7.1 Tools of the Trade

An organization's capabilities are supported by people, process, and tools, and CVD is no different. This section covers some commonly used tools.

Secure Communication Channels

Secure communications can be nuanced to maintain in operation, so we first turn our attention to establishing and maintaining this capability.

Email

Email is a simple, tried-and-true method of communication over the Internet. Simply because everyone has access to it, email is likely to remain a common way of receiving vulnerability reports and communications. Because of the potential for organizational changes like employee promotions and turnover, having an individual's email address as a security point of contact is generally discouraged. A better solution is to establish an email account specifically for vulnerability reports and CVD activity. Many vendors choose to reserve an email address like <security@example.com> or <psirt@example.com> for this purpose. RFC2350 [1] suggests you include contact information as part of your overall CSIRT disclosure policy and process publication.

Additionally, we recommend that this information be listed clearly under the Contact Us page or a similar page on your organization's website. Ideally, an Internet search for "<vendor> report vulnerability" should lead to that contact information. The security email account should be an alias that is forwarded to or shared by one or more people. This way, multiple team members can check the incoming mail and cover for each other when team members are out of the office. Even if the security team is only one person, having an alias makes it easier to adapt should that person leave the organization; to the outside world, no contact information needs to change.

Although receiving information via email is convenient, it is not a very good mechanism for tracking multiple cases at once. Rather than sending security emails to individual mailboxes, vendors, coordinators, and even large-scale reporters should consider setting up a web-based case tracking system instead. That way, received emails can automatically generate new cases or augment existing ones, which can then be assigned to team members and tracked as necessary. More on this topic can be found in Section 7.1.

Secure Email

Email by itself is not a secure medium. We recommend that emails containing sensitive information be encrypted [2]. This includes emails containing information about unpatched or otherwise sensitive vulnerabilities. The most common tools for email encryption are Pretty Good Privacy (PGP) [3] or Gnu Privacy Guard (GPG) [4], and Secure/Multipurpose Internet Mail Extensions (S/MIME) using X.509 certificates [5]. By far, PGP/GPG is the most commonly used by CVD participants. It has a learning curve but is relatively easy to use and maintain once set up. A number of tools exist to make working with PGP/GPG easier. In practice, very few individuals and organizations make use of S/MIME & X.509 for their CVD process; while you may offer S/MIME & X.509 as an option, PGP/GPG is recommended as the default. The greatest difficulty with PGP/GPG (and really, any email encryption scheme) is encryption key management. Key management will be discussed in Section 7.2.

Web Forms and Portals

Most vulnerability reports have a similar structure, making a web form a preferable method for receiving vulnerability reports for many organizations. To secure your web form, you will need to enable HTTPS and TLS, and obtain a TLS certificate from a Certificate Authority (CA). A free option is Let's Encrypt, maintained by the Internet Security Research Group (ISRG) [6].

TLS ensures that the transmission is secure, but does not authenticate who sent the report. If this is a requirement, you may also need to implement login accounts on your web form with adequate authentication mechanisms such as two-factor or X.509 certificates. However, this setup increases the complexity for reporters to provide information to you, and as a result is rare in practice and likely unnecessary. Once set up, an HTTPS web form provides an easy-to-use way for reporters to contact you with vulnerability reports and other information. However, a problem does come up: how do you acknowledge receipt of the report?

Possibilities include the following:

- *Echoing back the message in a confirmation email.* The reporter knows you received the text, but now the text was sent in plain text email for everyone to read. It is unlikely you want to do this.
- *Send a brief thank you message, but without details.* The reporter can now be assured that the report was received. But what if you have a follow up question for the reporter? It's likely you will need to send an email, which will require encryption to keep the details of the question and answer secure.
- *Send a brief thank you message, and ask the reporter to log in to a portal to view the message conversation.* This resolves the issue of two-way secure communications, but now you need to establish a public-facing portal and manage portal credentials. This raises additional operational considerations, for example those below:
  - user account management and password changes
  - two-factor authentication or X.509 as possible requirement to ensure user identity

Portal credential maintenance introduces more complications, and for many vendors is likely not worth the effort. Another disadvantage of using a portal is that vendors and organizations may be unwilling to create accounts on another vendor’s portal. This may discourage multi-party communication and coordination, effectively preventing a vendor or reporter from participating in multi-party CVD.

In the CERT/CC’s experience, PGP/GPG encrypted email is the most common solution for a primary contact channel; HTTPS websites or third-party platforms can complement but may not always be sufficient to replace email for your organization’s CVD needs.

Contact Management
For most reporters, the contact management process simply consists of maintaining a vendor’s email address and PGP/GPG key in compatible mail client software. Contact management becomes vitally important to multiparty CVD, and is a particular concern for third-party coordinators. A common choice is Thunderbird with Enigmail, but other open source solutions such as Outlook with gpg4win, or KMailing with Kigpg/Kleopatra and proprietary solutions such as Outlook with Symantec Encryption Desktop also exist.

Finding vendor contacts can be difficult. Not all vendors include contact information in an easily searchable page, such as a Contact Us page linked from the vendor’s homepage. Some alternatives include searching old mailing lists, using social media, or even sending physical letters to a business address.

In order to protect privacy during the disclosure process, mailing lists or simply carbon-copying all recipients to a single message is likely not an acceptable action in most scenarios. Vendors in many cases would like to keep their vulnerability information private except for what is specifically intended to be shared. At the CERT/CC, we have developed some in-house mailing scripts that auto-generate individual encrypted emails, one for each vendor we attempt to reach. In this way, we can maintain privacy up front, but can introduce two vendors should there be mutual interest in collaboration. Our current tools were written with our internal systems and network policies in mind. Other coordinators may look into similar efforts. We covered communication topologies for CVD in Section 5.5.

**Bug Bounty Platforms**

A number of third-party CVD platforms now exist to facilitate communication between vendors and reporters. Although they are often referred to as bug bounty platforms, the “bounty” aspect is in fact optional—vendors can use bug bounty platforms to receive reports without needing to compensate reporters unless they choose to do so.

CVD platforms provide a secure communications channel (HTTPS) for reporters to communicate with vendors. These platforms generally allow two-way communications, making it easy for ongoing discussion between vendor and reporter. This channel is usually hosted by a third party in a software-as-a-service model, which may be important to some organizations that are not able to maintain their own infrastructure due to resource constraints. Of course, having vulnerability information hosted on third-party infrastructure may also present a data privacy risk to some organizations, so it is important to consult internal policies before determining if a CVD platform fits your organization’s needs and requirements.

An important note regarding these platforms is that the CVD platform by its nature requires a login. As explained in our discussion in the last section, requiring an account may discourage some reporters or other organizations from joining the platform, locking them out of discussion. Organizations should consider whether the benefits of using a CVD service outweigh this concern.

**Case and Bug Tracking**

Case tracking systems such as bug trackers or trouble ticket systems are often used by vendors, coordinators, and reporters for tracking vulnerability reports. Such systems can centralize the vulnerability response process and provide the ability to track individual cases. Case tracking systems also provide a means of collecting data about recurring security issues and the performance of the response process itself.

We have found it important to distinguish that vulnerability reports are not always bug reports. A single vulnerability report might contain information on more than one bug; alternately, it may describe a configuration issue that would not typically be considered a bug in the first place. In general, vulnerability reports and bug reports should be thought of as having a many-to-many relationship, allowing one or more vulnerability reports to be linked to one or more bug reports.

That said, bug tracking systems can still be useful for vulnerability report tracking if this distinction is kept in mind, and the bug tracking system has the appropriate capabilities. At the CERT/CC, we’ve found that more complicated CVD cases—for example the multiparty cases described in Section 5.4—become more like projects than tickets.

**Code and System Inventories**

As we discussed in Section 5.4, software-based products are typically assembled from components rather than written from scratch. Economies of scale apply to code, and most organizations would consider it wasteful to write a feature from scratch when that feature is available at a reasonable licensing cost for inclusion into their own products. As a result, each product is not merely the result of a single vendor’s development process, but of an entire supply chain. Libraries get included in other libraries, which form subcomponents in the composition of larger and more complex products.

For product vendors, an important part of the vulnerability response process is knowing what weaknesses your products might have. You can address this by clearly identifying any third-party libraries or utilities that are included with your products, and being alert and responsive to vulnerability disclosures in any third-party products that may affect your own products.

In recent years, a class of Software Composition Analysis tools (such as those offered by WhiteSource Software, Black Duck Software, Sonatype, Synopsys, Flexera Software, and the like) have come into use to identify potential licensing conflicts in commercial and open source products. Many of these tools are potentially useful to vendors looking to create an inventory of libraries and third-party components used in their products for security analysis purposes as well.

As a vendor, identifying your products’ third-party dependencies is only the first step. After that, you should take steps to ensure that your vulnerability response capability maintains awareness of any security-related announcements about those upstream products on which your product depends. Some component vendors offer special communication channels (e.g., a mailing list) for their licensees. If you’ve worked with a coordinator like the CERT/CC in the past, ask if you can be placed in a special notification list for a particular library or product.

Furthermore, since it is likely that your products will in turn be used as components in some other vendors’ solution, it can be a good practice to provide an inventory of components along with your product. A number of data formats and specifications have emerged in the software supply chain management space and are in use by product vendors already.
Test Bench and Virtualization

Having an internal testing infrastructure is vital to proper triage and resolution of vulnerability reports as we discussed in Section 4.3. Not only is testing useful for confirming reports, or reproducing and isolating bugs; it can also serve as a platform for an organization to develop its own vulnerability discovery capability.

The analysts responsible for confirming incoming reports will over time develop a familiarity with the ways in which a product is vulnerable, and, given appropriate training and support, can begin to apply this knowledge directly to the product without having to wait for vulnerability reports to arrive from elsewhere. We have followed this exact path at the CERT/CC: tools such as Dranzer [23] and BFF [24] came directly out of vulnerability analysts applying broad knowledge of vulnerabilities gained from detailed analysis of reports toward the development of automated discovery processes.

By far the easiest way to build a vulnerability testing infrastructure is the use of virtualization technologies. Many different virtual machine environments can be built for receivers of vulnerability reports to verify or clarify the reports they receive.

At the CERT/CC, we maintain a firewalled testing network in which virtual machines can be placed for testing. We also maintain a few permanent pre-configured servers on this network (HTTP web servers, etc.) to allow easy testing of certain classes of vulnerabilities, such as "drive-by" browser downloads. Much of this infrastructure could be replicated entirely in a virtual network within a single machine, or in a cloud-based environment if desired.

Be sure your analysts have proper access to any necessary software needed for testing. This includes maintaining appropriate software licenses for proprietary software, although in many cases free and/or open source alternatives are available.

Toolkits often include the following:

- Virtualization platform, often a need to support multiple operating systems
- Debuggers
- Source code analysis tools
- Binary analysis tools (decompilers, etc.)
- Network sniffing tools
- Hex editors
- Text editors
- Visualization (often built into other tools)

Vendors with more hardware-centric products may need to additionally maintain more physical gear or specialized test bench equipment in order to have sufficient capacity to confirm reports.

References