As shown on the list in *Measures and Metrics: Example Coverity* there are thirteen vulnerabilities that are found by Coverity Quality. To name a few there is memory corruptions, use of uninitialized data, control flow issues, insecure data handling and resource leaks. Each one of the ones studied by Coverity the vulnerabilities all share a common problem; they all cause issues. Many of these vulnerabilities conducted similar behavior where they leak private information or cause harm to a computer. Each one out of the thirteen have many resolutions on how to minimize the impact of an eventful exploit of any of these vulnerabilities. Then there are some out of the thirteen that have far fewer resolutions so it can be challenging to maintain. These vulnerabilities can be found throughout any piece of technology, and are not just limited to a specific device. Memory corruptions for example can be fixed in multitude of ways like how the Cyber Defense Laboratory in North Carolina State University had a solution. They designed a diagnosis engine to allow safe transfer of information. The use of uninitialized data occurs more frighteningly than many of us hope for. Comprised data can be the result of a vulnerability exploit carried out intentionally seeking information. Kernel Pointers has a lot of attention when it comes down to use of uninitialized data. It provides a lot of the necessary tools required to monitor and repair this type of vulnerability. Control flow issue vulnerabilities arises when the use of CRAs or, code reuse attacks, are used. This exploit can disrupt flow or many programs and perform malicious actions without leaving a trace. Learning that a vulnerability has been potentially lurking in a system without leaving a mark would increase security levels. Many leading software engineers are actively combatting this growing issue. There are many ways that insecure data handling can occur for example, not sufficiently storing passwords or critical information or choosing weak algorithms. One of the simplest ways to securely handle data is to

be very careful and mindful where you let this information out. It can be to another person or even written down somewhere. Most common resource leaks are due to a result in general software reliability problems. There are many cases where these vulnerabilities turn into exploits and causes a series of damage. Many things can be done to fix these problems has quickly as possible. Depending on the source code it will have different processes but will be the same. If you are using a computer that has a memory leak vulnerability currently in use the performance of the computer drastically decreases until immediate actions were made. There are a total of thirteen complicated vulnerabilities that will only grow in number. These are the few that will become into a list of dozens. The future of vulnerabilities will be come at a cost of loss of data and privacy. Understanding how to fix them as soon as we can will allow us to live a safer future.

Sources:

Lab, Mengchen Cao Orion Security, et al. “Different Is Good: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security.” *ACM Conferences*, 1 Nov. 2019, https://dl.acm.org/doi/pdf/10.1145/3319535.3345654.

J. Zhang, B. Qi, Z. Qin and G. Qu, "HCIC: Hardware-Assisted Control-Flow Integrity Checking," in IEEE Internet of Things Journal, vol. 6, no. 1, pp. 458-471, Feb. 2019, doi: 10.1109/JIOT.2018.2866164.

Turner, Rob. “What Are the Consequences of Memory Leaks in Programming?” *TechDay*, 2020, https://techdayhq.com/community/articles/what-are-the-consequences-of-memory-leaks-in-programming.

Cho, Haehyun. “Exploiting Uses of Uninitialized Stack Variables ... - Usenix.” *Usenix*, Arizona State University, 2020, https://www.usenix.org/system/files/woot20-paper-cho.pdf.