Chapter 1: The Need for Secure Systems Overview

Secure software development requires you know how to design, build, test and document with security in mind. Secure software is able to withstand the most malicious attacks. Secure software is by nature robust software.

1. **Trust**
    a. What is trust?
    b. Who can we trust?
    c. How does trust translate into secure code?
    d. Does insecure code violate trust?

2. **CIA Triad**
    a. Confidentiality - Only authorized users and processes should be able to access or modify data
        i. Data Confidentiality – Data is not exposed
        ii. Privacy - Refers to individuals and what of their personal data is collected/shared
    b. Integrity - Data is protected from unauthorized changes to ensure that it is reliable and correct
        i. Data Integrity – Ensures data is only changed by those authorized
        ii. System Integrity – System performs as intended and unimpeded
    c. Availability - Authorized users have access to the systems and the resources they need

3. **CIA Triad** - is essential
    a. It provides vital security features
    b. It helps in avoiding compliance issues
    c. It ensures business continuity
    d. It prevents reputational damage to the organization

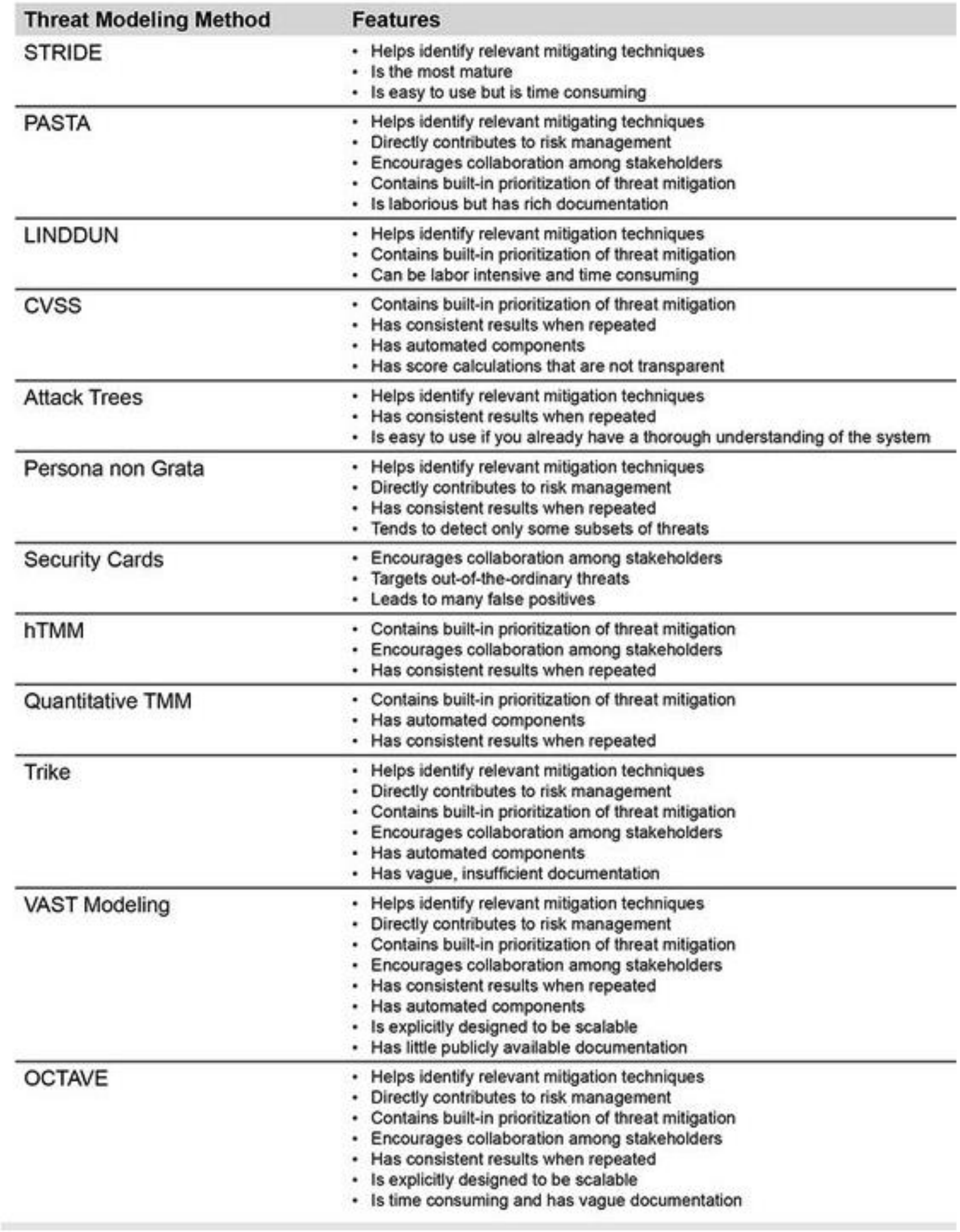
4. **Software Security** - All software is vulnerable to attack, regardless of the purpose for which it was developed.
    a. Never assume your app will be run in only a few (or one) environment
    b. Never assume your app will not persist

5. **Software Security** - Security should be a top priority, not an afterthought based on an attack
    a. If your software is attacked and is vulnerable and this knowledge goes public, what are the consequences for your company?
    b. Security after the fact is expensive
        i. Coordination of the fix
        ii. Developers must find vulnerable code (think how difficult this could be with a large product)
        iii. Code must be fixed
        iv. Code must be tested (and regression tested)
        v. Setup of fix must be tested (patch must be tested)
        vi. International versions may be necessary
        vii. Code must be digitally signed for authenticity
        viii. Code must be posted to website
        ix. Documentation for fix must be created
        x. Public relations
        xi. Bandwidth costs for download
        xii. Loss of productivity
        xiii. Customers must apply fix
            1. aside: patches, those who apply and those who don’t and the ramifications
        xiv. Lost revenue due to bad press

6. **Secure Software** – Produces quality products
    a. Well tested
    b. Robust
    c. Fail securely

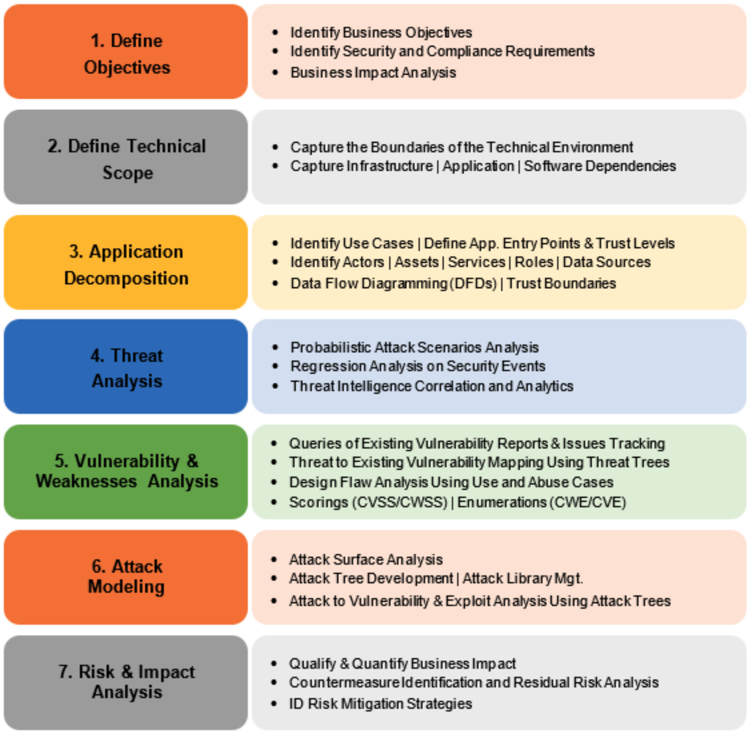
7. **Bug Fixes** - Should a discovered bug always be fixed ASAP?
    a. If it is serious, entire program and/or system on which it runs can be compromised
    b. Costs associated with security after the fact!
    c. Some products are shipped with known bugs due to time-to-market pressure

8. **Security Dilemma** - Attacker's Advantage / Defender's Dilemma
    a. Defender must defend all points; the attacker can choose the weakest point
        i. Threat Modeling Assessment (TMA) - Models what points must be defended



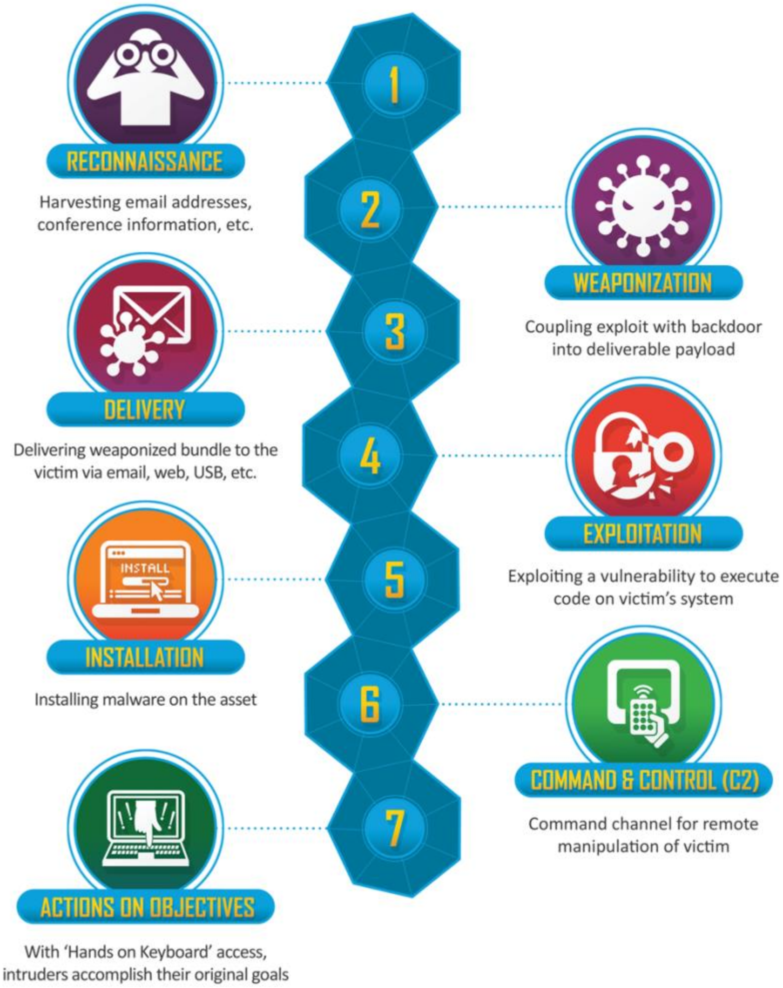
Threat Modeling Method	Features
STRIDE	• Helps identify relevant mitigating techniques • Is the most mature • Is easy to use but is time consuming
PASTA	• Helps identify relevant mitigating techniques • Directly contributes to risk management • Encourages collaboration among stakeholders • Contains built-in prioritization of threat mitigation • Is laborious but has rich documentation
LINDDUN	• Helps identify relevant mitigation techniques • Contains built-in prioritization of threat mitigation • Can be labor intensive and time consuming
CVSS	• Contains built-in prioritization of threat mitigation • Has consistent results when repeated • Has automated components • Has score calculations that are not transparent
Attack Trees	• Helps identify relevant mitigation techniques • Has consistent results when repeated • Is easy to use if you already have a thorough understanding of the system
Persona non Grata	• Helps identify relevant mitigation techniques • Directly contributes to risk management • Has consistent results when repeated • Tends to detect only some subsets of threats
Security Cards	• Encourages collaboration among stakeholders • Targets out-of-the-ordinary threats • Leads to many false positives
hTMM	• Contains built-in prioritization of threat mitigation • Encourages collaboration among stakeholders • Has consistent results when repeated
Quantitative TMM	• Contains built-in prioritization of threat mitigation • Has automated components • Has consistent results when repeated
Trike	• Helps identify relevant mitigation techniques • Directly contributes to risk management • Contains built-in prioritization of threat mitigation • Encourages collaboration among stakeholders • Has automated components • Has vague, insufficient documentation
VAST Modeling	• Helps identify relevant mitigation techniques • Directly contributes to risk management • Contains built-in prioritization of threat mitigation • Encourages collaboration among stakeholders • Has consistent results when repeated • Has automated components • Is explicitly designed to be scalable • Has little publicly available documentation
OCTAVE	• Helps identify relevant mitigation techniques • Directly contributes to risk management • Contains built-in prioritization of threat mitigation • Encourages collaboration among stakeholders • Has consistent results when repeated • Is explicitly designed to be scalable • Is time consuming and has vague documentation

1. STRIDE (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege)
2. PASTA (Process for Attack Simulation and threat analysis)



1. Define Objectives	• Identify Business Objectives • Identify Security and Compliance Requirements • Business Impact Analysis
2. Define Technical Scope	• Capture the Boundaries of the Technical Environment • Capture Infrastructure | Application | Software Dependencies
3. Application Decomposition	• Identify Use Cases | Define App. Entry Points & Trust Levels • Identify Actors | Assets | Services | Roles | Data Sources • Data Flow Diagramming (DFDs) | Trust Boundaries
4. Threat Analysis	• Probabilistic Attack Scenarios Analysis • Regression Analysis on Security Events • Threat Intelligence Correlation and Analytics
5. Vulnerability & Weaknesses Analysis	• Queries of Existing Vulnerability Reports & Issues Tracking • Threat to Existing Vulnerability Mapping Using Threat Trees • Design Flaw Analysis Using Use and Abuse Cases • Scorings (CVSS/CWSS) | Enumerations (CWE/CVE)
6. Attack Modeling	• Attack Surface Analysis • Attack Tree Development | Attack Library Mgt. • Attack to Vulnerability & Exploit Analysis Using Attack Trees
7. Risk & Impact Analysis	• Qualify & Quantify Business Impact • Countermeasure Identification and Residual Risk Analysis • ID Risk Mitigation Strategies

3. Cyber Kill Chain



Harvesting email addresses, conference information, etc.

Coupling exploit with backdoor into deliverable payload

Delivering weaponized bundle to the victim via email, web, USB, etc.

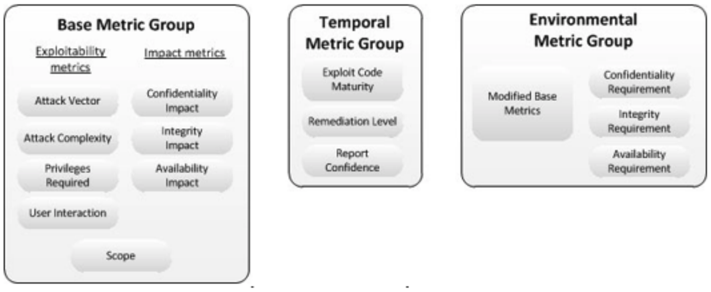
Exploiting a vulnerability to execute code on victim's system

Installing malware on the asset

Command channel for remote manipulation of victim

With 'Hands on Keyboard' access, intruders accomplish their original goals

        4. MITRE ATT&CK (attack.mitre.org)
        5. CVSS (Common Vulnerability Scoring System)
            a. NVD (National Vulnerability Database)
            b. CVS (Common Vulnerability Enumeration) - a dictionary of publicly known information security vulnerabilities and exposures



    b. Defender can defend against only known attacks?
    c. Attacker can probe for unknown vulnerabilities
    d. Defender must be constantly vigilant
    e. Attacker can strike at will
    f. Defender must play by the rules
    g. Attacker can play dirty

9. **Malicious Software** - Refers to any malicious program that causes harm to a computer system or network.
    a. Malware - Collective name for malicious software variants
        i. Worms
        ii. Trojans
        iii. Viruses
        iv. Bots
        v. Ransomware
        vi. Spyware
    b. Developed to cause extensive damage to data and systems or to gain unauthorized access to a network.