Web Development Cheat Sheet

by SalJose24 via cheatography.com/27693/cs/8372/

Web App Security Browser Isolation

Modes of attacks on web applications

● Attack the connection between browser and web server

○ Steal password

○ Hijack existing connection

● Attack the server

○ Inject code that does bad things

● Attack the browser

○ Inject code that does bad things

● Breach the browser, attack the client machine

● Fool the user (phishing)

Security Defenses

● Isolation in browsers

○ Web app run in isolated sandbox

● Cryptography

Same-Origin Policy

● General idea: separate content with different trust levels into different frames, restrict communication between frames

● One frame can access content in another frame only if they both came from the same origin

● Origin is

○ Protocol

○ Domain name ○ Port

● Access applies to DOM resource, cookies, XMLHttpRequest/AJAX requests

● Doesn't apply: <script> tags

○ Javascript executes with full privileges of the enclosing frame.

○ Protect information from unauthorized viewing

○ Detect changes

○ Determine origin of information

● Web development frameworks

○ Use patterns that help, avoid dangerous ones

same-origin policy is too restrictive

● There are times when it is useful for frames with different origins to
communicate

○ Example: Sub-domains of same organization

○ Web fonts

○ Content distribution network

HTML5 feature: Access-Control-Allow-Origin

Web App Security Browser Isolation (cont)

● Access-Control-Allow-Origin header in HTTP response:

Access-Control-Allow-Origin: http://foo.com

Access-Control-Allow-Methods: PUT, DELETE

HTML5 postMessage - safe messaging

● Sender (from domain a.com) to an embedded frame of different domain

Criticisms

Angular criticisms

Digest cycle overheads on pages with large numbers of items,Consider the watches on a large data table with multiple columns,HTML template with two-way binding,DOM access overhead,Access to the browser DOM is slow,Large size of JavaScript,Needs to download, initialize, and digest before anything appears,Problematic on mobile, Software engineering problems programming at scale, Scope inheritance, JavaScript lack of typing, interface definitions,

Node

Callback hell - TJ Holowaychuk's why Node sucks:

1. you may get duplicate callbacks
2. you may not get a callback at all (lost in limbo)
3. you may get out-of-band errors
4. emitters may get multiple “error” events
5. missing “error” events sends everything to hell
6. often unsure what requires “error” handlers
7. “error” handlers are very verbose
8. callbacks suck

JavaScript lack of typing checking

Concurrency support (e.g. crypto operations)

Mongo

Lots - Pretty lame database

○ Loses data, doesn't scale well

○ Large space overheads for objects and indexes

○ Query language

○ Limited concurrency control (only single object transactions)

○ Not SQL?

Network Security

"man in the middle" attacks

● Attacker has access to network communication between browser and server.

● Passive attacks:

○ Eavesdrop on network traffic

● Active attacks:

○ Inject network packets

○ Modify packets

○ Reorder, replay packets

○ Block packets

Certificate authority: well-known, trusted server that certifies public keys. Certificate: a document encrypted with the secret key of a certificate authority

○ Identifies a particular service along with its public key

Certificate authorities establish selfs as well known services on Internet

○ Browsers hard-wired to accept certificates from dozens of authorities

● Internet services compute keys, gives the public key to a certificate authority along with proof of identity

● Certificate authority returns a certificate for that service.

● Service can pass along this certificate to browsers

○ Browser can validate the certificate came from the certification authority and see who the certification authority thinks the browser is talking to.

● Trust: Browser trusts to certification authority

Secure Sockets Layer (SSL) & Transport Layer Security (TLS) - HTTPS

● Protocol used for secure communication between browsers and servers

Excuses for not using HTTPS for all Web traffic?

● Expensive: slows down Web servers

● Breaks web page caching

Problem: SSL

When server returns pages with HTTPS links, attacker changes them to HTTP.

Network Security (cont)

● When browser follows those links, attacker intercepts requests, creates its
own HTTPS connection to server, and forwards requests via that
Problem: Mixed content
● Main page loaded with HTTPS, but some internal content loaded via HTTP (e.
g. <script src="http://.../script.js">). ○ Network attacker can modify content to attack page.

Session Attacks

get user's session id - session ids must be unpredictable use framework
Use Https to protect cookies - change session id after logging in

Cross Site Forgery Request (CSFR)

visiting multiple pages (attacker's website) - mark pages, Don't accept POST submission directly from forms, HTTP GET should not have side effects, have JavaScript include special HTTP request header property with secret

Data Tampering

Message Authentication Codes (MAC) - MAC function takes arbitrary-length text, secret key, produces a MAC that provides a unique signature for the text, Server includes MAC with data sent to the browser, Browser must return both MAC and data, check for tampering need: Authentication - Know that we (the web server) authored the information Integrity - Known that it wasn't tampered with, Need encryption if we want confidentiality

Code Injection Attacks

What happens if someone inputs a comment with a script tag?
<script src="http://www.evil.com/damage.js" />, Called a **Cross Site Scripting Attack (XSS)**
Stored Cross Site Scripting Attack
● Attacker stores attacking code in a victim Web server, where it gets accessed by victim clients
Reflected Cross Site Scripting
● Attacker doesn't need to store attack on website, can just reflect it of the website
Modern JavaScript frameworks have better defences

Code Injection Attacks (cont)

● Angular bind-html - Sanitizes HTML to remove script, etc.
SQL Injection - Don't write SQL

Phishing Attacks

Fool people to disclose personal info
Problem: too easy to obtain certificates
Counter-measure: extended validation certificates - vet the organization, warn users
Spear phishing - Phishing with attacker having personal information

DoS Attacks

An attack that causes a service to fail by using up resources
None perfect - really hard problem
Do want to take steps to avoid accidental DOS and make purpose-driven DOS harder, Resource quotas, Track resource consumption per user and provide way of cutting off users, Good for catching accidents, less so for malicious attacks, Make resources cost money, Raises the cost or hassle for an attacker, Not always possible under business model, Network layer: Need to push back on attack stream

LargeScalableWebApps

scale-out architecture - make more instances, Benefits of scale-out
Can scale to fit needs: Just add or remove instances
Natural redundancy make tolerating failures easier: One instance dies others keep working
DNS (Domain Name System) load balancing:
○ Specify multiple targets for a given name
○ Handles geographically distributed system
○ DNS servers rotate among those targets
Load-balancing switch ("Layer 4-7 Switch")
● Special load balancer network switch
○ Incoming packets pass through load balancer switch between Internet and web servers
○ Load balancer directs TCP connection request to one of the many web servers
○ Load balancer will send all packets for that connection to the same server.

LargeScalableWebApps (cont)

Data sharding - Spread database over scale-out instances
○ Each piece is called data shard
○ Can tolerate failures by replication - place more than one copy of data (3 is common)
Memcache: main-memory caching system
Many useful services available:
○ Auto scaling (spinning up and down instances on load changes)
○ Geographic distribution (can have parts of the backend in different parts of the world)
○ Monitoring and reporting (what parts of web app is being used, etc.)
○ Fault handling (monitoring and mapping out failed services
Content Distribution Network (CDN)
● Consider a read-only part of our web app (e.g. image, html template, etc.)