A preliminary analysis of quantifying computer security vulnerability data in "the wild"

Katheryn A. Farris[a], Sean R. McNamara[b], Adam Goldstein[b], and George Cybenko[a]

[a]Dartmouth College, Thayer School of Engineering
[b]Dartmouth College, Computing Services

ABSTRACT

A system of computers, networks and software has some level of vulnerability exposure that puts it at risk to criminal hackers. Presently, most vulnerability research uses data from software vendors, and the National Vulnerability Database (NVD). We propose an alternative path forward through grounding our analysis in data from the operational information security community, i.e. vulnerability data from "the wild". In this paper, we propose a vulnerability data parsing algorithm and an in-depth univariate and multivariate analysis of the vulnerability arrival and deletion process (also referred to as the vulnerability birth-death process). We find that vulnerability arrivals are best characterized by the log-normal distribution and vulnerability deletions are best characterized by the exponential distribution. These distributions can serve as prior probabilities for future Bayesian analysis. We also find that over 22% of the deleted vulnerability data have a rate of zero, and that the arrival vulnerability data is always greater than zero. Finally, we quantify and visualize the dependencies between vulnerability arrivals and deletions through a bivariate scatterplot and statistical observations.

Keywords: Vulnerability Data, Operational Vulnerability Management, Nessus Scanner, Exploratory Data Analysis, Heavy-Tailed Distribution, Univariate Analysis, Bivariate Analysis

1. INTRODUCTION

Information security practitioners face a double challenge when it comes to patching network vulnerabilities. They have more patches than they can possibly respond to. Network penetration testing software outputs volumes of data to include Common Vulnerability Enumerations (CVE), Common Vulnerability Scoring System (CVSS), brief descriptions, and vulnerability patching information. In fact, the United States government and industry best practices are to correlate vulnerability risk with the CVSS base scores.[1] Yet, increasing evidence indicates that the methodology by which CVSS values are calculated may need to be re-evaluated.[1,2]

To make vulnerability mitigation even more challenging, security engineers work under fierce constraints. They have limited time, resources and manpower. In ideal circumstances, they would make targeted and precise decisions to patch the most critical machines first. Yet, external factors, such as the value of one machine over another, are neglected entirely in current vulnerability software risk metrics. As a consequence, system administrators have to bootstrap projects internally and

Further author information: (Send correspondence to Katheryn A. Farris)
K. A. F.: E-mail: katheryn.a.farris.th@dartmouth.edu

Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security, Defense, and Law Enforcement Applications XV, edited by Edward M. Carapezza, Proc. of SPIE Vol. 9825, 98250T · © 2016 SPIE · CCC code: 0277-786X/16/$18 · doi: 10.1117/12.2230589

partition enterprise-level networks into zones of different criticality levels. They then have to match up the zones with the vulnerability output to determine a down-selection process of determining which machines should be addressed first, the situation is dire and many factors stand in the way of truly aggressive vulnerability management in the operational IT environment.

In this paper, we offer a preliminary analysis of vulnerability data from "the wild". Our end goal is to facilitate solutions to address current IT challenges in vulnerability management and data insight. We achieve this through estimating the state of operational IT vulnerability exposure by performing an exploratory data analysis and quantifying the relationship between the vulnerability birth and death process. In Section 1.1, we review our target research outcomes and outline the remaining sections of our paper.

1.1 Target Research Outcomes

There are multiple aspects of the problem described above we address in our study, namely:

1. Develop a data parsing algorithm for streamlined vulnerability data processing.
2. Statistically characterize the vulnerability arrival and deletion process.
3. Clearly delineate which dependencies exist between vulnerability arrivals and deletions.
4. Quantify the univariate and bivariate relationships and interpret what it means in terms of total vulnerability exposure in "the wild".
5. Identify threats to internal and external validity for our study.
6. Summarize and outline future work that our study lends itself to.

We believe that the aforementioned outcomes are important to provide a more intimate understanding of the current state of operational vulnerability management and areas of opportunity to improve the state of the art.

In Section 2, we offer a comprehensive overview of related work in vulnerability analysis. Additionally, because this study focuses on assessing the overall vulnerability exposure that puts an organization at risk to criminal exploitation, we also offer an overview of different risk models and metrics for the interested reader. In Sections 3, 4, and 5 we describe the data, our methodology for analyzing the data, and our results. In Sections 6 and 7 we provide discussion on areas of opportunity for future work, as well as our conclusions.

2. RELATED WORK

In this section, we offer an overview of related work regarding vulnerability models, risk assessments and frameworks that impact vulnerability management.

Modeling and statistically understanding vulnerabilities can provide greater insights into vulnerability management and mitigation processes. More recently, two metrics have been proposed for assessing vulnerability exposure, namely: 1.) The median active vulnerabilities (MAV); and, 2.) The vulnerability free days (VFD).[1] These metrics were deduced from a vulnerability lifeline model, which was based on measuring when vulnerabilities are reported to a vendor to when that same vendor issues patches. Our work is different in that we are specifically studying operational

IT data. Basing an analysis on when the vendor learns of a new vulnerability to when they release the patch provides a solid first step in modeling vulnerabilities; however, it does not fully capture the successes and failures of the "real-world" deployment of a vulnerability patch. Just because a vulnerability is known, or the vendor makes a patch available does not necessarily mean that the patch is effectively being used. One solution is to incorporate field data from real vulnerability scans and enterprise-level penetration-testing to gather a more complete understanding.

Vulnerability management has also been studied based on the Common Vulnerability Scoring System (CVSS). CVSS metrics provide a base score for assigning vulnerability severity. The different levels are defined by organizations such as Carnegie-Melon University's CERT, software vendors and NIST. Vulnerabilities are most often triaged by the CVSS base value, which does not include temporal or environmental factors. Incorporating external factors such as black market exploit data into the CVSS base score is among more recent work.[1] Other work presented a statistical analysis of 18 security estimation metrics based on CVSS data with time-to-compromise of 34 successful attacks. They found that security modeling with CVSS data alone does not accurately portray the time-to-compromise of a system. They also found that the security models which base metrics on only the most severe CVSS data are less reliable than those that consider all vulnerabilities, regardless of their CVSS severity.[3]

Another area area of related work stems from computational and network risk models. Specifically, Clark, et. al links an organization's objectives and relationships to network hosts through mission trees to model risk.[4] Other risk models were developed to take into account network assets that need protection before taking the vulnerabilities into account.[5] Additionally, the Quantitative Evaluation of Risk for Investment Efficient Strategies (QuERIES) provides a powerful, and interdisciplinary method for cyber security risk assessments with emerging technologies. It offers a computational approach for navigating the most cost-efficient strategies for protections in examples such as software, and intellectual property (IP).[6]

Other work can be found with Microsoft's threat modeling methodology, which is an approach for modeling cyber threats throughout a system's development lifecycle. It categorizes threats according to STRIDE, which stands for **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service, **E**levation of privilege. The methodology tracks the associated attack technique, the target, and actions of remediation that can be executed.[7]

Finally, three other models for assessing risk are TARA, MORDA and DACCA offer methods for assessing risk. TARA is an acronym for Threat Assessment and Remediation Analysis. It is a system level methodology for responding to cyber threats, and uses tools such as scoring models and threat matrices. It can be performed on already deployed systems, as well as that which is still in the acquisition lifecycle. Its strength is that it provides a flexible approach through its ability to adjust assessment levels that best suit the system. Additionally, only the tools necessary for the given system have to be used.[8] MORDA is an acronym for Mission Oriented Risk and Design Analysis. It is a risk assessment framework for the system-level. It utilizes models of the adversary, the user and the service provider. The goal is to maximize value to the user, whether the system is under attack or not. The models rely on information from subject matter experts (SMEs) and are built upon known adversary information, and patterns of attack.[9] Finally, DACCA is an acronym for Decision Analysis to Counter Cyber Attacks. DACCA weighs out the attack impact with possible responses to that impact. It employs subject matter expert (SME) assessments to inform severity of an attack, likelihood of an attack, and sophistication of the adversary's capabilities.[8]

Another interesting approach is NetSpa, produced by MIT Lincoln Laboratory. It is a software that maps computer network that are most vulnerable to hacking. It takes network configuration information as the input, and outputs worst case scenario attack graphs. The final state needs to be determined as a goal, as well as which hosts can be trusted, and, finally, maximum recursive depth-first search size.[10]

In this section, we offered an overview of many different aspects of related work from vulnerability models to cyber risk management. In the next section, we review our data and the "cleaning" process we took to get from the raw data to the final data that we analyzed.

3. DATA

In this section, we describe the data we used for our analysis, as well as the data source and software. In Section 4, we outline our methodology and the steps we followed to pre-process the data.

3.1 Data Description

We analyzed the output from a Nessus vulnerability scanner for our study. Section 3.2 describes the specific software information for Nessus. We studied seventeen sets of scan data from the two-year period of 2013-2014. The data contained almost 3,000 hosts, and eleven different. Figure 1 displays a sample of what the raw vulnerability data looks like.[11]

Plugin ID	CVE	CVSS	Host	Protocol	Port	Name	Synopsis	Description	Solution	Plugin Output
11849	CVE-2003-0831	9	192.168.150.131	tcp	21	ProFTPD File Transfer Newline Character Overflow	Arbitrary code may be run on the remote server.	The remote host is running a version of ProFTPD which seems to be vulnerable to a buffer overflow when a user downloads a malformed ASCII file. An attacker with upload privileges on this host may abuse this flaw to gain a root shell on this host. *** The author of ProFTPD did not increase the version number *** of his product when fixing this issue, so it might be false	Upgrade to ProFTPD 1.2.9 when available or to 1.2.8p	
35777	CVE-1999-0502	10	192.168.150.131	tcp	22	Default Password (toor) for 'root' Account	An account on the remote host uses a known password.	The account 'root' on the remote host has the password 'toor'. An attacker may leverage this issue to gain total control of the affected system.	Change the password for this account or disable it.	It was possible to execute the command 'id' on the remote host: uid=0(root) gid=0(root) groups=0(root)
10061	CVE-1999-0103		192.168.150.131	udp	7	Echo Service Detection	An echo service is running on the remote host.	The remote host is running the 'echo' service. This service echoes any data which is sent to it. This service is unused these days, so it is strongly advised that you disable it, as it may be used by attackers to set up denial of services attacks against this host.	- Under Unix systems, comment out the 'echo' line in /etc/inetd.conf and restart the inetd process - Under Windows systems, set the following registry key to 0 : HKLM\System\CurrentControlSet\Services\SimpTCP\Parameters\EnableTcpEcho HKLM\System\CurrentControlSet\Services\SimpTCP\Parameters\EnableUdpEcho Then launch cmd.exe and type : net stop simptcp net start simptcp To restart the service.	

Figure 1: A sample of what a typical Nessus vulnerability output file looks like. Image was sourced from https://www.tenable.com/blog/new-nessus-feature-added-csv-export.[12]

The Nessus vulnerability output typically has the following eleven attributes:*

1. Plugin ID: A software code, each with a unique ID number, that identifies what attributes and vulnerabilities it is scanning for.
2. Common Vulnerability Enumeration (CVE) code: A system for labeling the most common vulnerabilities with a uniquely identifiable number.

*Note that for the purposes of our study, we focused primarily on the first eight.

3. Common Vulnerability Scoring System (CVSS): As already described in Section 2, the CVSS value provides a base score for assigning vulnerability severity.

4. Risk Level: A one word description to notate vulnerability severity. The software vendor typically uses descriptors such as "Critical", "High", "Medium", "Low", or "Info". The CVSS value informs the risk level description for most software vendors.

5. Host Name/IP Address: The host name is a label that identifies the machine or device connected to the network, e.g. www.dartmouth.edu. The IP address is a string of numbers that uniquely identifies a machine or device that is connected to the network, e.g. 172.16.254.1. Any machine or device may have both an IP address and host name, or just an IP address.

6. Protocol: The method of communication and set of rules used by the designated host, machine or device to "talk" to another host, machine or device. TCP and UDP are examples of protocols.

7. Port Number: An identifier for the endpoint of communication on the host, machine, or device that is determined by a logical construct.

8. Vulnerability Name: A brief title of the vulnerability. Often less than five words and is chosen by the software vendor.

9. Vulnerability Synopsis: A short, one sentence explanation of the vulnerability.

10. Vulnerability Description: An expanded and more comprehensive explanation of the vulnerability. May provide current and background information. Can be anywhere in 20-100 words in length.

11. Solution: Recommendations to patch the vulnerability. Examples include "Disable SSLv3", or "Contact the software vendor for this application."

3.2 Data Source and Software

The software that generated our data is the Nessus Vulnerability Scanner, version 5.0.1. Nessus is developed and maintained by Tenable Security Solutions.[11] Vulnerability scans are run on the Dartmouth Computing Services data center. Known as the "brains" of an organization, a data center centralizes the storage, management, and dissemination of data that is fundamental to an organization's computer and network operations. It can exist as either a virtual or a physical service.

3.3 Final Data Set

The data went through certain key processing phases to get from the raw source to the final data. In this section, we offer a brief overview of the final data set, and in Section 4 we describe the data parsing process in greater detail.

Table 1 displays the final output from our data pre-processing. Note that the values represent the overall rate of each category across time. They do not represent rates *between* intervals of time, e.g. from scan to scan. Analyzing changes in vulnerability data *between* intervals of time is discussed in Section 6 as future work.

Table 1: Sample of final vulnerability data.

Host	Arrival Rate	Deletion Rate	Disappearance Rate	Reappearance Rate
A	0.529411765	0.529411765	0.529411765	0
B	2.882352941	2.05882353	2.823529412	0.764705882
C	0.352941176	0.352941176	0.352941176	0
D	0.176470588	0	0	0
E	1.411764706	0.294117647	0.294117647	0

In the next section, we overview the methodology behind our data parsing algorithm and statistical analysis.

4. METHODOLOGY

In this section we review the methodology of analysis for our data. In this study, we analyze both the vulnerability arrival process, and the vulnerability deletion process. We begin by outlining the steps for our data pre-processing, and how we get from the raw data to something more manageable that can be analyzed and studied. Finally, we describe our assumptions for the data.

4.1 Data Pre-Processing

Often, a large part of an analysis is in the data pre-processing phase. Depending on the data, it can absorb as much as 50% to 80% of the total time to complete an analysis. Herein, we describe the key processing phases we used for our data.

As we know from the sample data output provided in Figure 1, the Nessus software provides more output attributes than we need for our analysis. The most critical component of our data pre-processing are to accurately store and identify unique vulnerabilities. To introduce some notation, suppose we denote the set of all vulnerabilities as V, then for each unique vulnerability i, we denote the associated set of vulnerabilities as v_i. We then defined a unique vulnerability (v_i) as the set of four attributes, namely: 1.) CVE code; 2.) host name; 3.) port number; and 4.) protocol. These four attributes were selected specifically because the uniqueness of a vulnerability cannot be determined with anything less than these four combined attributes. For instance, not all vulnerabilities in the Nessus output will have a CVE code. So identifying a unique vulnerability by the CVE code alone is not sufficient. Also, each host name can show up in a scan with many different port numbers and/or protocols. Therefore, the only way to identify unique vulnerability (v_i) is to parse the data according to the key combination of a CVE code, host name, port number and protocol. The resulting values are the set of all possible unique vulnerabilities. We then take this set of unique vulnerabilities and compare them from scan to scan to define their status as *new*, *existing*, *disappeared*, or *reappeared*. The final output is a set of files that represent the difference from one scan to the next with the aforementioned labels. For the purposes of our study, we needed rates and therefore further converted the data in counts, then divided the total count by the number

of scans. We summarize the source code for our data parser in Algorithm 1, which outputs the set of unique vulnerabilities as *new*, *disappeared*, or *reappeared*.

Algorithm 1

```
1:
2: status = {}                          ▷ Records if the key is new, already exists or disappears.
3: lastIterDict = {}                    ▷ Records when the key first appears.
4: delHistDict = {}                     ▷ Records when the key disappears.
5:
6: for each unique row do
7:     key = CVE + Host + Protocol + Port
8: if key in lastIterDict then
9:     status = "Exists"
10:    localDict[key] = lastIterDict[key]
11: else if key in delHistDict and key not in status then
12:    status = "Reappear"
13: else
14:    status = "New"
15:    delHistDict[key] = ""
16:    localDict[key] = file
17: for every key value in the status dictionary do
18:    if key not in localDict then
19:        delHistDict[key] = file
20:        localDelDict[key] = row
21:    else
22:        tempstatus[key] = status[key]
23:
24: status = tempstatus
25: lastIterDict = localDict
26: input.next()
```

4.2 Assumptions and Key Relationships

In this section, we overview preliminary assumptions and key relationships within our data.

Each unique vulnerability is defined by a unique key of the CVE code, host, protocol and port. As can be understood from Algorithm 1, the data are parsed in such a way that vulnerabilities are defined as new, already exists, or disappears. As explained below, there is a very critical and distinct difference between vulnerabilities that disappear as opposed to those that are deleted.

From scan to scan, vulnerabilities exist in one of four possible states, namely:

1. New: Vulnerabilities are considered new from one scan to the next if they have not been observed in any previous scans.
2. Disappear: Vulnerabilities can disappear for any myriad of reasons to include servers being decommissioned or taken on and offline for maintenance and upkeep, or even network re-organizations where servers host names an change.

3. Reappear: Vulnerabilities reappear if their unique key has appeared at least once in a scan as a new vulnerability, then, if that unique key has disappeared at any point in time in consecutive scans, and then reappears.

4. Deleted: Vulnerabilities can be considered deleted when they are patched. We collect the set of the overall rate of deleted vulnerabilities by taking the difference between the rate of disappearing vulnerabilities against the rate of reappearing vulnerabilities such that $DeletionRate = DisappearRate - ReappearRate$. Defining this relationship between the rates of vulnerability deletion, disappearance and reappearance requires some simplifications and assumptions. We believe that these simplifications are the best starting point for modeling our data. As our model matures, we could explore methods for incorporating other layers of uncertainty, which is further addressed in Section 6.

4.3 Univariate Analysis

In our univariate analysis, our goal to describe both the arrival and deletion process through distribution models. Our methodology to evaluate and and compare models follows as:

1. Empirical and theoretical densities: We plot the histogram of our empirical data against a theoretical distribution that we believe is the best fit. This is often a good starting point to gain an intuition for the data. The Q-Q plot (described next) offers deeper insight into whether or not the distribution is a good fit.

2. Quartile-quartile plot (Q-Q plot): The Q-Q plot compare the shapes of the histogram and theoretical densities. If the two distributions being compared are similar, the points in the q-q plot will approximately lie on the line y = x. If the distributions are linearly related, the points in the q-q plot will approximately lie on a line, but not necessarily on the line y = x.

3. Empirical and theoretical CDFs: We compare the CDFs of both the empirical and theoretical data. Our goal here is to look for an overall good fit and to minimize the distance between the empirical and theoretical distributions.

4. Probability-probability plot (P-P plot): A P-P plot compares the empirical cumulative distribution function (ecdf) of a variable with a specified theoretical cumulative distribution function. The ecdf is defined as the empirical cumulative distribution function that is derived from the data. Like Q-Q plots and probability plots, P-P plots can be used to determine how well a theoretical distribution models a data distribution. If the theoretical cdf reasonably models the ecdf in all respects, including location and scale, the point pattern on the P-P plot is linear through the origin and has unit slope.

4.4 Bivariate Analysis

Ultimately, our goal is to discover the relationship between the vulnerability birth and death process, i.e. the relationship between the arrival and deletion rates, and explore how it may generalize to other vulnerability data analysis in "the wild."

We analyze trends in the bivariate scatterplot, summarize the underlying statistics and evaluate its usefulness for future models as well as practical applications for analysts and security engineers.

5. RESULTS

In this section, we first overview some basic statistical characterizations and analysis of all four subsets of data (i.e. new, deleted, reappear, and disappear). Then, we dive into a deeper analysis of the set of birth rates of new vulnerabilities and the set of death rates of deleted vulnerabilities. Finally, we assess the dependencies between the birth and death process of the bivariate scatterplot.

5.1 Summary Statistics and Analysis

Recalling the final dataset as displayed in Figure 1, we offer a scatterplot of vulnerability arrivals, deletions and reappearances. We found that 87% of the vulnerabilities disappear for different reasons, to include deletions, servers being taken offline or decommissioned, or scan policy changes. We also find that 73% of the vulnerabilities are deleted through patching, and 14% of the vulnerabilities reappear at a later point in time on the same machine.

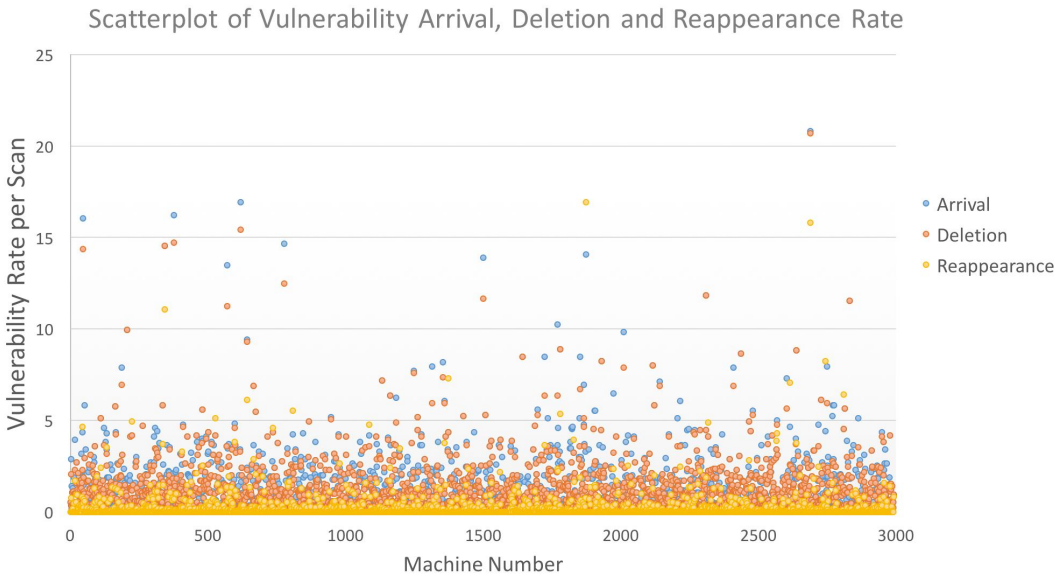

Figure 2: Overview of vulnerability arrival, deletion and reappearance rates.

In the next section, we offer a deeper inspection of the univariate distributions and characterizations.

5.2 Univariate Distributions and Characterization

In this section, we will overview approximations to the distributions of our vulnerability arrival and deletion rate, as well as statistical characterizations. We offer practical interpretations of what the distribution fit means regarding vulnerability management.

5.2.1 Vulnerability Arrivals

We studied vulnerability arrival rates of almost 3,000 machines from 17 reports of scan data over a two-year period. We denote the univariate distribution of arrivals as A, such that for each individual rate i per host, we denote the associated arrival rates per host as a_i.

We find that each host has at least one vulnerability count on at least one scan report; therefore, the final vulnerability arrival rate for each machine was greater than zero. This indicates that no host is immune from the risk of compromise. We also find that the vulnerability arrival data displays characteristics of extreme right skew. Such skewed distributions indicate high counts of low birth rates, and low counts of extremely high birth rates.

Table 2: Summary statistics for the distribution of vulnerability arrival rates.

Mean	Variance	Standard Deviation	Count	
1.28	2.25	1.50	2991	
Minimum	1st Quartile	Median	3rd Quartile	Maximum
0.06	0.35	0.88	1.71	20.82

Interestingly, more than half of the machines (54.90%) have a vulnerability birth rate of less than or equal to one, and almost all of them (97.89%) are less than or equal to five. With a maximum value of 20.82, this tells us that just over 2% of the data (specifically, 2.11%) account for the heavy-tailed value range of (5, 20.82]. Further statistical exploration is summarized in Table 2.

We find that the vulnerability arrival rates satisfy the log-normal distribution. Figure 3 displays four plots to help assess the distribution fit. The comparison of the empirical and theoretical densities of vulnerability arrivals provides validating visual cues to the reliability of the log-normal fit. For deeper insight, we see that the data in the Q-Q plot lie very close to the line with the exception of a few rightmost circles. When we observe the empirical cdf with the theoretical cdf, we see an overall good fit. Finally, the empirical data points follow the line in the P-P plot fairly well. There is a gentle curve in the first half of the data, indicating a mild variance between the empirical and theoretical probabilities, but overall, it is a very good fit.

The log-normal distribution makes intuitive sense when we observe examples from other fields of log-normal models, e.g. virus growth rates in epidemiological studies,[13] and fatigue-stress failures in mechanical structures.[14] Log-normal models often best represent natural phenomena with values that are all greater than zero.

5.2.2 Vulnerability Deletions

Using a similar approach to that employed in Section 5.2.1, we studied vulnerability deletion rates of almost 3,000 machines from 17 reports of scan data over a two-year period. We denote the univariate distribution of deletions as D, such that for each individual rate i per host, we denote the associated deletion rates per host as d_i.

We find that the set of vulnerability deletion rates differs from the set of arrival rates in that just over 22% of the machines have exactly zero deleted vulnerabilities over the two-year period. This indicates that almost a third of the machines have vulnerabilities which are not being resolved in any way. Similar to the vulnerability arrivals, the vulnerability deletion rate data is extremely

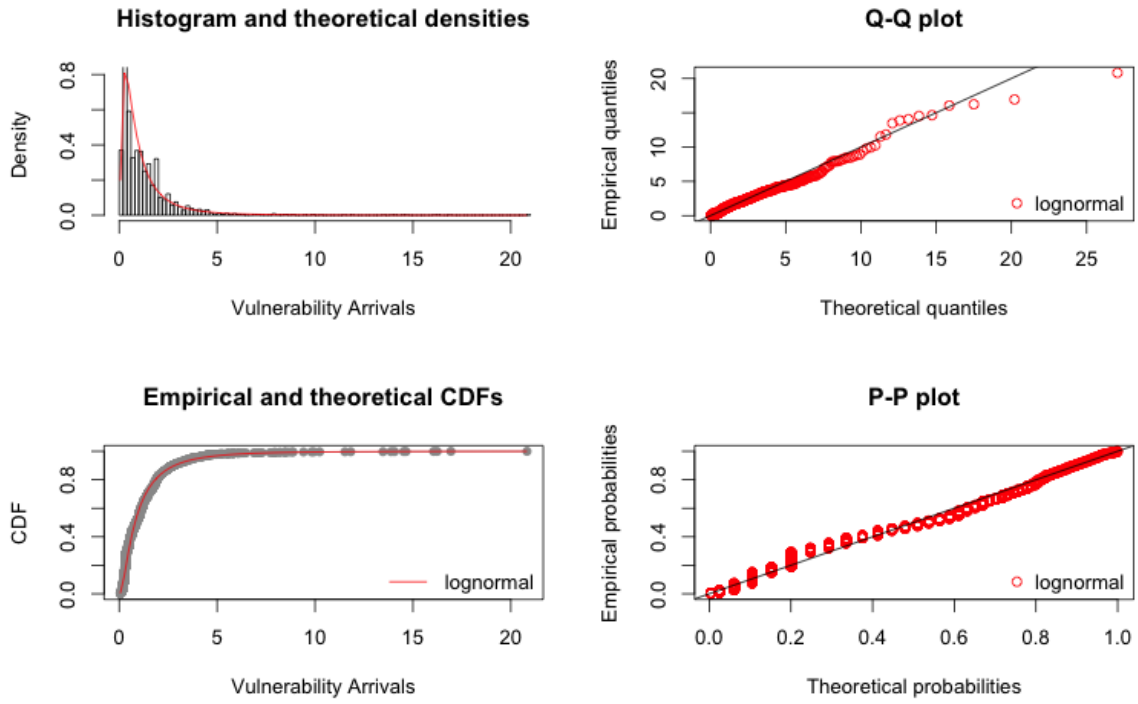

Figure 3: The log-normal distribution provides a possible model for the vulnerability arrival process.

right skewed. In the context of deleted vulnerability rates, one cause of the skewness could be from so-called "start-up effects", such as that which we might find in reliability analysis. Due to many different possible causes, there may be a large amount of "failures" in the beginning of the vulnerability remediation process.

Not surprisingly, significantly more than half of the machines (67.20%) have a vulnerability death rate of less than or equal to one, and almost all of them (98.36%) are less than or equal to five. With a maximum value of 20.71, this tells us that less than 2% of the data (specifically, 1.64%) account for the heavy-tailed value range of (5, 20.71]. Further statistical explorations are summarized in Table 3.

Table 3: Summary statistics for distribution of deleted vulnerability rates.

Mean	Variance	Standard Deviation	Count	
0.94	1.91	1.38	2991	
Minimum	1st Quartile	Median	3rd Quartile	Maximum
0.00	0.12	0.53	1.24	20.71

We also find that the set of deleted vulnerabilities can be modeled with the exponential distribution. Figure 4 displays four plots to help assess the best distribution fit. We find that the empirical and theoretical densities fit almost exactly. The Q-Q plot exhibits a slight upward bend to the upper left of the line, indicating that there may be a little bit more activity to the underlying process than what we can currently capture[†]. The P-P plot also displays some slight deviations in the lower left region; however, overall we find that the exponential distribution's probability values are very similar to that of the empirical.

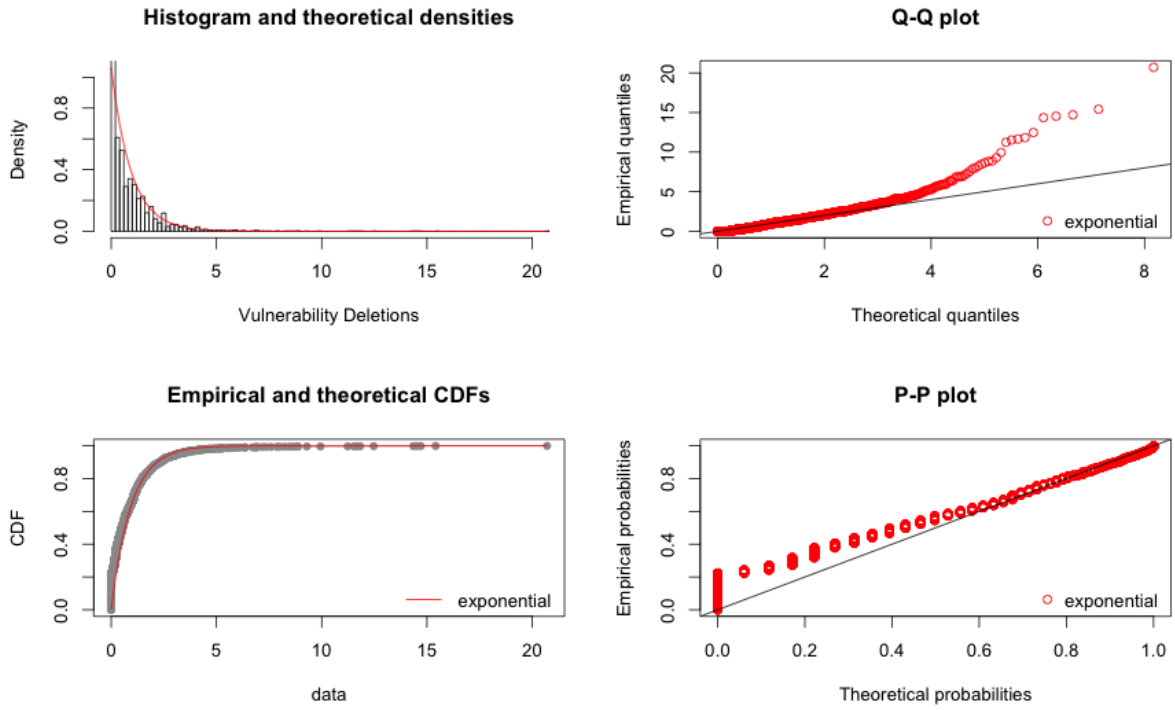

Figure 4: The exponential distribution provides a possible model for the vulnerability deletion process.

We suspect that the reason why the exponential distribution fit the best is primarily due to the large number of zeros in the data. As we outline in Section 6, future work is to explore how long vulnerabilities persist over time and how their arrival and deletion rates change *in between* scans. Depending on the outcome, the vulnerability lifeline could be investigated as a reliability theory problem.

5.3 Bivariate Relationship

In this section, we aim to specify the relationship through the bivariate scatterplot and correlation statistic. We know that vulnerability deletions have some level of dependency on vulnerability

[†]Further commentary on this matter can be found in "Future Work" (Section 6).

arrivals[‡]. A natural choice for terminology when modeling this dependency is birth rates (for vulnerability arrivals) and death rates (for vulnerability deletions). Note that we use this language interchangeably throughout this section of the analysis.

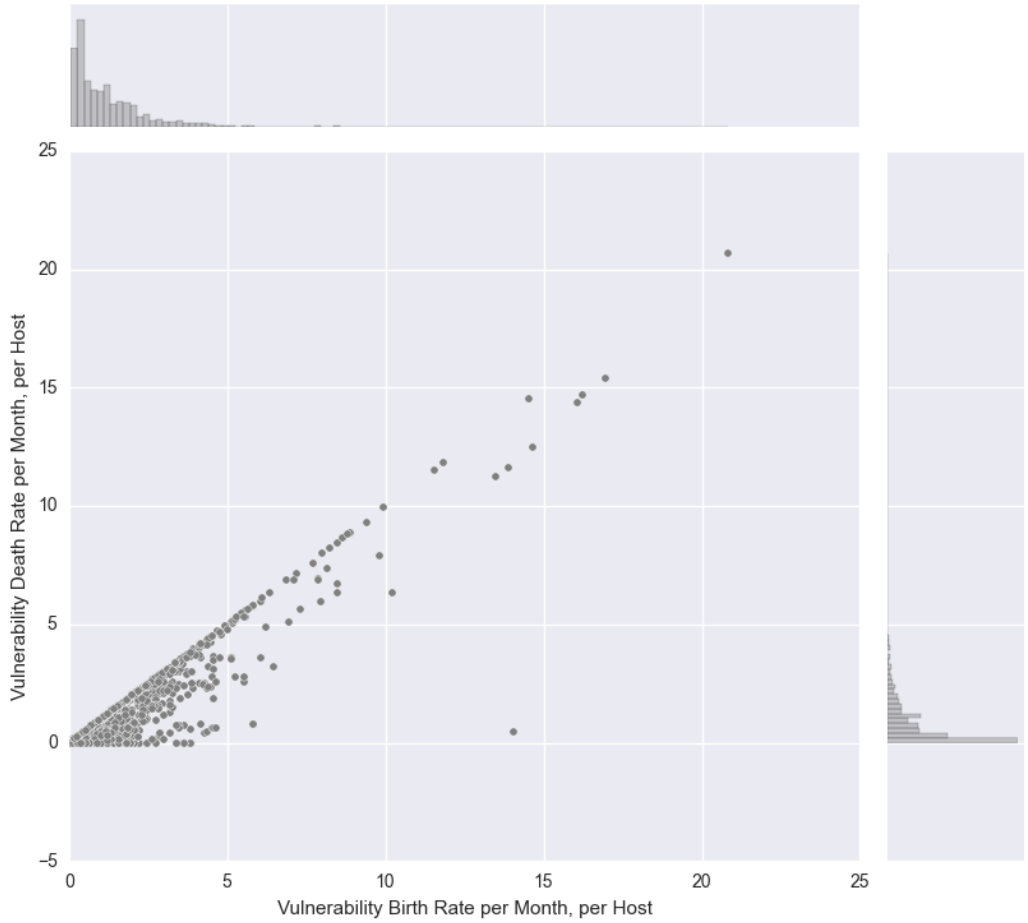

Figure 5: Bivariate scatterplot of vulnerability birth and death rates.

Table 4: Spearman correlation statistic on bivariate relationship between birth and death rates.

Spearman Correlation
0.76

[‡]See Section 4.2.

In Figure 5, we see a strong positive relationship between the vulnerability birth and death rates. Further evidence of correlation is indicated in Table 4, where the Spearman correlation statistic is positive and fairly close to one. We include the marginal (univariate) distributions on the outer edges of the x and y-axes of the bivariate plot to aide further analysis.

We find that the best case scenario is when vulnerability death rates are equal to the vulnerability birth rates. The points where the vulnerability birth rate and the death rate meet are considered points of equilibrium. Notice that there are some data points that deviate far from the x=y line. These data indicate machines with vulnerability birth rates much greater than their death rates.

What is so useful about this bivariate plot is that it not only provides the basis for models and the joint probability distribution, but it can also provide immediate feedback to analysts and security engineers on the current status of their vulnerability management. The more the data clusters *on* the x-y line, we know that the more aggressively machines were being patched over the 24-month period. The more the data points cluster *below* the x-y line, the less aggressively vulnerabilities are being patched, and the greater their need to be serviced. In conclusion, the bivariate scatterplot is useful for providing insight into the history of vulnerability management. They can also feed into predictive analysis for future vulnerability metrics based on past network behaviors.

5.4 Challenges to Internal and External Validity

In this section, we review a number of identified challenges to internal and external validity. Recognizing and mitigating these challenges are critical to both generating and processing the data that produces reliable results and thorough scientific analysis. With that being said, we recognize that there are some challenges to the internal and external validity in our study. Examples include, but are not limited to the rate of false-positives or false-negatives in the Nessus software. Sections 5.4.1, and 5.4.2 provide a thorough overview of these challenges.

5.4.1 Internal Validity

Internal validity refers to how well an experiment was conducted. Factors that support good internal validity are having as few confounding variables as possible. As the number of independent variables acting at the same time increases, so do the challenges to internal validity.

Challenges to internal validity for our study include:

Deletion Rate Assumptions We defined deletion rate as the difference between the set of rates of disappearing vulnerabilities and the set of rates of reappearing vulnerabilities. There could be other confounding variables within the relationship between disappearing and reappearing vulnerabilities. With that said, we also do not want to overfit the data, and believe our the accuracy is fairly high in calculating the set of vulnerability deletion rates.

Servers Taken Offline Servers can temporarily be taken offline or decommissioned for various reasons. It may be the case that a system has to be upgraded, or access policies change. Temporary safeguards may need to be imposed if a "bug" is found on a database. In which case, the server is taken offline to prevent further damage until it can be investigated later.

Scan Policy Changes An enterprise-level network undergoes periodic changes, such as permanently adding and deleting servers. When this happens, host names either need to be added or deleted from the list of vulnerability scans. There were three scan policy changes within the two years of vulnerability data.

Data Gaps Six of the 24 months are missing scan results. We compared the outcome of the data with various levels of assumptions for the missing months and found that there was no optimal strategy to bridge the missing months for this particular data set. In order to avoid introducing artifacts into the data or unintentionally skewing the results, we decided to treat the data as "scan-to-scan", rather than "month-to-month" (i.e. We left the missing months alone and treated our data as a set of 17 scans across a 24-month time span.).

Software Flaws Problems in the software of the Nessus Scanner itself can contribute to imperfect data. There is a body of research aiming to address vulnerability software false-positives and false-negatives. We direct the curious reader to References 3, 15, and 16.

5.4.2 External Validity

External validity refers to how well scientific results generalize to similar circumstances. We know that the less a scientific study generalizes to other studies, then the greater the challenges to external validity.

Challenges to external validity for our study include:

Data Perhaps the biggest challenge to external validity is the data. We are working with operational data from one source. It is ideal to have data from multiple sources. As this work is part of a preliminary study, we address expanding our analysis to more data sources in Section 6. As an effort to minimize this challenge, this is in part why we made the assumptions and generalizations found in Section 4.2 to avoid overfitting the data.

6. FUTURE WORK

This paper focused on the preliminary work of developing a process for parsing large sets of vulnerability data and exploring the statistical characterizations of the arrival and deletion process in an operational setting. Our study is unique in that it is the first to source vulnerability data from "the wild," and, with that comes a vast expanse of security research questions that can further aide the vulnerability mitigation and management process. Specifically, we pose the following research questions: *Can we quantify the operational vulnerability lifeline? What are the net expected costs to an organization based on their current vulnerability patching process? How can we measure or model risk for more efficient vulnerability triage?*

To address these questions, we suggest the following future work as an outcome of our results:

- Source a larger set of operational IT vulnerability scan data to better estimate the current state of vulnerability patching in the "wild";
- Incorporate the attribute time (t) into our model through measuring both the length of time a vulnerability persists, as well as changes in the birth-death process *between* intervals of time;

- Incorporate random variables time (t), vulnerability arrivals (a) and vulnerability deletions (d) into a risk model for improved vulnerability triage;
- Estimate net expected costs to an organization based on how vulnerabilities are managed in real time.

7. CONCLUSION

In conclusion, this paper offered an overview of the need for studying vulnerability data that is sourced from the operational IT environment. It offers a data parsing algorithm for wrangling in large and unruly vulnerability datasets. We then provided an in depth univariate and bivariate analysis that provides insight into the relationship between the vulnerability arrival and deletion process. The distributions we used to characterize the univariate variables can be used as prior distributions for future studies such as Bayesian predictions on vulnerability management and mitigation. Future work includes: 1.) Generalizing and automating our vulnerability parsing algorithm for use in any environment, from academic to operational; 2.) Quantifying the random variable time, (t), through measuring vulnerabilities in terms of both their persistence *across* time as well as changes *between* intervals of time; and, 3.) Devising a risk calculus based on the three random variables arrival (a), deletion (d), and time (t). The results of our study also lends itself naturally to informing a myriad of risk assessments and actuarial studies for fields such as cyber-insurance.

ACKNOWLEDGMENTS

This work was supported by the ARO MURI grant W911NF-13-1-0421. The authors are grateful to the DOD SMART Scholarship for student financial support, as well as Ben Priest for offering feedback, and Dartmouth Computing Services for providing the vulnerability data.

REFERENCES

[1] Allodi, L. and Massacci, F., "Comparing vulnerability severity and exploits using case-control studies," *ACM Transactions on Information and System Security (TISSEC)* **17**(1), 1 (2014).

[2] Joh, H. and Malaiya, Y. K., "A framework for software security risk evaluation using the vulnerability lifecycle and cvss metrics," in [*Proc. International Workshop on Risk and Trust in Extended Enterprises*], 430–434 (2010).

[3] Holm, H., Ekstedt, M., and Andersson, D., "Empirical analysis of system-level vulnerability metrics through actual attacks," *Dependable and Secure Computing, IEEE Transactions on* **9**(6), 825–837 (2012).

[4] Clark, K., Dawkins, J., and Hale, J., "Security risk metrics: Fusing enterprise objectives and vulnerabilities," in [*Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*], 388–393, IEEE (2005).

[5] Georg, G., Houmb, S. H., and Ray, I., "Aspect-oriented risk driven development of secure applications," in [*Data and Applications Security XX*], 282–296, Springer (2006).

[6] Carin, L., Cybenko, G., and Hughes, J., "Cybersecurity strategies: The queries methodology," *Computer* **41**(8), 20–26 (2008).

[7] Meier, J., Mackman, A., Dunner, M., Vasireddy, S., Escamilla, R., and Murukan, A., [*Improving web application security: threats and countermeasures*], Microsoft Redmond, WA (2003).

[8] Wynn, J., Whitmore, J., Upton, G., Spriggs, L., McKinnon, D., McInnes, R., Graubart, R., and Clausen, L., "Threat assessment and remediation process," Tech. Rep. Project No. W15P7T-10-C-F600, MITRE (October, 2011).

[9] Buckshaw, D. L., Parnell, G. S., Unkenholz, W. L., Parks, D. L., Wallner, J. M., and Saydjari, O. S., "Mission oriented risk and design analysis of critical information systems," *Military Operations Research* **10**(2), 19–38 (2005).

[10] Artz, M. L., *Netspa: A network security planning architecture*, PhD thesis, Massachusetts Institute of Technology (2002).

[11] Tenable Network Security, Nessus Vulnerability Scanner https://www.tenable.com/products/nessus-vulnerability-scanner. (Accessed: 18 March 2016).

[12] "New Nessus Feature Added: CSV Export." Tenable Network Security, https://www.tenable.com/blog/new-nessus-feature-added-csv-export. (Accessed: 18 March 2016).

[13] Buckland, S. T., Burnham, K. P., and Augustin, N. H., "Model selection: an integral part of inference," *Biometrics* , 603–618 (1997).

[14] Cruse, T. A., [*Reliability-based mechanical design*], vol. 108, CRC Press (1997).

[15] Holm, H., Sommestad, T., Almroth, J., and Persson, M., "A quantitative evaluation of vulnerability scanning," *Information Management & Computer Security* **19**(4), 231–247 (2011).

[16] Doupé, A., Cavedon, L., Kruegel, C., and Vigna, G., "Enemy of the state: A state-aware black-box web vulnerability scanner.," in [*USENIX Security Symposium*], 523–538 (2012).