# Web Hosting

Michael Tsai
2019/06/03

# Web Hosting Basics

- A daemon (server) listens for connection on TCP port 80
- Accept request for documents
- Transmits them to the requesting user's browser

# URL

- URI = {URL, URN}
- URI: Unified Resource Identifier
  URL: Unified Resource Locator
  URN: Unified Resource Name
  (e.g., urn:isbn:0-13-020601-6)
- URL: {protocol/app., hostname, [port, directory, filename]}

# URL Examples

| Proto | What it does | Example |
|---|---|---|
| file | accesses a local file | file:///etc/syslog.conf |
| ftp | accesses a remote file via FTP | ftp://ftp.admin.com/adduser.tar.gz |
| http | accesses a remote file via HTTP | http://admin.com/index.html |
| https | accesses a remote file via HTTP/SSL | https://admin.com/order.shtml |
| ldap | accesses LDAP directory services | ldap://ldap.bigfoot.com:389/cn=Herb |
| mailto | sends email to a designated address | mailto:linux@book.admin.com |

# How HTTP works

- **Stateless** client/server protocol
- TCP and later TLS (Transport Layer Security)
- A client
  asks the server for the **contents** of
  a specific **URL**
  - content + layout + script + image / video
- User-agent initiates the request (always)**;**
  The server responds with the data (or err. msg.)
- Try it: telnet to port 80

# 人肉Browser

- telnet to www.csie.ntu.edu.tw port 80 (http default TCP port)
- Type the following:
  GET / HTTP/1.1
  Host: www.csie.ntu.edu.tw
  (hit <enter> twice)
- What do you get?
- Try a nonexistent URL. What do you get?

# HTTP overview

- Spend 10 minutes to read this.
- HTTP overview: https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview#

# Proxy

- Hosts to relay HTTP messages; most transparent to the users
- Common functions
  - Caching
  - Filtering
  - Load balancing
  - Authentication
  - Logging

# Load Balancing

- Many factors affecting the maximum load a server can handle:
  - Hardware architecture
  - Operating system
  - System tuning
  - Sites being served
    (static vs dynamic (database))
  - And, in addition, network bandwidth
- Stress testing - CPU, I/O, or network-bound? (usually not network)

# Create Scalability

1. Round robin DNS (we've talked about it)
   Note that the order in the DNS record is **irrelevant**.
   (Think about its disadvantage)

2. Hardware solution (e.g., Big-IP from F5)
   Takes response time of individual servers into account

3. Software solution (e.g., Linux Virtual Server, proxy load balancing in Apache)

# Scaling Beyond Limits

- Cloud computing (e.g., Amazon Web Services)
- Co-location hosting (like us, or some NTU services)
- Content Distribution Networks (e.g., akamai, limelight, edgecast)
  - Putting static content close to users
  - Try: https://www.cdnplanet.com/tools/cdnfinder/
  - Pick a content-rich website (such as a news website) and see if it uses CDN

# Different Layers in Web

- Web page (HTML/XML)  - frontend engineer
  - Interface for interaction with users
- Program / script (PHP/ASP/etc.) - backend engineer
  - Generation of web pages, logics, database query
- Database (Mysql, PgSQL, etc.) - database admin
  - Data storage
- Web server (apache/nginx) - sys admin
- Operating System - sys admin

Our role

# Content Generation

- Dynamic content is better

1. CGI (Common Gateway Interface):
   Allow external program to interact with the web server

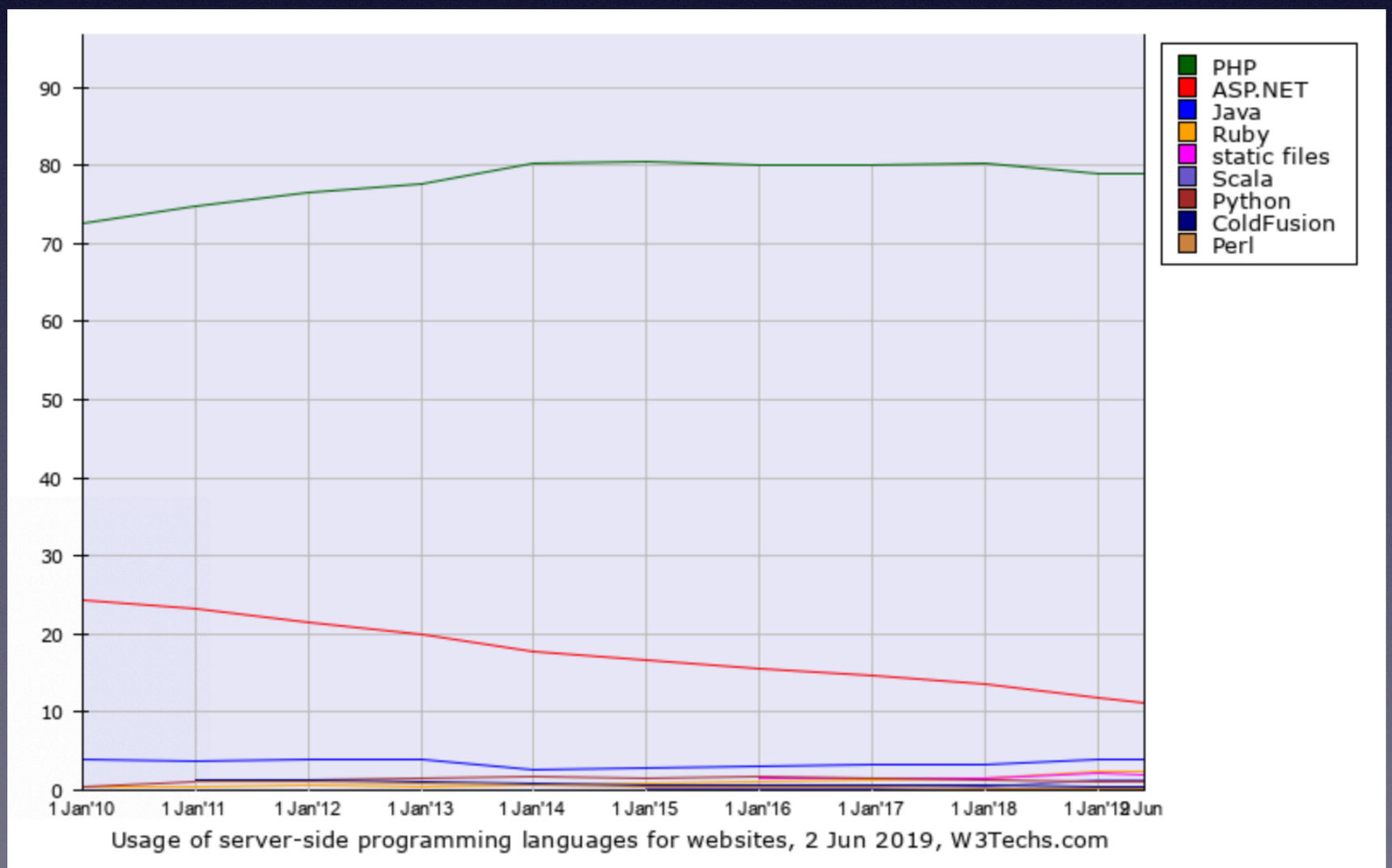
2. FastCGI: Allow external program to continue running to server multiple requests

3. Embedded interpreters:
   (e.g., Perl, PHP, Python, Ruby on Rails)
   Executing external script within the server (.php, .pl)
   e.g., LAMP: linux + apache + mysql + php/perl/python

4. Application servers:
   Entire, full-fledge, platform for web
   (e.g., Tomcat, WebSphere, WebLogic, Jetty)

# Market Share: Server-Side Programming Language

| | 2010 1 Jan | 2011 1 Jan | 2012 1 Jan | 2013 1 Jan | 2014 1 Jan | 2015 1 Jan | 2016 1 Jan | 2017 1 Jan | 2018 1 Jan | 2019 1 Jan | 2019 2 Jun |
|---|---|---|---|---|---|---|---|---|---|---|---|
| PHP | 72.5% | 74.8% | 76.6% | 77.7% | 80.3% | 80.6% | 80.0% | 80.0% | 80.2% | 78.9% | 79.0% |
| ASP.NET | 24.4% | 23.2% | 21.4% | 19.9% | 17.8% | 16.7% | 15.6% | 14.8% | 13.5% | 11.8% | 11.2% |
| Java | 4.0% | 3.8% | 3.9% | 4.0% | 2.6% | 2.8% | 3.1% | 3.3% | 3.4% | 4.0% | 4.0% |
| Ruby | 0.5% | 0.5% | 0.6% | 0.5% | 0.6% | 0.9% | 1.1% | 1.3% | 1.6% | 2.4% | 2.5% |
| static files | | | | | | | 1.5% | 1.5% | 1.6% | 2.1% | 2.1% |
| Scala | | | | | | 0.2% | 0.2% | 0.3% | 0.5% | 1.2% | 1.3% |
| Python | 0.3% | 1.0% | 1.3% | 1.5% | 1.7% | 1.6% | 1.7% | 1.6% | 1.3% | 1.1% | 1.1% |
| JavaScript | | | <0.1% | <0.1% | 0.1% | 0.1% | 0.2% | 0.3% | 0.4% | 0.7% | 0.7% |
| ColdFusion | | 1.3% | 1.2% | 1.1% | 0.8% | 0.7% | 0.7% | 0.6% | 0.6% | 0.5% | 0.5% |
| Perl | | 1.1% | 1.0% | 0.8% | 0.6% | 0.5% | 0.5% | 0.4% | 0.3% | 0.3% | 0.3% |
| Erlang | | | | | | 0.1% | 0.1% | 0.1% | 0.1% | 0.1% | 0.1% |
| Miva Script | | | | | | | 0.1% | <0.1% | <0.1% | <0.1% | <0.1% |



Usage of server-side programming languages for websites, 2 Jun 2019, W3Techs.com

# Security!

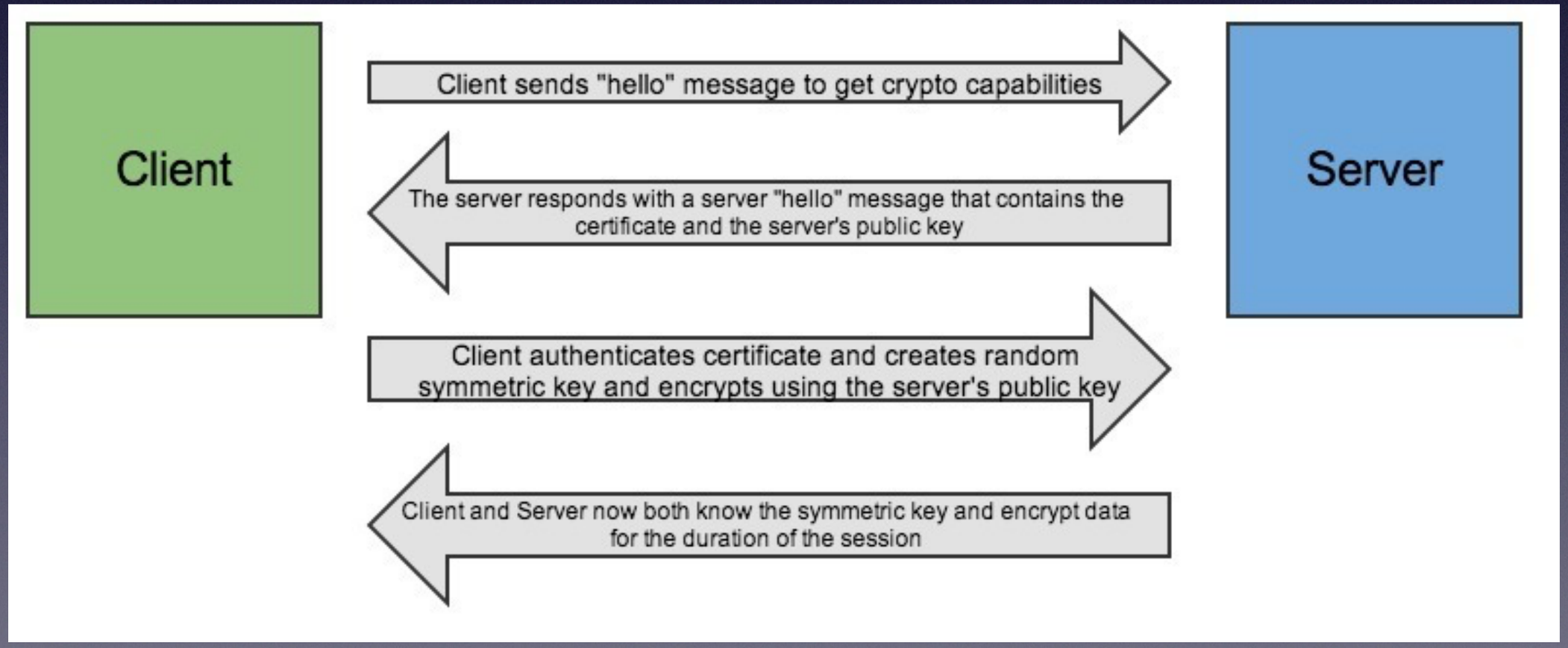
- Bottom line:
  you allow **the entire world** to execute a script on your server (access to files, networks, and more!)

- Need to make sure that the script is secure
  (as much as other network-accessible program)

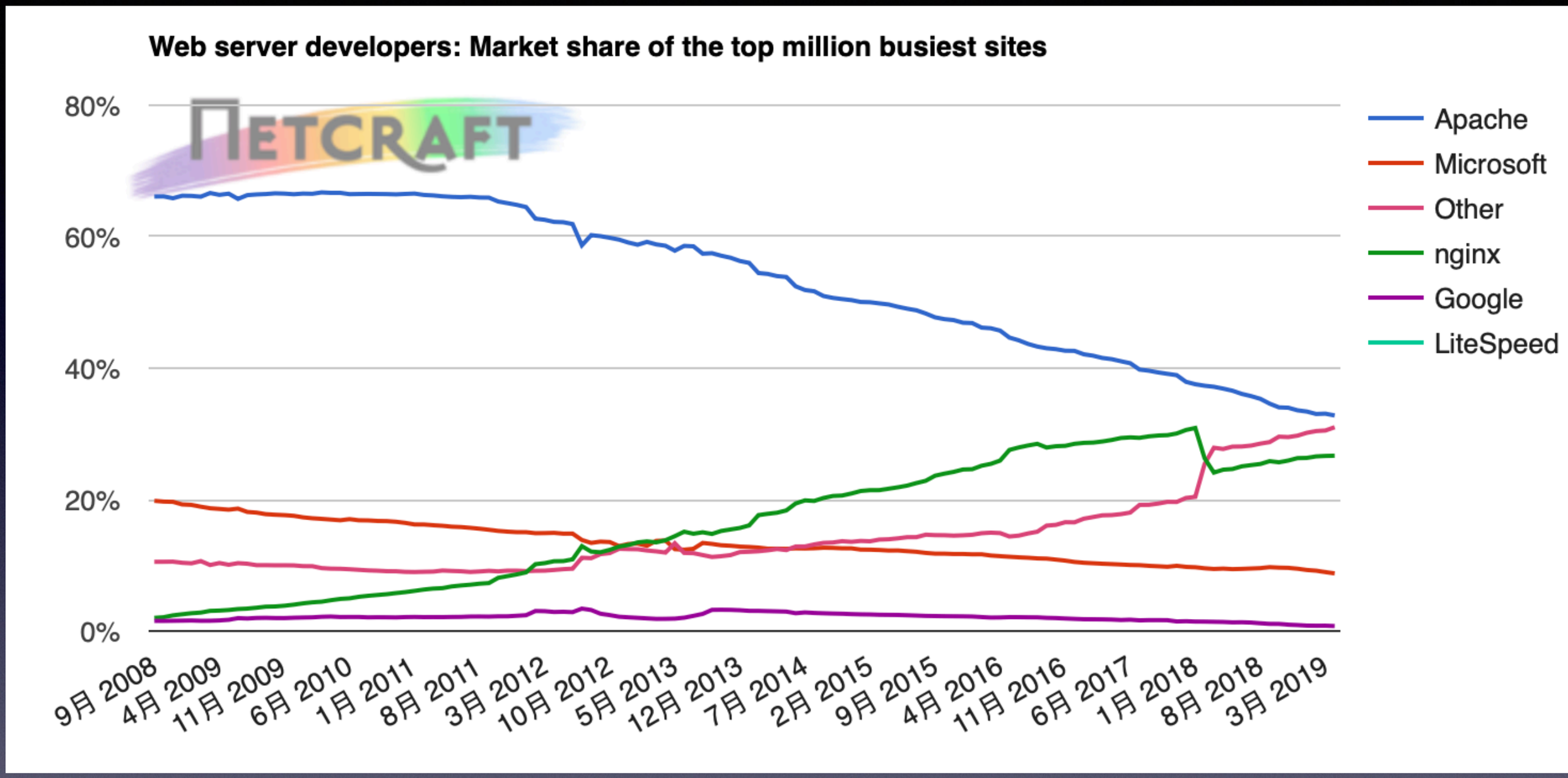
- Read: (20 minutes)
  OWASP Top 10 Application Security Risks 2017
  https://www.owasp.org/index.php/Top_10_2017-Top_10

# SSL

- SSL = Secure Sockets Layer
- http -> https, SSL runs as a layer below HTTP
- Prevent eavesdropping, tampering, and message forgery




Security-related topics were covered in the guest lecture




**Most websites now use nginx! (not shown in the figure above)**

source: https://news.netcraft.com/archives/2019/04/22/april-2019-web-server-survey.html

# Apache


- Web server with the largest market share (53.8% of top M busiest sites, 2014/04)
- Runner-ups:
  Microsoft & nginx take 17.8% and 12.4%
- Began in 1995
- First web server software to serve more than 100M sites (in 2009)
- Versatile

# Nginx

- (pronounced “engine  x”)
- Created in 2004
- Used by 57.0% of the top 10,000 websites. (W3Techs)
- Written with an explicit goal to outperform Apache (less memory, 4x more requests per second)
    - Less flexibility

# Different Layers in Web

- Web page (HTML/XML)  - frontend engineer
  - Interface for interaction with users
- Program / script (PHP/ASP/etc.) - backend engineer
  - Generation of web pages, logics, database query
- Database (Mysql, PgSQL, etc.) - database admin
  - Data storage
- Web server (apache/nginx) - sys admin
- Operating System - sys admin

Our role

# Zuvio Homework

- Complete the assignment in the lab:
  <Install wordpress>
- Install Apache JMeter to stress test your wordpress installation.
  Ref: https://www.digitalocean.com/community/tutorials/how-to-use-apache-jmeter-to-perform-load-testing-on-a-web-server
  **<<<DO NOT STRESS TEST ANY PRODUCTION SERVER!!!! >>>**
  Compare the latency:  100 users in 1 second & 10K users in 1 second.
- Post your result on Zuvio.