NATIONAL VULNERABILITY DATABASE TREND ANALYSIS

Engineering Honors Thesis

by

JOHN M. SULLIVAN

Thayer School of Engineering

Dartmouth College

Hanover, New Hampshire

Date: ______________________

Approved: ______________________

George Cybenko, Ph.D., Advisor

____

John M. Sullivan, Author

Abstract

Computer security is a growing international concern. The National Vulnerability Database (NVD) alone contains 100,000 descriptions of these computer vulnerabilities, of which 6,000 have been added in just the last three months (as of April 2018). Clearly, these vulnerabilities are a growing issue. The vulnerability reports in this valuable database consist of many data features, including scores and textual descriptions. This thesis contributes to the understanding and value of the NVD, with the goal of improving computer security. First, the presence of trends in this data was researched. Topic modeling revealed trends in the vulnerability description data. Second, the predictive power of the vulnerability features to determine whether a vulnerability had been exploited was tested. It was found that these features have significant predictive power. Combining these results, a model is presented that uses topics in the NVD to classify if a vulnerability has been exploited. At its peak performance, this model predicts whether a vulnerability has been exploited 85-90% of the time and can be adjusted to predict only true positives with 50-60% recall. It is possible that the methods of analysis set forth in this thesis can be used to prioritize the remediation of vulnerabilities before they are exploited and improve cybersecurity in general.

Preface

I would like to thank Professor Cybenko for his help as my Honors Thesis advisor and for being so generous with his time. I would like to thank Kate Farris, who provided me with advice and guidance during her time as a Ph.D. candidate at Thayer. I would also like to thank my parents for their consistent support throughout my education.

During my research, I was able to learn many new and important concepts. Before starting this Honors Thesis, I had not applied many of the statistical methods that I had learned. Throughout the course of my research, however, I was able to apply many of these new statistical methods to real data. In addition, I did not have much experience with machine learning techniques. My research taught me many modern machine learning techniques and how to apply them as well.

In the course of preparing of my Honors Thesis, I met consistently with Professor Cybenko. I am grateful to have had the opportunity to conduct independent research with such an esteemed advisor. Working independently on the research, I encountered many challenges and benefitted greatly from the direction that he provided. I am looking forward to applying what I have learned researching the NVD in a constructive way. Already, I used this research extensively for my economics culminating course study entitled, “Effects of Vulnerability Publications on Stock Returns” [1].

Contents

Abstract ii

Preface iii

1 Introduction **2**

1.1 Definitions 3

1.2 Familiarity with Databases 4

1.2.1 National Vulnerability Database 4

1.2.2 Exploit Database 5

1.3 Existing Research 6

1.3.1 Topic Modeling 6

1.3.2 Machine Learning Research 7

1.4 Objectives 7

2 Background **9**

2.1 Vulnerability Trends 9

2.2 Trend Correlation with Updates 12

2.3 Trend Analysis 14

2.4 Background Results Summary 15

3 Topic Modeling **16**

3.1 Word Frequency Modeling 17

3.2 Topic Modeling 19

3.3 Topic Modeling Results Summary 23

4 Machine Learning Models **25**

4.1 Data Preparation 25

4.2 Distribution of Time to Exploit 26

4.3 Base Rate Fallacy Issue 27

4.4 Models 28

4.4.1 Decision Trees 29

4.4.2 Bagged Trees 30

4.4.3 Neural Network 30

4.4.4 Support Vector Machine Models 31

4.4.5 Models Results and Discussion 32

4.5 Looking at Predictive Power Over Time 33

4.6 Return to Topic Modeling 34

4.7 Machine Learning Results Summary 38

5 Conclusion **40**

5.1 Future Directions 42

Bibliography **45**

Appendices **46**

A Sample Figures **47**

A.1 NVD 47

A.1.1 Example Entry 47

A.1.2 Same Entry in Machine Learning Data Table 48

A.2 Exploit DB 49

A.2.1 Example Entry 49

A.2.2 Same Entry in Exploit DB Data Table 49

A.2.3 Same Entry in Exploit DB / NVD Date Comb Table 50

B Programs **51**

B.1 XQuery Code 51

B.2 Data Processing Code 55

B.3 Background Code 62

B.4 Mallet Topic Modeling Code 64

B.5 Histogram Code 67

B.6 Models Code 69

B.7 Topic Modeling Prediction Code 72

List of Tables

1.1 NVD Overall Statistics (as of 12/10/17) 5
1.2 Exploit DB Overall Statistics (as of 12/11/17) 5

2.1 Summary Statistics of Vulnerability Severity Distribution 15
2.2 Summary Statistics of Vulnerability Severity by Operating System . . 15

4.1 NVD / Exploit DB Timeline Summary Statistics 26
4.2 Sample Statistics and Percent Exploited 27
4.3 Machine Learning Model Results on Training Set (59,121 samples) . . 33
4.4 Machine Learning Model Results on Test Set (10,433 samples) 33
4.5 Machine Learning Model Results Applied to 2017 Data (13,336 samples) 33
4.6 Words with highest probability of being associated with an exploit . . 35
4.7 Topic Modeling Approach Results (12 topics) 37

List of Figures

2.1 Distinct relationship in vulnerability trends: Microsoft-Linux over time (2015) 10

2.2 Regression of vulnerability frequencies for Microsoft on those for Linux 11

2.3 Unclear relationship in vulnerability trends: Apple-Linux over time (2015) 11

2.4 Dotted lines show time of Apple updates 12

2.5 Regression of Apple monthly vulnerabilities on updates 13

2.6 Dotted lines indicate vulnerability peaks for Apple without updates . 13

2.7 Vulnerability Severity Distributions 14

3.1 An example of a vulnerability description from the NVD 16

3.2 Effect of normalization on word frequency data 17

3.3 Examples of phrases in word frequency data 18

3.4 Examples of different trends in word frequency data 18

3.5 Microsoft Topic Model (words indicative of possible topic meaning bolded) 20

3.6 Microsoft Yearly Topic Model 21

3.7 Revised Microsoft Quarterly (3-month interval) Topic Model 22

3.8 Microsoft Quarterly (3-month interval) Topic Model with Smoothing 23

4.1 Histogram of Exploit and Vulnerability Timeline 26
4.2 Decision Tree Illustration [2] 29
4.3 Random Forest Illustration [3] 30
4.4 Neural Network Illustration [4] 31
4.5 Linear SVM Illustration [5] 32
4.6 High Exploit Probability Topic Cloud 36
4.7 Topic Model Bar Chart Threshold 36

A.1 NVD Website Entry 47
A.2 NVD Datasheet Entry 48
A.3 Exploit DB Website Entry 49
A.4 Exploit DB Datasheet Entry 49
A.5 NVD-EDB Cross Listing 50

Chapter 1

Introduction

Computer security continues to be a major concern. There are frequently stories in the news of computers being exploited to obtain data or threaten the security of some organization. To compromise a computer, an attacker must find and exploit a vulnerability in its software or operating environment. The U.S. National Institute of Standards and Technology (NIST) National Vulnerability Database (NVD) alone contains 100,000 of these known vulnerabilities, of which 6,000 have been added in just the last three months (as of April 2018) [6]. Clearly these vulnerabilities are a growing issue and the NVD contains an abundance of information that can be analyzed to improve cybersecurity.

This Honors Thesis is the product of three terms of research with the assistance of Professor George Cybenko. Kate Farris, Ph.D., used findings and data from the NVD obtained during this research. Results of this research were included in the paper entitled “Vulnerability survival analysis: A novel approach to vulnerability management” by Kate Farris, John Sullivan, and George Cybenko [7].

The first two terms of this research were part of the James O. Freedman Presidential Scholars program. They were spent analyzing computer vulnerabilities. In the first term, it became clear that there were underlying trends in the vulnerability data obtained from the NVD. In the second term, the Undergraduate Investigations course was completed in preparation for this Honors Thesis. During this term, textual analysis of the NVD vulnerability descriptions through topic modeling revealed distinct changes in the frequency of certain words and groups of words over time.

A focus of this Honors Thesis is on the textual analysis of vulnerability descriptions. Machine learning techniques were used to investigate more complex patterns in the vulnerability descriptions. Once these textual elements were better understood, connections were made between this new information and general analysis of the vulnerability trends. Finally, a model was created that connects topic modeling with whether a vulnerability has been exploited. Similar to the doctoral research done by Kate Farris, the objective of this research is to improve how vulnerabilities are understood in order to help with their remediation and to improve computer security generally.

1.1 Definitions

The following are some definitions that are important for understanding the terminology used throughout this thesis:

1. **Vulnerability** - A weakness in a computer system's security that could be exploited. In computer security, this weakness may be a segment of code.

2. **Exploit** - Commands that allow a user to take advantage of a vulnerability and cause a computer to behave in an unintended way.

3. **Remediate** - The act of addressing a computer vulnerability so that it no longer presents a weakness to the system.

4. **Patch** - A piece of software that can be used to update a computer program or system, possibly to address a vulnerability.

5. **Zero-day Exploit** - An exploit that takes advantage of a computer vulnerability that has not yet been discovered.

1.2 Familiarity with Databases

During the course of this research, two databases were used: the National Vulnerability Database (NVD) and the Exploit Database (EDB) [8].

1.2.1 National Vulnerability Database

The NVD is the U.S. government repository for vulnerability management data [6]. This database facilitates vulnerability management, security measurement, and compliance as an an automated process. This information can be analyzed to improve computer security.

The downloadable version of the NVD is organized into XML files by year. For this research, XQuery was used to restructure the XML files to extract the relevant data. Matlab was used to perform statistical analysis on the database. Stata was used for some simple statistical operations.

Table 1.1 shows some overall statistics for the NVD.

Table 1.1: NVD Overall Statistics (as of 12/10/17)

	No. Vulnerabilities
Total	98,183
Total (excluding missing entries)	93,295
Last 3 Months	4,037

1.2.2 Exploit Database

The Exploit Database is the most comprehensive collection of exploits consolidated from a variety of sources [8]. It is maintained by Offensive Security, an information security training company, and is CVE (Common Vulnerabilities and Exposures) compliant [8].

With regard to the Exploit Database, this research focused on exploits that had been mapped to the NVD (in other words, vulnerabilities in the NVD that had been exploited). Only about $\frac{1}{3}^{\text{rd}}$ of the exploits from the EDB are mapped to the NVD.

There are two possible reasons for the discrepancy between the number of exploits and vulnerabilities. First, there are likely some exploited vulnerabilities that have not been linked between the two databases. Second, there may be some exploits that have not been reported to the EDB but would be associated with vulnerabilities in the NVD.

Table 1.2 shows some overall statistics for the Exploit Database.

Table 1.2: Exploit DB Overall Statistics (as of 12/11/17)

	No. Exploits
Total	38,236
Mapped to NVD	9,267

1.3 Existing Research

Existing research that used the NVD was explored. Much of the existing research looked only at the number of vulnerabilities for a company or product at a set time [9] [10]. These findings were then used to present the relative number of vulnerabilities for the companies or products. This research did not make use of the fact that the NVD offers many other statistics on vulnerabilities, such as their severity levels. Some of the existing research, however, looked at how these severity levels have changed over time [11] [12].

Some existing field research that analyzed data from the NVD raised the possibility of its power to predict future vulnerabilities. Other research pointed out inherent limitations with the database because of its large scale and the general nature of its entries [13] [14]. The studies also showed the different types of analytics that focused on vulnerabilities in general, or on commonly used applications and operating systems. In the existing research, there was a large variety of tests that had been performed on data from the NVD.

1.3.1 Topic Modeling

One article used topic modeling, a novel technique, on data from the NVD. Topic modeling is an unsupervised machine learning technique that identifies topics in textual documents and how they change over time. In an article by Stephan Neuhaus and Thomas Zimmermann entitled “Security Trend Analysis with CVE Topic Models,” the authors found the presence of multiple topics with different trends over time in the NVD [15]. Their topic modeling approach makes use of Latent Dirichlet Allocation (LDA).

This topic modeling approach is mostly automated, except that the researcher has to classify the topics that are generated. Inspired by the results from Neuhaus's study, this thesis explores how topic modeling can be used to predict vulnerability features in the NVD.

1.3.2 Machine Learning Research

Other existing research used machine learning techniques in an attempt to predict whether a vulnerability had been exploited. After textual analysis of the NVD, the next step was to research the predictive power of textual descriptions from the NVD to determine whether a vulnerability had been exploited. An existing technical report explored this area. The paper by Recorded Future, a cyber threat intelligence provider headquartered in Boston, is titled "Anticipating Cyber Vulnerability Exploits Using Machine Learning" [16].

RecordedFuture's technical report was based on a Master's Thesis by Michel Edkrantz at Chalmers University of Technology in Sweden [17]. Edkrantz tested several machine learning models and their power to predict whether a vulnerability had been exploited. The best model from his study was able to predict whether a vulnerability had been exploited 83% of the time using just descriptions from the NVD. As part of this thesis, the results of Edkrantz's study are examined and expanded upon.

1.4 Objectives

The findings in this Honors Thesis are presented in the context of existing research. This thesis is divided into the following three sections:

1. **Background** - The relationship between the trends in vulnerabilities for many

commercial organizations and their various products is examined, rather than looking at such organizations on their own.

2. **Topic Modeling** - Correlations between trends in different vulnerabilities are explored through analysis of NVD descriptions and topic modeling.

3. **Machine Learning Models** - Machine learning models and techniques are used to analyze the power of textual descriptions to predict whether a vulnerability has been exploited.

Chapter 2

Background

Background research was conducted in order to better understand the NVD. Much of the existing research of the NVD (see section 1.3) involved looking at specific companies, not at the relationship of vulnerability trends between companies. This thesis explores relationships in vulnerability trends between companies and other factors that affect vulnerability occurrences.

In this chapter, vulnerability frequency correlations are found between companies and based on whether an update was just released (i.e. for Apple, which has accessible data on updates). The distribution of vulnerabilities by severity level is found to be consistent over time, while the distribution by operating system varies.

2.1 Vulnerability Trends

As stated in the first research objective in section 1.4, one goal of this thesis is to study trends between vulnerabilities. In this section, vulnerability relationships between different companies are examined. First, the number of vulnerabilities was normalized for different companies. This was achieved by dividing the frequency of vulnerabilities in a given month for a company by the maximum number of vulnerabilities within a time range. This method scaled the vulnerabilities for different companies so that they could be compared on the same graph, as in Figure 2.1.

For some companies, such as Microsoft and Linux, normalizing the number of vulnerabilities made clear similarities in the peaks in vulnerability frequencies. These similarities are evident by directly observing the graph of vulnerability frequencies in Figure 2.1.

In order to see the relationship numerically, the frequencies of vulnerabilities for the two companies (Microsoft and Linux) were regressed on each other for the same year (2015). The relationship was found to be significant, as indicated by a p-value of less than 10% in Figure 2.2. Despite this statistical relationship, its underlying cause is still unclear.

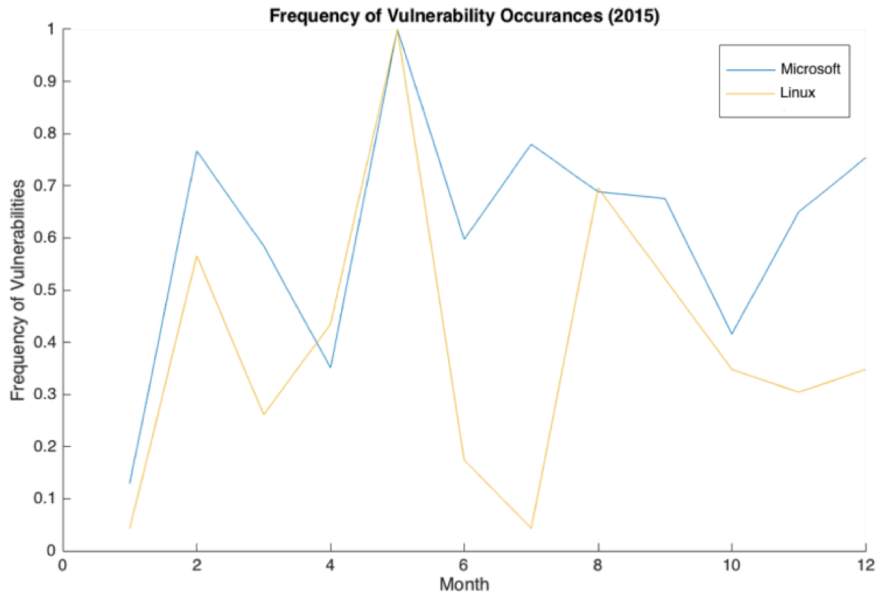


Figure 2.1: Distinct relationship in vulnerability trends: Microsoft-Linux over time (2015)

```
. reg Microsoft Linux

      Source |       SS           df       MS      Number of obs   =        11
-------------+----------------------------------   F(1, 9)         =      4.41
       Model |  .187444908         1  .187444908   Prob > F        =    0.0650
    Residual |  .382297014         9  .042477446   R-squared       =    0.3290
-------------+----------------------------------   Adj R-squared   =    0.2544
       Total |  .569741922        10  .056974192   Root MSE        =     .2061

------------------------------------------------------------------------------
   Microsoft |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       Linux |   .4804123   .2286949     2.10   0.065    -.0369315    .9977562
       _cons |   .4261999   .1074692     3.97   0.003     .1830877    .6693121
------------------------------------------------------------------------------
```

Figure 2.2: Regression of vulnerability frequencies for Microsoft on those for Linux

When replacing Microsoft with Apple, the relationship between the vulnerability frequencies of Apple and Linux was unclear. The graph in Figure 2.3 shows no distinct relationship between the vulnerability frequencies for these two companies.

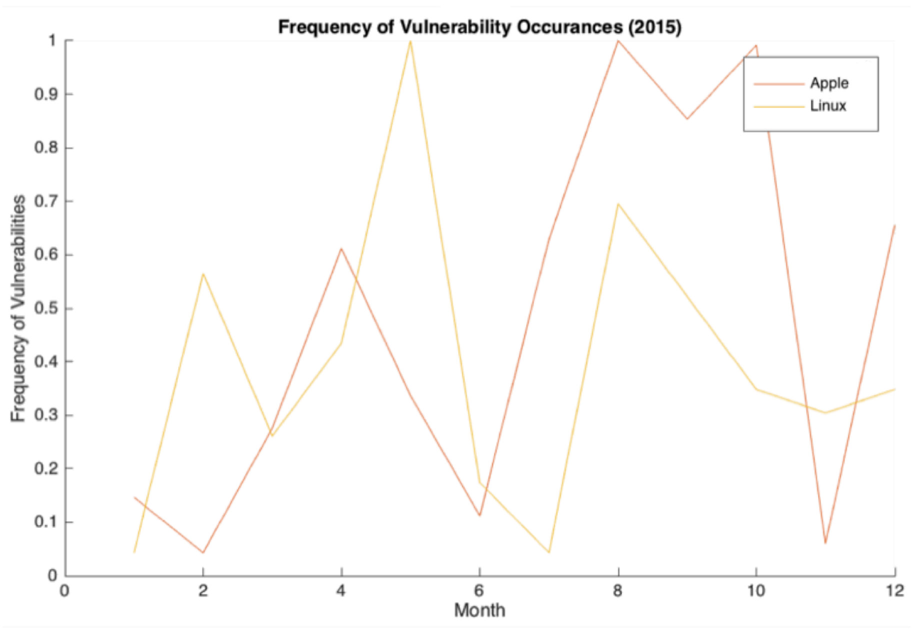


Figure 2.3: Unclear relationship in vulnerability trends: Apple-Linux over time (2015)

2.2 Trend Correlation with Updates

Apple vulnerabilities were examined because Apple has data readily available on update release dates. Figure 2.4 shows dotted lines when updates were released between 2010 and 2012. The peaks that correspond to updates are prominent.

It is evident that each of these updates corresponds to a month with a greater amount of vulnerabilities. This relationship illustrates that a software update by Apple may result in Apple experiencing more security issues than usual. For consumers, it suggests that they should update their device soon after Apple releases software updates to address these vulnerabilities.

In addition, for statistical understanding, the frequency of vulnerabilities in Apple products was regressed on their software updates and revealed a strong relationship, as indicated by a p-value of almost 0 in Figure 2.5. This finding also indicates that

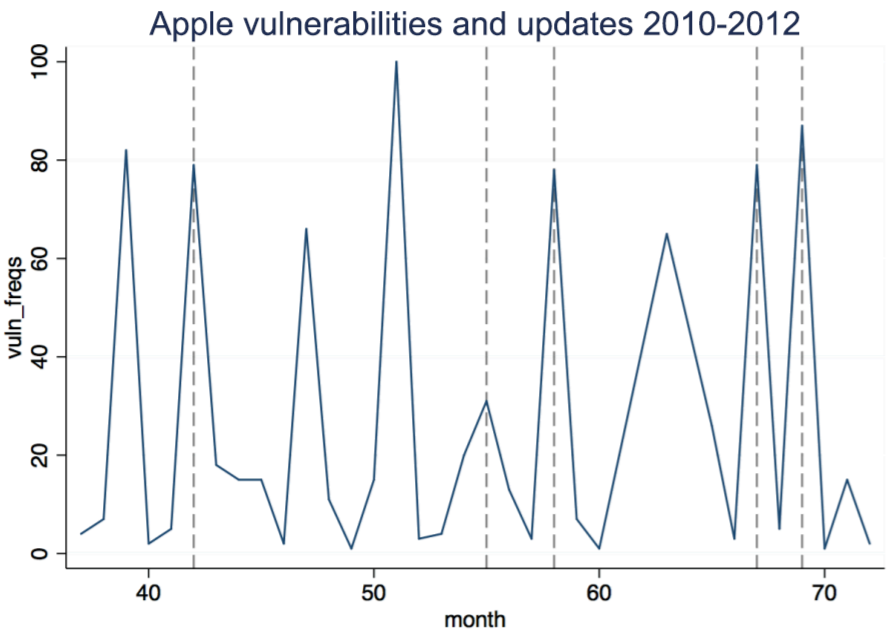


Figure 2.4: Dotted lines show time of Apple updates

```
. reg vuln_freqs i.update

      Source |       SS           df       MS      Number of obs   =       102
-------------+----------------------------------   F(1, 100)       =     14.39
       Model |  9565.05991         1  9565.05991   Prob > F        =    0.0003
    Residual |  66448.3126       100  664.483126   R-squared       =    0.1258
-------------+----------------------------------   Adj R-squared   =    0.1171
       Total |  76013.3725       101  752.607649   Root MSE        =    25.778

------------------------------------------------------------------------------
  vuln_freqs |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
    1.update |   27.34253   7.206706     3.79   0.000     13.04463    41.64043
       _cons |   22.72414   2.763646     8.22   0.000     17.24114    28.20713
------------------------------------------------------------------------------
```

Figure 2.5: Regression of Apple monthly vulnerabilities on updates

Apple's software updates are a strong predictor of when Apple will have a higher number of vulnerabilities.

However, there were some instances when the peaks in vulnerabilities for Apple

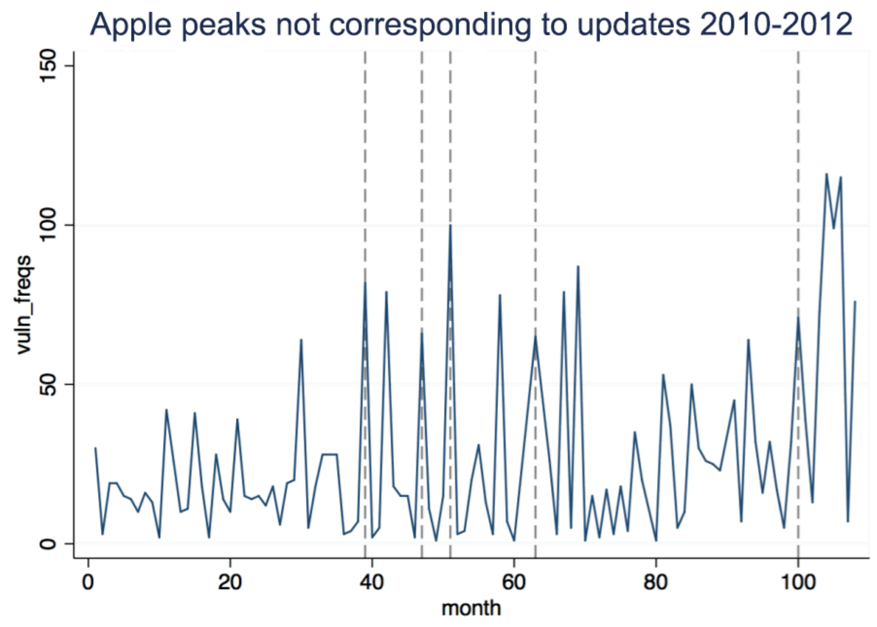


Figure 2.6: Dotted lines indicate vulnerability peaks for Apple without updates

products were not related to when they released software updates (shown in Figure 2.6). After further analysis, it seemed that the peaks that did not correspond to software updates for Apple were related in almost every instance to when the company released a new product, such as the iPad or Apple Watch.

2.3 Trend Analysis

The following analysis of general trends in the NVD was done as part of the paper "Vulnerability survival analysis: A novel approach to vulnerability management" by Kate Farris, John Sullivan, and George Cybenko [7].

In order to understand the NVD from an overall perspective, the authors studied the distribution of vulnerabilities by severity level and affected operating systems. Figure 2.7 shows the distributions of these trends.

The summary statistics of the data represented in these graphs were examined. The graph on the left in Figure 2.7 shows that the distribution of vulnerability severity is relatively consistent over time. This observation is supported by Table 2.1, which shows that the standard deviations for vulnerability severity are less than 0.1.

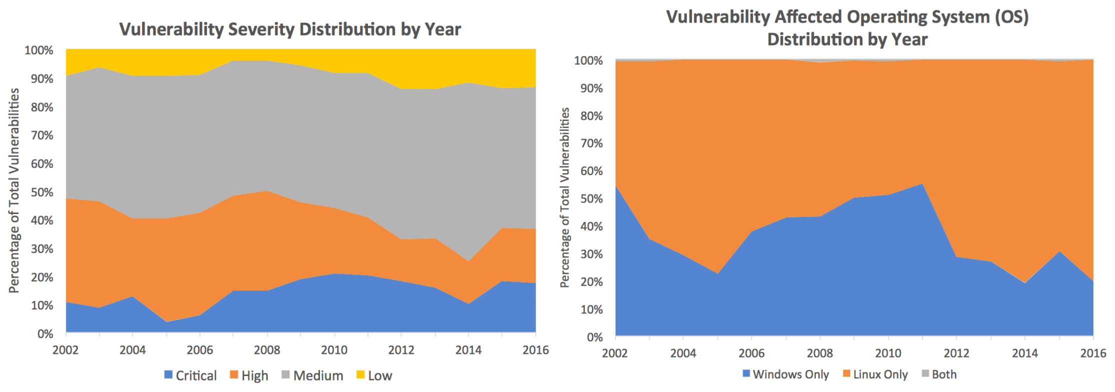


Figure 2.7: Vulnerability Severity Distributions

Table 2.1: Summary Statistics of Vulnerability Severity Distribution

Variable	**Obs.**	**Mean**	**Std. Dev**	**Min**	**Max**
Critical	15	0.1410	0.0523	0.0386	0.2098
High	15	0.2659	0.0867	0.1479	0.3749
Medium	15	0.4982	0.0443	0.4329	0.6302
Low	15	0.0949	0.0344	0.0395	0.1413

Table 2.2: Summary Statistics of Vulnerability Severity by Operating System

Variable	**Obs.**	**Mean**	**Std. Dev**	**Min**	**Max**
Windows Only	15	0.3643	0.1234	0.1925	0.5494
Linux Only	15	0.6323	0.1257	0.4472	0.8048
Both	15	0.0034	0.0038	0.0000	0.0103

For operating systems, the graph on the right in Figure 2.7 shows that the distribution of vulnerabilities by operating systems has varied. This observation is supported by Table 2.2, which shows that vulnerabilities for Windows and Mac have standard deviations greater than 0.1 (more than the standard deviation for the severity levels).

2.4 Background Results Summary

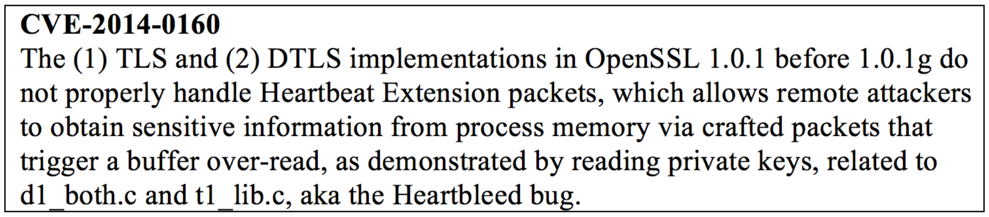
Focusing on a few companies reveals vulnerability frequency trends between some of them and not others. For Apple specifically, there is a correlation between the number of software vulnerabilities and when their software updates are released. The distribution of vulnerabilities by severity level is relatively consistent. However, the distribution of operating systems that are affected by specific vulnerabilities has varied.

Chapter 3

Topic Modeling

In order to further understand the underlying forces behind vulnerabilities for different companies, description data from the NVD was analyzed. The descriptions provided in the NVD have the potential to add insight because of their detail. Each vulnerability description typically offers a paragraph of text with information pertinent to that vulnerability. If this language could be properly analyzed, then it would be possible to better understand the nature of vulnerabilities. Further, this language could be used to determine trends in vulnerability types.

In this chapter, the frequency of certain words and phrases are studied to see if they exhibit trends in the NVD. By employing a topic modeling approach, it is possible to form relevant topics and observe how they evolve.



CVE-2014-0160
The (1) TLS and (2) DTLS implementations in OpenSSL 1.0.1 before 1.0.1g do not properly handle Heartbeat Extension packets, which allows remote attackers to obtain sensitive information from process memory via crafted packets that trigger a buffer over-read, as demonstrated by reading private keys, related to d1_both.c and t1_lib.c, aka the Heartbleed bug.

Figure 3.1: An example of a vulnerability description from the NVD

3.1 Word Frequency Modeling

First, the frequency at which words appeared in the vulnerability descriptions over time was studied. This approach is similar to how search analytics are done to determine how many times words appear in documents online at different points in time.

The first issue encountered in this process was that the frequency of the words in the NVD descriptions had not yet been normalized. To normalize the frequency of the words, the number of times that a word appeared within vulnerability descriptions had to be divided by the total number of vulnerabilities in that year. This calculation produced a weighted measure of the frequency of a word for a given year.

The effect of this change was noteworthy, as seen in the graphs in Figure 3.2. Initially, no distinct trend is present. However, when the data was normalized in this way, a distinct downward trend became apparent for the occurrence of the words "root" and "password."

This analysis was also conducted for phrases of text (bigrams as opposed to unigrams). The graphs in Figure 3.3 show the changing frequency of occurrences for the

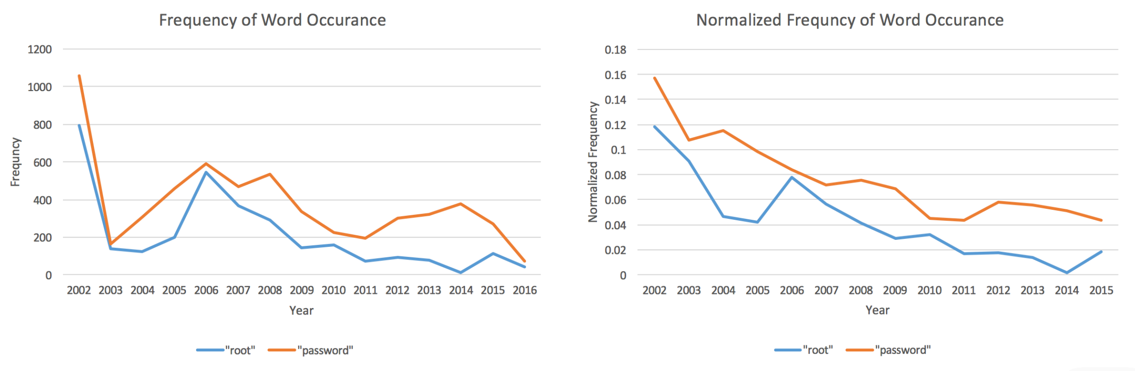
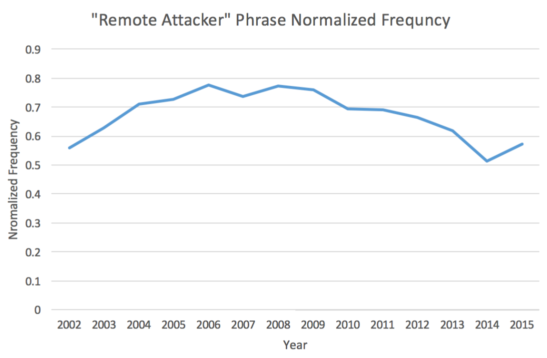


Figure 3.2: Effect of normalization on word frequency data



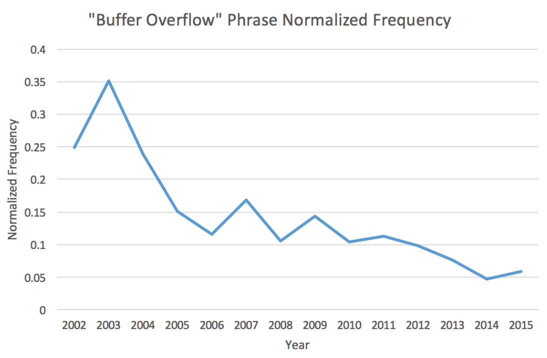
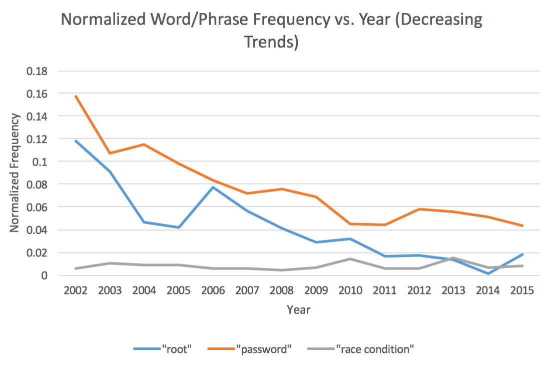


Figure 3.3: Examples of phrases in word frequency data

phrases "Remote Attacker" and "Buffer Overflow" over time.

This analysis was conducted for some other words and phrases and, as shown in Figure 3.4, they vary in their prevalence. For example, the occurrence of the phrase "denial of service" in vulnerability descriptions generally increases from 2002. This pattern is indicative that this type of vulnerability has become more prominent. This analysis is not possible using just the simple indicators provided by the NVD.



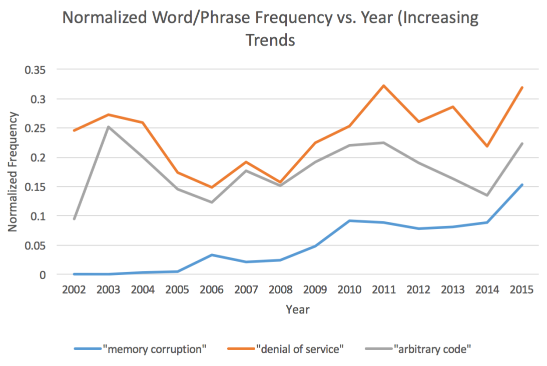


Figure 3.4: Examples of different trends in word frequency data

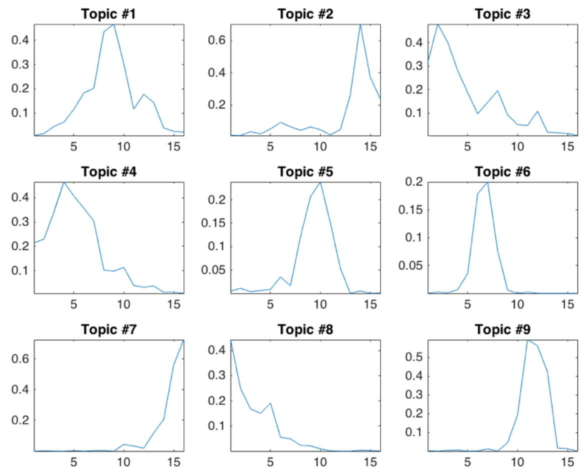
3.2 Topic Modeling

The results of the analysis obtained by word and phrase frequency modeling were instructive. This analysis was extended by using a topic modeling method recommended by Professor Cybenko. A computer program called Mallet was used to generate topics present in the descriptions and their frequencies over time. This program uses Latent Dirichlet Allocation to generate topics.

A topic is a collection of words that occur in a group at a specific time, which may be considered significant. For example, in common speech, general word usage changes with time. In the NVD, the frequency of the appearance of specific topics also show trends. The difficult part of this analysis was determining the meaning and significance of topics and optimizing how they were generated.

It is important to note that the normalized frequencies of each topic will add up to one. That means that these topics are being considered in relation to each other when forming the topic models. Further, a setting was used in Mallet to assign different weights to each topic. This setting caused the program to generate more meaningful topics. These various weights are listed alongside the word listing for each topic in Figure 3.5.

By observing the first topic model in Figure 3.5, it is clear that the different topics exhibit different trends. It is also evident that the topics appear and disappear with time. Within the topics generated by the program, some words are bolded that may be significant and indicative of the meaning behind the topic. In the first topic, the words "server," "crafted," and "remote" all appear and are bolded. This combination of words may be indicative of crafted code being used on servers. The graph of this topic in Figure 3.5 may be interpreted as showing when such attacks were prevalent.



Microsoft Topics with weight given each topic:

1. 0.80832 sp microsoft vulnerability **server** windows code office **crafted remote** execute
2. 0.66163 cve explorer internet **memory corruption** vulnerability aka attackers microsoft service
3. 0.75498 remote attackers microsoft internet execute **arbitrary** explorer **files file web**
4. 0.80438 windows attackers microsoft remote **arbitrary code** xp **buffer** execute internet
5. 0.30644 aka excel **gold** properly office **mac powerpoint corruption** unspecified converter
6. 0.13328 user assisted unspecified note issue **activex** crash vectors information related
7. 0.23908 windows sp vulnerability microsoft **server gold** aka crafted **remote office**
8. 0.31534 server service vulnerability windows **denial** user users properly nt earlier
9. 0.27023 sp windows vulnerability aka **server crafted** microsoft attackers **remote** win

Figure 3.5: Microsoft Topic Model (words indicative of possible topic meaning bolded)

If one looks at the trend for Topic 7, it is clearly peaking at the latest time [year 16 (2016)]. If it were to follow trends similar to those that some of the other topics exhibited, which was decreasing after a peak, one might expect the vulnerability issues presented in this topic to decrease in year 17 (2017) and later. This finding for Topic 7 may indicate that, although one should be currently concerned with these types of vulnerabilities, they might be remediated soon and not be seen as often in the future.

Some issues are readily apparent. For example, some of the topics contain common words that appear as part of the database descriptions. They also contain too many words. Thus, considering the topics on a yearly basis may not provide accurate enough results, as indicated by the rough and jagged nature of the lines in some of the graphs in Figure 3.5.

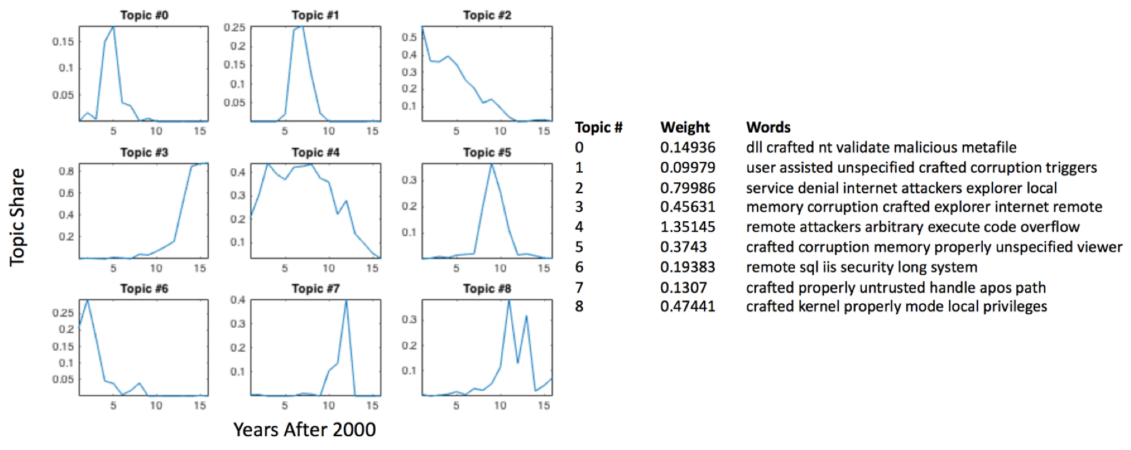
The following modifications were made to the topic modeling approach to address the issues described previously and to produce the new graphs below:

1. Revised stop words to exclude some common, unnecessary words that were part of the previous topics.

2. Adjusted certain topic modeling parameters, such as setting only 6 words per topic. The previous topics contained too many words.

3. Adjusted topic modeling to 3-month intervals (as shown in the second of the two graphs below). This approach might produce more accurate models.

These revisions appeared to lead to more meaningful results. The new topics in Figure 3.6 are much more concise and understandable. In addition, the topics exhibit trends similar to those shown previously. The difficult part is extracting meaning from the words for a given topic.



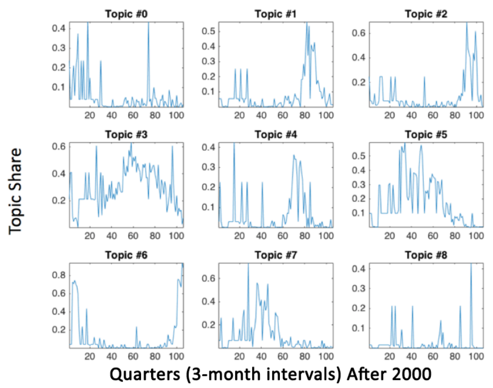
Topic #	Weight	Words
0	0.14936	dll crafted nt validate malicious metafile
1	0.09979	user assisted unspecified crafted corruption triggers
2	0.79986	service denial internet attackers explorer local
3	0.45631	memory corruption crafted explorer internet remote
4	1.35145	remote attackers arbitrary execute code overflow
5	0.3743	crafted corruption memory properly unspecified viewer
6	0.19383	remote sql iis security long system
7	0.1307	crafted properly untrusted handle apos path
8	0.47441	crafted kernel properly mode local privileges

Figure 3.6: Microsoft Yearly Topic Model

As previously stated, another modification was to use quarterly data. Unfortunately, some of the results produced by this model were harder to interpret as seen in Figure 3.7. The trends shown by these graphs are not as smooth in appearance as those using the yearly model. This difference shows that, on a quarterly basis, the data exhibits a lot of noise. The words generated within each topic are similar to those from the yearly data. For these reasons, the yearly data appears slightly more useful for analysis than the quarterly data.

In an attempt to account for the noise that was shown in the quarterly data, a smoothing function was applied. The results are shown in Figure 3.8. It is noteworthy that this change produced graphical results more consistent with the yearly data. There is more noise present, but it is easier to determine the general trend in the quarterly data with the smoothing.

From some of the existing research referred to in section 1.3, the article by Neuhaus generated and used topics similar to the ones that were generated here [15]. The topics that were categorized in Neuhaus's article resembled the names of certain security flaws. For example, some topics identified as present in the NVD in his



Topic #	Weight	Words
0	0.19525	bypass unspecified vectors edge html execution
1	0.2774	crafted memory corruption code properly execute
2	0.24845	crafted kernel local users properly privileges
3	1.07626	remote attackers arbitrary execute code explorer
4	0.1494	user crafted assisted unspecified dll code
5	0.504	service denial internet local attackers users
6	0.19945	memory corruption explorer internet crafted attackers
7	0.32617	sql iis nt commands attacker user
8	0.07859	antivirus anti virus command net firewall

Figure 3.7: Revised Microsoft Quarterly (3-month interval) Topic Model

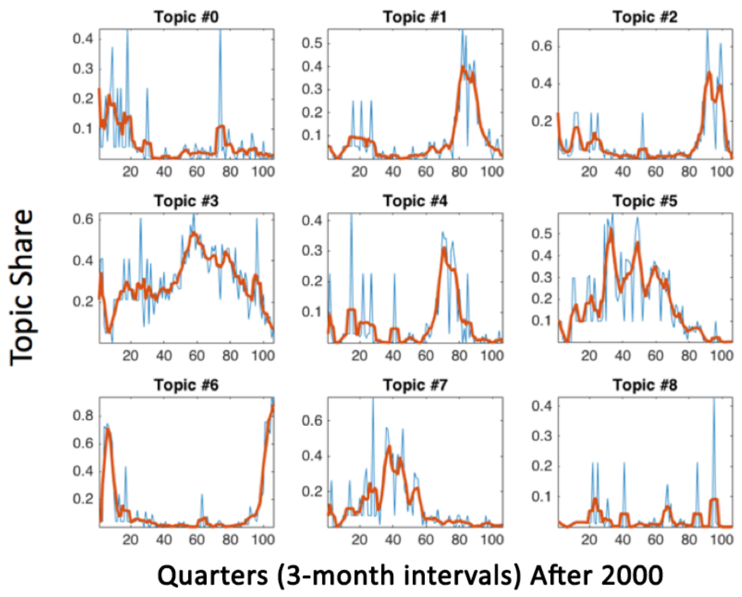


Figure 3.8: Microsoft Quarterly (3-month interval) Topic Model with Smoothing

article included: “Arbitrary Code,” “Buffer Overflow,” and “SQL Injection.”

From the analysis that had been done on topics in the NVD and the article by Neuhaus, it might be also be possible to form predictions on whether a vulnerability could be exploited based on which topic it belongs to. In order to achieve the necessary level of detail for each topic based on the variety of exploits, it was necessary to use more than nine topics. This approach is explored in section 4.6. Without knowing the specific topic meaning, it may be possible to group vulnerabilities that are similar.

3.3 Topic Modeling Results Summary

The frequency with which certain words and phrases occurred was normalized and revealed trends in their appearances over time. Topics for the vulnerability descriptions were formed using a program called Mallet, which implements Latent Dirichlet Allocation. The topic models and results were improved by revising the stop words, the words per topic, and the time interval for which the topics were generated. These adjustments led to more meaningful topics and results, which are explored further in chapter 4.

Chapter 4

Machine Learning Models

After observing the presence of trends in the NVD vulnerability descriptions, their predictive power was explored. At the beginning, machine-learning analysis was used. Then, just as Edkrantz found in his Master's Thesis, it became apparent that these descriptions could have a high power to predict whether a vulnerability had been exploited [17]. For this research, a similar dataset was created and machine-learning analysis was applied.

In this chapter, machine-learning models are used to predict whether a vulnerability was exploited based on its features. Further, an approach is explored using topic modeling to determine whether a vulnerability is at a high-risk of being exploited.

4.1 Data Preparation

The data from the NVD used in this section is complete up to December 10, 2017. The data from the Exploit Database is complete up to December 11, 2017. Statistics for this data are shown in section 1.2.

4.2 Distribution of Time to Exploit

The purpose of the work described in this chapter was to assess the predictive power of the vulnerability characteristics to determine whether a vulnerability had been exploited. To first understand the relationship between vulnerabilities and exploits, the timeframe between when these two events occur was examined.

It would seem logical that a vulnerability would be discovered before a related exploit. However, the data in Table 4.1 shows that, on average, an exploit is reported 3 months prior to an associated vulnerability being reported, with a standard deviation of 1 year. A histogram for this distribution is included in Figure 4.1. As Table 4.1 illustrates, the time between an exploit and its associated vulnerability is normally

Table 4.1: NVD / Exploit DB Timeline Summary Statistics

Variable	Mean	Std. Dev.	Min.	Max.	N
Month_Difference	-3.026	14.074	-145	167	9268

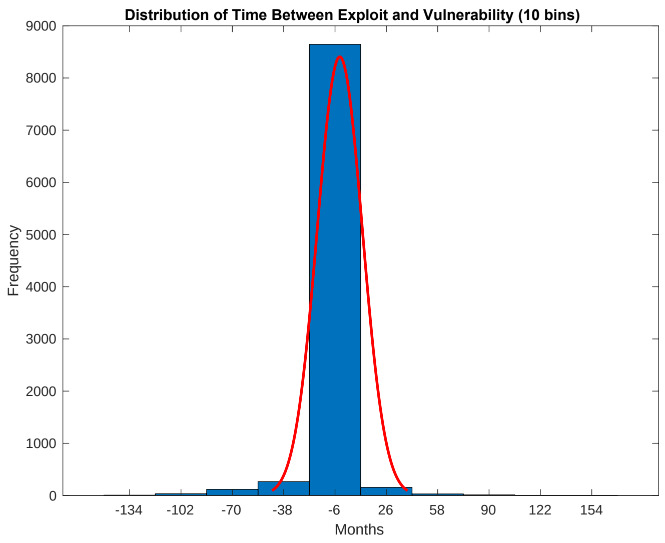


Figure 4.1: Histogram of Exploit and Vulnerability Timeline

distributed with a heavy concentration of the data towards the center.

The realization that an exploit might be created before the associated vulnerability appears in the NVD is representative of zero-day vulnerabilities. Zero-day vulnerabilities are vulnerabilities that are unknown to those who may be interested in fixing them. With these types of vulnerabilities, it is possible for an exploit to occur before it is reported in the NVD.

4.3 Base Rate Fallacy Issue

A major issue encountered with the analysis of this dataset was the problem of having a small number of positive samples (meaning vulnerabilities that have been exploited). In general, the number of exploited vulnerabilities was around 10% of the sample size (see Table 4.2).

Table 4.2: Sample Statistics and Percent Exploited

	Total Samples	**Exploited Samples**	**Percent**
Total	93,295	10,015	10.73%
2005-2016	69,554	9,037	12.99%

The original models created for this data predicted that nearly all of the vulnerabilities had been exploited. The computer-generated models made these predictions because, by misclassifying many exploited vulnerabilities, they could be correct about 90% of the time.

First, it was necessary to use some better measures of model accuracy, namely precision and recall, for dealing with data with a characteristically low base rate:

- **Precision** - For all positive predictions, the fraction that was actually true.
- **Recall** - For all true samples, the fraction that was predicted as being true.

Second, to ensure that the exploited samples had a greater effect on the data, the following solution was developed:

Increasing Cost for Exploited Samples. To adjust for the small number of positive (exploited) samples, a greater cost was applied to these samples when creating the models. By applying an increased cost to these samples, it was possible to ensure that they were well represented in the final model. Trial and error was used to find cost values that would balance the representation of these samples with maintaining good metrics in accuracy, precision, and recall.

4.4 Models

The NVD contains entries since 1988. However, only NVD data between the years 2005 and 2016 was used for the following reasons:

1. As new data continues to be released and technology continues to evolve, older vulnerabilities (i.e. prior to 2005) have less of a connection with newer ones.

2. As more years were added, it became more difficult computationally to run the models. Using only data since 2005 presented a good tradeoff between adding more years and increasing how long it took to run the models.

3. The original study by Edkrantz used data since 2005 and this eleven-year time-frame seemed comprehensive [17]

Data from 2017 was excluded when generating these models because the data for that year was not yet complete. However, the incomplete data for 2017 was used to test these models. The model performance based on 2017 data is discussed in section 4.5.

A Bag-of-Words model was created to represent the products, vendors, and descriptions. When combining this model with the scores, the dataset contained 1,525 features. One response variable was designated - whether the vulnerability had a corresponding entry in the Exploit Database.

In total, 70% of the data was used for training (59,121 samples). Another 15% was held out during training for validation when possible (10,433 samples). The final 15% was used as test data to see how the model performed on data that had not been incorporated (10,433 samples). The data for 2017 contained 13,336 samples.

The models described in the following sections were the most optimal for classifying the data.

4.4.1 Decision Trees

Individual trees were developed to make decisions based on the values of various features of the data. The Decision Tree model process is illustrated in Figure 4.2. The maximum number of splits in the tree may be specified. In this study, 100 maximum splits were used because this data consisted of a high level of granularity. Multiple

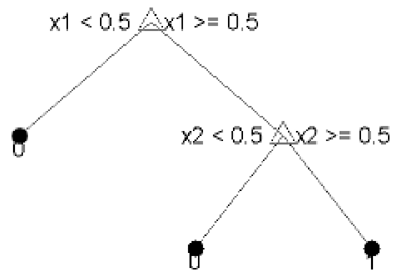


Figure 4.2: Decision Tree Illustration [2]

decision trees were used in forming the Bagged Trees classifier (see subsection 4.4.2).

4.4.2 Bagged Trees

The Bagged Trees classifier makes use of the Random Forrest algorithm. Multiple individual decision trees are generated and the majority consensus is taken to determine whether a vulnerability has been exploited (see Figure 4.3). A maximum number of trees may be specified. In this study, a maximum of 30 trees were used because this number was sufficient to demonstrate the predictive power of the algorithm without generating excess computationally expensive trees. An Individual Tree classifier was also used for comparison (see subsection 4.4.1).

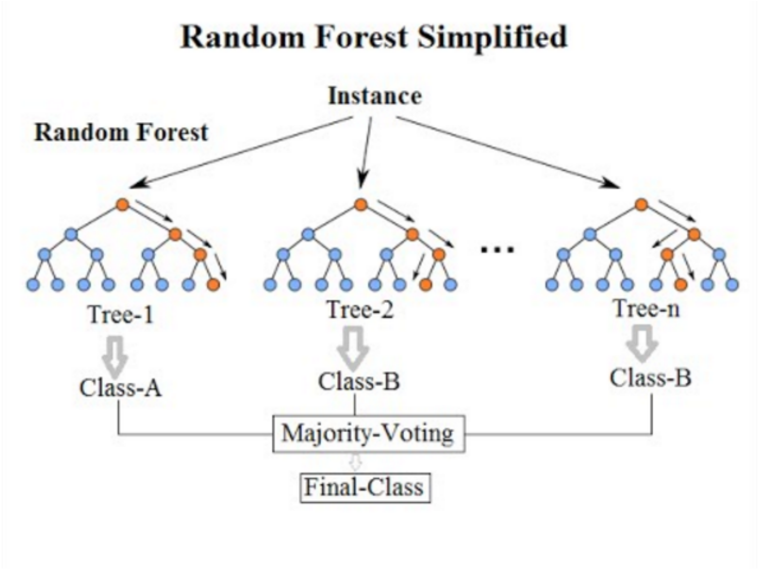


Figure 4.3: Random Forest Illustration [3]

4.4.3 Neural Network

In a Neural Network, features of the data samples are inputs to the Neural Network. These inputs are processed by a hidden layer, which consists of many neurons. A model is trained to determine how much weight each neuron should give its input. Once trained, the inputs are processed by the hidden layer using the weights that have been determined. Further weighting of the outputs from the hidden layer is used to determine the final output. This process is illustrated in Figure 4.4. In the architecture for this study, one hidden layer is used with 20 hidden neurons. Like the Bagged Trees structure, this set-up was adequate to demonstrate the model's predictive power.

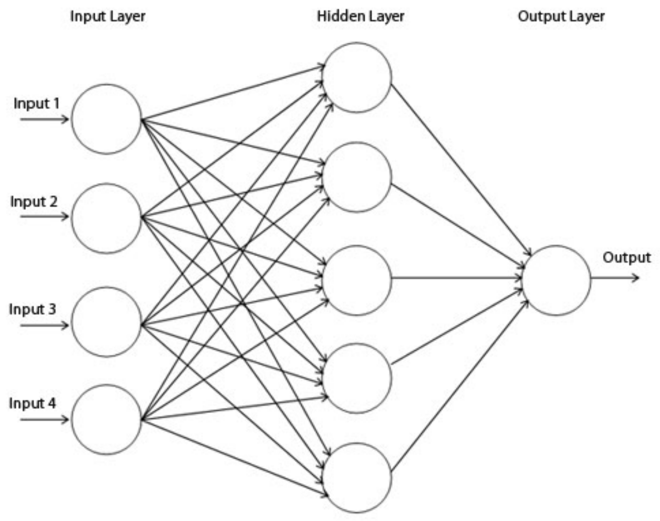


Figure 4.4: Neural Network Illustration [4]

4.4.4 Support Vector Machine Models

A Support Vector Machine (SVM) attempts to find the optimal separation boundary between data classes. For a Linear SVM, which was primarily used during this research, the separation between classes is represented by a linear equation in two variables. This method has been used for spam classification of email and would seem to be similarly applicable to classifying vulnerabilities based on descriptions.

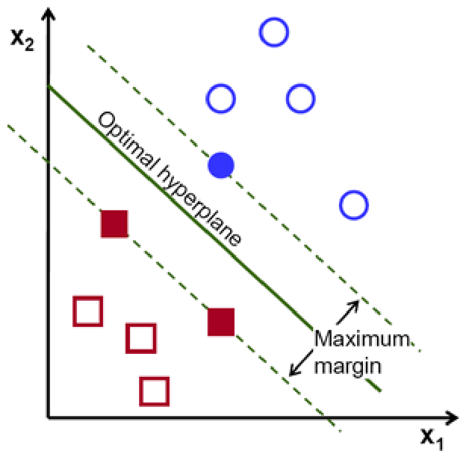


Figure 4.5: Linear SVM Illustration [5]

Figure 4.5 shows a separation boundary formed by a Linear SVM.

The model for this study was developed using the LibLinear package, as had been done by RecordedFuture [16]. This package is particularly useful for creating these types of models for large datasets.

4.4.5 Models Results and Discussion

A summary of the results from the models is shown in Table 4.3 and Table 4.4.

The results were consistent with those from the RecordedFuture paper [16]. That paper concluded that a Linear SVM classifier had the best predictive accuracy. The Bagged Trees model had the highest accuracy on the training and test set. In this study, the Bagged Trees model also had the highest precision and recall on the training set and highest precision on the test set. The LibLinear model had the highest recall on the test set. From these results on the training and test sets, the Bagged Trees model appeared to have the best performance overall.

Although the study in this thesis found higher accuracy (around 90% with the

Table 4.3: Machine Learning Model Results on Training Set (59,121 samples)

Model Name	Info	Accuracy	Precision	Recall
Fine Tree	Cost = 2	0.9030	0.6308	0.6164
Bagged Trees	Cost = 2	0.9742	0.8657	0.9495
Neural Net	Hidden Neurons = 20	0.9060	0.6730	0.5270
LibLinear	Cost = 21	0.9007	0.6071	0.6780

Table 4.4: Machine Learning Model Results on Test Set (10,433 samples)

Model Name	Info	Accuracy	Precision	Recall
Fine Tree	Cost = 2	0.8965	0.5964	0.5928
Bagged Trees	Cost = 2	0.9113	0.6614	0.6302
Neural Net	Hidden Neurons = 20	0.9030	0.6570	0.5260
LibLinear	Cost = 21	0.8903	0.5544	0.6659

models generated), the precision and recall results were lower. From RecordedFuture, precision and recall in their tests were generally found to be 80-90%, whereas the values resulting from this research were found not to exceed 70% [16]. This difference may be the result of using fewer description features for this research due to computational constraints than were used in the RecordedFuture study.

4.5 Looking at Predictive Power Over Time

The results found on the separate 2017 data are shown in Table 4.5. The 2017 data does not include a full year of vulnerabilities like the 2005-2016 data does.

Table 4.5: Machine Learning Model Results Applied to 2017 Data (13,336 samples)

Model Name	Info	Accuracy	Precision	Recall
Fine Tree	Cost = 2	0.9217	0.2843	0.0607
Bagged Trees	Cost = 2	0.9270	0.4062	0.0408
Neural Net	Hidden Neurons = 20	0.9270	0.4230	0.0350
LibLinear	Cost = 21	0.9038	0.2776	0.2134

From these results, it is clear that accuracy decreased when making predictions on the more recent 2017 data. Precision and recall decreased even more. However, for the LibLinear model, recall did stay above 20%, which signals that it performs well on new data. Due to the decline in performance on the new data, it is important to build a model with high recall that also takes into account vulnerabilities similar to those that have been exploited through textual analysis.

In the context of remediating vulnerabilities, it is important to have high recall in identifying the vulnerabilities for the same reason as it is in treating patients. In medicine, it is more important to make sure that people who are sick get treatment than people who are well do not get treatment. This preference is similar to the medical approach used by Kate Farris, Ph.D., in approaching vulnerability management [7].

4.6 Return to Topic Modeling

From Neuhaus's article on analyzing the NVD using trends, topics were apparent that could be assigned to a certain category [15]. In this thesis, a model is set up with many topics in order to classify select ones as high risk of being exploited (based on the occurrence of words within these topics).

The reasoning behind this approach is that, if a vulnerability is at high risk of being exploited, then it will likely fall within a topic that is representative of high-risk vulnerabilities. It will fall into this topic even if it does not directly contain the words that would cause another model to predict it as exploitable. When many of the exploits associated with vulnerabilities are unknown, such an approach can be valuable. For example, there are many zero-day exploits, for which a known vulnerability has not been reported. Further, there are many exploits in the Exploit Database not mapped to the NVD.

This model was created by taking the probability of each word falling within a specific topic and then multiplying it by the probabilities that those words are associated with an exploit. See Equation 4.1, where n is the current word and N is the final word.

$$\left(\begin{array}{c} \text{Exploit Probability} \\ \text{Score for Topic} \end{array} \right) = \sum_{n=1}^{N} \left(\begin{array}{c} \text{Probability of} \\ \text{Word in Topic} \end{array} \right) \times \left(\begin{array}{c} \text{Historical Probability} \\ \text{of Exploit} \end{array} \right) \tag{4.1}$$

The words in Table 4.6 were at a high risk of being associated with an exploit (similar to the approach in Edkrantz's paper [17]). The probabilities represent the proportion of all of the vulnerabilities that were exploited that had that word in their description.

Table 4.6: Words with highest probability of being associated with an exploit

	Word	**Probability**
1	"magic_quotes_gpc"	0.6805
2	"register_globals"	0.6552
3	"joomla"	0.6046
4	"inclusion"	0.5911
5	"catid"	0.5556
6	"detail"	0.5472
7	"cid"	0.5469
8	"mambo"	0.5436
9	"mdb"	0.5373
10	"disabled"	0.4933

The result of Equation 4.1 is a matrix where the columns may be summed to find the risk level of that topic being associated with exploits. Figure 4.6 shows word clouds for the top four topics found to present a high risk of being exploited (out of

12 topics that were generated). Larger words occur more frequently in the respective topic.

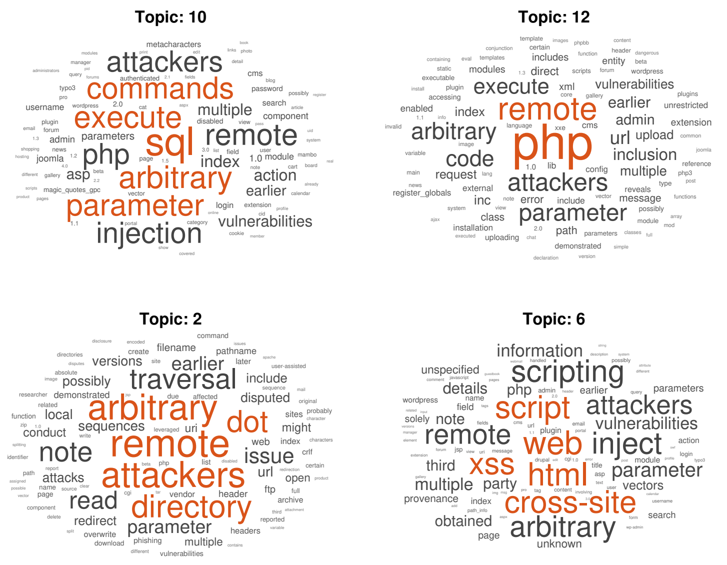


Figure 4.6: High Exploit Probability Topic Cloud

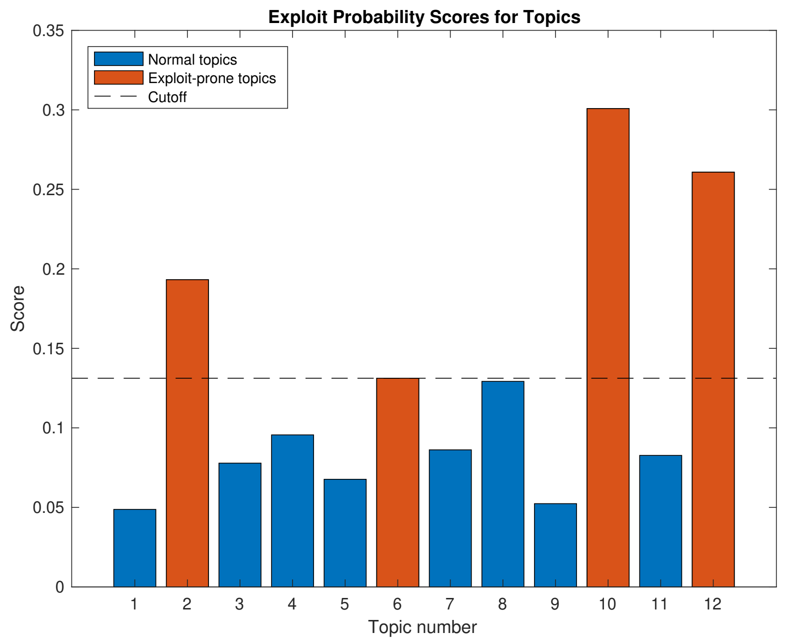


Figure 4.7: Topic Model Bar Chart Threshold

It is possible to assign thresholds to these topics to determine if they are high risk. For example, if a description is likely to fall within one of the top $\frac{1}{3}^{\text{rd}}$ of at-risk topics, then it should be classified as at high risk of being exploited. The bar chart in Figure 4.7 illustrates this example of probability scores and the threshold cutoff for the 12 topics generated.

In Table 4.7, the results of this approach for various cutoffs are presented (shown with p in the first column). As the table shows, the accuracy, precision, and recall of this model are consistent with the results of the other models. The proportion cutoff can be easily adjusted based on the desired number of exploits. At its peak

Table 4.7: Topic Modeling Approach Results (12 topics)

	Accuracy	**Precision**	**Recall**	**No. Samples**
p=1/12				
Training Data	0.8642	0.4627	0.3183	59,121
Test Data	0.8597	0.4661	0.3195	10,433
2017 (New) Data	0.9153	0.2753	0.1109	13,336
p=1/6				
Training Data	0.8537	0.4427	0.5114	59,121
Test Data	0.8555	0.4657	0.5282	10,433
2017 (New) Data	0.8940	0.2049	0.1663	13,336
p=1/4				
Training Data	0.8315	0.4018	0.6222	59,121
Test Data	0.8328	0.4192	0.6397	10,433
2017 (New) Data	0.8620	0.1635	0.2249	13,336
p=1/3				
Training Data	0.7170	0.2701	0.6990	59,121
Test Data	0.7176	0.2809	0.7091	10,433
2017 (New) Data	0.8149	0.1272	0.2699	13,336
p=1/2				
Training Data	0.5559	0.2057	0.8519	59,121
Test Data	0.5592	0.2161	0.8706	10,433
2017 (New) Data	0.6846	0.1050	0.4519	13,336

performance, this model predicts whether a vulnerability was exploited 85-90% of the time and it can be adjusted to predict only true positives with 50-60% recall (with slightly lower overall accuracy). It also performs well on the new 2017 data.

The most important feature of this model is that the exploits are categorized into topics with similar characteristics. By using this method to classify vulnerabilities, it makes it possible to understand exploits and vulnerabilities in groups, making it easier to address them. In addition, based on this model, it is possible to list the vulnerabilities in order based on the probabilities that they have of belonging to certain topics. The order of the results could be useful to determine which vulnerabilities should be researched and receive first priority, if they have not already been exploited.

$$\begin{pmatrix}\text{Exploit Probability}\\ \text{Score for Vulnerability}\end{pmatrix} = \begin{bmatrix}\text{Probability of Vulnerability}\\ \text{belonging to a Topic}\end{bmatrix} \times \begin{bmatrix}\text{Historical Probability of}\\ \text{Topic being Exploited}\end{bmatrix} \tag{4.2}$$

Such a model that combines machine learning and topic modeling methods presents an innovative way to determine which vulnerabilities are at a high risk of being exploited. Vulnerabilities that score high in Equation 4.2 are at a high risk of being exploited.

4.7 Machine Learning Results Summary

In this chapter, vulnerabilities were found to be reported after an exploit had already been published (indicating zero-day vulnerabilities). Various machine-learning models were used to predict whether a vulnerability had been exploited based on its descriptions and severity scores. The Bagged Trees model was found to perform well.

With the previous machine learning models, the false positives were not understood in detail (i.e. vulnerabilities that were predicted by a model to have been exploited, but were actually not). With a new topic modeling based model, vulnerabilities are predicted to be exploited based on whether they belong to certain high-risk topics. This categorization allows high-risk vulnerabilities to be associated with each other, whether or not they have actually been exploited. This new grouping will ideally help with the identification and remediation of high-risk vulnerabilities before they are exploited.

Chapter 5

Conclusion

The following findings from this research on the NVD are important in the context of existing research:

1. **Correlation with updates** - It was clear that there was a strong correlation between the incidence of vulnerabilities and when updates for products were released (i.e. for Apple, which has accessible data on updates). Further, when an update could not explain why there was an increase in vulnerabilities, there was usually another factor present, such as a new product release.

2. **Negative time between exploits and vulnerabilities** - Many exploits are reported before an associated vulnerability being published in the NVD. This finding indicates that most vulnerabilities are being exploited before vendors have reported the vulnerability (referred to as zero-day exploits).

3. **Feature consistency** - The distribution of vulnerabilities by severity level remained consistent over time. However, the distribution of vulnerabilities by operating system varied. When analyzing the data, it is important to recognize

this variability in how the vulnerabilities are distributed.

4. **Topics are present in the data** - It is clear that there are distinct topics that are present in the NVD data. These topics vary over time, indicating the changing nature of vulnerabilities that are recorded in the NVD. These topics may be used to support vulnerability classification.

5. **Machine learning model findings** - The machine learning models that were tested in this study performed well on the data and generally had high accuracy, precision, and recall. However, when applied to new data that was outside of the timeframe of the model, these models did not perform as well.

6. **Topic modeling classification model** - It is possible to form a classification model based on topics in the data. This approach may be preferable because it would support classification of not yet exploited vulnerabilities that fall within topics that are associated with exploits.

Computer security is a major concern. The National Vulnerability Database (NVD) alone contains 100,000 descriptions of these known vulnerabilities, of which 6,000 have been added in just the last three months (as of April 2018). Clearly, these vulnerabilities are a growing issue. The vulnerability descriptions in the NVD consist of many data features, including scores and textual descriptions.

This thesis contributes to the understanding and value of the NVD, with the goal of analyzing it for useful information and improving computer security. At first, the presence of trends in the NVD data was researched and confirmed through topic modeling. Next, the power of these features to predict whether or not a vulnerability had been exploited was examined and substantiated. Combining these two results, this thesis presents a model that uses topics in the NVD to classify whether vulnerabilities have been exploited. At its peak performance, this model predicts whether a vulnerability has been exploited 85-90% of the time and can be adjusted to predict only true positives with 50-60% recall.

5.1 Future Directions

It is clear from the results of the machine learning analysis that it is possible to predict with reasonable accuracy whether certain vulnerabilities will be exploited and to group vulnerabilities based on topics. It would be interesting to look at vulnerabilities that have not yet been exploited but fall within commonly exploited topics. These vulnerabilities may be at a high risk of being exploited in the future or may have already been compromised.

Further, it would be beneficial to use these models to help address what vulnerabilities system support specialists should prioritize and focus on remediating. Vulnerabilities that these models identify as having a high likelihood of being exploited should be addressed first. In the future, such an approach may make it more difficult to take advantage of high-risk vulnerabilities that do not yet have documented exploits.

Bibliography

[1] J. Sullivan, "Effects of vulnerability publications on stock returns," Dartmouth College, Economics 46 Final Paper, May 2018.

[2] "Choose classifier options," Mathworks, 2017, [Accessed Dec 2017]. [Online]. Available: https://www.mathworks.com/help/stats/choose-a-classifier.html

[3] "Random forest algorithm, an interactive discussion," LinkedIn, June 2016, [Accessed Dec 2017]. [Online]. Available: https://community.tibco.com/wiki/random-forest-template-tibco-spotfirer-wiki-page

[4] "Single hidden layer neural network titanic tests," Medium, Nov 2017, [Accessed Dec 2017]. [Online]. Available: https://www.nicolamanzini.com/single-hidden-layer-neural-network/

[5] "Introduction to support vector machines," OpenCV, 2017, [Accessed Dec 2017]. [Online]. Available: https://docs.opencv.org/2.4/doc/tutorials/ml/introduction_to_svm/introduction_to_svm.html

[6] "NVD - General information," National Institute of Standards and Technology, [Accessed Dec 2017]. [Online]. Available: https://nvd.nist.gov/general

[7] K. Farris, J. Sullivan, and G. Cybenko, "Vulnerability survival analysis: A novel approach to vulnerability management," pp. 10 184 – 10 184 – 14, 2017. [Online]. Available: http://dx.doi.org/10.1117/12.2266378

[8] "About the Exploit Database," Offensive Security, [Accessed Dec 2017]. [Online]. Available: https://www.exploit-db.com/about-exploit-db/

[9] C. Manes, "2015's MVPs - The most vulnerable players," TechTalk, Apr 2016, [Accessed July 2016]. [Online]. Available: https://techtalk.gfi.com/2015s-mvps-the-most-vulnerable-players/

[10] I. Guneydas, "Using the National Vulnerability Database to reveal vulnerability trends over time," Rapid7, Apr 2016, [Accessed July 2016]. [Online]. Available: https://blog.rapid7.com/2016/04/20/using-the-national-vunerability-database-to-reveal-vulnerability-trends-over-time/

[11] R. Kuhn and C. Johnson, "Vulnerability trends: Measuring progress," *IT professional*, vol. 12, no. 4, pp. 51–53, 2010.

[12] Y. Y. Chang, P. Zavarsky, R. Ruhl, and D. Lindskog, "Trend analysis of the CVE for software vulnerability management," in *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, Oct 2011, pp. 1290–1293.

[13] S. Zhang, D. Caragea, and X. Ou, "An empirical study on using the National Vulnerability Database to predict software vulnerabilities," in *Database and Expert Systems Applications*, A. Hameurlain, S. W. Liddle, K.-D. Schewe, and X. Zhou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 217–231.

[14] E. Carabott, “The pitfalls of interpreting vulnerability data,” TechTalk, Mar 2015, [Accessed July 2016]. [Online]. Available: https://techtalk.gfi.com/interpreting-data/

[15] S. Neuhaus and T. Zimmermann, “Security trend analysis with CVE topic models,” in *2010 IEEE 21st International Symposium on Software Reliability Engineering*, Nov 2010, pp. 111–120.

[16] “Anticipating cyber vulnerability exploits using machine learning,” Recorded Future, Somerville, MA, Tech. Rep., July 2015.

[17] M. Edkrantz, “Predicting exploit likelihood for cyber vulnerabilities with machine learning,” Master’s thesis, Chalmers Unversity of Technology Department of Computer Science and Engineering, Gothenburg, Sweden, 2015.

Appendices

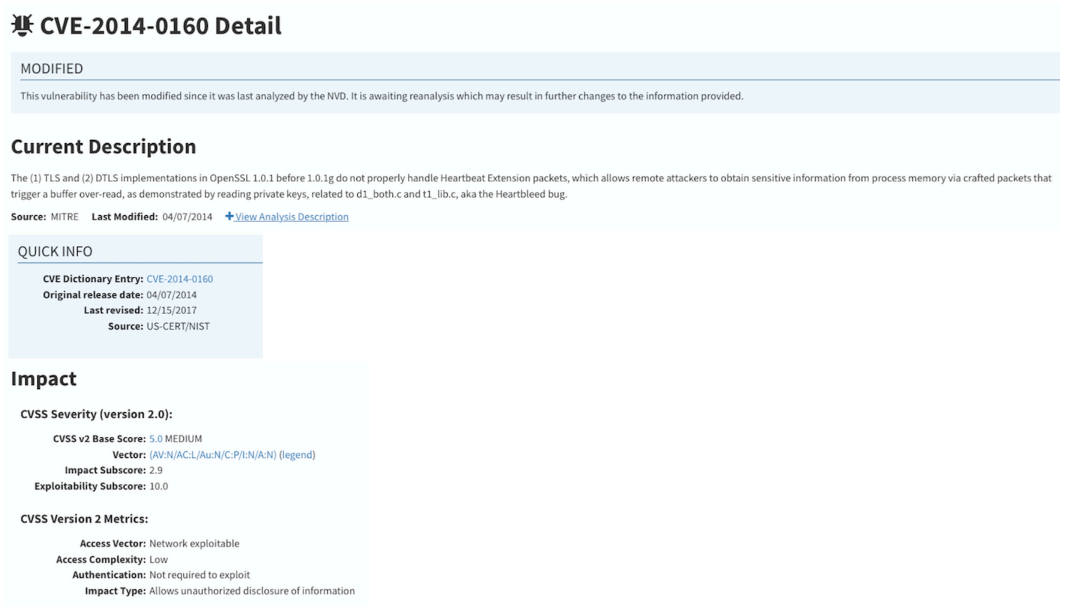
Appendix A

Sample Figures

A.1 NVD

A.1.1 Example Entry

Figure A.1 shows the Heartbleed vulnerability on the NVD website.



CVE-2014-0160 Detail

MODIFIED

This vulnerability has been modified since it was last analyzed by the NVD. It is awaiting reanalysis which may result in further changes to the information provided.

Current Description

The (1) TLS and (2) DTLS implementations in OpenSSL 1.0.1 before 1.0.1g do not properly handle Heartbeat Extension packets, which allows remote attackers to obtain sensitive information from process memory via crafted packets that trigger a buffer over-read, as demonstrated by reading private keys, related to d1_both.c and t1_lib.c, aka the Heartbleed bug.

Source: MITRE **Last Modified:** 04/07/2014 +View Analysis Description

QUICK INFO

CVE Dictionary Entry: CVE-2014-0160
Original release date: 04/07/2014
Last revised: 12/15/2017
Source: US-CERT/NIST

Impact

CVSS Severity (version 2.0):

CVSS v2 Base Score: 5.0 MEDIUM
Vector: (AV:N/AC:L/Au:N/C:P/I:N/A:N) (legend)
Impact Subscore: 2.9
Exploitability Subscore: 10.0

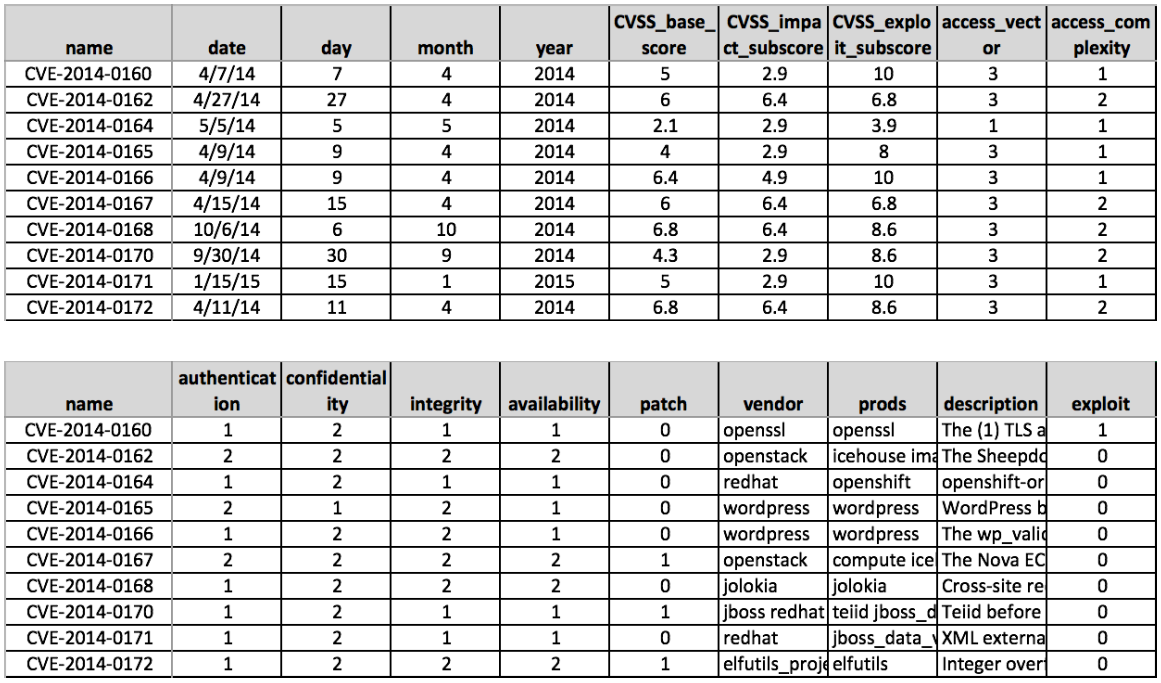
CVSS Version 2 Metrics:

Access Vector: Network exploitable
Access Complexity: Low
Authentication: Not required to exploit
Impact Type: Allows unauthorized disclosure of information

Figure A.1: NVD Website Entry

A.1.2 Same Entry in Machine Learning Data Table

The top entry in Figure A.2 is the same as the entry shown on the website in Figure A.1. This table shows the format of the dataset that was used for the analysis in this research.



name	date	day	month	year	CVSS_base_score	CVSS_impact_subscore	CVSS_exploit_subscore	access_vector	access_complexity
CVE-2014-0160	4/7/14	7	4	2014	5	2.9	10	3	1
CVE-2014-0162	4/27/14	27	4	2014	6	6.4	6.8	3	2
CVE-2014-0164	5/5/14	5	5	2014	2.1	2.9	3.9	1	1
CVE-2014-0165	4/9/14	9	4	2014	4	2.9	8	3	1
CVE-2014-0166	4/9/14	9	4	2014	6.4	4.9	10	3	1
CVE-2014-0167	4/15/14	15	4	2014	6	6.4	6.8	3	2
CVE-2014-0168	10/6/14	6	10	2014	6.8	6.4	8.6	3	2
CVE-2014-0170	9/30/14	30	9	2014	4.3	2.9	8.6	3	2
CVE-2014-0171	1/15/15	15	1	2015	5	2.9	10	3	1
CVE-2014-0172	4/11/14	11	4	2014	6.8	6.4	8.6	3	2

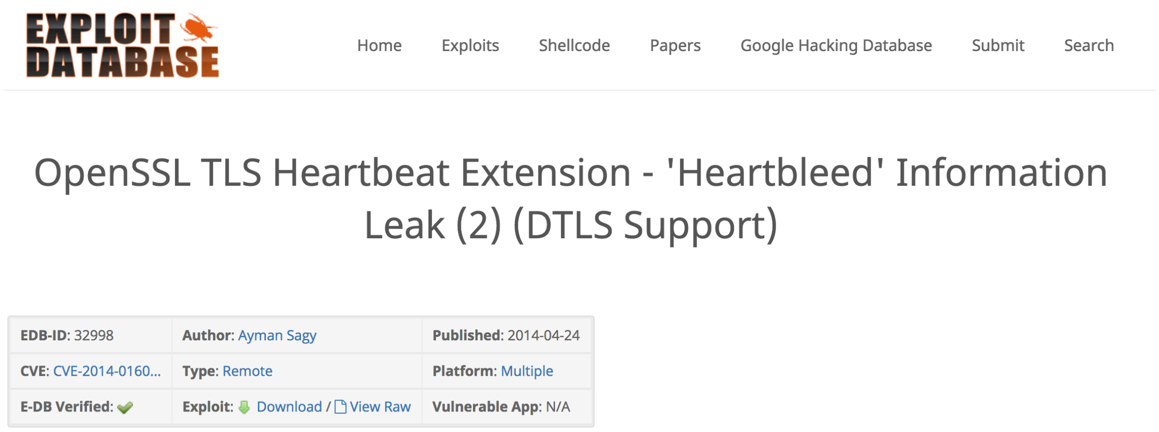
name	authentication	confidentiality	integrity	availability	patch	vendor	prods	description	exploit
CVE-2014-0160	1	2	1	1	0	openssl	openssl	The (1) TLS a	1
CVE-2014-0162	2	2	2	2	0	openstack	icehouse ima	The Sheepdo	0
CVE-2014-0164	1	2	1	1	0	redhat	openshift	openshift-or	0
CVE-2014-0165	2	1	2	1	0	wordpress	wordpress	WordPress b	0
CVE-2014-0166	1	2	2	1	0	wordpress	wordpress	The wp_vali	0
CVE-2014-0167	2	2	2	2	1	openstack	compute ice	The Nova EC	0
CVE-2014-0168	1	2	2	2	0	jolokia	jolokia	Cross-site re	0
CVE-2014-0170	1	2	1	1	1	jboss redhat	teiid jboss_d	Teiid before	0
CVE-2014-0171	1	2	1	1	0	redhat	jboss_data_v	XML externa	0
CVE-2014-0172	1	2	2	2	1	elfutils_proje	elfutils	Integer over	0

Figure A.2: NVD Datasheet Entry

A.2 Exploit DB

A.2.1 Example Entry

For the same vulnerability, Figure A.3 shows an exploit that takes advantage of the vulnerability from the Exploit DB website.



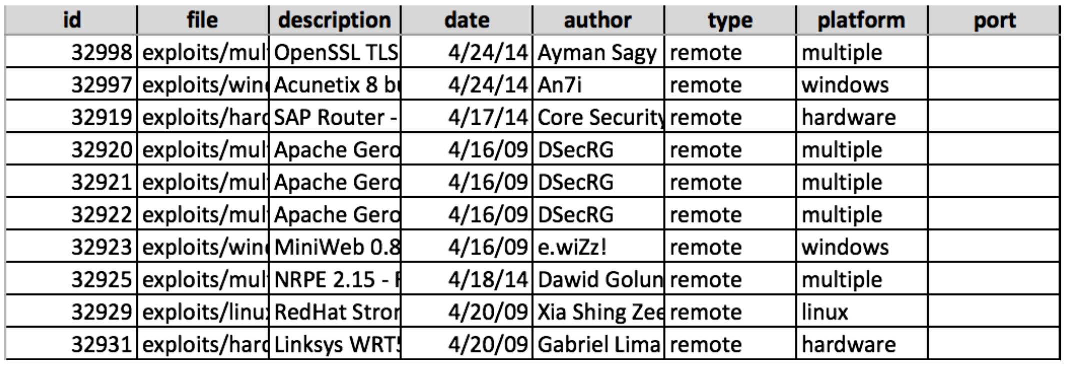
OpenSSL TLS Heartbeat Extension - 'Heartbleed' Information Leak (2) (DTLS Support)

EDB-ID: 32998	**Author:** Ayman Sagy	**Published:** 2014-04-24
CVE: CVE-2014-0160...	**Type:** Remote	**Platform:** Multiple
E-DB Verified: ✓	**Exploit:** Download / View Raw	**Vulnerable App:** N/A

Figure A.3: Exploit DB Website Entry

A.2.2 Same Entry in Exploit DB Data Table

The exploit from Figure A.3 is shown in the Exploit DB dataset in the top of the table in Figure A.4.

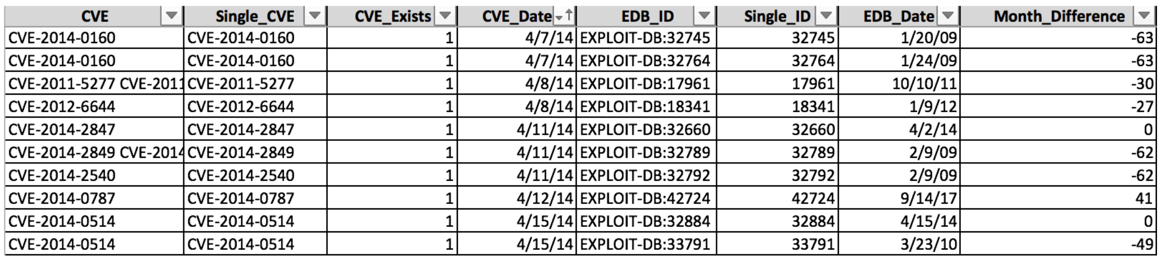


id	file	description	date	author	type	platform	port
32998	exploits/mul	OpenSSL TLS	4/24/14	Ayman Sagy	remote	multiple	
32997	exploits/win	Acunetix 8 bu	4/24/14	An7i	remote	windows	
32919	exploits/hard	SAP Router -	4/17/14	Core Security	remote	hardware	
32920	exploits/mul	Apache Gero	4/16/09	DSecRG	remote	multiple	
32921	exploits/mul	Apache Gero	4/16/09	DSecRG	remote	multiple	
32922	exploits/mul	Apache Gero	4/16/09	DSecRG	remote	multiple	
32923	exploits/win	MiniWeb 0.8	4/16/09	e.wiZz!	remote	windows	
32925	exploits/mul	NRPE 2.15 - I	4/18/14	Dawid Golun	remote	multiple	
32929	exploits/linu	RedHat Stror	4/20/09	Xia Shing Zee	remote	linux	
32931	exploits/hard	Linksys WRT!	4/20/09	Gabriel Lima	remote	hardware	

Figure A.4: Exploit DB Datasheet Entry

A.2.3 Same Entry in Exploit DB / NVD Date Comb Table

Figure A.5 shows a cross listing of this vulnerability and exploit. This table was used to generate the histogram in section 4.2.



CVE	Single_CVE	CVE_Exists	CVE_Date	EDB_ID	Single_ID	EDB_Date	Month_Difference
CVE-2014-0160	CVE-2014-0160	1	4/7/14	EXPLOIT-DB:32745	32745	1/20/09	-63
CVE-2014-0160	CVE-2014-0160	1	4/7/14	EXPLOIT-DB:32764	32764	1/24/09	-63
CVE-2011-5277 CVE-2011	CVE-2011-5277	1	4/8/14	EXPLOIT-DB:17961	17961	10/10/11	-30
CVE-2012-6644	CVE-2012-6644	1	4/8/14	EXPLOIT-DB:18341	18341	1/9/12	-27
CVE-2014-2847	CVE-2014-2847	1	4/11/14	EXPLOIT-DB:32660	32660	4/2/14	0
CVE-2014-2849 CVE-2014	CVE-2014-2849	1	4/11/14	EXPLOIT-DB:32789	32789	2/9/09	-62
CVE-2014-2540	CVE-2014-2540	1	4/11/14	EXPLOIT-DB:32792	32792	2/9/09	-62
CVE-2014-0787	CVE-2014-0787	1	4/12/14	EXPLOIT-DB:42724	42724	9/14/17	41
CVE-2014-0514	CVE-2014-0514	1	4/15/14	EXPLOIT-DB:32884	32884	4/15/14	0
CVE-2014-0514	CVE-2014-0514	1	4/15/14	EXPLOIT-DB:33791	33791	3/23/10	-49

Figure A.5: NVD-EDB Cross Listing

Appendix B

Programs

This appendix includes programs that were written for the analysis throughout this thesis. Unless otherwise stated, the following is Matlab code.

B.1 XQuery Code

The following code was used to extract data from the raw XML files provided by the NVD.

Listing 1: Code to process data from raw XML files from the NVD (XQuery)

```
declare namespace g = "http://nvd.nist.gov/feeds/cve/1.2";

let $doc := (doc("NVD/nvdcve-2002.xml"),
doc("NVD/nvdcve-2003.xml"),
doc("NVD/nvdcve-2004.xml"),
doc("NVD/nvdcve-2005.xml"),
doc("NVD/nvdcve-2006.xml"),
doc("NVD/nvdcve-2007.xml"),
doc("NVD/nvdcve-2008.xml"),
doc("NVD/nvdcve-2009.xml"),
doc("NVD/nvdcve-2010.xml"),
doc("NVD/nvdcve-2011.xml"),
doc("NVD/nvdcve-2012.xml"),
doc("NVD/nvdcve-2013.xml"),
doc("NVD/nvdcve-2014.xml"),
doc("NVD/nvdcve-2015.xml"),
doc("NVD/nvdcve-2016.xml"),
doc("NVD/nvdcve-2017.xml")
)
return
<wrapper>{
  for $v in $doc//g:entry
    return
      if ($v/@CVSS_base_score != "") then
          <div class="namegroup">{$v/@name}
          {
            <date>{string($v/@published)}</date>
          }
          {
            <CVSS_base_score>{string($v/@CVSS_base_score)}</CVSS_ba⌋
            ↪  se_score>
          }
          {
            <CVSS_impact_subscore>{string($v/@CVSS_impact_subscore)⌋
            ↪  }</CVSS_impact_subscore>
          }
          {
            <CVSS_exploit_subscore>{string($v/@CVSS_exploit_subscor⌋
            ↪  e)}</CVSS_exploit_subscore>
          }
          {
            <access_vector>{
                switch (substring(string($v/@CVSS_vector),5,1))
                    case "L" return "1"
                    case "A" return "2"
                    case "N" return "3"
                    default return ""
            }</access_vector>
          }
          {
            <access_complexity>{
                switch(substring(string($v/@CVSS_vector),10,1))
```

```
                    case "L" return "1"
                    case "M" return "2"
                    case "H" return "3"
                    default return ""
            }</access_complexity>
          }
          {
            <authentication>{
                switch(substring(string($v/@CVSS_vector),15,1))
                    case "N" return "1"
                    case "S" return "2"
                    case "M" return "3"
                    default return ""
            }</authentication>
          }
          {
            <confidentiality>{
                switch(substring(string($v/@CVSS_vector),19,1))
                    case "N" return "1"
                    case "P" return "2"
                    case "C" return "3"
                    default return ""
            }</confidentiality>
          }
          {
            <integrity>{
                switch(substring(string($v/@CVSS_vector),23,1))
                    case "N" return "1"
                    case "P" return "2"
                    case "C" return "3"
                    default return ""
            }</integrity>
          }
          {
            <availability>{
                switch(substring(string($v/@CVSS_vector),27,1))
                    case "N" return "1"
                    case "P" return "2"
                    case "C" return "3"
                    default return ""
```

```
                 }</availability>
              }
              {
                 if (exists($v/g:refs/g:ref/@patch)) then
                    <patch>{string("1")}</patch>
                 else <patch>{string("0")}</patch>
              }
              {
                for $e in distinct-values($v/g:vuln_soft/g:prod/@vendor)
                return <vendor>{concat(string($e), ' ')}</vendor>
              }
              {
                for $e in distinct-values($v/g:vuln_soft/g:prod/@name)
                return <prods>{concat(string($e), ' ')}</prods>
              }
              {
                for $e in distinct-values($v/g:desc/g:descript/text())
                return <description>{string($e)}</description>
              }
              </div>
          else ()
}</wrapper>
```

B.2 Data Processing Code

The following code was used to process the data to a format usable for analysis in Matlab once the data had already been extracted and was in a spreadsheet format. For example, all the text was tokenized into numbers.

Listing 2: Code to load data from NVD datasheet

```
close all;
clear all;

T = readtable('/Users/SULLIVAN/Documents/Research/Data/Extracted_Dat⌋
↪  a/Main/final_data.xlsx','TextType','string');
```

Listing 3: Code to process data from NVD into format usable for modeling

```
close all;
clear all;

% Paramaters

desc_freq_lim = 120;  % Remove description words with frequency < 75
vends_freq_lim = 50; % Remove vendor words with frequency < 25
prods_freq_lim = 50; % Remove product words with frequency < 25

data_rng = true;     % Limit to recent data
yr_rng_lo = 2005;    % Lower bound on years for data
yr_rng_hi = 2016;    % Upper bound on years for data

% repetitions = 7;

% Main Code

load /Users/SULLIVAN/Documents/Research/Data/Extracted_Data/Main/Raw⌋
↪  _NVD_Data
stpwrds = [importdata('stpwrds.txt'); stopWords'];
```

```

% Taking specified data range

if data_rng == true
    inds=T.year>=yr_rng_lo & T.year<=yr_rng_hi;
    T2=T(inds,:);
else
    T2 = T;
end

% Extracting scores

scores = table2array(T2(:,6:16));

% Processing vendor, product, and description data

[descs descBag] = text_proc(T2.description,desc_freq_lim,stpwrds);
[vends vendBag] = text_proc(T2.vendor,desc_freq_lim,[]);
[prods prodBag] = text_proc(T2.prods,prods_freq_lim,[]);

% Printing Top Words

descTop = topkwords(descBag,20)
vendTop = topkwords(vendBag,20)
prodTop = topkwords(prodBag,20)

% Combining dataset

x = [scores vends prods descs];
y = T2.exploit;
xy = [x y];

% Repeating exploited entries for base rate fallacy issue
% (Commented out because done when
% splitting data on the training set)

% x2 = [x; repmat(x(logical(y),:),repetitions,1)];
% y2 = [y; repmat(y(logical(y),:),repetitions,1)];
% xy2=[x2 y2];

% Clearing extra variables

clearvars -except scores descs vends prods descBag vendBag prodBag
↪   ...
    x y xy T2
```

Listing 4: Code to fit dataset format to new 2017 data

```
% Builds new data based on existing setup

% Paramaters

data_rng = true;    % Limit to recent data
yr_rng_lo = 2017;   % Lower bound on years for data
yr_rng_hi = 2017;   % Upper bound on years for data

% Main Code

load /Users/SULLIVAN/Documents/Research/Data/Extracted_Data/Main/Raw⌋
↪   _NVD_Data
stpwrds = [importdata('stpwrds.txt'); stopWords'];

% Taking specified data range

if data_rng == true
    inds=T.year>=yr_rng_lo & T.year<=yr_rng_hi;
    T3=T(inds,:);
else
    T3 = T;
end

% Extracting scores

scores_new = table2array(T3(:,6:16));

descs_new = text_cleaner(T3.description,stpwrds);
vends_new = text_cleaner_simp(T3.vendor);
prods_new = text_cleaner_simp(T3.prods);

```

```
desc_mat = full(encode(descBag,descs_new));
vend_mat = full(encode(vendBag,vends_new));
prod_mat = full(encode(prodBag,prods_new));

x_new = [scores_new vend_mat prod_mat desc_mat];
y_new = T3.exploit;
xy_new = [x_new y_new];

clearvars -except scores descs vends prods descBag vendBag prodBag
↪   ...
    descs_new vends_new prods_new desc_mat vend_mat prod_mat...
    x y xy x_new y_new xy_new T2
```

Listing 5: Code to split data into training (85%) and test set (15%)

```
inds=randsample(length(x),round(.85*length(x)));

log_inds=zeros(length(x),1);
log_inds(inds)=1;
log_inds=logical(log_inds);

x_train = x(log_inds,:);
y_train = y(log_inds,:);

x_test = x(~log_inds,:);
y_test = y(~log_inds,:);

% Repeating exploited entries for base rate fallacy issue

% repetitions = 7;
%
% x_train_2 = [x_train;
↪   repmat(x_train(logical(y_train),:),repetitions,1)];
% y_train_2 = [y_train;
↪   repmat(y_train(logical(y_train),:),repetitions,1)];
% xy_train_2=[x_train_2 y_train_2];
```

Listing 6: Code to process text from database entries

```
% Processes text and cleans up
function [counts,cleanBag] = text_prod(T,freq_lim,stpwrds)
    if isempty(stpwrds)
        clean_text = text_cleaner_simp(T);
    else
        clean_text = text_cleaner(T,stpwrds);
    end
    cleanBag = bagOfWords(clean_text);
    cleanBag = removeInfrequentWords(cleanBag,freq_lim);

    tmp = cleanBag();
    counts = full(tmp.Counts);
end
```

Listing 7: Code to tokenize text from database entries without removing stopwords

```
% Source: https://www.mathworks.com/help/textanalytics/examples/pr⌋
↪  epare-text-data-for-analysis.html

function [documents] = text_cleaner_simp(textData)
% Erase punctuation.
% cleanTextData = erasePunctuation(textData);

% Convert the text data to lowercase.
cleanTextData = lower(textData); % lower(cleanTextData);

% Tokenize the text.
documents = tokenizedDocument(cleanTextData);
documents = removeShortWords(documents,2);
end
```

Listing 8: Code to tokenize text from database entries and remove stopwords

```
% Source: https://www.mathworks.com/help/textanalytics/examples/pr⌋
↪  epare-text-data-for-analysis.html

```

```
function [documents] = text_cleaner(textData,stpwrds)
% Erase punctuation.
% cleanTextData = erasePunctuation(textData);

% Convert the text data to lowercase.
cleanTextData = lower(textData); % lower(cleanTextData);

% Tokenize the text.
documents = tokenizedDocument(cleanTextData);

% Remove a list of stop words.
documents = removeWords(documents,stpwrds);

% Remove words with 2 or fewer characters, and words with 15 or
↪  greater
% characters.
documents = removeShortWords(documents,2);
% documents = removeLongWords(documents,15);

% Normalize the words using the Porter stemmer.
% documents = normalizeWords(documents);
end
```

Listing 9: Stopwords used when processing text in addition to those from Matlab

```
allows
vulnerability
cve
cwe
mitre
org
com
aka
os
http
microsoft
apple
linux
adobe
```

```
windows
mac
win
sp
office
excel
word
powerpoint
gold
server
xp
vista
file
files
application
via
allow
attack
```

B.3 Background Code

The following is a selection of code used in the Background chapter (see chapter 2).

Listing 10: Code to plot frequency of vendor experiencing vulnerability over time

```
close all;
clear all;

col_num = 4;

BaseName='Extracted_Data/vends_';
Type = '.xlsx';
for i=2012:2016

    k = i - 2012;

    filename = [BaseName,num2str(i),Type];

    fprintf('Using %s\n', filename)

    T = readtable(filename);

    x = datetime(T{1:end,col_num},'ConvertFrom','excel').Month;
    x = x(isfinite(x(:, 1)), :);

    a = unique(x);
    freq = histc(x(:),a);

    if exist('a_f')
        a_f = vertcat(a_f, (k*12 + a) );
        freq_f = vertcat(freq_f, freq);
    else
        a_f = a;
        freq_f = freq;
    end
end

perc_f = freq_f/max(freq_f);
```

```
plot(a_f,perc_f)
```

Listing 11: Testing Apple vulnerability correlation with updates (Stata)

```
use Extracted_Data/apple_vulns

// 2007-2015
// graph twoway connected vuln_freqs month, xline(10 19 30 32 42
↪  55 58 67 69 81 82 93 94 105, lpattern(dash) lcolor(gray))
↪  msymbol(i) title(Apple vulnerabilitiles and updates 2007-2015)

// 2013-2015
// graph twoway connected vuln_freqs month if month > 72, xline(81
↪  82 93 94 105, lpattern(dash) lcolor(gray)) msymbol(i)
↪  title(Apple vulnerabilitiles and updates 2013-2015)

// 2010-2012
// graph twoway connected vuln_freqs month if month <= 72 & month
↪  > 36, xline(42 55 58 67 69, lpattern(dash) lcolor(gray))
↪  msymbol(i) title(Apple vulnerabilitiles and updates 2010-2012)

// 2007-2009
// graph twoway connected vuln_freqs month if month <= 36,
↪  xline(10 19 30 32, lpattern(dash) lcolor(gray)) msymbol(i)
↪  title(Apple vulnerabilitiles and updates 2007-2009)

graph twoway connected vuln_freqs month, xline(39 47 51 63 100,
↪  lpattern(dash) lcolor(gray)) msymbol(i) title(Peaks not
↪  corresponding to updates)

reg vuln_freqs update
```

B.4 Mallet Topic Modeling Code

The following is a selection of code used in the Topic Modeling chapter (see chapter 3).

Listing 12: Code to process data with Mallet

```
./bin/mallet import-dir --input sample-data/web/en --output
↪  tutorial.mallet --keep-sequence --remove-stopwords

./bin/mallet train-topics  --input tutorial.mallet  --num-topics 10
↪  --optimize-interval 10 --output-state topic-state.gz
↪  --output-topic-keys tutorial_keys.txt --output-doc-topics
↪  tutorial_composition.txt


./bin/mallet import-dir --input data_3mos/microsoft --output
↪  microsoft.mallet --keep-sequence --remove-stopwords
↪  --extra-stopwords nvd-stopwords.txt

./bin/mallet train-topics  --input microsoft.mallet  --num-topics 9
↪  --num-top-words 7 --optimize-interval 10 --output-state
↪  topic-state.gz  --output-topic-keys microsoft_keys.txt
↪  --output-doc-topics microsoft_comp.txt
```

Listing 13: Code to produce topic model plots from Mallet output (annual)

```
close all;
clear all;

data = xlsread('Extracted_Data/mallet_comps/adobe_comp','C2:T17');

A = data(:,1:2:end);
B = data(:,2:2:end);

A = A + 1;

out = zeros(size(A));
```

```

for r = 1:size(A,1)
    for c = 1:size(A,2)
        out(r,A(r,c)) = B(r,c);
    end
end

for i = 1:size(A,2)
    subplot(3,3,i);
    plot(1:size(A,1),out(:,i));

    axis tight
    title(['Topic #' int2str(i-1)]);
end
```

Listing 14: Code to produce topic model plots from Mallet output (quarterly)

```
close all;
clear all;

data = xlsread('Extracted_Data/mallet_coms_and_keys/3mos/comps/apple
↪  _comp','C2:T107');

A = data(:,1:2:end);
B = data(:,2:2:end);

A = A + 1;

out = zeros(size(A));

for r = 1:size(A,1)
    for c = 1:size(A,2)
        out(r,A(r,c)) = B(r,c);
    end
end

for i = 1:size(A,2)
    subplot(3,3,i);
    plot(1:size(A,1),out(:,i));
    hold on;
    plot(1:size(A,1),smooth(out(:,i)),'LineWidth',2);
    hold off;

    axis tight
    title(['Topic #' int2str(i-1)]);
end
```

B.5 Histogram Code

The following code was used to generate the histogram in section 4.2.

Listing 15: Code to produce histogram of time to exploit

```
close all;
clear all;


T = readtable(['/Users/SULLIVAN/Documents/Research/Data/' ...
    'Extracted_Data/Main/Exploit_Vuln_Dist_12-11-17.xlsx']);

% T = readtable(['/Users/SULLIVAN/Documents/Research/Data/'...
%     'Extracted_Data/EDB/Exploit_Vuln_Dist_11-16-17.xlsx']);

histfit(T.Month_Difference,10)

title('Distribution of Time Between Vulnerabilities and Exploits (10
↪  bins)')
xlabel('Months')
ylabel('Frequency')

print -depsc2 hist1_norm

figure(2)
histogram(T.Month_Difference,10)

title('Distribution of Time Between Vulnerabilities and Exploits (10
↪  bins)')
xlabel('Months')
ylabel('Frequency')

print -depsc2 hist1

figure(3)
histogram(T.Month_Difference,25)

```

```
title('Distribution of Time Between Vulnerabilities and Exploits (25
↪  bins)')
xlabel('Months')
ylabel('Frequency')

print -depsc2 hist2

figure(4)
histogram(T.Month_Difference,50)
title('Distribution of Time Between Vulnerabilities and Exploits (50
↪  bins)')
xlabel('Months')
ylabel('Frequency')

print -depsc2 hist3
```

B.6 Models Code

The following is the code used to generate and make predictions with the machine learning models (see section 4.4).

Listing 16: Code to train Fine Tree model

```
numSplits = 100;
costs = [0 1; 2 0];
fineTree = fitctree(x_train,y_train,'cost',...
    costs,'MaxNumSplits',numSplits)
```

Listing 17: Code to make predictions based on Fine Tree model

```
function [] = tree_predict(x,y,model)

[labels, scores, cost] = predict(model,x);

out = labels;

plotconfusion(y',out')

acc = sum(out==y)/length(y)
prec = sum(y(logical(out)))/sum(out)
rec = sum(out(logical(y)))/sum(y)
```

Listing 18: Code to train Bagged Trees model

```
numTrees = 30;
costs = [0 1; 2 0];
baggedTrees = TreeBagger(numTrees,x_train,y_train,'cost',costs)
```

Listing 19: Code to make predictions based on Bagged Trees model

```
function [] = tree_bagger_predict(x,y,model)

[labels, scores, cost] = predict(model,x);

out = str2num(cell2mat(labels));

plotconfusion(y',out')

acc = sum(out==y)/length(y)
prec = sum(y(logical(out)))/sum(out)
rec = sum(out(logical(y)))/sum(y)
```

Listing 20: Code to produce LibLinear model and test on data

```
% Test training and test sets and plot confusions

cost=21;

model=train(y_train,x_train,['-c ' num2str(cost)]);

out=predict(y_train,x_train,model);

prec1 = sum(y_train(logical(out)))/sum(out)
rec1 = sum(out(logical(y_train)))/sum(y_train)

figure(1);
plotconfusion(y_train',out')

out2=predict(y_test,x_test,model);

prec2 = sum(y_test(logical(out2)))/sum(out2)
rec2 = sum(out2(logical(y_test)))/sum(y_test)

figure(2);
plotconfusion(y_test',out2')

out3=predict(y_new,x_new,model);

```

```
prec2 = sum(y_new(logical(out3)))/sum(out3)
rec2 = sum(out3(logical(y_new)))/sum(y_new)

figure(3);
plotconfusion(y_new',out3')
```

B.7 Topic Modeling Prediction Code

The following is the code used for the predictive topic modeling approach (see section 4.6).

Listing 21: Code to generate topics for NVD descriptions

```
% Fitting topics to vulnerability descriptions

tm_descBag = removeDocument(descBag,setdiff(1:length(x),inds));
mdl = fitlda(tm_descBag,12,'Verbose',0);

% Graphing
% figure;
% for topicIdx = 1:4
%     subplot(2,2,topicIdx)
%     wordcloud(mdl,topicIdx);
%     title("Topic: " + topicIdx)
% end
```

Listing 22: Code to produce predictive topic model from topics

```
close all;

% Proportion of topics to take
prop = 1/3

% Ranking topics by likelihood of being exploited
desc_bin = double(tm_descBag.Counts ~= 0);

desc_probs = sum(desc_bin.*y_train)./sum(desc_bin);
[sorted_probs, sorted_inds] = sort(desc_probs,'descend');
top_words = tm_descBag.Vocabulary(sorted_inds(1:10))';
top_word_probs = sorted_probs(1:10);

exp_score = desc_probs*mdl.TopicWordProbabilities;

```

```
% Sorting the most exploited topics
[m, t] = sort(exp_score,'descend');

% Setting top set proportion of topics as likely to be exploited
sub = t(1:round(length(t)*prop));

% Graphing topics with high chance of exploit

for topicIdx = 1:4
    subplot(2,2,topicIdx)
    wordcloud(mdl,t(topicIdx));
    title("Topic: " + t(topicIdx));
end

print -depsc2 topic_cloud

% Graphing a bar chart
data1=exp_score;
data2=exp_score;
data1(~sub)=0;
data2(setdiff(1:length(t),sub))=0;

figure(2)
bar(data1)
hold on
bar(data2)

% Plotting cutoff
xlim=get(gca,'xlim');
hold on
plot(xlim,[min(exp_score(sub)) min(exp_score(sub))],'--k')

% Formatting
set(gca,'xtick',1:12)
title('Exploit Probability Scores for Topics');
ylabel('Score');
xlabel('Topic number');
legend('Normal topics','Exploit-prone topics','Cutoff',...
    'Location','northwest');

```

```
print -depsc2 topic_cutoffs

% Determining to which topic each vulnerability belongs
A = mdl.DocumentTopicProbabilities;
[~, q] = max(A, [], 2) ;

% Classifying whether vulnerability likely to have
% an exploit based on the proportion criteria
out = sum(double(q==sub),2) ; % Row-wise sum

% Plotting confusion matrix of these results
figure(3)
plotconfusion(y_train',out')

% Finding accuracy, precision, and recall
acc = sum(out==y_train)/length(y_train)
prec = sum(y_train(logical(out)))/sum(out)
rec = sum(out(logical(y_train)))/sum(y_train)
```

Listing 23: Code to test predictive topic model on new data

```
% Running Test Set

stpwrds = [importdata('stpwrds.txt'); stopWords'];

descs_test = text_cleaner(T2(setdiff(1:height(T2),inds),:).descripti
 ↪  on,stpwrds);

testTopicMixture=transform(mdl,descs_test);

% Determining to which topic each vulnerability belongs
[~, q] = max(testTopicMixture, [], 2) ;

% Classifying whether vulnerability likely to have
% an exploit based on the proportion criteria
out = sum(double(q==sub),2) ; % Row-wise sum

% Plotting confusion matrix of these results
figure(3)
```

```
plotconfusion(y_test',out')

% Finding accuracy, precision, and recall
acc1 = sum(out==y_test)/length(y_test)
prec1 = sum(y_test(logical(out)))/sum(out)
rec1 = sum(out(logical(y_test)))/sum(y_test)

% Running New Set

newTopicMixture=transform(mdl,descs_new);

% Determining to which topic each vulnerability belongs
[~, q] = max(newTopicMixture, [], 2) ;

% Classifying whether vulnerability likely to have
% an exploit based on the proportion criteria
out = sum(double(q==sub),2) ; % Row-wise sum

% Plotting confusion matrix of these results
figure(4)
plotconfusion(y_new',out')

% Finding accuracy, precision, and recall
acc2 = sum(out==y_new)/length(y_new)
prec2 = sum(y_new(logical(out)))/sum(out)
rec2 = sum(out(logical(y_new)))/sum(y_new)
```