Session Hijacking
Module - 11

Types of Session Hijacking :

1. **Application Level** : Session ID Sniffing, Session Fixation, Session Donation, Session ID brute Force.
2. **Network Level** : Blind hijacking, Session Sniffing, IP Spoofing, UDP hijacking

# Attack Vector :

1. Man in the middle Attack : rogue wifi, DNS spoofin
2. Man in the Browser : Malware, browser add-ons
3. Brute Forcing : enumerating session IDs
4. Cross Site Scripting : Social Engineering
5. Session ID Leakage : referrer in outbound links, exposed log files

# The Impact of Session Hijacking :

The Victim can be impersonated by the attacker
The System believes the attacker is a legitimate user

# Session Persistence in Web Applications :

# The Stateless Nature of HTTP :
Each Assets require a new connection request

Session ID
Token
Auth Cookie

# Session Persistence in Cookies :

Good :
1. Cookies are automatically persisted on each Request
2. Cookies usually aren't logged in normal requests
3. The Session ID can be Persistent once the site is left or browser closed

Bad :
1. Cookies can be vulnerable to cross site scripting (XSS) attacks
2. You can only have one active session per browser

# Session Persistence in Website :

Good :
1. You can have multiple simultaneous Sessions in the one browser
2. This mechanism still works if the client doesn't accept the cookies

Bad :
1. The Session ID is lost once you leave the website
2. The Session ID will be sent in the referrer from an HTTP Source
3. The Session ID Will be stored In the cache or logs

# Session Persistence in Hidden Form Field :

Good :
1. You can have multiple simultaneous sessions in the one browser
2. The Mechanism still works if the client doesn't accept the cookies

Bad :
1. The Session ID is lost once you leave the website
2. Every request need to be POST (Session ID is Lost on GET)
3. You can't Secure embedded resources (i.e. videos in the page)

# Hijacking Sessions in Web Applications :

- Hijacking Cookies with Cross Site Scripting (XSS)

-> Fiddler is a http Proxy tool , Debugging for web application

HTTP Session Hijacking (Sometimes called "Sidejacking") is when attacker gets a hold of a user's cookie, allowing them to do anything the user can do on a particular Websites.

# Session Sniffing :

# Session Fixation :

# Brute Forcing Session IDs

# Network and Client Level Session Hijacking :

TCP SPOOFING :

# Blind Hijacking : Attacker unable to see the traffic bu it only predict the sequence number
# Man in the Middle :
# IP Spoofing : Doesn't require to guess the sequence number , Attacker created the source routed packets
Attacker need to desynchronizing the connection

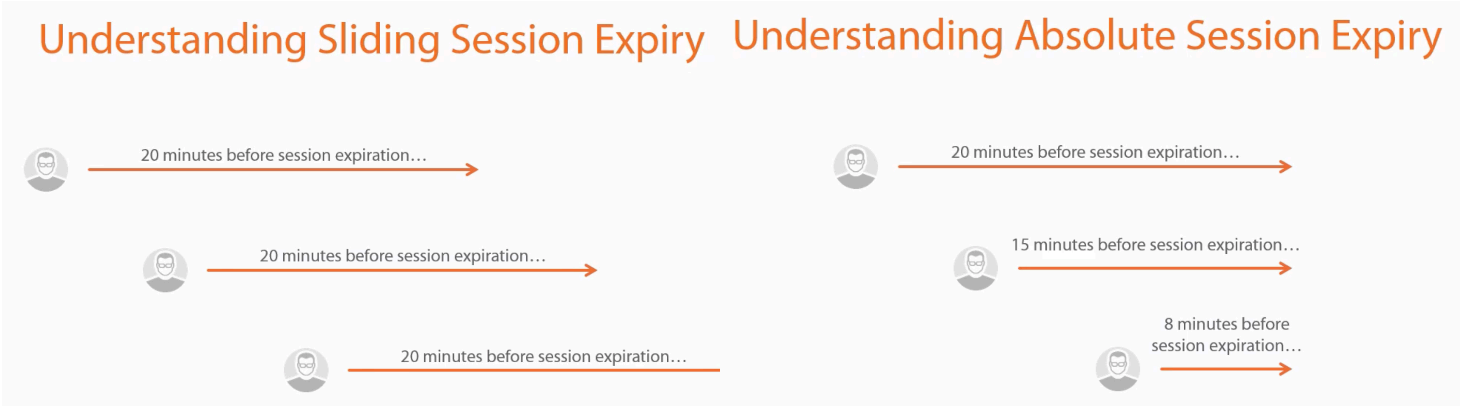
# UDP SPOOFING :

# UDP Hijacking :
There is no concept of desynchronizing of connection as TCP
as victim send any request … before the server respond the attacker responds back means their is a race conditions.

#Man in the Browser :

# Mitigation the Risk of Session Hijacking :

1. Use Strong Session ID
2. Keep session ids out of the url
    People will copy and paste URL
    They will be passed in the referred header
    They'll get logged at various points
3. Don't reuse session id for auth
    Session Fixation exploits reused session IDs
    Transfer the session to a new ID post-auth
    or don't attach the authid state to the session
4. Always Flag Session id cookie as HTTP Only
    Session IDs are used by the server to persists sessions
    There is no reason to access then from client script
    Flagging the cookie as HTTP Only makes it inaccessible via java script
5. Transport Layer Security :
    Consider Session IDs to be sensitive
    Sensitive data requires transport layer security
    Transport layer security mitigates multiple session hijacking risks
        A man in the middle sniffing the cookie
        DNS poisioning
        The referer header is not sent
6. Always Flag session ID Cookies as Secure
    The Session ID cookies should never be sent in the clear
    It can be flagged to ensure that can't inadvertently occur
7. Session Expiration and Using Session Cookies
    Session can only be hijacked while there's valid reason
    Expire the Cookie and the actual Session early
        But remember there's a usability aspect
    Don't "rehydrate" an expired session with the same ID


Understanding Sliding Session Expiry
20 minutes before session expiration…
20 minutes before session expiration…
20 minutes before session expiration…
Understanding Absolute Session Expiry
20 minutes before session expiration…
15 minutes before session expiration…
8 minutes before session expiration…

9. Encoursge user to log out
10. Re authentication Before key actions

The session hijacking risk should not mean credentials are exposed
if an authentication challenge is presented before an action, session hijacking is mitigated
This is also protects aginst attacker having direct access to the machine

# Automating Session Hijack Attacks :

Tools:

1. OWASP Zed Attack Proxy (ZAP)
2. BurpSuite
3. Netsparker
4. Juggernaut
5. Hunt
6. Ettercap