Chapter 4: Threat Modeling

→ You cannot build a secure system until you understand your threats

1. Threat model: a security-based analysis that helps determine the (highest level) security risks posed to an app *and* how attacks can manifest themselves
2. Threat models provide structure in terms of security to the design process
3. Threat models help formalize the way you think about security as related to an app
4. Most important skill in threat modeling: the ability to correctly question any and all assumptions about the data
5. A secure system cannot be built without threat evaluation (duh)
6. Benefits of threat modeling
    a. You will better understand your app
    b. You will find bugs
    c. New team members will get up to speed faster
    d. Other teams that work on your app will be better able to understand it
    e. Testers will be able to better test your app
7. Basic threat modeling process
    a. Assemble a (threat modeling) team: should include those with some security background
    b. Decompose the app: DFDs (identify processes, data stores, interactors, data flows), Activity diagrams (and other UML components)
    c. Determine threats
        i. Use STRIDE to categorize threats
            1. Spoofing: attacker poses as another (trusted) user or rogue server poses as a trusted one (examples include DNS cache poisoning)
            2. Tampering with data: malicious modification of the data
            3. Repudiation: user denies performing some action (like an online purchase) – non-repudiation is the ability to prove the transaction occurred
            4. Information Disclosure: data that should be hidden/protected is viewed in the open
            5. Denial of Service: app does not respond because it has crashed or is overloaded with requests
            6. Elevation of privilege: allows for access to restricted data and files – if an attacker "gets root" on your machine, you are "owned"
            7. Some threats interrelate
            8. Elevation of privilege is probably the worst (why?)
        ii. Other threat analysis discussion: http://www.cert.org/octave
        iii. Use threat trees to identify vulnerable points in app
            1. they describe the decision making process an attacker would go through to compromise a given component
            2. they help identify the most vulnerable areas of an app

3. vocabulary
    a. threat: a potential event that will have an unwelcome consequence if it becomes an attack
    b. vulnerability: a weakness such as a coding bug or design flaw
    c. attack: a malicious user takes advantage of a vulnerability to threaten an asset (something that is important to the app/system/user)

iv. aside: sniffers (network protocol analyzers)– used to monitor network traffic
    1. if you care about your data at all, you should never allow unencrypted transmission from a wireless router

d. Rank threats by decreasing order of risk (meaning list most important first!)
    i. DREAD can be used to calculate risk
        1. Damage potential: scale of 1-10 on how much damage can be done by the threat (elevation of privilege threats usually rate a 10)
        2. Reproducibility: how easy is it to perform attack again? (a race condition threat may be hard to reproduce)
        3. Exploitability: what effort and expertise is required to pull it off (race condition might be difficult)
        4. Affected users: what percentage of users of the app can be affected/impacted by the attack (server attacks can be much more severe in whom is affected)
        5. Discoverability: perhaps most difficult to determine (side note on vulnerability disclosures…)
        6. Add each of the 5 up and divide by 5 to see how bad on a scale of 1-10
    ii. There may be more than one way to perform an attack (end up in a place where exploit is possible) due to multiple paths – one path may be higher risk than another

e. Table relating STRIDE threat categories to items in a DFD

Threat Type	Affects Processes (transforms or manipulates data)	Affects Data Stores (stores temporary or permanent data)	Affects Interactors (input to system)	Affects Data Flows (flow from data stores, processes, or interactors)
S	Y		Y	
T	Y	Y		Y
R		Y	Y	Y
I	Y	Y		Y
D	Y	Y		Y
E	Y			

        i. Spoofing user – accessing credentials; spoofing process – replacing with rogue
        ii. Tampering with process – replacing existing binary image or patching existing one in memory
        iii. Information disclosure on processes – reverse engineering process to discover secret data
        iv. Interactors are not subject to information disclosure
        v. Dos cannot be directed at an interactor – must go through a data store, data flow, or process

    f. Threat examples
        i. Confidential, on-the-wire payroll data is viewed
        ii. Attacker uploads rogue web page and associated code
        iii. Attacker denies service to app
        iv. Payroll data is manipulated
        v. Elevation of privilege by leveraging the service client request process
        vi. Spoof computer executing process client request process
    g. Always consider most likely path of attack first (usually the path of least resistance)

8. How to respond to threats
    a. Do nothing!
    b. Inform the user
    c. Remove the problem
    d. Fix the problem
9. Threat mitigation techniques: know your technologies and where they can be effectively applied

Threat Type	**Mitigation Techniques**
Spoofing	Appropriate authentication Protect secret data Don’t store secrets
Tampering	Appropriate authorization Hashes Message authentication codes Digital signatures Tamper-resistant protocols
Repudiation	Digital signatures Time stamps Audit trails
Information Disclosure	Authorization Privacy-enhanced protocols Encryption Protect secrets Don’t store secrets
Denial of Service	Appropriate authentication Appropriate authorization

	Filtering Throttling Quality of service
Elevation of privilege	Run with least privilege

a. Authentication: the process by which an entity (principal: user, executable code, computer) verifies that another entity is who or what it claims to be
    i. requires evidence in the form of credentials (password, private key, biometrics)
    ii. many protocols (what is a protocol?) are available
        1. basic (insecure: password not protected – base64 encoded)
        2. digest (password does not travel from browser to server in clear text)
        3. forms-based (different form technologies provide different tools – know their shortcomings)
        4. passport (allows access to multiple sites via one login (https://accountservices.passport.net/ppnetworkhome.srf?vv=450&lc=1033)
        5. Kerberos (allows mutual authentication) http://www.ietf.org/rfc/rfc1510.txt
        6. ipsec (what is ipsec?) (authenticates servers only)
b. Authorization
    i. ACLs (rwx on *nix side) (ACEs are put together to make an ACL)
    ii. Privileges (user based)
    iii. IP restrictions (only allow access from certain IP addresses)
    iv. Server-specific permissions
c. Tamper-resistant and privacy-enhanced technologies
    i. SSL/TLS: uses message authentication codes (MACs) to provide data integrity – TLS is the latest version of SSL
    ii. IPSec
    iii. DCOM and RPCs support authentication, privacy, and integrity
    iv. EFS (encrypting file system): also provides tamper detection – win 2k and greater
d. Protect secrets: require user to provide secret data from memory as needed
e. Encryption, Hashes, MACs, and Digital Signatures
    i. MAC: message data and secret data are hashed – receiver calculates digest by hashing data with secret data – if same MAC is produced, transaction was secure
    ii. Hashing: passing data through a crypto function (hash or digest function) to produce a small (128 or 160 bits) value that uniquely identifies the data
    iii. Digital Signature: like a MAC, but secrets aren't used – data is hashed and then encrypted using private key known only by sender – recipient uses a public key associated with sender's private key to decrypt hash and then calculating the hash – same results verify data has not been tampered with and it came from a trusted source
f. Auditing: logging app activity

g. Filtering: inspecting data as it is received and rejecting bad packets
h. Throttling: limiting the number of requests on a system
i. Quality of Service: a set of components that allow you to provide preferential treatment of specific traffic
j. Least Privilege: …