Web Application Security

- **Logical Flaws**
  - Password Change/Forgotten Functions
  - Brute Forcible Login
  - Verbose Failure Messages
  - Insecure Credential Distribution
  - Predictable Usernames/Initial Passwords
  - Predictable Account Activation Link/ID
  - Bad Passwords
  - Fail-Open Login
  - defective multi-stage login
- **Same Origin Policy**
  - only allow to read data originating from the same source
    - sources are the same if the following matches
      - protocol
      - hostname
      - port (not for IE?)
    - enforced by utilizing domain names
      - only the exact same domain can be accessed
        - explicit exceptions can be made
          - flash has xml policy files that define exceptions called /crossdomain.xml
          - XMLHttpRequest
            - new Header called Access-Control
            - in xml <?access-control processing instruction
          - if want access to a subdomain both pages must explicitly set the document.domain to the same value
            - problem
- **Sessions**
  - attacks
    - Session Hijacking
      - CSRF
        - simple attack huge impact on web app security
        - perform actions in users behalf via users browser
          - possible because browser implicitly appends Cookies to every request for the allowed domain(s)
        - doesn't obey to same-origin policy
          - cannot read from other domain, but influence it
        - XSS and CSRF can let unsuspecting user attack a webserver on attackers behalf
        - counter measures
          - incorporate nonces into state-changing requests
            - XSS vulnerability would bypasses this
              - nonce could be read by JS
              - JS enables to act with web site like the user
      - Session Fixation
        - webapp accepts provided tokens as new sessions
        - webapp doesn't create new token on login
        - attacker can smuggle new session token into users browser and continue his session after login
      - weak Session IDs
        - predictable tokens
          - concealed sequences
            - reproducible algorithm enabling brute-forcing
          - time dependency
            - token-generation process depends only/mostly on time
              - the keyspace can be minimized the better the timeframe of generation is known
          - weak (P)RNG
        - guessable meaningful tokens
        - static tokens
          - user doesn't need to log in
            - tokens holds authorization data
    - disclosure of tokens on network
      - fallback to http after login
      - http for all static content
        - usage of https only cookies
      - tricked by attacker to make http request to server, not matter if successful, token will be disclosed
    - vulnerable session termination
      - logout doesn't invalidate session
      - no session expiration
        - user didn't log out
        - possibly user got force quit
    - liberal cookie scope
      - domain
        - cookies are default sent in every request web-servers sharing the same right-hand-side of the domain
          - so in default the cookie is sent in all requests to subdomains and the domain itself
        - domain value can be changed upto one below the tld
          - enabling also other webservers on a subdomain to read the cookie from users
      - path
        - similar to domain path defines to which subdirectories on the server cookies will be passed
          - too liberal path value again might enable other users on the system to obtain cookies from users of that application
  - counter measures
    - creating additional page tokens
      - pro
        - can be leveraged to track user movement on page
      - con
        - opening several tabs at the same time will be interpreted as an attack
    - reactive session termination
      - deauthenticating user when suspicious activity is noticed like modified hidden form fields or strings associated with SQLi or XSS
  - technologies
    - session management needed to identify different users across requests
      - session management closely related to authentication mechanism
      - http is stateless
        - webapplications becoming like desktop applications -> need state
    - http authentication
      - main technology Cookies
      - basic
      - digest
      - NTLM
      - (Kerberos)
      - client certificates
    - encoded session state being pushed to client
      - in cookie
      - in hidden form fields
- **Client Side**
  - XSS
    - 3 types
      - reflected XSS
        - problem
          - client-supplied data gets echoed back to the screen
          - most common via search box
      - DOM based XSS
        - unique form of XSS
          - JS payload doesn't need to be sent or echoed by web server
        - similar to reflected XSS
          - can be persistent if incorporates cookies
        - DOM (Document Object Model)
          - object structure to access & manipulate nodes, values and attributes
        - input validation vulnerability on the client side JS
          - completely relies on JS and insecure use of dynamically obtained data from DOM structure
      - stored XSS
    - resulting from missing/bad input/input handling
      - user is able to define content which gets evaluated as javascript or create own javascript blocks
      - developer doesn't apply needed sanitization for given context of user supplied input
    - technique to force web sites to display malicious code which gets executed in users web browser
      - browser doesn't have to be susceptible to any vulnerability
    - user supplied data doesn't only reach the server via GET and POST
      - all headers may be spoofed by user including cookie, referer and host
        - logging infrastructure must implement proper escaping for everything logged
          - the target is usually the admin, which will gain the attacker admin rights
    - encoding can pose a problem
      - variable width encoding can make escaping invalid
      - server can't correctly check and filter the encoding the browser uses to interpret the page is a different
    - possible attack methods
      - history stealing
        - getComputedStyle API
          - visited style
      - intranet hacking
      - defacement
        - JS can completely alter look of web site
        - can be leveraged for phishers to occur on the real website
      - abuse users browser to run applications without user approval
        - as filesharing node
        - as part of anonymity network
  - Flash Security
    - compiled bytecode
      - unaware developer might include secrets into code
        - those can be revealed by decompilers and reversing
    - cross domain requests
      - each exact domain gets an own sandbox
        - crossdomain.xml for cross sandbox communication
    - sandboxing
  - Java Security
  - DNS Pinning Vulnerabilities
    - attack without dns pinning
      - browser connects to malicious server with dns timeout of 1 second
      - JS tells browser to connect back in 2 seconds
      - browser resolves name again since no longer valid
      - dns server replies with arbitrary ip address it wants to get the contents from
        - this ip may be in the intranet providing attacker access to the inner network
    - the browser pins once received IPs to domains to prevent the attack above
      - browser protection to prevent attackers from breaking the same-origin policy through DNS tricks
    - anti-dns pinning
      - the browser is re-requesting dns if the web server isn't reachable anymore
      - attack
        - same as above, but this time the base attacker system firewalls itself to be unreachable for the browser in the second request
          - it needs to firewall the port in question not the whole server, so the whole process can be carried out for different IPs on a single server
      - anti-anti-dns pinning
        - checking the host header if the targeted system by the browser is really ours
          - this would already happen if virtual hosting is used, but there usually is a default
        - anti-anti-anti-dns pinning
          - exploits in client side request building mechanisms to spoof the host header defeats this kind of protection
    - it is not possible to hijack session via automated sending of cookies with this, because cookie sending is determined by domain not IP
    - intranet scanning can be combined with an automated search for vulnerabilities in the intranet
      - possibility to take over systems in the intranet
      - use XSS or CSRF exploit to attack users browser to steal information or authentication tokens
  - JavaScript Hijacking
    - request json from other side via <script> include
      - redefine Array object to have the results accessible
      - XMLHttpRequest wouldn't work because of same origin policy
    - user is authenticated on other web service via cookie
    - if web service discloses privacy relevant information over json
- **Server Side**
  - Attacks
    - SQL Injection
      - SQL databases are usually called indirected by a web application, which passes a complete query containing parts of user supplied content to the database subsystem
      - attacks
        - SELECT
          - requests to retrieve information
          - injection point usually the WHERE clause
        - INSERT
          - creates new row
          - attackers usually have to guess database schema and therefore doesn't know the order and number of columns
            - brute-forcing the number with 1s (or 2000s for possible dates) or NULLs
        - UPDATE
          - change one or more data rows
          - usual entry point the WHERE clause and or the SET clause
        - DELETE
          - removes one or more rows
          - usual entry point is WHERE clause
        - UNION operator
          - combine two or more SELECTs
          - can be leveraged to gain information from different tables than original query
          - selected fields must be same number and type (structure)
          - names of tables and fields must be known
            - names can be queried from database specific metadata tables
          - usual steps
            - inject ORDER BY $colnum until error to check for number of columns
              - search for columns of string type by inserting NULL in all but one field, which holds a string until string is displayed
        - Blind SQL injection
        - timing attacks
        - Second Order SQL Injection
          - correctly escaped user content is written to the database
            - user content is fetched from database and used again without further sanitization
          - foo' gets correctly escaped to foo'' as username
            - foo' is fetched from database and used to create update statement for password without futher sanitization
    - Remote File Inclusion
    - Remote Code Execution
      - usually interpreted languages
        - executed code is built at runtime containing user supplied content
        - utilizing special characters to break out of data context therefore being able to execute arbitrary commands
      - in compiled languages attack not leverages syntactic features of programming language
        - payload contains machine code
    - Path Traversal
  - Web Application Firewall
    - ModSecurity (Apache)