1.2. CVD Context and Terminology Notes

Before we proceed to place CVD in context, we start with a few definitions.

**Vulnerability**

A vulnerability is a set of conditions or behaviors that allows the violation of an explicit or implicit security policy. Vulnerabilities can be caused by software defects, configuration or design decisions, unexpected interactions between systems, or environmental changes. Successful exploitation of a vulnerability has technical and risk impacts. Vulnerabilities can arise in information processing systems as early as the design phase and as late as system deployment.

NIST offers the following definitions of vulnerability [1]:

1. "Weakness in an information system, system security procedures, internal controls, or implementation that could be exploited or triggered by a threat source"
2. "A weakness in a system, application, or network that is subject to exploitation or misuse"
3. "Weakness in an information system, system security procedures, internal controls, or implementation that could be exploited by a threat source"

Those familiar with the CERT Resiliency Management Model (RMM) may be accustomed to the more general definition of vulnerability in the Vulnerability Analysis and Resolution (VAR) practice: "A vulnerability is the susceptibility of an asset and associated service to disruption"[2]. A summary of the VAR process area of the CERT RMM can be found below.

While vulnerabilities can be found in many assets belonging to an organization—people, information, technology, and facilities—in this document we primarily focus on vulnerabilities in software or software-centric products and to a lesser degree services built on software-dependent products. While precisely defining vulnerability can be difficult, for our purpose a vulnerability may be thought of as an undesirable, exploitable, and likely unintended feature of software or hardware components that allows an attacker to perform actions that wouldn't otherwise be available to them. The impact of such vulnerabilities can vary greatly, from being able to access someone's private data, to taking control of a computer, to causing physical damage and bodily injury.

**Exploits, Malware, and Incidents**

We also need to get a few quick technical terms out of the way so they don't cloud the remaining discussion.

- An exploit is software that uses a vulnerability to achieve some effect. Sometimes the effect is as simple as demonstrating the existence of the vulnerability. Other times it plays a role in enabling adversaries to attack systems.
- Malware is software used by adversaries to compromise the security of a system or systems. But not all malware involves exploits.
- Finally, an incident is a violation or an attempted violation of a security policy, and may involve malware, exploits, or vulnerabilities (or none of these?)

**Vulnerability Response (VR)**

Vulnerability Response (VR) is the overall set of processes and practices that deal with the existence of vulnerabilities in systems. VR encompasses everything from reducing the introduction of vulnerabilities as part of a Secure Development Lifecycle (SDL) through the remediation of deployed vulnerabilities via patch deployment.

Vulnerability response in the design and development phases often takes the form of practices such as threat modeling [3,4], secure coding [5,6,7], and architecture risk analysis [8,9]. However, such practices seem unlikely to ever completely eliminate vulnerabilities from being introduced into released software and deployed systems. For those vulnerabilities that do escape detection by these early lifecycle practices, it is necessary to plan for their eventual discovery and disclosure.

The goals of vulnerability response include the following:

- Limit attacker advantage over defenders.
- Reduce the population of vulnerable product instances as quickly as possible.
- Reduce the impact of attacks against vulnerable systems.

**Vulnerability Discovery**

Vulnerability discovery can take many forms, from specifically targeted software testing to simple use of a system by a security-aware individual who notices some feature that seems out of place. In order for that discovery to be relevant to our discussion, it must result in a vulnerability
Vulnerability Management (VM)

Vulnerability Management (VM) is the common term for tasks such as vulnerability scanning, patch testing, and deployment. VM practices nearly always deal with the output of CVD practices, not the inputs. VM practices focus on the positive action of identifying specific systems affected by known (post-disclosure) vulnerabilities and reducing the risks they pose through the application of mitigations or remediation such as patches or configuration changes. NIST Special Publication 800-40 provides a Guide to Enterprise Patch Management Technologies [18]. VM practices also appear within the Vulnerability Analysis and Resolution operational process of the CERT RMM [2].
Vulnerability Analysis and Resolution (VAR)

Vulnerability Analysis and Resolution (VAR) is an operational process described within the CERT RMM that closely overlaps with the concept of Vulnerability Management. Although the RMM is designed with a focus on operational resilience for organizations, there is sufficient overlap with our topic that it's worth highlighting here. Within the RMM's VAR process area, a number of goals and practices are identified:

- Prepare for Vulnerability Analysis and Resolution.
- Establish Scope – The assets and operational environments that must be examined for vulnerabilities are identified.
- Establish a Vulnerability Analysis and Resolution Strategy.
- Establish and maintain a process for identifying and analyzing vulnerabilities.
- Identify Sources of Vulnerability Information.
- Discover Vulnerabilities.
- Analyze Vulnerabilities to determine whether they need to be reduced or eliminated.
- Manage Exposure to Vulnerabilities – Strategies are developed and implemented to manage exposure to identified vulnerabilities.
- Identify Root Causes – The root causes of vulnerabilities are examined to improve vulnerability analysis and resolution and reduce organizational exposure. Perform review of identified vulnerabilities to determine and address underlying causes.

Products and Instances

In talking about things that have vulnerabilities, we try to maintain a clear distinction between a product being vulnerable, and an instance of a product being vulnerable. For example, Windows 10 (the product) might be vulnerable to a specific flaw, but that is a separate situation from a server running Windows 10 (an instance) being vulnerable. Vulnerabilities affecting products may not always affect every instance of a product; for example, a vulnerability may require a special configuration or setup to be exploited, so any instance not in that configuration state would actually be unaffected by the vulnerability, despite the product at-large being vulnerable.

This distinction becomes important when one is talking about the practices associated with Vulnerability Management (VM)—namely vulnerability scanning—in contrast to CVD and vulnerability discovery. VM entails the identification of instances of a product on which action must be taken to remediate known vulnerabilities in the product. VM is concerned with the eradication of the instances of known vulnerabilities in deployed systems, whereas CVD is concerned with the repair of vulnerabilities at the product level.

Incident vs. Vulnerability Response

Sometimes the term “Incident Response” is used synonymously with Vulnerability Response. These two concepts are related, but different; Vulnerability Response specifically indicates responding to reports of product vulnerabilities, usually via the CVD process, whereas Incident Response is more general and can also include other security events such as network intrusions. We will generally stick to the Vulnerability Response terminology since this work is specifically about CVD.

References

