CTMSCS S1 P3: Web Application Security

Unit 2. Web Application Security Vulnerability Terminology

Prepared By – Dr. Jay B. Teraiya

Supported By – Dr. Digvijaysinh Rathod

Agenda

- Introduction
- What is VAPT?
- Methodology for VAPT
- Common Issues in VAPT
- False Positive and False Negative
- Known and Unknown vulnerabilities
- CVE (Common Vulnerabilities and Exposures)
- Common Weakness Enumeration
- Common Vulnerability Scoring System
- Threat Modelling – STRIDE – DREAD
- Secure Coding Practice

Introduction

- **SIMPLIFIED ATTACK LIFECYCLE**
  - **Initial Reconnaissance**: It has two main steps, Selection of the target and Research of the Target
  - **Penetration**: The intruder uses certain methods to compromise the target.
  - **Gaining Foothold**: Maintain foothold of the compromised system. It is generally achieved by setting up a backdoor so that the machine can be accessed later.

Introduction

- **SIMPLIFIED ATTACK LIFECYCLE**

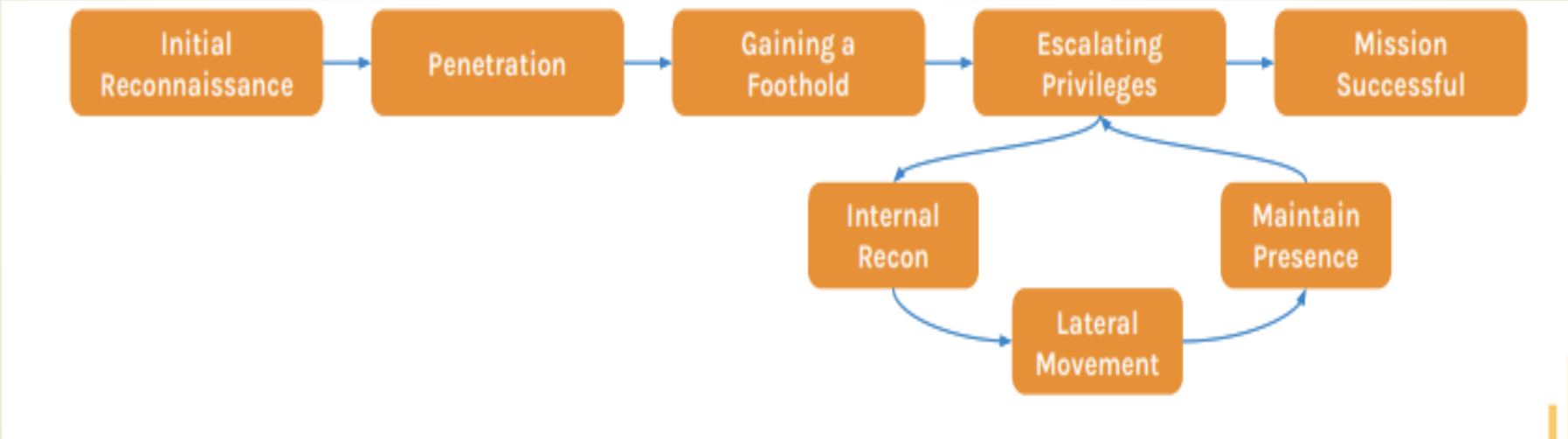
  - **Escalate Privileges:** The intruder will obtain a higher level of access to the compromised machine by multiple methods with the aim of obtaining the administrator login.

  - **Lateral Movement:** The intruder generally does not find the desired information on the **first machine he has compromised**. The attacker now tries to expand the exploitation process to other systems within the same network. If the intruder is caught during lateral movement, we get the exact intent of the intruder or the information he might be seeking.

Introduction

- **SIMPLIFIED ATTACK LIFECYCLE**

  - **Maintain Presence:** The intruder will ensure remote access to the complete environment by installing multiple variants of malware backdoors.



What is VAPT ?

- A form of stress testing that exposes **weaknesses or flaws in a computer** system
- The art of finding an **Open Door**
- A valued Assurance Assessment tool
- VAPT can be used to find flaws in
  - Specifications, Architecture, Implementation, Software, Hardware, and many more..

What is VAPT ?

- Vulnerability assessment is the process of **identifying, quantifying, and prioritizing** the vulnerabilities in a system.

- The penetration test is an attack on a computer system that looks for **security weaknesses**, **potentially gaining access to the computer's features and data.**

- Typically VAPT is performed using three approaches
  - Blackbox
  - Graybox
  - Whitebox

What is VAPT ?

- **BLACKBOX TESTING**

- Also known as “Zero-Knowledge” testing, the tester needs to acquire Knowledge and Penetrate.
- Acquire Knowledge using tools or Social Engineering Techniques.
- Publicly available information may be given to the penetration tester
- **Benefit:**
  - Intended to closely **replicate the attack made by an outsider** without any information of the system. This kind of testing can give an insight into the security robustness when under attack by script kiddies.

What is VAPT ?

- **WHITEBOX TESTING**
- Also known as “Internal Testing”, testers are given full information about the target system they are supposed to attack.
- Information includes:
  - Technology overviews
  - Data flow and network diagrams
  - Code snippets
- **Benefits:**
  - Reveals more vulnerabilities and maybe faster
  - Compared to replicating an attack from a criminal hacker who knows the company’s infrastructure well.
  - Checks robustness against internal threats.

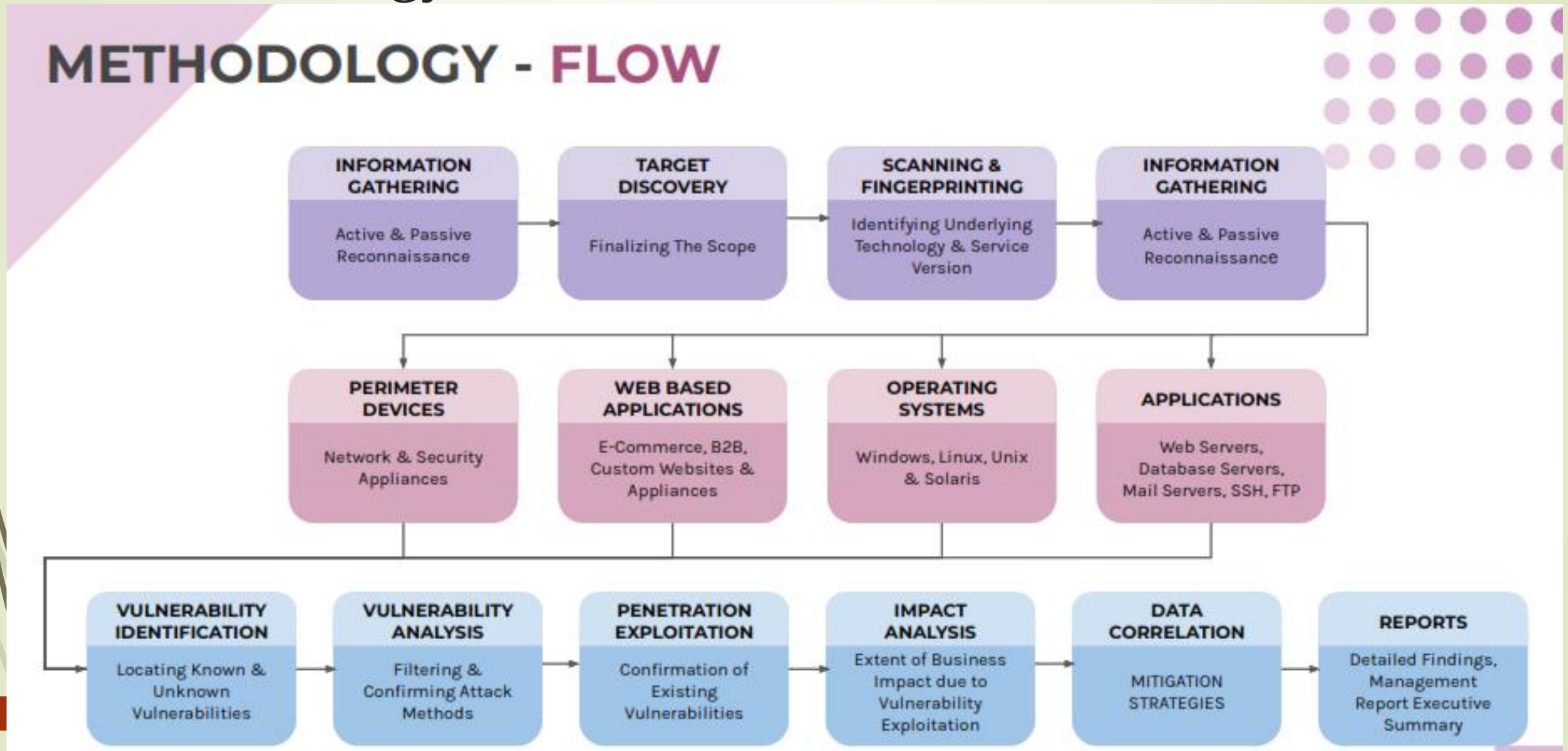
What is VAPT ?

- **GRAYBOX TESTING**
- This usually is **a combination of black-and-white testing**
- This testing aims to search for the defects, if any, due to improper structure or improper usage of applications.
- The tester simulates **an inside Employee**. The tester is given an account on the internal network and standard access to the network. This test assesses internal threats from employees within the company.
- **Benefit:**
  - Takes the straightforward black-box testing technique and combines it with the code-targeted systems in white-box testing.

Methodology for VAPT

- It is intended to reveal flaws in the security mechanisms, protect data, and maintain functionality as intended.
- The methodology is derived from the following best practices:
  - OWASP, OSSTMM, and WASC security guidelines
  - Business logic vulnerability verification
  - False positive elimination
  - Utilization of automated commercial, proprietary, and other industry-leading tools
  - Manual testing for identification and verification of critical and exploitable vulnerabilities
  - Reassessment to ensure all gaps are fixed.

Methodology for VAPT



Methodology for VAPT

METHODOLOGY - OUTCOMES

Methodology	Expected Outcome
Information Gathering 1. The point of this exercise is to find the number of reachable systems to be tested without exceeding the legal limits. 2. In this parameter, no intrusion is being performed directly on the systems. 3. The hosts discovered later may be inserted in the testing as a subset of the defined testing with concurrence of the Client's internal security team.	▪ Domain Names ▪ Server Names ▪ IP Addresses ▪ Network Map ▪ ISP / ASP information ▪ System and Service Owners ▪ Possible test limitations
Port Scanning 1. Port scanning is the invasive probing of system ports on the transport level. 2. This parameter is to enumerate live or accessible Internet services as well as penetrating the firewall to find additional live systems.	▪ Open, closed or filtered ports ▪ IP addresses of live systems ▪ List of discovered tunneled and encapsulated protocols ▪ List of discovered routing protocols supported ▪ Active services
Operating System Fingerprinting 1. System fingerprinting will be done for active probing of system for responses that can distinguish unique systems to operating system and version level.	▪ OS Type ▪ Other information such as uptime

Methodology for VAPT

METHODOLOGY - OUTCOMES

Methodology	Expected Outcome
Services Fingerprinting 1. This is the active examination of the application listening behind the service. In certain cases more than one application exists behind a service where one application is the listener and the others are considered components of the listening application. 2. A good example of this is PERL installed for use in a Web application. In that case the listening service is the HTTP daemon and the component is PERL.	▪ Service Types ▪ Service Application Type and Patch Level
Vulnerability Scanning 1. Testing for vulnerabilities using commercial tools to determine existing holes and system patch level.	▪ List of system vulnerabilities ▪ Type of application or service by vulnerability ▪ Patch levels of systems and applications ▪ List of possible denial of service vulnerabilities
Denial of Service Testing 1. Denial of Service (DoS) is a situation where a circumstance, either intentionally or accidentally, prevents the system from functioning as intended. 2. Ascertain whether the system functions exactly as designed and handles the load, scope, or parameters being imposed upon it.	▪ List weak points in the Internet presence including single points of failure ▪ List system behaviors to heavy use ▪ List of systems/ applications susceptible to DoS vulnerability
Buffer Overflow 1. Buffer overflow attacks are used to exploit specific vulnerabilities in OS and applications by giving inputs that are longer than defined memory buffers	Administrative access to the vulnerable servers/ protected resource

Methodology for VAPT

VAPT

VULNERABILITY ASSESSMENT

- Compliance-based reports (ISMS, PCI, HIPAA, NIST and SOX)
- Customizable, multi-view reports that make the most of existing security investment
- Internal and external vulnerability scans
- Best practices (OWASP, ITIL, OSSTMM and ISO 27001 standard)
- Provide on-demand proactive vulnerability management for organizations
- Bring visibility, awareness and consistency to your organization
- Track asset ownership, pinpoint rogue devices, and view detailed asset discovery and profile reporting
- Reduce investment in tools and technology
- Comprehensive solutions and countermeasures to mitigate identified vulnerabilities

PENETRATION TESTING

- Executive summaries (jargon-free, true executive-level summaries and action plan)
- Identify technical and logical vulnerabilities such as SQL injection, cross site scripting, I/O data validation, exceptio management, etc.
- Impact analysis of the identified vulnerabilities
- Findings and recommendations to improve security postures
- Knowledge transfer to clients' internal security team
- Reduced investment in employing full time security analyst, tools and technology
- Part of an overall risk management solution that addresse the audit requirement of policy & compliance frameworks such as ISO 27001, SOX, HIPAA, PCI etc.

Common Issues in VAPT

Information Gathering
- Bile-Suite
- Cisco torch
- SpiderFoot
- W3af
- Maltego
- In-House sdFinder

Privilege Escalation
- Cain & Abel
- OphCrack
- Fgdup
- Nipper
- Medusa
- Hydra
- Ncrack

Social Engineering
- Social-Engineering Toolkit (SET)
- Firecat
- People Search
- Hoxhunt

Application Security Assessment
- AppCheck
- Tenable / Qualys
- Burp Suite
- Sandcat
- Greenbone
- W3af
- Nikto
- Paros
- OWASP ZAP
- SonarQube

Port Scanning
- Nmap
- Amap
- hPing

Exploitation
- Saint
- SQL Ninja
- SQL Map
- Inguma
- Metasploit

Network & System Vulnerability
- Assessment
- Metasploit
- Nessus
- SAINT
- Inguma
- SARA
- Nipper
- GFI
- Safety-Lab
- Firecat
- OWASP CLASP
- Themis

Commercial Tools
- Rapid7
- Qualys
- Tenable Nessus
- F-secure Radar
- Fortify WebInspect
- Fortify SCA

Common Issues in VAPT

OWASP Top 10

- SQL Injection
- Cross Site Scripting (XSS)
- Broken Authentication and Session Management
- Insecure Direct Object References
- Cross Site Request Forgery (CSRF)
- Security Misconfiguration
- Insecure Cryptographic Storage
- Failure to Restrict URL Access
- Insufficient Transport Layer Protection
- Invalidated Redirects and Forwards

Insecure Configuration

- SSL configuration
- Directory Listing Enabled
- Directories with executable permission enabled
- Directories with write permissions enabled
- Insecure (TRACE / DELETE / PUT / HTTP) Method Enabled

Business Logic

- Abuse of functionality
- Insufficient Process Validation
- Information Leakage
- Predictable Resource Location and Insufficient Authorization
- Transaction details manipulation
- Bypass payment process validation
- Weak password recovery validation
- User Account Hijack
- Escalation of user privilege

Other Vulnerabilities

- Server/service fingerprinting
- Default passwords
- Backup / Sensitive files Security Vulnerability
- Code execution Vulnerability
- Directory Traversal
- Local / Remote File inclusion
- Path disclosure
- Possible sensitive files
- Sensitive data not encrypted
- Source code disclosure

Authentication

- Password guessing
- Password cracking
- Bypass authentication
- Session Id prediction
- Cryptographic strength validation
- Cookie tampering

WASC Classification

- Brute Force
- Buffer Overflow
- Credential / Session Prediction
- Fingerprinting
- HTTP Response Splitting
- Integer Overflows
- Null Byte Injection
- Session Fixation
- Server Misconfiguration

False Positive and False Negative

- False positives and false negatives are two important aspects of the VAPT process.
- **A false positive** is when vulnerability does not exist but gets reported.
- **A false negative** is when vulnerability exists but is not reported.
- **Burpsuite, ZAP, and Metasploit** are commonly used tools for conducting PT.

Known and Unknown vulnerabilities

- **Known vulnerabilities** are those that have been publicly reported and have a unique identifier, such as a **CVE (Common Vulnerabilities and Exposures)** number.
- These vulnerabilities are usually easier to detect and patch, as some databases and sources provide information and solutions.
- To scan for known vulnerabilities, you can use tools that compare your system or network configuration with the latest vulnerability data, such as **Nmap, Nessus, or OpenVAS**.
- Depending on your needs and goals, these tools can perform different types of scans, such as **port, service, or vulnerability scans**.

Known and Unknown vulnerabilities

- **Unknown vulnerabilities** have not been disclosed or discovered yet and, therefore, have no identifier or reference.

- These vulnerabilities are harder to detect and exploit as they require more analysis and creativity. To scan for unknown vulnerabilities, you can use tools that test your system or network for flaws or weaknesses, such as **Metasploit, Burp Suite, or ZAP**.

- Depending on the target and the attack vector, these tools can perform different types of tests, such as fuzzing, injection, or brute force.

CVE (Common Vulnerabilities and Exposures)

- CVE (Common Vulnerabilities and Exposures) is a list of publicly known cybersecurity vulnerabilities and exposures that provides a **standardized reference method for identifying and tracking vulnerabilities** in software and hardware systems. Here are some key points regarding
  - **Identification**: CVE assigns a **unique identifier** to each vulnerability or exposure, known as a **CVE ID**. This ID format includes the prefix "**CVE**" followed by the **assignment year and a unique number** (e.g., **CVE-2022-1234**).
  - **Standardization**: CVE aims to standardize the names for vulnerabilities across various organizations and products, allowing easier information sharing and coordination in the cybersecurity community.

CVE (Common Vulnerabilities and Exposures)

- **Centralized Database:** The CVE database is maintained by the MITRE Corporation, a not-for-profit organization. It is a **centralized repository** for information on **vulnerabilities, including descriptions, impact assessments, and solutions.**

- **Publicly Accessible:** The CVE List is **publicly accessible**, allowing **security researchers, vendors, and organizations to search and reference vulnerabilities**. This transparency facilitates collaboration and helps organizations prioritize security efforts.

- **Importance in Security Practices: CVE identifiers** are widely used in security practices such as **vulnerability management, patch prioritization, and risk assessment**. Security teams use CVE IDs to track and address vulnerabilities in their systems.

CVE (Common Vulnerabilities and Exposures)

- **Link to Security Advisories and Patches:** CVE entries typically include **references to security advisories, patches**, or mitigation measures provided by vendors. This information helps users identify and implement solutions to mitigate the risks associated with the vulnerabilities.

- **Lifecycle**: Each CVE entry undergoes a lifecycle, from **initial discovery** or disclosure through the coordination of information between vendors, researchers, and the CVE program to eventual resolution or mitigation.

- **Severity Assessment**: CVE entries often **include severity ratings**, such as the **Common Vulnerability Scoring System (CVSS)** score, which helps users assess the potential impact of a vulnerability on their systems.

CVE (Common Vulnerabilities and Exposures)

- **Ongoing Updates:** The CVE List **continuously updates** as new vulnerabilities are **discovered, disclosed, and addressed**. Regular monitoring of the CVE database is essential for staying informed about emerging threats and vulnerabilities.

- **Integration with Security Tools:** Many security tools and platforms integrate with the **CVE database** to provide automated vulnerability **scanning, detection, and remediation capabilities**.

- CVE promotes cybersecurity awareness, collaboration, and risk management across the global IT ecosystem.

CVE (Common Vulnerabilities and Exposures)

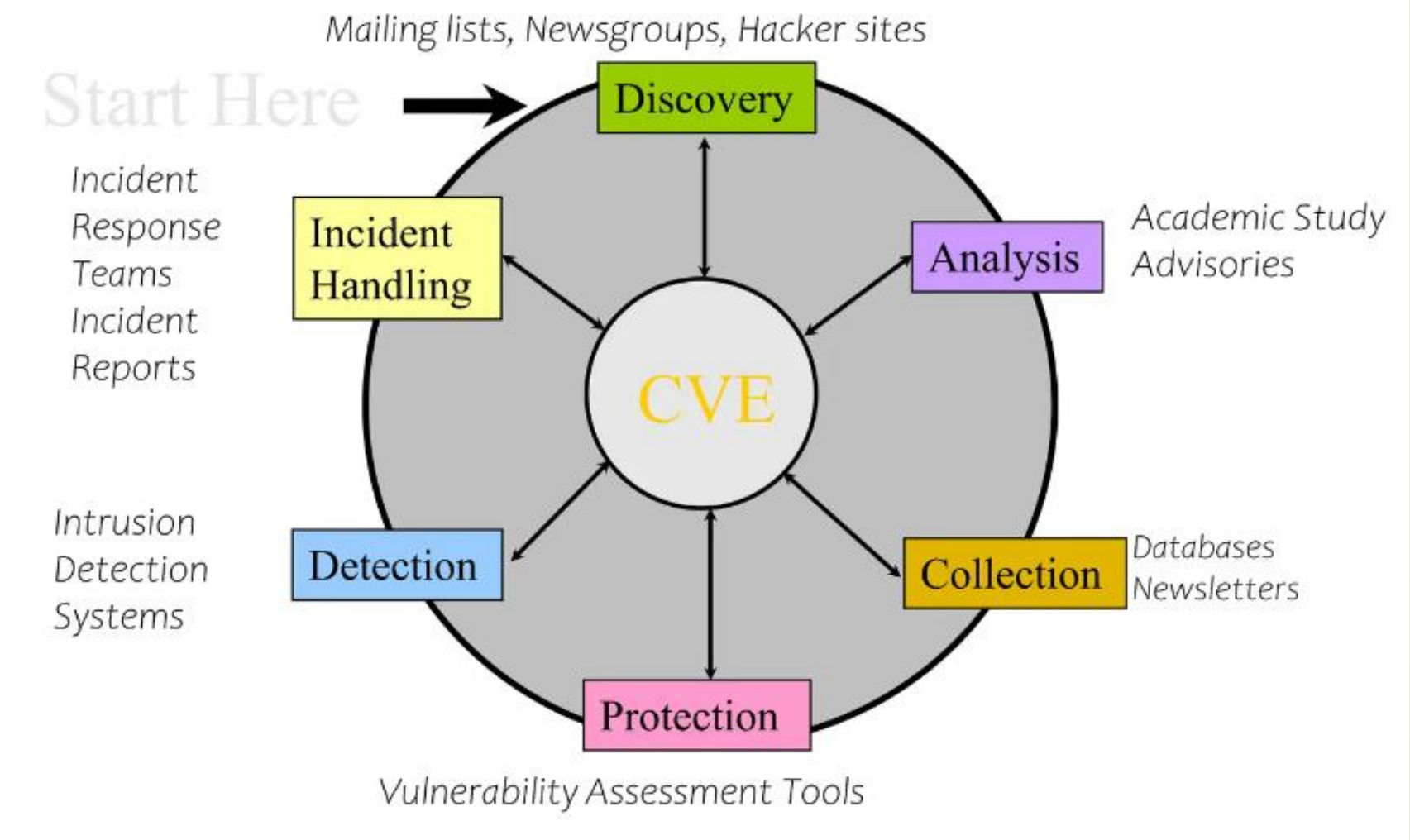
What is a CVE

- **Common Vulnerabilities and Exposures** (CVE®) is a list of common identifiers for publicly known cyber security vulnerabilities.

- Use of "CVE Identifiers (CVE IDs)," which are assigned by CVE Numbering Authorities (CNAs) from around the world,
  - ensures confidence among parties when used to discuss or share information about a **unique software vulnerability**,
  - provides a **baseline for tool evaluation**, and
  - enables **data exchange for cyber security automation**.

✓ https://cve.mitre.org/ https://www.cvedetails.com/

CVE (Common Vulnerabilities and Exposures)



CVE (Common Vulnerabilities and Exposures)

Common Vulnerabilities and Exposures (CVE): One Common Language

Name	Description
CVE-1999-0003	ToolTalk (rpc.ttdbserverd) buffer overflow
CVE-1999-0006	Buffer overflow in qpopper
CVE-1999-0067	Shell metacharacters in phf
CVE-1999-0344	Windows NT debug-level access bug (a.k.a. Sechole)

- Lists all publicly known security problems
- Assigns **unique identifier** to each problem
- Remains **independent** of multiple perspectives
- Is **publicly open** and shareable
- **Community-wide effort** via the CVE Editorial Board

CVE (Common Vulnerabilities and Exposures)

Addressing Common Misconceptions of CVE

- Not a full-fledged vulnerability database
  - Simplicity avoids competition, limits debate
  - Intended for use by vulnerability database maintainers
- Not a taxonomy or classification scheme
- Focuses on vulnerabilities instead of attacks
  - Does not cover activities such as port mapping
- Not just “vulnerabilities” in the classical sense
  - Definitions of “vulnerability” vary greatly
  - “Exposure” covers a broader notion of “vulnerability”
- Competing vendors are working together to adopt CVE

CVE (Common Vulnerabilities and Exposures)

CVE Editorial Board

- Members from 25 different organizations including researchers, tool vendors, response teams, and end users
- Mostly technical representatives
- Review and approve CVE entries
- Discuss issues related to CVE maintenance
- Monthly meetings (face-to-face or phone)
- Publicly viewable mailing list archives

CVE (Common Vulnerabilities and Exposures)

Adding new entries

- Board member submits raw information to MITRE
- Submissions are grouped, refined, and proposed back to the Board as candidates
  - Form: CAN-YYYY-NNNN
  - Strong likelihood of becoming CVE-YYYY-NNNN
  - Delicate balance between timeliness and accuracy
- Board reviews and votes on candidates
  - Accept, modify, recast, reject, reviewing
- If approved, the candidate becomes a CVE entry
- Entry is included in a subsequent CVE version
  - Published on CVE web site
- Entries may later be modified or deprecated

CVE (Common Vulnerabilities and Exposures)

- CVE is:

  - One name for one vulnerability or exposure
  - One standardized description for each vulnerability or exposure
  - A dictionary rather than a database
  - How disparate databases and tools can "speak" the same language
  - The way to interoperability and better security coverage
  - A basis for evaluation among tools and databases
  - Free for public download and use
  - Industry-endorsed via the CVE Numbering Authorities, CVE Board, and CVE-Compatible Products

CVE (Common Vulnerabilities and Exposures)

CVE Identifier

- CVE Identifiers (also referred to by the community as "CVE IDs," "CVE entries," "CVE names," "CVE numbers," and "CVEs") are ***unique***, ***common identifiers*** for publicly known cyber security vulnerabilities.

- Each CVE Identifier includes the following:
  - **CVE identifier number** with four or more digits in the sequence number portion of the ID (i.e., "CVE-1999-0067", "CVE-2014-12345", "CVE-2016-7654321").
  - **Brief description** of the security vulnerability or exposure.
  - Any **pertinent references** (i.e., vulnerability reports and advisories).
  - CVE Identifiers are used by information security product/service vendors and researchers as **a standard method for identifying vulnerabilities** and for **cross-linking with other repositories** that also use CVE Identifiers.

CVE (Common Vulnerabilities and Exposures)

State of CVE IDs

- **RESERVED ->** it has been reserved for use by a CVE Numbering Authority (CNA)
  1. the CVE is **populated** with details and **published on the CVE List**,
  2. it will become **available** in the **U.S. National Vulnerability Database (NVD)**
  3. As one of the final steps in the process, the **NVD Common Vulnerability Scoring System (CVSS) scores for the CVE ID** are assigned by the **NIST NVD team**.
- **DISPUTED** -> When one party disagrees with another party's assertion that a particular issue in software is a vulnerability
- **REJECT** -> Not accepted as CVE

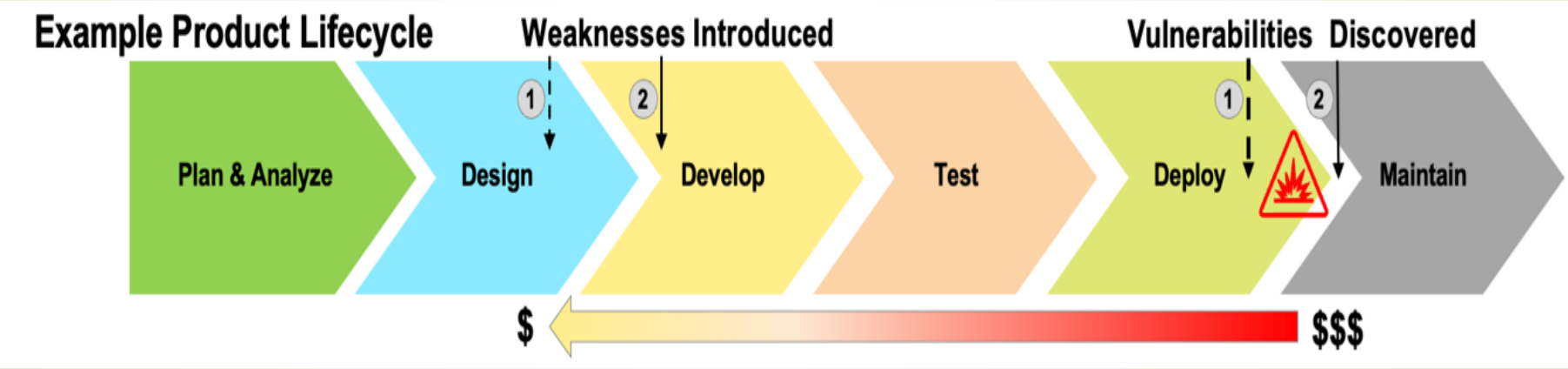
Common Weakness Enumeration

- Common Weakness Enumeration (CWE™) is a community-developed list of **common software and hardware weaknesses**.
- A "**weakness**" is a condition in a **software, firmware, hardware, or service** component that, under certain circumstances, could contribute to introducing **vulnerabilities**.
- The **CWE** List and associated classification taxonomy identify and describe weaknesses regarding CWEs.
- Knowing the weaknesses that result in **vulnerabilities** means **software developers, hardware designers, and security architects** can eliminate them before deployment when it is much easier and cheaper to do so.

Common Weakness Enumeration

- The CWE List is fully searchable and may be viewed or downloaded.

- There is also the **CWE REST API** to make CWE content available to community applications and websites more conveniently.



Common Weakness Enumeration

- Using the CWE List
  - The **Software Development (CWE-699)** view organizes items by concepts frequently used or encountered during software development.
  - The **Hardware Design** view organizes weaknesses around concepts that are frequently used or encountered in hardware design and
  - **Research Concepts** (CWE-1000) facilitates weakness-type research by organizing items by behaviors.
- **Other views provide insight into** a certain domain or use cases, such as **weaknesses introduced during design or implementation**.
- Weaknesses with indirect security impacts include software written in C, C++, Java, PHP, mobile applications, and many more.

Common Weakness Enumeration

- Another useful feature is the external mappings of CWE content to related resources, including the annual CWE Top 25, OWASP Top Ten, Seven Pernicious Kingdoms, Software Fault Pattern Clusters, and SEI CERT Coding Standards for C, Java, and Perl.
- About CWE - https://cwe.mitre.org/about/index.html
- Details about CWE with example - https://cwe.mitre.org/about/new_to_cwe.html
- CWE List Version 4.15 - https://cwe.mitre.org/data/index.html
- View by software development - https://cwe.mitre.org/data/definitions/699.html
- View by Hardware Design https://cwe.mitre.org/data/definitions/1194.html

Common Weakness Enumeration

- View by Research Concepts - https://cwe.mitre.org/data/definitions/1000.html
- External Mappings: These views are used to represent mappings to external groupings, such as a Top-N list, and to express subsets of entries that are related by some external factor. - https://cwe.mitre.org/data/index.html
- CWE Top 25 Most Dangerous Software Weaknesses - https://cwe.mitre.org/top25/
- 2021 CWE Most Important Hardware Weaknesses - https://cwe.mitre.org/scoring/lists/2021_CWE_MIHW.html

Common Weakness Enumeration

- CWE List Version 4.15 – Downloads https://cwe.mitre.org/data/downloads.html
  - Navigate CWE
  - External Mappings
- https://cwe.mitre.org/about/new_to_cwe.html

Common Vulnerability Scoring System

- The Common Vulnerability Scoring System (CVSS) provides a way to capture the principal characteristics of a vulnerability and produce a numerical score reflecting its severity.
- **Scores ranging from 0 to 10**
- The numerical score can then be translated into a qualitative representation **(low, medium, high, and critical)** to help organizations properly assess and prioritize their vulnerability management processes.
- The **CVSS Special Interest Group (SIG)** is proud to announce the official publication of CVSS v4.0

Common Vulnerability Scoring System

- **CVSS is owned and managed by FIRST.Org**, Inc. (FIRST), a US-based non-profit organization whose mission is to help computer security incident response teams worldwide.
- A self-paced online training course (https://learn.first.org/) is available for CVSS v4.0. It explains the standard without assuming any prior CVSS experience. (https://www.first.org/cvss/)
- The SIG is composed of representatives from a broad range of **industry sectors, from banking and finance to technology and academia**. Organizations and individuals interested in joining the SIG, or observing progress via the CVSS SIG mailing lists.

Common Vulnerability Scoring System

- CVSS consists of four metric groups:
  - **Base** - The Base group represents the intrinsic qualities of a vulnerability that are **constant over time** and across user environments
  - **Threat** - reflects the characteristics of a vulnerability that **change over time**
  - **Environmental** - represents the characteristics of a vulnerability that are **unique to a user's environment**
  - **Supplemental** - do not modify the final score; they are used as **additional insight** into the characteristics of a vulnerability.

Common Vulnerability Scoring System

- Base metric values are combined with default values that assume the highest severity for Threat and Environmental metrics to produce a score ranging from 0 to 10.
- Threat and Environmental metrics can be amended based on applicable threat intelligence and environmental considerations to further refine a resulting severity score.
- Supplemental metrics do not modify the final score and are used as additional insight into the characteristics of a vulnerability. https://www.first.org/cvss/v4.0/examples

Common Vulnerability Scoring System

Common Vulnerability Scoring System

- Rest of the topic discussed on the basis of - https://www.first.org/cvss/calculator/4.0
- Example - https://www.first.org/cvss/v4.0/examples

Threat Modelling – STRIDE - DREAD

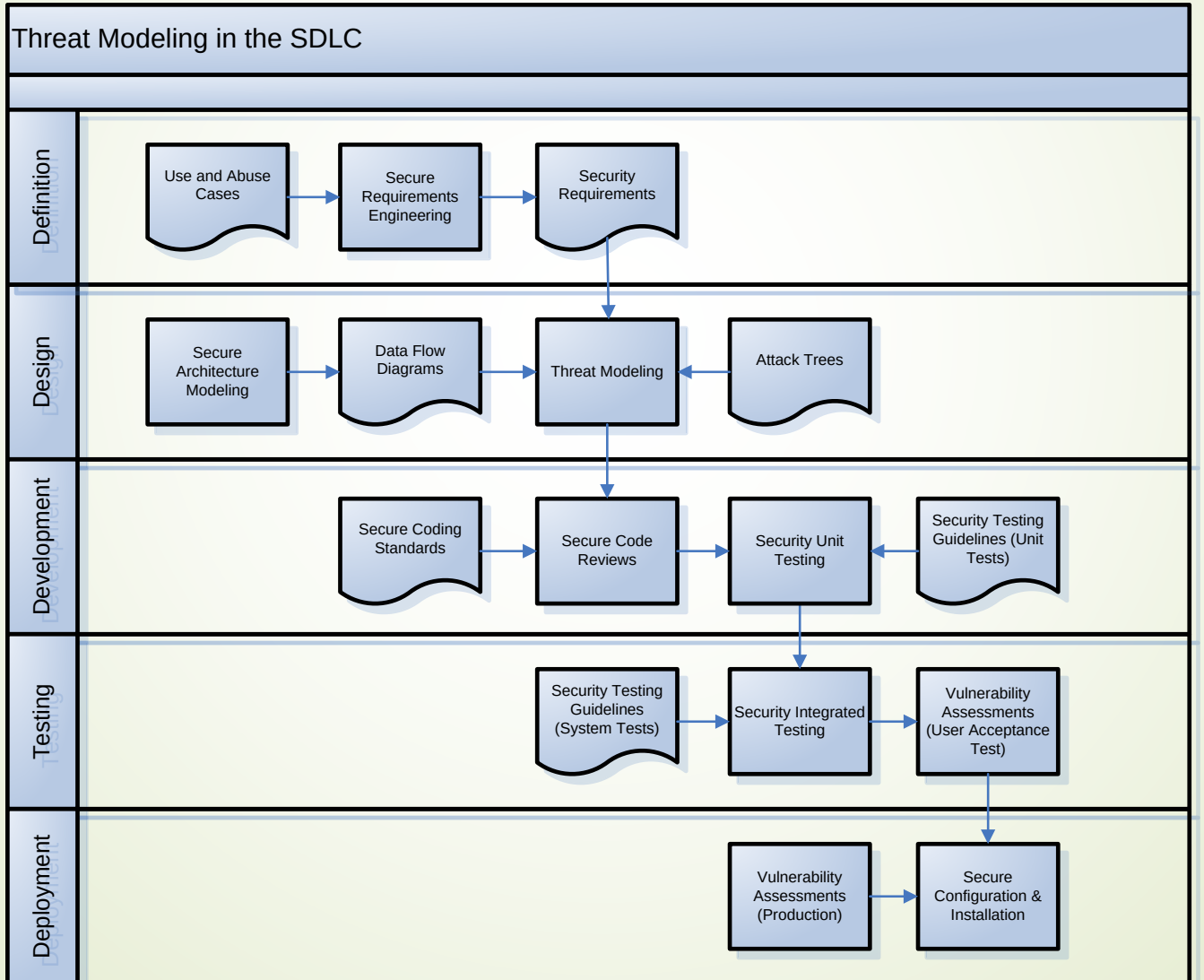
- Threat modeling is an approach for **analyzing the security of an application.**
- It is a **structured approach** that enables you to **identify, quantify, and address** the security risks associated with an application.
- Threat modeling is **not an approach to reviewing code** but complements the **security code review process**.
- Including threat modeling in the SDLC can help ensure that applications are being developed with **security built-in from the very beginning.**
- This, combined with the **documentation produced as part of the threat modeling process,** can give the reviewer a greater understanding of the system.

Threat Modelling – STRIDE - DREAD

- This allows the **reviewer to see where the entry points to the application are** and the **associated threats** with each entry point.
- The concept of threat modeling is not new, but there has been a clear mindset change in recent years.
- Modern threat modeling looks at a system from a **potential attacker's perspective, as opposed to a defender's viewpoint**.
- Microsoft has been a strong advocate of the process over the past number of years, **SDL (Security Development Life Cycle)**

Threat Modelling – STRIDE - DREAD

- Application Threat Modeling In the SDLC



Threat Modelling – STRIDE - DREAD

- They have made threat modeling a core component of their SDLC, which they claim to be one of the reasons for the increased security of their products in recent years.
- When **source code analysis is performed outside the SDLC**, such as on existing applications, the results of the threat modeling help in reducing the complexity of the source code analysis by promoting an **in-depth first approach vs. a breadth-first approach**.
- Instead of reviewing all source code with equal focus, **you can prioritize the security code review of components whose threat modeling has ranked with high-risk threats.**

Threat Modelling – STRIDE - DREAD

- **Step 1: Decompose the Application.**
- The threat modeling process can be decomposed **into 3 high level steps**:
- **The first step** in the threat modeling process is **concerned with gaining an understanding of the application and how it interacts with external entities.**
- This involves creating use-cases to understand
  - how the application is used,
  - identifying entry points to see where a potential attacker could interact with the application

Threat Modelling – STRIDE - DREAD

- **Step 1: Decompose the Application.**

- This involves creating use cases to understand
  - **identifying assets** i.e. items/areas that the attacker would be interested in,
  - **And identify trust levels** that represent the access rights that the application will grant to external entities.
- **This information is documented in the Threat Model document,** and it is also used to produce data flow diagrams (DFDs) for the application. The **DFDs show the different paths through the system**, **highlighting the privilege boundaries.**

Threat Modelling – STRIDE - DREAD

- **Step 2: Determine and rank threats.**
- Critical to the identification of threats is using a threat categorization methodology.
- A threat categorization such as **STRIDE** (See Next Slide) can be used, or the **Application Security Frame** (ASF) that defines threat categories such as
  - Auditing & Logging,
  - Authentication, Authorization,
    - Authentication = login + password (who you are)
    - Authorization = permissions (what you are allowed to do)

Threat Modelling – STRIDE - DREAD

- **The STRIDE Threat Model**

- When you are considering threats, it is helpful to ask questions such as these:
  - How can an attacker change the authentication data?
  - What would the impact be if an attacker could read the user profile data?
  - What happens if access is denied to the user profile database?

- You can **group threats into categories** to help you formulate these kinds of pointed questions.

Threat Modelling – STRIDE - DREAD

- **The STRIDE Threat Model**
- One model you may find helpful is **STRIDE**, derived from an acronym for the **following six threat categories:**
  - **Spoofing identity.** An example of identity spoofing is illegally accessing and using another user's authentication information, such as username and password.
  - **Tampering with data**. Data tampering involves the malicious modification of data. Examples include unauthorized changes to persistent data, such as that held in a database, and the alteration of data as it flows between two computers over an open network, such as the Internet.

Threat Modelling – STRIDE - DREAD

- **Example of Repudiation**,
- Suppose a customer sends a letter to a vendor agreeing to pay a hefty amount for a product.
- The vendor ships the product and then demands payment. The customer denies having ordered the product and, by law, is entitled to keep the unsolicited shipment without payment.
- The customer has repudiated the origin of the letter. If the vendor cannot prove that the letter came from the customer, the attack succeeds.
- A variant of this is denial by a user that he created specific information or entities such as files. Integrity mechanisms cope with this threat. (Email Spoofing)

Threat Modelling – STRIDE - DREAD

- **The STRIDE Threat Model**
  - **Nonrepudiation** refers to the ability of a system to counter repudiation threats. For example, a user who purchases an item might have to sign for the item upon receipt. The vendor can then use the signed receipt as evidence that the user did receive the package.
  - **Information disclosure**. Information disclosure threats involve the exposure of information to individuals who are not supposed to have access to it—for example, the ability of users to read a file that they were not granted access to or the ability of an intruder to read data in transit between two computers.

Threat Modelling – STRIDE - DREAD

- **The STRIDE Threat Model**
  - **Denial of service.** Denial of service (DoS) attacks deny service to valid users—for example, by making a Web server temporarily unavailable or unusable. You must protect against certain types of DoS threats simply to improve system availability and reliability.
  - **Elevation of privilege.** In this type of threat, an unprivileged user gains privileged access and thereby has sufficient access to compromise or destroy the entire system. Elevation of privilege threats include those situations in which an attacker has effectively penetrated all system defenses and become part of the trusted system itself, a dangerous situation indeed.

Threat Modelling – STRIDE - DREAD

STRIDE Threat & Mitigation Techniques List	
Threat Type	**Mitigation Techniques**
Spoofing Identity	1. Appropriate authentication 2. Protect secret data 3. Don't store secrets
Tampering with data	1. Appropriate authorization 2. Hashes 3. MACs 4. Digital signatures 5. Tamper resistant protocols
Repudiation	1. Digital signatures 2. Timestamps 3. Audit trails
Information Disclosure	1. Authorization 2. Privacy-enhanced protocols 3. Encryption 4. Protect secrets 5. Don't store secrets
Denial of Service	1. Appropriate authentication 2. Appropriate authorization 3. Filtering 4. Throttling 5. Quality of service
Elevation of privilege	1. Run with least privilege

Threat Modelling – STRIDE - DREAD

- **DREAD Model**
- Another method for determining risk is the DREAD model:
  - **Damage potential** – How great is the damage if the vulnerability is exploited?
  - **Reproducibility** – How easy is it to reproduce the attack?
  - **Exploitability** – How easy is it to launch an attack?
  - **Affected users** – As a rough percentage, how many users are affected?
  - **Discoverability** – How easy is it to find the vulnerability?
- **Risk = Min(D, (D+R+E+A+D) / 5)**

Threat Modelling – STRIDE - DREAD

- **Step 3: Determine countermeasures and mitigation.**

- A lack of protection against a threat might indicate a vulnerability whose risk exposure could be mitigated with the implementation of a countermeasure.
- Such countermeasures can be identified using threat-countermeasure mapping lists.
- Once a risk ranking is assigned to the threats, it is possible to sort threats from the highest to the lowest risk, and prioritize the mitigation effort, such as by responding to such threats by applying the identified countermeasures.

Threat Modelling – STRIDE - DREAD

- **Step 3: Determine countermeasures and mitigation.**

- The risk mitigation strategy might involve evaluating these threats from the business impact that they pose and reducing the risk.
- Other options might include taking the risk, assuming the business impact is acceptable because of compensating controls, informing the user of the threat, removing the risk posed by the threat completely, or the least preferable option, that is, to do nothing.

Threat Modelling – STRIDE - DREAD

- Each of the above steps are documented as they are carried out.
- The resulting document is the threat model for the application.
- We will learn this with Exanoke: The example that will be used is a college library website. At the end of the guide we will have produced the threat model for the college library website.
- Each of the steps in the threat modeling process are described in detail below.

- The goal of this step is to gain an understanding of the application and how it interacts with external entities. This goal is achieved by information gathering and documentation. The information gathering process is carried out using a clearly defined structure, which ensures the correct information is collected. This structure also defines how the information should be documented to produce the Threat Model.

Threat Modelling – STRIDE - DREAD

- **Threat Model Information**

- The first item in the threat model is the information relating to the threat model. This must include the the following:
  - **Application Name -** The name of the application.
  - **Application Version -** The version of the application.
  - **Description -** A high level description of the application.
  - **Document Owner -** The owner of the threat modeling document.
  - **Participants -** The participants involved in the threat modeling process for this application.
  - **Reviewer -** The reviewer(s) of the threat model.

Secure Coding Practice

- Input Validation
- Output Encoding
- Authentication and Password Management
- Session Management
- Access Control
- Cryptographic Practices
- Error Handling and Logging

Secure Coding Practice

- Data Protection
- Communication Security
- System Configuration
- Database Security
- File Management
- Memory Management
- General Coding Practices


QUESTIONS?
COMMENTS?
CONCERNS?