CSEP 561 – Web Protocols

David Wetherall
djw@cs.washington.edu
Web protocols

- Focus:
  - Applications and their transport needs
- HTTP as an example
Transports we have

- **TCP**
  - Reliable, congestion-controlled bytestream
- **UDP**
  - Unreliable individual short messages
  - Error detection if you are nice
  - (Packets!)
Example applications and their needs

• Video conferencing
  – Unreliable video stream (congestion friendly)
• Video-on-demand (streaming media)
  – Reliable bytestream with buffered playback (congestion control)
• DNS
  – Request / reply
  – Reliable, short messages

• Web
  – Series of related request / replies
  – Reliable, variable length messages (congestion control)

• Not exactly a great match to what we have …
Web architecture
Web Protocol Stacks

- HTTP is a request/response protocol mainly used for retrieving Web resources that normally runs on TCP port 80
Simple HTTP 1.0

- HTTP is a tiny, text-based language
- The GET method requests an object
- There are HTTP headers, like “Content-Length:”, etc.
- Try “telnet server 80” then “GET index.html HTTP/1.0”
  – Other methods: POST, HEAD,… google for details
HTTP 1.0 Request/Responses

Time sequence diagram for a series of HTTP request/responses
HTTP 1.0 Performance

- Well, it’s lower than it could be:
  - Web pages are made up of many files.
  - Most are very small (< 10k)
  - Each file is mapped to a separate TCP connection

- For each file
  - Setup/Teardown (and Time-Wait table bloat for server)
  - 2RTT “first byte” latency
  - Slow Start + AIMD Congestion Avoidance

- The goals of HTTP and TCP protocols are not aligned!
TCP Behavior for Short Connections

**RTT=70ms**

![Graph showing throughput vs. connection length](image)

**Figure 3-2: Throughput vs. connection length, RTT = 70 msec**

Figure 3-2 shows that, in the remote case, using a TCP connection to transfer only 2 Kbytes results in a throughput less than 10% of best-case value. Even a 20 Kbyte transfer achieves only about 50% of the throughput available with a reasonable window size. This reduced throughput translates into increased latency for document retrieval. The figure also shows that, for this 70 msec RTT, use of too small a window size limits the throughput no matter how many bytes are transferred.
HTTP1.1: Persistent Connections

• Idea: Use one TCP connection for multiple page downloads (or HTTP methods). This is application layer multiplexing.

• Q: What are the advantages?
• Q: What are the disadvantages?
HTTP/1.1

- Only one TCP connection setup for multiple HTTP requests
- Can do HTTP request/responses serially (left) or pipeline them (right)
- Reduces network idle time and slow-start
- Also pipelining: send multiple requests before responses arrive
Effect of Persistent HTTP

Figure 6-2: Latencies for a remote server, image size = 45566 bytes
Caching

- It is faster and cheaper to get data that is closer
- Origin server may be geographically distant
- Closer can be:
  - Local browser cache (file system) (1-10ms)
  - Client-side proxy (institutional proxy) (10-50)
  - Content-distribution network (CDN -- “cloud” proxies) (50-100)
Browser Caches

- Bigger win: avoid repeated transfers of the same page
- Check local browser cache to see if we have the page
- GET with If-Modified-Since makes sure it’s up-to-date
Proxy Caches

- Insert further levels of caching for greater gain
- Share proxy caches between many users (not shown)
  - If I haven’t downloaded it recently, maybe you have
- Your browser has built-in support for this
Consistency and Caching Directives

• Key issue is knowing when cached data is fresh/stale
  – Otherwise many connections or the risk of staleness

• Caching directives provide hints
  – Expires: header is basically a time-to-live
  – Also indicate whether page is cacheable or not

• Browsers typically use heuristics
  – Check freshness once a “session” with GET If-Modified-Since and then assume it’s fresh the rest of the time
  – Possible to have inconsistent data.
Proxy Cache Effectiveness

- Can help significantly
- But benefits grow slowly with cache size
SSL – Transport layer security

- HTTPS is HTTP on top of Secure Socket Layer (SSL)
  - Used by client to verify server and protect messages

- Initially uses public key encryption to verify server and then switches over to secret session key

- Q: what happens in what order?
  - TCP connect, SSL handshake,
  - HTTP request/response?

- Q: how is the public key distributed? who verifies who?
  - Too many public keys for client to have the key for each server

<table>
<thead>
<tr>
<th>HTTP</th>
<th>SSL</th>
<th>TCP</th>
</tr>
</thead>
</table>

djw // CSE 461, Autumn 2011
Certificates (Public Key Authentication Chains)

- Use a trust hierarchy to distribute public keys with a Public Key Infrastructure (PKI)
- Encoded as certificates ("CA says public key for X is K")
  - Certificates issued by Certificate Authorities (CAs)
  - Clients need to trust (have public keys) a small number of CAs
    - Commonly distributed with OS, browser
- Now server can send client its public key in a certificate
  - Client can verify the public key by checking certificate
  - Then client can use public key, e.g., to verify server in SSL
X.509 Certificates

• **X.509**
  – Certificate format standard, global namespace
• Widely used, e.g., in Web browsers
  – Comes with 100 CAs?
Unfortunately: Public Key Revocation

• What if a private key is compromised?
  – Hope it never happens?

• Need certificate revocation list (CRL)
  – And a CRL authority for serving the list
  – Everyone using a certificate is responsible for checking to see if it is on CRL
  – ex: certificate can have two timestamps
    • one long term, when certificate times out
    • one short term, when CRL must be checked
    • CRL is online, CA can be offline
Microsoft Security Bulletin MS01-017
Erroneous VeriSign-Issued Digital Certificates Pose Spoofing Hazard

Originally posted: March 22, 2001
Updated: June 23, 2003

Summary
Who should read this bulletin:
All customers using Microsoft® products.

Impact of vulnerability:
Attacker could digitally sign code using the name "Microsoft Corporation".

Recommendation:
All customers should install the update discussed below.

Technical description:
In mid-March 2001, VeriSign, Inc., advised Microsoft that on January 29 and 30, 2001, it issued two VeriSign Class 3 code-signing digital certificates to an individual who fraudulently claimed to be a Microsoft employee. The common name assigned to both certificates is "Microsoft Corporation". The ability to sign executable content using keys that purport to belong to Microsoft would clearly be advantageous to an attacker who wished to convince users to allow the content to run.

The certificates could be used to sign programs, ActiveX controls, Office macros, and other executable content. Of these, signed ActiveX controls and Office macros would pose the greatest risk, because the attack scenarios involving them would be the most straightforward. Both ActiveX controls and Word documents can be delivered via either web pages or HTML mails. ActiveX controls can be automatically invoked via script, and Word documents can be automatically opened via script unless the user has applied the Office Document Open Confirmation Tool.
Update Available to Revoke Fraudulent Microsoft Certificates Issued by VeriSign

View products that this article applies to.

This article was previously published under Q293811

On This Page

SUMMARY
Important Notes
MORE INFORMATION

SUMMARY

In March, 2001, VeriSign, Inc. announced that it had issued two digital certificates to an individual who fraudulently claimed to be a Microsoft employee. This issue is discussed at length in Microsoft Security Bulletin MS01-017. VeriSign has revoked these certificates, and they are listed in the current VeriSign Certificate Revocation List (CRL). However, because the VeriSign code-signing certificates do not specify a CRL Distribution Point (CDP), it is not possible for any browser’s CRL-checking mechanism to locate and use the VeriSign CRL. Microsoft has developed an update that rectifies this problem. The update package includes a CRL that contains the two certificates, and an installable revocation handler that consults the CRL on the local computer, rather than attempting to use the CDP mechanism.
Speeding up the Web

- Page load delays come from many different sources …
With SSL!

<table>
<thead>
<tr>
<th>DNS Lookup</th>
<th>Initial Connection</th>
<th>SSL Negotiation</th>
<th>Time to First Byte</th>
<th>Content Download</th>
<th>Start Render</th>
<th>Document Complete</th>
<th>8xx result</th>
<th>9xx+ result</th>
</tr>
</thead>
</table>
| https://www.cs.washington.edu
2. www.cs.washington.edu - cse.css
5. www.cs.washington.edu - e_logo_80x133.png
8. www.cs.washington.edu - arrow.png
15. www.cs.washington.edu - yesjs.png
17. www.cs.washington.edu - 1.png
18. www.cs.washington.edu - favicon.ico | 0.5 | 1.0 | 1.5 | 2.0 | 2.5 | 3.0 | 3.5 | 4.0 | 4.5 | 5.0 | 5.5 | 6.0 | 6.5 | 7.0 | 7.5 |
| | | | | | | | | | | | | | | | |
| | 6360 ms | 910 ms | 748 ms | 139 ms | 134 ms | 300 ms | 138 ms | 952 ms | 206 ms | 635 ms | 445 ms | 131 ms | 146 ms | 141 ms | 136 ms |
Possibilities

• Google public DNS
  – Attention to server provisioning, caching for performance
• SPDY
  – An HTTP replacement with multiplexed streams (incl. priority), header compression, and server push
• WebP
  – A new image format, ~1/4-1/3 smaller than JPEG or PNG
• SSL False Start
  – Cuts one round-trip off SSL handshake. (Also SSL Snap Start.)
• Mobile web
  – Beware adverse interactions with cellular data connections