

Detection and Improvement of Data Quality Inconsistencies in Public Vulnerability Datasets

Final Project Report

Master of Science in Information Quality

University of Arkansas, Little Rock

Submitted by:

Raja sekhar reddy Panditi

T00670350

Supervised by:

Dr. John R. Talburt

Organizational Supervision by:

Dr. Philip Huff, Ph.D.

Assistant Professor, Computer Science

Emerging Analytics Center(EAC)

Table of Contents

Executive Summary.....	3
Deliverables	3
Introduction	4
Data Collection and Extraction.....	6
Data Quality Assessment	9
• Representational Consistency	9
• Data Completeness.....	11
• Data Uniqueness.....	13
• Data Relevancy	14
Exploratory Data Analysis	15
• Vulnerability Source Analysis.....	15
• Threat severity distribution	16
• Exploitability Analysis:	17
• Vendor and Product Analysis.....	18
• Vulnerability Impact trends	20
Conclusion.....	22
Acknowledgement	22
References	23
Appendix A.....	24
• A.1 Web Application preview to load data from local server	24
• A.2 Organization data listing Applicable Vulnerabilities	24
Appendix B : Pipeline Overview	25

Executive Summary

The George W. Donaghey Emerging Analytics Center™ (EAC) at the University of Arkansas in Little Rock is a research and development center that houses energetic group of faculty and student researchers. The EAC's focus of research is in the areas of virtual reality, data visualization, cybersecurity, and interactive technologies in the context of complex and big data. A research group at EAC has been working on a Machine Learning based Vulnerability and patch management application with a focus on prediction and improvement of software patching. As part of this research work the team is heavily reliant on National Vulnerability Database(NVD) and other disparate sources of information to track vulnerabilities and analyze security trends. Although these databases have achieved great success in vulnerability disclosure and mitigation, there is a growing concern for their information quality and consistency.

The main objective of this project was to uncover inconsistencies in NVD data through systematic investigation and address these discrepancies. Creation of data fusion center that lists software products and applicable vulnerabilities is one of the milestones of this project. I have designed web scraping scripts for extracting the vulnerability information and corresponding CVE(Common Vulnerability Enumerator) details. I had also created a web application based on Angular framework for facilitating front end activities and Django REST framework for serving the data requests through web API. A python class was created to address data quality issues and perform data transformation before loading it to the relational database. I had made use of celery Django framework for an up-to-date data access of vulnerability and advisory information through distributed message passing.

As a result of this project, vulnerability information and corresponding security advisories present in different domains is successfully extracted to a local relational database. Through exploratory data analysis, inconsistencies of data quality were addressed. This eliminated the problem of machine learning models having to rely on data imputation methodologies for missing information. Creation of data fusion center with curated dataset containing various software products and applicable vulnerabilities helped in matching advisory/patch information. Leveraging the benefits of data with improved quality, A Natural Language Processing(NLP) based recommender solution was built to reduce human efforts needed for software product vulnerability matching.

Deliverables

Following were the list of deliverables that I submitted to Name of sponsor as part of my graduate project.

- Web scraping scripts for extracting vulnerabilities, advisories, and exploitability details.
- Database schema design in MySQL for storage and maintenance of extracted data.
- RESTful Web Service designed based on Django framework for serving data provision.
- Jupyter Notebook with exploratory data analysis results and relevant visualizations.

Introduction

Software bugs are a fact of life and security-related issues are no different. Following security best practices and timely update of software products are essential in maintaining the cyber health of IT systems within an enterprise. Any weakness that an attacker can exploit to perform unauthorized actions is termed as a vulnerability and several specification languages exist for describing them. Software vulnerabilities are inevitable and are bad in way that software bugs are bad. There is an important distinction that effects the use of vulnerability in a way that they apply only to a particular range of IT systems.

Every publicly identified vulnerability is assigned to a Common Vulnerability Enumeration(CVE) ID with CVE Numbering Authority(CNA), such as MITRE. A CVE ID uses the format CVE-YYYY-NNNNN, where YYYY is the year in which a vulnerability is registered and NNNNN is a unique number. Basic information about a vulnerability like product, version, and description are also tagged with this CVE number.

A total of 1,61,790 unique CVE ids are registered till date and looking at the data from past three years one can observe, on average 1450 CVE are reported each month. This counts to more than 50 CVE each day. Although not all of these are vulnerabilities, but since people don't publicly report all vulnerabilities, the number could be much higher.

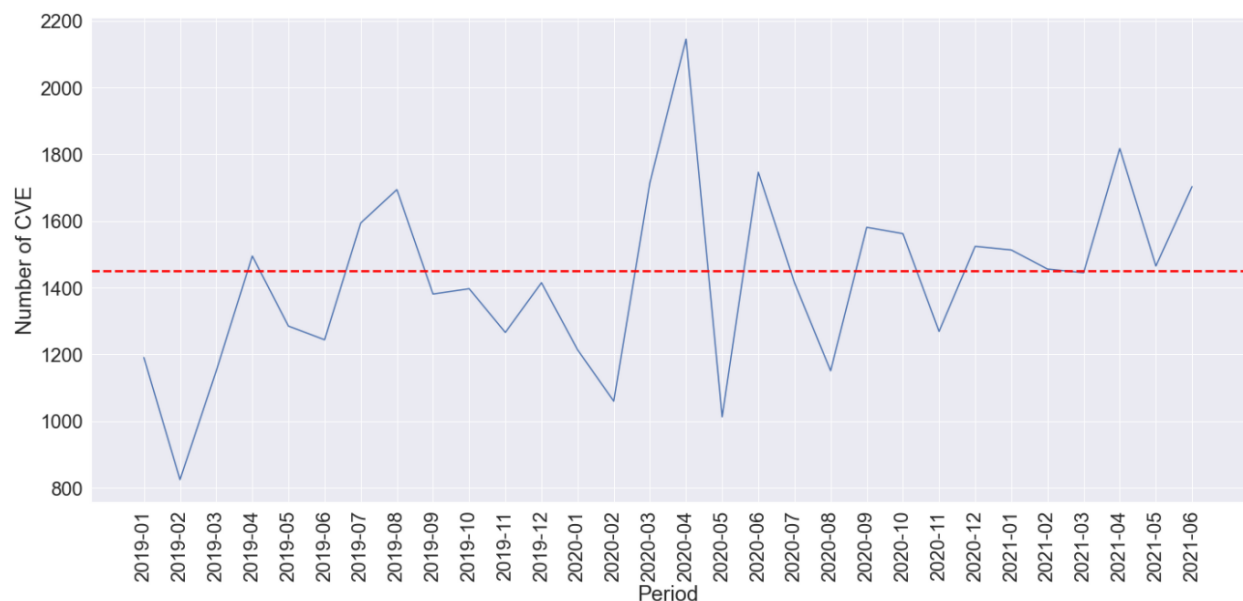


Figure 1 : Number of vulnerabilities reported plotted against date of publication

From the threat proliferation graph plotted between cumulative sum of vulnerabilities reported and the published year, it is evident that number of known threats continues to grow. Figure 2 shows the threat proliferation range over time and nearly 60% of all the vulnerabilities reported so far are found to occur in the last 5 years.

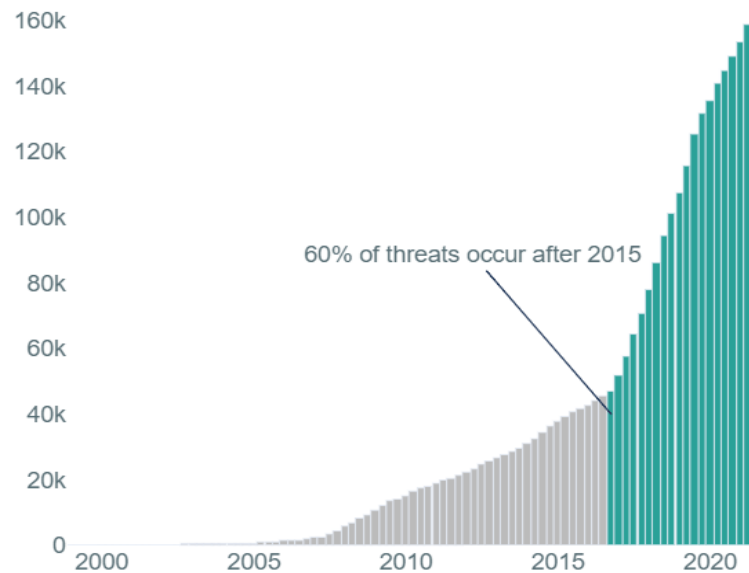


Figure 2 : Proliferation of Vulnerabilities over time

The aim of this project was to build an ETL pipeline towards creation of data fusion center that would help serve prediction and automation of security patching. Performance of Machine Learning models built for this purpose will be strongly affected by the quality of data being fed to them. This project also uncovers possible inconsistencies of data quality when relying on public vulnerability datasets for this purpose.

Advanced web scrapping scripts have been developed for data extraction and a RESTful web service was created to help serve data provision. Data transformation logic was implemented to address data quality issues and finally a dataset was created mapping software products with their applicable vulnerabilities. Exploratory data analysis has been performed on the resulting dataset to visualize trends and seasonality of vulnerability impacts over the years.

Data Collection and Extraction

Vulnerability details from NVD

The U.S. government's repository of public vulnerability information, maintained by National Institute of Standards and Technology(NIST) has been chosen to extract vulnerability details.

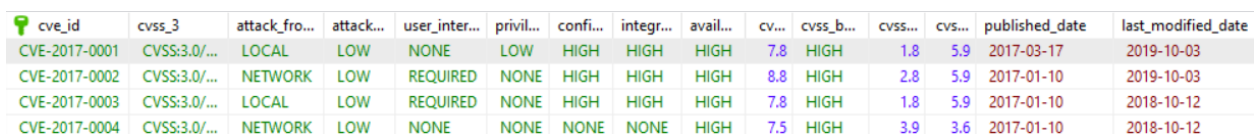
<https://nvd.nist.gov/feeds/json/cve/1.1/>

With over 60% of the vulnerabilities are reported in past 5 years, web scrapping script was developed to extract CVE vulnerabilities reported from year 2017 till date. I have extracted the following features of each vulnerability reported and further mapped them with the software products affected.

Attribute	Data Type	Description
cve_id	VARCHAR(30)	Unique CVE id assigned by NVD
cvss_3	VARCHAR(50)	Common Vulnerability Scoring System(Version 3)
attack_from_where	VARCHAR(50)	Attack vector
attack_complexity	VARCHAR(50)	Complexity of vulnerability attack
user_interaction	VARCHAR(50)	Authenticity level required for the attack
privileges_required	VARCHAR(50)	Authorization level required for the attack
confidentiality_impact	VARCHAR(50)	Vulnerability impact on confidentiality of the system
integrity_impact	VARCHAR(50)	Vulnerability impact on integrity of the system
availability_impact	VARCHAR(50)	Vulnerability impact on availability of the system
cvss_base_score	DECIMAL(3,1)	Common Vulnerability Scoring System base score
cvss_base_severity	VARCHAR(10)	Common Vulnerability Scoring System severity score
cvss_exploitability_score	DECIMAL(3,1)	Common Vulnerability Scoring System exploit score
cvss_impact_score	DECIMAL(3,1)	Common Vulnerability Scoring System impact score
published_date	DATE	Date when vulnerability is published
last_modified_date	DATE	Most recent date of action on vulnerability reported
description	LONGTEXT	Short summary of the vulnerability details
exploitability	VARCHAR(255)	Exploitability check of the vulnerability
exploit_id	INT	Exploit id of the vulnerability

Table 1: Metadata details for Vulnerabilities from NVD data

I have loaded all vulnerability details into MySQL workbench. Table 1 lists the metadata details of vulnerabilities from NVD, and Figure 1 represents sample records stored in MySQL for reference.



cve_id	cvss_3	attack_from...	attack...	user_inter...	privil...	confi...	integr...	avail...	cv...	cvss_b...	cvss...	cvss...	published_date	last_modified_date
CVE-2017-0001	CVSS:3.0/...	LOCAL	LOW	NONE	LOW	HIGH	HIGH	HIGH	7.8	HIGH	1.8	5.9	2017-03-17	2019-10-03
CVE-2017-0002	CVSS:3.0/...	NETWORK	LOW	REQUIRED	NONE	HIGH	HIGH	HIGH	8.8	HIGH	2.8	5.9	2017-01-10	2019-10-03
CVE-2017-0003	CVSS:3.0/...	LOCAL	LOW	REQUIRED	NONE	HIGH	HIGH	HIGH	7.8	HIGH	1.8	5.9	2017-01-10	2018-10-12
CVE-2017-0004	CVSS:3.0/...	NETWORK	LOW	NONE	NONE	NONE	NONE	HIGH	7.5	HIGH	3.9	3.6	2017-01-10	2018-10-12

Figure 1: Vulnerabilities from NVD data

Vulnerability to Package Mappings

This data source refers to vulnerability(CVE) details mapped with software packages as released by different vendors in the market. I have developed a web scrapping script that would extract these mappings as listed by the following vendors in corresponding web repositories.

- Debian: <https://security-tracker.debian.org/tracker/data/json>
- RedHat: <https://www.redhat.com/security/data/metrics/rpm-to-cve.xml>
- SUSE: <https://www.suse.com/security/cve/>
- Ubuntu: <https://ubuntu.com/security/cve>

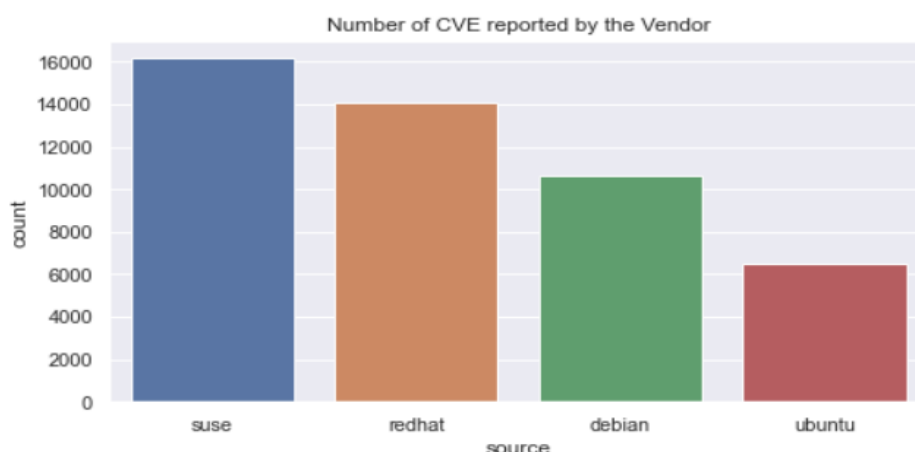


Figure 3: Bar plot showing number of CVE reported by software vendor

Table 3 represents the metadata details of the data source Vulnerability to Package mappings and Figure 3 shows records extracted to MySQL database.

Attribute	Data Type	Description
cve_id	VARCHAR(255)	CVE id of the vulnerability
package	VARCHAR(255)	Name of the software package affected
operatingsystem_cpe	VARCHAR(255)	Operating System CPE
Source	VARCHAR(255)	Name of the vendor
record_updated	DATETIME	Date and time when the record is added

Table 3 : Metadata details of Vulnerability to package mappings

cve_id	package	operatingsystem_cpe	source	record_updated
CVE-2020-28589	tinyobjloader	cpe:2.3:o:canonical:ubuntu_linux:21.04	ubuntu	2021-09-07 00:55:54.073324
CVE-2021-20314	libspf2	cpe:2.3:o:canonical:ubuntu_linux:21.04	ubuntu	2021-09-07 00:55:54.073324
CVE-2021-29988	mozjs78	cpe:2.3:o:canonical:ubuntu_linux:21.04	ubuntu	2021-09-07 00:55:54.073324
CVE-2021-29984	mozjs78	cpe:2.3:o:canonical:ubuntu_linux:21.04	ubuntu	2021-09-07 00:55:54.073324
CVE-2021-29981	mozjs78	cpe:2.3:o:canonical:ubuntu_linux:21.04	ubuntu	2021-09-07 00:55:54.073324
CVE-2021-29981	thunderbird	cpe:2.3:o:canonical:ubuntu_linux:21.04	ubuntu	2021-09-07 00:55:54.073324

Figure 3: Vulnerability to package mappings

Security Advisory Details

Security Advisory is a basic explanation that contain information about the issues concerning security of software systems. This is an important service as users of the software can refer to these for crucial details like reported vulnerabilities and corresponding fixes like patches or updates. There is a wide variety of security advisories that span software space from those serving large enterprise software products to those for more niche communities of small projects.

Before a vulnerability is published on a security repository, it first needs to be discovered. Owners of the software product will be notified once it is discovered, and a timeline is set before is it made public. At this point, the information is published on relevant advisory, available for public consumption. I have extracted the security advisories reported by following prime vendors in the market.

Oracle: <https://www.oracle.com/security-alerts/public-vuln-to-advisory-mapping.html>

Intel: <https://www.intel.com/content/www/us/en/security-center/default.html>

Cisco: <https://api.cisco.com/security/advisories/cvrf/all>

Adobe: <https://helpx.adobe.com/security/security-bulletin.html>

Microsoft: <https://api.msrmc.microsoft.com/sug/v2.0/en-US/affectedProduct>

Attribute	Data Type	Description
patch_id	VARCHAR(255)	Software Patch id
Vendor	VARCHAR(255)	Name of the software vendor
advisory	VARCHAR(255)	Advisory id
Title	VARCHAR(255)	Advisory title
release_date	DATE	Date when advisory is reported
updated_date	DATE	Most recent date when advisory is updated
Href	VARCHAR(255)	Web page reference for the advisory
description	VARCHAR(255)	Short summary of advisory details
record_updated	DATETIME	Date and time when record is updated to database

Table 4 : Metadata details of Security Advisories

patch_id	v...	advisory	title	release...	updated_d...	href	description	record_updated
CSCv43704	cisco	cisco-sa-wsa-dos-fmHdKswk	Cisco Web ...	2021-10-06	2021-10-06	https://tools.cisc...	A vulnerability in the pr...	2021-10-06 22:15:38.761741
CSCvy86528	cisco	cisco-sa-ise-info-disc-pNXLhdp	Cisco Identi...	2021-10-06	2021-10-06	https://tools.cisc...	A vulnerability in the w...	2021-10-06 22:15:38.777315
CSCvx60178	cisco	cisco-sa-esa-url-bypass-sGfsDrp	Cisco Email...	2021-10-06	2021-10-06	https://tools.cisc...	A vulnerability in the an...	2021-10-06 22:15:38.792974
CSCvx85812	cisco	cisco-sa-ipphone-arbfileead-NPdtE2Ow	Cisco IP Ph...	2021-10-06	2021-10-06	https://tools.cisc...	A vulnerability in the de...	2021-10-06 22:15:38.808565
CSCvy18258	cisco	cisco-sa-dnac-infodisc-KyC6YncS	Cisco DNA ...	2021-10-06	2021-10-06	https://tools.cisc...	A vulnerability in the A...	2021-10-06 22:15:38.824225
CSCvy84939	cisco	cisco-sa-cvdsd-xss-fvdj6HK	Cisco Visio...	2021-10-06	2021-10-06	https://tools.cisc...	A vulnerability in the w...	2021-10-06 22:15:38.839815
CSCvz38781	cisco	cisco-sa-anyconnect-lib-hija-cAFB7x4q	Cisco AnyC...	2021-10-06	2021-10-06	https://tools.cisc...	A vulnerability in the sh...	2021-10-06 22:15:38.855542

Figure 5: Security Advisory details stored in MySQL database

Data Quality Assessment

Advisory Details

Representational Consistency

Consistency of data in dataset is the extent to which data meet specific rules. Representational consistency defines that data formatting should be consistent throughout the system. A strict definition of representational consistency specifies that two data values drawn from separate datasets must not conflict with each other. Same data being represented in multiple formats causes difficulty in interpretation although consistency does not necessarily imply completeness.

Representational Consistency Assessment

Extraction of security advisories from multiple web repositories corresponding to each vendor showed that most of the fields with date formats is represented in different forms. Furthermore, data published by single vendor itself is lacking representational consistency, often leading to failure of web scrapping scripts. Representing attributes of date type in uniform format is necessary to comply with RDBMS data consistency principles. Also, this data is used in lag time analysis to understand the time intervals in vulnerability life cycle.

Solution

To address the representational consistency of date fields, I have utilized the Python regular expressions to match the incoming data with accepted format. If the data fails to match, then it is verified against one of the following formats shown in Table 1(a) and is passed through data formatting scripts.

Vendor	Source format	Target format
Adobe	MM-DD-YYYY	YYYY-MM-DD
	MM/DD/YYYY	YYYY-MM-DD
Oracle	2021-October-19	2021-10-19
	2021-Oct-19	2021-10-19
Intel	October 19, 2021	2021-10-19
	Oct 19, 2021	2021-10-19
Microsoft	2021-10-19T07:00:00+00:00	2021-10-19
Cisco	2021-10-19T07:00:00+00:00	2021-10-19

Table 1(a) : Data Inconsistency formats represented by each vendor

This resolved the problem of inconsistency of data at source level itself before being loaded into MySQL database for further consumption. Figure 1(a) below explains the date field analysis before and after converting to required format.

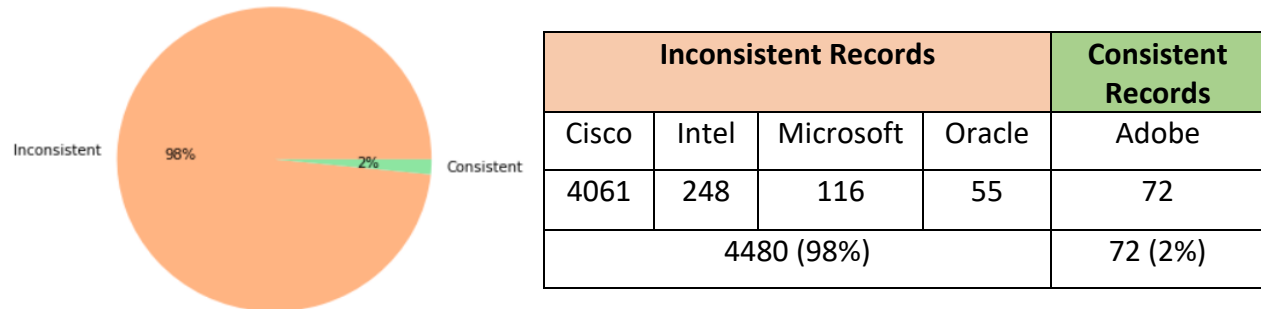


Figure 1(a) Date field analysis for Representational Consistency

Measurement

Representational consistency is used to determine how consistent the data is across multiple sources.

Metric Definition

$$\begin{aligned}
 \text{Consistency metric} &= 1 - (\text{Number of inconsistent records} / \text{total number of records}) \\
 &= 1 - (4480 / 4552) \\
 &= 0.015
 \end{aligned}$$

All the data in published date and last updated date has been converted to single acceptable format by MySQL. Table below shows the statistics of consistency metric before and after the data quality issue has been addressed.

Data Field	Inconsistent Values		Consistency Metric	
	Before	After	Before	After
Published Date	4480	0	0.015	1
Last Updated Date	4480	0	0.015	1

Table 1(b) inconsistent record count and consistency metric results of data cleaning

There is a significant improvement of consistency metric for both published date and last updated date field values. It has been improved from 0.015 to 1. As almost 98% of the records were missing consistency, data formatting scripts have helped the data to be cleaned at extraction phase itself. This resulted in complying with data consistency guidelines by RDBMS and the data is represented consistently within the database.

Data Completeness

In data quality framework, data completeness refers to the degree in which all data in a data set is available. It is also a measure of the data's ability to effectively deliver all the required values that are available to meet current and future business demand.

Data Completeness Assessment for publication date of vulnerability

Completeness of data in vulnerability database can heavily impact vulnerability tracking and trend analysis. There can be lags in the reporting of any given CVE entry in public vulnerability data sources like National Vulnerability Database(NVD). These records have only publication date which indicates the date when entry is added to database. However, it is important to have date when vulnerability is first made public along with publication date. This helps security analysts in prioritizing patching by considering how long a vulnerability is made public before an entry is made.

For example, CVE-2020-8899 is first made public on 2020-01-28 however, it is officially published on 2020-05-14 leaving a time gap of about 104 days. Entry present in NVD database corresponding to this vulnerability contains only publication date i.e., 2020-05-14 making it difficult to understand it's life cycle.

Solution

Vulnerabilities reported by Computer Emergency Response Team Coordination Center(CERT/CC) has both public date and publication date. To address the data completeness issue of date when vulnerability is made public, I have created a web crawler that will extract vulnerability information from CERT/CC. From the extracted vulnerability information, I have listed all the CVEs tagged and calculated lag time by subtracting the date when vulnerability made public from its publication date. Lag time is the number of days after estimated disclosure date when a vulnerability enters the NVD database.

CERT/CC Vulnerability Notes Database: <https://www.kb.cert.org/vuls/bypublished/desc/>

Table 2(a) shows the sample records with vulnerability details and corresponding lag time.

cve	vu_id	public_date	published_date	updated_date	lag
CVE-2000-0673	VU#32650	2000-09-26	2000-07-27	11/29/2000	61 days
CVE-2000-0482	VU#35958	2000-09-26	2000-06-05	4/5/2001	113 days
CVE-2000-0892	VU#22404	2000-09-26	2000-09-26	10/25/2001	0 days
CVE-2000-0201	VU#25249	2000-09-26	2000-03-01	4/12/2004	209 days
CVE-2000-0891	VU#5962	2000-09-26	1997-08-15	6/26/2001	1138 days

Table 2(a): Lag time of the vulnerabilities

I have calculated the Cumulative Distribution Function(CDF) of vulnerability lag times and plotted a line graph. Figure 2 shows the percentage of CVEs withing a lag time. About 30% of the vulnerabilities have zero lag time. The growth of vulnerabilities by lag time slows after accounting for the vulnerabilities with a lag of less than a week.

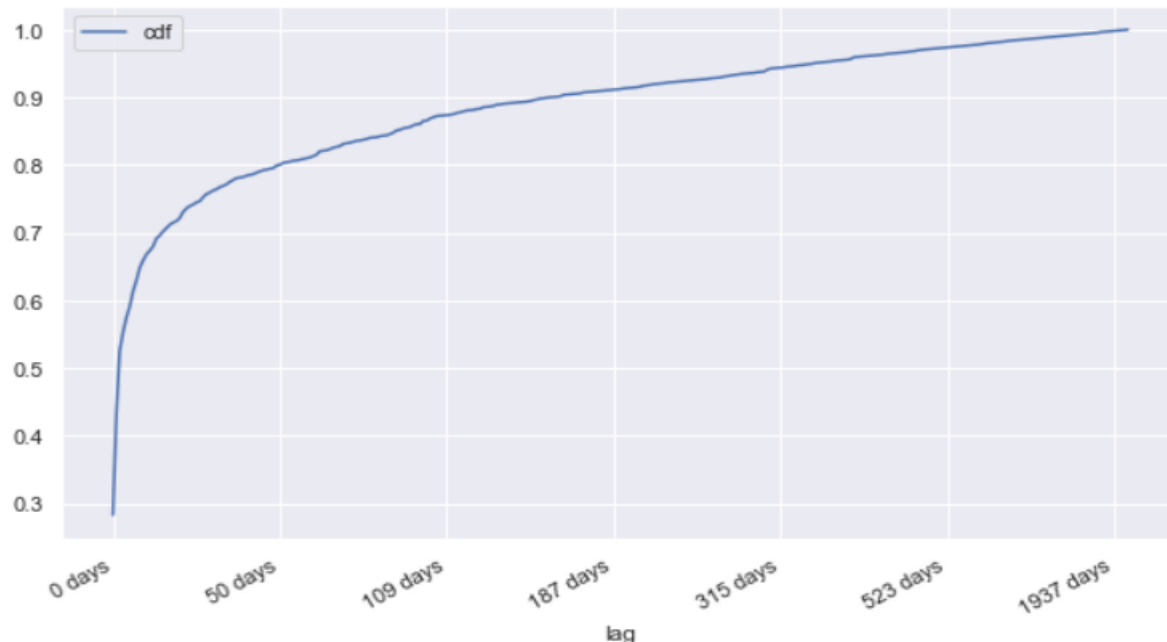


Figure 2: CDF of vulnerability lag times

Measurement

Data Completeness metric is used to define how complete the data is to support exploratory analysis.

$$\begin{aligned}
 \text{Completeness} &= 1 - (\text{number of incomplete records} / \text{total number of records}) \\
 &= 1 - (\text{number of records with non-zero lag times} / \text{total number of records}) \\
 &= 1 - (1584 / 2209) \\
 &= 0.28
 \end{aligned}$$

For all the vulnerabilities with non-zero lag time, public date is populated along with publication date and for remaining records public date and publication date are same. Table 2(b) represents the data completeness assessment results before and after data transformation.

Data Field	Inconsistent Values		Consistency Metric	
	Before	After	Before	After
Public Date	1584	0	0.28	1

Table 2(b) : Data Completeness measurement before and after data cleaning

Data Uniqueness

Uniqueness attribute of data quality is the most critical dimension for ensuring no duplication or overlaps. It defines the extent to which expected attributes are unique in your dataset. This metric ensures no duplicate records present in the dataset.

Data Uniqueness Assessment for vendor and software packages

Vulnerability to package mappings extracted from multiple sources often leads to a situation where same CVE and package combination might have been reported by different vendors. These however needs to be filtered out to maintain unique records in data fusion center of applicable vulnerabilities. Applicable vulnerabilities are mappings of software assets with appropriate vulnerabilities.

Solution

I have grouped all the extracted vulnerability to package mappings by CVE and package name. Aggregating all the groups to find out the count gave an idea of how many unique CVE to package mappings published by multiple vendors. While filling out the applicable vulnerability mappings, I have made a look up against already existing records to avoid duplicates being loaded into the database. Figure 3(a) shows the proportion of duplicate and unique records found and Figure 3(b) represents number of CVE to package mappings reported by each vendor

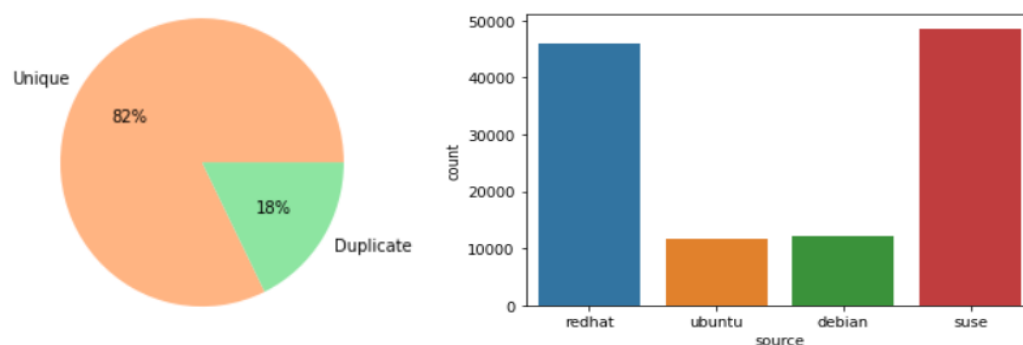


Figure 3(a) & 3(b) representing data uniqueness statistics of CVE to package mappings

Measurement

Uniqueness = $1 - (\text{number of duplicate records} / \text{total number of records}) = 0.83$

Unique vulnerability to package mappings were retained and Table 3 shows the statistics of data uniqueness assessment before and after data cleaning.

Operation	Total records	Duplicate records	Uniqueness Metric
Before filtering	118400	20978	0.83
After data cleaning	97422	0	1

Table 3 : Data Uniqueness measurement before and after data cleaning

Data Relevancy

Data Relevancy is an important dimension of data quality that refers to the extent in which data is useful for the purpose of which they were collected. Vulnerabilities reported by the software vendors often register a unique CVE number for tracking. However, some issues have not been assigned CVE numbers, but are still tracked by their database. In such cases, the system automatically assigns a unique name. These names are not stable and can change when the database is updated. Such records should not be used as external references in vulnerability and threat analysis as they were subject to change.

Data Relevancy Assessment for vulnerabilities

Vulnerabilities reported by Debian vendor include records with CVE number associated and the records with system assigned names which are subject to change over time. The automatically generated names come in two flavors: the first kind starts with "TEMP-000000-". This means that no Debian bug has been assigned to this issue (or a bug has been created and is not recorded in the database). In the second kind of names, there is a Debian bug for the issue, and the '000000' part of the name is replaced with bug number.

Solution

Initially I have extracted all the unique CVE reported by Debian. A pattern matching script using Python regular expressions was created to filter out the records with vulnerability id starting with 'CVE' before loading into the database.

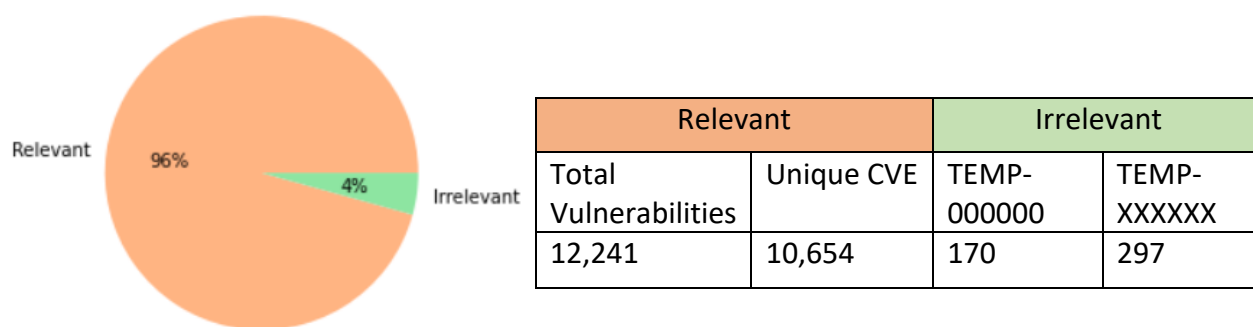


Figure 4: Data Relevancy Assessment statistics of vulnerabilities reported by Debian

Measurement

$$\text{Relevancy} = 1 - (\text{Number of irrelevant records} / \text{Total number of records}) = 0.96$$

Table 4 shows the results of data relevancy by passing the records through data filtering scripts.

Operation	Total records	Irrelevant records	Relevancy Metric
Before filtering	10654	467	0.96
After data cleaning	10187	0	1

Table 4 : Data relevancy statistics before and after data cleaning

Exploratory Data Analysis

Vulnerability Source Analysis

Often when vulnerabilities are published, corresponding source is listed along with CVE details. Figure 1 represents the top 10 sources based on the number of vulnerabilities reported for the past 10 years. Analysis results states that over 40% of the vulnerabilities reported so far are originated from source MITRE and about 75% of the reported vulnerabilities belong to top 10 sources listed below.

Source	Number of Vulnerabilities	% Of Vulnerabilities
MITRE	26170	43.6
Microsoft Corporation	3500	5.8
Oracle	2954	4.9
IBM Corporation	2054	3.4
Cisco Systems, Inc.	2023	3.3
Google(Android)	1766	2.9
Adobe System Incorporated	1600	2.6
Apple Inc.	1423	2.3
Qualcomm, Inc.	1363	2.2
Red Hat, Inc.	1354	2.2

Table 1 : Top 10 sources by reported vulnerability count for the past 10 years

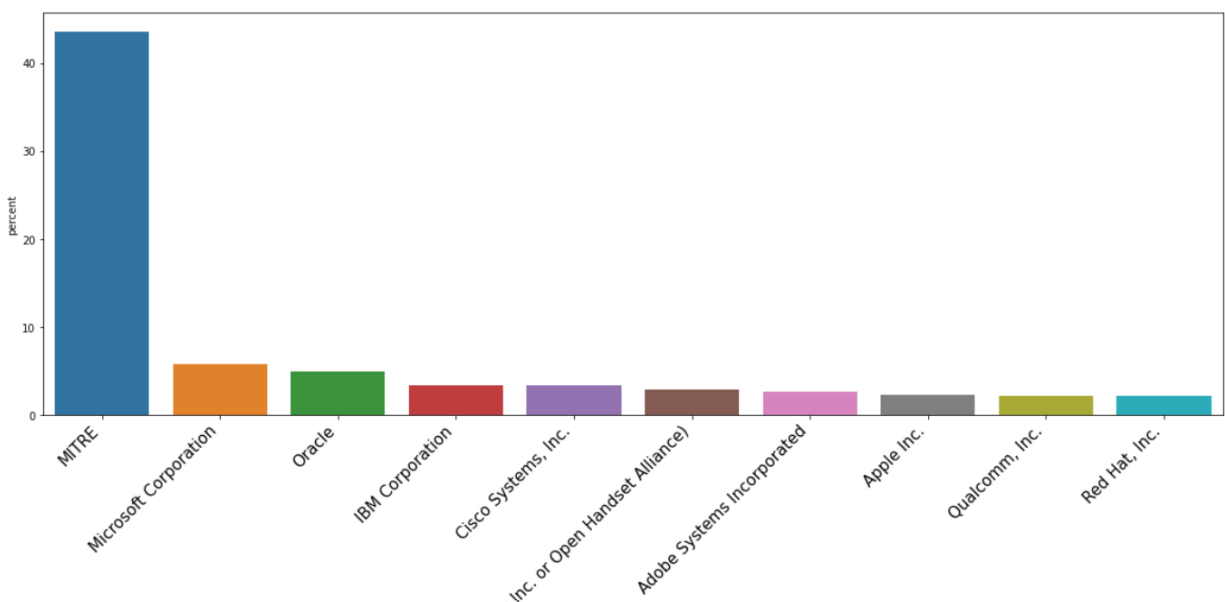


Figure 1: representing vulnerability percentage for the past 10 years by top 10 sources

Threat severity distribution

As thousands of vulnerabilities are reported each year, it is important to classify so that security practitioners can prioritize them based on severity level. Furthermore, understanding what fraction of vulnerabilities receives each severity level allows them to identify how many vulnerabilities they may need to contend with. For the security community, it is also important to understand whether reported vulnerabilities skew towards low severity or high severity ones. Figure 2 shows over 75% of vulnerabilities fall under medium threat category(4.0 – 6.9) with a thicker tail towards the higher end of spectrum.

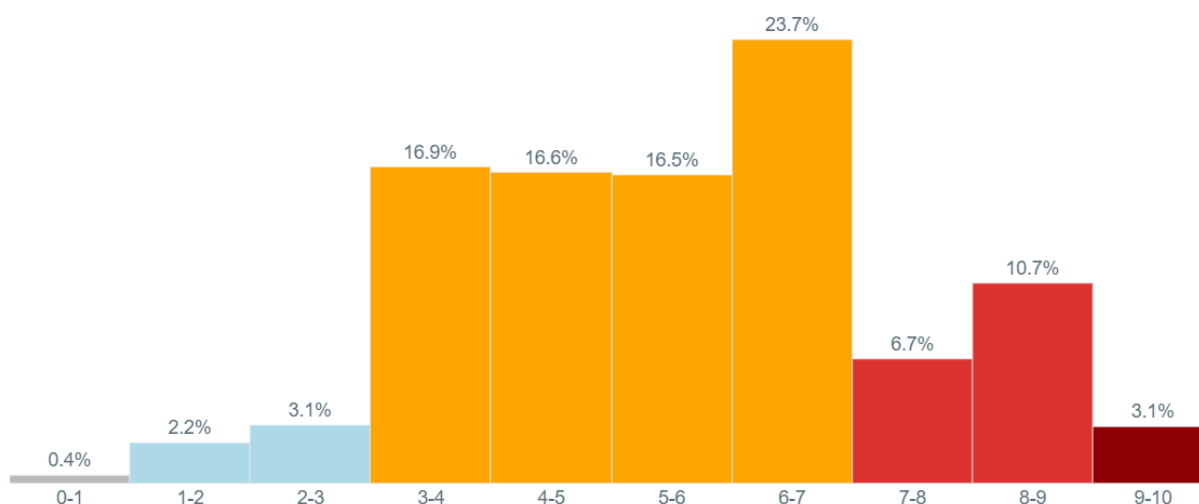


Figure 2 : Vulnerability threat severity distribution

Table 2 lists the Common Vulnerability Scoring System(CVSS) score ranges for each severity level as per version 3.

Label	Abbreviation	V3
None	-	0.0
Low	(L)	0.1 – 3.9
Medium	(M)	4.0 – 6.9
High	(H)	7.0 – 8.9
Critical	(C)	9.0 – 10.0

Table 2: Threat severity levels and associated score ranges

Exploitability Analysis:

In the NVD a CVE is associated with a vulnerability type under CWE classification. Developers and security analysts leverage this vulnerability type to understand attack vectors that may impact their software. Understanding which vulnerability types are associated with most critical CVEs is useful to prioritize which tools to invest in or investigate.

Figure 3 plots yearly rolling averages of top 10 vulnerability threat types and how they are changing over time. These threats are 10 most common, but their relative prominence is shifting. Code injection and SQL injection is becoming less common while cross-site scripting and input validation are on the rise. It is also identified that SQL injection has the most critical CVEs with almost twice as many as the next vulnerability type.

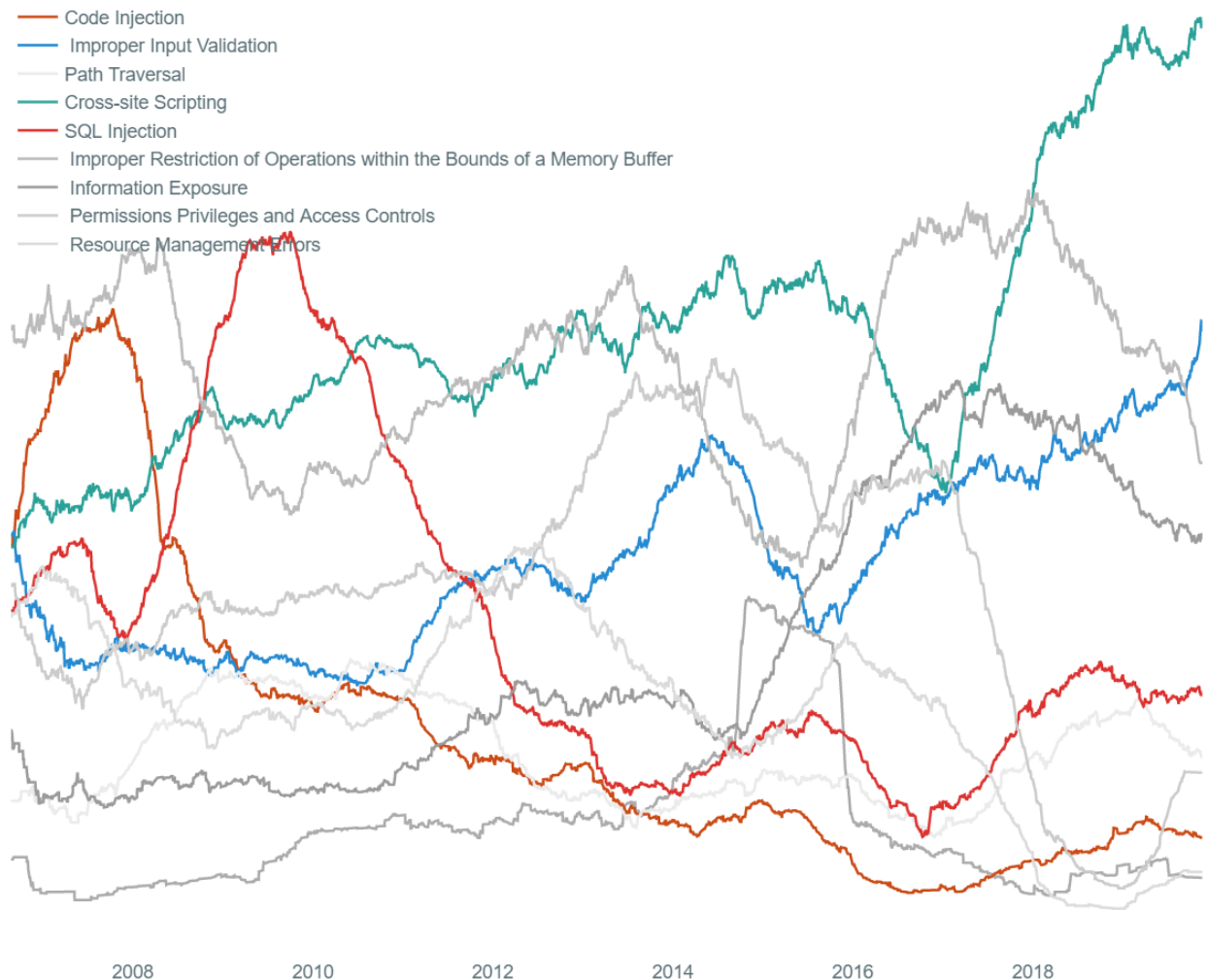


Figure 3: Trends of vulnerability threat types

Vendor and Product Analysis

Security analysts depend on list of vendors and products affected by a CVE to identify vulnerabilities affecting software they use or to monitor the security trends of software systems. Figure 4 shows the top 20 vendors by associated product count. It is observed that top vendors represent significant portion of products and nearly 40% of all the available products belong to these vendors. Figure 5 lists top 20 vendors by CVE count and results shows that 50% of the products affected by any vulnerability are distributed among these vendors.

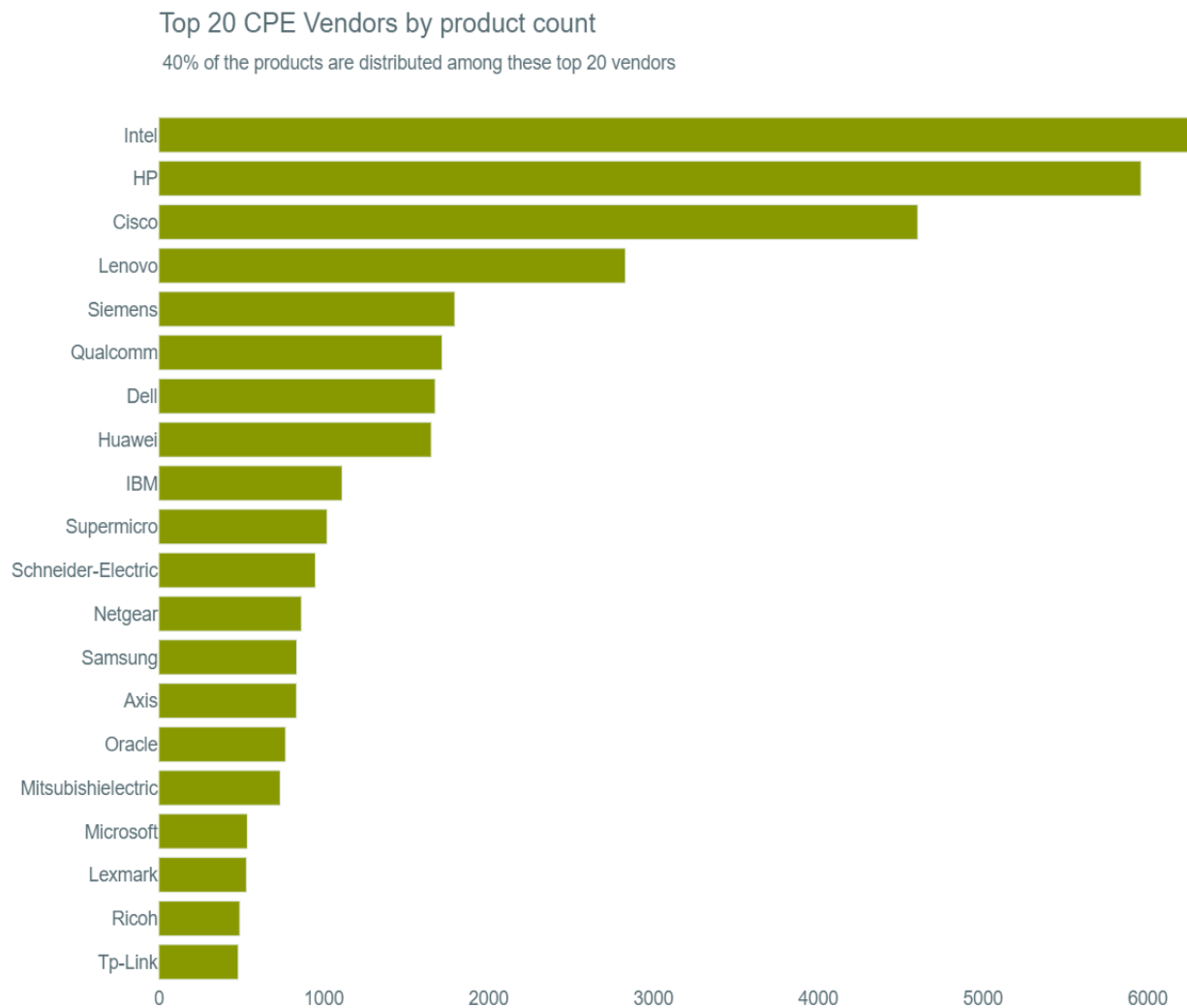


Figure 4 represents top 20 vendors by product count

It is interesting to note that top vendors by CVE count are different than those by product count, with only 4 common vendors. This difference clearly suggests the fact that concentration of CVEs among top vendors is not simply due to these vendors having wide range of products.

Figure 6 shows top 25 affected products and their associated system types. It is observed that most of these are either operating system based or browser-based products.

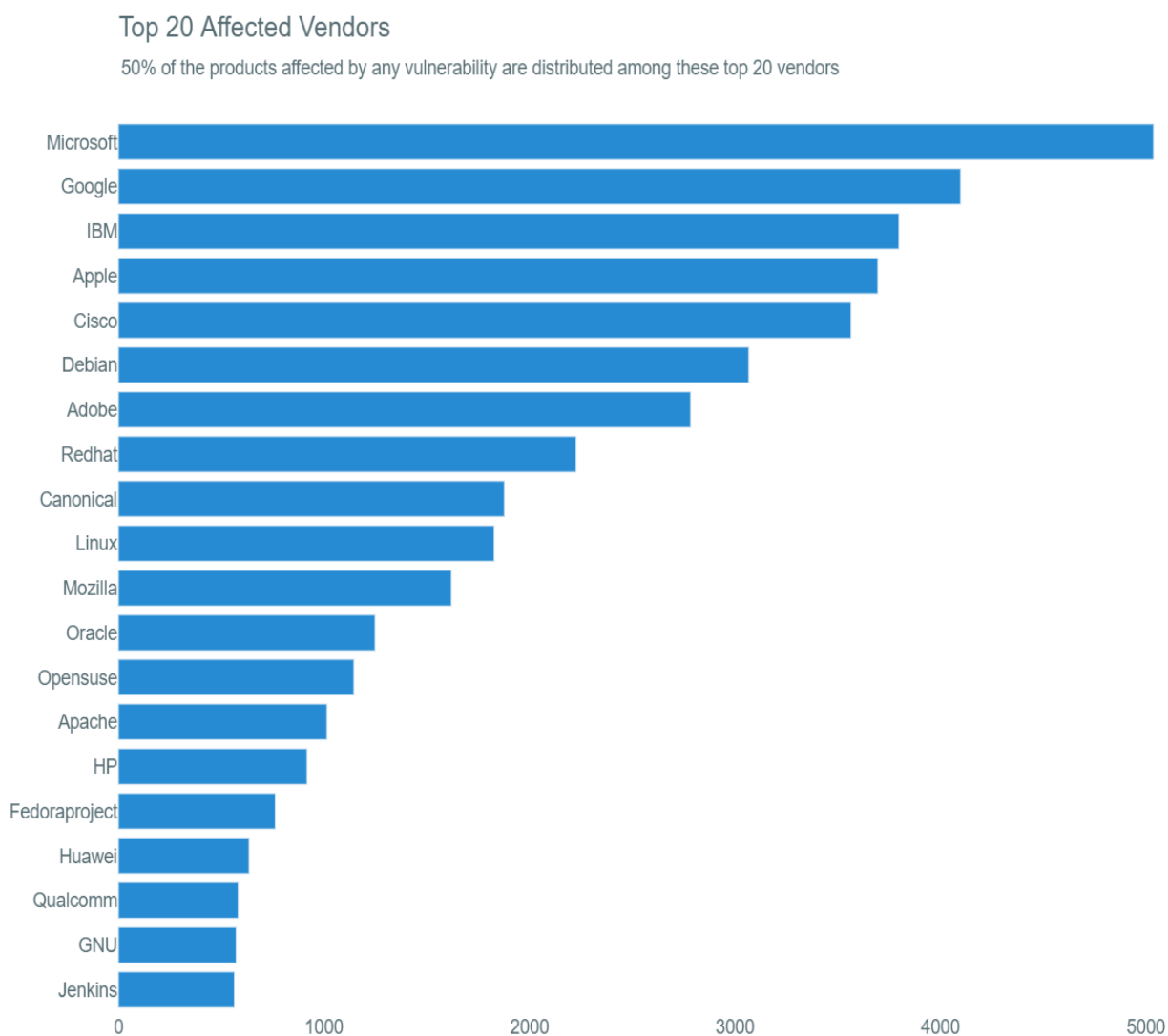


Figure 5 :Top 20 vendors by CVE count

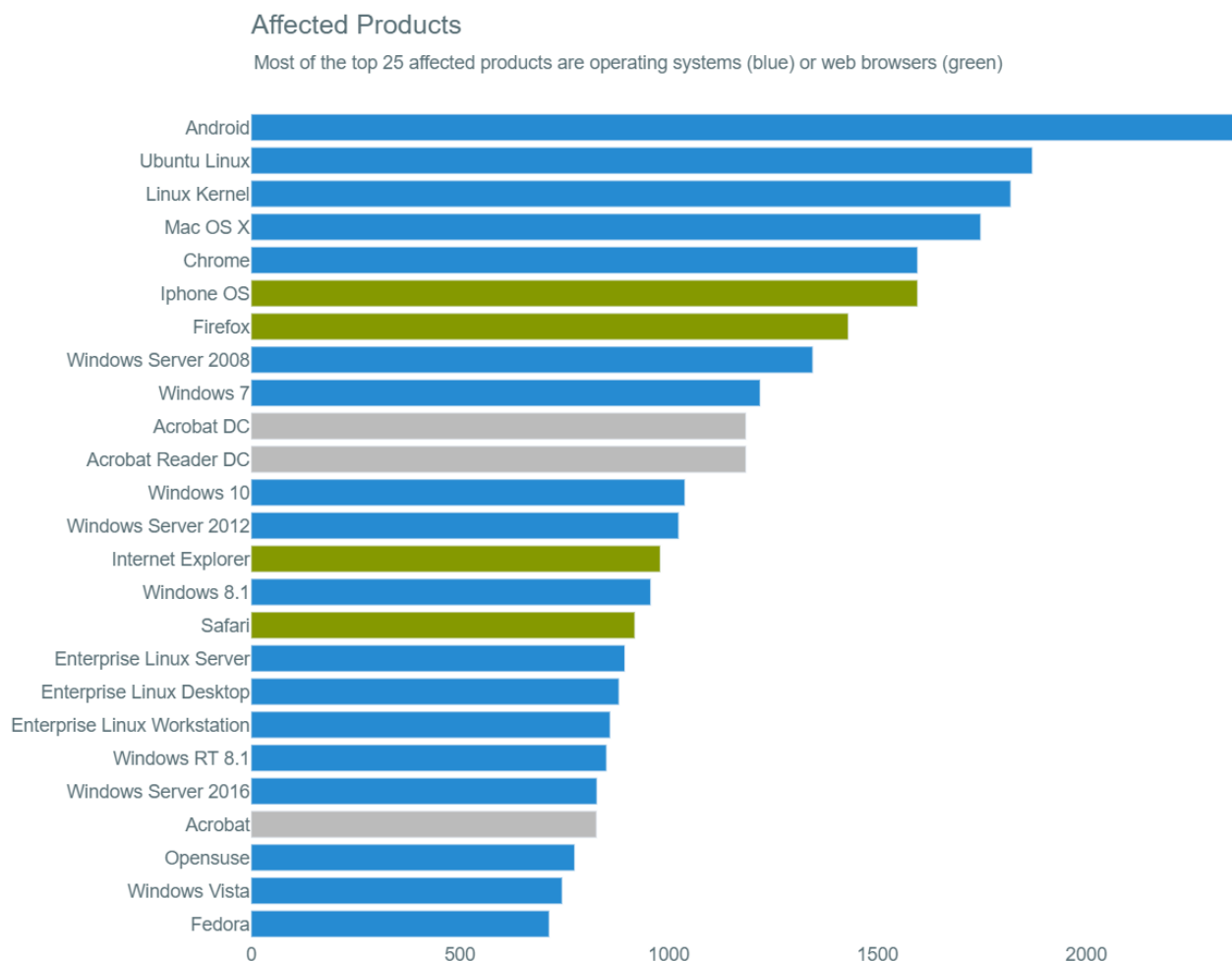
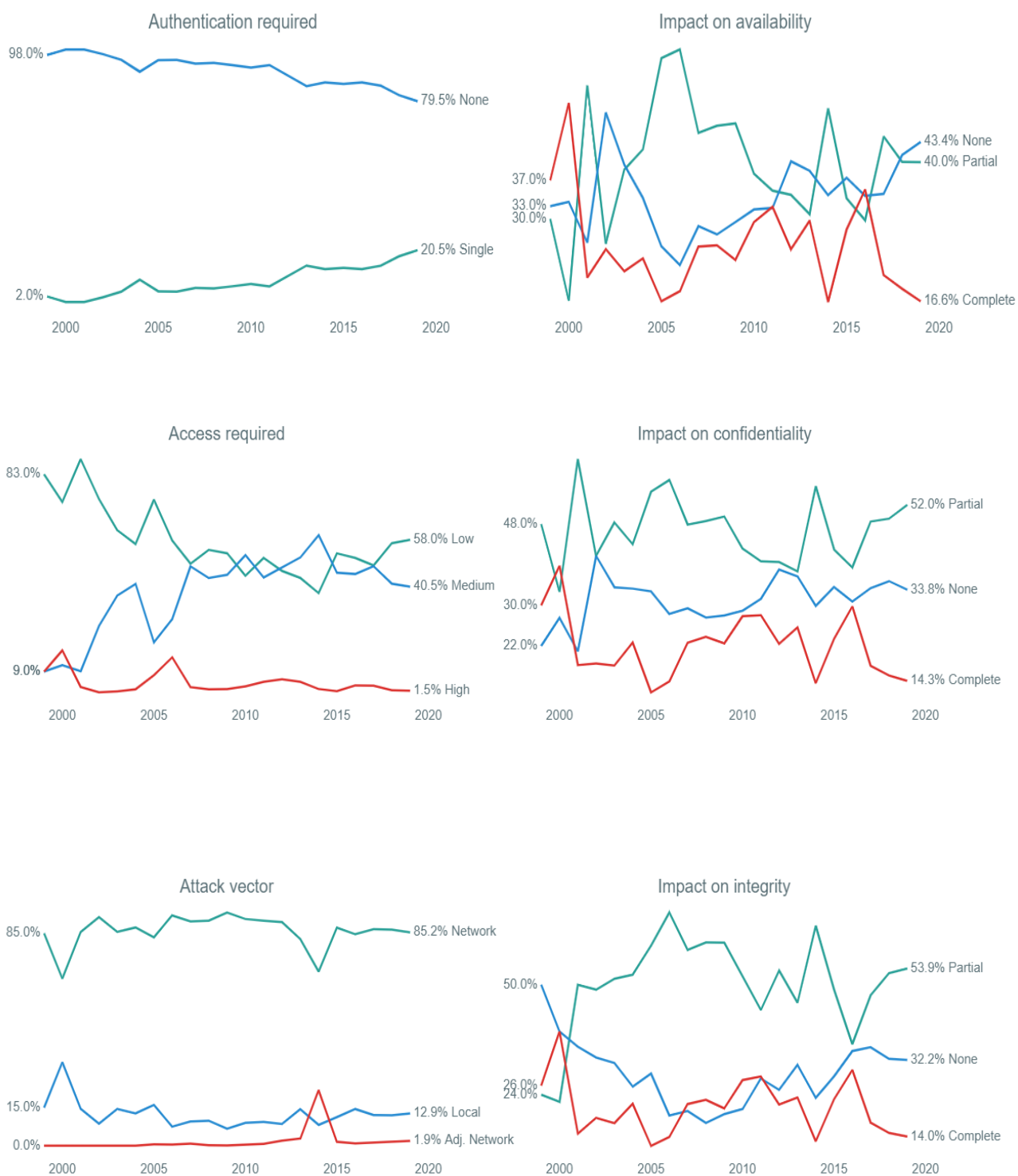


Figure 6 : Top 25 affected products by vulnerabilities

Vulnerability Impact trends

Every CVE record in NVD database is associated with different levels of impact corresponding to confidentiality, integrity, and availability. Figure 7 shows the trends of vulnerability impacts on these aspects over the years. It is observed that of all the vulnerabilities recorded so far, over 85% of the vulnerabilities are through network attack. Required authentication levels are periodically increasing for the vulnerability to occur.

**Figure 7 : Vulnerability impact trend analysis**

Conclusion

The objective of this project was to build an ETL pipeline that will extract the vulnerability details from public vulnerability database and create a data fusion center by addressing data quality issues. Vulnerability records are subject to rise over time, so the data extraction scripts were set to run in scheduled mode. These processes helped in acquiring up to date information. Maintaining good quality of data is necessary as this pipeline was designed to support the cause of rising trends on software security automation. During this project, I tried to address the data quality inconsistencies and offered solutions that helped in addressing these problems during data extraction phase.

Given the importance such a database as NVD for security operations, identifying, measuring, and fixing the inconsistencies is essential. In designing the ETL pipeline, the goal was to identify the root cause of these data quality issues and thus prevent them at source level. This was pursued through multi-sourced web scrapping and manual vetting. In the process of maintaining uniform data format across the database, about 98% of the extracted information on vulnerability to package mappings were transformed to single format. Exploring alternative channels for information on vulnerabilities helped in addressing the data completeness and this further cleared the barriers for carrying out lag time analysis. Identifying and removing duplicate records of vulnerability to package mappings helped in matching software products with applicable vulnerabilities in consistent manner. Identifying and addressing these inconsistencies further supported in carrying out exploratory data analysis and resulted in drawing important conclusions.

Acknowledgement

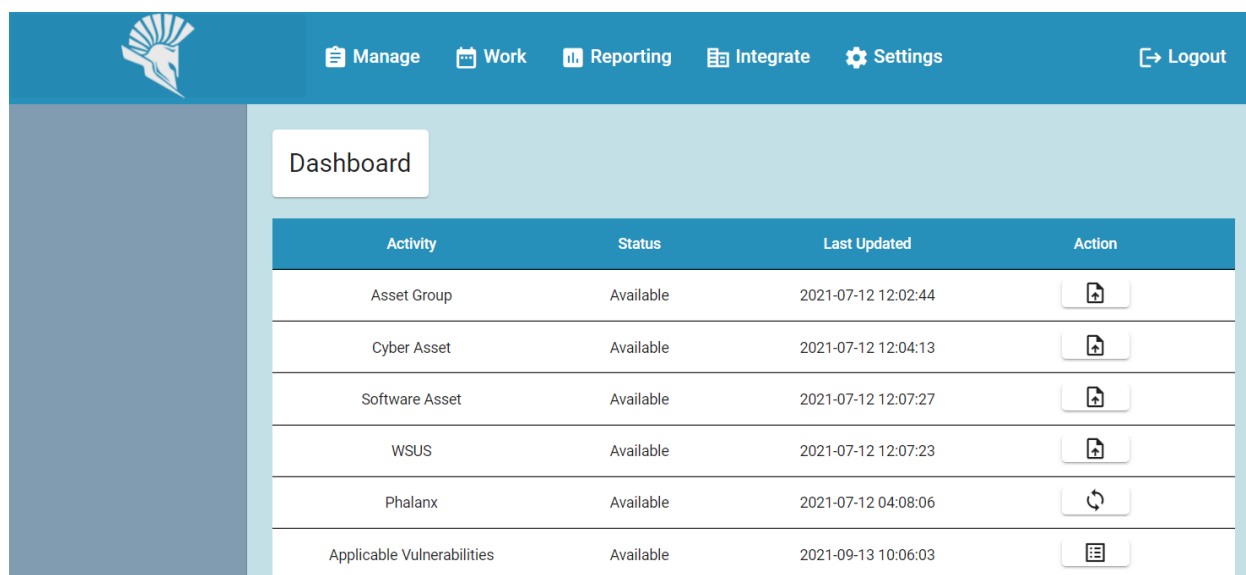
I take this opportunity to express my special thanks of gratitude to my project supervisor Dr. Philip Huff, for his dedicated support and guidance throughout the course of this project. I would also like to highlight amazing learning experience I had working with the SPARTAN team and be thankful to them for always motivating me with their passion for research. I would also like to thank the Emerging Analytics Center(EAC) at UALR for sponsoring and giving me a chance to work with resources needed for the project. I would also thank my Faculty Advisor for his timely instructions and detailed orientation for aligning me with the project guidelines. Above all it is my time here at the University of Arkansas at Little Rock that paved my way towards all possibilities, and I will always be grateful for its career-oriented curriculum and amazing faculty.

References

- [1]. Allodi, L., Banescu, S., Femmer, H., and Beckers, K. Identifying relevant information cues for vulnerability assessment using CVSS.” *In Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, CODASPY (2018), pp. 119–126.
- [2]. “NVD – National Vulnerability Database – NIST.gov” 2021, nvd.nist.gov[Online]. Available : <https://nvd.nist.gov/>.
- [3]. “CPE- Common Platform Enumeration, naming convention and standards”, 2021[Online]. Available: <https://cpe.mitre.org/specification/>.
- [4]. “NVD. Vulnerability Metrics – Common Vulnerability Scoring System, 2021” [Online]. Available: <https://nvd.nist.gov/vuln-metrics/cvss/>.
- [5]. “Towards the detection of inconsistencies in public security vulnerability reports. Retrieved November 1, 2021, https://www.usenix.org/system/files/sec19fall_dong_prepub.pdf.
- [6]. “VINCE – Vulnerability Information and Coordination Environment, 2021”[Online]. Available: <https://kb.cert.org/vuls/bypublished/desc/>.

Appendix A

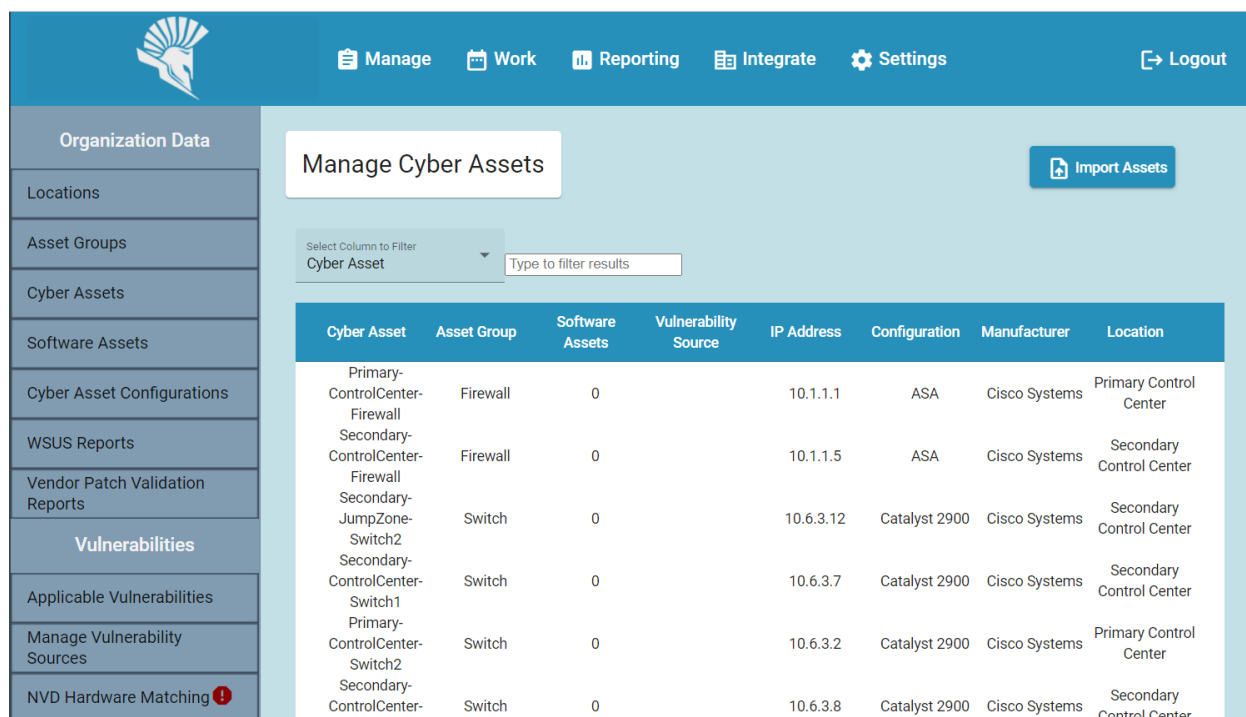
A.1 Web Application preview to load data from local server



The screenshot shows a web application dashboard with a blue header bar containing a logo and navigation links: Manage, Work, Reporting, Integrate, Settings, and Logout. The main content area is titled 'Dashboard' and features a table with the following data:

Activity	Status	Last Updated	Action
Asset Group	Available	2021-07-12 12:02:44	
Cyber Asset	Available	2021-07-12 12:04:13	
Software Asset	Available	2021-07-12 12:07:27	
WSUS	Available	2021-07-12 12:07:23	
Phalanx	Available	2021-07-12 04:08:06	
Applicable Vulnerabilities	Available	2021-09-13 10:06:03	

A.2 Organization data listing Applicable Vulnerabilities



The screenshot shows the 'Manage Cyber Assets' page in the web application. The left sidebar contains a menu with 'Organization Data' selected, and sub-items: Locations, Asset Groups, Cyber Assets, Software Assets, Cyber Asset Configurations, WSUS Reports, Vendor Patch Validation Reports, Vulnerabilities, Applicable Vulnerabilities, Manage Vulnerability Sources, and NVD Hardware Matching. The main content area has a 'Manage Cyber Assets' title and an 'Import Assets' button. Below the title is a filter section with a dropdown menu set to 'Cyber Asset' and a text input field 'Type to filter results'. The main table displays the following data:

Cyber Asset	Asset Group	Software Assets	Vulnerability Source	IP Address	Configuration	Manufacturer	Location
Primary-ControlCenter-Firewall	Firewall	0		10.1.1.1	ASA	Cisco Systems	Primary Control Center
Secondary-ControlCenter-Firewall	Firewall	0		10.1.1.5	ASA	Cisco Systems	Secondary Control Center
Secondary-JumpZone-Switch2	Switch	0		10.6.3.12	Catalyst 2900	Cisco Systems	Secondary Control Center
Secondary-ControlCenter-Switch1	Switch	0		10.6.3.7	Catalyst 2900	Cisco Systems	Secondary Control Center
Primary-ControlCenter-Switch2	Switch	0		10.6.3.2	Catalyst 2900	Cisco Systems	Primary Control Center
Secondary-ControlCenter-Switch2	Switch	0		10.6.3.8	Catalyst 2900	Cisco Systems	Secondary Control Center

Appendix B : Pipeline Overview

