5.4 Multiparty CVD

Most of the interesting cases in CVD involve more than two parties, as these are the cases where the most care must be taken. Automation of the process can help somewhat, but the impact technology can have on the problem is limited by the inherent complexities involved in trying to get numerous organizations to synchronize their development, testing, and release processes in order to reduce the risk to users.

From a coordinator's perspective, it can be difficult to be fair, as you're almost guaranteed to either miss some downstream vendor or wind up with one or more vendors ready to release while everyone is waiting for the other vendors to catch up. We discuss this conundrum further in Section 6 below.

The FIRST Vulnerability Coordination SIG [1] has published its "Guidelines and Practices for Multi-Party Vulnerability Coordination and Disclosure" [2] which we strongly recommend reading.

Success at multiparty coordination has more to do with understanding human communication and organization phenomena than with the technical details of the vulnerability. The hard parts are nearly always about coordinating the behavior of multiple humans with diverse values, motives, constraints, beliefs, feelings, and available energy and time. The vulnerability details may dictate the "what" of the response, but to a large degree human social behaviors decide the "how."

In the next few subsections we discuss a number of issues that we have observed in performing multiparty CVD over the years.

Multiple Finders / Reporters

If one person can find a vulnerability, another person can too. While documented instances of independent discovery are relatively rare, independent discovery of vulnerabilities can and does happen.

Perhaps the best known example of multiple finders is Heartbleed (CVE-2014-0160).

In part because of the complexity of the coordinated disclosure process, a second CVE identifier was assigned to the same vulnerability (CVE-2014-0346) [3].

We discuss independent discovery further in Section 6.5.

Complicated Supply Chains

Many products today are not developed by a single organization. Instead, they are assembled from components sourced from other organizations.

For example, software libraries are often licensed for inclusion into other products. When a vulnerability is discovered in a library component, it is very likely that not only does the originating vendor of the library component need to take action, but all the downstream vendors whose products use it need to take action as well. Complex supply chains can increase confusion regarding who is responsible for coordinating, communicating, and ultimately fixing vulnerabilities, leading to delays and systems exposed to unnecessary risk.

Historically, the Android ecosystem provides a clear example of this phenomenon. A vulnerability in a library component used in the Android operating system might have to be fixed by the library developer, then incorporated into the Android project by Google, followed by the device manufacturer updating its custom build of Android, and by the network carrier performing its customizations and testing before finally reaching the consumer's device. Each additional step between the party responsible for fixing the code and the system owner (the device user, in this case) reduces the probability that the fix will be deployed in a timely manner, if at all [4].

At the CERT/CC, we often find it useful to distinguish between vertical and horizontal supply chains. While the vertical supply chain is most common, we do occasionally need to navigate horizontal supply chains in the course of the CVD process.

Vertical Supply Chain

In a vertical supply chain, a vulnerability exists in multiple products because they all share a dependency on a vulnerable library or component. One vendor originates the fix. Many vendors then have to incorporate the originating vendor's fix into their products. Many vendors have to publish documents, distribute patches, and cause deployers to take action.

One example of a CVD process following a vertical supply chain is as follows: a vulnerability might be initially identified in product X, but is then isolated to a library that product X includes as a dependency. In this case, the library developer must be engaged as another party to the coordination process in the role of patch originator.

The complexity does not end there though. Once the library vendor has completed its patch, not only does the vendor of product X have to integrate the fix, but all the other vendors that include the library need to update their products as well. We have done this kind of coordination in the past with vulnerabilities affecting MS-SQL [5], Oracle Outside In [6], and so on. The cascading effects of library vulnerabilities often result in significant subsets of users left vulnerable while they await their product vendor's updates.

Horizontal Supply Chain

Even more complex in terms of coordination are cases where multiple products implement the same vulnerability, which is the primary characteristic of a horizontal supply chain. Examples include vulnerabilities arising from underspecified protocols, design flaws, and the like. Luckily these kinds of vulnerabilities are rare.

CVD can become quite complicated when they occur, because multiple vendors must originate fixes for their own implementations. Many such cases combine with each originating vendor's downstream vertical supply chain as well, which only serves to increase the complexity. Many vendors have to publish docs and distribute patches, leading to deployers needing to take multiple actions.

Multiple implementation vulnerabilities can sometimes result from widespread copying of vulnerable code examples from books or websites or from developer tutorials that ignore or intentionally disable security features in order to simplify the learning process. While we cannot place the entirety of the blame for widespread Android SSL Man-in-the-Middle vulnerabilities such as VU#582497 [7] on any specific phenomenon, our spot checks of some of the vulnerable apps made it clear that parallel implementation of the same errors was a contributing factor in many of the affected apps.

In that case, we identified more than 23,000 distinct apps [8], and coordinated with thousands of vendors.

A more pernicious example of multiple implementation is a vulnerability whose root cause lies in the specification or reference implementation of a network protocol. Because most vendor's products will specifically test for compatibility with these reference artifacts, such cases usually imply that every product supporting that feature will need to be fixed. Multi-originator cases can be very complex to coordinate. The SNMP vulnerabilities found in 2002 via the OUSPG PROTOS Test Suite c06-snmpv1 [9,10,11,12] represented just such a case, and stand to this day as the most complex disclosure case the CERT /CC has ever coordinated.

Mass Notifications for Multiparty CVD

In their Usenix Security 2016 paper, Stock and colleagues [13] examined issues surrounding large-scale vulnerability notification campaigns.

In this work, which focused on notifying website operators of vulnerabilities in their sites, they highlight significant difficulty in establishing a direct communication channel with vulnerable sites.

The following is from their conclusion:

How do we inform affected parties about vulnerabilities on large scale? Identifying contact points remains the main challenge that has to be addressed by the Internet society, including network providers, CERTs, and registrars. We imagine that this problem could, for example, be tackled by centralized contact databases, more efficient dissemination strategies within hosters/CERTs, or even a new notification channel or trusted party responsible for such notifications. Until we find solutions to the reachability problem, the effects of large-scale notifications are likely to remain low in the future.

< 5.3 Two-Party CVD | 5.5 Response Pacing and Synchronization >

References

1. FIRST, "Vulnerability Coordination SIG," [Online]. Available: https://www.first.org/global/sigs/vulnerability-coordination. [Accessed 17 May 2017].
2. FIRST, "Multi-Party Coordination and Disclosure," [Online]. Available: https://www.first.org/global/sigs/vulnerability-coordination/multiparty. [Accessed 6 June 2017].
3. Codenomicon, "The Heartbleed Bug," 29 April 2014. [Online]. Available: http://heartbleed.com/. [Accessed 16 May 2017].
4. W. Dormann, "Supporting the Android Ecosystem," 19 October 2015. [Online]. Available: https://insights.sei.cmu.edu/cert/2015/10/supporting-the-android-ecosystem.html. [Accessed 23 May 2017].
5. J. P. Lanza, "Vulnerability Note VU#484891 Microsoft SQL Server 2000 contains stack buffer overflow in SQL Server Resolution Service," 26 July 2002. [Online]. Available: https://www.kb.cert.org/vuls/id/484891. [Accessed 23 May 2017].
6. W. Dormann, "Vulnerability Note VU#916896 Oracle Outside In 8.5.2 contains multiple stack buffer overflows," 20 January 2016. [Online]. Available: https://www.kb.cert.org/vuls/id/916896. [Accessed 23 May 2017].
7. W. Dormann, "Vulnerability Note VU#582497 Multiple Android applications fail to properly validate SSL certificates," CERT/CC, 3 September 2014. [Online]. Available: https://www.kb.cert.org/vuls/id/582497. [Accessed 16 May 2017].
8. W. Dormann, "Android apps that fail to validate SSL," 29 August 2014. [Online]. Available: https://docs.google.com/spreadsheets/d/1t5GXwjw82SyunALVJb2w0zi3FoLRIkfGPc7AMjRF0r4. [Accessed 16 May 2017].
9. University of Oulu, "PROTOS Test-Suite: c06-snmpv1," 2002. [Online]. Available: https://www.ee.oulu.fi/research/ouspg/PROTOS_Test-Suite_c06-snmpv1. [Accessed 16 May 2017].
10. I. A. Finlay, S. V. Hernan, J. A. Rafail, C. Dougherty, A. D. Householder, M. Lindner and A. Manion, "Multiple Vulnerabilities in Many Implementations of the Simple Network Management Protocol (SNMP)," CERT/CC, 12 February 2002. [Online]. Available: https://www.cert.org/historical/advisories/CA-2002-03.cfm. [Accessed 16 May 2017].
11. I. A. Finlay, "Vulnerability Note VU#854306 Multiple vulnerabilities in SNMPv1 request handling," CERT/CC, 12 February 2002. [Online]. Available: https://www.kb.cert.org/vuls/id/854306. [Accessed 16 May 2017].
12. I. A. Finlay, "Vulnerability Note VU#107186 Multiple vulnerabilities in SNMPv1 trap handling," CERT/CC, 12 February 2002. [Online]. Available: https://www.kb.cert.org/vuls/id/107186. [Accessed 16 May 2017].
13. B. Stock, G. Pellegrino and C. Rossow, "Hey, You Have a Problem: On the Feasibility of Large-Scale Web Vulnerability Notification," in *25th USENIX Security Symposium*, 2016.