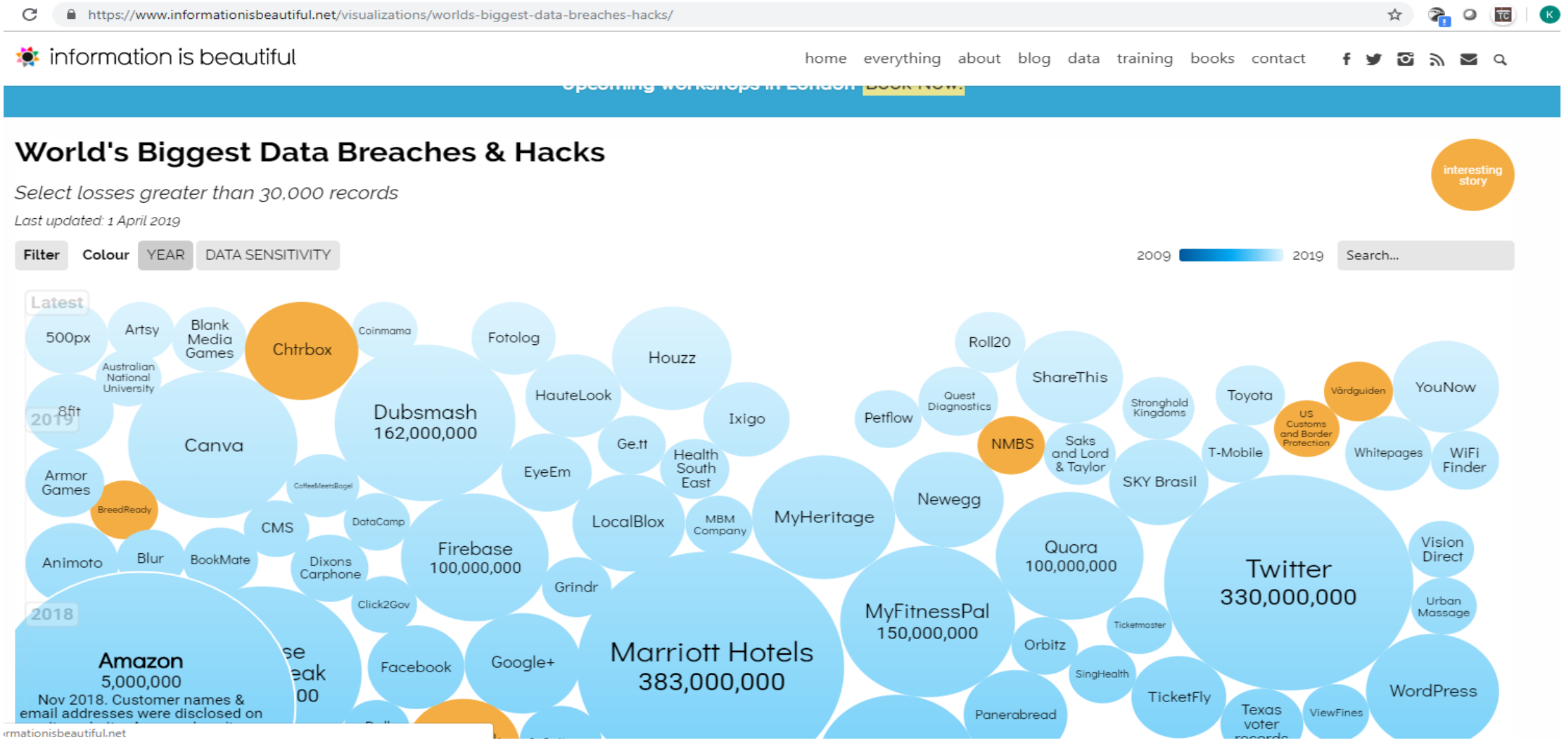
Ethical Hacking: Hacking Web Servers

World Biggest Data Breaches& Hacks



Example Of Web Server Breaches

Unpatched server led to GlobalSign breach

GlobalSign failed to update one of its web servers, which allowed a hacker to access it, and led to the company ceasing operations for more than a week.

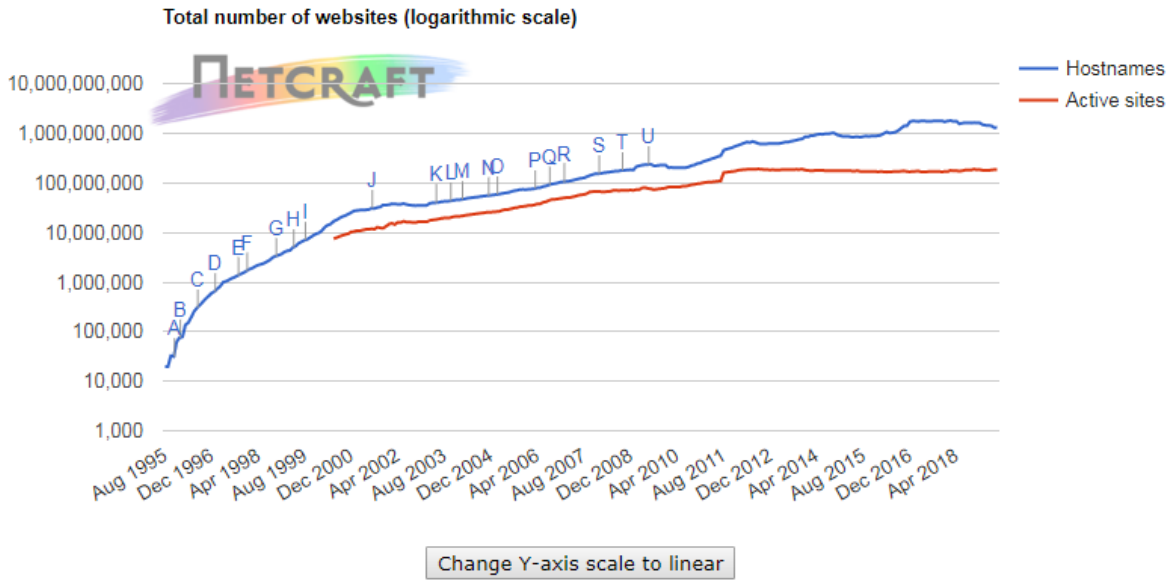
EA games web server was hosting PHISHING SITE – securobod

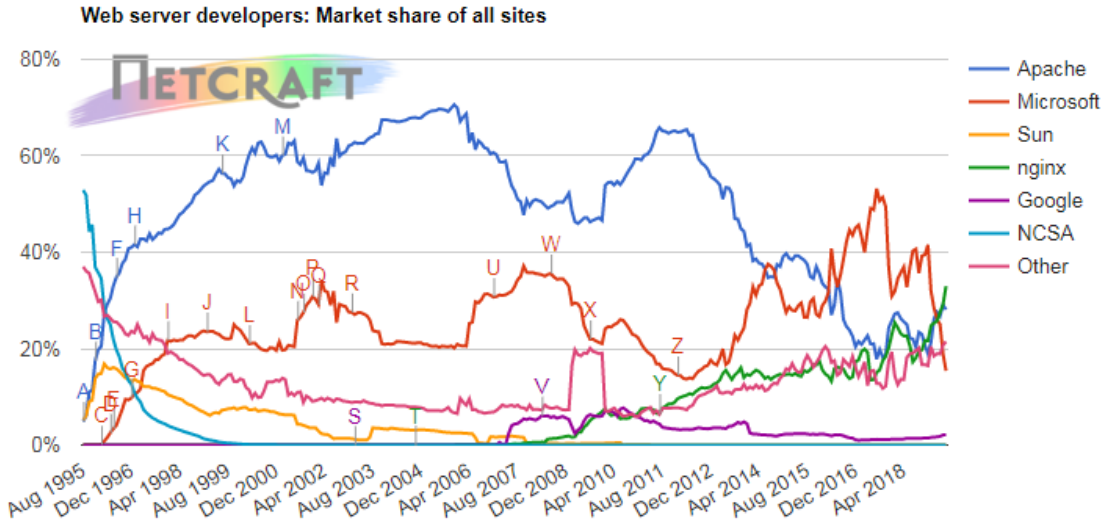
Old vulnerable software gave hackers a way in, claims researcher

HealthCare.gov Server Hacked, Infected With Malware

Cybercriminals managed to breach one of the servers used for HealthCare.gov, the official website of the United States' health insurance marketplace, federal officials reported on Thursday.

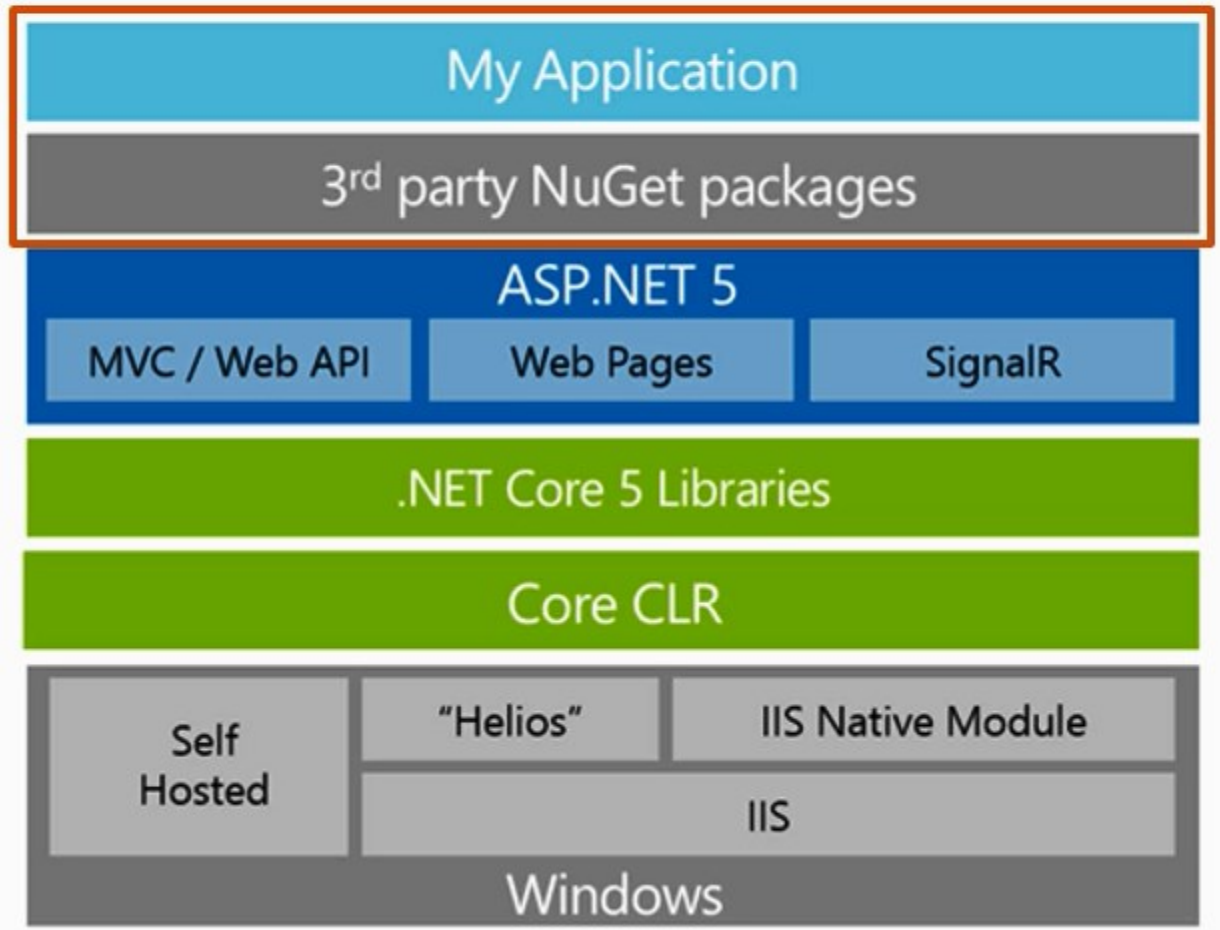
Market Share Of Web servers



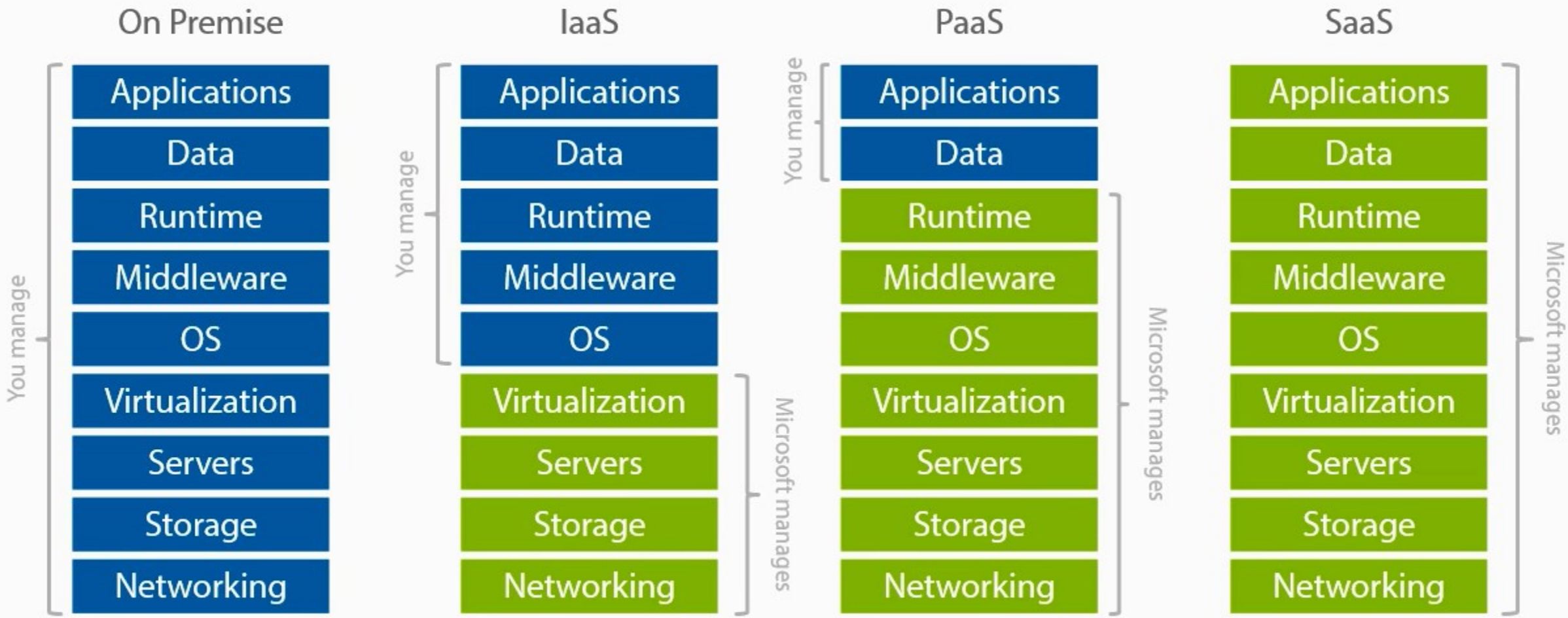


Developer	May 2019	Percent	June 2019	Percent	Change
nginx	387,416,889	29.20%	439,626,713	32.97%	3.77
Apache	385,685,252	29.07%	374,360,949	28.08%	-1.00
Microsoft	250,440,887	18.88%	205,235,291	15.39%	-3.49
Google	27,711,375	2.09%	28,181,744	2.11%	0.02

Web Servers Versus Web Applications



The Role of Cloud

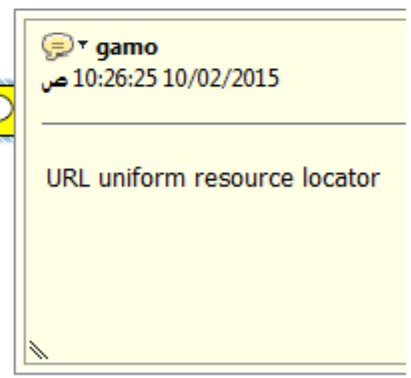


Discovering Risks in Web Servers

The HTTP Protocol

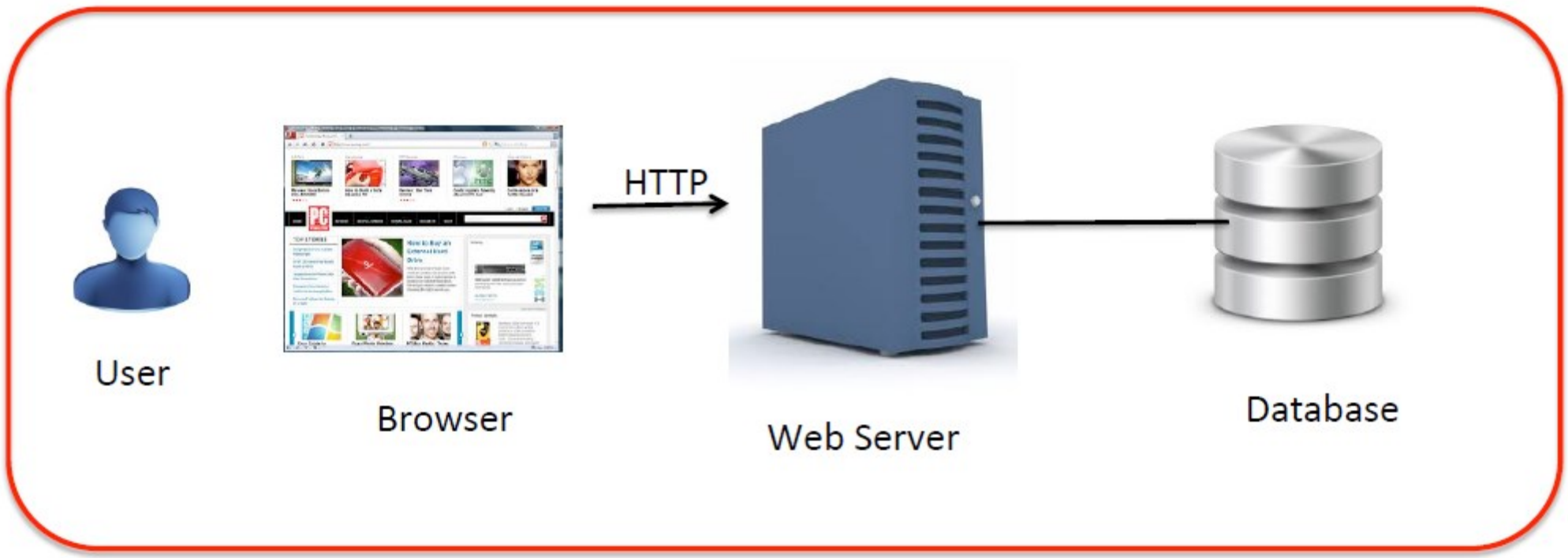
Hypertext Transfer Protocol

- Client-Server based architecture
- Request-Response model to serve Resources
- Resources are identified by URI / URL

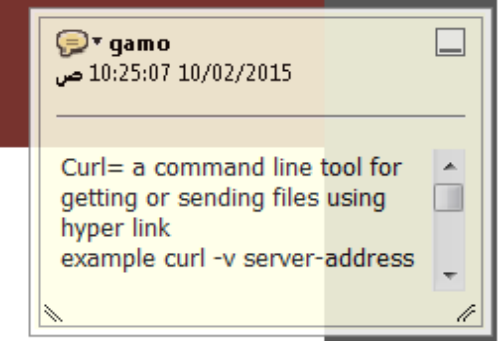


- Versions – HTTP 1.0/1.1
  - 1.1 can reuse connection for multiple URIs

How does it really work?



Netcat / Curl / Browser




```
PentesterAcademy# curl -v www.securitytube.net

* About to connect() to www.securitytube.net port 80 (#0)
*   Trying 74.125.135.121...
* connected
* Connected to www.securitytube.net (74.125.135.121) port 80 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.26.0
> Host: www.securitytube.net
> Accept: */*
>
* additional stuff not fine transfer.c:1037: 0 0
* additional stuff not fine transfer.c:1037: 0 0
* HTTP 1.1 or later with persistent connection, pipelining supported
< HTTP/1.1 200 OK
< Content-Type: text/html; charset=utf-8
< Cache-Control: no-cache
< Vary: Accept-Encoding
< Date: Fri, 30 Aug 2013 11:46:11 GMT
< Server: Google Frontend
< Alternate-Protocol: 80:quic,80:quic
< Transfer-Encoding: chunked
```

HTTP Header

```
▾ Hypertext Transfer Protocol
  ▾ GET /?q=pentesting HTTP/1.1\r\n
    ▸ [Expert Info (Chat/Sequence): GET /?q=pentesting HTTP/1.1\r\n]
      Request Method: GET
      Request URI: /?q=pentesting
      Request Version: HTTP/1.1
    Host: www.securitytube.net\r\n
    User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:22.0) Gecko/20100101 Firefox/22.0 Iceweasel/22.0\r\n
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
    Accept-Language: en-US,en;q=0.5\r\n
    Accept-Encoding: gzip, deflate\r\n
    Referer: http://www.securitytube.net/\r\n
    Cookie: __utma=93873311.193485658.1377862705.1377862705.1377868083.2; __utmz=93873311.1377862705.1.1.utmcsr=(
    Connection: keep-alive\r\n
    \r\n
```

HTTP Request Methods

- GET
  - Parameters in URL
- POST
  - Form submissions, data in message body
- OPTIONS
  - List of methods supported for URL
- HEAD
  - Response for GET but no message body
- TRACE
  - Echo client request back for diagnostics
- PUT
  - Store in URI
- DELETE
  - Delete resource

OPTIONS – Might not be allowed!

```
PentesterAcademy# curl -v -X OPTIONS http://www.google.com
* About to connect() to www.google.com port 80 (#0)
*   Trying 173.194.36.113...
* connected
* Connected to www.google.com (173.194.36.113) port 80 (#0)
> OPTIONS / HTTP/1.1
> User-Agent: curl/7.26.0
> Host: www.google.com
> Accept: */*
>
* additional stuff not fine transfer.c:1037: 0 0
* HTTP 1.1 or later with persistent connection, pipelining supported
< HTTP/1.1 405 Method Not Allowed
< Content-Type: text/html; charset=UTF-8
< Content-Length: 962
< Date: Fri, 30 Aug 2013 13:12:31 GMT
< Server: GFE/2.0
< Alternate-Protocol: 80:quic
```

GET vs HEAD

```
PentesterAcademy# curl -v -X HEAD http://www.vivekramachandran.com
* About to connect() to www.vivekramachandran.com port 80 (#0)
*   Trying 67.205.50.44...
* connected
* Connected to www.vivekramachandran.com (67.205.50.44) port 80 (#0)
> HEAD / HTTP/1.1
> User-Agent: curl/7.26.0
> Host: www.vivekramachandran.com
> Accept: */*
>
* additional stuff not fine transfer.c:1037: 0 0
* HTTP 1.1 or later with persistent connection, pipelining supported
< HTTP/1.1 200 OK
< Date: Sun, 01 Sep 2013 12:52:08 GMT
< Server: Apache
< Last-Modified: Mon, 14 Feb 2011 04:38:27 GMT
< ETag: "18dd-49c369e6db6c0"
< Accept-Ranges: bytes
< Content-Length: 6365
< Vary: Accept-Encoding
< Content-Type: text/html
<
* additional stuff not fine transfer.c:1037: 0 0
* additional stuff not fine transfer.c:1037: 0 0
* transfer closed with 6365 bytes remaining to read
* Closing connection #0
curl: (18) transfer closed with 6365 bytes remaining to read
```

HEAD - Security Risks

- Authentication Bypass
  - Auth only applied for GET, POST and not HEAD
  - http://www.fishnetsecurity.com/6labs/blog/jboss-jmx-console-authentication-bypass
- HTTP Verb Tampering
  - Aspect Security
  - http://jeremiahgrossman.blogspot.in/2008/06/what-you-need-to-know-about-http-verb.html

HTTP Basic Authentication

HTTP Response Codes

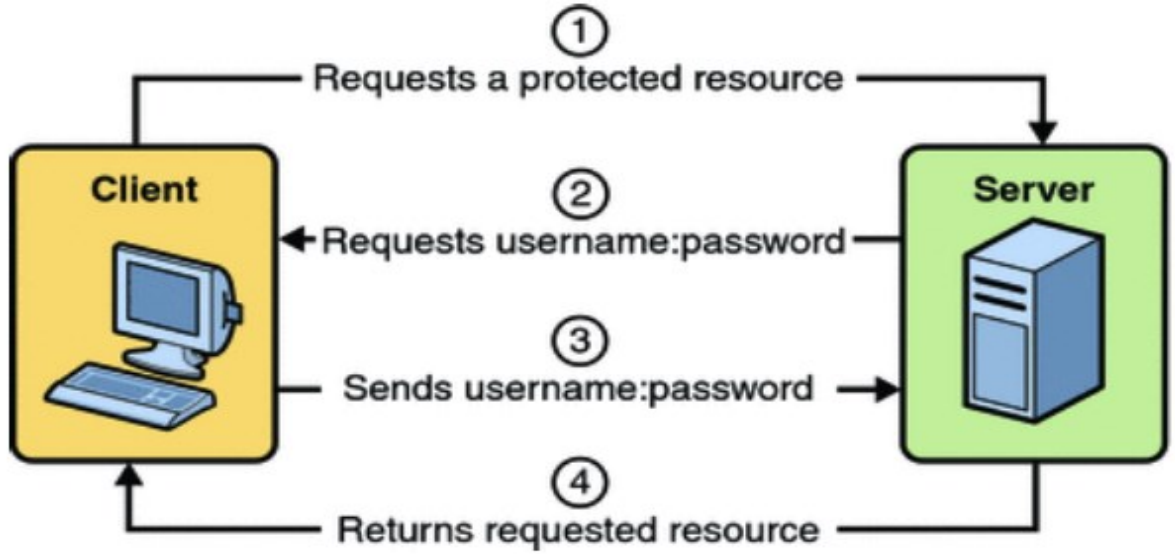
- 1xx = Informational
- 2xx = Request Successful e.g. 200 OK
- 3xx = Redirects e.g. 302 Moved Temporarily
- 4xx = Client Request Errors e.g. 401 Unauthorized
- 5xx = Server Side Errors

Authentication in HTTP

- Basic Authentication

- Digest Authentication

Basic Authentication



Source: http://docs.oracle.com/cd/E19226-01/820-7627/bncbo/index.html

HTTP Digest Authentication

- Basic Authentication sends User:Pass in plaintext
- Digest Authentication sends a Hash of the password
- RFC 2069, 2617

Initial Version – RFC 2069

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest     realm="testrealm@host.com",
                             nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
                             opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

The client may prompt the user for the username and password, after which it will respond with a new request, including the following Authorization header:

```
Authorization: Digest        username="Mufasa",
                             realm="testrealm@host.com",
                             nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
                             uri="/dir/index.html",
                             response="e966c932a9242554e42c8ee200cec7f6",
                             opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

Response Calculation

Hash1 = MD5(Username:Realm:Password)

Hash1 = MD5(admin:Pentester Academy:asdss)

Hash2 = MD5(method:URI)

Hash2 = MD5(GET:/lab/webapp/digest2/1)

Response Calculation

Hash1 = MD5(Username:Realm:Password)

Hash2 = MD5(method:URI)

Response = MD5(Hash1:Nonce:Hash2)

Hash1 Calculation

Hash1 = MD5(Username:Realm:Password)

```
PentesterAcademy# python
Python 2.7.1 (r271:86832, Jul 31 2011, 19:30:53)
[GCC 4.2.1 (Based on Apple Inc. build 5658) (LLVM build 2335.15.00)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> import hashlib
>>>
>>> hash1 = hashlib.md5('admin:Pentester Academy:asdss').hexdigest()
>>>
>>> hash1
'a524e9245a8bf88560d2bb74a02a8779'
>>>
```

Hash2 Calculation

Hash2 = MD5(method:URI)

```
PentesterAcademy#
PentesterAcademy# python
Python 2.7.1 (r271:86832, Jul 31 2011, 19:30:53)
[GCC 4.2.1 (Based on Apple Inc. build 5658) (LLVM build 2335.15.00)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> import hashlib
>>>
>>> hash2 = hashlib.md5('GET:/lab/webapp/digest2/1').hexdigest()
>>>
>>> hash2
'210e62c14f54e4a5f76da73fc1cfa73b'
>>>
```

Response Calculation

Response = MD5(Hash1:Nonce:Hash2)

```
>>>
>>> import hashlib
>>>
>>> nonce = "c671e71e6105016b797f16b809a0ac69"
>>>
>>> hash1 = hashlib.md5('admin:Pentester Academy:asdss').hexdigest()
>>>
>>> hash2 = hashlib.md5('GET:/lab/webapp/digest2/1').hexdigest()
>>>
>>> response = hashlib.md5("%s:%s:%s" % (hash1, nonce, hash2)).hexdigest()
>>>
>>> hash1
'a524e9245a8bf88560d2bb74a02a8779'
>>> hash2
'210e62c14f54e4a5f76da73fc1cfa73b'
>>> response
'8b8e22a52910eaeab8c9eff78beed84c'
```

HTTP Digest Authentication (RFC 2617)

RFC 2617 – Security Enhanced

- Adds Client Nonce
  - Mitigate chosen Plain-text attacks
- Adds "QOP" – Quality of Protection
  - auth for Authentication
  - auth-int for Authentication and Integrity
    - Rarely used and not well supported

HTTP Digest Authentication with QOP=auth

The first time the client requests the document, no Authorization header is sent, so the server responds with:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest
        realm="testrealm@host.com",
        qop="auth,auth-int",
        nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
        opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

The client may prompt the user for the username and password, after which it will respond with a new request, including the following Authorization header:

```
Authorization: Digest username="Mufasa",
        realm="testrealm@host.com",
        nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
        uri="/dir/index.html",
        qop=auth,
        nc=00000001,
        cnonce="0a4f113b",
        response="6629fae49393a05397450978507c4ef1",
        opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

Response Calculation (RFC 2617)

Hash1 = MD5(Username:Realm:Password)

Hash2 = MD5(method:URI)

Response = MD5(Hash1:Nonce:NonceCount:Client-Nonce:QOP:Hash2)

Digest Authentication

```
GET / HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:22.0) Gecko/20100101 Firefox/22.0 Iceweasel/22.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
If-Modified-Since: Tue, 23 Jul 2013 12:26:26 GMT
If-None-Match: "486ae-b1-4e22ce6d50080"
Authorization: Digest username='vivek", realm="Pentester-Academy", nonce="cmMXCA/nBAA=7002cad884ece9b87dd63d4a0aa7f3b1cf9f731b", uri="/",
algorithm=MD5, response='9444743d2960f562e0145a53cc4e2390", qop=auth, nc=00000001, cnonce="c5470d4d075843c9"

HTTP/1.1 304 Not Modified
Date: Mon, 23 Sep 2013 15:54:32 GMT
Server: Apache/2.2.22 (Debian)
Connection: Keep-Alive
Keep-Alive: timeout=5, max=100
ETag: "486ae-b1-4e22ce6d50080"
Vary: Accept-Encoding
```

Response Calculation (RFC 2617)

Hash1 = MD5(Username:Realm:Password)

```
PentesterAcademy# python
Python 2.7.3 (default, Jan  2 2013, 13:56:14)
[GCC 4.7.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> import hashlib
>>>
>>> hash1 = hashlib.md5("vivek:Pentester-Academy:pentesteracademy").hexdigest()
>>>
>>> hash1
'4260744fc3c98fd3223426fc152374a4'
>>>
```

Response Calculation (RFC 2617)

Hash2 = MD5(method:URI)

```
>>>
>>>
>>> hash2 = hashlib.md5("GET:/").hexdigest()
>>>
>>> hash2
'71998c64aea37ae77020c49c00f73fa8'
>>>
```

Response Calculation (RFC 2617)

Response = MD5(Hash1:Nonce:NonceCount:Client-Nonce:QOP:Hash2)

```
>>>
>>> nonce = "cmMXCA/nBAA=7002cad884ece9b87dd63d4a0aa7f3b1cf9f731b"
>>>
>>> nonceCount = "00000001"
>>>
>>> clientNonce = "c6470d4d075843c9"
>>>
>>> qop = "auth"
>>>
>>> response_string = hash1 + ':' + nonce + ':' + nonceCount + ':' + clientNonce + ':' + qop + ':' + hash2
>>>
>>> response = hashlib.md5(response_string).hexdigest()
>>>
>>> response
'9444743d2960f562e0145a53cc4e2390'
```

HTTP Statelessness and Cookies

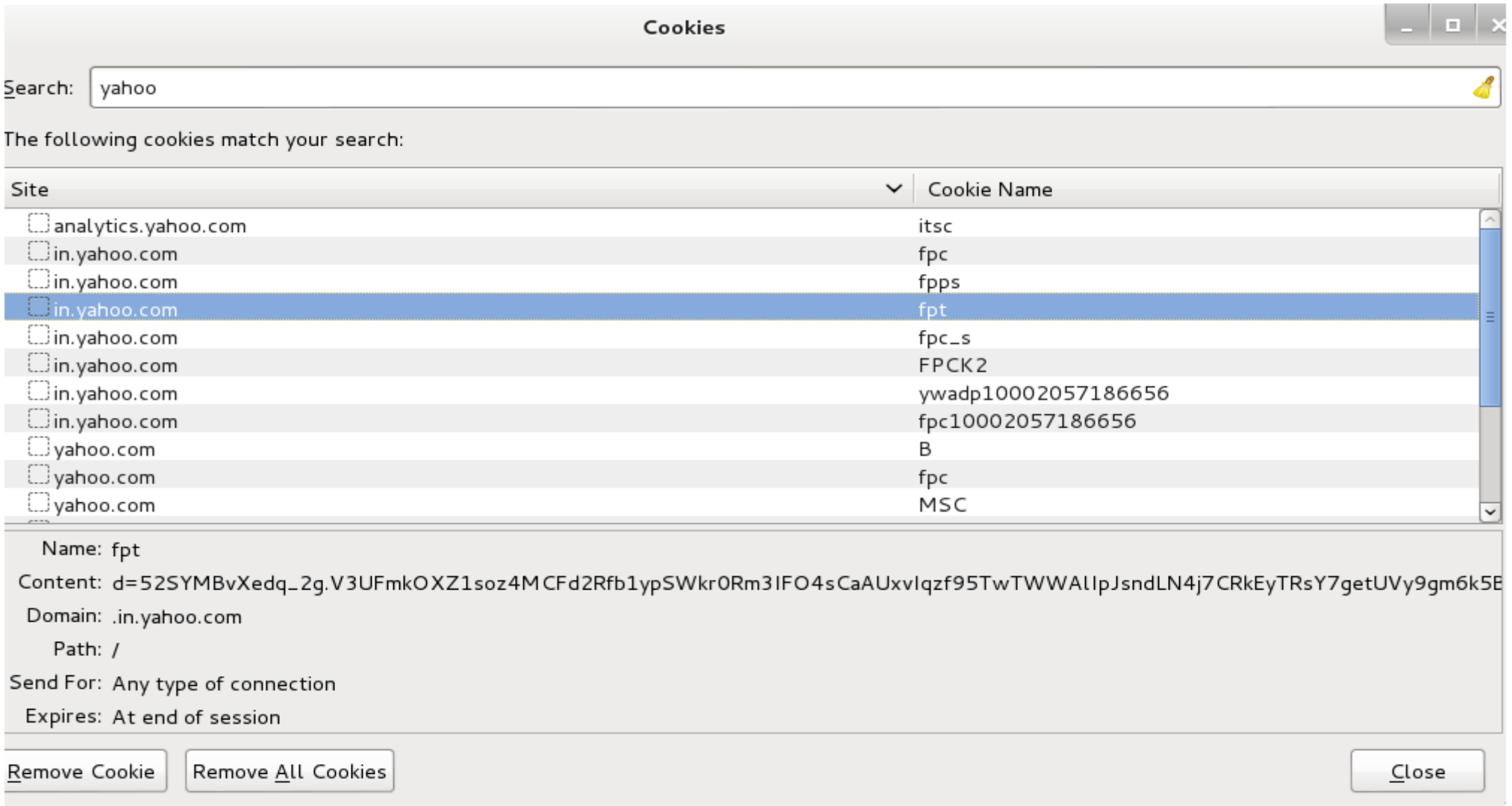
HTTP is Stateless

- Every request is treated independently
- Server does not retain state for clients
- What does this mean?
  - Every request needs to be separately authenticated
  - Every request MUST carry auth information

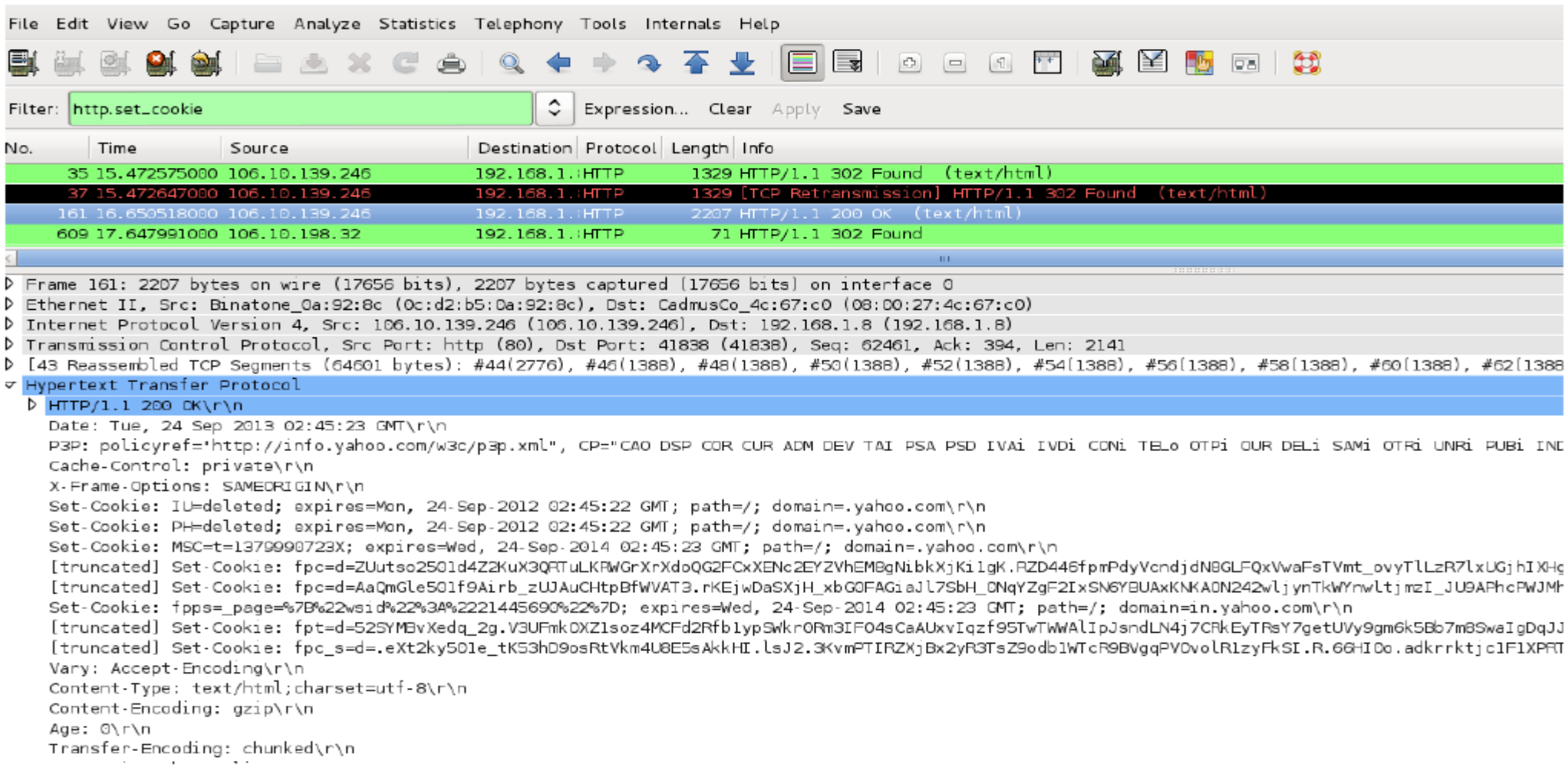
Cookies

- Allows server to store and retrieve data from the client
- Typically stored in a file on the client side
- Text only; No executable code
- Cannot exceed 4K in size
- Allows for retaining state with the Client's help
  - Session Management
  - User Preferences

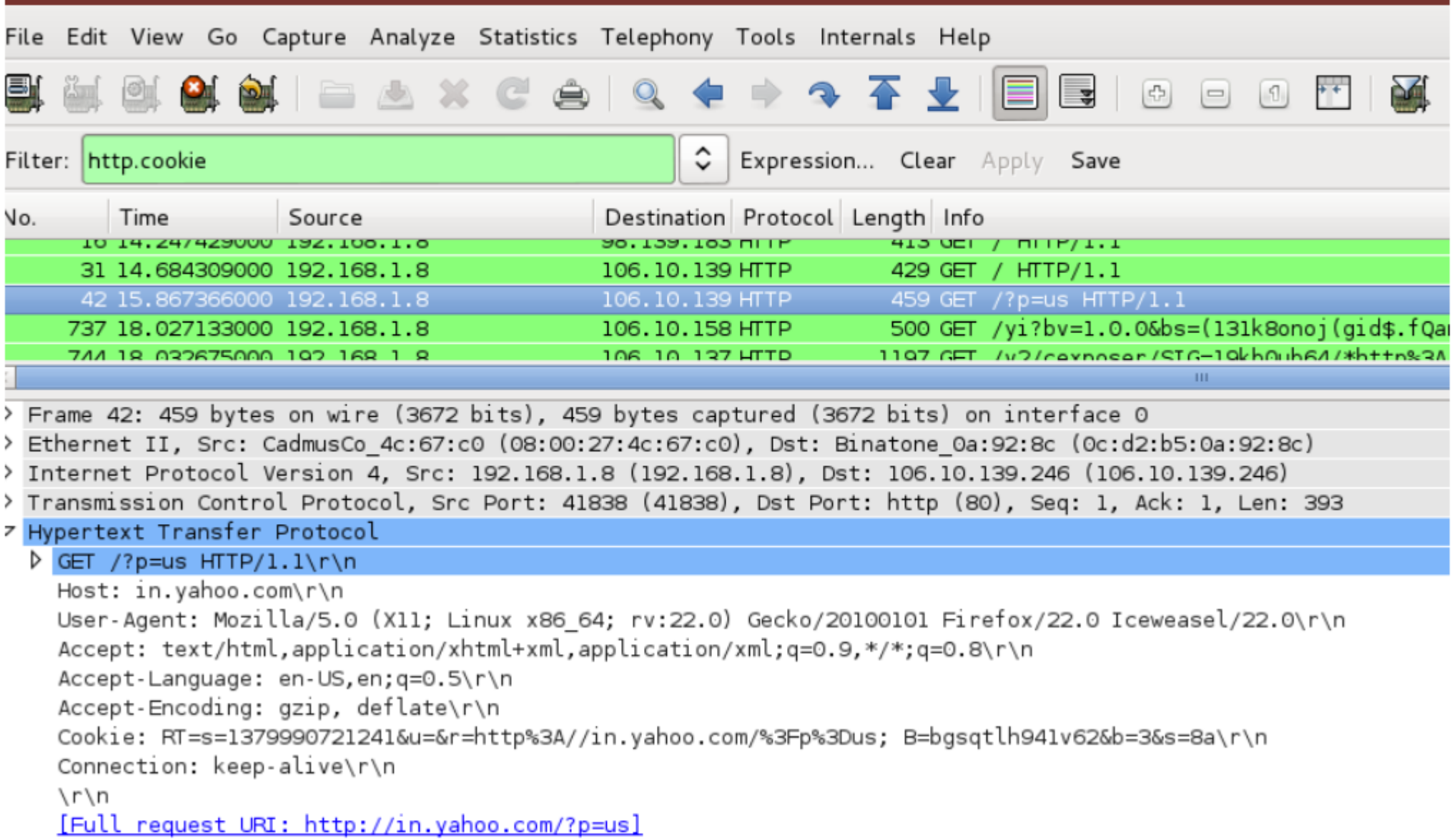
How does a Cookie look?



How is a Cookie set by the Server?



How is a Cookie sent by the Client?



Session ID

What is Session ID?

- Unique identifier or token to identify a user and session

- Maybe provided to both authenticated and anonymous users

http://en.wikipedia.org/wiki/Session_ID

Where is it stored?

- URL
- Form Field
- Cookie
- ...

Ideal Session ID

- Long
- Random
- Name should not suggest functionality
- Should timeout and not be recycled
- Should not be derived using shared secrets e.g. passwords, usernames
- Sent over secure channel
  - Depends on application importance e.g. banking

SSL – Transport Layer Protection

HTTP is Plain Text

- Eavesdropping on a connection
- Use of insecure Wi-Fi at Coffee Shops, Airports most common example
- Credentials stolen
  - Username + Password
  - Cookies and Session ID

HTTPS

- Transport Layer Protection
- Create an encrypted tunnel first and send data through it
- Typically port 443

How does HTTPS look like?

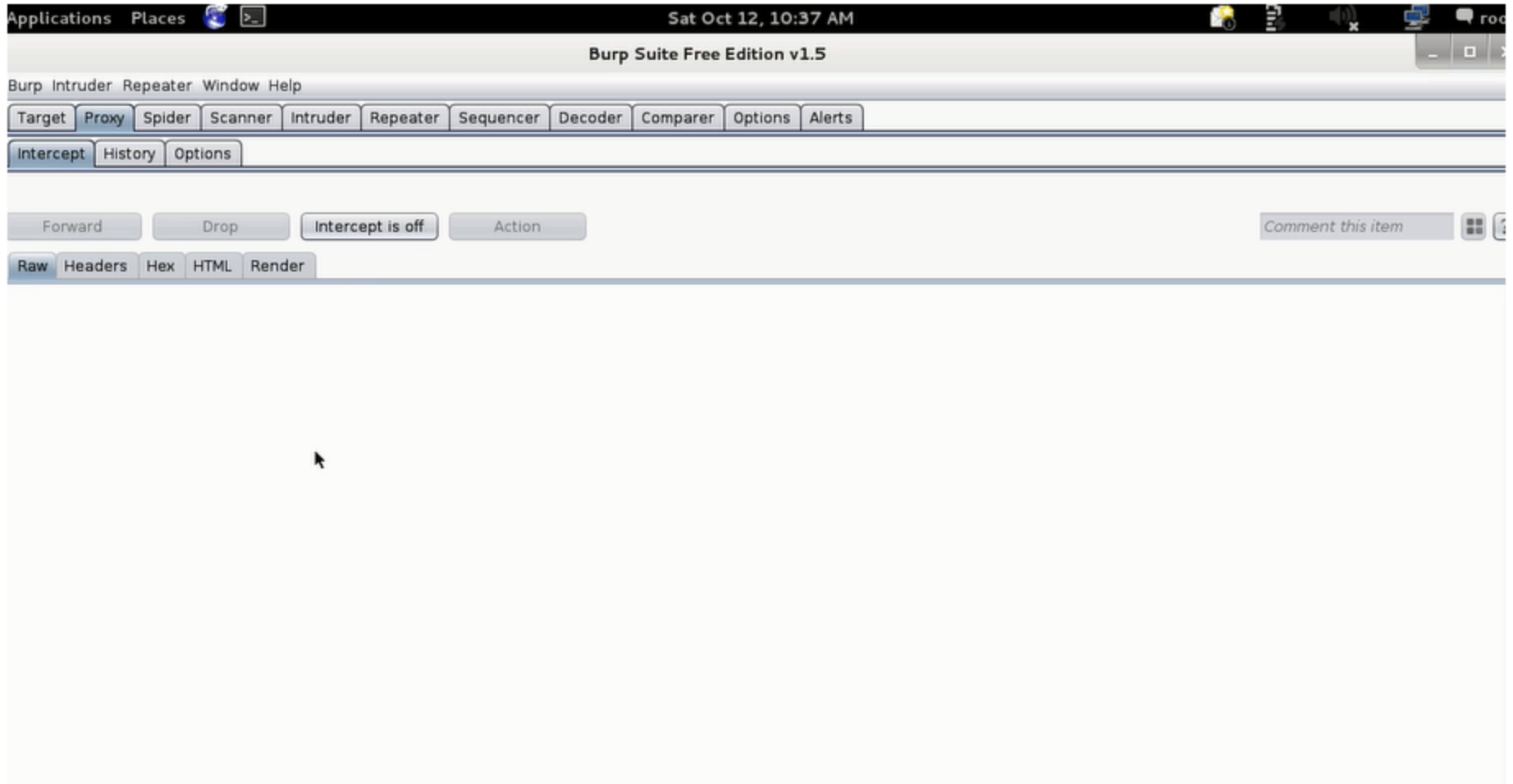
- Can we at least see the URL?

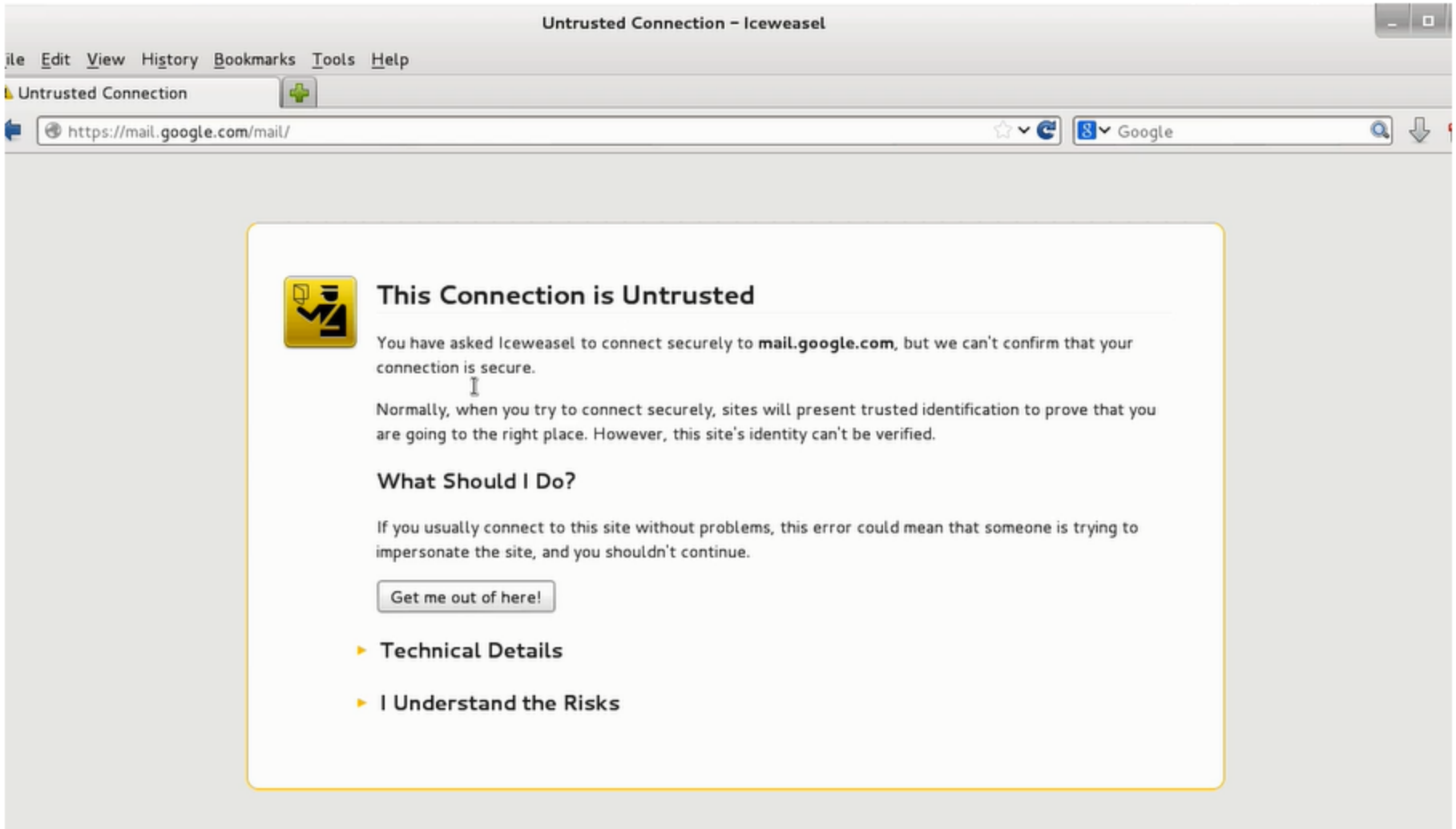
- Can we figure out any data at all?

How do we peer into a HTTPS connection?

- Have access to private key for decryption
- MITM attack with self signed certificate
- Tools like SSLStrip to force https to http
  - Network Pentesting course

SSL MITM using Proxies


Applications
Places
Sat Oct 12, 10:37 AM
Burp Suite Free Edition v1.5
Burp Intruder Repeater Window Help
Target
Proxy
Spider
Scanner
Intruder
Repeater
Sequencer
Decoder
Comparer
Options
Alerts
Intercept
History
Options
Forward
Drop
Intercept is off
Action
Comment this item
Raw
Headers
Hex
HTML
Render


Untrusted Connection – Iceweasel
ile Edit View History Bookmarks Tools Help
Untrusted Connection
https://mail.google.com/mail/
Google
This Connection is Untrusted
You have asked Iceweasel to connect securely to mail.google.com, but we can't confirm that your connection is secure.
Normally, when you try to connect securely, sites will present trusted identification to prove that you are going to the right place. However, this site's identity can't be verified.
What Should I Do?
If you usually connect to this site without problems, this error could mean that someone is trying to impersonate the site, and you shouldn't continue.
Get me out of here!
Technical Details
I Understand the Risks


Burp Suite Free Edition v1.5
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Options Alerts
Intercept History Options
Request to https://mail.google.com:443 [74.125.236.86]
Forward Drop Intercept is on Action
Comment this item
Raw Params Headers Hex
GET /mail?gxlu=demo%40gmail.com&zx=1381588810598 HTTP/1.1
Host: mail.google.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:22.0) Gecko/20100101 Firefox/22.0 Iceweasel/22.0
Accept: image/png,image/*;q=0.8,*/*;q=0.5
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: https://accounts.google.com/ServiceLogin?service=mail&passive=true&rm=false&continue=https://mail.google.com/mail/&ss=1&scc=1&ltmpl=default&ltmplcache=2&emr=1
Cookie: PREF=ID=1fce3f33684473b2:U=617476b107259f23:FF=0:TM=1377862670:LM=1378039737:S=D0iwj8gFc5QfZ60d;
NID=67=bwiSzMVeBsYWbtA68QxNR5HmZl0anw31AsQ9VbDc1n5YxC9turL_v9qbxJ4xI_jluLcw5FI2Bd5PP11SBfzYGcumi7LV2gcdfnSmeioX_fs9xS4FMHfGGOMKL-NHIQ1v; GMAIL_RTT=1579
Connection: keep-alive