# Vulnerability Management Lifecycle


## 01 Discover

The first phase is the information gathering phase where we collect the inventories of all assets to be scanned. It is also known as the preparatory phase in vulnerability management process. Discovery phase will find every server, applications and network connected components.

## 02 Prioritise Assets

Once the list of assets is confirmed, the next step is to prioritise them based on their importance to the business. It is also known as the Pre-Assessment Phase. Prioritising assets according to their criticality is the key determining where to take the action first.

## 03 Assess

This phase includes network (internal and external) scanning, firewall logging, configuration scan, penetration testing and using the automated vulnerability scanners. Identify and verify known vulnerabilities on the servers, applications and systems. All the identified and verified vulnerabilities needed to be classified based on severity and the level of risk it presents to organisation.

## 04 Report

Once the assessment is complete and all the verified vulnerabilities have been discovered then a report is generated which includes the list of findings and the remediation plans. Reports can be used for tactical operational tasks, to provide visibility and business oriented risk metrics to management.

## 05 Remediate

This phase involves implementing the relevant technical measures such as applying the relevant patches, re-configuration of firewalls or other network & host based vulnerabilities. Remediate the vulnerabilities based on risk level, risk of compromise and impact of compromise.

## 06 Verify

After the application of patches and completion of the remediation process a re-assessment of network connected assets should be carried out to ensure that the fix works properly and it does not cause other network devices, services or applications to malfunction.

© Martin Covill

# NVD Vulnerability Severity Ratings

The Common Vulnerability Scoring System (CVSS) provides an open framework for communicating the characteristics and impacts of IT vulnerabilities. Its quantitative model ensures repeatable accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the scores. Thus, CVSS is well suited as a standard measurement system for industries, organizations, and governments that need accurate and consistent vulnerability impact scores. Two common uses of CVSS are prioritization of vulnerability remediation activities and in calculating the severity of vulnerabilities discovered on one's systems. The National Vulnerability Database (NVD) provides CVSS scores for almost all known vulnerabilities.

https://nvd.nist.gov/vuln-metrics/cvss

| Severity | BASE SCORE RANGE |
|---|---|
| 0.0-3.9 | LOW |
| 4.0–6.9 | MEDIUM |
| 7.0–10 | HIGH |

**CVSS v2.0 Ratings**

| Severity | BASE SCORE RANGE |
|---|---|
| 0.0 | NONE |
| 0.1-3.9 | LOW |
| 4.0–6.9 | MEDIUM |
| 7.0–8.9 | HIGH |
| 9.0–10 | CRITICAL |

**CVSS v3.0 Ratings**

© Martin Coyill

# CVSS 3.0 Scoring System

**Base Score**

**Attack Vector (AV)**
Network (N) Adjacent (A) Local (L) Physical (P)
**Attack Complexity (AC)**
Low (L) High (H)
**Privileges Required (PR)**
None (N) Low (L) High (H)
**User Interaction (UI)**
None (N) Required (R)
**Scope (S)**
Unchanged (U) Changed (C)
**Confidentiality (C)**
None (N) Low (L) High (H)
**Integrity (I)**
None (N) Low (L) High (H)
**Availability (A)**
None (N) Low (L) High (H)

**Temporal Score**

**Exploit Code Maturity (E)**
Not Defined (X) Unproven (U) Proof-of-Concept (P) Functional (F) High (H)
**Remediation Level (RL)**
Not Defined (X) Official Fix (O) Temporary Fix (T) Workaround (W) Unavailable (U)
**Report Confidence (RC)**
Not Defined (X) Unknown (U) Reasonable (R) Confirmed (C)

The **Base** metric group represents the intrinsic characteristics of a vulnerability that are constant over time and across user environments. It is composed of two sets of metrics: the Exploitability metrics and the Impact metrics.
The Exploitability metrics reflect the ease and technical means by which the vulnerability can be exploited. The Impact metrics reflect the direct consequence of a successful exploit, and represent the consequence to the *thing that suffers the impact*, which we refer to formally as the *impacted component*.
The **Temporal** metric group reflects the characteristics of a vulnerability that may change over time but not across user environments. For example, the presence of a simple-to-use exploit kit would increase the CVSS score, while the creation of an official patch would decrease it.
The **Environmental** metric group represents the characteristics of a vulnerability that are relevant and unique to a particular user's environment. These metrics allow the scoring analyst to incorporate security controls which may mitigate any consequences, as well as promote or demote the importance of a vulnerable system according to her business risk.

**Environmental Score**

**Confidentiality Requirement (CR)**
Not Defined (X) Low (L) Medium (M) High (H)
**Integrity Requirement (IR)**
Not Defined (X) Low (L) Medium (M) High (H)
**Availability Requirement (AR)**
Not Defined (X) Low (L) Medium (M) High (H)
**Modified Attack Vector (MAV)**
Not Defined (X) Network Adjacent
Network Local Physical
**Modified Attack Complexity (MAC)**
Not Defined (X) Low High
**Modified Privileges Required (MPR)**
Not Defined (X) None Low High
**Modified User Interaction (MUI)**
Not Defined (X) None Required
**Modified Scope (MS)**
Not Defined (X) Unchanged Changed
**Modified Confidentiality (MC)**
Not Defined (X) None Low High
**Modified Integrity (MI)**
Not Defined (X) None Low High
**Modified Availability (MA)**
Not Defined (X) None Low High

https://www.first.org/cvss/

© Martin Covill

# Risk Analysis & Mitigation


## CALCULATE RISKS

Objectively calculate risk using various models:

- **Qualitative**
  - Likelihood x Impact (H,M,L)
  - Threat Source (STRIDE) x Severity (DREAD)
  - Threat x Vulnerability Impact (OWASP)
- **Quantitative**
  - ALE = SLE x ARO

## DEVISE MITIGATION STRATEGY

Use holistic measures to devise mitigation strategy:

- **Preventative & Detective Controls**
- **Apply Defence In Depth Countermeasures at various layers**
  - i.e. Browser, Web Application, Infrastructure
- **Implement Processes to provide strong Governance**
  - i.e. risk based testing, fraud detection, threat analysis, cyber threat intelligence

© Martin Covill

# Threat Modelling

Many threat-modeling methods have been developed and can be combined to create a more robust view of potential threats.
Threat modeling should be performed early in the software development cycle as potential issues can be captured and remedied relatively simply, preventing more expensive remediation later in the cycle. A couple of well known threat modeling methods are shown below:

**STRIDE and Associated Derivations**

**Invented in 1999 and adopted by Microsoft in 2002, STRIDE is currently the most mature threat-modeling method. STRIDE has evolved over time to include new threat-specific tables. STRIDE evaluates the system detail design and models the in-place system. By building data-flow diagrams (DFDs), STRIDE is used to identify system entities, events, and the boundaries of the system. STRIDE applies a general set of known threats based on its name, which is a mnemonic, as shown in the following table:**

| | Threat | Property Violated | Threat Definition |
|---|---|---|---|
| S | Spoofing identity | Authentication | Pretending to be something or someone other than yourself |
| T | Tampering with data | Integrity | Modifying something on disk, network, memory, or elsewhere |
| R | Repudiation | Non-repudiation | Claiming that you didn't do something or were not responsible; can be honest or false |
| I | Information disclosure | Confidentiality | Providing information to someone not authorized to access it |
| D | Denial of service | Availability | Exhausting resources needed to provide service |
| E | Elevation of privilege | Authorization | Allowing someone to do something they are not authorized to do |

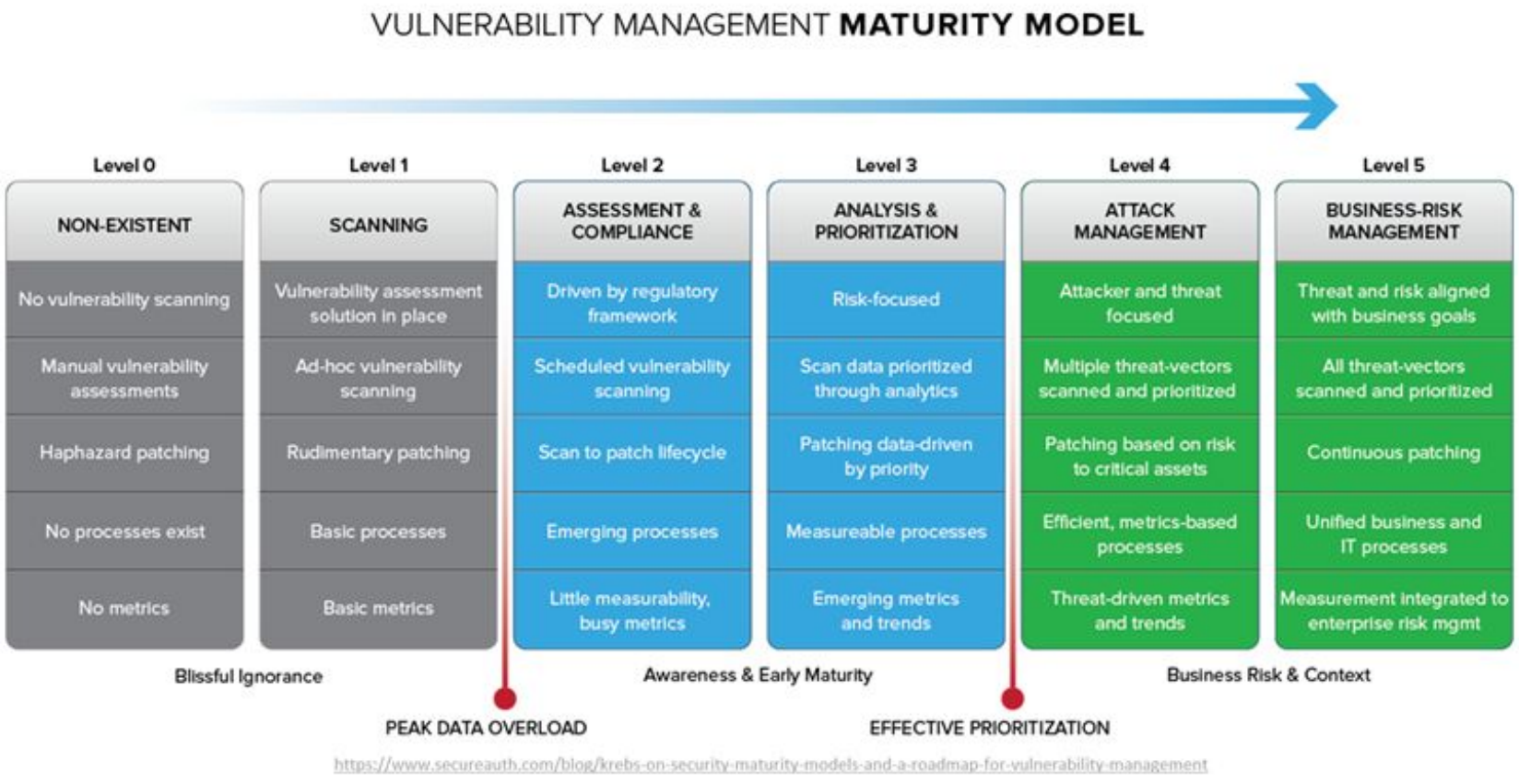
**PASTA**

**The Process for Attack Simulation and Threat Analysis (PASTA) is a risk-centric threat-modeling framework developed in 2012. It contains seven stages, each with multiple activities, as shown below:**


https://insights.sei.cmu.edu/sei_blog/2018/12/threat-modeling-12-available-methods.html

© Martin Covill

# Vulnerability Management Maturity Model




https://www.secureauth.com/blog/krebs-on-security-maturity-models-and-a-roadmap-for-vulnerability-management

© Martin Covill