Executive Summary

Software-based products and services have vulnerabilities—conditions or behaviors that allow the violation of an explicit or implicit security policy. This should come as no surprise to those familiar with software. What many find surprising nowadays is just how many products and services should be considered software based. The devices we depend on to communicate and coordinate our lives, transport us from place to place, and keep us healthy have in recent years become more and more connected both to each other and to the world at large. As a result, society has developed an increasing dependence on software-based products and services along with a commensurate need to address the vulnerabilities that inevitably accompany them.

Adversaries take advantage of vulnerabilities to achieve goals at odds with the developers, deployers, users, and other stakeholders of the systems we depend on. Notifying the public that a problem exists without offering a specific course of action to remediate it can result in giving an adversary the advantage while the remediation gap persists. Yet there is no optimal formula for minimizing the potential for harm to be done short of avoiding the introduction of vulnerabilities in the first place. In short, vulnerability disclosure appears to be a wicked problem. The definition of a wicked problem based on an article by Rittel and Webber [1] is given in Section 2.7.

Coordinated Vulnerability Disclosure (CVD) is a process for reducing adversary advantage while an information security vulnerability is being mitigated. CVD is a process, not an event. Releasing a patch or publishing a document are important events within the process, but do not define it.

CVD participants can be thought of as repeatedly asking these questions: What actions should I take in response to knowledge of this vulnerability in this product? Who else needs to know what, and when do they need to know it? The CVD process for a vulnerability ends when the answers to these questions are nothing, and no one.

CVD should not be confused with Vulnerability Management (VM). VM encompasses the process downstream of CVD, once the vulnerability has been disclosed and deployers must take action to respond. Section 1 introduces the CVD process and provides notes on relevant terminology.

- Principles of CVD
- Roles in CVD
- Phases of CVD
- CVD Process Variation
- Troubleshooting CVD
- Operational Considerations
- Open Problems in CVD
- Conclusion and Appendices
- References

Principles of CVD

Section 2 covers principles of CVD, including the following:

- **Reduce Harm** —Decrease the potential for damage by publishing vulnerability information; using exploit mitigation technologies; reducing days of risk; releasing high-quality patches; and automating vulnerable host identification and patch deployment.
- **Presume Benevolence** —Assume that any individual who has taken the time and effort to reach out to a vendor or a coordinator to report an issue is likely benevolent and sincerely wishes to reduce the harm of the vulnerability.
- **Avoid Surprise** —Surprise tends to increase the risk of a negative outcome from the disclosure of a vulnerability and should be avoided.
- **Incentivize Desired Behavior** —It’s usually better to reward good behavior than try to punish bad behavior. Incentives are important as they increase the likelihood of future cooperation between security researchers and organizations.
- **Ethical Considerations** —A number of ethical guidelines from both technical and journalistic professional societies can find application in the CVD process.
- **Process Improvement** —Participants in the CVD process should learn from their experience and improve their process accordingly. CVD can also provide important feedback to an organization’s Software Development Lifecycle (SDL).
- **CVD as a Wicked Problem** —As we’ve already mentioned, vulnerability disclosure is a multifaceted problem for which there appear to be no “right” answers, only “better” or “worse” solutions in a given context.

Roles in CVD

CVD begins with finding vulnerabilities and ends with the deployment of patches or mitigations. As a result, several distinct roles and stakeholders are involved in the CVD process. These include the following:

- **Finder (Discoverer)** —the individual or organization that identifies the vulnerability
- **Reporter** —the individual or organization that notifies the vendor of the vulnerability
- **Vendor** —the individual or organization that created or maintains the product that is vulnerable
- **Deployer** —the individual or organization that must deploy a patch or take other remediation action
- **Coordinator** —an individual or organization that facilitates the coordinated response process
It is possible and often the case that individuals and organizations play multiple roles. For example, a cloud service provider might act as both vendor and deployer, while a researcher might act as both finder and reporter. A vendor may also be both a deployer and a coordinator.

Reasons to engage a coordinator include reporter inexperience, reporter capacity, multiparty coordination cases, disputes among CVD participants, and vulnerabilities having significant infrastructure impacts.

Users, integrators, cloud and application service providers, Internet of Things (IoT) and mobile vendors, and governments are also stakeholders in the CVD process. We cover these roles and stakeholders in more detail in Section 3.

**Phases of CVD**

The CVD process can be broadly defined as a set of phases, as described in Section 4. Although these phases may sometimes occur out of order, or even recur within the handling of a single vulnerability case (for example, each recipient of a case may need to independently validate a report), they often happen in the following order:

- **Discovery** – Someone discovers a vulnerability in a product.
- **Reporting** – The product's vendor or a third-party coordinator receives a vulnerability report.
- **Validation and Triage** – The receiver of a report validates it to ensure accuracy before prioritizing it for further action.
- **Remediation** – A remediation plan (ideally a software patch, but could also be other mechanisms) is developed and tested.
- **Public Awareness** – The vulnerability and its remediation plan is disclosed to the public.
- **Deployment** – The remediation is applied to deployed systems.

**CVD Process Variation**

As an endeavor of human coordination at both the individual and organization levels, the CVD process can vary from participant to participant, over time, and in varying contexts. Some points of variation include those below:

- **Choosing a disclosure policy** – Disclosure policies may need to be adapted for different organizations, industries, and even products due to variations in business needs such as patch distribution or safety risks.
- **Coordinating among multiple parties** – Coordination between a single finder and a single vendor is relatively straightforward, but cases involving multiple finders, or complex supply chains often require extra care.
- **Pacing and synchronization** – Different organizations work at different operational tempos, which can increase the difficulty of synchronizing release of vulnerability information along with fixes.
- **Coordination Scope** – CVD participants must decide how far to go with the coordination process. For example, it may be preferable to coordinate the disclosure of critical infrastructure vulnerabilities all the way out to the system deployers, while for a mobile application it may be sufficient to notify the developer and simply allow the automatic update process take it from there.

Variation points in the CVD process are covered in Section 5.

**Troubleshooting CVD**

CVD does not always go the way it's supposed to. We have encountered a number of obstacles along the way, which we describe in Section 6. These are among the things that can go wrong:

- **No vendor contact available** – This can occur because a contact could not be found, or the contact is unresponsive.
- **Participants stop responding** – Participants in CVD might have other priorities that draw their attention away from completing a CVD process in progress.
- **Information leaks** – Whether intentional or not, information that was intended for a private audience can find its way to others not involved in the CVD process.
- **Independent discovery** – Any vulnerability that can be found by one individual can be found by another, and not all of them will tell you about it.
- **Active exploitation** – Evidence that a vulnerability is being actively exploited by adversaries often implies a need to accelerate the CVD process to reduce users’ exposure to risk.
- **Relationships go awry** – CVD is a process of coordinating human activities. As such, its success depends on building relationships among the participants.
- **Hype, marketing, and unwanted attention** – The reasons for reporting and disclosing vulnerabilities are many, but in some cases they can be used as a tool for marketing. This is not always conducive to the smooth flow of the CVD process.

When things do go askew in the course of the CVD process, it's often best to remain calm, avoid legal entanglements, and recognize that the parties involved are usually trying to do the right thing. In some cases, it may help to consider publishing earlier than originally planned or to engage a third-party coordinator to assist with mediating disputes. Regardless of the resulting action, CVD participants should learn from the experience.
Operational Considerations

Participation in the CVD process can be improved with the support of tools and operational processes such as secure communications (e.g., encrypted email or https-enabled web portals), contact management, case tracking systems, code and system inventories, and test environments such as virtualized labs.

Operational security should also be considered. CVD participants will need to address key management for whatever communications encryption they decide to use. Policy guidelines for handling sensitive data should be clearly articulated within organizations. Furthermore, recipients of vulnerability reports (e.g., vendors and coordinators) should be wary of credulous action in response to reports. Things are often not what they originally seem. Reporters may have misinterpreted the impact of a vulnerability to be more or less severe than it actually is. Adversaries may be probing an organization's vulnerability response process to gain information or to distract from other events.

As happens in many security operations roles, staff burnout is a concern for managers of the CVD process. Job rotations and a sustained focus on CVD process improvement can help.

Further discussion of operational considerations can be found in Section 7.

Open Problems in CVD

Organizations like the CERT Coordination Center have been coordinating vulnerability disclosures for decades, but some issues remain to be addressed. The emergence of a wider diversity of software-based systems in recent years has led to a need to revisit topics once thought nearly resolved. Vulnerability identity has become a resurgent issue in the past few years as the need to identify vulnerabilities for purposes of CVD and vulnerability management has spread far beyond the arena of traditional computing. A number of efforts are currently underway to improve the way forward.

More broadly, the rising prevalence of IoT products and their corresponding reliance on embedded systems with constrained hardware, power, bandwidth, and processing capabilities has led to a need to rethink CVD in light of assumptions that are no longer valid. Patching may be comparatively easy on a Windows system deployed on an enterprise network. Patching the firmware of a home router deployed to all the customers of a regional ISP is decidedly not so simple. The desktop system the doctor uses to write her notes might be patched long before the MRI machine that collected the data she's analyzing. Fixing a vulnerable networked device atop a pipeline in a remote forest might mean sending a human out to touch it. Each of these scenarios comes with an associated cost not usually factored into the CVD process for more traditional systems.

The way industries, governments, and society at large will address these issues remains to be seen. We offer Section 8 in the hope that it sheds some light on what is already known about these problems.

Conclusion and Appendices

Vulnerability disclosure practices no longer affect only the computer users among us. Smart phones, ATMs, MRI machines, security cameras, cars, airplanes, and the like have become network-enabled software-dependent systems, making it nearly impossible to avoid participating in the world without the potential to be affected by security vulnerabilities. CVD is not a perfect solution, but it stands as the best we've found so far. We've compiled this guide to help spread the practice as widely as possible.

Six appendices are provided containing background on IoT vulnerability analysis, Traffic Light Protocol, examples of vulnerability report forms and disclosure templates, pointers to publicly available disclosure policy templates, and pointers to additional resources for web vulnerabilities. An extensive bibliography is also included.

References