Book 1

Core Goals in security:

- Confidentiality:Prevent disclosure of information to unauthorized personnel.
- Integrity:Data should not be modified without authorization
- Availability: data should be available when needed

Web hacking activities:

- Random Hacking :
    - Continuous scanner looking for weakness followed by manual attack
    - Worms, automatic spreading
- Targeted Hacking
    - Focused attack fixating on specific targets
    - Attackers know what they are after

Hacking Steps: 4 step approach hackers take

- Reconnaissance: the attacker gathers as much information about the site as possible using any possible means.
- Mapping: the attacker maps out relationships between d/t hosts within the target or in case of application, the attacker maps out the pages and resources within the site.
- Discovery: attacker attempts various techniques in an attempt to see if any part of the host or site is vulnerable.
- Exploitation: after vulnerabilities are discovered the attacker proceeds to attack.
- Some steps may be skipped in radom hacking

Web malware:

- Websites are commonly compromised to host malware

0-day via web:

- Internet explorer, Adobe Acrobat, Java JRE and Adobe Flash are all commonly attacked
- All these platforms are accessible via the web
- Attackers compromise websites to spread these malwares
- Malwares generate high revenues

Social Networking sites:

- The common problem with this site is that content is submitted by all users who are allowed to customize their portion of the site.
- Your browser assumes you want to render all content on the site you visit
- *Social networking sites Scams:* scamming user with something that looks like from OS(like windows update popup)
- Social networking Worm: attack that tricks users to execute code in the browser. Code changes user's social network profile and spreads the code within the social network

Hacktivism via Web attacks:

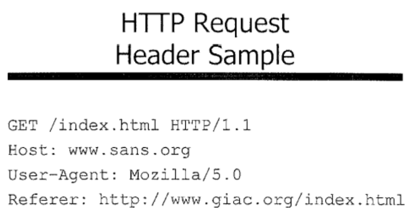
- Hacktivism uses computer hacks to promote political messages
- This groups use mostly SQL injection for their agenda
- Extremely successful
- The main contributing factor is most of the websites and applications on the internet have very poor security levels.
- Common activities:
    - Compromise one site by SQL injection
    - Gather info on the site(user credentials)
    - Use the same credentials on other websites
    - Repeat many times
    - Release on web to cause embarrassment

HTTP Overviews

- HTTP: is the communication protocol enabling the world wide web.
- HTTP 1.1:
- Client Server architecture:
    - Client sends request and servers response
    - Possible to have intermediate points b/n them(proxy, firewall, gateway)
    - Usually works over TCP/IP protocol, TCP(port 80)

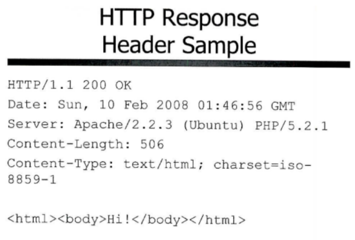
HTTP Communications

- HTTP request: essentially asks for a resource or sends certain input strings to the server.
- HTTP Response: the server then sends back a response.
- Both contain header and body:
    - Header section: has the administrative information about the message.
    - The body contains the content
- Header and body within the same message are separated by a blank line



HTTP Request
Header Sample

```
GET /index.html HTTP/1.1
Host: www.sans.org
User-Agent: Mozilla/5.0
Referer: http://www.giac.org/index.html
```



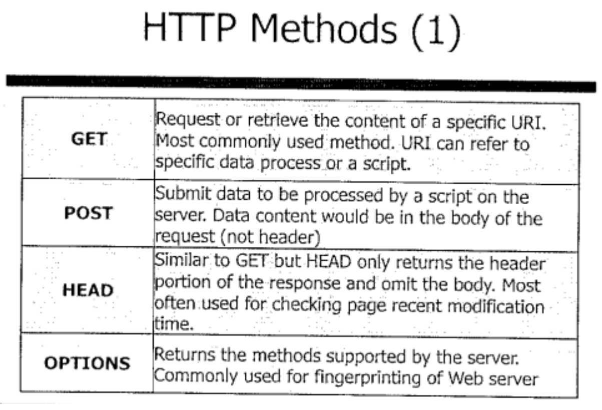
HTTP Response
Header Sample

```
HTTP/1.1 200 OK
Date: Sun, 10 Feb 2008 01:46:56 GMT
Server: Apache/2.2.3 (Ubuntu) PHP/5.2.1
Content-Length: 506
Content-Type: text/html; charset=iso-
8859-1

<html><body>Hi!</body></html>
```

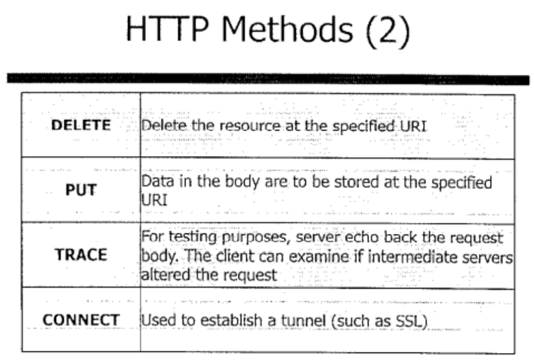
- Referring field in request: lets the server know how the client has been led to this resource.
- Response body content
    - Can contain the html that browser renders on screen
    - Can contain: HTML, MP3, Javascript, JPG, GIF

- MIME(Multipurpose Internet Mail Extension) type lets the browser know hot to handle the data



HTTP Methods (1)

GET	Request or retrieve the content of a specific URI. Most commonly used method. URI can refer to specific data process or a script.
POST	Submit data to be processed by a script on the server. Data content would be in the body of the request (not header)
HEAD	Similar to GET but HEAD only returns the header portion of the response and omit the body. Most often used for checking page recent modification time.
OPTIONS	Returns the methods supported by the server. Commonly used for fingerprinting of Web server

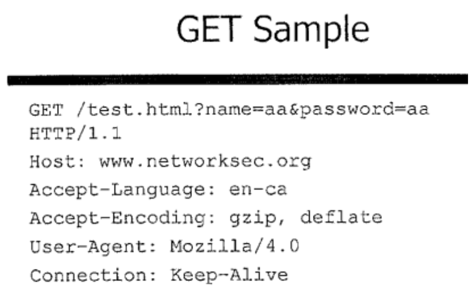


HTTP Methods (2)

DELETE	Delete the resource at the specified URI
PUT	Data in the body are to be stored at the specified URI
TRACE	For testing purposes, server echo back the request body. The client can examine if intermediate servers altered the request
CONNECT	Used to establish a tunnel (such as SSL)

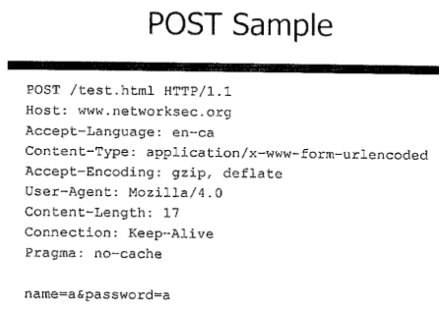
Basic Essential Methods:

- Server must support GET and HEAD
- 3 methods most sites support: GET, POST and HEAD
- Some methods can have a security impact when allowed



GET Sample

```
GET /test.html?name=aa&password=aa
HTTP/1.1
Host: www.networksec.org
Accept-Language: en-ca
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Connection: Keep-Alive
```



POST Sample

```
POST /test.html HTTP/1.1
Host: www.networksec.org
Accept-Language: en-ca
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0
Content-Length: 17
Connection: Keep-Alive
Pragma: no-cache

name=a&password=a
```

- GET vs POST
    - Use POST where possible
    - GET content is right in the url bar
    - Slight security benefit for POST

WebDAV: web-based distributed authoring and versioning.

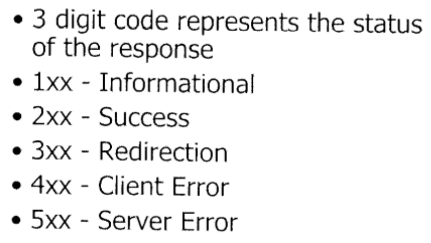
- It's a set of extension to HTTP(added methods: PROPFIND, PROMATCH,MKCOL, COPY, MOVE, LOCK, UNLOCK,)
- It was developed as a way to update websites and perform content management.
- Lets users manage files on web server
- From a security standpoint, an in-channel management functionality is never well regarded! (the management interface is better to be separated from the content)

URL: uniform resource locator

- http://[host][:port]/[path][?query]
- Protocol: for web its http or https(which is http over SSL)

- Port: is the TCP port on the server
- Path: is the location of the resource on the server
- Query: is the GET method parameters which are sent to the server

HTTP Response Status Code



- 3 digit code represents the status of the response
- 1xx - Informational
- 2xx - Success
- 3xx - Redirection
- 4xx - Client Error
- 5xx - Server Error

- Most Common
    - 100: tells the browser to continue the request process and that server has not rejected the partial request yet.
    - 200: the request sent by the user has successfully been processed by the server, and no error was encountered.
    - 302: the server is telling the client the page requested has been temporarily moved to another URL and give the new URL
    - 404: the resource requested is not found
    - 500's series: indicates that the server is unable to process the request.

Referrer(Referer)r: Brower lets web server know the previous page from which a link was followed

- Not a security feature, can be easily forged

User-agent String:

- Part of the header includes info about the browser, version and OS
- Wb sites often use the string to distinguish between bots and humans
- Easily spoofed: so should not be trusted as authentication data.

Compression

- Gzip + deflate is the de facto standard for compressing HTTP content
- Compression saves bandwidth
- Affects how the content looks on wire
- Client sends- Accept-Encoding: gzip
- Server then sends compressed content

Persistent Connections: use one TCP connection for multiple HTTP requests

- Keep-alive is used by client and server

Pipelining: HTTP requests are sent without waiting for responses.

Chunked Encoding:
- Sends data in smaller chunks, best for dynamic content
- Server start sending data to the user as soon it's possible
- content-length header not sent
- Reduces user perceived latency

Web Mechanism Overview

HTTP Basic Authentication:
- It's supported by every HTTP browser and proxy server
- Userna,e and password are sent encoded with base64
    - NOT encryption
    - Can be easily broken(pen and paper)
- If used over SSL passwords are encrypted

HTTP digest Authentication
- Supported by most recent browsers
- Similar to basic authentication
    - It's designed to address security issues in basic authentication
    - Difference: Password are not sent on the wire
- Based on challenge-response based authentication(uses MD5 hashes)
- Elements of the digest WWW-authenticate header:
    - Nonce: unique string of characters that are combined with client data to complete the authentication
    - Opaque: string that the client should return unchanged in the same protection space
    - Stale: is a flag used to report a repeated nonce.
    - The algorithm: defines the hash method to be used
    - QQP: optionally used field which defines the quality of protection provided.
- The client response repeats all the server entries and adds few, such as:
    - Username to authenticate
    - NC(nonce-count): incremented by 1 to prevent replays with each HTTP request/reply pai.
    - Response is the actual authentication value(hash of combined username:realm:password:nonce)
    - Cnonce can be used if server is expected to authenticate the client in which case the hash value is username:realm:password:nonce:cnonce

Certificate Authentication
- One of the most secure ways to authenticate a user.
- Works with most of the current browsers

- Both client and server won certificates signed by CA(Certificate Authority)
- Two way authentication(both parties authenticate each other)
    - Client verifies the server's identity
    - Server verifies the client's identity
- Cryptography is strong: also protects the http data
- Transparent to the user after the certificate is set up

Integrated Windows Authentication
- Microsoft's proprietary technology: windows server and clients only
- Uses kerberos or NTLM authentication in the OS
- Does not work through most HTTP proxies
- Kerberos is strong, and NTLM is weaker than Digest
- Can be transparent to the user
    - When properly implemented in windows network
    - When both server and client are in the same domain or trusted domain

Form based authentication
- Does not rely on the http header
- Customized(non-standard) solution by the developer
- Best user friendly experience
- Highly flexible for the programmer
    - Back end can be a flat file database
    - Credentials can be multiple items at the same time(account no, PIN, BD, and …)

Access Control: aka authorization
- Comes after authentication
- It is the process of granting user access to certain resources on the website
- Also known as authorization
- The dynamic nature of web applications makes access control difficult

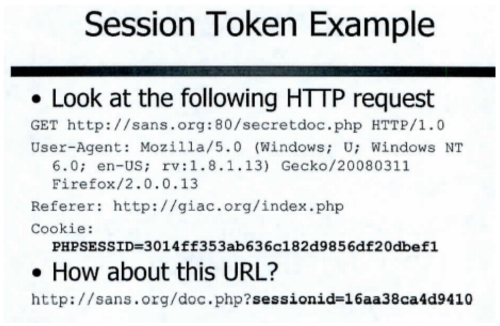
Stateless Nature of HTTP
- Each http request is independent of each other
- No concept of state, rely on other mechanism to tie d/t HTTP requests together
- A unique identifier is usually tagged to each request to achieve this!

Session Tracking Mechanism: Need to track users interacting with web app
- Most popular technique is the use of unique session IDs set by the server
- Session ID
    - Regulate access to one or more websites
    - Identify authenticated users

        - Provide identity for authorization
        - Leverage a database to store user data

Cookie
- Used between server and client
- Used for authentication, tracking of information
- Cookies are set by the server using the “Set-Cookie” header
- Browser will send Cookies back to server, if certain conditions are met, using a “Cookie” header

Session Token Example
- o



Session Token Example

- Look at the following HTTP request

```
GET http://sans.org:80/secretdoc.php HTTP/1.0
User-Agent: Mozilla/5.0 (Windows; U; Windows NT
  6.0; en-US; rv:1.8.1.13) Gecko/20080311
  Firefox/2.0.0.13
Referer: http://giac.org/index.php
Cookie:
  PHPSESSID=3014ff353ab636c182d9856df20dbef1
```

- How about this URL?

```
http://sans.org/doc.php?sessionid=16aa38ca4d9410
```

- Looking at the HTTP request, Prior to this request the server must have already implanted the cookie into the browser, then in subsequent requests the browser automatically sends along the cookie previously set by the server.
- The URL request also carries a session token where the variable name is sessionid(carried by get method)
- In both cases the session token is a random looking string.

Input to Web Application
- URL: 1st processed by the server before the web programming code.
- Header fields: can be processed by both
- Cookie
- URL query string(GET method)
- HTTP body(POST method)
- Other system: such as service backend, database, or LDAP server
- If a developer doesn’t treat all of this input with great care, then security is at stake

Cookie Parameters:
- Name: of the cookie
- Value: some value to be stored on memory or disk
- Expiration: If set, the cookie will be stored until this time expires.
- Domain: the cookie will only be sent to this particular domain
- Path: to further limit the location cookie will be sent to, path may be specified

- Secure: if set, cookie will only be sent over SSL
- Httponly: if set, javascript may not access the cookie

Cookie2
- Set-Cookie2 and Cookie2 work similar to regular cookie
- But focuses on session cookies
- Has to come from the domain it is set for
- Has to originate from the path it s set for

Architecture and Defense-in-Depth

Web Application architecture:
- Components in web applications: there are three well known triers in which web applications can be broken down. Each of the triers performs a d/t distinct function:
    - *Presentation Tier*:is the front end of the web app. It directly communicates with clients. It is responsible for collecting the user input and presenting output to the user in HTML. The data processing of this trier is minimal. It passes data to the application trier to be processed and sent back to it. Which then it will present to the user. Commonly used packages are: Microsoft IIS, Apache, and Zeus
    - *Application Tier*: is the backend of the web application. The heavy lifting happens at this tier. Common languages used in this tier are: Java, PHP, and C.
    - Persistent Tier: is basically the data repository, or database, for the web application. Datas are sent to this tier for persistent storage. The application tier also requests this tier for data stored. Common software: Oracle, IBM DB2, MYSQL, and PostgreSQL.

2-Tier Web Application
- Very simple design and sufficient for smaller web apps.
- The major difference b/n 2-tier and 3-tier is the integration of the presentation and application tier. Meaning the server that serves the pages also processes the data, and applies the business logic to the data.
- Cuz both presentation and application tier are in the same host, “all packages are in one basket” making the risk higher for a 2-tier design.
- Difficult to lock down the web server(due to complexity of multiple components running together)
- Scalability could be an issue
- The applications upgrade capability is limited

3-tier Application
- More Complicated
- There is a clear separation b/t the tiers.

- Has more security benefits
- Cuz there is physical separation b/t tiers monitoring is now possible
- Firewall and IDS possible, web application Firewall possible, Access control possible
- Some of the web app Firewall products are KavaDo InterDo, Sanctum WebShield, and Teros Secure Application Gateway.

N-tiers application
- Physical and logical separation b/n tires(to more than 3 places)
- The persistent layer: is data at the back end, is responsible for storing and providing info to the integration tier. Often multiple databases are present at the data tier.
- The data Access Tier: provides a vendor-neutral and database-neutral API, usually converting all elements of the data tier into XML(or other easy read format)
- Hibernate and Castor are common engineering methods of creating an integration tier.
- The business tier: provides all the application-specific functionality such as rules-engine processing, financial data numer crushing, tax computation rules, credit card authorization, etc
- The presentation tier: is responsible for converting the results from business logic processing and preparing the data to be sent to the client.
- The client Tier: is the layer which the user directly interacts with. There is no data processing here.
- The security benefits of N-tier architecture over other methodologies is huge.
- Here there is stronger separation of developer duties.

Load Balancing
- A load balancing technique is used to spread the load amongst multiple servers. There are multiple ways to load balance the web traffic, the most common ways are: DNS round-robin and load balancers.
- DNS load balancing: is simple. When an IP resolution is requested to the DNS server on the website, the DNS server replies with d/t IP every time. This allows all the IPs that are d/t servers to serve HTTP requests in a round-robin fashion. No guarantee that all requests to the same URL are served by the same server. Vulnerability can exist on one out of many servers

Web Proxy
- Web proxy forwards HTTP request from client to server
- Common uses of web proxy:
    - Filter users based on policies
    - Catching of content so that commonly requested sites can be cached locally, reducing bandwidth
    - Authentication

    - Manipulation(on requests, response …)

Defense in Depth
- The concept comes from military strategy
- Increases the difficulty in compromising critical hosts
- In information security, Defense-in-depth is defined as multiple layers of defense all contributing to protect the critical information.
- Some layers of defense may achieve the same goal(ex. firewalls)
- The protection may be in a form of delays introduced to slow down the attacker till the defender detects it and respond or till the attacker give up
- Very relevant in web apps

Architecture Data-Flow
- Effectively evaluate the architecture components handling the data of the web app
- Components include software, hardware and packages
- Data flow analysis should start with the internet and work through its way toward the data component(persistent tier) and possible to other backend as well
- In case of internal applications, the starting point should be internal workstations.
- For each component handling the data, ask the following questions:
    - How can it be compromised
    - What other components can be exposed through this component?
    - What countermeasure are in place
    - What detective controls are in place
    - Does the component have sufficient protection

WAF - Detection and Prevention
- WAF(web app Firewall)
- When WAF gets information on an input field either on GET or POST, WAF can inspect each of the data fields to check for length, data type, and range of values among other checks.
- WAF is also able to keep track of sessions and detect any tampering.
- WAF is also used for rate limiting of application(ex. Slowing down denial of service or brute force attack)

WAF-Virtual Patching
- When vulnerability is found, it may take a long time to roll out a patch for the vulnerability
- WAF can be used for quick patching of a specific problem
- EX. Filtering HTML tags in input to block cross site scripting
- Also very common for off the shelf software

WAF - Logging and Monitoring

- WAF can be used for logging HTTP layer details
- Especially good for apps that did not have logging in mind
- Logging HTTP field details can drain resources on the server!

Authentication

Authentication:

- Authentication verifies the identity of the person and ensures the person is who he/she claims to be.

Hard coded Database Credentials:

- Having multiple copies of database credentials amongst the code base creates few security issues. Whenever database credentials change we have to change them everywhere and that takes time and prone to error
- Defense 1:
    - Make the credentials static variable in a file
    - Store the file outside of the web directory
    - From program code, include the local file
    - Apache allows "Include" keyword to include files via configuration
- Defense 2:
    - For .Net applications, use web.config file: store all kinds of settings for the application. Its protected by default in IIS
    - Web.config maybe encrypted in shared hosting environment
- Defense 3:
    - In java, the most common approach is to encrypt database credentials: use JNDI
- Testing:
    - Difficult to test by runtime analysis
    - So use Code review
        - Look for SQL statement or the database connection
        - Back trace the connection string
        - See if connection string has username and password

Implementation issue brute forcing tools:

- Brutus: Allows a user to launch a brute force attack on a specific URL. its mostly suited for sites with BASIC authentication
- Crowbar: It is one of the best tools for brute forcing form-based authentication.

Weak authentication Mechanism:

- Some authentication mechanisms are insufficient
- Authentication based on IP or Referer are weak, cuz certain components can be easily spoofed or forged

- So, this authentications should never be trusted
- Defense:
    - Username and Password are the minimum requirement
    - Two-factor authentication is much better to use
    - Using short term mitigation like using WAF authentication of URLs

Browser caching of Credentials:
- In making the web more user-friendly. Many browsers support autocomplete for forms
- Browser automatically caches the information entered be the user(and when the user visits the same site, the information is automatically populated)
- Defense:
    - Set AUTOCOMPLETE of HTML to off!
- Testing:
    - Much easier to test with runtime analysis
    - View source for all login pages and look for AUTOCOMPLETE

Lack of Account Lockout:
- Hackers commonly brute force credentials
- Without any control, hackers can crack passwords via trial and error, which takes a long time but is effective.
- So, brute force of the account should be controlled
- Defense:
    - Most lockout implementations should be custom coded. Other easier options include using standard authentication backends
    - If not handled correctly, account lockout could backfire!
    - PCI-DSS requirement - 6 times, lockout 30 minutes or more.
    - Alternatives:
        - Add delay to each incorrect credential(3-5 sec usually)
        - Alert administrator for a high number of login failures

- Short Term fix
    - Put in rate limit and brute force detection(using custom code)
    - Use web Application Firewall
        - To detect repeated requests for the same resource(login)
        - Rate limit or block clients(ex. Allow 5 requests per minute to a particular URL from a single client)

Weak Password Rest
- Password reset is as important as authentication
- If it;s based on public knowledge of a user the it’s easy to crack!
- It;s often overlooked

- Problematic Implementation:
    - Prompt for a username/email and send the password via email
    - Ask reset questions that are public information
    - Users can select a small subset of questions to answer!
    - Defense:
        - One secure way to do it:
            - Aske user his/her email
            - Ask the user a secret question
            - Sends a temporary URL link to change the password(time-limited)
            - User changes the password or link expires
        - Password never sent in clear
        - Time-sensitive(won't be in the mailbox forever).

Authentication Best Practices:

- An authentication mechanism should be based on risk level
- Authentication is only as good as user management
- Re-authenticate where necessary
- Authenticate the user and the transaction

Sensible Level of security:

- Really annoys the user when authentication is too tedious
- Long sign up once, short authentication afterwards!
- Compromise b/n security and usability

Preferred Mechanism

- Avoid digest and basic authentication
- Form authentication is preferred for all external facing applications
- Certificate Authentication and windows integrated authentication works well for internal apps
- Use SSL for all authentication

User Management

- It's like handling jets out users
- Delete user accounts and privileges when no longer needed

Out of Band Channel Authentication

- Phone or SMS commonly used to verify online activities
- Very strong authentication(as long as phone and sms records are accurate)
- One Time Password distributed through paper medium(only for high risk transactions)

Origin based protection

- Known bad IPs
    - Not terribly effective
    - XBL, malwaredomains.com, TOR exit nodes or RBN network ranges
- IP History
    - Previously offending IPs

- Geo-location mapping

Timing and Referrer Protection

- Timing based detection(if the user is able to login within half second after loading the login page!)
- Referrer tag detection(can be forged though!)
- If the login page is submitted but doesn't have the referrer tag then something is definitely wrong

Per Transaction/Operation Enhanced Authentication

- Sign the actual transaction
    - Man in the middle would be much more difficult if the two-factor authentication is performed at every risky operation or transaction
    - Depending on the authentication type, make the user enter the transaction/operation details

Multi Factor Authentication

Multi factor authentication

- Most authentication is password-dependent! Attacker gets the password, game over!
    - Enhance security by adding multi-factor authentication
- Usually provided by proprietary solutions
    - Vendor provides API
    - Makes things simple for developer
- Not entirely resistant to attacks

Common Authentication solutions

- Time based tokens
- Per-use tokens
- Out of band channels(phone, SMS)
- Challenge response tokens
- Solutions- Non-commercial vs commercial

Two-factor authentication: stopping trojan attack

- Also there is a way get pass two factor authentication still it is better than one factor authentication
- Time based token could prevent trojan attack

Mutual Authentication

- Many attack target the users' inability to distinguish look alike site(user inputs credentials without knowing it's evil site)
- Site can apply mutual authentication, so user can authenticate the site as well using
    - SSL authentication
    - Using Cookie

    - Using a flash or java object to identify the user’s machine

1.5 Factor Authentication:

1. User logs in to the website by username and password
2. Website prompts the user for secret questions
3. User enters answers for secret questions
4. Website allows users to select one graphic icon
5. Website implants a cookie to track the user’s computer
6. User logs in again, the website displays the chosen graphical icon

- Problems:
    - User acceptance factor: it annoying if done again and again
    - Mobile with computer rather than user
    - Malware wipes cookies before sniffing HTTP traffic or keystroke
    - Trojanized PC can circumvent the 1.5 factor authentication protection

Two factor authentication: does not prevent man in the middle attack!

Access Control | Authorization

Authorization: allows access to resources only those who are permitted

- Authorization always comes after authentication

Book 2

Encryption

Encryption (1):

- Protect the integrity and confidentiality of data
- Encryption is important to ensure confidentiality and integrity of data
- Storage encryption: Data is often encrypted on disk to avoid unauthorized access.
- Transport encryption: SSL is a good example of encryption use in Web applications. SSL protects the confidentiality and integrity of data while in transit.

SSL - Secure Socket Layer

- SSL secures data in transit
- HTTPS:// signifies the use of SSL
- *Certificate* is based on Fully Qualified Domain Name(FQDN)
- Relies on trust(web of trust: I trust you, you trust him so I trust him!)
- TLS 1.2 is the latest (Transport Layer Security): uses both MD5 and SHA-1 algorithms for message authentication.
- Over time, SSL has been plagued with security problems. The Slapper Worm!

The Web of Trust: Establishing Trust

- As noted previously, the idea behind the web of trust is based on relationships.
- Customer wants to send secure messages to the server, Customer trusts CA, CA trusts server, Customer trusts CA's trust for server

Cryptography

- Cryptography is used to protect sensitive data(Cryptography not only assures the confidentiality of the data, but the integrity as well.)
    - Sniffing or eavesdropping:
    - Altered during transit:
- Common data to be encrypted: Passwords, Credit Card Numbers and Social Security Numbers
- Two main uses of cryptography in a Web application:
    - *Protect data In transit:* Data in transit would be data that is being transported from one point to another, such as over a LAN or the Internet. Data in transit can be sniffed or even altered if the attacker has access to any intermediate point the traffic passes through.
    - *Protect data in storage*: Data in storage can become compromised if the OS hosting the data becomes compromised, or the administrator of the OS (not the application) abuses his/her privilege to access the data in an authorized manner.

Encryption Terminology: What you need to know about encryption:

- *Ciphers*: Algorithms used to encrypt data and protect privacy. Encryption depends on the ciphers to encrypt data.
- *Hashes*: Algorithms used to sign data and detect changes in transit. Encryption depends on hashes to authenticate data. A hashing algorithm is used to ensure the data did not change during transit.
- In the case of SSL encryption, versions are negotiated during the handshake phase.SSL supports many cipher algorithms, and these are negotiated next in the handshake.
- Encryption can be symmetric and asymmetric.
    - *Symmetric encryption* would mean: encryption and decryption all happens with one key, this type of encryption is usually fast but requires the encrypting party and the decrypting party to share the same secret key.
    - *Asymmetri*c addresses the problem of having to share the same key, since it allows one key for encrypting (public key), and another key for decrypting (also known as the private key which is not shared with anyone).

Lack of Storage Encryption

- It is a common misconception that SSL takes care of all encryption needs within a Web app. SSL only protects information while it is between two network points!
- Data that is stored on a server can be vulnerable to attacks as well.
- Data can be compromised
    - by attackers that compromise the server hosting the data, or
    - by rogue administrators who abuse their privilege.
- Database Encryption:
    - The main goal of database encryption is to guard against unauthorized access to the data hosted in the database. Hackers compromise the database and database Administrators as well.
    - It is essential to keep in mind that database encryption does not necessarily protect data from being leaked or compromised by using vulnerabilities in the Web application itself.
- Common Approaches to Database Encryption
    - As algorithms such as cipher and hash are commodities, the hardest part of encryption is key management.
    - Some database vendors are building encryption into the database package. Database vendors built in key management in the database where the database encrypts the encryption key until a user with proper identity and permission request the key (and the data).
    - Some other database vendors use PKI and asymmetric encryption algorithm to perform the encryption.
    - Storing key within the code/application is not a good idea!

Key Management:

- The encryption key can be stored on the file system of the Web application server, outside of the Web accessible directories, and protected with File system permission.
- Depending on the usage, databases can store the encryption key as well. These solutions generally do not block the DBA from viewing the keys and decrypting the content!
- There is also external storage of encryption keys. The most common way is having an LDAP directory storing the encryption key, then the DBA would likely not have access to the LDAP directory and, hence, the key.

Commercial Solutions:

- There are many commercial vendors available to tackle the problem of encrypting data on the database.
- nCipher, protegrity, netLib (MSSQL only) and Safenet are some names.

Best Practice for Key Management

- Avoid storing keys in the code.
- All access to the encryption should be logged and regularly reviewed.
- When initiating keys, make sure you do it over a secure channel. channel. For example, e-mailing of keys is very insecure

Database Encryption Common Gotchas

- Performance issue: Encryption costs lots of processing power and needs to be planned properly.
    - Careful when encrypting the primary key or index field as they are accessed very often
    - Some databases oniy support "all or nothing' encryption
    - Encrypted fields are larger than unencrypted fields
- Exposed key: If keys can easily be exposed to the attackers you are trying to protect against, the game is over. Keys should be only accessible to the process running the Web applications (e.g. Service account) and no one else.
- Keys can be lost due to disk corruption and other disasters. Make sure the key is being backed up properly and test it periodically by restoring the key.

Alternative to Database Encryption

- Use object security - Use SQL grant and revoke statements to restrict which accounts can access the data.
- Store a secure hash - Rather than storing the data directly, store a hash of the data.
- Do not store the data - Do you really need to store the data to begin with?
- Store the data somewhere else - Perhaps the database is not the right place for this data.

Storing Password in Cleartext:
- Password should never be stored as cleartext in database
- Anyone having access to database has access to ALL passwords
- Passwords are never meant to be recoverable
- Cryptographic hashes are the proper way of storing passwords
- Hashing:
    - is a one way function, the process of hashing takes the original text and processes it by a one way function, the result can then be stored in the database.
    - The resulting text (hash) is computationally difficult to convert back into the original form.
    - Increase time taken to recover the original password
    - MD5 (too old), SHA-1, SHA-256
- Storing Password in Cleartext Hash Cracking
    - Brute Force cracking: try every combination. It takes a long time
    - Rainbow table cracking: -Pre~computed hashes to be compared to all possible passwords. Takes minimal time to compare once a table is generated.

SSL Insecure Settings
- SSLv2 is ancient and should be avoided. It'a weaker than other SSL versions (SSLvB and TLSvl)
- Watch out for NULL cipher:
    - and Null cipher = No encryption (clear text). It should be used For debugging only!
- Mitigations
    - Microsoft IIS Server: SCHANNEL key in registry
    - Apache: SSL CipherSuite in config file

Cryptography Best Practices
- Determine requirements for encryption early in development cycle
- Use Risk based approach
- Know what you are protecting and who you are protecting against

SSL Best Practices
- Use SSL for all of the following situations
    - Any personal or sensitive information
    - All login credentials
    - Any page or content past login
- Input forms before login are often missed! So use ssl for them as well

Storage Encryption Best Practice:

- Regulatory and compliance requirements are critical in storage encryption
- Ask the question, "who are you protecting against?"
- Determine what the data has to be safeguarded against while on disk.

Reasonable Level of Crypto Technologies

- Hashing Algorithm: -SHA-l, SHA-256, RIPEMD-160
- Symmetric Algorithm: 3DES, AES, Blowfish, Twofish, Serpent
- Asymmetric Algorithm: RSA, DSA

DHE cipher:

- Diffie-Hellman cipher suite is resistant to attacks through packet collection and. decrypt using the original SSL private certificate; which means, even if the attacker is able to sniff the packets on the network and is able to steal the SSL private certificate, the traffic still cannot be decrypted.
- DHE cipher cannot be sniffed and decrypted because each side of the communication establishes the encryption key at the beginning of the communication and also re-key periodically, each key is not dependent on it's preseder

EV Cert

- Extended Validation Certificate
- SSL certificates becoming easier to get but EV cert requires more validation on identity of the owner
- EV cert makes the browser show "green" signal

Multi Domain SSL Certificate/Hosting

- Unified Communication Certificate
    - Multiple Subject Alternative Name
    - One cert for multiple domains

HTTP Strict Transport Security (HSTS)

- Uses Strict-Transport-Security Header
- Forces the browser to connect using SSL/TLS
    - Cert error = denied access
    - -Auto change http:// to https://

Public Key Encryption

- is the underdog of encryption systems and is much less known than Symmetric encryption.
- With proper implementation, Public Key Encryption can provide a very strong encryption system for scenarios that a symmetric system cannot properly secure.

- To understand Public Key Encryption, we must know that this system leverages two keys.
- One of the keys is called public key and the other private key.
- The private key is intended to be kept secret, while the public key is widely distributed.
- These two keys are mathematically related: the data encrypted with the public key can only be decrypted by the private key.
- The Public Key Encryption system can be very useful in Web applications. The application server can encrypt data that only backend servers can decrypt
- Even if an application server is compromised, no data would be lost

Session

Session

- Session mechanism is a critical component of Web applications. Without session, users will have to put in credentials at every single Web page.
- Weak Session Token
    - Extremely simple attack
    - Determine victim's session token and impersonate as victim
    - The session token is as important as username and password
    - Protect session token with your life
- Session Attacks: *Interception*
    - Evil Computer intercepts communication and copies session token
    - Evil computer contacts Web app using a stolen token
    - Evil computer is now authenticated with stolen credentials
- Session Attacks: Prediction and Brute Force
    - Predicting valid session IDs and Brute Forcing valid session IDs are the same
    - First session seen was 123, Second was 124, Guess next is 125,
- Weak Session Token Mitigation
    - Do not use URL SessionlDs
    - Use the Crypto system to generate random session token
    - Use SSL
    - Weak Session Token Extra Validation - User Agent
        - User-agent is a string sent by the browser in the header
        - Do not change unless the user switches browser and can be forged, use only as added protection
        - this is only an added security feature on top of the session token, it can't be used alone
        - User agent validation process
            - Store user-agent string at login
            - Every subsequent request, validate user-agent string
            - If user-agent changed, terminate session immediately

        - Weak Session Token Extra Validation - IP Address
            - The IP address does not normally change during an online session
            - Load balanced proxy can make IP inconsistent: Best to be used on the Intranet
            - Will not provide protection against attack from the same physical network

Session Fixation

- Session fixation allows an attacker fixes a victim's session ID before they login to the application, eliminating the need to acquire the session ID after the victim logs in.
- Some countermeasures for session fixation include:
    - Never accept "chosen" sessionIDs. Always generate a new session ID for each new login request.
    - Assign a sessionID after successful login.
    - Bind the session ID to an immutable property of the client; such as network address or SSL Client certificate.
- Session Fixation Mitigation (1)
    - Avoid GET/POST based sessions. Use cookie or SSL identifier for session token
    - Only accept server generated session token(reject any other)
- Session Fixation Mitigation (2)
    - Regenerate session ID on each request
    - Generate another token with random content during login

Lack of Session Binding

- Some data may be tracked outside of session scope
- Data sent to the client is subject to manipulation
- All session data should be stored on the server
- MItigation:
    - *Short Term Mitigation*: Some WAF devices allow validation on specific variables, to guard against manipulation
    - *Long Term Mitigation*: Use the session mechanism to store session data on the server

Cookie Insecure Settings

- Many sites do not set cookies flag properly.Cookie flags are used to tell the browser how to treat the cookie and under what circumstances to send the cookie. Improperly set cookies lead to many different risk exposures.
- Since cookies are usually used to cany a session token, which is a critical piece of information, any security exposure can be catastrophic.
- Protection
    - Secure: -Indicates that the cookie can only be sent over SSL

        - Httponly: -Prevent JavaScript from accessing cookie and -Normal session operation works, stop Cross Site Scripting
    - **Surf Jack (1)**
        - Surf Jack is a type of attack that targets sites that do not set a cookie as secure.
        - Example:
            - User login to his bank account and the bank server gives him a cookie
            - The cookie was not set to secure
            - The attacker waits for the user to visit http link
            - Then when he does the attacker sends http version the bank page(instead fo the original https) - since the attacker is closer to the victim his page will reach victim first(not the right page)
            - The browser sends the cookie in clear
            - The attacker uses the cookie
    - **Surf Jack (2)**
        - The attack was possibly mostly due to the wireless network injection making it possible. Attacker in proximity responds quicker than the remote website
        - Also work for another man-in-the-middle situation
    - **Mitigations**
        - Short term solution: WAF devices allow injection of flags to cookie
        - Set the cookie security setting properly in the code(“secure”)
        - All sessions or sensitive cookies should be SSL protected and httponly

Improper Logout

- When user clicks on logout button, server needs to destroy all session data
- Often, sites claim that a user is logged off, but it's not the case! The back button does magic
- Mitigation
    - Logout should immediately invalidate the session on the server
    - Set some hard limits on , session length

Session Best Practices

- Use cookies or SSL ID
- Use proper timeout for inactivity
- Session data should all be stored on the server
- Use SSL to protect session ID
- Use built-in session toolkit

Sharing Session with 3rd Party

Sharing Session with 3rd Party

- Sometimes an application needs to forward an authenticated user to 3rd party application

- Share the authentication but not the session ID. Remember session ID is the same as a credential, not for sharing
- Scenarios for Sharing(this are just some possible solutions)
    - Be part of the subdomain and share cookie
    - Authentication token passed between two sites, with back-channel to validate
    - Use encrypted/signed token
- *Subdomain Solution*
    - Make 3rd party site subdomain of the authentication site
    - Parent and subdomain can share cookie
    - Benefits
        - Very easy to setup
        - Cookie mechanism well understood
        - More trustworthy looking URL. It will avoid suspicion from users regarding redirect
        - Lets us use Mature implementation such as Pubcookie
    - Potential Problems
        - Inviting 3rd party into your domain perimeter might not be a good idea
        - Cookies can be stolen if domain not set properly
        - JavaScript allowed to run as the parent site (document, domain)
- *Redirection Token Solution*
    - Cookie does not work across different domains
    - when two sites in different domains have to share authentication, some other mechanisms have to be brought in to handle the cross site nature of session sharing.
    - A solution commonly used is redirection with a token to the 3rd party site and let 3rd party validate the token through a back channel.
    - Once the user visits the 3rd party site, the token is collected by the 3rd party site and is immediately verified by the main site for validation.
    - Once that process is done, client can visit the 3rd party site
    - Benefit
        - Able to allow 3rd party to share session data: using the back-channel
        - Hardly notice the change in domain
        - No need to expose any same-domain issue
    - Potential Problems
        - The token is sent in the URL
        - Token has to be one-time only: -Prone to Hijack
        - Must be SSL protected
- *Crypto Token*
    - When you only need to know the identity of the user, back-channel = overkill

    - A crypto token that can carry a small amount of data may be the best solution for this kind of requirement.
    - User carries an encrypted token which can be decrypted by third-parties
    - Benefits
        - No need to build back-channel
        - Low latency, less back and forth traffic
    - Potential Problems
        - Time synchronization
        - Need to invalidate token once received, further avoid replays
        - Higher server load due to encryption

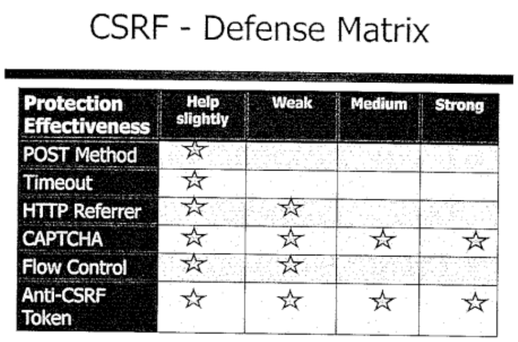
CSRF

CSRF

- Cross site request forgery is a vulnerability that exploits the trust that a site has on the user.
- Also known as one-click attack or session riding
- An application usually looks at the session mechanism and determines whether a user is who they claim they are, and once it is validated, the site trusts the user.
- However, the user is not always who they claim they are.
- In a Cross site request forgery attack, the attack leverages the fact that the client is already logged in and anything that the client sends to the server will be regarded by the server as client activity.
- The attacker tried to trick the victim's browser into sending a request to the server.
- *Quick walkthrough of CSRF attack*
    - Attacker controls some website, which has a link to the target site
    - Victim visits the attacker-controlled site
    - The victim's Web browser creates an HTTP request to target site:
        - Victim had already authenticated to target site with a valid session cookie
    - Triggers an action on the target site
- *CSRF - Attack Dependencies*
    - Victim has logged into a Web application
        - Have a valid session token
        - Automatically works in the case of Windows integrated authentication
    - The attacker knows the URL or parameters to pass to the Web application in order to trigger an action
    - Attackers have to be able to trick the user to visit a pre-constructed link
- *CSRF - Potential Effects*
    - The vulnerability covers every Web application function that requires a single request
    - So Attacker can attack any existing functions in the applications

- *CSRF - Common Techniques to Trigger*
  - Attacker commonly uses the following tags to trigger request
    - -Image Tag <IMG>
    - -Script <script>
    - -IFRAME <IFRAME>
    - -XMLHTTP
- *CSRF - Defense* - POST Method
  - The highest risk targets are the GET method requests
  - Make afl forms leading to action only accept POST
  - This still does NOT fully mitigate CSRF!!!
- *CSRF - Defense* - Session Timeout
  - CSRF attacks leverage the existing authenticated session to slide in the additional requests
  - The session has to be valid before the attacker can trick the victim to make the request
  - Therefore Short or reasonable timeout schedule is important
  - Not a full mitigation, but it helps
- *CSRF - Defense* - HTTP Referrer
  - HTTP referrer can be helpful to mitigating CSRF - It is not a full solution since the HTTP referrer can easily be spoofed
  - There is really no reason for some sites to link directly to your form actions
  - You should investigate all HTTP referrer failures or errors
- *CSRF - Defense* - **CAPTCHA**
  - A CAPTCHA randomizes the response required to submit to the form(automate submission will be impossible)
  - If implemented correctly, it provides strong protection against CSRF
  - Not very practical for every form though
- *CSRF - Defense* - Flow Control
  - The attacker does not have any knowledge of the victim's state with the target site
  - For every critical form in the site
    - Put a boolean state to track whether a user has visited the form page before submission
- *CSRF - Defense* - Anti-CSRF Token
  - In a CSRF attack, the attacker does not see HTML form being sent from server to client
  - Generating a dynamic token for every form mitigates CSRF
    - Token has to be random in nature
    - Token must be verifiable by the server
    - Attacker should not be able to guess the token

        - Attach a token to each form and verify the values when user submits token back with the form
            - **Hidden form field**



CSRF - Defense Matrix

Protection Effectiveness	Help slightly	Weak	Medium	Strong
POST Method	☆			
Timeout	☆			
HTTP Referrer	☆	☆		
CAPTCHA	☆	☆	☆	☆
Flow Control	☆	☆		
Anti-CSRF Token	☆	☆	☆	☆

Business Logic & Concurrency

Business Logic & Concurrency

- Business logic dictates the actions of the application
- Concurrency enhances scalability but it can also cause issues

Business Logic Flaw

- Business logic models real life business objectives meaning Business rules and workflow drive the application
- Business logic flaws can be -serious and QA and vulnerability scanner might not detect them
- Mitigation
    - At the design phase of the SDLC, make sure the abuse cases are carefully considered.
    - Have security expert understand the whole business process
    - Ensure security testing with qualified security personnel post-development

Concurrency

- Concurrency is the ability for a system to run multiple threads of execution simultaneously
- Its tough to master
- Two threads can access the same resource in the systems
- Concurrency Control
    - Certain resources need to be used exclusively
    - *Pessimistic* - Block operations that would cause integrity issues
    - *Optimistic* - Do the execution first, when done, check and resolve conflict
- Concurrency Deadlock: While using lock to isolate variables, it can create deadlock
- Mitigation:
    - Isolation - Locking only what's needed

- Optimistic concurrency control is more suitable for HTTP stateless nature
- No general fix, unique solution to each scenario

Input Validation

Input Validation Related Flaws

- Input is essential to the operation of an application, but it can also cause serious harm
- Buffer Overflow
    - Buffer overflow is possible in custom Web applications
    - Most Web scripting languages (Java, Perl, and PHP)are immune or have protection against buffer overflows
    - It is difficult to exploit in Web applications:
    - Likely to only cause a denial of service attack
    - Mitigation
        - Input validation: validate length of input
        - Educate developers on buffer overflow
        - Avoid using languages/functions that are vulnerable to buffer overflow
        - Ensure all Web components are free of known vulnerabilities
- Client Side Validation
    - Some validations are done on the client side only. Usually done by JavaScript
    - The client side validation acts as a blueprint for the attacker. Easily circumvented
    - Mitigation
        - WAF can provide very quick validation based on URL and field
        - Make sure all validation is done on the server side
        - If validation exists on client side and server gets an exception, most likely a malicious attempt
- OS Command Injection
    - Some Web applications run OS commands to perform certain functions. This is generally a bad idea.
    - Badly designed applications use user input as the command line argument. User could input anything he wanna run on the OS and cause some damage
    - Executing user input in the command line is bad
    - In this case, not filtering the meta is unforgivable
    - Mitigation
        - Input validation
            - -Constraint input : At minimum, block semicolon, backtick, and pipe characters
        - Do not use OS commands in Web apps
        - WAF devices have blocking rules for OS injection so se that

Remote File Include

Remote File Include

- Attacker can run malicious code on the victim server
- Can happen in many development platforms PHP is most vulnerable
- Developer uses code directly from a file: The filename is a user supplied input which attacker can manipulate
- Mitigation
    - Validate and initialize variables before using them
    - Minimize user influence on the execution path
    - register_globals
    - allow_urL_fopen and allow url include

HTTP Response Splitting

HTTP Response Splitting

- HTTP response splitting allows attackers to inject information into the HTTP response header
- Does not directly attack the Web application itself - Targets are usually clients of the website or infrastructure components
- Rather than an attack by itself, it is usually used in conjunction with other attacks
- Possibilities
    - Session fixation, as demonstrated/ is one of the possible attacks
    - Proxy Cache poisoning is also possible
    - Cross site scripting - XSS
        - Easier XSS attacks
        - Can be leveraged to further poison the victim
- Defense:
    - General input validation rules apply here
    - Infrastructure components also help to block the attack
        - NIPS (Network Intrusion Prevention System) and Application firewall

Book 3
SQL Injection

SQL Injection

- One of the most commonly exploited vulnerabilities in Web applications and -It's easy! You just need some basic SQL understanding
- Caused by Lack of checking on the input. User input gets passed without checking to the back-end SQL database
- Attack Potentials
    - Change meaning of existing SQL queries
    - Extract data from database
    - Alter data and structure in database
    - Control the host running the database, jumping to other hosts on the network
    -
- A List of Common Database Error Messages
    - SQL errors in Web Applications give very distinct messages. Based on these error messages, attackers can plan the injection attack
- Blind SQL Injection
    - Although the attacking method for SQL injection without error messages can be a little different than error message based SQL injection, the effects are essentially the same.
    - Host does not return error messages when Injected with meta-characters - It can still be vulnerable to SQL injection
    - Blind SQL injection is based on questions and answers
        - Yes or No answer from the server
        - Not as easy as being tipped off by error messages
    - Example: http://www.sans.orq/vuln.php?id=10 or id=10' or id=9+1 . . .
    - *Mitigation:* Filtering Input and Validation
        - Constrain input: -Allow only "known good" input, -Type, Length, Format and Range of input should be checked,
        - Reject the bad input: -According to your database Reject reserved SQL words "insert" "delete"n-" "char" and SQL metacharacters ‘ # …
    - Filtering is Hard
        - Filtering and catching the bad input is a very difficult task
        - How about: -SE/**/LECT/**/
        - This is not to say that filtering defense does not work, it is simply the nature of filtering bad input not being the best solution.
    - *Mitigation*: Escaping Input
        - Escape the quote'
            - Database dependent, it could be double quote " or \'
        - Should escape all database metacharacters (-, #f @)

        - Effective against most SQL injection
- *Mitigation*: Using Language Built-in Mitigation
    - Many languages have built-in functions to escape special characters
    - Some examples of built-in functions
        - Perl - DBI:Quote
        - PHP - mysqLreaLescape_string
        - Most of these functions add a slash before the special character
- Escaping Challenges!
    - Need to escape according to the database
    - Numeric data type can be a problem since these fields do not require quotes
    - Need to escape all metacharacters, not just quotes
- *Mitigation*: Parameterized Query
    - Dynamic SQL statement is used
    - Pass parameter to SQL statement: Pre-construct the static part of the query
    - It’s Available in most Web languages
    - User input is passed as "input" to a dynamic statement
    - In the end, the statement is assembled properly and sent to the database
    - Easy solution to a complicated problem and it works better
- *Mitigation*: Stored Procedure
    - Stored procedures are small programs on the database that execute from the Web application
    - Not all Stored Procedures can mitigate SQL injection
    - Never use dynamic SQL within the stored procedure, It will be vulnerable to SQL injection and defeat stored procedure's protection
- *Mitigation*: Database Permission and Hardening
    - The Web app should use low-privilege accounts to authenticate with DB. If Web app only needs read access, ONLY grant read access to the data
    - Harden the SQL server appropriately
        - Delete unneeded stored procedures
        - Delete default user accounts (especially Oracle)
        - Monitor SQL server outbound connection to detect potential 2nd stage intrusion
- *Mitigation*: Limiting SQL Error Messages
    - Limiting the SQL error message is simply security by obscurity.
    - Error message aids the attacker during the attack phase
    - Production Web application should have all error messages sent to local logs and NOT to the user

Cross Site Scripting (XSS)

Cross Site Scripting (XSS)

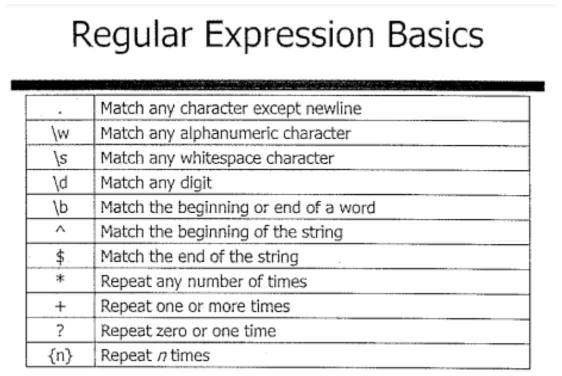
- The malicious intent of Cross Site Scripting (XSS) is to trick the browser to execute malicious scripting commands.
- The victim's browser gets fed some scripting commands by the attacker through some unsecure Web application.
- attacks, XSS targets the clients instead of the Web application itself.
- Caused by insufficient input validation
- In a general XSS attack, there are usually three parties involved:
    - Client or the victim with the browser.
    - The server which may or may not be malicious.
    - Attacker.
- Cause of XSS
    - When Web application fails to validate user input
    - User input to the Web application is displayed back in its original form!
    - Attacker simply injects malicious content/code to the legitimate website. Code injected will appear to the client as coming from the legit website
- Potential Effects of XSS
    - Disclosure of Cookies. Which Lead to hijack of user's session
    - Force redirection which can help with phishing
    - Modifying content of the Web page
        - -Substitute words
        - Changing images
        - Inserting words or images
    - Running of custom scripting commands
- XSS - Reflection
    - The attacker sends a crafted URL with malicious scripts embedded
    - The URL is sent to the user in an e-mail or through other ways that the user will click on
    - The user clicks on the link and executes the malicious script
    - Malicious content does not get stored on the server
        - Sent to the server by the victim
        - Server bounces the original input to the victim unmodified
- XSS - Persistent
    - An attacker first posts a message containing a scripting command to a Web-based bulletin board system.
    - The scripting command instructs the browser to send in the cookie for the target site.
    - The bulletin board system site does not perform any sort of validation on the user input; therefore, the attacker's script is stored and displayed to anyone in its original form.

    - Next, a victim casually browses the site. He downloads the page sent by the attacker, and the scripting command is executed by the browser of the victim. The browser happily sends the victim's cookie to the attacker,
- XSS - Local (DOIM-based)
    - The Injected script does not traverse to the Server
    - Vulnerability within the scripts in the HTML file
    - The offensive content gets displayed back to the victim by the local scripting language
    - Arising fast as the major threat, as the other two types of XSS are getting cleaned
- XSS Common Attacks
    - Steal Cookies
    - Alter Contents on the vulnerable site
    - Keystroke logger & redirect to exploit site
    - Cross leverage CSRF and spread the XSS
- *XSS Defense* - Filtering
    - Filter out the HTML meta characters(<>';&\%n type of characters)
    - Need to watch for encoding
    - Since It may backfire, don't just blindly filter characters
    - This has many pitfalls
- *XSS Defense* -Output Encoding
    - Escape or encode all HTML entities to be sent to the clients
        - Write an output function and use it to perform HTML encoding
- XSS Defense - Encoding & DOM Based XSS
    - Set the proper character set in the HTML code(force the client to that character set). To set encoding, there are two ways to do so, either through meta tags within the HTML file or through an HTTP response header. It good idea to use both.
    - Review client side code for DOM based XSS. Avoid using client side rewriting or redirection based on client input
    - For the DOM based XSS, the defense is pretty much the same as server side XSS.
- Input Filtering or Output Encoding?
    - Simple answer is Do both
    - Input Filtering cuts down on the possible scripting in the input
    - Cut down the noise on the input and then really get rid of it at the output
        - - XSS happens on the output data
- Encoding Complications
    - HTML encoding still does not work in some scenarios. HTML encoding works for HTML content and inside <div>

Input Validation Strategies

Input Validation Strategies

- By far the hardest process in protecting Web applications
- No one solution fits all
- Centralize as much as possible
- Validate before processing, as early as possible
- Validation Steps in an Application
    - Client Side Validation
    - Web Application Firewall
    - Web server filter
    - Validation within dev framework
    - Customized validation for fields
- Input Validation; Save Yourself Some Work
    - Only use user inputs when necessary
    - Back and forth hauling of information is unnecessary
    - Less input = less risk
    - Design application so inputs are easy to validate
- Validate Source of Data
    - Specify the source of data POST or GET method, Cookie or Header
    - Have huge security implications, often ignored by developers
    - Simply validate the source of variable is sufficient
- **Canonicalization**
    - Canonicalization is a huge issue in input validation, most of the advanced Web attacks leverage some form of Canonicalization to circumvent security controls.
    - Some of the common encoding that causes issues are Unicode and URL encode.
    - If input is a URL, decode the URL
- Regular Expression
    - Regular expression is most commonly used to describe a search pattern on text.
    - Used to match text with a specific pattern
    - Extremely powerful for input validation(supported by all languages)
    - Regular Expression Library http://regexlib.com/
    - Basic:



Regular Expression Basics

.	Match any character except newline
\w	Match any alphanumeric character
\s	Match any whitespace character
\d	Match any digit
\b	Match the beginning or end of a word
^	Match the beginning of the string
$	Match the end of the string
*	Repeat any number of times
+	Repeat one or more times
?	Repeat zero or one time
{n}	Repeat *n* times

-

- Input Validation - **Whitelist**
    - For simple and fixed data fields, use whitelist
    - Accept ONLY expected acceptable input that means we must Must know the expected input format and range and implement it
    - Good example is phone number
    - Downfall: does not work for all inputs smooth
- Input Validation - **Blacklist**
    - Not as good as whitelist, but required in many situations
    - Needed when the type, length, format or ranges are not known ahead of time
    - Blacklist is always attack driven
- Input Validation - Blacklist Candidates
    - -SQL Injection
    - -OS injection
    - -Buffer Overflow
    - -Cross Site Scripting
    - -Path Manipulation
    - -Response Splitting
- Input Validation - Situational Awareness
    - If you are disallowing quotes (') and double quotes ("), need to be clear to the user
    - Input validation needs to be well planned
- SQL Injection Blacklist
    - Filtering quote character might be too aggressive
    - Filter list
        - union, select, drop, delete, --, @@, char, exec
        - Al! cases since SQL is case insensitive
    - Also use Defense in-depth, filtering cannot be used as the sole protection mechanism
- Cross Site Scripting Blacklist
    - Input validation is not the preferred solution for defending cross site scripting
    - Output encoding also essential
    - Input validation for Cross Site Scripting is really to trim down on the noise and try to catch the obvious attack attempts. One of the effective regular expressions to use is the following1
- Path Manipulation Blacklist
    - Path manipulation can be simple to pick up
    - Block list: -\ • \• i •
- Response Splitting Blacklist
    - Only filter while the input is used in the header
    - Block List include [\r\n]* or just \n
- Reading File

        - File can be a data source
        - All data needs to be validated before processing
        - Pay attention to malformed file
    - Handling HTML Input
        - Use alternative markup languages(-BBCode, Wikitext, Textile)
        - There is also an HTML purifier for PHP platform, which is built to strip HTML input of any scripting commands or any other unwanted HTML tags.
        - OWASP also has a project called Anti-Samy which also handles HTML purification.
    - PHP Filter Functions
        - PHP provides Filter functions for input validation
        - Can examine input through a pre-set list of expected format (e-mail, IP...)
        - Can sanitize some input (URL, encoded/ special character)
        - Intended to be used as part of the validation solution

File Upload

Defense & Evasive Practices

- There are other defense techniques to avoid potential security issues
- Handling File Upload
    - File upload can be a serious risk to the application
    - A form of input to the application
    - Especially risky if content uploaded becomes part of the site later
    - File Upload Risks
        - Unexpected size
        - File name and extension
        - File type (e.g. PHP, JSP), executable content
        - Virus or malware
- File Upload Handling Strategies (1)
    - Specify the size limit in HTML
    - Check MIME type on server(don't trust fully)
    - Check the file name and rename
    - The file should be saved at a directory location outside of the web directories.
- File Upload Handling Strategies (2)
    - Virus check
    - Check file content
        - - Use "file" command
        - - Inspect into the header and content of the file
        - - Be careful how you open the file, various libraries can be vulnerable
    - Resize graphic image then re-save

Security Config and Environment

Environment & Config
- Configuration affects security

Directory Browsing
- Directory browsing gives the user a listing of files in a directory. It Only gives the listing when the default index files do not exist. This could cause Leaks of important information
- It can be enabled on a per directory basis so it can be hard to hunt down, unless config for every directory is checked.
- Mitigation
    - Remember it is a per-directory setting
    - In Apache
        - make sure that -Indexes is included for each directory
    - For MS IIS, Directory browsing is configured at the directory permission pane within IIS configuration.
    - **Add an empty index.html into all directories**

Data Leakage
- Data Leakage can be a problem when the overall tidiness of the code or web directories is lacking.
- The problem could be multi-fold, such as having odd and unnecessary files or backup files that are not necessary for the Web application operations.
- Mitigation
    - Periodically audit the web directory
    - it is an option to turn on access time (atime) on file system, which allows tracking of last file access time.
    - Doing Code review to pick out the comments
    - Coding standards should include not commenting HTML code

Cross Site Tracing
- TRACE is an HTTP method. Its main purpose is for debugging of HTTP protocol.
- When a TRACE request is sent to a server, the server simply echoes back the original header. It’s like ping
- all HTTP compliant servers by default support it.
- Cross Site Tracing Attack Process
    - The attacker poisons the target website using persistent XSS attack.
    - The user (victim) requests the XSS injected page from the target site.
    - The website sends the content and also the script command to the ask browser to resend the request with TRACE method
    - The client's browser sends the TRACE method to the Web server. The client's browser automatically sends a cookie to the Web server.

- The Web server responds to the TRACE request with a response containing the original request with the cookie.
    - 6. Once the data arrives at the user's browser, the original XSS code processes the returned data and parses out the cookie.
- Cross Site Tracing Dependency
    - Browser exploits and weak script isolation required to happen
    - Flash, Java, VBScript may also be used
    - Undermine httponly in cookie setting
    - In IIS 5, there is also TRACK
- Cross Site Tracing Mitigation
    - Apache: enable mocLrewrite and TraceEnable Off
    - MS IIS: -Urlscan -Disabled by default on IIS 6 & 7
    - Web Application Firewall can filter on the TRACE method also!

Backdoor & Service Isolation
- Running multiple services on the same physical host could increase vulnerability
- Sharing backend storage between critical hosts can also be security risk
- The administrative interface can be on the same sewer listening on a different port. This allows an attacker to access the administrative interface very easily and potentially compromise it directly.

Isolation of Multiple Services on One Host:
- Running multiple services on the same host might be risky, so consider this:
    - Running a writable FTP server and Web on the same host
    - Overlap directory structure for both services
    - Upload file using FTP and execute the file from the Web server

Isolation DMZ Share Internal Resource
- DMZ is a zone dedicated to external facing hosts
- Careful when sharing internal resources in DMZ hosts
- Share SAN (Fiber Channel, iSCSI) or NAS in internal network
- Using Virtual environment to use one physical box running both internal and external hosts is also risky

Isolation Administrative Interface:
- The administrative interface should remain isolated
- Different access method or URL
- Separate credentials required and it’s Better to be a separate system all together

Mitigation: for the above
- Run single purpose host cuz Machines are cheap these days
- Reduce sharing of resources to a minimal
- Clear security boundaries between logical components

Vulnerability on Hosts

- Applications run on server software and Operating System, which can have vulnerabilities
- Vulnerability Scanning
    - Infrastructure scan has great value for Web application
    - Nessus is easy, low cost solution
    - And NO, Nessus does not scan all Web app issues!
    - Secunia CSI is one such solution. It is more thorough than remote vulnerability scanning and can also pick up outdated libraries that need to be upgraded.
- Mitigation
    - Stay tuned to industry news, get the whole security team involved
    - Application owners should subscribe to vulnerability feed
    - Patch early, patch frequently
    - Use Firewall rules and Lock down hosts

Logging

Logging & Error Handling

- Logs are useful in troubleshooting and forensics
- Information Leak via Error Message
    - Error messages can reveal too much information
    - Attackers often leverage error messages to craft their attack
    - SQL Injection attack is a lot easier with proper error messages
- Mitigation
    - the programming code should handle the error condition instead of letting the Web platform handle it.
    - Design error messages carefully
    - No internal state displayed to the user

Log Injection

- It’s Similar to other injection problems
- It’s Injection into audit logs
- Attacker can cover tracks and mislead any forensics attempt
- Log injection may not be a problem. Depending on your logging method, it is most susceptible when logs are simply stored in a text file on the local machine.
- Mitigation
    - To prevent log injection from happening, try to avoid using any input as part of the log file.
    - Any user input should go through a whitelist
    - Make sure the log viewer does not render or interpret the code in the logs (control character, HTML)

Error Handling Best Practice

- Handle all exceptions
- Every single call to other components should prepare to handle errors
- Leverage standard framework (Iog4j, Iog4php)
- Send logs off for storage

General Approach to Handling Errors

- Log
- Redirect to error page
- Stop execution

What to Log: Authentication and Access Control

- For logging, all authentication and access control events should be logged.This includes successes and failures.
- All other authentication and access control-related events, such as account lockouts and policy violations, should also be logged.
- Logs should include accessed resources and the reason for denying access

What to Log: Data Access

- It is critical to log the following actions:
    - - Data reads
    - - Data writes
    - - Data deletes
- The data read log might be huge.
- Writes and deletes are generally much more important than reads
- Changes to data characteristics—Data structure
- User and admin events are all high impact events to monitor. Any change to the data set, database schema, user settings, roles, and permissions should be logged

What to Log: Errors

- Log all error conditions:
    - - Failed queries
    - - File not found errors and cannot open errors
    - - Unexpected states
    - - Connection failure (database)
    - - Timeout or performance problem
- Some errors can be related to availability of the application

WAF Device for Logging

- Quick logging solution
- Does not rely on code to generate log

- Does not require resources on server
- Log input fields and output
- Safer for logs to be outside of the origin machine

Incident Handling Plan

Incident Handling Plan

- Incident plans must be drafted before incidents happen
- Organizations often overestimate their incident handling capability
- Bring in experts if you have to
- 6 Steps of Incident Handling
    - *Preparation*: happens before the incident occurs and is a critical part of the incident handling process.It makes sure the team is ready to handle the incident
    - *Identification*: is to determine whether an incident exists or not.Containment
    - *Eradication*: Eradication is to actually clean up the incident, stop whatever is causing the harm, and make sure it doesn't come back.
    - *Recovery*: is the place where things are returning to normal.
    - *Lessons Learned*:handling. It gives a team a chance to look back and seek ways to improve the process.

Anti-automation

Anti-automation & Anti-spam

- Automation against Web applications may cause security issues
- Various abuses, brute force, data scraping and DoS attack could happen

CAPTCHA

- Completely Automated Public Turing Test to tell Computers and Humans Apart
- It's Challenge-response to determine the user is not a computer
- Careful! CAPTCHA is Broken
    - Attacker could Pay for "skilled" worker to enter it manually
    - OCR (Optical Character Recognition) is another reasonably effective attack against CAPTCHA. By trying to recognize the shape of the objects, the OCR programs are able to solve most CAPTCHA

Rate Limit

- The rate limit might work for brute force and data scraping type attacks. The defender can set the page to be only accessible 3 times a minute, which would significantly slow down any brute force or scraping attempts.

Web Link Spam

- Most search engines rank sites by the number of links and relevance of links
- Spammer leverages that and spam other sites for links back to them
- Drug and adult sites have been active in spamming

- Mitigation
    - Check for referer
    - Blacklist (user-agent, XBL, open proxies)
    - Use JavaScript tricks
    - **SpamBam and Akismet:** are two projects that deal with comments spam. They are designed A for a blogging platform, but similar concepts can be applied to contact form spam.
    - Although it won't directly stop Web link spam., making the comment form or guestbook " with the "NOFOLLOW" meta tag injected can prevent content from being linked

Intrusion Detection in Web App

- Design Intrusion Detection into the Web application
- It has to be explicitly designed, intrusion detection does not automatically happen
- Traffic based approach: look into the network for attack
- Server based approach: is to detect attack at the server layer.
- Traffic Based Approach (1):
    - Decrypt network traffic & re-assemble HTTP traffic
    - Inspect the content against known attack patterns Can also match against anomaly behavior
- Traffic Based Approach (2)
    - Also monitor for odd traffic coming out of the Web server
    - Any traffic pattern that seems odd should be investigated cuz May be able to block inline on the network or send a reset packet
- Traffic Based Approach Pitfall
    - Too many evasive techniques to make sensor see a different traffic than the host

Application Intrusion Detection

- Looking for attack by application code
- Detect impossible cases in input validation and in a normal operations as well
- Pitfall
    - Takes time to develop.
    - Harder to incorporate up-to-date attack information
    - May miss lower level protocol attack

Honeypot

Honeypot

- Set designated variables and conditions to detect manipulation and hacking
- Any manipulation leads to shunning of user and alert administrators

- Some advanced setup might re-direct (implicitly or explicitly) the attacker to a virtual environment that looks exactly like the real application so the administrators can learn what the attacker wants to do.
- **Honeypot Ideas**
    - Session ID, which is just to trap the attacker
    - Additional hidden form value
    - Additional cookie value
    - Fake admin page in robots.txt
    - Web pages that are only linked by the comments or hidden fields

Safe Redirection

Safe Redirection

- A redirect mechanism that is opened for manipulation is called an open redirect
- Spammers often abuse these open redirects to make their e-mail look more legitimate. From a legitimate looking URL, the client is led to a site that is not what the client believes they are visiting.
- This is also very commonly used in phishing attacks
- Likely Places for Redirect
    - From the login page, to another page
    - Internal URL redirection is sometimes used throughout the site to get the user to the right place on the site.
    - Search engines are commonly used as open redirects, simply because search engines want to keep track of where the user went
- Safe Redirection Best Practice
    - Inspect whether internal or off-site, -If external, log and throw a warning
    - Signing redirect: The link should be signed with a hash. The hash is based on a secret key and the URL itself.
    - Check Referer: The redirection can then happen on the HTTP header. Before sending out the final redirection header, check to make sure the referer tag is from an internal source, not somewhere else on the Internet or blank.
    - Avoid full URL if possible
- Signing Redirection: involves Key generated by hashing the URL with a key

Web Security Testing

Web Security Testing

- Security Testing is necessary
- It should be done
    - During development: -Source code and runtime analysis of components
    - After development: Penetration test

Runtime Analysis

- Interact with the running Web application to assess security posture
- Examine behavior of the application: Focuses on the input and output of the application which is Usually performed after the code is developed

Code Analysis

- Review code line by line and reveal al security problems
- Sounds simple but can be very complicated
- Trace down the code path to reveal problems

Penetration Test

- Simulating an attack to analyze the security of a system
- Penetration testing is a very structured and methodical practice
- Penetration testing not only uncovers the software design and implementation flaws, but it also uncovers the configuration flaws
- Useful to Demonstrate impact of vulnerabilities

Types of Testing

- White box
- Black box
- Gray box

Internal vs. Externa (3rd Party)

- General rule - External = High risk
- Watch for regulatory requirements
- Perform internal tests before going external

Testing Challenges

- Inadequate testing knowledge
- Coverage
- Thoroughness
- Time
- Lack of manual validation

Pentest Frameworks

- OSSTM
- OWASP Testing Guide
- Frameworks specify higher level methodologies for testing. However Frameworks do not guarantee coverage or quality of the test

Reconnaissance

- Identify the target and its profile
- Gathering information on the target
- Configuration and settings info gathering

Mapping
- Determine the relationship between the pages or resources
- Spidering is the main technique used here
- Critical step, not as easy as it seems
- Determine the coverage of the test as well

Discovery
- Find potential vulnerabilities in the application
- Seek the weak points. To do that Test all points of entry according to the map and Test for all potential vulnerabilities

Exploitation
- Validate the vulnerability by actually attacking the application
- Always continue past initial exploitation
- Usually a time consuming phase and much more difficult than discovery

Tools
- Tools automate certain attack techniques, not the other way around
- Keep an arsenal of tools up-to-date
- Skilled pentesters often develop their own tools

Basic Toolkit of Web Application Pen testers
- **Webcrawler** - Spider the application to identify the relationship between pages and locate the input fields for testing.
- **Proxy tools** — Allow the tester to alter communication between browser and server.
- **Scanner** (infrastructure and application aware): Looking for commonly known vulnerability in the infrastructure and in the Web application.
- **Enumeration** — Locate the files that are not linked from the main group of files.
- **Brute Forcing tool** - The fuzzing tool and password cracker fit into this category.

Pentesting Best Practice (1)
- Have an overall game plan and stick to it
- Follow testing framework
- Be knowledgeable in the systems being tested
- Ensure good coverage of the site
- Go well beyond discovering standard known vulnerabilities
- Keep going after initial vulnerabilities are found
- Keep very good notes throughout the test and revisit suspicious areas
- Maintain good control of time

Pen test Reporting
- Stick to the truth, report on actual problems
- Write a decent executive summary
- Separate actual vulnerabilities from best practice recommendations

- Make useful and sensible recommendations
- Show proof of vulnerabilities, if possible; photos, screen cap, HTTP header