Methodology, Insights, Limitations, and Future Work Summary

Methodology

Tools Used Visual Studio Code, Jupyter Notebook, Python's Pandas and Matplotlib.pyplot

Obtaining the Data An online Kaggle dataset that came in a CSV file. The file was read into a pandas data frame in the program.

Data Cleaning After preforming exploratory data analysis of the CVE dataset, it was quickly discovered that there were several data cleaning steps that needed to be taken, including:

- Removing unnecessary columns
- Renaming some columns to be shorter and all lowercase
- Removing all rows that have a value of 'NaN' in any of the columns—the implications of removing this data were considered and after thoroughly searching through the dataset, it was found that there were approximately 867 rows containing a NaN value compared to the 88,776 rows that did not; because of this, the decision to remove the rows that contained a NaN was made since it would not have has a significant impact on the dataset as a whole from a statistical view
- Removing all rows that contained '***REJECT***' in the summary column—there were approximately 16 rows containing rejected values the should not have been used
- In total, 883 rows were removed from the dataset either due to containing a NaN value or containing a ***REJECT*** value.

About the Dataset After cleaning the data, the dataset contained 12 columns, 88,776 rows, over 1 million data points, and CVE data ranging from 1999-2019.

Questions Several key questions drove the analysis of data, with further questions being generated as some of the preliminary questions were answered, some of such questions include:

- What is the most common vulnerability (CWE name) found in the dataset? What is the least common?
- What year has the highest recorded CVEs associated to it? What year has the lowest?
- What is the average CVSS score?
- What is the trend of recorded CVEs? Increasing? Are any spikes seen in recorded CVEs? Any dips?

Insights

Notable Results

- Most common vulnerability: Buffer Overflow—CWE name of 'Improper Restriction of Operations within the Bounds of a Memory Buffer' with 12,305 occurrences
- 2018 saw the all-time high of recorded CVEs in the dataset, with 14,855 CVEs associated with the year
- 2000 saw the all-time low of recorded CVEs in the dataset, with 18 CVEs associated with the year
- The average count of CVEs recorded per year is 4,227 CVEs
- Stating in 2017, there was a drastic increase in recorded CVEs
- The average CVSS score across the dataset was, 6.0—making the average impact level a 'Medium,' according to the CVSS impact scale

Recommendations Developers, cybersecurity analysts, and IT professionals alike (amongst many others) can obtain great value from the data analysis of CVEs. Such analysis can aid in describing current trends and patterns, discovering the most severe and exploitable CVEs and vulnerabilities, and visualizing over a decade's worth of data in a concise manner for non-IT personnel. Using the conclusions drawn by data analysts, several recommendations can be proposed, including:

- Developers should harden software, websites, and applications to prevent and/or mitigate the most common and severe vulnerabilities—in the case of this dataset, the buffer overflow vulnerability was found to be the most common, which can be prevented by secure coding practices.
- Organizations and IT departments should prioritize patching the most severe CVEs first, with Critical and High-scoring vulnerabilities needing to be remediated first. Prioritizing the most severe and exploitable CVEs is crucial to ensure that the organization's critical assets are protected, and data analysis of such vulnerabilities can provide insights that greatly aid both IT personnel and business leaders alike.

Limitations Despite discovering significant findings and conclusions in the dataset, there remains several important limitations that should be addressed regarding the validity of some of the results, including:

- The average count of CVEs per year is likely greatly skewed by the incredibly low counts recorded from 1999-2005 in the dataset compared to the much higher counts seen starting in 2007 and on. While it is good to note that as technology advances and becomes more accessible to people, CVEs will increase, the counts seen in the beginning years of this dataset my not be representative of the true CVE counts recorded (in the dataset).
- The CVSS scoring is inconsistent across the dataset due to the many revisions the grading scale has undergone and the fact it was released in 2005 (CVSSv1) despite the dataset beginning in 1999. The second version of CVSS (CVSSv2) was released in 2007 and the third version was released in 2015 (CVSSv3), with the scale undergoing minor revisions every few years as well. Because of this, a 2010 CVE with a score of 7.1 may not be equal in terms of severity to a 2017 CVE with the same score.

Future Work Moving forward, there would be several future projects to work on and questions to be answered to further the analysis of CVE data, including:

- Modify the code to pull directly from either reputable websites with CVE repositories or CVE databases using Python's web scraping or SQL capabilities—this would allow for a more streamlined approach to loading in the data in the program.
- Automate the process of pulling down the most up-to-date CVEs by creating a script that runs the program every specified amount (i.e. every other day, every week, etc.).
- Include more information on the data regarding the patches/remediations available for the CVE, affected products, vendors, etc. to perform further statistical analysis on the data—this would allow for greater projection on the most common vulnerabilities, see the relationship between the most common CVEs/vulnerabilities and its exploitability, the commonly affected and targeted products, and much more.