Tyler Small

Dr. Smotherman

3220 Project 1

15 May, 2018

Operating System Vulnerabilities

CVE stands for common vulnerabilities and exposures. The purpose of CVE documentation is to create a standardized dictionary for vulnerabilities and exposures identified. CWE is common weakness enumeration. A weakness is different from a vulnerability in that a weakness is an error that may lead to a vulnerability and may not necessarily be a vulnerability itself. CVSS is common vulnerability scoring system, a way to score severity of vulnerabilities (5.). A CWE can lead to a CVE and CVSS is a way to score CVEs

There are a few terms that should be defined before continuing. All of which being types of vulnerabilities. The first, a denial of service vulnerability, is a weakness in the OS (operating system) that makes the system susceptible to the making of resources unable to function properly. These are usually perpetrated in order to use the device as a bot to perform further DoS attacks on other devices. Code execution, being the second vulnerability, is one that allows a hacker to run arbitrary code (more than likely malicious) on someone else's PC, this includes downloading malicious software without the owner's consent or even connecting to malicious servers. An overflow vulnerability is one that allows a hacker to take advantage of a flaw in preventative overflow mechanisms and allows that hacker to overwrite code in memory. This is different from memory corruption, in that a memory corruption vulnerability is one that is exploited via memory left open by the operating system, rather than currently used memory being overwritten. A bypass vulnerability is more commonly known as a back-door, and are usually achieved by a hacker getting around user or OS set security authentication. This can be done by anything from a password leak to software that changes permissions. Lastly, are gain information and gain privilege vulnerabilities. Both involve a hacker gaining access to private data, gain information being the exploitation of an OS fault to access private information, and gain privilege being the use of a fault to gain a heightened security privilege allowing a hacker to access documents and items that would otherwise be hidden to users of a lower privilege.

With the vulnerability types defines and laid out, we can talk about real examples. The three main operating systems have had a number of vulnerabilities in 2018. The distributions for Linux were 42 denial of service attacks, 1 code execution, 11 overflow attacks and 3 memory corruption, 2 bypass somethings, 5 gain informations and 0 gain privileges. The distributions for Mac OSX were 31 denial of service attacks, 37 code executions, 25 overflows, 24 memory corruptions, 15 bypass somethings, 10 gain informations and 0 gain privileges. And finally, the distributions for Windows 10 were as follows: 6 denial of services, 10 code executions, 4 overflows, 0 memory corruptions, 13 bypass somethings, 35 gain informations and 1 gain privilege.

CVE 2018-0825 will be the first example of a vulnerability in this paper, rated according to CVSS standard as critical. This vulnerability "is a remote code execution vulnerability that could give an attacker control of a targeted system if they are logged into their Windows PC with administrator user rights" (threatpost). Since this bug is particular is classified as an attack vector, code can be executed by simply viewing an email. This is different than most execution attacks, in that the user is usually prompted to click a link to a stealth download - usually via phishing. Meaning this vulnerability in particular is much more dangerous than traditional means of execution attacks. CVE 2018-0825 is enabled by CWE 119. CWE 119 is a heap overflow attack. This means, according to (9.) that if memory were to be controlled, possibly via an out of bounds array access, then arbitrary code may be ran on the user's computer. This is how the vulnerability described was able to inject code without clicking on any links.

CVE 2018-4160 is an example of an overflow attack. The root of this vulnerability is a call to the macro bcopy, in MacOS's NFS Diskless Boot Implementation. According to (12.) the use of bcopy to copy blocks of data implements a size type of size_t being an unsigned integer. Therefore, if a negative size were given as an argument to the function call, it would cause buffer overflow - giving the application a very large buffer. This is a problem, because if the size buffer given to the macro is larger than that of the destination buffer, there could not only be the obvious problem of overflow, but if the destructing calls do not know the size of block to deallocate, then that can also lead to a memory corruption attack from memory being left open after the destructors are called. This vulnerability is enabled by a CWE 125, an out-of-bounds read. This happens when the software reads data past the end, or before the beginning of the intended buffer. One exploitation of this weakness allows the implementer of the attack to read information in memory locations not usually able to be accessed. It also potentially causes program crashing via excess data being read. Assume a string is declared and the program looks at data outside of memory. If there is no NULL terminated character in the memory being read, then the string's size will grow into an unhandlable amount and crash. (10.). According to Backhouse, the exploit on the MacOS causes the kernel to crash when a negative size is given, so the likely problem is the latter.

CVE 2018-5344 is the last attack I will talk about. This attack is considered a denial of service attack, with varying severities. The reported use of the attack is that the loop can be used by another process even after the loop counter has decremented to 0. The reason according to the Github commit is stated to be insufficient serialization in the lo_release() function. Since the loop can be used in a way that it never shuts down, the attack is considered a denial of service attack. This type of vulnerability is an example of a CWE 416 or a "Use After Free (11.)". The description on the MITRE website is listed to be "referencing memory after it has been freed (11.)", this can cause not only a program to crash, but also to either use unexpected values and even run outside code. The most likely use of a CWE 416 is to use already freed memory to execute user code or cause an infinite loop.

Certain parts of an operating system are more likely to be subject to error than other parts according to Chou. "[D]river code has error rates three to seven times higher for certain types of errors than code in the rest of the kernel (1.)". pg 1. This follows logical thinking, considering driver code is most of an operating system's code base. But as Chou points out, even when corrected for code size,

operating system drivers are still strikingly the most bug-rich environment. The explanation for clustering of bugs in an operating system is that "for large systems with many programmers, dependent errors will cause error clustering, where parts of the OS have much higher error rates than the global error rate (1.)" pg 11. This could be do a variety of reasons, be it the ignorance of the developers or an uncaught fault in the code compounding. Another interesting subject is the lifetime of bugs within an operating system. "The average bug lifetime is about 1.8 years with median around 1.25 years (1.)", pg 10. It's important to note that bugs that were less likely to cause overflows were less likely to be caught as quickly. Noting that medians are not affected by outliers like means are, it is important to realize some bugs go much longer without being caught.

References

(1.) A. Chou, et al., "An Empirical Study of Operating Systems Errors," Proc. 2001 ACM Symp. Operating Systems Principles (SOSP 01), pp. 73-88.

(2.) C. Kanaracus, "Two Nasty Outlook Bugs Fixed in Microsoft's Feb. Patch Tuesday Update," blog, 13. Feb. 2018; https://threatpost.com/two-nasty-outlook-bugs-fixed-in-microsofts-feb-patch-tuesday-update/129931/

(3.) D. Wiles, D. Dugal, "Common Vulnerability Scoring System SIG" https://www.first.org/cvss/

(4.) K. Backhouse, "Negative Integer Overflows in Apple's NFS Diskless Boot (CVE-2018-4136, CVE-2018-4160)," blog, 25 Apr. 2018; https://lgtm.com/blog/apple_xnu_nfs_boot_CVE-2018-4136_CVE-2018-4160

(5.) L. Torvalds, "Loop: Fix Concurrent lo_open/lo_release," 6 Jan. 2018; https://github.com/torvalds/linux/commit/ae6650163c66a7eff1acd6eb8b0f752dcfa8eba5#diff-01765c0f9a2dc5c24c5a424b82b97b62

(6.) M. Base-Bursey, "Mobile Vulnerabilities: The Culprits Your Business Needs to Know About," blog, 2 Nov. 2017; https://www.wandera.com/blog/mobile-vulnerabilities-ios-android/.

(7.) Mitre Corp, "CWE List Version 3.1", 13 April, 2018 https://cwe.mitre.org/data/definitions

(8.) Mitre Corp, "Frequenty Asked Questions", 30 March, 2018

https://cwe.mitre.org/about/faq.html#A.1

(9.) Mitre Corp, "CWE-119: Improper Restriction of Operations within the Bounds of a Memory Buffer", 29 March, 2018 https://cwe.mitre.org/data/definitions/119.html

(10.) Mitre Corp "Out-of-bounds Read", 29 March, 2018 https://cwe.mitre.org/data/definitions/125.html

(11.) Mitre Corp "Use After Free", 29 March, 2018 https://cwe.mitre.org/data/definitions/416.html

(12.) T. Armerding, "What is CVE, its definition and purpose?", blog, 10 July 2017 https://www.csoonline.com/article/3204884/application-security/what-is-cve-its-definition-and-purpose.html