**Topic 1: Secure Web Communication**

**Public key encryption / Cryptographic hash function / Message digests / Digital Certificates**

1. RSA: public key cryptosystem
2. Cryptographic Hash Function: a function that maps a domain of values into a range of numbers
   1. Two well known cryptographic hash functions: SHA, MD5
3. A message digest is the number produced by applying a cryptographic function to a message
4. Digital signatures: public key cryptography + message digests
5. SSL (Secure Sockets Layer) is a protocol for establishing an encrypted link between client and server, that uses authentication and encryption of transactional data (https, port 443)
   1. Authentication: digital certificates
   2. Privacy: encryption
   3. Message integrity: message digests

**Topic 2: Web Server Performance**

1. Estimating server performance requirements:
   1. connections per second
   2. bytes transferred
   3. how much of the link the web server is allowed to use
2. A web server farm consists of multiple server machines and load balancing hardware that distributes web requests across the server
3. Benchmarking for testing web server performance:
   1. Number of requests that can be served per second
   2. Latency response time in milliseconds for each new connection or request
   3. Throughput in bytes per second
4. 3 typical Web Server: Apache, Lighttpd, Nginx
   1. Lighttpd: optimized for speed-critical environments
   2. Nginx: strong focus on high concurrency, performance and low memory usage
5. Improving Apache Web Server performance:
   1. additional RAM
   2. load only the required modules
   3. set AllowOverride to None
   4. do not set MaxClients too low or too high
   5. tune MinSpareServers and MaxSpareServers
   6. enable HTTP compression
   7. use two versions of apache
   8. use mod\_fastcgi module
   9. use “direct\_modules”: mod\_perl, mod\_php, mod\_python, etc

**Topic 3: Web Server as a Proxy Server**

1. Proxy Server is an intermediary server that accepts requests from clients and either forwards them or services the request from its own cache
2. Why use a Proxy Server:
   1. to prevent access to restricted sites
   2. to control access to restricted sites
   3. to enhance security by controlling which application level protocols are permitted
   4. to improve performance by maintaining a cache
   5. to modify content before delivery to the client
   6. to act as an anonymizer, removing identifying information from HTTP messages
3. Caching and revalidation refer to the process of storing copies of documents retrieved by the proxy server to local storage (disk) or memory so it is readily available
   1. Pro: caching improves performance, reduces latency, saves network bandwidth
   2. Con: the risk of receiving stale data
4. Conditional GET: retrieve a document based upon whether it has been modified since the last access
   1. Using conditional GET for all requests guarantees that stale data is never returned
   2. but is very wasteful as most requests will return “304 not modified”
   3. So, instead, servers use heuristics
5. Validation:
   1. The most common validator: Last-Modified time
   2. HTTP1.1 introduced a new validator: ETag
      1. ETags are unique identifiers that are generated by the server and changed every time the object does
6. A cached object is considered fresh if:
   1. It has an expiry time or other age-controlling directive set, and is still within the fresh period
   2. A browser cache has already seen the object, and has been set to check once a session
   3. A proxy cache has seen the object recently, and it was modified relatively long ago
7. Cache Control Response Header: “cache-control = xxxxxx”
   1. public: responses from this server may be stored without restriction
   2. max-age=3600: the document should be considered stale after 3600 seconds
   3. must revalidate: the caching service must revalidate the document after it becomes stale from the originating server, or report an error
   4. no-cache: responses can be cached, but the cached copy may not be reused for subsequent requests without revalidating the cached copy with the originating server
   5. no store: this response may not be cached on nonvolatile storage

**Topic 4: Cookies and Privacy**

1. Cookies are short pieces of text generated during web activity and stored in the user’s machine by the user’s browser for future reference
   1. Elements: name, value, path, expiration date
   2. What they can do:
      1. store and manipulate any information you explicitly provide to a site
      2. track your interaction with the site such as pages visited, time of visits, number of visits
      3. use any information available to the web server including: your IP address, Operating System, Browser Type
   3. What they cannot do:
      1. automatic access to personal information
      2. read or write data to disk
      3. read or write information in cookies placed by other sites
      4. run programs on your computer
      5. carry viruses, install malware on the host computer
2. Cookie types and taxonomy
   1. By Lifespan:
      1. Session Cookies
      2. Persistent Cookies
   2. By Read-Write Mechanism:
      1. Server-Side Cookies
      2. Client-Side Cookies
   3. Structure:
      1. Simple Cookies
      2. Array Cookies
   4. others:
      1. Secure cookies
      2. Third-party cookies
3. Define: Session cookie
   1. A cookie that is erased when the user closes the Web browser. The session cookie is stored in temporary memory (RAM) and is not retained after the browser is closed.
4. Define: Persistent cookie
   1. A cookie that is stored on the user’s hard drive (disk) until it expires (persistent cookies are set with expiration dates) or until the user deletes the cookie.
5. Define: Third party cookie
   1. A cookie set with a different domain (or subdomain) than the one in the browser’s address bar.
6. Server-side Cookie example in PHP
   1. Set cookie:
      1. setcookie(“name”, “value”, time() + 3600);
   2. View cookie:
      1. if(isset($\_COOKIE[“name”]))
      2. $\_COOKIE[“name”]
7. Client-side Cookie example in JS
   1. Cookies can be accessed on the client side using the “document.cookie” object in JavaScript
   2. Create cookie: “name=value; expires=date”
      1. setCookie(name, value, expireDate)
         * expireDate.toGMTString()
         * escape(name), escape(value)
   3. Retrieving cookie:
      1. getCookie(name)
         * unescape(name)
         * posit, last
         * indexOf, substring
         * return unescape(document.cookie.substring(posit, last))
   4. Removing cookie:
      1. removeCookie(name)
         * creates an early date, attaches it to the expires directive and assigns the name to the null string: var date = new Date(90,1,1);
8. 6 ways to Opt Out of Cookies
   1. Select “do not track” in the browser settings
   2. Download opt-out cookies
   3. Use the cookie management tools in the web browser
   4. View current cookies and delete what you don’t need
   5. Check your account preference on registration sites
   6. User browser Add-ons

**Topic 5: Web Security**

**Authentication Attacks**

* Brute Force Attacks
* Insufficient Authentication
* Weak Password Recovery Validation

**Authorization Attacks**

* Session Hijacking (Credential / Session Prediction)
* CSRF (Cross-site request forgery)

**Client Side Attacks**

* Cross-site Scripting (XSS)
* Browser and Plugin Vulnerabilities
* Clickjacking

**Injection Attacks**

* SQL Injection
* JavaScript Hijacking

**Recent Attacks**

* Worms
* E-mail Hacks: DKIM
* Account Breaches

**Privacy Tools**

* TOR
* PGP & S/MIME
* Passphrases

1. Define: Brute Force Attacks
   1. An automated process of trial and error used to guess a person’s username, password, session ids, credit-card, cryptographic key or anything that is unique to the user and authenticates him
   2. Two types: normal / reverse
   3. Target: websites do not implement any form of account lockout policy
   4. Solutions:
      1. limit the amount of unsuccessful logins to a small number and then lock out the account
      2. block IP addresses where consecutive trial and errors come to quick for a human typist
2. Define: Weak Password Recovery Validation
   1. Weak Password Recovery Validation is when a web site permits an attacker to illegally obtain, change or recover another user’s credentials
   2. Target: when a hacker can easily foil the recovery mechanism by easily guessing the answers to the secret questions
      1. Example of weak recovery mechanism:
         * Information Verification: Asking the user to supply their email address along with their phone number. Note that these are both publicly available
         * Password Hints: Many users have a tendency to embed the password in the hint itself
         * Secret Question + Answer: Something like “In which city were you born?” for a password recovery system is easily circumventable today because most of the information is public due to social networking sites
3. Define: Cross Site Request Forgery (CSRF)
   1. CSRF is an attack which forces an end user to execute unwanted actions on a web application in which he/she is currently authenticated
4. Define: Cross-site Scripting (XSS)
   1. Due to breaches of browser security, XSS enables attackers to inject client-side script into Web pages viewed by other users. A cross-site scripting vulnerability may be used by attackers to bypass access controls such as the same origin policy
   2. Two types:
      1. non-persistent
      2. persistent
   3. Solutions:
      1. Contextual output encoding/escaping: use escaping schemes like HTML entity encoding, JavaScript escaping, CSS escaping and URL (or percent) encoding
      2. Tie session cookies to the IP address of the user who originally logged in and only permit that IP to use that cookie
      3. Use HttpOnly flag which allows a web server to set a cookie that is unavailable to client-side scripts
5. Define: Browser and Plugin Vulnerabilities
   1. Vulnerabilities in the client browser software or client plugins that can either enable other attacks, can enable execution of arbitrary code, raise privileges, compromise user’s privacy or simply crash the browser
6. Define: Clickjacking
   1. Clickjacking is a method used by malicious individuals to trick users into clicking something without knowing what they’ve clicked
7. Define: Injection Attacks
   1. Injection Attacks occurs when an application does not properly validate user supplied input and then includes that input blindly in further processing
8. Define: SQL Injection
   1. SQL injection is where malicious SQL statements are inserted into an entry filed for execution. The SQL statements exploit a security vulnerability in an application’s software, for example, when user input is either incorrectly filtered or user input is not strongly typed
9. Define: JavaScript Hijacking
   1. allows an unauthorized attacker to read confidential data from a vulnerable application using a technique similar to the one commonly used to create mashups
   2. JSON array is directly vulnerable to JavaScript Hijacking
   3. allow the attacker to bypass the Same-Origin policy
10. AJAX Security Holes
    1. Increased attack surface
    2. Information Leakage
    3. Repudiation of Requests
    4. Amplifies XSS
    5. Untrusted Information Sources
11. Bypassing the Same-Origin policy
    1. JSON and the dynamic SCRIPT tag (JSONP)
    2. Ajax Proxy
    3. Browser Extensions and Plugins
12. Worms: Stuxnet
13. E-mail Hacks: Domain Key Identified Mail (DKIM) Spoofing
14. Account Breaches:
    1. Linkedin breach: SHA-I
    2. Yahoo breach: “union-based SQLK injection” attack
    3. Heartbleed Bug: OpenSSL
15. Privacy Tools
    1. TOR: The TOR network is a protective layer that sits between the user and the Internet. It provides an anonymous path between the user and the sites the user visit
    2. PGP (Pretty Good Privacy) & S/MIME:
       1. PGP is data encryption technology commonly used for encrypting files, especially e-mails
       2. S/MIME is a standard for public encryption and signing of MIME data in e-mails
    3. Passphrases:
       1. Diceware

**Topic 6: Responsive Web Design**

1. 3 approaches to design for the mobile web
   1. Build an entirely separate .mobi mobile site
   2. Host the mobile site within your current domain (m.mycompany.com)
   3. Configure your current website for mobile display using RWD technologies
2. Reasons why hosting a separate .mobi website is not recommended:
   1. Require duplication of content
   2. Over time can result in synchronization of content issues
   3. May work only on a specific mobile size
3. Reasons why hosting a mobile website within your current domain (m.mycompany.com) is not recommended:
   1. Redirect take time
   2. Redirect can hinder/annoy search engines
   3. May work only on a specific mobile size
4. 3 Major Technology Features of RWD:
   1. fluid, proportion-based grids
   2. flexible images
   3. CSS3 media queries
5. Fluid Grids:
   1. fluid grids: relative-based dimensions VS. adaptive grids: pixel-based dimensions
   2. In fluid grids:
      1. defined using relative-based dimensions
      2. define a grid divided into a specific number of columns
      3. each grid element is designed with “proportional” width and height, not in pixels
6. Flexible images:
   1. To avoid having an image deformed due to the screen size, one should avoid specific definitions of width and height, and instead use CSS’s max-width property setting to 100%
   2. The browser will resize the images as needed using CSS to guide their relative size
7. CSS3 media queries:
   1. <link rel=”stylesheet” type=”text/css” media=”screen and (max-device-width: 480px)” href=”min.css” />
8. Usability Guidelines for Websites on Mobile Devices:
   1. Reduce the amount of content
   2. Single column layouts work best
   3. Present the navigation differently
   4. Minimize text entry
   5. Decide how many mobile sites you need
   6. Design for touchscreen and non-touchscreen users
   7. Take advantage of smartphone built-in functionality

**Topic 7: High Performance Websites**

Rule 1: Make fewer HTTP requests

1. To reduce the number of HTTP requests:
   1. Combine scripts
   2. Combine style sheets
   3. Combine images into an image map
   4. Combine images using CSS sprites
2. Drawbacks:
   1. Images must be contiguous
   2. Defining area coordinates is error prone

Rule 2: Use a CDN (Content Distribution Network)

1. It improves website performance by copying content to multiple servers around the world so pages are delivered more quickly

Rule 3: Add an Expires Header

1. When “cache-control: max-age” is present, the response is stale if its current age is greater than the age value given at the time of a new request for the resource

Rule 4: Gzip Components

* can be GZIP-ed: HTML, CSS, JS, JSON, XML, text
* cannot be GZIP-ed: images, PDF

Rule 5: Put stylesheets at the top

1. Reason: stylesheets block rendering in IE, it will examine all stylesheets before starting to render

Rule 6: Move scripts to the bottom

1. Reason: loading of JavaScript can cause the browser to stop rendering the page until the JavaScript is fully loaded

Rule 7: Avoid CSS expressions

1. Reason: expression may execute many times

Rule 8: Make JS and CSS external

1. Reason: external JS and CSS can be cached

Rule 9: Reduce DNS lookups

1. Reason: DNS lookups block parallel downloads

Rule 10: Minify JS

1. Reason: reduce the amount of data that must be transferred

Rule 11: Avoid redirects

1. Reason: redirects insert an extra HTTP round-trip between user and origin server

Rule 12: Remove duplicate scripts

1. Reason: hurts performance because it will have extra HTTP requests and extra executions

Rule 13: Configure Etags

Rule 14: Make AJAX cacheable

New Rules:

1. Avoid empty src or href
   1. Reason: corrupt user data, waste server computing cycles
2. Use GET for AJAX requests
   1. Reason: GET sends the headers and the data together, while POST sends the headers and the data separately
3. Reduce the number of DOM elements
4. Avoid HTTP 404 (not found) error
5. Reduce cookie size
6. Use cookie-free domains
7. Do not scale images in HTML
8. Make favicon small and cacheable

**Topic 8: HTML5**

1. Major New Elements:
   1. New semantic elements: <header>, <footer>, <article>, <section>
   2. <video> and <audio>
   3. Canvas
   4. Scalable Vector Graphics (SVG)
   5. Geolocation
   6. Persistent local storage
   7. Offline web applications
   8. Improvements to HTML web forms
   9. Microdata
2. HTML removed elements:
   1. because their effect is purely presentational and therefore better handled by CSS:
      1. basefont, big, center, font, s, strike, tt, u
   2. because their usage affected usability and accessibility for the end user in a negative way:
      1. frame, frameset, noframes
   3. because they have not been used often, created confusion or can be handled by other elements
      1. acronym, applet, isindex, dir
3. HTML removed attributes
4. <canvas> element: (.fillStyle = …; .fillRect(10, 10, 190, 190); .moveTo(.); lineTo(.); .stroke(); . createLinearGradient(0, 0, 100, 200); .addColorStop(0, “red”); )
   1. resolution-dependent bitmap
   2. no support for event handlers
   3. poor text rendering capabilities
   4. can be saved as .png or .jpg
   5. well suited for graphic-intensive games
5. <svg> element:
   1. resolution-independent
   2. support for event handlers
   3. best suited for applications with large rendering areas (Google Maps)
   4. slow rendering if complex
   5. not suited for game applications
6. Document Structure: <header>, <footer>, <article>, <section>, <nav>, <aside>
7. Video on the Web:
   1. <video> tag
   2. video containers:
      1. MPEG4 compressed video (mp4 or m4v)
      2. QuickTime (.mov)
      3. Flash Video (.flv)
      4. Ogg (.ogv)
      5. WebM (VP8/VP9 + Vorbis)
      6. Audio Video Interleave (.avi)
   3. video codecs:
      1. MPEG-4
      2. H.264/H.265
      3. Theora
      4. WebM(VP8/VP9)
      5. Sorenson Spark
   4. <video> attributes:
      1. preload: start downloading the video file as soon as the page loads
      2. autoplay: start downloading the video file as soon as the page loads, and start playing the video automatically as soon as possible
8. Audio on the Web:
   1. <audio>
   2. audio codecs:
      1. MP3 (contain up to 2 channels of sound)
      2. AAC and AAC+ (support up to 48 channels of sound)
      3. Vorbis
9. Local Storage
   1. a client-side key-value database
   2. per browser not per computer
   3. only support storing of strings
10. Session Storage

**Topic 9: Ajax**

1. It’s not a technology but several technologies: HTML, CSS, JS, XML, JSON, XHTML, XSLT, DOM, Web Remoting (XMLHttpRequest)
2. Characteristics of Ajax applications:
   1. they are applications, not just web sites
   2. allow for smooth, continuous interaction
   3. “Live” content
   4. visual effects
   5. animations, dynamic icons
   6. single keystrokes can lead to server calls
   7. new widgets (selectors, buttons, tabs, lists)
   8. new styles of interactions (drag-and-drop, keyboard shortcuts, double-click)
3. The Ajax Web Application Model – Ajax Engine
   1. an intermediary between the user and the server
   2. responsible for:
      1. rendering the interface the user sees
      2. communicating with the server on the user’s behalf
4. JavaScript Ajax + XML: XMLHttpRequest object ( 5 examples)
5. Security issues:
   1. same-domain security policy: scripts are only allowed to access data which comes from the same domain (protocol + hostname + port)
   2. solutions:
      1. install a web proxy on the server
      2. Corss-origin resource sharing (CORS)

**Topic 10: jQuery**

1. jQuery simplifies:
   1. HTML document traversing
   2. Event Handling
   3. Animating
   4. AJAX interactions
2. Basic selectors:
   1. All selector (“\*”)
   2. Class selector (“.class”)
   3. Element selector (“element”)
   4. ID selector (“#id”)
   5. Multiple selector (“selector1, selector2, selectorN”)
   6. Others:
      1. Attribute: (“input[value=’Hot Fuzz’]”)
      2. Basic Filter: (“:header”)
      3. Child Filter: (“div span:first-child”)
      4. Content Filter: (“div:contains(‘John’)”)
3. 7 jQuery examples
   1. document.getElementById(id).style.color => $(“#id”).
   2. document.getElementsByTagName => $(“element”)
   3. document.getElementById(id).innerHTML(string) => $(“#id”).html(string)
   4. document.getElementById(id).style.left = value => $(“#id”).css(‘left’, ‘value’)
   5. int = parseInt(string)
4. jQuery AJAX
   1. $.func(url[, params][, callback]);
   2. $(selector).load(URL, data, callback);
   3. 4 AJAX examples
5. jQuery usage examples

**Topic 11: JSON**

Define: JSONP

JSONP or “JSON with padding” is a JSON extension wherein the name of a callback function is specified as an input argument of the call itself

JSON properties

1. Subset of ECMA-232 Third Edition
2. Language independent
3. text-based
4. light-weight
5. Easy to parse
6. Not a document format
7. Not a markup language
8. Not a serialization format

**Topic 12: Web Services and REST**

Define: REST

REST stands for Representational State Transfer. It is a style of software architecture for distributed hypermedia systems like WWW. Quickly gained popularity through its simplicity.

3 fundamental aspects of the REST design pattern:

1. client
2. servers
3. resources
   1. Every resource is uniquely identified by a URL
   2. Resources are typically represented as documents

4 simple operations of REST: PUT, GET, POST, DELETE

Characteristics: Platform-independent; Language-independent; Standards-based; used in firewalls; no built-in security features; No cookies;

//my own:

JSON.parse(); e.g.

JSON.parse('{"p": 5}', (key, value) =>

typeof value === 'number'

? value \* 2 // return value \* 2 for numbers

: value // return everything else unchanged

);

// { p: 10 }

JSON.parse('{"1": 1, "2": 2, "3": {"4": 4, "5": {"6": 6}}}', (key, value) => {

console.log(key); // log the current property name, the last is "".

return value; // return the unchanged property value.

});

// 1

// 2

// 4

// 6

// 5

// 3

// ""

JSON.stringify();

$(“div, span, p.myClass”).css(“border”, “3px solid red”);

$(“#content”).append(“<p>…</p>”);

.css(“border”, “1px solid red”);

$(“p: first).css(“border”, “3px solid red”);

$(“h2:not(.selectors)”).css(“border”, “3px solid red”);

Var newP = $(“<p>”);

newP.append(“<em>Hello</em>”);

$(“#example”).html(newP);

.prepend

.on – start listening

$.ajax("sampletextcontent.txt",

{ success: setContent,

type: "GET",

dataType: "text" });

$(“#example”).load(“samplehtml.html”);

$(‘#counter2’).css(‘top’, ‘15px’);

var onReady = function() {

$(".reverse").on("click", function() {

var kids = $("body").children();

var kidLength = kids.length;

for(var i = kidLength - 1; i >= 0; i--) {

var c = $(kids[i]).remove();

$("body").append(c);

}

onReady();

});

}

$(onReady);

setInnerHTML(this.options[this.selectedIndex].value, form.t.value

$(document).ready(function(){

$("button").click(function(){

$("#div1").load("demo\_test.txt #p1");

});

});

Prevent an event: .preventDefault();

e.g.:

$(‘#menu’).click(function(evt) {

//js

Evt.preventDefault();

}

$(‘:text’) --- select all text fields;

Determine if a checkbox is checked:

If($(‘#aa’).attr(‘checked’)) {

…}

$(window).width();

API:

* + - 1. bolt-on approach;
      2. Greenfield Strategy(start from scratch);
      3. Façade Strategy(between)

3 simple rules:

Don’t worry about tools, have a consistent process, It doesn’t count unless it’s written down.

Bootstrap cdn:

https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css

<https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js>

XMLHttpRequest: open(), setRequestHeader(), getResponseHeader(), getAllResponseHeaders();

CORS: Cross-origin resource sharing (CORS);

SOAP: simple object Access Protocol;

Initial 14 rules:

1. Make fewer HTTP requests
2. Use a CDN (content distribution network)
3. Add an Expires header
4. Gzip components
5. Put stylesheets at the top
6. Move scripts to the bottom

…

New rules

1. **Avoid empty src or href**
2. **Use GET for AJAX requests(headers and data together)**
3. **Reduce the number of DOM elements**
4. **Avoid HTTP 404 (Not Found) error**
5. **Reduce cookie size**
6. **Do not scale images in HTML**
7. Make favicon small and cacheable

**Usability Guidelines for Websites on Mobile Devices**

1. Reduce the amount of content
2. Single column layouts work best
3. Present the navigation differently (in a menu, dropdown, or at the bottom)
4. Minimize text entry
   * request PIN instead of password
5. Decide how many mobile sites you need
   * consider users with slower connections and possibly make a lower resolution site
6. Design for touchscreen and non-touchscreen users
7. Take advantage of smartphone built-in functionality

var today = new Date();

var expiry = new Date(today.getTime() + 30 \* 24 \* 3600 \* 1000); // plus 30 days

function setCookie(name, value, expiry)

{

document.cookie=name + "=" + escape(value) + "; path=/; expires=" + expiry.toGMTString();

}

new RegExp(name + “=([^;]+)”);