Threat Modeling II

John E. Savage
Computer Science
Brown University
Source Material

Schneier on Attack Trees

• “Attack trees provide a formal, methodical way of describing the security of systems, based on varying attacks. Basically, you represent attacks against a system in a tree structure, with the goal as the root node and different ways of achieving that goal as leaf notes.” (Schneier, 1999)
Example of an Attack Tree for Opening a Safe
Attack Trees

• Attack trees are used to enumerate threats
  – Can use STRIDE, brainstorming, libraries

• Trees are usually project specific
  – One or more trees may be used

• A tree may model
  – A system component, such as a license server, or
  – An attacker’s goal, such as opening a safe
Where to Start Building Trees

• Start at the top
  – Determine what data flows across boundaries
  – Ask what can go wrong

• Iterate at the next level
  – Across levels
  – Across threat types (STRIDE, libraries, brainstorm)

• Clearly this process requires a good diagram describing data flows in a system
Constructing Attack Trees

• Create a root node modeling *component* or *goal*
  – Subnodes of a *component* model vulnerabilities
  – Subnodes of a *goal* model subgoals

• Nodes can be AND or OR nodes
  – AND node: all subnode actions/goals must occur
  – OR node: any subnode action/goal needs to occur
Creating Subnodes

• Identify subnode actions or goals
  – Use brainstorming or structured approach

• Examples of first-level subnode types
  – Attacking system via
    • Physical access, subverting software
    • People, process, or technology
  – Attacking a product during
    • Design, production, distribution, usage, after discard
Assessing Attack Trees

• **Completeness**
  – Does the tree captures all threats
  – If not, do you need other trees?
  – Ask about each node, “could this happen another way?”

• **Prune** the tree as you progress in your design
  – Ask if a threat has been mitigated or duplicated
  – Mark nodes: I = impossible, P = possible, but don’t delete

• **Readability**
  – Keep trees to a page
  – If necessary, explode a node into separate tree
  – Make tree notations simple and easy to read
Why Not Use Lists?

• An example of a list

  1. Attack voting equipment
     1.1 Gather knowledge
        1.1.1 From insider
        1.1.2 From components
     1.2 Gain insider access
        1.2.1 At voting system vendor
        1.2.2 By illegal insider entry

• AND and OR nodes preferable to linked list

• Outline format OK, but trees get more attention

Not clear whether 1. is AND or OR node
Same for subnodes 1.1 and 1.2, etc.
Physical Security Attack Tree

- Gather information
- Compromise Perimeter Security
  - Place authorized insider in plant
    - Blackmail / intimidate existing, cleared employee
    - Insert sympathizer into plant
  - Force Entry
- Sabotage pumps and/or coolant pipes
  - Cause electrical damage
  - Blow up pumps

AND node
OR node
AND node
Web Attack Tree

**Attack effect**

**Active components**

**Involved Standards**

**Triggering Properties**

- Goal: stop web application
- Component: web server
- Component: database engine
- Functionality:
  - TCP
  - HTTP
  - JavaScript
  - SQL

- same properties as on the left

- Triggering Properties:
  - amount of msg.
  - content of msg.
  - order of msg.
  - size of msg.
  - amount of req.
  - content of req.
  - order of req.
  - size of req.
  - amount of cmd.
  - content of cmd.
  - order of cmd.
  - size of cmd.
  - amount of qu.
  - content of qu.
  - order of qu.
  - size of qu.

(msg.=message; req.=requests; cmd.=commands; qu.=queries)
Attack DAG for Smart Grid
AFCE Fraud Tree

**Goal**

**Sub-goal**

**OR node**
Attack Libraries

• Libraries are useful in constructing attack trees
  – They can be thought provoking
  – Be mindful that they are not always complete

• Libraries differ in
  – Audience – e.g. security engineer, network operator
  – Detail – library should match level of detail needed
  – Structure – It helps to find attacks and classify them
  – Scope – e.g. web, network traffic
Checklists

• Very useful in preventing some problem types
• But beware of checklist security
• However, some vulnerabilities are so serious that it should behoove every security person to have a list of them and use it.
  – Example: Emergency room personal treating a patient with a knife wound did everything right except ask the patient for the size of the knife!
Diagramming Checklist

Can your team answer these question?

1. Is the story clear without changing the diagram?
2. Can it be told without “sometimes” or “also”?
3. Does the diagram indicate exactly where the software will make security decisions?
4. Does the diagram show all trust boundaries?
5. Are all UIDs, application roles and network interfaces covered?
6. Does the diagram reflect current or planned code?
7. Is it clear where the data goes and who uses it?
8. Are the processes moving data between stores visible?
Threat Checklist

Can your team answer these question?

1. Have we looked for all STRIDE threats?
2. Have we looked at every element of the diagram?
3. Have we looked at every data flow in the diagram?
Checklist to Validate Threats

Can your team answer these question?

1. Have we written down or filed a bug for each threat?
2. Is there a proposed/planned/implemented way to address each threat?
3. Do we have a test case per threat?
4. Has the software passed the test?
Profit from Experience

• It is useful to examine a similar system to learn what threats were found
  – A threat library can be assembled this way
• Since building a library is time intensive, try to use existing libraries
Mitre’s CAPEC Library

- CAPEC – Common Attack Pattern Enumeration and Classification (See [https://capec.mitre.org/](https://capec.mitre.org/))
  - A catalog of attack patterns and a comprehensive schema and classification taxonomy
- It classifies attacks by mechanism and domain
CAPEC Mechanisms

- Gather Information – e.g. port scan or side-channel
- Deplete Resources – e.g. TCP SYN flood attack
- Injection – e.g. SQL injection or Shellshock, Unix Bash shell injection attack
- Deceptive Interactions – e.g. increase bank funds, send diff to attacker
- Manipulate Timing and State – e.g. provide incorrect response to DNS lookup
- Abuse of Functionality – e.g. passing local file name when URL intended
- Probabilistic Techniques – e.g. fuzzing
- Exploitation of Authentication – e.g. misuse of web session variables
- Exploitation of Authorization – e.g. cross-site scripting
- Manipulate Data Structures – e.g. buffer overflow, integer overflow
- Manipulate Resources – e.g. change library routines, play with logs
- Analyze Target – e.g. reverse engineering, break short crypto keys
- Gain Physical Access – e.g. listen to keystrokes, side-channel attacks
- Malicious Code Execution – e.g. obvious
- Alter System Components – e.g. inject trojans onto chips
- Manipulate System Users – e.g. social engineering
CAPEC Attack Domains

- Social Engineering
- Supply Chain
- Communications
- Software
- Physical Security
- Hardware

The CAPEC library has about 550 vulnerabilities!
OWASP Top Ten

- P1  Web Application Vulnerabilities
- P2  Operator-sided Data Leakage
- P3  Insufficient Data Breach Response
- P4  Insufficient Deletion of personal data
- P5  Non-transparent Policies, Terms and Conditions
- P6  Collection of data not required for primary purpose
- P7  Sharing of data with third party
- P8  Outdated personal data
- P9  Missing or Insufficient Session Expiration
- P10 Insecure Data Transfer