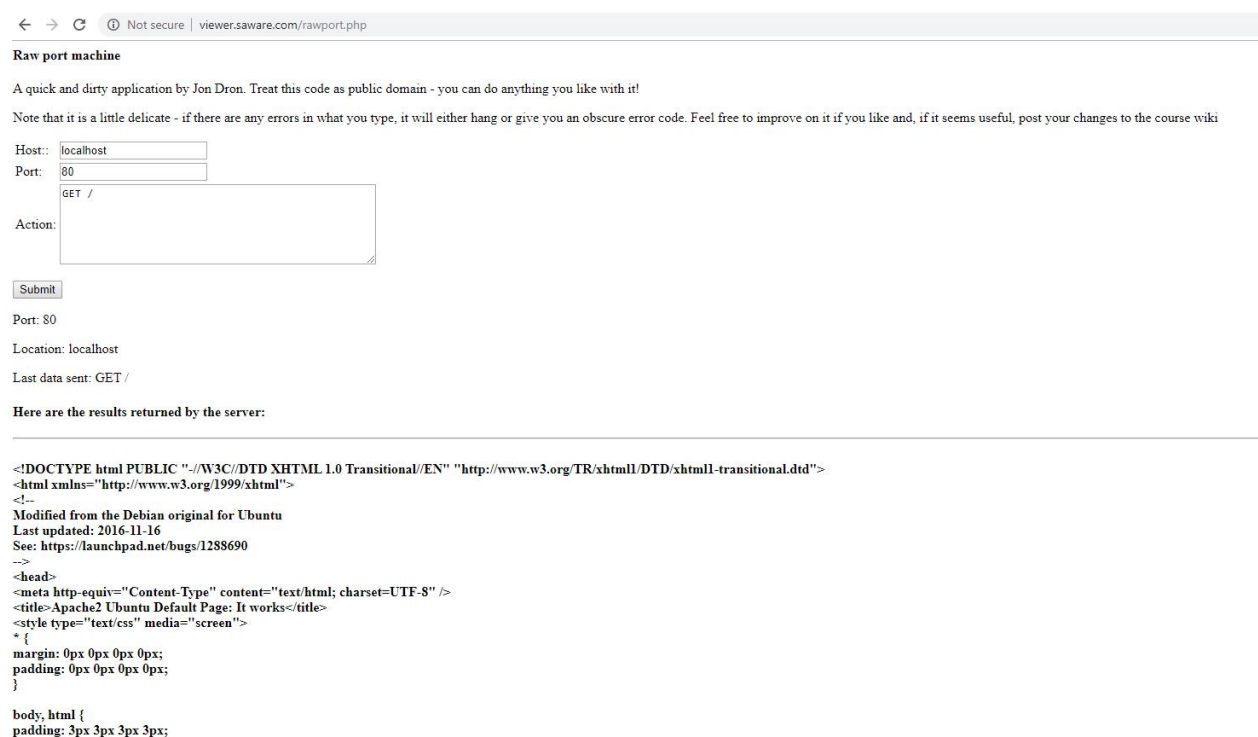


HTTP Lab

1. Using HTTP 0.9 – the simplest possible connection

After checking the contents of the php file and the resulting code, I think it's just to print out the contents of "index.html", which is the default file of my virtual host "viewer.saware.com." However, I have no clue to answer the questions at the first glance. What's the relation between these questions and the resulting code?



Raw port machine

A quick and dirty application by Jon Dron. Treat this code as public domain - you can do anything you like with it!

Note that it is a little delicate - if there are any errors in what you type, it will either hang or give you an obscure error code. Feel free to improve on it if you like and, if it seems useful, post your changes to the course wiki

Host:

Port:

Action:

Port: 80

Location: localhost

Last data sent: GET /

Here are the results returned by the server:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--
Modified from the Debian original for Ubuntu
Last updated: 2016-11-16
See: https://launchpad.net/bugs/1288690
-->
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Apache2 Ubuntu Default Page: It works</title>
<style type="text/css" media="screen">
* {
margin: 0px 0px 0px 0px;
padding: 0px 0px 0px 0px;
}
body, html {
padding: 3px 3px 3px 3px;
```

Q: If you were a web browser, what would you do next?

I think I would retrieve all resources from URLs provided in the extracted file and then present them to user.

Q: What problems might you face? Hint—pictures? Another hint—what if you wanted to get the page again?

I think you mean I need another connection to load those resources such as pictures. It may be better to get them altogether with the index file, so a new connection is not needed.

Q: Is there any way you could avoid having to get the whole file?

I think you mean cache. If the whole file was cached before, I only need to retrieve the contents which have been modified since last visit.

2. Using HTTP 1.1

Raw port machine

A quick and dirty application by Jon Dron. Treat this code as public domain - you can do anything you like with it!

Note that it is a little delicate - if there are any errors in what you type, it will either hang or give you an obscure error code. Feel free to improve on it if you like and, if it seems useful, post your changes to the course wiki

Host:

Port:

Action:

Port: 80

Location: localhost

Last data sent: GET / HTTP/1.1 Host: localhost Accept: */*

Here are the results returned by the server:

```
HTTP/1.1 200 OK
Date: Sun, 20 Oct 2019 22:04:29 GMT
Server: Apache/2.4.38 (Ubuntu)
Last-Modified: Sun, 29 Sep 2019 17:18:56 GMT
ETag: "2aa6-593b4506ad43d"
Accept-Ranges: bytes
Content-Length: 10918
Vary: Accept-Encoding
Content-Type: text/html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--
Modified from the Debian original for Ubuntu
Last updated: 2016-11-16
See: https://launchpad.net/bugs/1288690
-->
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Apache2 Ubuntu Default Page: It works</title>
<style type="text/css" media="screen">
* {
```

Now I understand a little more about what you want me to learn.

Q: And press return twice (why not once, as before?).

As per the answer in the self-test quiz, HTTP 1.1 keeps the connection open to allow extra commands to be sent, each of which is separated by a single carriage return- a blank line on its own signals that the connection is complete. Here pressing return twice generates a blank line, so the server knows it's the time to close the connection.

Q: What differences do you notice?

The main differences is: before the file content is delivered, a few additional information is sent as response header. The information include HTTP version, status code, response time, web server version, last modification time, modification tag, range unit for partial requests, content length, encoding info, and content type. These information are useful to reduce the network payload and improve the response time.

3. Redirection and default documents

Host::	<input type="text" value="localhost"/>
Port:	<input type="text" value="443"/>
Action:	<input type="text" value="GET /member HTTP/1.1
Host: localhost
Accept: */*"/>

Port: 443

Location: localhost

Last data sent: GET /member HTTP/1.1 Host: localhost Accept: */*

Here are the results returned by the server:

```
HTTP/1.1 400 Bad Request
Date: Thu, 24 Oct 2019 23:54:43 GMT
Server: Apache/2.4.38 (Ubuntu)
Content-Length: 446
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
Reason: You're speaking plain HTTP to an SSL-enabled server port.<br />
Instead use the HTTPS scheme to access this URL, please.<br />
</p>
<hr>
<address>Apache/2.4.38 (Ubuntu) Server at member.saware.com Port 443</address>
</body></html>
```

Q: What is the server telling you now? What should you have typed?

The server is telling me that this is a bad request because a plain HTTP is sent to a SSL-enabled server port. I think this is not what you want me to test.

Then I created an empty directory called "user" at the document root of the unsecured site, and tried again.

Host:: localhost
Port: 80
Action: GET /user HTTP/1.1
Host: localhost
Accept: */*

Submit

Port: 80

Location: localhost

Last data sent: GET /user HTTP/1.1 Host: localhost Accept: */*

Here are the results returned by the server:

HTTP/1.1 301 Moved Permanently
Date: Fri, 25 Oct 2019 23:38:52 GMT
Server: Apache/2.4.38 (Ubuntu)
Location: http://localhost/user/
Content-Length: 305
Content-Type: text/html; charset=iso-8859-1

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="http://localhost/user/">here</a>.</p>
<hr>
<address>Apache/2.4.38 (Ubuntu) Server at localhost Port 80</address>
</body></html>
```

This time it's a 301 response "moved permanently".

What if I add a trailing slash?

Host: localhost
 Port: 80
 Action: GET /user/ HTTP/1.1
 Host: localhost
 Accept: */*

Submit

Port: 80

Location: localhost

Last data sent: GET /user/ HTTP/1.1 Host: localhost Accept: */*

Here are the results returned by the server:



```
HTTP/1.1 200 OK
Date: Fri, 25 Oct 2019 23:42:45 GMT
Server: Apache/2.4.38 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 734
Content-Type: text/html; charset=UTF-8

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final/EN">
<html>
<head>
<title>Index of /user</title>
</head>
<body>
<h1>Index of /user</h1>
<table>
<tr><th valign="top"><th><a href="?C=N;O=D">Name</a><th><a href="?C=M;O=A">Last modified</a><th><a href="?C=S;O=A">Size</a><th><a href="?C=D;O=A">Description</a><th></tr>
<tr><th colspan="5"><hr></th></tr>
<tr><td valign="top"><td><a href="/">Parent Directory</a><td><td align="right">-<td><td>&nbsp;</td></tr>
<tr><th colspan="5"><hr></th></tr>
</table>
<address>Apache/2.4.38 (Ubuntu) Server at localhost Port 80</address>
</body></html>
```

This time it's a 200 response "OK"

The explanation is: when a trailing slash is added, the request is interpreted as a directory; when it's not added, the request is interpreted as a file (Gunnars, 2019).

4. Security

As HTTP authentication is not secured, I only implemented it in the SSL-enabled host in TME 5. So for TME6, I created another "admin" directory with HTTP authentication at the unsecured host viewer.saware.com and changed the configuration of virtual host viewer.saware.com.

However, it's weird that the authentication works in the normal web page, but does not work in this rawport.php.

I guess that there's something wrong in the host name "localhost," so I tried below

Host:: localhost
Port: 80
Action: GET /admin/ HTTP/1.1
Host: viewer.saware.com
Accept: */*

Submit

Port: 80

Location: localhost

Last data sent: GET /admin/ HTTP/1.1 Host: viewer.saware.com Accept: */*

Here are the results returned by the server:

HTTP/1.1 401 Unauthorized
Date: Sat, 26 Oct 2019 00:29:31 GMT
Server: Apache/2.4.38 (Ubuntu)
WWW-Authenticate: Basic realm="Restricted Content"
Content-Length: 464
Content-Type: text/html; charset=iso-8859-1

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>401 Unauthorized</title>
</head><body>
<h1>Unauthorized</h1>
<p>This server could not verify that you
are authorized to access the document
requested. Either you supplied the wrong
credentials (e.g., bad password), or your
browser doesn't understand how to supply
the credentials required.</p>
<hr>
<address>Apache/2.4.38 (Ubuntu) Server at viewer.saware.com Port 80</address>
</body></html>
```

This time WWW-Authenticate is shown properly.

I think this is because localhost is not the host name specified in my server. As the server can't find a host configuration file, it goes to the default configuration file where no HTTP authentication is set. This acts like a back door to allow unauthorized user to access restricted content. Therefore, I also edited the default host configuration file. Now it runs as expected. A 401 response is returned.

Host:	<input type="text" value="localhost"/>
Port:	<input type="text" value="80"/>
Action:	<pre>GET /admin/ HTTP/1.1 Host: localhost Accept: */*</pre>

Port: 80

Location: localhost

Last data sent: GET /admin/ HTTP/1.1 Host: localhost Accept: */*

Here are the results returned by the server:

HTTP/1.1 401 Unauthorized
Date: Sat, 26 Oct 2019 22:41:35 GMT
Server: Apache/2.4.38 (Ubuntu)
WWW-Authenticate: Basic realm="How do you come here"
Content-Length: 456
Content-Type: text/html; charset=iso-8859-1

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>401 Unauthorized</title>
</head><body>
<h1>Unauthorized</h1>
<p>This server could not verify that you
are authorized to access the document
requested. Either you supplied the wrong
credentials (e.g., bad password), or your
browser doesn't understand how to supply
the credentials required.</p>
<hr>
<address>Apache/2.4.38 (Ubuntu) Server at localhost Port 80</address>
</body></html>
```

After the authorization code is entered, a 200 response is received as expected.

Host:: localhost
Port: 80
Action: GET /admin/ HTTP/1.1
Host: localhost
Accept: */*
Authorization: Basic c2FtbXk6aml1c2hpdGExMg==

Submit

Port: 80

Location: localhost

Last data sent: GET /admin/ HTTP/1.1 Host: localhost Accept: */* Authorization: Basic c2FtbXk6aml1c2hpdGExMg==

Here are the results returned by the server:

```
HTTP/1.1 200 OK
Date: Sun, 27 Oct 2019 00:10:49 GMT
Server: Apache/2.4.38 (Ubuntu)
Vary: Accept-Encoding
Content-Length: 736
Content-Type: text/html; charset=UTF-8

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<html>
<head>
<title>Index of /admin</title>
</head>
<body>
<h1>Index of /admin</h1>
<table>
<tr><th valign="top"></th><th><a href="?C=N;O=D">Name</a></th><th><a href="?C=M;C
<tr><th colspan="5"><hr></th></tr>
<tr><td valign="top"></td><td><a href="/">Parent Directory</a></td><td>&nbsp;</
<tr><th colspan="5"><hr></th></tr>
</table>
<address>Apache/2.4.38 (Ubuntu) Server at localhost Port 80</address>
</body></html>
```

Q: Why do you think that HTTP requires the username and password to be encoded?

The Base64 can be easily decoded, so its purpose is not for security. It is escaping the special character (Adkins, 2013).

5. Non-existent pages

Q: What does the server tell you? Have you seen anything like that before?

Apparently, a 404 response is received since this file does not exist. I didn't see this response in the previous practices in this task, but I saw it from other web server. This message just tells user that the requested resource is not found on the server.

Host:

Port:

Action:

Port: 80

Location: localhost

Last data sent: GET /nosuchfileexists.html HTTP/1.1 Host: localhost Accept: */*

Here are the results returned by the server:

```
HTTP/1.1 404 Not Found
Date: Sun, 27 Oct 2019 00:18:28 GMT
Server: Apache/2.4.38 (Ubuntu)
Content-Length: 271
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL was not found on this server.</p>
<hr>
<address>Apache/2.4.38 (Ubuntu) Server at localhost Port 80</address>
</body></html>
```

Raw port machine, ©Jon Dron, 2008

6. Getting pictures

Q: What do you think will happen if you change Accept: */* into Accept: text/* ?

Since the “accept” directive is set by the browser, it depends on how the server algorithm use this information for provide contents. As described in “hints and tips,” Apache server returns a

200 response with a file of image/jpeg type, even if I've disabled mod_negotiation or removed "MultiViews" option or removed jpg from MIME mapping file (/etc/mime.types).

Disable mod_negotiation: sudo a2dismod negotiation.conf

Removed "MultiViews": Option -MultiViews

Host::	localhost
Port:	80
Action:	GET /test.jpg HTTP/1.1 Host: localhost Accept: text/*

Submit

Port: 80

Location: localhost

Last data sent: GET /test.jpg HTTP/1.1 Host: localhost Accept: text/*

Here are the results returned by the server:

HTTP/1.1 200 OK
Date: Sun, 27 Oct 2019 15:29:06 GMT
Server: Apache/2.4.38 (Ubuntu)
Last-Modified: Sun, 27 Oct 2019 15:07:19 GMT
ETag: "55c89-595e5bd4c9764"
Accept-Ranges: bytes
Content-Length: 351369
Content-Type: image/jpeg

□□□□□□□□□□□□□□□□□□□□

\R

@□□Q@□□Q@□□Q@□E%-□

f
8

ppw(

□~I.

I read through the Apache document of MIME configuration (“Apache Module mod_mime”), which provides the ways of mapping MIME types to the resource name or extension, but there is no configuration for following the “accept” directive in the request header.

```
<FilesMatch “\.(jpg)$”>  
    ForceType image/jpeg  
</FilesMatch>
```

I also tried to add type-map handler, and get a 500 response. This may be a sign that I’m closer to what matters, but I have no clue of the next step (“Content Negotiation”).

```
AddHandler type-map .var
```

In the article *Content Negotiation*, it’s mentioned that httpd uses a negotiation algorithm which is not configurable (“Content Negotiation”).

As of now, I can’t find a way to cheat the algorithm for a “406” response.

7. Caching

Q: What do you see? Make a note of the ETag header value.

I only see the header info. The file itself is not replied. The ETag value is “2aa6-593b4506ad43d”

Host::

Port:

Action:

Port: 80

Location: localhost

Last data sent: HEAD /index.html HTTP/1.1 Host: localhost

Here are the results returned by the server:

HTTP/1.1 200 OK
Date: Mon, 28 Oct 2019 13:56:42 GMT
Server: Apache/2.4.38 (Ubuntu)
Last-Modified: Sun, 29 Sep 2019 17:18:56 GMT
ETag: "2aa6-593b4506ad43d"
Accept-Ranges: bytes
Content-Length: 10918
Vary: Accept-Encoding
Content-Type: text/html

Raw port machine, ©Jon Dron, 2008

When it's resubmitted, the only change is the response time. The ETag is not changed.

Host::

Port:

Action:

Port: 80

Location: localhost

Last data sent: HEAD /index.html HTTP/1.1 Host: localhost

Here are the results returned by the server:

HTTP/1.1 200 OK
Date: Mon, 28 Oct 2019 13:59:26 GMT
Server: Apache/2.4.38 (Ubuntu)
Last-Modified: Sun, 29 Sep 2019 17:18:56 GMT
ETag: "2aa6-593b4506ad43d"
Accept-Ranges: bytes
Content-Length: 10918
Vary: Accept-Encoding
Content-Type: text/html

Raw port machine, ©Jon Dron, 2008

After just a space being added, both the last modification date and the ETag value is changed.

Host::

Port:

Action:

Port: 80

Location: localhost

Last data sent: HEAD /index.html HTTP/1.1 Host: localhost

Here are the results returned by the server:

HTTP/1.1 200 OK
Date: Mon, 28 Oct 2019 14:01:38 GMT
Server: Apache/2.4.38 (Ubuntu)
Last-Modified: Mon, 28 Oct 2019 14:01:20 GMT
ETag: "2aa7-595f8ef240301"
Accept-Ranges: bytes
Content-Length: 10919
Vary: Accept-Encoding
Content-Type: text/html

Raw port machine, ©Jon Dron, 2008

The name and version of the web server for the sites www.athabascau.ca is Apache/2.2.3 (Red Hat)

Host:

Port:

Action:

```
HEAD / HTTP/1.1
Host: www.athabascau.ca
```

Port: 80

Location: www.athabascau.ca

Last data sent: HEAD / HTTP/1.1 Host: www.athabascau.ca

Here are the results returned by the server:

HTTP/1.1 301 Moved Permanently
Date: Mon, 28 Oct 2019 14:03:59 GMT
Server: Apache/2.2.3 (Red Hat)
Location: <https://www.athabascau.ca/>
Content-Type: text/html; charset=iso-8859-1

Raw port machine, ©Jon Dron, 2008

8. What is this "host" thing?

I established two virtual hosts.

Viewer.saware.com, which is an unsecured site

Host::

Port:

Action:

Port: 80

Location: localhost

Last data sent: GET / HTTP/1.1 Host: viewer.saware.com Accept: */*

Here are the results returned by the server:

HTTP/1.1 200 OK
Date: Mon, 28 Oct 2019 14:11:50 GMT
Server: Apache/2.4.38 (Ubuntu)
Last-Modified: Mon, 28 Oct 2019 14:01:20 GMT
ETag: "2aa7-595f8ef240301"
Accept-Ranges: bytes
Content-Length: 10919
Vary: Accept-Encoding
Content-Type: text/html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www
<html xmlns="http://www.w3.org/1999/xhtml">
<!--
Modified from the Debian original for Ubuntu
Last updated: 2016-11-16
See: https://launchpad.net/bugs/1288690
-->
```

And member.saware.com, which requires SSL connection

Host::	<input type="text" value="localhost"/>
Port:	<input type="text" value="443"/>
Action:	<input type="text" value="GET / HTTP/1.1
Host: member.saware.com
Accept: */*"/>

Port: 443

Location: localhost

Last data sent: GET / HTTP/1.1 Host: member.saware.com Accept: */*

Here are the results returned by the server:

HTTP/1.1 400 Bad Request
Date: Mon, 28 Oct 2019 14:13:16 GMT
Server: Apache/2.4.38 (Ubuntu)
Content-Length: 446
Connection: close
Content-Type: text/html; charset=iso-8859-1

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
Reason: You're speaking plain HTTP to an SSL-enabled server port.<br />
Instead use the HTTPS scheme to access this URL, please.<br />
</p>
```

9. Cookies and a bit more redirection

I created a cookie.php at the root of the unsecured host viewer.saware.com. The cookie info is replied in the header.

Host::

Port:

Action:

Port: 80

Location: localhost

Last data sent: GET /cookie.php HTTP/1.1 Host: localhost Accept: */*

Here are the results returned by the server:

HTTP/1.1 200 OK
Date: Mon, 28 Oct 2019 21:16:14 GMT
Server: Apache/2.4.38 (Ubuntu)
Set-Cookie: LastVisit=1572297374; expires=Wed, 27-Nov-2019 21:16:14 GMT; Max-Age=2592000; path=/
Vary: Accept-Encoding
Content-Length: 133
Content-Type: text/html; charset=UTF-8

<html>
<body>

**Current Time : October 28, 2019 05:16:14 PM
Cookie named 'LastVisit' is not set!**
</body>
</html>

Reference Material

Adkins, B. "Why does HTTP Basic authentication encode the username and password with base64?" *Stack Exchange*. Jan. 30, 2013. Retrieved from: <https://security.stackexchange.com/questions/29916/why-does-http-basic-authentication-encode-the-username-and-password-with-base64>. Accessed on Oct. 30, 2019.

"Apache Module mod_mime." *Apache*. Retrieved from: https://httpd.apache.org/docs/trunk/mod/mod_mime.html. Accessed on Oct. 30, 2019.

"Content Negotiation." *Apache*. Retrieved from: <https://httpd.apache.org/docs/trunk/content-negotiation.html>. Accessed on Oct. 30, 2019.

Gunnars, K. "Should You Have a Trailing Slash at the End of URLs?" *Search Facts*. Aug. 15, 2019. Retrieved from: <https://searchfacts.com/url-trailing-slash/>. Accessed on Oct. 30, 2019.