layout.
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--    1 0        0             413 May 07 02:02 test
226 Directory send OK.
ftp> get test
local: test remote: test
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for test (6 bytes).
226 File send OK.
413 bytes received in 0.03 secs (12.4 kB/s)
ftp> bye
221 Goodbye.
```

In the above transcript, I was greeted by the FTP service, and I logged in anonomously using my email address as the password. This is just so the administrator can see who has been using the server. Some FTP administrators ban clients that use known-default email addresses, and **vsftpd** can easily support this policy. After logging in, I obtained a directory

listing and retrieved a file. I then disconnected. This is fairly basic stuff, but we haven't tested that local users can log in yet.

```
Make files on client and server for testing
client1$ hostname > ~/file_to_upload.txt
server1$ hostname > ~/file_to_download.txt

client1$ ftp ftp.localdomain
Connected to ftp.localdomain.
220 This is the localdomain FTP service
Name (ftp.localdomain:mal): mal
331 Please specify the password.
Password: mal's password
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--    1 1000     1000            7 May 07 02:33 file_to_download.txt
226 Directory send OK.
ftp> get file_to_download.txt
local: file_to_download.txt remote: file_to_download.txt
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for file_to_download.txt (7 bytes).
226 File send OK.
7 bytes received in 0.02 secs (0.3 kB/s)
ftp> put file_to_upload.txt
local: file_to_upload.txt remote: file_to_upload.txt
200 PORT command successful. Consider using PASV.
150 Ok to send data.
226 File receive OK.
8 bytes sent in 0.00 secs (12.5 kB/s)
ftp> bye
221 Goodbye.
```

There are two modes that FTP uses to transfer files. The first is `binary`, the other `ascii`[2]. When it is set to `ascii`, it will translate line-endings sequences, such as `\r\n` to `\n` when transferring from a Windows machine to a Unix system. Binary mode makes no such transformation.

Two other commands of note are `active` and `passive`. Passive (PASV) mode is used when you are behind a firewall, and the incoming data connection from the internet is disallowed. So instead of the server connecting to the client to establish the data connection (the PORT protocol command), the client connects to the server.

1.1. Assessment

This lab is optional, but these questions will help you to test your understanding.

1. Describe the default authentication policy that was supplied in the Ubuntu 10.04 LTS version of vsFTPd. Why might a policy that allows only anonymous users (denying authentication of users by default) be preferable from a security point-of-view?

2. You must have a screenshot showing that you have been able to log in as a normal user and download/upload files.

3. Take a screenshot of **wireshark** following the TCP ftp-control stream (or **tcpdump -A ...**) showing at least the authentication process.

[2]There is also ebcdic, which you would only be likely to see on IBM mainframes.

1.2. FTP Resources

RFC 2577: FTP Security Considerations
Worth reading to find out how FTP can be abused and what you can do to mitigate it.

2. Web Caching with Squid

In this section, we shall look at a tool that can both save bandwidth, account for bandwidth, provide some means of access control to the web and FTP, and if that wasn't enough, it can make the web go faster.

2.1. Transparent Proxies

This section on transparent proxies does not constitute lab work, but will be of interest to you to get a better picture of how proxy caches are commonly deployed.

Most caches we use are configured by an administrator, or a user, or autodetected (we'll see how that works later). It is common practice for ISPs and other networks to transparently cache objects. This is more about saving the ISP bandwidth (money), which, one would hope, should make the service more competitive[3]. Since we haven't played with firewalling yet, we won't be touching on transparent proxying in this lab.

One of the things ISPs are starting to do is cache Peer-to-Peer traffic, an early example of which is shown in the paper Deconstructing the Kazaa Network [http://portal.acm.org/citation.cfm?coll=GUIDE&dl=GUIDE&id=837393]. There can be some old legislation in various countries that make for interesting legal interactions but these are being addressed. For example, New Zealand explicitly made automated, temporary caching of material legal under the Copyright (New Technologies) Amendment passed in 2008. Details can be found under Section 93E of the Copyright Act [http://legislation.govt.nz/act/public/1994/0143/latest/whole.html], so you can expect this to become increasingly common practice as a mechanism for ISPs to save on traffic costs and provide enhanced service levels.

Transparent proxies work by redirection. Web requests are redirected to the proxy. If the proxy can satisfy that request, it sends it back to the requesting client, with a source IP of the webserver it would have otherwise been coming from. If the request cannot be fulfilled, the proxy gets it on behalf of the client, returning and caching the object as it returns.

The destination server will see the connection request coming from the proxy, so it will consider the proxy as the client. This artefact can be used in a number of ways. As an example of a good purpose, if you're at home, but need to get an academic paper from a journal, for example, but in order to do so you need to be seen as coming from your University, you could use the University's proxy. This is almost identical to the situation with NAT, whereby a number of different machines are seen as one IP address (and presumably, therefore, as one client), so if the server blocks one of those clients, it blocks them all. Proxies can also be used to hide the original client's identity, such as is done when using the Tor proxy system.

You can attempt to find out what proxies are being used (whether transparent or not,) by inspecting the HTTP headers returned from the server. You can use **curl -I URI** and look for any `Via` or `X-Cache` headers[4], as these are typically, though not always, added by any intermediaries. Here is an example, using **curl**, which you may need to install. The **-I** option shows only the headers.

[3] Or allow them to take more of our money.
[4] The `Via` header is documented in RFC 2616 [http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.45]

```
$ curl -I http://www.ubuntu.com
HTTP/1.0 200 OK
Date: Mon, 10 Jan 2011 23:43:40 GMT
Server: Apache/2.2.8 (Ubuntu) mod_python/3.3.1 Python/2.5.2 PHP/5.2.4-2ubuntu5.10 …
X-Powered-By: PHP/5.2.4-2ubuntu5.10
Vary: Accept-Encoding
Content-Type: text/html; charset=utf-8
Age: 1
Content-Length: 33890
Connection: close

HTTP/1.0 200 OK
Date: Mon, 07 May 2007 05:44:30 GMT
Server: Apache/2.2.3 (Ubuntu) …
…
Via: 1.1 privet.canonical.com:80 (squid/2.7.STABLE7)  Note: “reverse proxy”
…
X-Cache: HIT from privet.canonical.com
X-Cache-Lookup: HIT from privet.canonical.com:80      Note
…
Connection: close
```

Here we see that even though I have not configured **curl** to use a proxy my connection is being passed through a proxy anyway (by the network, hence the “transparent”). In this particular case, the proxy shown is one belonging to the target site: this is a “reverse proxy”, which is put in front of dynamic web-servers to reduce the amount of work that the web-server scripts need to. We can see that “privet” was able to satisfy this request from its cache, and thus did not have to pass it through to the back-end.

Another common way of encountering transparent proxies is closer to the client: for example you could imagine that all TCP/80 traffic going through Server1 gets automatically forwarded to a transparent proxy. This means clients don’t need to configure a cache, although it does have some limitations (such as no user-authentication).

2.2. [Reference Only] Proxy Autodetection

This section on proxy autodetection is for your reference only, because most people would have seen an option for it in their browser, but won’t know much about it. You are not expected to do anything on your machines for this section.

We can autoconfigure web browsers to use a particular proxy using WPAD (Web Proxy Auto-Detection), although support for it in some versions of some browsers can be spotty. WPAD was never a formal standard, its progress seems to never have made it past a draft, but its still used by most browsers, although only to minimal conformance.

Important

WPAD has a number of problems stemming from a poor design. This can be problematic for individual pieces of software that implements WPAD. I suggest having a look at Beau Butler’s talk on Web Proxy Autodetection and your network [http://2008.nznog.org/conferencepapers.html#wpad]. You should be aware of these issues, even if your don’t want to use WPAD.

The idea of WPAD is to find a Configuration URL (CURL) that will result in a HTTP request for a Configuration File (CFILE). The CFILE is the *contents* of the file that is retrieved via the CURL. If the CFILE is invalid, it moves on to the next CURL. If it exhausts the search, it decides to use a direct connection (ie. it doesn’t use a proxy).

Tries DHCP, WINS and DNS

WPAD tries to find a suitable CURL using a variety of means, including DHCP, WINS (used in Windows networking), and DNS lookups for A/CNAME. There are other mechanisms but they are not worth mentioning. The protocols mentioned are commonly used. DHCP should be tried first.

WPAD Configuration via DHCP

The DHCP method involves sending a DHCP INFORM message to a DHCP server (preferably using unicast, if the agent knows the IP address of your DHCP server). This does not involve getting a new IP address. The query is for the DHCP option 252, and should return the CURL, such as `http://server.domain/proxy.pac`.

To configure this behaviour in your DHCP server, you would need the following code in your `dhcpd.conf`. You don't need to do this now, its just a useful reference. It should be noted that although the WPAD Draft standard would prefer DHCP, you're more likely to get more success with the DNS well-known alias method.

```
…
option wpad code 252 = string;

subnet 192.168.1.0 netmask 255.255.255.0 {
    option wpad "http://webserver.example.com/wpad.dat";   CURL to advertise
}
```

DNS Query for Well-Known Alias

The other method that would be tried after DHCP is usually a DNS query[5]. This involves determining the client's fully qualified domain name, and removing the host component. So if the client was bob.engineering.bigco.com, the Target Domain (`target_domain`) would be engineering.bigco.com. It would then do various DNS queries using that `target_domain` as the DNS name to lookup. The most common request would be an A or CNAME lookup for the well-known name, wpad.`target_domain`.

If the search using the `target_domain` fails, it removes the leftmost component, and tries again. It will stop searching when it finds a valid CFILE, or the `target_domain` is one of the official Top Level Domains (TLD) such as com, org, net etc. It should not stop when the `target_domain` is not official, such as localdomain. It would try the following CURLs in this order.

1. http://wpad.engineering.bigco.com

2. http://wpad.bigco.com

3. http://wpad.com, but this would not be looked up, because the `target_domain` is a TLD.

CFILE format

The file that is returned with a particular CURL, should look something like the following. Basically, the file needs to contain the Javascript definition of a single function FindProxyForURL. More information can be found in the Squid documentation, including information on functions you can use to make your autoconfiguration file smarter. Here are

[5]The WPAD specification also uses the Service Location Protocol (SLP), but it isn't widely used.

some examples of the function you might care to define; you could use any *one* of them. You can make a function that is rather more complicated than shown here, that does things like look at the URL or the requesting host to decide which proxy (if any) should be used.

```
EITHER This version would tell the browser not to use a proxy
function FindProxyForURL(url, host) {
    return DIRECT;
}

OR This version would tell the browser to use a cache proxy
function FindProxyForURL(url, host) {
    return "PROXY cache.domain.example:3128";
}

OR The same, but with failover to using no proxy.
function FindProxyForURL(url, host) {
    return "PROXY cache.domain.example:3128; DIRECT";
}

OR Finally, if you have to use SOCKS style proxy
function FindProxyForURL(url, host) {
    return "SOCKS cache.domain.example:3128";
}
```

Client Configuration

To configure a real client, you go into the options, and under Proxies select the option that says samething like “Autodetect Proxy Settings”. You could do this on your Mac using Firefox, as you don’t have permission to change the system default proxy settings which Safari uses. Don’t enter a CURL, although it can be useful, as that defeats the purpose of resource discovery and makes it harder for laptops to stay mobile.

In Firefox, the proxy settings can be found under Edit → Settings Advanced tab, Network sub-tab, Settings… button. However, in environments where it is possible, setting the proxy settings for all programs at once is generally preferable.

2.3. Basic Squid Configuration

In this section, you will configure the web proxy with basic settings, and allow everyone to use the proxy. Unless otherwise specified do this section on Server1.

Install the squid package. When it starts for the first time it will create some indexes (on other systems, you may need to initialise these yourself).

The first thing to do is check the configuration file `/etc/squid3/squid.conf`. Make the following changes.

- Ensure that ICP is disabled (`icp_port` should be 0). This disables the Internet Cache Protocol, which is sometimes used when you have a collection of caches. It isn’t useful to us, so we’ll disable it.
- Squid can be monitored using a protocol called SNMP, which we shall see in a later lab. For now, ensure that `snmp_port` is off by setting it to 0, as we have no need for it.
- If you were to create a cache hierarchy, you would insert a `cache_peer` line for every peer you wish to feed off. We don’t need any such lines for our configuration.
- Depending on the amount of disk-space available on the proxy server, you may want to alter the `maximum_object_size` option, although for our purposes, it should be fine. Also, you

should change the `cache_dir` line to set an appropriate cache size. The default is 100MB, which is plenty for our virtual machines, but may be too little for some sites, especially considering disk space is very much cheaper compared to Internet usage.

- Set the `visible_hostname` to `cache.localdomain`.
- Notice the `cache_dir` entry. Squid will keep its cache in `/var/spool/squid3/`[6]. This is something that would be good to keep an eye on to get a feel for how much space your network needs, and to make this a large area when partitioning the disks of a server designed as a proxy.
- The file specified by the `access_log` option is an interesting file to keep an eye on, as this is where accounting information can be sourced. There are many pieces of software available for Squid log analysis, the most popular would probably be Calamaris, but we'll not go into that today. You would, however, need to add some rules for **logrotate**, in order to manage the logs.
- It is good Netiquette to supply a more useful anonymous FTP username than Squid's default. So change `ftp_user` to `proxyuser@localdomain`. Note that in our private environment, this is less useful, but can be more useful on the internet. Some FTP servers will not let you in if you provide a password that is known to be a default password. The email address that is used should normally be aliased to a real person and monitored. Ensure that it also gets filtered for SPAM, just like any other common mail-box name (eg. hostmaster@…, postmaster@… etc.) as these accounts will likely get a higher-than-average amount of spam.
- The email address of the cache administrator will appear in the error messages, and is configured by the `cache_mgr` directive.
- Find the ACL section, and under the line that defines the ACL called `all`, add a line called `clients`, that specifies the IP network address and subnet mask. You can specify multiple networks on the same line. This creates an Access Control List, but does not apply it to anything.

```
…
acl all src 0.0.0.0/0.0.0.0 Matches everything
acl clients src 192.168.1.0/24
…
```

- In the `http_access` area, apply the newly created ACL to allow the clients to access HTTP, by adding in the following line. Squid's default is to otherwise deny to all, so you have to configure it to allow the clients before you can use it.

```
…
#
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS
http_access allow clients
…
```

That's all we need to edit for now. Time to restart it:

```
# service squid3 stop
Control returns immediately, but Squid will hang around for 30 seconds more…
# tail -f /var/log/squid/cache.log
# service squid3 start
```

[6] Other systems may keep it under `/var/lib/squid/cache/`

Tip

For simple little reconfigurations, you can use **sudo squid3 -k reconfigure**, which works immediately.

Check your console messages (syslog) to make sure its not complaining. Note that you will get more detailed messages as to what the cache is doing in the log file `/var/log/squid3/cache.log`. Check to see that its listening for connections on TCP/3128.

```
Edited for clarity
# lsof -i | grep squid
COMMAND  USER   TYPE  NODE  NAME
squid3    squid  IPv4  UDP   *:3072            For outgoing DNS
squid3    squid  IPv4  TCP   *:3128  (LISTEN)
```

Now that squid is running, let's test it from the client. Setup Firefox on Client1 to access the proxy, using `cache.localdomain` as the hostname, 3128 as the port, and adding `.localdomain` to the list of domains not to use a proxy for. Visit any website, and look at Squid server logs to ensure that it is accessing the server and working correctly. Try HTTP and HTTPS URIs, and note that for HTTPS services you get a CONNECT request sent to the proxy, not a GET etc.

503 Service Unavailable

If, during the following tests, you get a `503 Service Unavailable` response, it means that the connection from the proxy to the target server, as specified in the URL, is failing. This may be because the web-server software is not running or installed on the web-server, or because you have requested a URL that contains a server name that is not resolvable via DNS, perhaps because of a miss-spelling.

2.4. [Optional] User-based Access Control

So far, we've configured minimal access control, allowing only our client IP range access. Squid can do access control using a large number of criteria. Some of the more interesting are as follows.

- Source/Destination IP address
- Source/Destination Domain
- Regular Expression match of requested domain
- Words in the requested URL
- Words in the source or destination domain
- Current day/time
- Protocol (FTP, HTTP, SSL)
- Username (according to the Ident protocol)
- Username/Password pair

- Method (GET, POST, CONNECT, …)
- There are more, see the Squid documentation

We're going to have a closer look at the Username/Password pair, using Basic authentication. Since this could be a large list, we'll put it into a file. Simple Username/Password lookups can be done using a variety of methods. One of the most common is the NCSA lookup, which use a Unix-format passwd file; not neccessarily `/etc/passwd`, but it looks rather similar, though with fewer columns. To configure Squid to use this method of authentication, you need to find the following directives, uncomment and change them to match the following; you should find them all in one block.

```
…
Uncomment the following
auth_param basic program /usr/lib/squid3/basic_ncsa_auth /etc/squid/passwd
auth_param basic children 5 startup=5 idle=1
auth_param basic realm Squid proxy-caching web server
auth_param basic credentialsttl 2 hours
…
```

Create the `passwd` file mentioned in the configuration above, with mode 600, owned by the `proxy` user. We don't need to create local users on the system for the users that will be authenticating to the proxy, we just need to put the usernames and hashed passwords in `/etc/squid/passwd`. Here is how we create and set the permissions appropriately:

```
# touch /etc/squid/passwd
# chmod 600 /etc/squid/passwd
# chown proxy /etc/squid/passwd
```

Squid can use other Username/Password mechanisms, such as local Unix users (`getpwnam`), LDAP, NIS, NTLM (Windows Domain authentication), and PAM; lots of acronyms, but don't worry about that, they just represent different ways that authentication can happen in different environments.

Create an entry for yourself in the file `/etc/squid/passwd`, using the method shown below to create the password hash.

```
htpasswd is actually part of Apache
# htpasswd -d /etc/squid/passwd mal
New password: Not echoed
Re-type new password: Not echoed
Adding password for user mal
```

You may need to install the `apache2-utils` to get the **htpasswd** command.

Edit `squid.conf` so it denies people with a bad username/password pair.

```
…
#Recommended minimum configuration:
acl all src 0.0.0.0/0.0.0.0
acl clients src 192.168.1.0/24
acl password proxy_auth REQUIRED
…
# search for the line below to find where to put the following…
# INSERT YOUR OWN RULE(S) HERE TO ALLOW ACCESS FROM YOUR CLIENTS
#
#http_access allow clients  comment out this line
http_access allow password
…
```

Notice that we commented out the ACL option `http_access` that allows in known clients. Reconfigure Squid:

```
# squid -k reconfigure
```

Checking the logs to check that it isn't complaining about anything.

2.5. Assessment

1. Give at least one description of how WPAD could be used as an attack vector. You may need to do some research.

2. **[Optional]** Show that you can get denied when you supply an incorrect password, and that you can also get access when you supply a correct username/password.

3. **[Optional]** What would have been the effect if we didn't comment out the ACL option `http_access` that allows access to known clients? Give a scenario where this may be useful.

2.6. Resources

Cacheability Query [http://www.ircache.net/cgi-bin/cacheability.py]
Check how cacheable your website is.

Deconstructing the Kazaa Network [http://people.cs.uchicago.edu/~matei/PAPERS/kazaa.pdf]
Look at how transparent proxies can be used for non-HTTP traffic.

WPAD Draft RFC (Expired) [http://www.wpad.com/draft-ietf-wrec-wpad-01.txt]
The full details on how web proxy autodetection was supposed to be performed (this in an expired RFC, so it has little weight).

Upside-Down-Ternet [http://www.ex-parrot.com/pete/upside-down-ternet.html]
A humourous example of how proxies can be used to modify content. A more practical example would be recompressing large images that are to be sent to mobile devices.