# Web Server Design

## Lecture 8 – Authentication

Old Dominion University
Department of Computer Science
CS 431/531 Fall 2019
**Sawood Alam** <salam@cs.odu.edu>

2019-10-17

Original slides by Michael L. Nelson

# Authentication: Basic & Digest

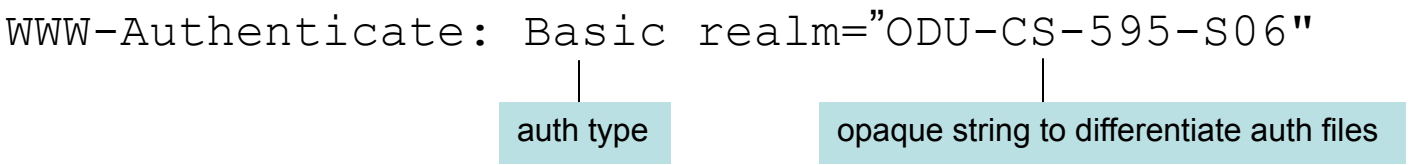
- Originally defined in RFC 2617, now covered in RFCs 7235, 7616, 7617
  - *Basic*
    - Very simple
    - Sends password in the clear (very bad)
    - Suitable for personalization; not real security
  - *Digest*
    - Uses cryptographic hashes; password not sent in the clear
    - Stronger than Basic, but client support not as prevalent
    - Does not encrypt content…
      - TLS, S-HTTP, or equivalent needed for that
- Extensible: other authentication mechanisms defined elsewhere
  - e.g.: OAuth, RFC 5849 (and RFCs 6749, 8252)

# Authentication Structure

- Both methods are structurally similar:
  - When the server receives a request for a protected resource, it responds with:
    - Status code “401 Unauthorized”
    - “WWW-Authenticate:” response header
  - The client reissues the same request with the addition of:
    - “Authorization:” request header

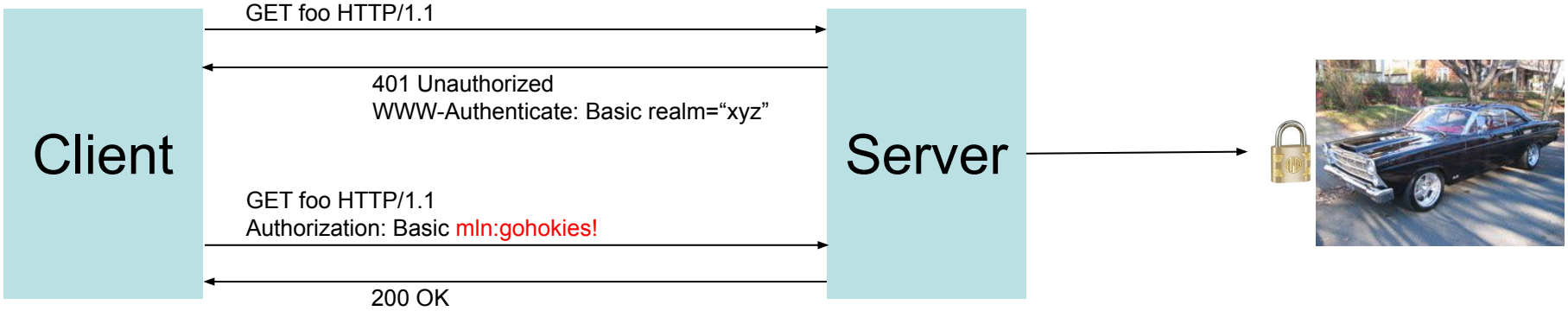
# Basic

- “WWW-Authenticate:” response header:




- “Authorization:” request header:

# Scenario: Initial 401 Response




Scenario 2: the client could have sent the Authorization string with the initial request

mln:gohokies! would be base64'd

# How Apache Does It?

*(Note: our syntax will be different!)*

- In either <Directory> entries in the config file, or ".htaccess" files in directories:

```
AuthType Basic
AuthName "Restricted Files"
AuthUserFile /usr/local/apache/passwd/passwords
Require user mln
```

- Many more options possible:
  - http://httpd.apache.org/docs/current/howto/auth.html

# Apache Example

```
$ htpasswd -c ./passwd mln
New password:
Re-type new password:
Adding password for user mln
$ ls
encode.pl*  foo.txt  passwd
$ more passwd
mln:Ggkaf46I4CTRI
$ ls -al
total 48
drwxr-xr-x  2 mln faculty 1024 2012-03-12 18:32 ./
drwxr-xr-x 14 mln faculty 1024 2012-01-30 10:48 ../
-rw-r--r--  1 mln faculty  151 2012-03-12 18:32 .htaccess
-rwxr-xr-x  1 mln faculty   94 2012-01-09 19:02 encode.pl*
-rw-r--r--  1 mln faculty   24 2012-01-09 19:02 foo.txt
-rw-r--r--  1 mln faculty   18 2012-03-12 18:32 passwd
```

Note: Having your password file accessible by the web server is normally a terrible idea!

# Apache Example (2)

```
$ cat .htaccess
AuthType Basic
AuthName "This is What the RFC Calls a Realm"
AuthUserFile /home/mln/public_html/teaching/cs595-s12/restrict/passwd
Require user mln
$ cat encode.pl
#!/usr/bin/perl

use MIME::Base64;

$string = encode_base64("mln:mln");

print "$string\n";

$ ./encode.pl
bWxuOm1sbg==
```

Note base64 “=“ padding: https://en.wikipedia.org/wiki/Base64#Decoding_Base64_with_padding

# Original Request

```
$ telnet www.cs.odu.edu 80
Trying 128.82.4.2...
Connected to xenon.cs.odu.edu.
Escape character is '^]'.
HEAD /~mln/teaching/cs595-s12/restrict/ HTTP/1.1
Host: www.cs.odu.edu

HTTP/1.1 401 Authorization Required
Date: Mon, 12 Mar 2012 22:40:55 GMT
Server: Apache/2.2.17 (Unix) PHP/5.3.5 mod_ssl/2.2.17
OpenSSL/0.9.8q
WWW-Authenticate: Basic realm="This is What the RFC Calls a Realm"
Content-Type: text/html; charset=iso-8859-1

Connection to xenon.cs.odu.edu closed by foreign host.
```

# Second Request

```
$ telnet www.cs.odu.edu 80
Trying 128.82.4.2...
Connected to xenon.cs.odu.edu.
Escape character is '^]'.
HEAD /~mln/teaching/cs595-s12/restrict/ HTTP/1.1
Host: www.cs.odu.edu
Authorization: Basic bWxuOm1sbg==

HTTP/1.1 200 OK
Date: Mon, 12 Mar 2012 22:42:49 GMT
Server: Apache/2.2.17 (Unix) PHP/5.3.5 mod_ssl/2.2.17 OpenSSL/0.9.8q
Content-Type: text/html;charset=ISO-8859-1

Connection to xenon.cs.odu.edu closed by foreign host.
```

# Wrong user:pw == 401

```
$ telnet www.cs.odu.edu 80
Trying 128.82.4.2...
Connected to xenon.cs.odu.edu.
Escape character is '^]'.
HEAD /~mln/teaching/cs595-s12/restrict/ HTTP/1.1
Host: www.cs.odu.edu
Authorization: Basic bWxuOm1alsdkfsldjslkdjfl==

HTTP/1.1 401 Authorization Required
Date: Mon, 12 Mar 2012 22:46:05 GMT
Server: Apache/2.2.17 (Unix) PHP/5.3.5 mod_ssl/2.2.17 OpenSSL/0.9.8q
WWW-Authenticate: Basic realm="This is What the RFC Calls a Realm"
Content-Type: text/html; charset=iso-8859-1

Connection to xenon.cs.odu.edu closed by foreign host.
```

# 401 comes before 404 et al.

```
$ telnet www.cs.odu.edu 80
Trying 128.82.4.2...
Connected to xenon.cs.odu.edu.
Escape character is '^]'.
HEAD /~mln/teaching/cs595-s12/restrict/THISDIRDOESNOTEXIST/ HTTP/1.1
Host: www.cs.odu.edu
Authorization: Basic bWxuOm1alsdkfsldjslkdjfl==

HTTP/1.1 401 Authorization Required
Date: Mon, 12 Mar 2012 22:53:51 GMT
Server: Apache/2.2.17 (Unix) PHP/5.3.5 mod_ssl/2.2.17 OpenSSL/0.9.8q
WWW-Authenticate: Basic realm="This is What the RFC Calls a Realm"
Content-Type: text/html; charset=iso-8859-1

Connection to xenon.cs.odu.edu closed by foreign host.
```

# Why Not a “403 Forbidden”?

RFC 7231

6.5.3. 403 Forbidden

The 403 (Forbidden) status code indicates that the server understood the request but refuses to authorize it. A server that wishes to make public why the request has been forbidden can describe that reason in the response payload (if any).

If authentication credentials were provided in the request, the server considers them insufficient to grant access. The client SHOULD NOT automatically repeat the request with the same credentials. The client MAY repeat the request with new or different credentials. However, a request might be forbidden for reasons unrelated to the credentials.

An origin server that wishes to "hide" the current existence of a forbidden target resource MAY instead respond with a status code of 404 (Not Found).

# 403 Reminder

```
$ touch bar.txt
$ chmod 000 bar.txt
$ telnet www.cs.odu.edu 80
Trying 128.82.4.2...
Connected to xenon.cs.odu.edu.
Escape character is '^]'.
HEAD /~mln/teaching/cs595-s12/restrict/bar.txt HTTP/1.1
Host: www.cs.odu.edu
Authorization: Basic bWxuOm1sbg==

HTTP/1.1 403 Forbidden
Date: Mon, 12 Mar 2012 22:56:15 GMT
Server: Apache/2.2.17 (Unix) PHP/5.3.5 mod_ssl/2.2.17 OpenSSL/0.9.8q
Content-Type: text/html; charset=iso-8859-1

Connection to xenon.cs.odu.edu closed by foreign host.
```

# Digest Authentication

- Still based on the challenge / response mechanisms first employed in “Basic” authentication
  - Same:
    - Status code 401
    - WWW-Authenticate response header
    - Authorization request header
  - New:
    - Authentication-Info response header

# Response

```
WWW-Authenticate: Digest
(1) realm="mln@www.cs.odu.edu",
(2) domain="http://www.cs.odu.edu/~mln/teaching/cs595-s07",
(3) qop="auth,auth-int",
(4) nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
(5) algorithm="MD5",
(6) stale="false",
(7) opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

(1) Similar to Basic's notion of realm
(2) List of URIs for which this challenge applies (all URIs at the server if not listed)
(3) Quality of protection: authentication, authentication with integrity
(4) Opaque data stream issued by the server per 401 challenge (not per access!)
(5) The algorithm used to encode the secrets
(6) If "true", then password is good, but previous nonce is old (retry w/ new nonce, don't prompt user)
(7) Opaque data stream issued by the server for a particular domain

# Request

```
Authorization: Digest username="mln",
(1) realm="mln@www.cs.odu.edu",
(2) uri="http://www.cs.odu.edu/~mln/teaching/cs595-s07/a1-test/",
(3) qop=auth,
(4) nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
(5) nc=00000001,
(6) opaque="5ccc069c403ebaf9f0171e9517f40e41",
(7) cnonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
(8) response="6629fae49393a05397450978507c4ef1"
```

(1) Specifies the realm
(2) Specifies the URI in a particular domain
(3) Quality of protection: authentication, authentication with integrity
(4) The nonce provided by the server in the 401 response (hex value)
(5) Nonce count -- how many times this nonce has been used
(6) Opaque data stream issued by the server for a particular domain
(7) Client-generated nonce
(8) The request digest

# Constructing the Request Digest

```
request_digest = md5(md5(A1):nonce:ncount:cnonce:qop:md5(A2))

        if algorithm == MD5,
            A1 = username:realm:password
        elsif algorithm == MD5-sess
            A1 = md5(username:realm:password):nonce:cnonce

        if qop == auth
            A2 = method:URI
        elsif qop == auth-int
            A2 = method:URI:md5(entity-body)
```

Originally derived from sections 3.2.2.1 - 3.2.2.3 in RFC 2617;
Now see section 3.4 for RFC 7616, which now allows SHA-256, among others.

# Generating Nonce / Opaque Values

“nonce” suggestion from 3.3:

```
base64(time-stamp md5(time-stamp:ETag:private-key))
```


“opaque” discussed in 3.3, but no suggestions are given. This field can be used to encode server state information; esp. useful if after authentication, a 301/302 is generated. We’ll use:

```
md5(URI:private-key)
```

# Authentication-Info Response Header

```
Authentication-Info:
(1) nextnonce="1a28b7102dd2f0e8b11d0f600bfbdd441",
(2) qop=auth,
(3) rspauth="d3b07384d113edec49eaa6238ad5ff00",
(4) nc=00000001,
(5) cnonce="dcd98b7102dd2f0e8b11d0f600bfb0c093"
```

(1) Optional, allows 1 time nonce values (at expense of efficiency; consider nonce count instead)
(2) Quality of protection: authentication, authentication with integrity
(3) Optional, supports mutual authentication (server knows client's password)
(4) Nonce count -- how many times this nonce has been used
(5) Client-generated nonce

# Constructing the Response Authorization

Same as constructing the request digest that goes into the “Authorization” request header, except:

```
if qop == auth
   A2 = :URI
elsif qop == auth-int
   A2 = :URI:md5(entity-body)
```

(Same as before, but the method is not applicable. Note leading colon!)

Section 3.5 in RFC 7616

# Request for A Digest Protected URI

```
$ telnet www.cs.odu.edu 80
Trying 128.82.4.2...
Connected to xenon.cs.odu.edu.
Escape character is '^]'.

GET http://www.cs.odu.edu:80/~mln/teaching/cs595-s09/a4-test/limited2/foo/bar.txt HTTP/1.1
User-Agent: CS 595-S07 A3 automated Checker
Host: www.cs.odu.edu
Connection: close

HTTP/1.1 401 Authorization Required
Date: Sun, 29 Mar 2009 15:17:40 GMT
Server: Apache/2.2.0
WWW-Authenticate: Digest realm="Colonial Place",
 nonce="AARmQ3eCGoo=642d940339fe011ff1eb3d026d9ed55266b61183", algorithm=MD5, qop="auth"
Content-Length: 471
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>401 Authorization Required</title>
</head><body>
<h1>Authorization Required</h1>
<p>This server could not verify that you
[deletia]
```

# Re-Request With Authorization Header

```
$ telnet www.cs.odu.edu 80
Trying 128.82.4.2...
Connected to xenon.cs.odu.edu.
Escape character is '^]'.

GET http://www.cs.odu.edu:80/~mln/teaching/cs595-s09/a4-test/limited2/foo/bar.txt HTTP/1.1
Authorization: Digest username="mln", realm="Colonial Place",
 uri="http://www.cs.odu.edu:80/~mln/teaching/cs595-s09/a4-test/limited2/foo/bar.txt",
 qop=auth, nonce="AARmQ3eCGoo=642d940339fe011ff1eb3d026d9ed55266b61183",
 nc=00000001, cnonce="014a54548c61ba03827ef6a4dc2f7b4c", response="099f6f84cd7d2ff4e92d01adea40b2a9"
Host: www.cs.odu.edu
Connection: close

HTTP/1.1 200 OK
Date: Sun, 29 Mar 2009 15:17:40 GMT
Server: Apache/2.2.0
Authentication-Info: rspauth="e3cd2569795632cca41d52a4610ed4c3",
 cnonce="014a54548c61ba03827ef6a4dc2f7b4c", nc=00000001, qop=auth
Last-Modified: Fri, 09 Jan 2009 16:53:23 GMT
ETag: "13267f-12-985006c0"
Accept-Ranges: bytes
Content-Length: 18
Connection: close
Content-Type: text/plain

Can you see this?
```

# Client-Side Code Snippets

```
import hashlib

uri = f"http://{hostport}/a4-test/limited2/foo/bar.txt"

# Parse nonce and opaque (if it exists) values from the server response
cnonce = hashlib.md5(b"go hokies").hexdigest()
ncount = "00000001"
a1 = hashlib.md5("mln:Colonial Place:mln".encode()).hexdigest()
a2 = hashlib.md5(f"GET:{uri}".encode()).hexdigest()
response = hashlib.md5(f"{a1}:{nonce}:{ncount}:{cnonce}:auth:{a2}".encode()).hexdigest()

# Prepare a GET request with Authorization header
f'''GET {uri} HTTP/1.1
Authorization: Digest username="mln",
 realm="Colonial Place",
 uri={uri}",
 qop=auth,
 nonce="{nonce}",
 nc={ncount},
 cnonce="{cnonce}",
 response="{response}"
Host: {hostport}
Connection: close

'''
```

Note that values for nc & qop are not quoted

# How?

```
$ cat /a4-test/limited2/.htaccess
AuthType Digest
AuthName "Colonial Place"
AuthDigestProvider file
AuthUserFile /home/mln/passwd-digest
Require valid-user
$ cat ~/passwd-digest
bda:Colonial Place:b8e13248f7bb96682093c850d5c7da46
jbollen:Colonial Place:c5d7f97a6ac34b393ba2d252c7331d5a
mln:Colonial Place:53bbb5135e0f39c1eb54804a66a95f08
vaona:Colonial Place:fbcc0f347e4ade65a337a4febc421c81

$ cat /a4-test/limited2/WeMustProtectThisHouse\!
#
# A4 password file
#
authorization-type=Digest
#
realm="Colonial Place"
#
bda:Colonial Place:b8e13248f7bb96682093c850d5c7da46
jbollen:Colonial Place:c5d7f97a6ac34b393ba2d252c7331d5a
mln:Colonial Place:53bbb5135e0f39c1eb54804a66a95f08
vaona:Colonial Place:fbcc0f347e4ade65a337a4febc421c81
```

Apache

Our Server

https://www.youtube.com/watch?v=dnECY26PSHk