March 2017

Visualizing Contextual Information for Network Vulnerability Management

Andrew Jack Mokotoff
Worcester Polytechnic Institute

Barrett Mitchell Wolfson
Worcester Polytechnic Institute

Zachary Reid Robbins
Worcester Polytechnic Institute

Visualizing Contextual Information for Network Vulnerability Management

Major Qualifying Project

Advisor:

PROFESSOR L. HARRISON

Written By:

ANDREW MOKOTOFF
ZACHARY ROBBINS
BARRETT WOLFSON

A Major Qualifying Project Report submitted to the Faculty of the WORCESTER POLYTECHNIC INSTITUTE in partial fulfillment of the requirements for the Degree of Bachelor of Science in Computer Science.

DATE: MARCH 24TH, 2017

ABSTRACT

The threat of data breach rises every day, and many organizations lack the resources to patch every vulnerability they might have. Yet, these organizations do not prioritize what vulnerabilities to patch in an optimal way, in part due to a lack of context needed to make these decisions. Our team proposes the Vulnerability Visualization (VV) tool, a web visualization dashboard for increasing analyst prioritization capabilities through visualization of context for network scans. Evaluations demonstrate that the VV tool enhances the vulnerability management (VM) process through augmenting the discovery and prioritization of vulnerabilities. We show that adding context to the VM process through visualization allows people to make better decisions for vulnerability remediation.

TABLE OF CONTENTS

List of Tables

Table **Page**

List of Figures

Figure **Page**

CHAPTER 1

INTRODUCTION

The world of information security is the setting of intense warfare, one where threats move quickly, stealthily, and intelligently. Breaches in the information security sector are a constant threat faced by organizations of all varieties, regardless of size or industry. With technology advancing at a blistering pace, those with malicious intent have an ever expanding arsenal with a potential only limited by the creativity of the wielder. With such unpredictable enemies, those tasked with defending from breaches can often only make best guesses on how to approach zero-day threats that have never been exploited before. Most importantly, the internet's increasingly cheap and widespread access opens up new channels for infiltration into an ever-growing number of vulnerable systems.

1.1 Analyzing Modern Security Trends

One of the best windows into the overall state of the information security sector is the Verizon Data Breach Investigations Report (DBIR), an annual summary of the major trends, motivations, and demographics surrounding the data breaches of that year [1]. The 2016 report offers insights from big data, and is an invaluable tool in characterizing the threats organizations should expect to face. For instance, breaches driven by a financial or espionage motive constituted 89% of all breaches in 2016. These two motivations can be found alongside a few others in Figure 1.1, where we can see that financial motivation is unsurprisingly far above the rest.

As more companies move sensitive information to modern servers and databases for scalability and efficiency, it can open the door to data breaches. The public sector, financial, and information industries continue to be the most heavily affected by security incidents which resulted in confirmed data loss. Table 1.1 shows the number of security incidents with confirmed data loss by victim industry and organization size according to the 2016 Verizon DBIR. While there are

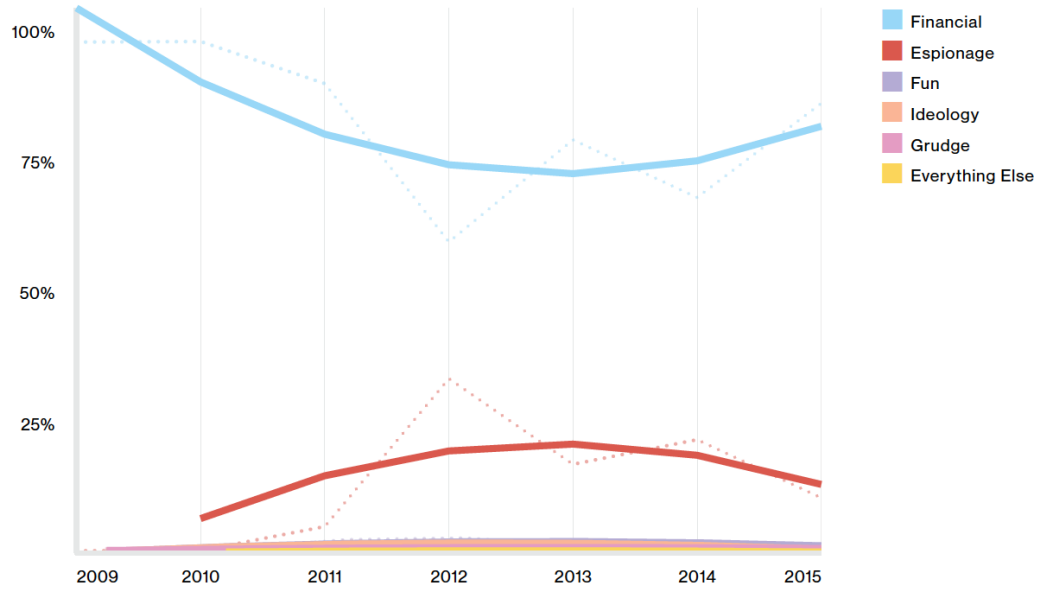

Figure 1.1: *The percent of breaches per threat actor motive over time [1]. While there was a steep dip in 2012 in favor of espionage, finances has been the biggest motive of breaches in the last few years. Despite how much technology has evolved over the years, the motivations behind breaches have remained relatively constant.*

certainly industries that are inherently more at risk to falling prey to a data breach, this data shows that the problem of malicious data acquisition is a far-reaching one.

The cost of data breaches is one of the biggest motivating factors to bolstering one's defenses against them. The IBM 2016 Cost of Data Breach study reveals trends that match up with Verizon's report, and further show the widespread problem malicious exploitation poses. IBM's 2016 study sampled 383 companies from 12 countries, finding that the average cost of data breach has raised 29% since 2013 to $4 million [4]. Over the years this report has been conducted, a few major trends have been identified:

1. The cost of data breach is a relatively constant cost that organizations must be prepared to deal with
2. The biggest financial consequence from a data breach comes from lost business and customer trust
3. The longer it takes to identify and resolve a data breach, the higher the cost it incurs
4. Improving data governance practices helps reduce the cost of breach
5. Investing in data loss prevention services like endpoint security and encryption helps prevent breaches

Industry	Total	Small	Large	Unknown
Accommodation (72)	282	136	10	136
Administrative (56)	18	6	2	10
Agriculture (11)	1	0	0	1
Construction (23)	4	0	1	3
Educational (61)	29	3	8	18
Entertainment (71)	38	18	1	19
Finance (52)	795	14	94	687
Healthcare (62)	115	18	20	77
Information (51)	194	12	12	170
Management (55)	0	0	0	0
Manufacturing (31-33)	37	5	11	21
Mining (21)	7	0	6	1
Other Services (81)	11	5	2	4
Professional (54)	53	10	4	39
Public (92)	193	4	122	67
Real Estate (53)	5	3	0	2
Retail (44-45)	182	101	14	67
Trade (42)	4	2	2	0
Transportation (48-49)	15	1	3	11
Utilities (22)	7	0	0	7
Unknown	270	109	0	161
Total	2,260	447	312	1501

Small = organizations with fewer than 1,000 employees, Large = organizations with 1,001+ employees.

Table 1.1: *Number of security incidents with confirmed data loss in 2016. Incidents are broken down by industry type and size of organization [1]. As can be seen, finance is much more likely to be targeted than other organization types.*

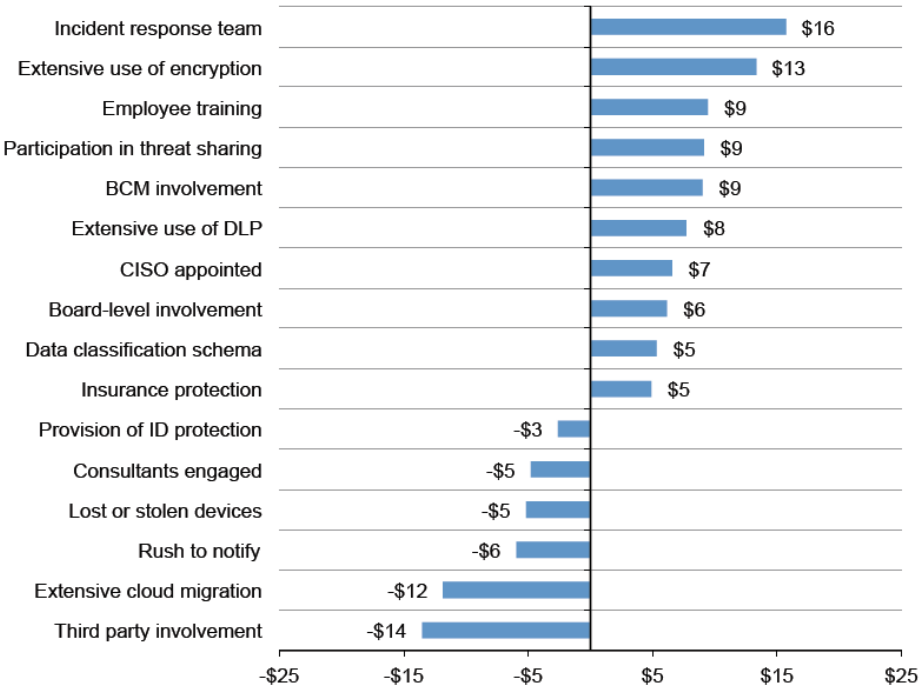

Figure 1.2: *The impact of 16 factors on the per capita cost of data breach (measured in US dollars) [4]. Positive values mean a reduction of cost is the result of the factor being active in a data breach situation. Therefore, factors like having an incident response team mitigate the cost of a data breach, whereas a lost device makes a data breach cost more.*

In Figure 1.2, we see what factors organizations should concentrate on bolstering or mitigating to reduce the cost of a breach should one occur. For instance, the best ways to reduce breach cost include a dedicated response team, dedicated encryption use, and employee training, whereas data breaches become more of a financial threat when third parties become involved, cloud migration is overdone, or devices are lost/stolen. Unsurprisingly, there isn't a single solution to nullifying the potential impact of a breach, but rather a myriad of factors that must be balanced based on an organization's needs and resources.

A facet of data breaches that cannot be overlooked is the rapid timeline over which they occur. Unfortunately, once a breach occurs data exfiltration happens extremely quickly, often before vulnerability analysts even detect the breach. In fact, the deficit between time to compromise and time to analyst discovery has been consistently growing over the past decade [1]. On average, compromise of a system almost always happens in days or less (if not minutes), and exfiltration of the compromised data happens within days after compromise. In contrast, identification and containment of a breach took on average two orders of magnitude more time to complete [4]. In many cases, a successful infiltrator will extract all the information they desire long before a breach is even suspected by the victim. This data is shown in Figures 1.3 and 1.4.

Both Verizon and IBM's broad impact studies lead towards the same conclusion: data breaches

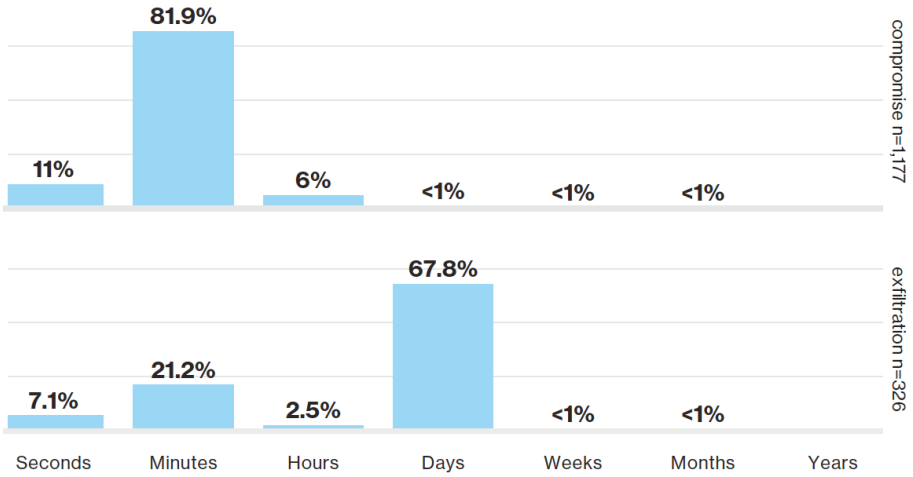

Figure 1.3: *Time to compromise and exfiltration during a data breach [1]. In about 93% of data breaches, compromise of the targeted system happened in minutes or less. Additionally, in nearly all cases, exfiltration of targeted data occurred in days or less.*

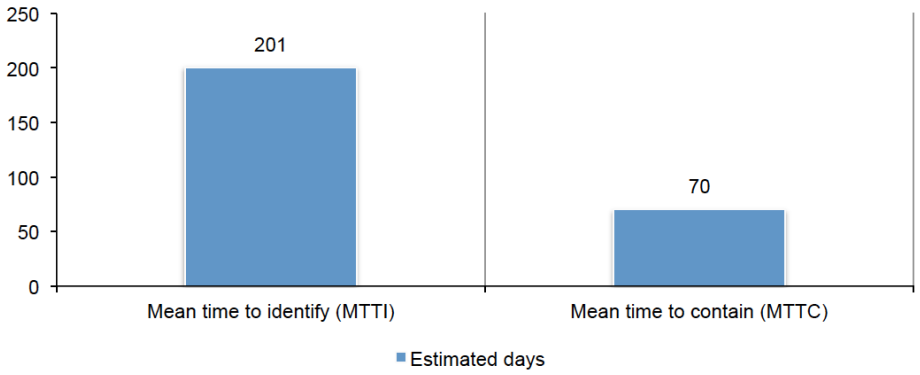

Figure 1.4: *Mean time to identify and contain data breach incidents [4]. On average, most data breach incidents remain undiscovered for 201 days, and once discovered, take 70 days to contain/remediate. Considering almost all exfiltration happens in under a few days (Figure 1.3), this paints a bleak picture for current vulnerability management tactics.*

are a constant threat that will always remain, and preventing data breach is preferable to remediating one. It is therefore in the interest of companies/organizations to follow security best practices, train their employees, and use the latest defensive technologies to preemptively stop breaches before they occur. One of the best defenses an organization can provide is a strong vulnerability management protocol, in which a team of dedicated analysts diligently monitor and patch any security holes on the network. This is far preferable to something such as remediation after a breach, which will cost significantly more in resources and lost business than preventing the breach in the first place. Thus, in the next section we will delve into what exactly constitutes an effective vulnerability management protocol.

1.2 Defining Effective Vulnerability Management

Preventing data breaches often relies on effective vulnerability management, in which potential holes in system security are identified, and then remediated to prevent exploitation. A vulnerability by definition is "a weakness of an asset or group of assets that can be exploited by one or more threats" [9]. Vulnerability scanning is a subtask of vulnerability management where a computer program is used to scan a system or network for known vulnerabilities. Organizations often use these scanners, but they are only part of an effective vulnerability management process, and should not be used exclusively. There is no one consensus among researchers what the ideal vulnerability management process should look like, so we chose to define an effective process as a mix between a number of well-known definitions. Our working definition of an effective vulnerability process is as follows [10][11][12][13]:

Vulnerability Management Process

1. Preparation (define scope of assets to protect and threats anticipated)
2. Scan for Vulnerabilities (typically using automated vulnerability scanners like Tenable's Nessus)
3. Identification and Prioritization (review the results of the scan(s) and categorize/rank the vulnerabilities discovered, deciding which to tackle first)
4. Remediation (implement the patch fix over an agreeable timeframe)
5. Rescan (restart cycle, scan again to confirm vulnerabilities are successfully neutralized and check for new ones)

The very nature of information security means that security analysts are always on the defensive, and closing every potential hole is by no means an easy task. Many analysts and information security industry professionals regard vulnerability management as both an extremely laborious and frustrating task. The number of vulnerabilities almost always overwhelms

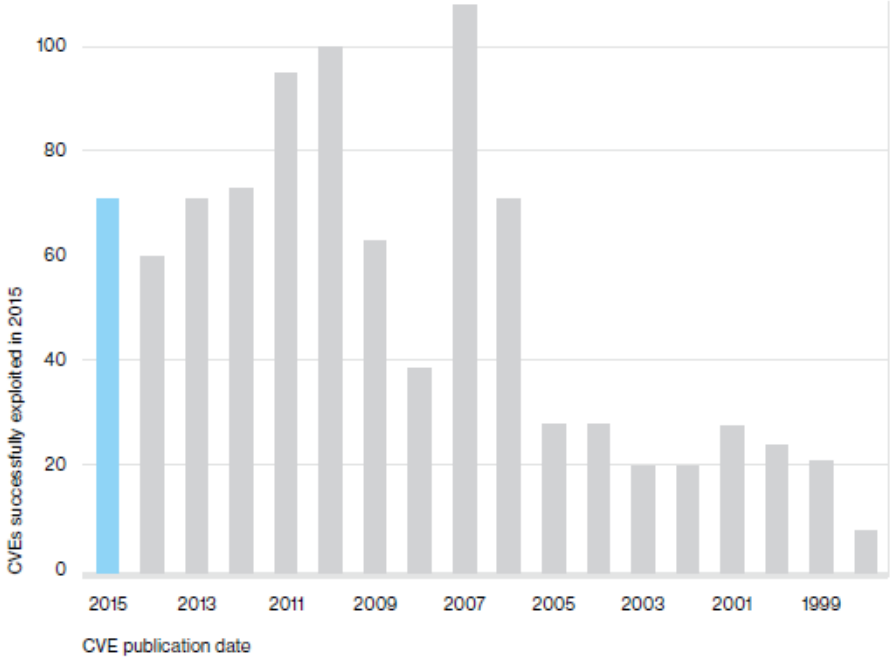

Figure 1.5: *Graph of CVEs successfully exploited in 2015 [1]. The vast majority of successfully exploited vulnerabilities are from years before 2015, with most occurring about 4-8 years prior. Clearly, focusing on only the most recently discovered vulnerabilities is an ineffectual vulnerability management strategy by itself.*

the resources available to mitigate them, and with such a wide array to tackle, many security analysts default to tackling the biggest and newest threats they find. Unfortunately, this leaves many systems wide open to exploitation from smaller legacy vulnerabilities that flew under the radar. This pattern is displayed in Figure 1.5, which shows the count of Common Vulnerabilities and Exposures (CVE) exploited in 2015 by CVE publication date [1].

According to the Verizon DBIR, a cohesive patching strategy which emphasizes coverage at the expense of speed is recommended. This isn't to say that quickly implementing patches isn't important, however. The report also states that "Half of all exploitations happen between 10 and 100 days after the vulnerability is published, with the median around 30 days. This provides us with some general guidelines on which software vulnerabilities to prioritize along with some guidance on time-to-patch targets" [1]. In addition, there are a relatively low amount of vulnerabilities that hackers tend to exploit. There are ten top vulnerabilities accounting for 85% of successful exploit traffic. These are extremely high priority, and then the other 15% consists of over 900 other distinct vulnerabilities. This provides a decent starting point for organizations who many not have a robust vulnerability management process to vastly decrease their chances of being targeted and breached.

We have established that vulnerability management is a viable solution to controlling and

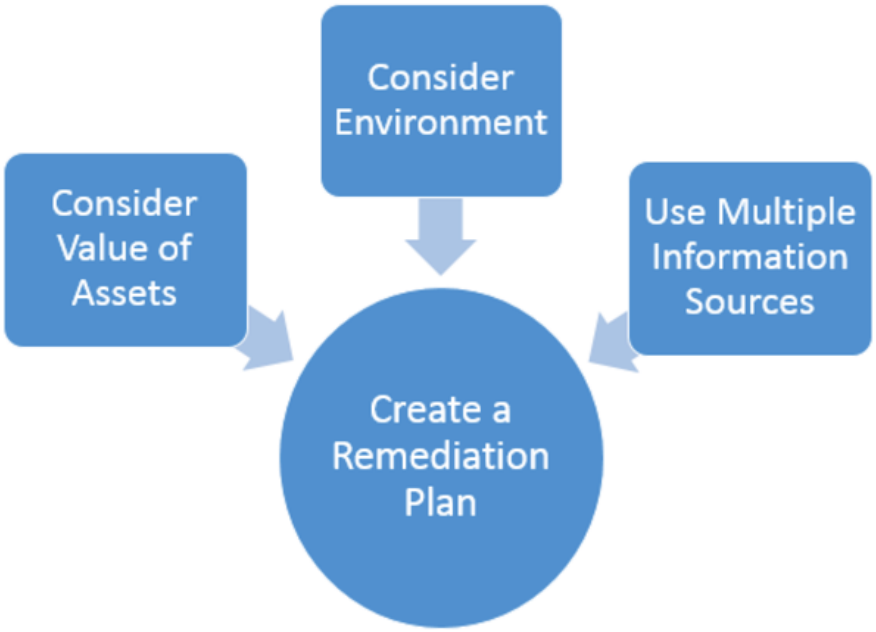

Figure 1.6: *Diagram of effective prioritization. While prioritization plans differ from organization to organization, generally they all aim to create a remediation plan through consideration of situational factors and multiple information sources.*

preempting security breaches before they can occur. The question is, how do we address how to best utilize what are often limited resources for vulnerability management? It boils down to one overarching issue: people do not prioritize what they patch in an optimal way. Many organizations fall prey to the natural instinct to patch as many things as possible as quickly as possible. Not only is this a bad approach to vulnerability management, it is often infeasible, as there are simply too many vulnerabilities and too few resources to handle them [14][15][16][17]. Effective prioritization requires good situational awareness, knowing the value of your assets, thinking like an attacker, multiple tools and information sources, and frequent vulnerability scanning. CVSS alone is not enough to determine priority [14][16]! Imagine there is a server with a "critical" vulnerability on it, but it's not connected to a network. There is no threat exposure in this case. However, a low criticality vulnerability exposed to the internet, and that took a backseat to this critical server vulnerability, could be the undoing of your assets. Figure 1.6 shows what effective prioritization looks like.

As a result of our research and tool development, we reach the conclusion that effective prioritization needs **context**. Context includes information about your assets, your threat exposure, your environment, and many other variables that vary drastically for each organization/company/entity. Unfortunately, many of the tools currently available to vulnerability analysts for scanning and management lack the context needed to allow for correct prioritization. Thus,

in this project our team's goal was to consolidate multiple sources of contextual information regarding vulnerabilities into a single view, allowing analysts to quickly sift through greater amounts of meaningful information to improve their patching process. As a result of our work, we were able to accomplish the following:

- Gathered valuable industry opinions on the shortcomings of modern vulnerability management software and desired qualities in ideal vulnerability management software.
- Confirmed that the need for context for effective prioritization is one of the biggest challenges facing the vulnerability management field.
- Created a modern web application tool - the Vulnerability Visualization (VV) tool - using React + D3 for vulnerability analysts, based on a proven-effective existing tool, Nessus Visualization (NV).

CHAPTER 2

BACKGROUND

Research in the space of computer security has become increasingly robust as society continues to ramp up its reliance on digital information. The risk of network vulnerabilities from a security standpoint has lead to a substantial amount of research on the topic of enterprise vulnerability management.

In 2005, one of the most reputable technological leaders of the United States, the National Institute of Standards and Technology (NIST), created a step-by-step process for integrating an effective security patching and vulnerability management process into any organization [18]. This is by no means a simple feat, and quickly after, many companies began designing automated vulnerability scanners in order to make the vulnerability management process less strenuous on the time and resources of organizations. Companies such as Rapid7, Nessus, Qualys, and many others all design network scanners with this capability.

Additionally, there has also been research conducted to evaluate the effectiveness of vulnerability scanning [2]. Automated scanners have received criticism for not capturing the entire picture or context of a computer network. Some research has begun to emerge targeting the contextual risk of vulnerabilities as well as prioritization in the patching process [19]. It's possible this research will be able to target some of the weaknesses in automated scanning.

Even data visualization has been applied to this problem. However, the security visualization tools developed have mostly focused on a specific subset of the security picture such as port-based events [20], netflow [21], attack graphs [22], and many others [23].

Our project attempts to apply a broader data visualization approach to compensate for the shortcomings of automated scanners. We hope this will allow security analysts to begin to better understand the state of the networks they are protecting and draw more meaningful intuition about patch solutions.

2.1 Vulnerability Identification

2.1.1 Analysis of Currently Used Tools

There are a plethora of vulnerability scanners available to security analysts that make vulnerability management a lot less daunting. Vulnerability scanners essentially scan and traverse a network and automatically search for vulnerabilities in the network. Once these scanners find a vulnerability they will report back information about the vulnerability, including IP address, port, operating system, and other basic information about the source. Once the scanner finds and reports these vulnerabilities, it will compare the information it found with third-party databases including **CVE**[1], **OVAL**[2], **OSVDB**[3], and more.

2.1.1.1 Tenable Nessus

Tennable Nessus is the worlds most popular vulnerability scanner. In 2005, Tennable reported that it was used by over 75,000 organizations worldwide. Nessus scans for the following type of vulnerabilities:

1. Vulnerabilities that allow a remote hacker to gain access to sensitive data on a system
2. Misconfigurations in systems, including missing patches, open mail relays, and etc.
3. Checking for default and weak passwords
4. Denial of Service attacks by using malformed packets
5. Preparation for **PCI DSS**[4] audits

The results of the scan can be reported in various formats. They include plain text, XML, HTML, and LaTeX. Tennable also performs vulnerability checks daily. This allows the scanner to constantly have the updated vulnerabilities that have been released.

2.1.1.2 QualysGuard

Qualys is one of the first vulnerability scanners to enter the market. They were founded in 1999, and successfully launched QualysGuard in 2000. Qualys now has multiple products that are available for organizations to purchase. One of them is called the QualysGuard Intranet

[1]**CVE:** Common Vulnerabilities and Exposures, a dictionary of common names for publicly known information security vulnerabilities.

[2]**OVAL:** Open Vulnerability and Assessment Language, a community effort to standardize how to assess and report upon the machine state of a computer.

[3]**OSVDB:** Open Sourced Vulnerability Database, a database to provide accurate, detailed, current, and unbiased technical information on security vulnerabilities.

[4]**PCI DSS:** Payment Card Industry Data Security Standard, a proprietary information security standard for organizations that handle branded credit cards from major card schemes.

Scanner which is a product that automatically scans corporate LANs for vulnerabilities and searches for available patches. In 2008, Qualys introduced Qualys Guard Policy Compliance, which allowed the scanner to collect IT compliance data across an organization to be linked to policies to document compliance for auditing. Finally, in 2010, Qualys introduced Qualys Browser Check which is a service that scans web browsers and plugins for potential vulnerabilities.

2.1.1.3 Rapid7 Nexpose

Nexpose is Rapid7's vulnerability management product. Nexpose has a large amount of interesting features. One of the features is Advanced Exposure Analytics, which essentially finds vulnerabilities and prioritizes them based on what will be exploited first, so you can avoid the list of stale alerts. In addition, it has live monitoring of exposures, which essentially is an automatic alerts to changes and exposures to to the network, giving it a live view into the vulnerabilities as they happen. Nexpose Adaptive Security automatically detects new devices as they enter the network and identifies which devices have critical vulnerabilities as soon as they are released. Furthermore, it has liveboards which allow the analyst to see the score from compliance to progress. It takes exposure data and translates it into detailed visualizations so it is easily viewable. Finally, it is integrated with **Metasploit**[5] for exploiting vulnerabilities to help determine priorities and for testing purposes.

2.1.1.4 Core Impact

Core Impact is another vulnerability management solution for assessing and testing security vulnerabilities throughout a network. This product has multi-vector testing which allows you to replicate attacks that traverse across systems, devices, and applications which can allow the analysts to understand how chains of exploitable vulnerabilities open paths to secure systems. It also claims to test more common vulnerability exploits than the competition. Finally, you can perform complex attacks with a simple interface. It gathers network information and performs attacks to test the systems ability to identify and remediate.

2.1.1.5 SAINT

Security Administrator's Integrated Network Tool (SAINT) is another product used for scanning vulnerabilities in a network. The SAINT scanner investigates every live system on a network. For all the systems running, it launches probes designed to detect anything that could allow an attacker to gain unauthorized access, create a DOS attack, or gain sensitive information about the network. The SAINT scan operates in four steps:

1. SAINT screens all live systems for TCP and UDP services

[5] **Metasploit:** A tool for developing and executing exploit code against a remote target machine.

2. For each live system, probes are launched to detect any malicious activity

3. Scanner checks for vulnerabilities

4. When vulnerabilities are detected, the results are categorized in several ways, allowing customers to target the data they find most useful.

SAINT groups vulnerabilities according to severity, type, or count. It describes each vulnerability by referencing CVE, CERT advisories, and IAVA. This allows them to describe ways to correct the vulnerabilities.

2.1.1.6 OpenVAS

The Open Vulnerability Assessment System (OpenVAS) is a framework of services and tools offering a comprehensive and powerful vulnerability scanning and vulnerability management solution.

2.1.2 More on Limitations of Current Tools

As seen above, there are many vulnerability identification tools already in use. The following lays out some of the most commonly used scanners on the market today and compares them using qualitative properties such as ability to detect software flaws, ability to detect configuration errors, ability to detect vulnerabilities, ability to perform network sniffing, ability to validate discovered vulnerabilities against possible exploits, and a few others [2]. Table 2.1 gives a side-by-side comparison of the top scanning tools on the market, and how they stack up against each other.

Despite this imperfect picture, it's extremely important for the output from a vulnerability scanner to be interpreted with care. It's the goal of all vulnerability scanners to provide the decision maker with information that helps remediate and manage vulnerabilities. Subsequently, it's of huge importance that the scanner is seen as easy to use [2]. Part of our goal in this study is to visualize the output of the Nessus Vulnerability scanner in a way that allows the output to be more carefully interpreted and more easily utilized.

2.2 Vulnerability Management, Analysis, and Response

Patch and vulnerability management is a practice within cybersecurity that is designed to proactively prevent the exploitation of vulnerabilities that may exist within an organization's network. This proactive approach of managing the vulnerabilities of a network will significantly reduce the probability of exploitation, which in turn leads to massive reduction of time and effort required to prevent a successful attack.

Property	AVDS	McAfee VM	Nessus	NeXpose	Patchlink scan	QualysGuard	SAINT
Software flaws	x	x	x	x	x	x	x
Configuration errors	x	x	x	x	x	x	x
All ports	[d]	x	x	x	[d]	x	x
Active scanning	x	x	x	x	x	x	x
Passive scanning			[b]				
Exploits				x			x
Authenticated scanning	[c]	x	x	x	x	x	x
Vulnerability signatures	6,000	22,000	41,000	53,000	500	6,000	40,000
Web application scans	x	x	x	x		x	
Applications assessment				x			
Patch deployment					x		
SCAP compliance	[a]	x	x	x	x	x	x

Notes: [a]While AVDS is not in the list of SCAP validated tools defined by NIST it seems to be as compliant with SCAP as the other tools; [b]while Nessus itself cannot carry out passive scanning, another tool by Tenable can; [c]only for Windows hosts; [d]only for TCP, not for UDP

Table 2.1: *Comparison of vulnerability identification tools [2]. All currently available tools share very similar functionality. Using a combination of these tools is often the most reliable way to get an accurate reading of vulnerabilities in a system.*

Patches are additional segments of code that are developed to fix an unexpected problem of the original software. They often times will add functionality to the software or address security concerns within the program. Vulnerabilities are flaws that can be exploited by a malicious entity to gain greater access or privileges than it is authorized to have on a network. Since vulnerabilities do not all have related patches that address them, it is necessary for system administrators to be aware of applicable vulnerabilities, available patches, and other methods of remediation including device or network configuration changes and etc.

Fixing security flaws in a network in a timely manner is imperative to maintaining the operational availability, confidentiality, and integrity of a network. However, this is much easier said than done. Lack of an adequate vulnerability management system is a very common issue for many organizations and often leads to them dealing with a much worse situation after an attack. New patches are released daily, and this makes it very difficult for organizations to stay up to date on them. Performing a patch will often require downtime. This makes it hard for organizations to constantly perform the patches. With that being said, when new patches are released, attackers will often make a concerted effort to reverse engineer the patch and identify the vulnerability so they can try and attack a system that hasn't been patched yet. Furthermore, the time after the release of a patch is an extremely vulnerable moment for the organization due to the time lag in obtaining, testing, and deploying a patch.

2.2.1 Vulnerability Management

2.2.1.1 Patch and Vulnerability Groups (PVG)

This complicated environment requires that organizations have a sophisticated system in place with sufficient documentation and accountability that manage exposures to vulnerabilities through timely deployment of patches. These systems are often managed by a Patch and Vulnerability Group (PVG) to facilitate the identification and distribution of patches within the organization. These groups are often responsible for:

1. Inventory the organization's IT resources to determine which hardware equipment, operating systems, and software applications are used within the organization
2. Monitor security sources for vulnerability announcements, patch and non-patch remediations, and emerging threats that correspond to the software within the PVG's system inventory
3. Prioritize the order in which the organization addresses remediation vulnerabilities
4. Create a database of remediations that need to be applied to the organization
5. Conduct testing of patches and non-patch remediations on IT devices that use standardized configurations
6. Oversee vulnerability remediation
7. Distribute vulnerability and remediation information to local administrators
8. Perform automated deployment of patches to IT devices using enterprise patch management tools
9. Configure automatic updates of applications whenever possible and appropriate
10. Verify vulnerability remediation through network and host vulnerability scanning
11. Train administrators on how to apply vulnerability remediations

2.2.1.2 Automated Patch Management Tools

Manually patching networks has become ineffective as networks expand and the number of needed patches exponentially increases. The solution of choice is a tested and integrated patching process that makes use of automated patching technology. Most medium-sized to large organizations use automated patching tools for the majority of their network. However, these tools must be used very carefully. If a hacker were able to break into the automated patch management system, it would be a very easy way for the hacker to distribute large amounts of malicious code.

2.2.1.3 Standardized Configurations for IT Resources

It is imperative that organizations follow a standardized configuration within the enterprise, as it will reduce the labor related to patch and vulnerability management. It is possible that the PVG will find it difficult to effectively test patches if devices use nonstandard configurations. In addition, patch management tools may be ineffective if devices are configured uniquely, because the side effects of the various patches on the different configurations will be unknown.

2.2.1.4 Attack Graphs

When networks grow exceptionally large it becomes harder to preempt all possible attacks from malicious sources, as the number of vulnerabilities in a network often grows with the number and diversity of endpoints in that network. Additionally, attackers launch more complex attacks each day, often combining multiple steps and hosts with the goal of incrementally penetrating the target network [24]. One of the most useful tools in a security analyst's arsenal is the attack graph, which Jha, Sheyner, and Wing (2002) define as "a succinct representation of all paths through a system that end in a state where an intruder has successfully achieved his goal" [25]. Therefore, attack graphs are well-suited at displaying chain vulnerabilities, where multiple small exploits performed in a certain order can lead to a major breach. Additionally, thanks to their simple visual representation, attack graphs quickly communicate the causal relationship between vulnerabilities to viewers.

2.3 Vulnerability Prioritization

2.3.1 Probabilistic Attack Graphs

While certainly useful for high level analysis of a network's vulnerabilities, attack graphs lack the granular information that analysts need to make decisions on which vulnerabilities to direct their efforts at first. Probabilistic attack graphs - which include the likelihoods of each vulnerability in a chain to be exploited - were developed to satisfy this need. Drawing upon already existing metrics such as CVSS base and temporal score to model exploit likelihood, probabilistic attack graphs give analysts more information to help triage vulnerabilities and thus make better decisions [5]. An example network configuration and associated probabilistic attack graph is depicted in Figure 2.1. Much work has gone into improving the efficiency and robustness of probabilistic attack graph generation, making them attractive for use even in enterprise networks [24].

2.3.2 Common Vulnerability Scoring System (CVSS)

Many security analysts and information security professionals consult some form of standardized vulnerability scoring system with the intent of simplifying and standardizing their vulnerability

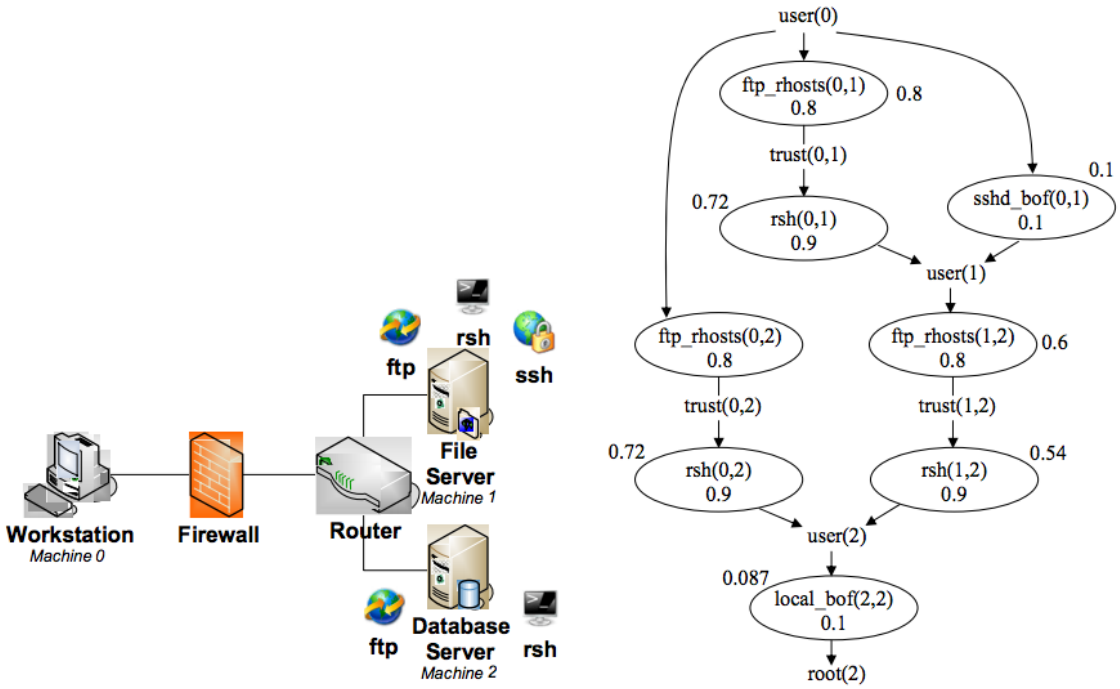

Figure 2.1: *An example network configuration and probabilistic attack graph [5]. Probabilistic attack graphs give analyst more information to help triage vulnerabilities and make better decisions than using CVSS alone. These attack graphs show the likelihood of each vulnerability in a chain to be exploited, useful for getting a better idea of the big picture threat facing a system.*

management process. The most commonly used scoring system is the Common Vulnerability Scoring System (CVSS).

CVSS is a public framework created by the Forum of Incident Response and Security Teams (FIRST) which is used by many organizations in an attempt to quantify the severity of known software vulnerabilities. This allows responders and analysts to prioritize threats and respond accordingly with patch solutions to their systems. In it's lifetime to date, FIRST has rolled out two versions of CVSS and currently has specifications for an updated version 3. While many organizations utilize CVSS and the information it provides in their vulnerability management process, it certainly still has it's drawbacks. Attempting to quantify the risk and severity of software vulnerabilities with ever-growing networks and system complexities into a scale between zero and ten is no easy task. Most of the criticism and feedback towards CVSS is a result of this challenge. A brief summary of the score calculation process is included below, followed by descriptions regarding the areas of improvement for CVSS [6].

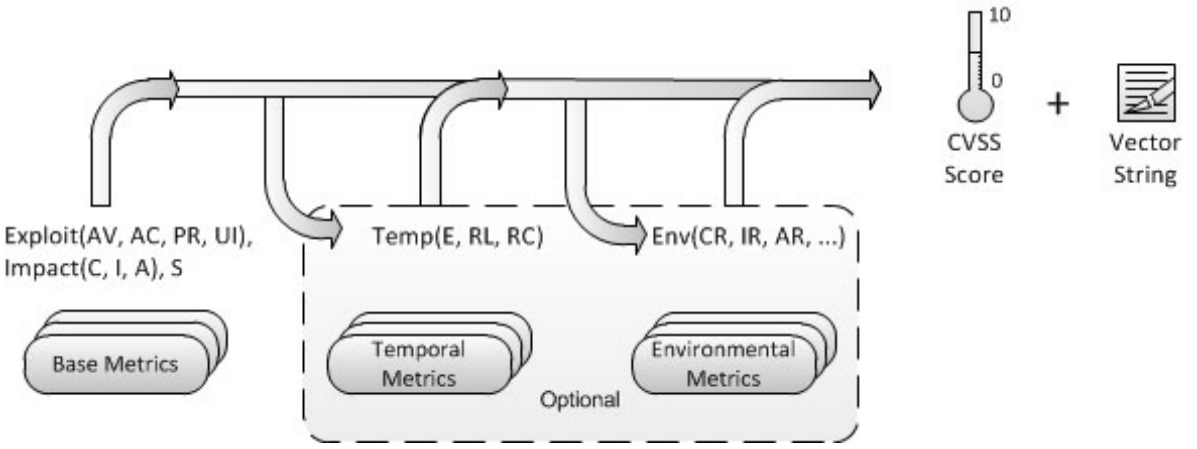

Figure 2.2: *CVSS Metrics and Equations [6]. A CVSS score is first calculated by aggregating all base metrics, then subtracting temporal and environmental metrics. A high severity exploit that is very old for instance, poses less threat than one that is newer (and thus less likely to have patch out for). Only the base metrics are needed to create a CVSS score, the temporal and environmental metrics should be added when the information is available.*

2.3.2.1 CVSS Calculation Process

A score between zero and ten is assigned based off of three metric groups: base, temporal, and environmental. The base metric group represents innate qualities of the vulnerability itself independent of the user's system. There are two metric groups within the base metric group: exploitability metrics, and impact metrics. Exploitability metrics aim to characterize the vulnerable component of a system and include attack vector, attack complexity, privileges required, and user interaction. Impact metrics aim to characterize the confidentiality, integrity, and availability implications of penetrating the system via the vulnerable component. The temporal metric group attempts to add into the score information about how the characteristics of the vulnerability could change with time by observing exploit code maturity, remediation level, and report confidence. Finally, the environmental group represents qualities that are unique to a specific user's environment. Figure 2.2 illustrates score calculation via these metrics [6].

2.3.2.2 Criticisms and Areas of Improvement

After announcing the proposal for CVSS version 3 in 2013, the Open Security Foundation wrote an open letter to FIRST delineating all of version 2's shortcomings. The problems mentioned include: lack of granularity, limitations of Access Complexity and Access Vector, and authentication scoring. The underlying themes across these areas of improvement which are most pertinent to this project and paper are ones of context-dependence and downfalls of attempted uniformity [26].

With the CVSS score being based off of a 3-level scoring system (None, Partial, Complete) for Impact metrics, the scenario arises where very different vulnerabilities end up receiving the

same score. This can lead to misinformed perceptions about the risk of certain vulnerabilities and may lead to poorly allocated resources in terms of implementing a patch solution. An interesting example of this laid out in the letter is the case of uncertainty with respect to software application privileges. CVSS recommends analysts to assume a default configuration or a most commonly used configuration when looking at privileges of certain applications. The problem here is the configuration isn't always obvious due to bundling and customized execution. The decision between classifying a vulnerability as either 'Complete' or 'Partial' confidentiality becomes quite convoluted. The impact in context in this example comes between a hypothetical server administrator who does everything with root privileges vs. a user who only acts within the context of their own account. The same vulnerability has very different implications for each of these people [26].

The second big problem voiced against CVSS version 2 was the weakness of the Access Complexity abstraction. This metric is designed to account for level of user interaction required to exploit the vulnerability which could come in the form of social engineering, visiting certain websites, taking actions on the websites, etc. The National Vulnerability Database stores examples of Man-in-the-Middle attack vulnerabilities with Access Complexity scores ranging from low, to medium, to high, due to the subjectivity of this category, excessive room for interpretation, and lack of support for context-dependent attacks. Analysts sometimes even specifically tweak the Access Complexity score because it's so malleable in an attempt to account for some context-dependence and to ensure different resulting scores [26].

The next area of improvement for CVSS was the Access Vector which also breaks down into three components: Local, Adjacent Network, and Network. The problem with this three-pronged approach is there's no distinction made between the requirement of an account on the system and a true physical attack - both are scored as Local. Additionally, context-dependent attacks are similarly grouped as Network (true direct remote attacks). This can be misleading in cases where a direct Denial of Service attack's severity is much greater than a context-dependent Denial of Service attack [26].

Finally, the strongest criticism against CVSS surrounds the dichotomy of authentication it imposes: a user either authenticates once or twice. The definition fails to include cases where not only the hacker, but also the target need to be authenticated for a certain vulnerability to be exploited. Additionally, the Authentication metric fails to account for permission levels granted via authentication. A vulnerability which requires authenticating for administration-level access is much more severe and this should be reflected through CVSS [26].

2.4 Data Visualization

We live in a world that has become overrun with massive amounts of data. By the end of 2015, more data had been produced in the previous two years than in the entire preceding history

of the human race. The rate of production is rising exponentially; it is estimated that around 1.7 megabytes of new information will be produced every second for every human on earth [27]. Frequently, one of the biggest challenges is displaying the data in such a way that trends become evident and meaningful conclusions can be drawn. In fact, one survey revealed that 62% of marketers felt overwhelmed by the volume of incoming data, and 85% felt that they could not extract the full value of the information coming from these sources [28]. Data visualization, which aims to communicate data through visual representations, is one strategy that has seen increasing use as a means to make sense of large datasets.

The most concise way to describe data visualization is a method of transforming data into useful information that can be used to gain insight into a process we are interested in. Many times, an individual can draw new conclusions or focus on key features more easily from data when it is arranged in a more pleasing and/or informative format, which is the motivation behind visualizing data. If a visualization is neither usable or actionable, than it likely holds no advantages over the raw data. It is important to strike a balance between a practical interface and one that is visually pleasing, as one that emphasizes one aesthetics too much will fail to capture the user's attention and effectively deliver information. Telea (2008) asserts that an effective visualization will often be more effective than raw text data for the following reasons [7]:

1. Will allow the user to draw perceptual inferences quickly rather than struggling to draw difficult logical inferences from dense data

2. Will take the user less time to search for the information they are interested in or need to complete the task at hand

The actual application that performs the visualization is only a part of the entire visualization process, as diagrammed in Figure 2.3. This process details the role a user plays in visualization, which is to not only view (and interact, if possible) with the application, but to also search for novel information - i.e. insight - that may be revealed from the data in it's visualized format. Analyzing and quantifying this information can allow it to be visualized as well, furthering the process to gain greater knowledge.

With so many factors to balance in order to produce an effective visualization, it should come as no surprise that there isn't a one-size-fits-all graph or diagram that can convey insight from every possible set of raw data. Significant preparatory research and testing should be performed when deciding how to visualize a given data set to solve a problem, as this will lead to more effective visualizations and greater insight into the data.

2.4.1 Visualization in Security

One useful area of application of data visualization is in the field of security and vulnerability management. There is much potential for visualization in security, as analysts often find themselves overwhelmed with large amounts of vulnerability data in the form of text or other

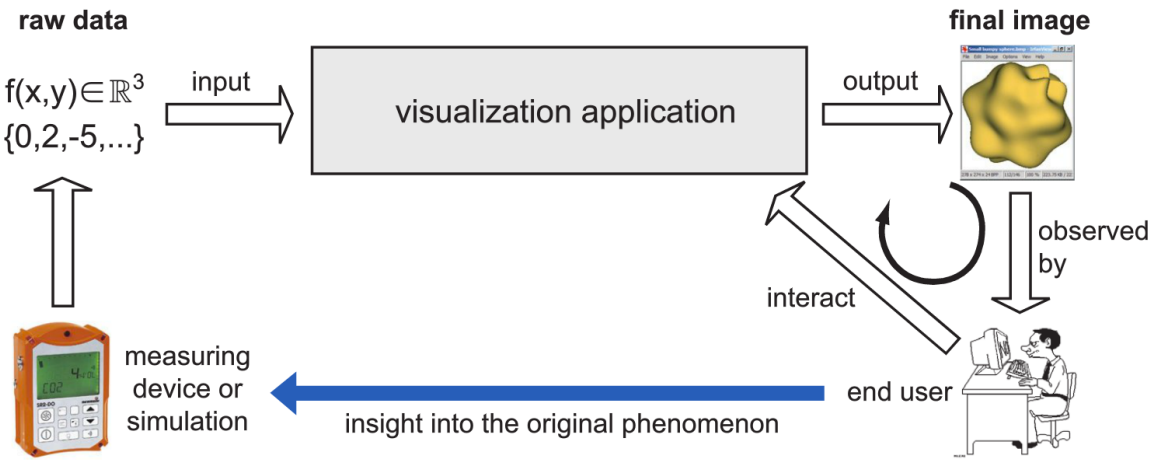

Figure 2.3: *The visualization process [7]. The main purpose of visualization is to display raw data in a manner that allows a user to gain insight from it. This insight can be transformed into raw data and the process restarted, allowing for even deeper observations to be made.*

low-level representations, which can be hard to see patterns in. Indeed, this is where visualization shines, as "by carefully crafting graphical windows into data to exploit the high-bandwidth visual recognition capability of humans, we can see patterns and anomalies as well as detect malicious activity that would be impossible to detect using traditional computing techniques or text" [29].

Especially on the enterprise level, gigabytes of data can be generated a day for a single organization, and the task of making sense of this data is quickly becoming an intractable problem. Analysts must be able to distinguish between useful security data and noise of legitimate operations, a task made harder when one can barely keep up with just being able to see the most recent alert. With too much information to process effectively, and immense amounts of discouraging false positives, analysts may begin to start ignoring security alerts, which opens the door for real vulnerabilities to slip past unnoticed.

One of the most important contexts of security visualization is large networks. As a network grows - whether the nodes be single machines or something else - so too does the number of possible vulnerabilities. Complex networks not only have the greatest likelihood of being exploited due to their immensity, but also can suffer the greatest possible loss. Tremendous amounts of security data raises the chances of an analyst missing an important vulnerability. As time goes on, networks face a greater number of exploits at a higher level of complexity. Greater number of nodes in the network create a greater number of entry points for possible exploits. And successful exploits affect a far greater number of people/assets than that of a small network. This is where data visualization can make a difference. Harrison and Lu explored recently developed tools for security visualization and general network visualization, finding that both categories of tools

have downfalls, yet when combined, could provide all-purpose solutions satisfying scalability, accuracy, and effectiveness at conveying important information [23].

There are many ways to visualize security data, and the method chosen is often suited best for one or more specific contexts. For instance, an attack graph may provide a powerful bird's eye view of vulnerable pathways to exploitation in a network, but be absolutely useless when applied to a single machine in the network. In the remainder of this section we will explore different prominent methods of security visualization, while making note of the context they are most useful in, if applicable.

2.4.2 Hierarchical Visualization

Complicated networks often have countless nodes, each of which can have many subnodes, each of which may have their own substructure, and so on. One visualization technique - known as hierarchical visualization/graphing - is built to effectively represent data with this architecture. Hierarchical visualization excels displaying data such as file systems, store inventory, and stock portfolio allocation, for instance. The most common implementations of hierarchical visualization include but aren't limited to: sunburst diagrams, tree diagrams, dendrograms, and treemaps. A couple examples of these are detailed in Figure 2.4.

Oftentimes, when visualizing hierarchical data one requires some concept of relative weights between nodes to most accurately represent the dataset. For instance, displaying the contents of a file system conveys more useful information to a user when the visualization clearly communicates the different sizes of files. This is where treemaps, designed by Ben Shneiderman in 1991, really shine. Treemaps were developed to utilize 100% of the space they are placed in, a novel approach to conveying as much information as possible given the available space [30]. Additionally, they take much less space to display large hierarchies as compared to a standard node-link tree diagrams. Combine this with their easy understandability from a user standpoint, and treemaps become one of the best choices for displaying large data sets. With the option to see all elements of the hierarchy in one overview, and differing color schemes amongst nodes, treemaps can effectively display sets of data with high-dimensionality and multiple levels if used correctly.

There is a wide array of studies focused on treemaps and related hierarchical visualization styles, and we chose to list a few of the most pertinent to our work here. Landesberger, et. al (2011) conducted research into the current state of large graph visualizations in 2011. The key findings of this research can be summarized as such [31]:

1. The need for scalable graphs that remain readable, especially in the case of hierarchical data, increases each year as datasets of interest grow larger.

2. In recent years, the variety of available graph types as increase substantially, especially in the categories of dynamic and compound graphs.

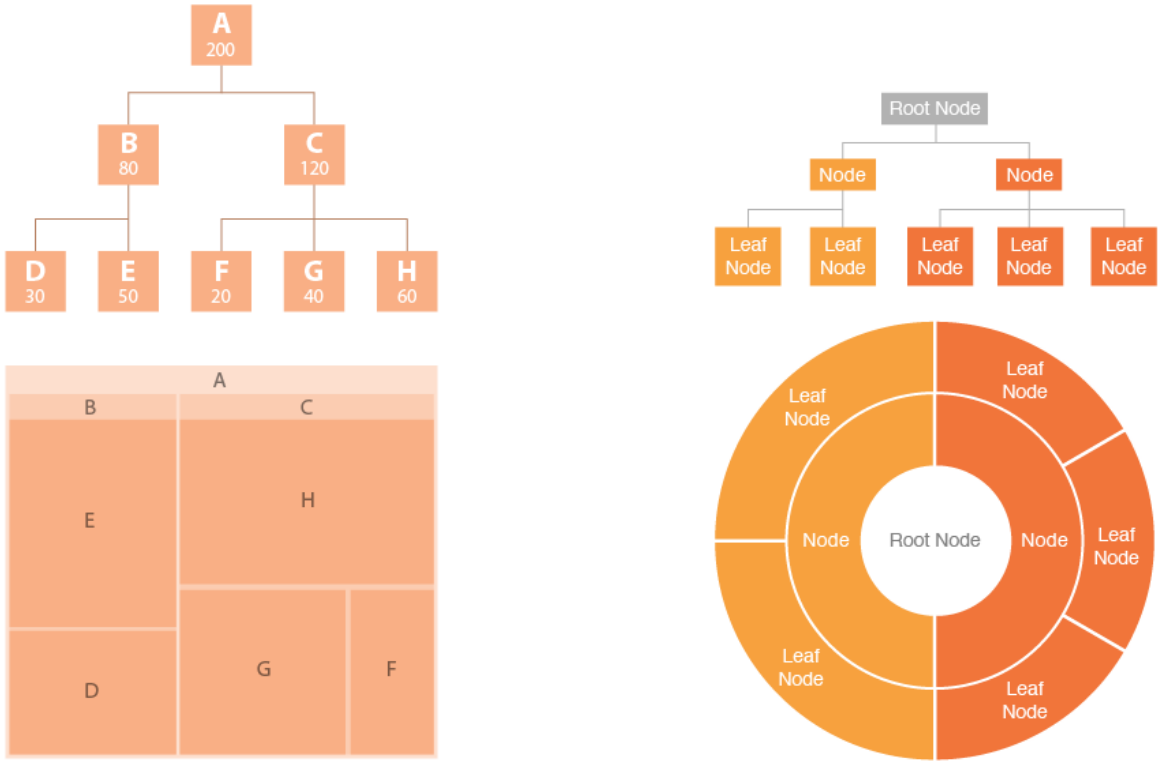

Figure 2.4: *Common hierarchical visualization styles. On the left, a sunburst diagram with it's equivalent tree (node-link diagram) to compare. On the right, a treemap, with it's equivalent tree (datavizcatalogue.com).*

3. Although more frequently a problem in the field of visual analytics, finding methods to effectively deal with graph uncertainty - and how to best represent that - is an ongoing topic in visualization research.

4. The effectiveness of a graph/visualization in conveying useful information to a user is strongly dependent on human perception ability. As more research is done into human perception, visualization methods will need to change to best take advantage of new findings.

5. The ability to interact with graphs is a feature with rising popularity, especially techniques that take advantage of structural properties of the graph to enable navigation of the data.

Wang, Teo, & Ma performed a study in 2006 to determine if tree visualization systems were more effective at enabling knowledge discovery (i.e. gaining insight into the data) than traditional directory-style structures. They determined that when asked to perform exploration tasks on the same data set via a node-link representation, treemap representation, and file directory representation, users found it easier to both navigate and extract information when using the node-link or treemap diagram [32].

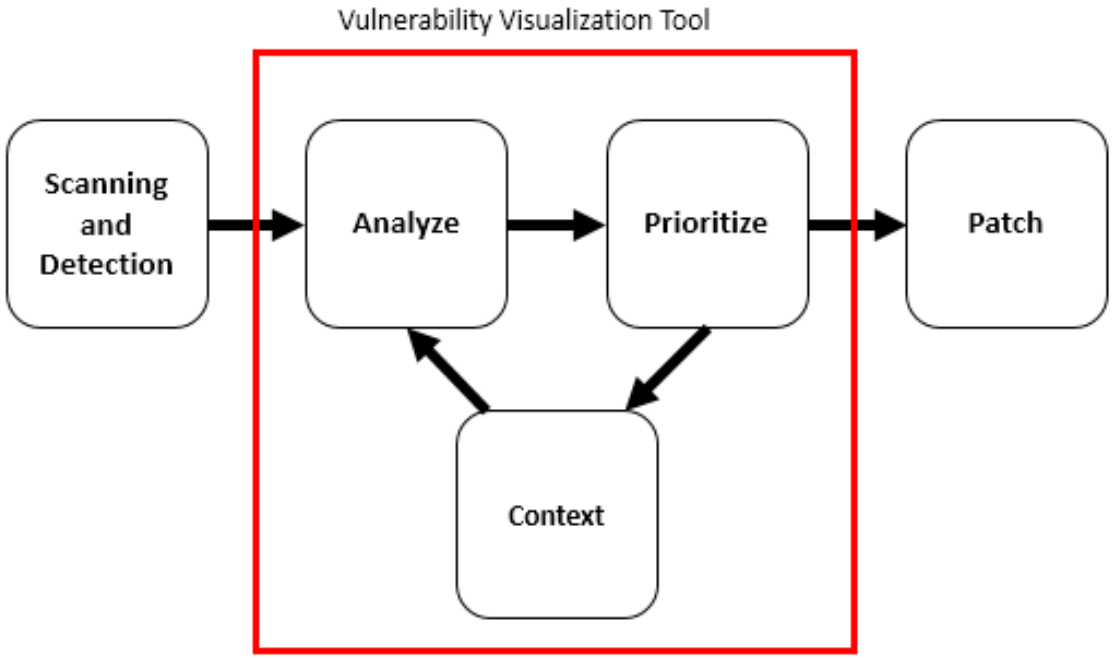

Figure 2.5: *Overview of proposed tool function. The tool will facilitate efficient analysis and prioritization of vulnerabilities through use of context. This is in contrast to simply just choosing which vulnerability to patch via CVSS score alone.*

Finally, Harrison, et. al (2012) formed the foundation for our project with their tool NV. Described in their paper, NV: Nessus vulnerability visualization for the web, this tool takes vulnerability scan output from Tenable Security's Nessus network scanning program, and maps the results to the network structure via a treemap visualization [33]. The researchers involved in the project took the flat data output of Nessus and inferred a hierarchical structure, a novel idea that allowed NV to present a unique way to gain insight into Nessus data. This interactive visualization is meant to support the vulnerability discovery, analysis, and management performed by security analysts and network administrators. NV is ideal for quickly locating and triaging the most severe vulnerabilities in a network, something that is far harder to achieve when observing the raw Nessus text output alone.

In contrast to what has been done so far - as described in the papers listed in this section - we are specifically designing a tool to improve the prioritization capabilities of analysts. Using visualization is highly effective for conveying trends and patterns to viewers, so transforming context of a network into easily viewable trends in the form of visualizations is how we are planning to improve prioritization. This strategy is especially helpful for users that do now know what they are looking for in a network scan and/or have never attempted to prioritize vulnerabilities before. Figure 2.5 lays out the key features of our tool: analysis and prioritization utilizing contextual information.

CHAPTER 3

FUNDAMENTAL CHALLENGES IN VULNERABILITY MANAGEMENT

Prior to building a tool that could improve the vulnerability management process, our team needed to gain an understanding of the current challenges facing the industry. By identifying challenges, we would be able to much more effectively direct our efforts towards building something that would both provide value and prevent repetition of previous approaches.

Our steps to identifying challenges were as follows:

- **Background Research:** We began by gathering general info on vulnerability/risk management through our literature review and general research. This established a solid foundation upon which we could launch other info-gathering methods.
- **Survey Vulnerability Management Professionals:** Next, our team employed a number of direct information gathering techniques to further narrow our focus. The first of these techniques was sending out a survey to a number of security analysts from different companies to learn more about the best-practices used by these professionals, and what sort of difficulties they may face in their day-to-day work.
- **Interview NIST:** We also had the opportunity to interview a member of the National Institute of Standards and Technology (NIST), which further influenced our project direction.
- **Attend IEEE VizSec 2016 Symposium:** Finally, in order to make sure our project would be relevant to current proceedings in the vulnerability management industry, our team attended the IEEE VizSec 2016 Symposium to take notes on the most recent research in the field.

With this research methodology, we were able to not only establish a strong working knowledge of the entire field of vulnerability management, but also gain relevant, precise details on the

subfields of cybersecurity and security visualization. This helped us take the numerous challenges we identified along the way and narrow them down to the one we wished to tackle with our tool: the necessity of contextual information to effectively prioritize remediation of vulnerabilities.

3.1 Investigation and Research

3.1.1 Background Research

Before we spoke with any vulnerability management professionals, our team wanted to have a decent background in the field itself. Thus, we read through a number of prominent papers, which we summarize in our literature review. We came across numerous articles and sources that frequently expressed frustration with and/or attempted to improve the one-size-fits-all approach of vulnerability management into more specialized approaches for different situations. We also gained a greater understanding of the most common processes, best practices, and tools. Using what we learned from this preliminary research, we went into our subsequent information gathering methods (e.g. interviews) with confidence and armed with the most pertinent questions to make the most of all participants' time.

3.1.2 Survey Vulnerability Management Professionals

Following our preliminary research, we constructed a survey to be taken by knowledgeable individuals from the vulnerability management and threat mitigation community. Our goal was to use answers from the survey to help gain a further understanding of how different types of companies perform vulnerability management, narrow down the scope of our project, and make sure that our work had true value to it. Our first course of action was to create a survey. Our goal was to get a better understanding of how different types of companies perform vulnerability management. We targeted firms from a variety of industries that have sophisticated security practices such as finance, government, software security, and consumer electronics.

Our survey starts by collecting basic information about the company, including type of industry, size, and amount of resources dedicated to security. This allowed us to better categorize the data further in the survey. Next, we gather some information about how the company performs vulnerability management. We look for a description of the workflow of the vulnerability management process to get an understanding how standard the processes are. We also have them describe the size of the network and what tools/software they use to scan and/or manage their vulnerabilities. After they describe to us how they generally scan for vulnerabilities, we ask for their opinions on the best tools, and how they currently feel about their process as of now. These first two sections can be seen in Figure 3.1 below.

Next, we investigate how they plan and prioritize fixes for their discovered vulnerabilities. This gives us a good framework for future design choices in our tool. In addition, we ask them to describe the information they gather when they investigate context when they are performing an

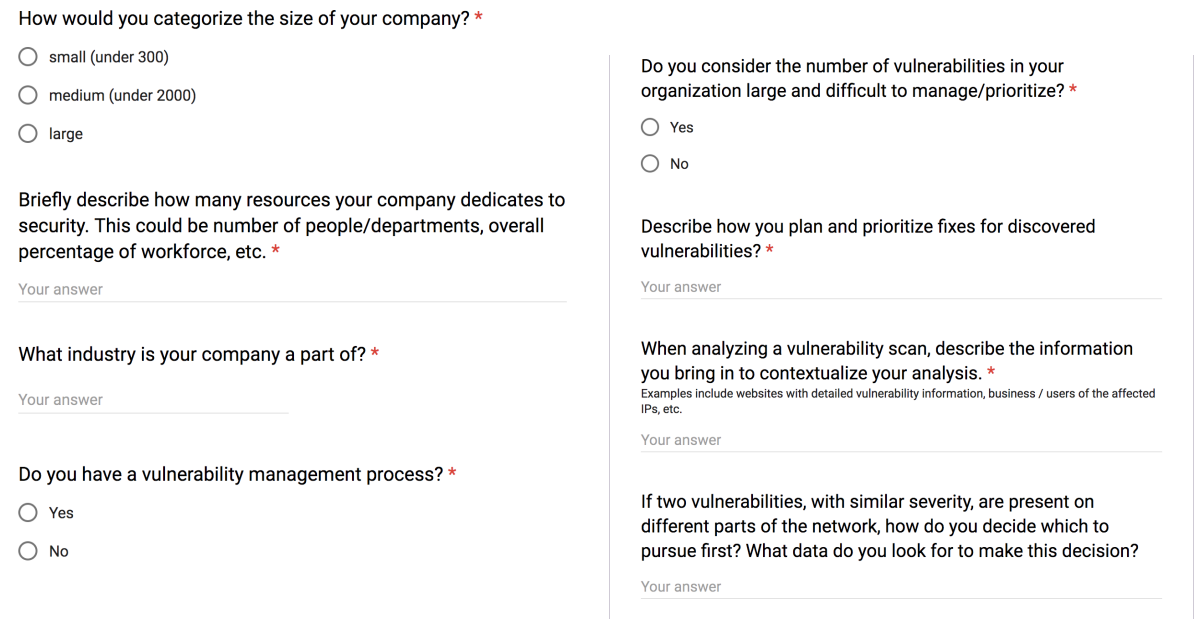

Figure 3.1: *From left to right, the Basic Information section and Vulnerability Management section. These sections helped us characterize the type of each company that completed the survey and give us an overview of the methods and processes each company used in their vulnerability management, respectively.*

analysis on a vulnerability. We also ask how they prioritize two vulnerabilities and what data they gather, if they have two similar vulnerabilities with similar severities on separate parts of the network. Next, we get into the frequency and time frame of their vulnerability scans. We ask them to elaborate on how often they run their scans and if they scan certain parts of the network more than others. Then they explain how long it takes to identify these vulnerabilities and how they schedule the patching of them. We conclude the vulnerability management section by asking them how they incorporate CVSS into their analysis, and if there are any other features that their current toolset lacks that would make vulnerability management easier.

In the final part of the survey, we ask some general questions to help us with our project. We ask the security analysts to describe their picture of an ideal vulnerability scanning tool. They also tell us about their experience with data visualizations and the extent to which they exist in the tools they already use now. Finally, we briefly ask them about attack graphs and machine learning techniques and if they apply to any of their strategies. Sample questions from this final section and the third section (security resources) can be seen in Figure 3.2.

3.1.3 Interview NIST

Our next course of action in information gathering was interviewing a computer scientist from the National Institute of Standards and Technology (NIST) in the Cyber Security Division. Additionally, this employee had plenty of familiarity with the National Vulnerability Database,

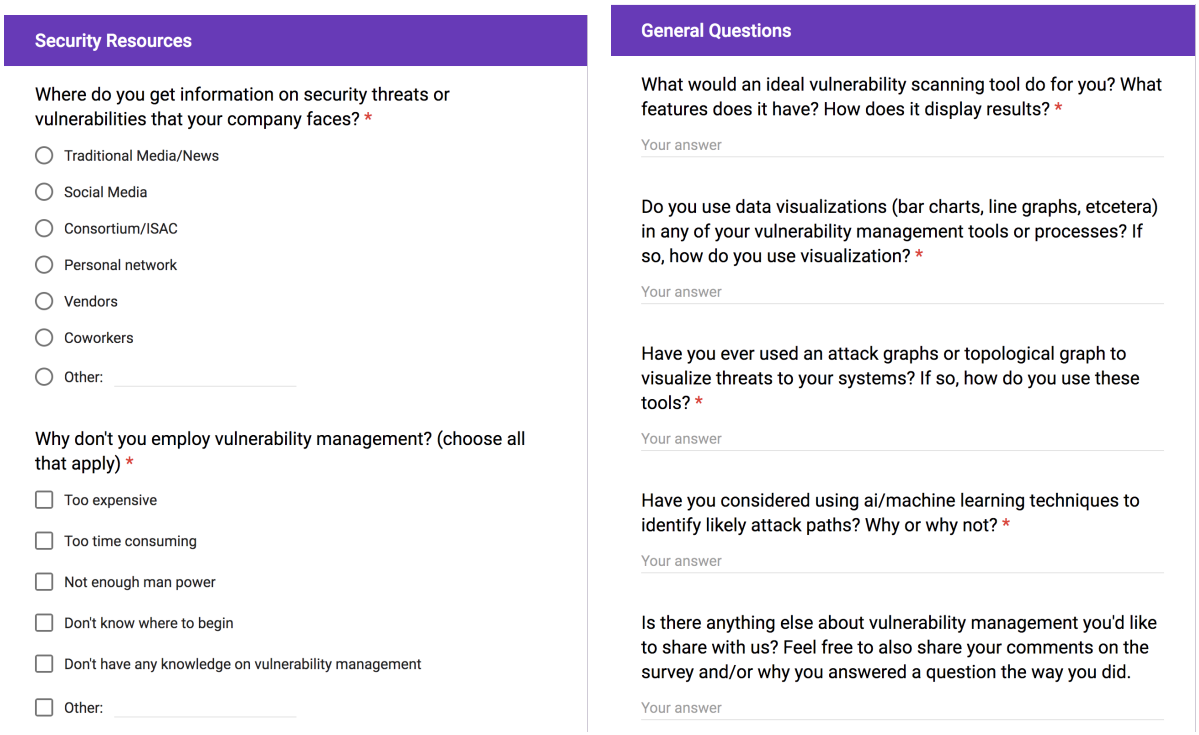

Figure 3.2: *Sample questions from the last two sections of the security survey: Security Resources and General Questions. The Security Resources section helped characterize how companies deal with discovered vulnerabilities and prioritize remediation, while the General Questions section gave our group some additional direction for reach goals in our project.*

and could give us an in-depth explanation of the purpose of CVSS, it's limitations, and ways to prioritize risk outside of just CVSS scores.

In our conference call, the employee highlighted many topic areas that would be good areas of research for our team. They mentioned that it would be very helpful for someone to draw a connection between exploits that are being used and vulnerabilities, as well as highlighting a dire need of standardization of context across different organizations. NIST's current approach to solve the standardization problem is FIPS 199 [3]. FIPS 199 provides a mechanism to determine relative importance of an information system (low, moderate, high). This rating is based off of confidentiality, availability, and integrity of the system. A breakdown of these three security objectives, along with the potential impact levels of each, is shown in Table 3.1. As of now it is quite difficult to identify risk of a system, and we can only identify the severity of vulnerabilities. Thus the problem that arises is how to establish a set of questions to characterize a system and identify the common characteristics between vulnerabilities. This standard has to be made without enforcing a particular viewpoint and not be too revealing of the organization's attributes.

Later, our contact talked about the secure automation team. The secure automation team aims to automate processes so humans can concentrate on more important things. They try to develop frameworks to allow computers to understand the context of the problems they are trying

	POTENTIAL IMPACT		
Security Objective	**LOW**	**MODERATE**	**HIGH**
Confidentiality Preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information. [44 U.S.C., SEC. 3542]	The unauthorized disclosure of information could be expected to have a **limited** adverse effect on organizational operations, organizational assets, or individuals.	The unauthorized disclosure of information could be expected to have a **serious** adverse effect on organizational operations, organizational assets, or individuals.	The unauthorized disclosure of information could be expected to have a **severe or catastrophic** adverse effect on organizational operations, organizational assets, or individuals.
Integrity Guarding against improper information modification or destruction, and includes ensuring information non-repudiation and authenticity. [44 U.S.C., SEC. 3542]	The unauthorized modification or destruction of information could be expected to have a **limited** adverse effect on organizational operations, organizational assets, or individuals.	The unauthorized modification or destruction of information could be expected to have a **serious** adverse effect on organizational operations, organizational assets, or individuals.	The unauthorized modification or destruction of information could be expected to have a **severe or catastrophic** adverse effect on organizational operations, organizational assets, or individuals.
Availability Ensuring timely and reliable access to and use of information. [44 U.S.C., SEC. 3542]	The disruption of access to or use of information or an information system could be expected to have a **limited** adverse effect on organizational operations, organizational assets, or individuals.	The disruption of access to or use of information or an information system could be expected to have a **serious** adverse effect on organizational operations, organizational assets, or individuals.	The disruption of access to or use of information or an information system could be expected to have a **severe or catastrophic** adverse effect on organizational operations, organizational assets, or individuals.

Table 3.1: *The FIPS 199 Security Categorization methodology in table format [3]. While there is no one-stop solution to categorizing risk in information security, many frameworks have been proposed that when applied can improve vulnerability management. FIPS 199 is NIST's proposed standards for security categorization, and boasts usability through simplicity. With three security objectives per information system, each with three potential impact levels, FIPS 199 is a great starting point for vulnerability management teams who are trying to build a prioritization and/or remediation process. This categorization system influenced development of our final approach to prioritization in this project.*

Figure 3.3: *In October of 2016, our group attended the IEEE Symposium on Visualization for Cyber Security (VizSec) where we sat in on a number of speakers share the latest research in the field.*

to solve. They want to use FIPS or other framework/methods to describe a system they are trying to protect, so that it is understandable by other computers. They want to avoid having to describe the context of every instance on different systems of the same vulnerability. They are currently in the process of creating an ontology that will hopefully allow for machine learning to pre-score vulnerabilities.

We concluded the conference call with some ideas to concentrate our project on. The employee gave us some background on their work on the CVSS standard, it's purpose, and it's pitfalls. They highly discouraged sole use of CVSS to characterize a vulnerability, and that above all context of the vulnerability was crucial for establishing situational risk.

3.1.4 Attend IEEE VizSec 2016 Symposium

In October of 2016, we attended the IEEE Symposium on Visualization for Cyber Security (Figure 3.3). The symposium took place in Baltimore, Maryland. The primary purpose of the symposium was to bring together researchers and practitioners in information visualization and security to address the specific needs of the cyber security community through new and insightful visualization techniques.

3.1.4.1 Visualization of Actionable Knowledge to Mitigate DRDoS Attacks

One of the main panels at the symposium was about using visualizations to mitigate **DRDoS**[1] attacks. A DRDoS attack is essentially a UDP based-amplification for Reflective Denial of Service (RDoS) attacks. The issue at hand from the point of view of an ISP, is that malicious traffic uses bandwidth and decreases quality of service for real users, and costs the ISP a lot of money. ISPs would like a way to block the malicious traffic only.

[1]**DRDoS:** Distributed Reflected Denial of Service, a type of Reflective Denial of Service (RDoS) attack where spoofed packets are sent to amplifiers that multiple the response up to 500 fold.

The tool proposed in this panel was a visualization tool for the ISP to detect and mitigate DRDoS attacks. By using Qatar Cyber Security Platform (QaCIP) architecture, this visualization can collect massive amounts of data, perform real time processing, perform indexing, and provide visualizations and analytics.

The data source of this software will be the packets of data captured on a network. Actionable attributes of the packets (the headers) will provide lots of information about the origins and intent of the data. In addition, there is additional information from informational attributes from the enrichment process.

This software can block attacker packets by filtering packets that match malicious packets like a regex. However, filtering is very difficult to design. They require expert knowledge, evaluation of impact, and comparison skills. To combat the difficulty, we can use VizFilt to design a visual analytic interface to assist us in designing filters. VizFilt is an original modeling technique that combines qualities of stacked bar charts, parallel sets, and treemaps.

3.1.4.2 Context of Network Traffic Alerts

This panel was all about understanding the context of network traffic alerts. On the topic of Advanced Persistent Threats (APTs) it is important to be able to understand and detect the infiltration, expansion, and the type of sabotage. A common data source that can explain context, is the data packets over the network. By using Wireshark, you can capture network traffic on the network and perform deep packet inspection.

The motivation for learning more about context is to find attacks that happen over a longer period of time by using historical tracking. We can look at exploration of the traffic as well. By analyzing if alerts have connections to past alerts, if messages explain context of alerts, and making widgets for attributes for better analysis, it can lead to a much better understanding of context. More specifically, profile traffic vs time using graphing techniques by individual IPs and conversations between IPs can provide a large amount of the context.

To conclude, the project has many strengths. These include dynamic exploration which essentially provides visual querying, saving intermediate results, and the ability to be expressive through interaction. The main weakness is that it does not scale well. You can only analyze up to a million messages. The future of the project is up to their ability to scale it to larger environments. This project is a fantastic look into the use of context to give a greater understanding of the threat a system faces, and bolstered our team's interest in researching the subject further.

3.1.4.3 Visualizing Malicious Logins

Another panel at the symposium was about detecting malicious logins on enterprise networks by using visualization. This was motivated by how difficult it is to identify an Advanced Persistent Threat (APT). Each year, APTs steal and destroy massive amounts of data from companies. For example, Target had 40 million credit card accounts taken from an APT attack.

The panelists called the visualization "APT-Hunter". By providing a visualization, the users are able to discover login rules, express login rules, and detect suspicious logins. In their initial tests, it was found to be moderately successful, as around 50% of malicious logins were detected from the visualization.

3.1.4.4 V3SPA

The final panel of note was about visually analyzing security enhanced Linux security policies. Security enhanced Linux implements mandatory access control, where access to files must be granted explicitly by an administrator. This results in less damage of compromised processes. 81% of Android devices use security enhanced Linux for Android. Security enhanced Linux Tresys reference policy consists of around 94,000 access rules. Changing these manually is tedious and results in frequent errors. This is usually modified using vim and emacs.

With that being said, policy graph clustering and grouping has been implemented in order to try and combat this problem. In addition, policy protection comparison, security enhanced Linux attack log analysis, Gephi, and NodeXL graphing have all been created in order to try and combat this problem.

V3SPA tries to combat this by visualizing use cases of policies. V3SPA has a policy explorer view which dramatically helps the user understand the policy structure on the system. In addition, the policy differ view helps the user understand policy difference, which essentially allows you to see the implication of changing a setting.

V3SPA is not very computationally expensive. It uses caching, so it can use parsed versions of data. In addition, it uses JSONH for its data compression. Finally, it optimizes the graph encoding since it has such a large effect on the size.

3.2 Research Results and Project Direction

From our information gathering methods - background research, industry professional survey, NIST employee interview, and IEEE Symposium - we uncovered a plethora of challenges that hindered effective vulnerability management. We have summarized the four biggest challenge areas below:

- **A Sisyphean Task:** There are almost always more threats than can be solved with the resources available, and attackers always have the “zero-day” advantage. Analysts are always on the defensive - as is the nature of the security field - which puts them at a disadvantage.

- **Finding the Right Net to Cast:** Many automated vulnerability scanners are set to configurations that are too strict or too lenient, leading to constant false positive vulnerability

alerts or low-profile vulnerabilities slipping through, respectively. Lowering both false positive and slip-through rates are fundamental challenges in vulnerability management.

- **Lack of Context:** Vulnerability analysts need context of a vulnerability to make the most informed decision on how to approach it. Context is an extremely hard thing to generalize, as it is specific to the organization/situation in which it exists. Gathering context often involves considering the environment, the value of assets, and the timeframe.
- **The Need for Multiple Sources:** Drawing upon multiple information sources is a critical component in correctly characterizing a vulnerability and it's risk to an organization. Having to draw from many information sources means having to switch between many different applications/screens, which can greatly impact time-to-response and overall concentration as an analyst.

The largest commonality between our research results was the critical importance of contextual information in order to prioritize effectively. Without context of the situation posed by a vulnerability - such as type of vulnerability, presence of an exploit, lifetime in the system, size of the company, etc. - analysts were making decisions on how to prioritize remediation based off a single number: CVSS score. This practice of only using CVSS score leaves decisions woefully under-informed, and can result in the vulnerabilities that pose the greatest threat to an organization flying under the radar until it's too late. These concerns were echoed by results from our survey and discussions with industry members at the symposium, as well as underlying many of the papers we investigated. Out of the four above, this was the challenge with the farthest reaching implications. With this in mind, our team decided our focus for this project would be on context. Specifically, we would develop a tool that could consolidate multiple sources of contextual information regarding vulnerabilities into a single visualization dashboard, allowing analysts to find and prioritize vulnerabilities in a more time-efficient and knowledgeable manner. A standard vulnerability management life cycle is depicted in Figure 3.4, with red circles around the steps of the process that are easier to perform when using VV to provide contextual information.

3.2.1 Challenges with Context

Soon after choosing context to be the focus of our project, we were faced with a number of new challenges. As mentioned above, context is already hard to generalize, however we soon realized that we would also have to fetch, integrate, and show context in a robust, repeatable manner for our tool to be useful.

To begin, we had to define the context we wished to include in the tool, and by extension, determine the sources we would fetch the context from. As the goal of our project was to demonstrate the usefulness of a tool that utilized context, rather than aggregate every single possible source of context that could be of use to an analyst, we settled for using a few sources. These sources are:

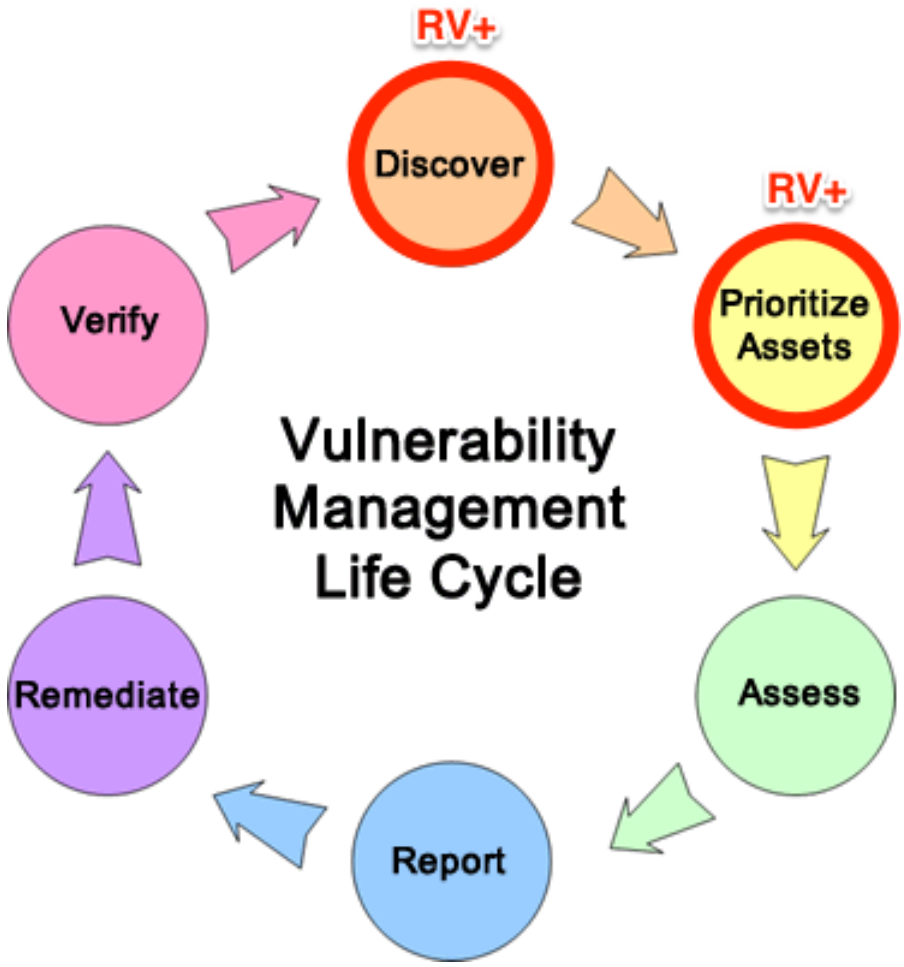

Figure 3.4: *The Vulnerability Management Life Cycle as defined by the CDC [8]. The circles outlined in red represent steps in the life cycle that are augmented by the VV tool. These steps are directly integrated into the vulnerability management process through use of the VV tool.*

- CVSS Base + Temporal Metrics (extracted from CVSS vector strings in .nessus scans)
- Exploit Presence + Exploit Distribution (extracted through .nessus scan info and Metasploit API)
- Presence in Qualys Top 10 Vulnerabilities List for a given year (extracted from Qualys website)

We believe that these sources of context effectively demonstrate the added usefulness of contextual information in prioritization as opposed to using CVSS score alone.

We next had to determine how we would integrate and show the context we had chosen in our tool. Visualization was the natural choice for this project, as linked visualizations offered an option for a strong, easily-understandable visual for analysis purposes. To make the tool as

seamless as possible - and drawing from the strengths of the NV tool - we decided to design the treemap and added context to all fit on a single page. The treemap would be surrounded by smaller charts representing the context data that corresponds to the current state of the treemap. Therefore, as an analyst navigates through the treemap, the context charts would update accordingly, giving new insight into each level of a network scan.

One drawback of using purely visualization for network scan exploration is the loss of data granularity. Visualization is ideal for showing trends in data, but sacrifices the details and attributes about each data point to accomplish that. To combat this disadvantage, we included a separate page in our tool called the Vulnerability List page. As the name suggests, this page contains a list of all the vulnerabilities detected in the chosen scan, and allows the user to filter and sort items based on multiple attributes. This portion of the tool is best suited to an analyst that is looking for a certain vulnerability, and already knows a few attributes about it.

We explore the implementation of these design choices in the following chapter.

CHAPTER 4

BUILDING A TOOL TO VISUALIZE CONTEXT

4.1 Project Design

NV - Nessus Visualization - is where this project evolved from. NV is a client-side web application built in Node.js, Backbone.js, and D3.js, among a few other frameworks. NV originally would accept version 1 output from Tennable Nessus' scans. The application would parse the output of the scan, and the build it into an intuitive interface. More specifically, using the power of D3.js, the output is embedded into a multi-layer treemap interface. The user has the ability to change how the layers are ordered, but by default it starts at IP address, goes to ports, and then it displays all the vulnerabilities under that specific IP and Port. In addition, the treemap sizes and color codes the blocks based on either severity, criticality, or count.

Despite all of its great features, NV does not have any functionality to convey context-based risk behind specific vulnerabilities. Our research has shown that security analysts primarily value vulnerability context and the vulnerability's life cycle. Our goal was to modify the existing project and add functionality to convey context and add flexibility to align it with vulnerability life cycles. In addition, NV cannot support and effectively visualize over ten thousand vulnerabilities. Due to these large requirements, it became clear that it would require a redesign of the web application in order to be possible.

4.1.1 Technology Design Decisions

Since a redesign was necessary, we decided to take advantage of this situation and migrate to modern frameworks. Modern frameworks offer improved performance, more flexible design options, and a more active support community.

4.1.1.1 Front End

The front-end JavaScript framework was our first major design decision. We were torn between Backbone.js, Angular2.js, and React.js. We decided to reject Backbone.js because it was used in the previous project, its community is decreasing, and it is outperformed by more modern frameworks.

Angular2.js is a modern framework created by Google and has gained a lot of popularity recently. It is extremely powerful and yields some of the best performance possible for web applications. However, Angular2.js has an extremely large learning curve because it is very close to being a fully-fledged framework. It is packaged with almost every tool that you could possibly need to build a web application. In addition, Angular2 has very "angular-specific syntax" which contradicts common and known design strategies. Angular2 is written in typescript, which is a superset language of JavaScript that was developed by Microsoft. It was created to provide new data structures and object oriented design concepts, with the intention of making the code easier to read and maintain when compared to original JavaScript. Recent versions of typescript have used arrow and other ES6 specific notation, which has caused it to start to resemble JavaScript.

Due to the extensive learning curve and the tedious configurations required, we decided to reject Angular2. We then decided to do some research into Facebook's React.js JavaScript framework. React is very different from Angular but provides almost all of the same benefits of Angular. React is far more dynamic than Angular2. Instead of being given a large library of tools, you are given almost nothing and are required to import packages that the developer would like to use. This is favorable because you only need to import what is necessary and can omit the large packages that Angular2 may not use.

Furthermore, React uses JSX (Java Serialization to XML), which is a XML type extension that renders JavaScript and HTML. The syntax and structure are far more similar to JavaScript than typescript. In addition, within the HTML markup, JSX allows the mixing of variables and JavaScript data structures. Due to these similarities, learning React will have stronger long term effects on our team's JavaScript skills as opposed to learning a specialized language that may disappear in the future.

Finally, React.js proved to work well with D3.js due to its ability to easily reference the DOM. In addition, React provides a specific method that is called after the DOM is ready for manipulation. Since D3.js is entirely done on the DOM, it made integration with React possible.

4.1.1.2 Back End

The backend framework decision was also important to project success. We considered Python, Java, Go, and Node.js. Python can be a great language for backends. It is mature and has extensive library support. It has many advanced web APIs and is extremely portable. However, due to its low latency and slow performance, developers have been recently moving away from it. In addition, in large code bases, explicit declarations can be a nightmare to navigate.

Java can also be a very powerful backend, but due to its age and lack of modern features, it would increase development time. Since our project does not require a lot of speed at an enterprise level, we would not harness the benefits associated with using Java.

Albeit not as common, Go can also be a very powerful language and a great language to use for large systems. Due to its poor dependency management, cumbersome type systems, difficult memory management, we would again not harness the benefits associated with using Go.

Node.js is widely used and has extensive support from the community. Since it is in JavaScript, it would result in the entire project being written in one language. This includes the client, the server, and the database. This would allow the transfer of JSON objects to be trivial. In addition, it also has the Node Package Manager (NPM) which allows for easy installation of third party packages, version management, and dependency management. It allows you to keep packages isolated from other projects, so that you can avoid version conflicts.

Finally, Node.js is a JavaScript runtime that uses the V8 engine created by Google for use in Google Chrome. V8 is very fast because it compiles JavaScript into native machine code. In addition, it has the event loop, which sends asynchronous tasks with callback functions, so that the server can continue to execute the program. Traditional backends spawn parallel threads that consume extensive memory and are very difficult to program.

4.1.2 Development Practices

Our team consisted of three developers all working on different parts of the project. Zach primarily focused on the backend, Barrett was focused on the interface and parts of the frontend, and Andrew was focused on the D3.js visualizations and their integrations into the front end.

We would meet weekly and Scrum at the beginning of every meeting. We would each present the work we accomplished in the past week and whether we completed the assigned tasked items. This structure allowed us to collaborate, stay up to date on the other parts of the project, and hold us accountable for deliverables.

We held numerous work sessions where we would schedule three hour segments and develop the entire time. This allowed us to perform project merges and effectively work on sections that required help from another team member.

4.2 Project Results

In our completed project, we successfully created an interactive visualization that allows the user to investigate a vulnerability scan output and its relevant context. The Vulnerability Visualization (VV) tool compares the output data to Qualys Top 10, Metasploit presence, and CVSS Vector String constituent factors. We provide the results of this context in the form of graphs on our interface in Figure 4.1.

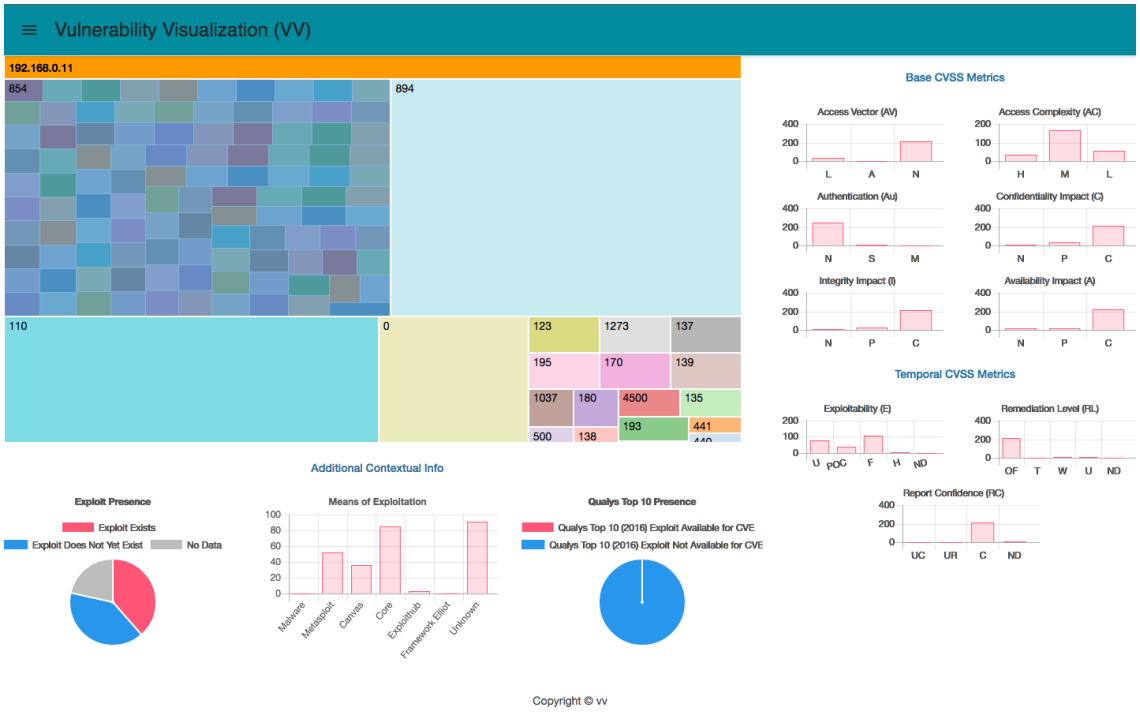

Figure 4.1: *Example of VV in action. The treemap is shown in the top left, with the various contextual graphs surrounding it.*

As you click on a spot on the treemap, it will descend to the next level and update the graphs accordingly. For example, if you clicked the 854 (port) box, it would zoom in and show all of the respective vulnerabilities with that port. It would also update all of the graphs for only that port.

4.2.1 Zoomable Treemap

Implementing a zoomable treemap in React.js was tricky for a of couple reasons. The treemap in D3.js is completely rendered on the DOM. So as the page is loaded, technically the page does not contain the treemap. But shortly after the page is loaded, JavaScript functions will manipulate the DOM and animate it immediately. If any clicks occur, all of the zooming is done by manipulating the DOM in real time. This proves difficult in React because React uses a virtual DOM to massively improve performance and simplicity of large systems.

React modifies the virtual DOM by calling render when state variables are changed and/or the forceupdate() function is called. This made it difficult to implement zoom because a typical D3 treemap would start with a null state and then build itself after it was rendered. Since we had to render the tree immediately, it required us to design the treemap a little differently.

In addition to the DOM issues, in React, if you click on a component, regardless of the children below, it will only call back to the component that was clicked. In our design, the Treemap component is the entire treemap, with children square components directly below. Since we had a

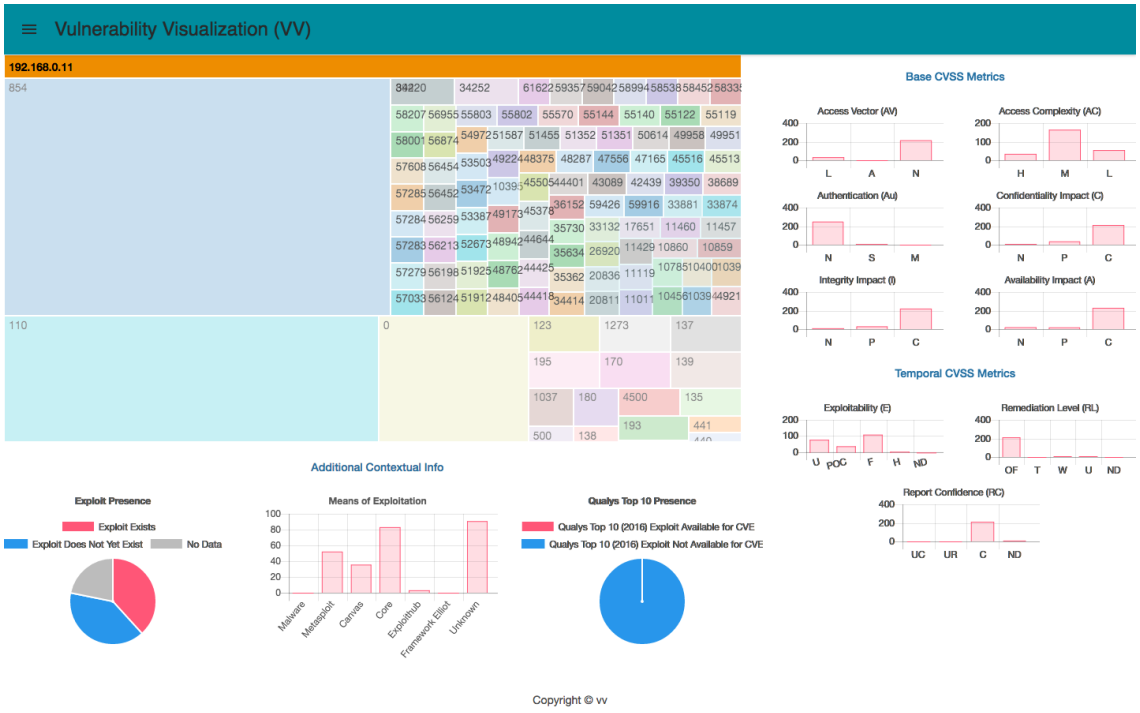

Figure 4.2: *Example of VV's Zoom Out during an animation. In this example, we are exiting the top right square and returning to the initial tree before it. However, since the square still has colored and numbered squares, it is easy to see where we came from.*

hard time finding the location of the clicks and zooming to the right square, we decided to create children components to represent all of the squares underneath.

To summarize our design, our treemap component creates the treemap and receives the data from our parser. It then builds a grandparent component, followed by treemap children, and then sets up the SVG properly. So, once the initial treemap is made, if you click a location, we send that child component back to the treemap via callbacks, and use that child's data as a new treemap. Using that new data we can build the treemap that is directly below, simulating a "zoom". Each component also saves the data of its parent, so by leveraging the parent data, when grandparent is clicked, we use a callback to treemap, and use the parent as the new data. Once that data is changed, the state changes as well causing React to re-render the page.

Finally, since we are simulating the zoom by building a new treemap with the child data, we lost the zoom and unzoom D3 transitions. This was a unique feature of the NV treemap that allowed the user to easily see where he/she traveled to and from. This was a difficult problem to solve in React because render() simply builds the entire frame at once when data changed. We got around this by using a framework called FlipMove, which essentially investigates the previous state (before zoom) and the new state (after zoom) and simulates an animation of the previous state morphing into the new state. Fortunately, due to this framework, we were able to restore the zoom animation feature, as is displayed in Figure 4.2.

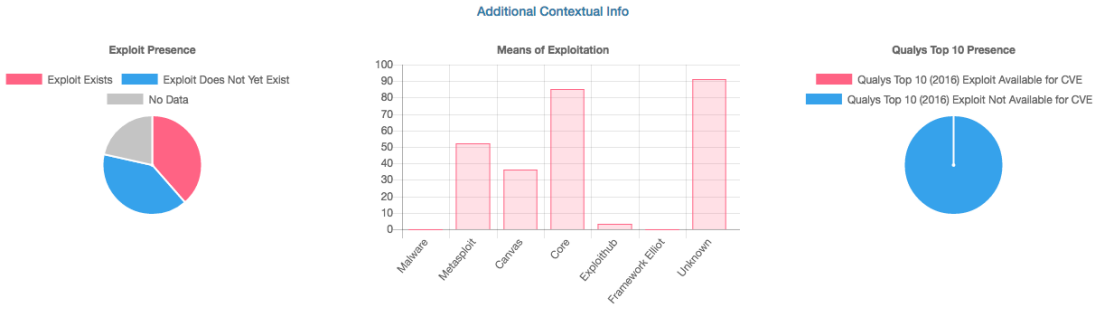

Figure 4.3: *Example of VV's graphs displaying the additional contextual information. This information includes exploit presence, the means of exploit if one is available, and how many vulnerabilities in the scan are a Qualys Top 10 Exploit.*

4.2.2 Graphs

Surrounding our treemap we have graphs for contextual information, base CVSS metrics, and temporal CVSS metrics. These all provide valuable insight to the security analysts investigating vulnerabilities. All of these graphs are aggregate totals on all of the vulnerabilities in the currently visible level of the treemap. They provide a good way to see the distribution of specific attributes of the vulnerabilities. You can see the graphs in Figures 4.3 and 4.4.

When the treemap is traversed (you move to another level) the aggregate totals are updated for only the vulnerabilities currently in the treemap. For example, if you were at a specific IP address and then clicked a port, the graphs would update for only the vulnerabilities with that specific port. This allows analysts to see the distribution of these data values in specific groups of vulnerabilities.

4.2.3 Vulnerability List

In addition to our main page, for convenience you can go to the "Vulnerability List" page by clicking the menu in the top left corner. This page provides a large table of all the vulnerabilities. Not only can you view all of the vulnerabilities, but you can see all of their attributes in each column. A screenshot of the Vulnerability List page can be seen in Figure 4.5.

Similar to Excel, if you click on a column header, you have the option of sorting all of the vulnerabilities by the specific attribute. In addition, as seen in the figure, the user can search for vulnerabilities by attribute. On an attribute (column) there is filter input with the options of "contains", "equals", "not equals", "starts with", and "ends with". This allows security analysts to quickly find specific vulnerabilities they need information about. This feature was intended to compensate for the lack of granular data searching that visualization incurs. In the next chapter, we dive deeper into the VV tool through evaluation of its capabilities.

Figure 4.4: *Example of VV's graphs displaying the CVSS metrics. Both Base and Temporal CVSS Metrics are extracted from their respective CVSS vector strings in each vulnerability report item from a .nessus scan.*

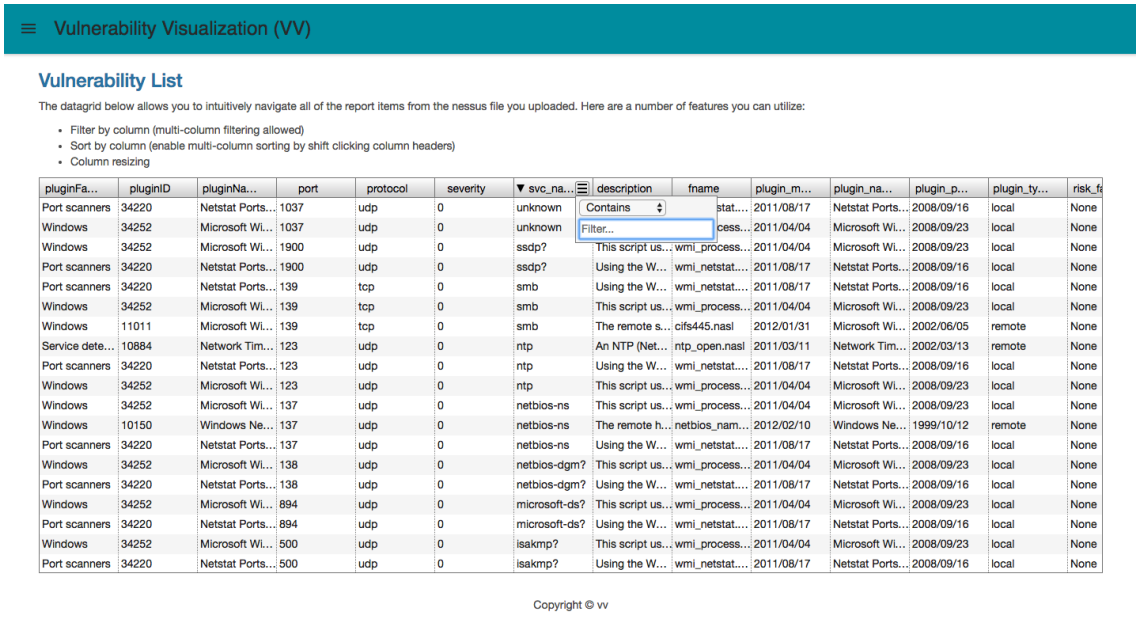

Figure 4.5: *Example of what VV's Vulnerability List page looks like with a fairly large vulnerability scan. The Vulnerability List feature is best suited for analysts that are looking for specific vulnerabilities that they already know some information about. Multi-column sorting and filtering functionality makes it easy to do this.*

Chapter 5

Evaluation

After completing the build phase of our tool, we evaluated it's effectiveness through a number of methods. Our main goal for this project was to use context to address the issue of ineffective prioritization that expends valuable analyst time. Thus, when evaluating our tool we kept this goal in mind, and looked for functionality that supported it. As one survey respondent described, "[Vulnerability] management is a terrible process to have to do. It is broken and inefficient but is required. Creating an easier way would be worth its weight in gold." This project was just that: an attempt to make vulnerability management easier through more accurate and intuitive prioritization.

To begin our evaluation, we judged the general effectiveness of our VV tool based on its overall usability and features included that were requested in the survey and found in our other research methods. For one, the tool is still very easy to navigate like the original NV tool. All visualizations are kept in a single page view with almost no scrolling needed. Every visualization is clearly labeled and drawn with bold colors that make them easy to distinguish, and each visualization provides tooltips (or in the case of the treemap, a functionally equivalent description page at the deepest level of each vulnerability) for greater description of the quantitative attributes being viewed. The website was designed with simplicity in mind, so that the analyst spends as little time as possible trying to figure out where the information they needed is. As much of our research indicated, a speedy response is absolutely key to mitigating the potential damage caused by a vulnerability.

Many of the survey respondents indicated that they would use visualizations to help illustrate trends and metrics in the vulnerability data, therefore we included visualizations to facilitate just that in our project. In particular, the context charts included around the treemap all default to displaying data from all vulnerabilities found in the current scan. This allows users to see

the overall trends of things such as CVSS base and temporal metric distribution, means of exploitation and/or presence of exploit kits, and labeling as a Qualys Top 10 Exploit for a given year. Additionally, context was repeatedly mentioned in the materials we investigated during our preliminary research, by the people we surveyed, and at the IEEE Symposium - another compelling reason for us to include the context charts with the original treemap to help improve analyst prioritization.

Finally, a large portion of this project that can be easily overlooked is our foray into new technologies during our design and build phases. While the original NV tool was built using Backbone.js, and easily could have been expanded upon, we decided to start from the ground up using React.js. While described in more detail in the previous chapter, we determined that React works very well when combined with D3, as React effectively modularizes complex D3 visualizations (like the zoomable Treemap) into simple, reusable pieces. This makes our tool's visualizations very extensible and the subject of future improvement.

5.1 Case Study: VV vs. NV

To better judge our tool's performance, we evaluated the use of both the NV and VV tools to visualize the same network scan data. The data file we used was the testNetworkOpen.nbe data file, the same file used as the default for the online NV web tool. This scan contained multiple hosts with a wide variety of data, and thus was a good average case scan to test a wide swath of each tool's functionality. As the VV tool was developed to work with Nessus V2 file types (.nessus files), and NV uses V1 (.nbe files), we built a converter to transform testNetworkOpen.nbe into a .nessus file that could be used by the VV tool. One limitation of this method is that the V2 file type contains more information about each vulnerability than V1 files do, so even after converting it, the testNetworkOpen file will not be able to utilize the full functionality of the VV tool. For instance, testNetworkOpen does not include any CVSS temporal vector strings or exploit presence information.

VV was designed to include much of the core functionality of NV that made it useful to analysts (e.g. zoomable treemap, vulnerability details, etc.), but also to include the context that would help with prioritization. A high level view of the features preserved and added when comparing the two tools is presented in Figure 5.1.

When loading the file into **NV**, the following is a likely order of actions an analyst might take:

1. Analyst is on a time limit and needs to find the most important vulnerabilities to their particular organization and remediate them.

2. Analyst loads data into NV tool with default settings.

3. Analyst uses treemap to find boxes with darkest coloring (indicating highest CVSS score by default).

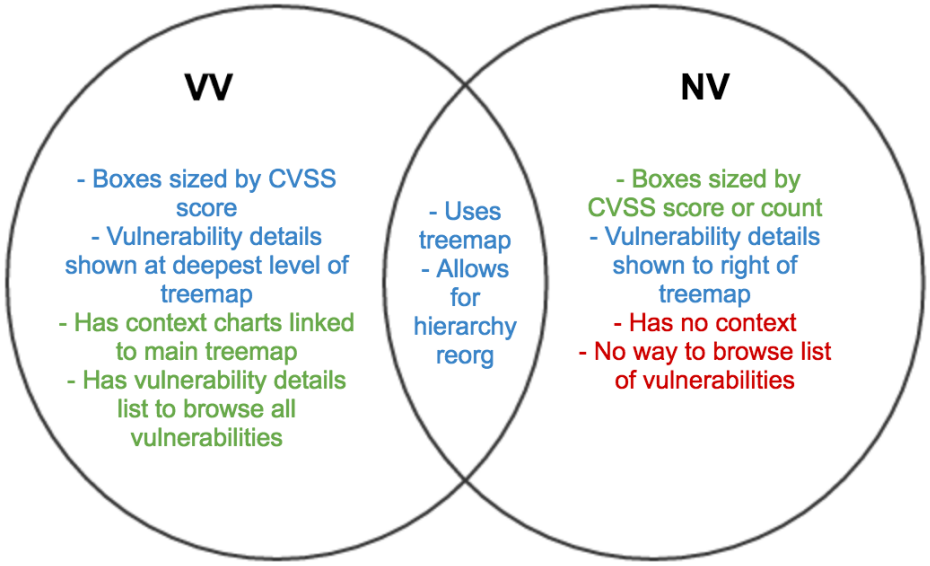

Figure 5.1: *Comparison of the new VV tool versus the previous NV tool. Advantages are marked in green, standard functionality is marked in blue, and disadvantages are marked in red. As can be seen, much of the functionality from NV was preserved in VV, with new helpful features like context charts added.*

4. Analyst mouses over darkest boxes to read descriptions of high threat vulnerabilities and takes note of them.

5. Analyst can use hierarchy reorganization pills above treemap and/or aggregate CVSS/vuln/hole/hote bar charts below treemap to reorganize treemap into different hierarchies based on the their priorities.

When loading the file into **VV**, the following is a likely order of actions an analyst might take:

1. Analyst is on a time limit and needs to find the most important vulnerabilities to their particular organization and remediate them.

2. Analyst loads data into VV tool with default settings.

3. Analyst uses treemap to discover boxes with the largest size indicating high CVSS score.

4. Analyst looks to the charts to the right of the treemap indicating aggregate CVSS base metrics and temporal metrics for the current treemap scope.

5. Analyst looks to the charts below the treemap indicating a number of contextual variables (e.g. exploit presence) relating to the current treemap scope.

6. Using the context variables provided, analyst can more intelligently navigate through treemap to find vulnerabilities of interest (which may not necessarily have the highest CVSS score).

7. Analyst can read details of a vulnerability upon reaching the bottom depth of a treemap box.

8. Analyst can reorganize hierarchy of treemap as needed, like in NV.

9. If the analyst finds that rather than observe/explore trends through visualization, they need to find a specific type of vulnerability (e.g. certain CVE), they can use the Vulnerability List page to filter/sort all vulnerabilities from the scan.

Obviously, this is just an assumption of the likely actions a user may take; an analyst's priorities - and therefore actions - may differ vastly from what is described here. However, the key takeaway is that use of the context charts in VV greatly augments the prioritization capabilities of an analyst than using the standard treemap alone. We explore this further by doing side-by-side analysis of the two tools.

We first compare the main treemap views of both tools in Figure 5.2. Both treemaps are functionally similar, and allow for navigation of the hierarchical data in an almost identical fashion. The major difference between the two views are the smaller charts surrounding the treemap. The NV tool includes four small bar charts below the treemap that show the distribution of vulnerability CVSS score, vulnerability type, top holes, and top notes. These charts can be clicked on to rearrange the treemap respectively. The VV tool lacks these charts, and instead includes charts for contextual data. Instead of clicking on the each chart to rearrange the treemap, the context charts update to display data of the currently displayed treemap level. For instance, when at the highest treemap level, the context charts show data for the entire scan. Then, when the user clicks on a box to zoom in on within the treemap, the context charts update to show data for only that box. We believe that seeing how the distribution of contextual variables change at different levels of the treemap is very useful for an analyst.

The second comparison of the two tools is shown in Figure 5.3. Here we show the lowest level view of the NV treemap, in other words, the treemap cannot descend any further. The NV tool stops at the port level, and will show all the vulnerabilities associated with that port. The VV tool can go one level deeper, and allows display of a single vulnerability. While the user must mouse over the vulnerability they wish to view details for in the NV tool, the VV tool displays details when viewing the lowest level, as shown above. Additionally, it is worth noting how the bar charts in the NV tool portion of Figures 5.2 and 5.3 are exactly the same, whereas the context charts in the VV tool portion have changed between the two pictures.

This comparison between the two tools helps highlight the additional functionality the VV tool has which allows it to offer greater assistance to analysts for prioritization of vulnerabilities.

Figure 5.2: *Side-by-side comparison of the main treemap view of the NV tool (top) and VV tool (bottom), separated by the red line. The NV tool and VV tool share very similar functionality treemaps, albeit with different color schemes. The NV tool has a few bar charts indicating distribution of holes, notes, and vulnerability type / severity in the scan below the treemap. The VV tool has it's contextual information contained within a number of small charts surrounding the treemap.*

Figure 5.3: *Side-by-side comparison of a deep level treemap view of the NV tool (top) and VV tool (bottom), separated by the red line. At this point, the NV treemap cannot descend any further. The NV tool shows all the vulnerabilities for the given port, and details can be viewed via mouseover. The VV tool goes one level deeper, and allows viewing of a single vulnerability, where details are shown in the treemap box. Also, context charts update to represent the data currently in view for the VV tool. The NV tool bar charts do not change in this manner.*

Chapter 6

Conclusion

The goal of this project was to combine multiple sources of contextual information regarding vulnerabilities into a single, cohesive view that would allow analysts to quickly sift through greater amounts of meaningful information to improve their prioritization and remediation process. While there were plenty of challenges uncovered through our research work, we chose to tackle the problem of context due to its far-reaching implications on the security of organizations of all sizes. Contextual information of a vulnerability is key to making informed decision on how to deal with it; with only a simple CVSS score, analysts often incorrectly prioritize how they should deal with vulnerabilities in their systems.

Over the course of this project, our team successfully accomplished the key milestones we set out to complete since we began this undertaking. These achievements are:

- Gathered valuable industry opinions on the shortcomings of modern vulnerability management software and desired qualities in ideal vulnerability management software.
- Confirmed that the need for context for effective prioritization is one of the biggest challenges facing the vulnerability management field. Researched, identified, and utilized contextual variables for the purpose of more effective prioritization of security vulnerabilities.
- Created a modern web application tool using React + D3 for vulnerability analysts, based on a proven-effective existing tool, NV. Our application uses a single page to communicate all pertinent information of the vulnerabilities present in a scanned system to the analyst, reducing navigation time. The main focal point of the tool is a treemap visualization to effectively represent the hierarchical data of network scanning software output, with contextual charts on the sides to augment it.

Through the achievements above, we believe that our team has made a strong foray into finding an extensible, flexible, and robust solution to the ubiquitous problem of vulnerability context scarcity. However, our tool has plenty of room to improve before it can fill the substantial role of a one-stop solution. We detail these areas for improvement in the next section.

6.1 Future Work

Much was accomplished during our limited project time period, yet much more can still be done to improve upon our work. We have identified a number of areas of improvement:

6.1.1 More Sources of Context

In the VV tool, our team decided to draw from a few different sources to provide the contextual information next to the treemap chart. Our first source of context is the Qualys Top 10 Vulnerability List for 2016, a list of the top 10 most critical vulnerabilities of 2016 as decided by Qualys. Our second source is querying the Metasploit database to determine if the vulnerability in question is registered and/or has an available exploit kit. We also check for the vulnerability's presence in other exploit databases to provide a more comprehensive picture of exploitability. Finally, our last source is the constituent metrics used to calculate the CVSS score of a vulnerability, in particular the 6 base metrics and 3 temporal metrics. These metrics are often abstracted away by other programs to provide the CVSS score for vulnerabilities, but they provide valuable contextual information that can be of great use to analysts.

Time constraints prevented us from providing as large a suite of context as we would of liked. With the prototype dashboard now functional, however, future teams who may work on this can leverage the powerful React + D3 framework and simply add in more context once they discover it. Future work should focus on spending time surveying and researching more information to discover new, useful sources of context. This could include discovering more research/technical papers, interviewing analysts from the industry, and/or user studies.

6.1.2 Discover What Views are Most Useful

What is more useful, a single page with the treemap and 3 context bar charts? One large page with just the treemap and another page with all the context charts? One large scrolling page with the treemap and context? These are the questions that still need to be answered, as our team did not have enough time to test multiple view types and determine which was most effective.

Future work would include doing a deep dive into UX/UI of the application we developed to discover what is the most effective layout for communicating the information that analysts are interested in knowing. As discussed before, two of the challenges facing the industry are working quickly but also having access to multiple sources of information. Finding a way to tackle both of these challenges at the same time could be the subject of much future work. Allowing a

customizable dashboard (where context are widgets you can drag and drop) could be a very useful addition that allows analysts to arrange things as they see fit. Since every company is different, this customizability helps account for the differing needs of each analyst.

6.1.3 Expand Functionality of Vulnerability List

The vulnerability list is the portion of the application that lists every individual vulnerability discovered in the chosen scan. It is presented as a scrollable table with each vulnerability and every attribute known about it. This feature was added to the application to give users a more granular view of the vulnerabilities in each scan, as well as have greater search/sort functionality that the visualizations could not offer.

Currently, there is a lot of information made available by the list feature to comb through, and it doesn't differentiate itself much from the raw scan report other than being in a table view. There are very powerful sort-by-column and filtering capabilities, but it would be helpful to users to offer preset filters and sorts (e.g. sort by severity, then filter by solution available, then sort by date found). Allowing users to save presets themselves would also be a welcome addition. This would save users time and headache trying to find the right combination of filters and sorts to find what they are looking for. Another addition to consider is offering export options into new files (e.g. csv, original json file, rich text, etc.). These downloaded files could optionally retain the applied sorts and filters. Finally - and this holds true for the other portions of the VV tool as well - a UI upgrade could make information more readable.

Appendix A: Vulnerability Management Survey

This page is intentionally left blank.

Cyber Vulnerability and Data Visualization Survey

Your responses to these questions will remain anonymous, and will help our research into vulnerability management.

Please take the time to answer all questions honestly and to the best of your ability.

Thank you!

* Required

1. **How would you categorize the size of your company?** *
   Mark only one oval.

   - ◯ small (under 300)
   - ◯ medium (under 2000)
   - ◯ large

2. **Briefly describe how many resources your company dedicates to security. This could be number of people/departments, overall percentage of workforce, etc.** *

   .....

   .....

   .....

   .....

   .....

3. **What industry is your company a part of?** *

   .....

4. **Do you have a vulnerability management process?** *
   Mark only one oval.

   - ◯ Yes *Skip to question 5.*
   - ◯ No *Skip to question 21.*

Vulnerability Management

5. **What role do you play in the incident response process?**
   Mark only one oval.

   - ◯ analyst/engineer
   - ◯ team/shift manager
   - ◯ executive in charge of the process
   - ◯ Other: _____

6. **Briefly describe the workflow of your vulnerability management process.** *

   _____

   _____

   _____

   _____

   _____

7. **Approximately how many endpoints/servers does your organization scan for vulnerabilities?** *

   _____

8. **Does your organization scan subsets of networks?** *
   Mark only one oval.

   - ◯ Yes
   - ◯ No

9. **What tools and software do you use to scan and/or manage vulnerabilities?** *
   Network Scanners Include: Nessus, OpenVAS, Nexpose, QualysGuard, Core Impact, SAINT, etc. Feel free to list any other tools that you may use that don't necessarily fall into the Network Scanners category.

   _____

   _____

   _____

   _____

   _____

10. **Do you prefer the use of one scanning tool over another? Why?** *

11. **Do you consider the number of vulnerabilities in your organization large and difficult to manage/prioritize?** *

Mark only one oval.

- Yes
- No

12. **Describe how you plan and prioritize fixes for discovered vulnerabilities?** *

13. **When analyzing a vulnerability scan, describe the information you bring in to contextualize your analysis.** *

Examples include websites with detailed vulnerability information, business / users of the affected IPs, etc.

14. **If two vulnerabilities, with similar severity, are present on different parts of the network, how do you decide which to pursue first? What data do you look for to make this decision?**

15. **How often does your organization run vulnerability scans in general?** *
Mark only one oval.

- Daily
- Weekly
- Monthly
- Annually

16. **Does your organization scan high risk areas with valuable assets more often than other network locations?** *
Mark only one oval.

- Yes
- No

17. **How long does it typically take your analysts to identify vulnerabilities they intend to patch after running a scan?** *
Mark only one oval.

- Less than a day
- Less than a week
- Less than a month
- Less than a year

18. **Do you use CVSS scores in your analysis? If so, how do you use these scores in your vulnerability management process? Do you use any other indicators of severity?** *

19. **What features does your current toolset and vulnerability management workflow lack that you would like to have?** *

.....

.....

.....

.....

.....

20. **Do you have a designated technical team specifically focused on vulnerability management?** *

Mark only one oval.

- ◯ Yes
- ◯ No

Security Resources

21. **Where do you get information on security threats or vulnerabilities that your company faces?** *

Mark only one oval.

- ◯ Traditional Media/News
- ◯ Social Media
- ◯ Consortium/ISAC
- ◯ Personal network
- ◯ Vendors
- ◯ Coworkers
- ◯ Other: .....

22. **Why don't you employ vulnerability management? (choose all that apply)** *

Check all that apply.

- ☐ Too expensive
- ☐ Too time consuming
- ☐ Not enough man power
- ☐ Don't know where to begin
- ☐ Don't have any knowledge on vulnerability management
- ☐ Other: .....

General Questions

23. **What would an ideal vulnerability scanning tool do for you? What features does it have? How does it display results?** *

24. **Do you use data visualizations (bar charts, line graphs, etcetera) in any of your vulnerability management tools or processes? If so, how do you use visualization?** *

25. **Have you ever used an attack graphs or topological graph to visualize threats to your systems? If so, how do you use these tools?** *

26. **Have you considered using ai/machine learning techniques to identify likely attack paths? Why or why not?** *

27. **Is there anything else about vulnerability management you'd like to share with us? Feel free to also share your comments on the survey and/or why you answered a question the way you did.**

BIBLIOGRAPHY

[1] J. Supra, "Alert: 2016 verizon data breach investigations report," 2016.

[2] H. Holm, T. Sommestad, J. Almroth, and M. Persson, "A quantitative evaluation of vulnerability scanning," *Information Management & Computer Security*, vol. 19, no. 4, pp. 231–247, 2011.

[3] D. Evans, P. Bond, and A. Bement, "Fips pub 199 standards for security categorization of federal information and information systems," *The National Institute of Standards and Technology (NIST)*, 2004.

[4] P. Institute, "2016 cost of data breach study: Global analysis," Ponemon Institute, Tech. Rep., June 2016. [Online]. Available: http://www-03.ibm.com/security/data-breach/

[5] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia, *An Attack Graph-Based Probabilistic Security Metric*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, vol. 5094, pp. 283–296.

[6] "Cvss v3.0 specification document." [Online]. Available: https://www.first.org/cvss/specification-document

[7] A. Telea and Safari Books Online, *Data visualization: principles and practice*, 2nd ed. Boca Raton: CRC Press, Taylor & Francis Group, 2015.

[8] CDC, "Vulnerability management life cycle," accessed: 2017-03-21. [Online]. Available: https://www.cdc.gov/cancer/npcr/tools/security/vmlc.htm

[9] O. I. de Normalización, *ISO/IEC 27005: Information technology-Security techniques -Information security risk management*. ISO, 2008. [Online]. Available: https://books.google.com/books?id=K1HbZwEACAAJ

[10] W. Shanks, "Building a vulnerability management program - a project management approach," SANS, techreport, March 2015. [Online]. Available: https://www.sans.org/reading-room/whitepapers/projectmanagement/building-vulnerability-management-program-project-management-approach-35932

[11] T. Palmaers, “Implementing a vulnerability management process,” SANS, techreport, March 2013. [Online]. Available: https://www.sans.org/reading-room/whitepapers/threats/implementing-vulnerability-management-process-34180

[12] W. Kandek, *Vulnerability Mangement for Dummies*, 2nd ed. John Wiley & Sons, Ltd, 2015. [Online]. Available: https://www.qualys.com/docs/vm-for-dummies-2nd-edition.pdf

[13] S. A. Merrell and J. F. Stevens, “Improving the vulnerability management process,” *EDPACS*, vol. 38, no. 1, pp. 13–22, 2008. [Online]. Available: http://dx.doi.org/10.1080/07366980802138673

[14] E. Bellis, “Common vulnerability management mistakes to avoid,” accessed: 2017-02-06. [Online]. Available: https://www.kennasecurity.com/resources/common-vulnerability-management-mistakes/

[15] O. Rochford, “Prioritizing vulnerability prioritization,” August 2013, accessed: 2017-02-06. [Online]. Available: http://www.securityweek.com/prioritizing-vulnerability-prioritization

[16] AlienVault, “Vulnerability management: Think like an attacker to prioritize risks,” 2014. [Online]. Available: http://bluekarmasecurity.net/wp-content/uploads/2014/09/AlienVault_Vulnerability-Management_Think-Like-An-Attacker-To-Prioritize-Risks_whitepaper.pdf

[17] D. Kelley, “Remediating it vulnerabilities: Quick hits for risk prioritization,” September 2011. [Online]. Available: http://searchsecurity.techtarget.com/tip/Remediating-IT-vulnerabilities-Quick-hits-for-risk-prioritization

[18] P. Mell, T. Bergeron, and D. Henning, “Creating a patch and vulnerability management program,” *NIST Special Publication*, vol. 800, p. 40, 2005.

[19] Z. Khalil and M. Elammari, *Vulnerability Scanning & Management: An Approach to Managing the Risk Level of a Vulnerability*, 2015.

[20] J. McPherson, K.-L. Ma, P. Krystosk, T. Bartoletti, and M. Christensen, “Portvis: A tool for port-based detection of security events,” in *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, ser. VizSEC/DMSEC ’04. New York, NY, USA: ACM, 2004, pp. 73–81. [Online]. Available: http://doi.acm.org/10.1145/1029208.1029220

[21] K. Lakkaraju, W. Yurcik, and A. J. Lee, “Nvisionip: Netflow visualizations of system state for security situational awareness,” in *Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security*, ser.

VizSEC/DMSEC '04. New York, NY, USA: ACM, 2004, pp. 65–72. [Online]. Available: http://doi.acm.org/10.1145/1029208.1029219

[22] X. Ou, W. F. Boyer, and M. A. McQueen, "A scalable approach to attack graph generation," in *Proceedings of the 13th ACM conference on Computer and communications security*. ACM, 2006, pp. 336–345.

[23] L. Harrison and A. Lu, "The future of security visualization: Lessons from network visualization," *IEEE NETWORK*, vol. 26, no. 6, pp. 6–11, 2012.

[24] A. Singhal, X. Ou, N. I. of Standards, and T. (U.S.), "Security risk analysis of enterprise networks using probabilistic attack graphs," U.S. Dept. of Commerce, National Institute of Standards and Technology, Tech. Rep., 2011.

[25] S. Jha, O. Sheyner, and J. Wing, "Two formal analyses of attack graphs," vol. 2002-. IEEE, 2002, pp. 49–63.

[26] C. Eiram and B. Martin, "An open letter to first."

[27] B. Marr, "Big data," 2015. [Online]. Available: http://www.forbes.com/sites/bernardmarr/2015/09/30/big-data-20-mind-boggling-facts-everyone-must-read/#622c8b7c6c1d

[28] 2016. [Online]. Available: http://review.content-science.com/2016/07/2016-big-and-small-data-fact-sheet/

[29] G. Conti, B. I. P. Collection, E. A. Complete, and I. Books24x7, *Security data visualization: graphical techniques for network analysis*. San Francisco: No Starch Press, 2007.

[30] B. Johnson and B. Shneiderman, "Tree-maps: a space-filling approach to the visualization of hierarchical information structures." IEEE Computer Society Press, 1991, pp. 284–291.

[31] T. v. Landesberger, A. Kuijper, T. Schreck, J. Kohlhammer, J. J. . J. Wijk, van, J. Fekete, and D. Fellner, "Visual analysis of large graphs : state-of-the-art and future research challenges," *Computer Graphics Forum*, vol. 30, no. 6, pp. 1719–1749, 2011.

[32] Y. Wang, S. T. Teoh, and K.-L. Ma, "Evaluating the effectiveness of tree visualization systems for knowledge discovery," 2006.

[33] L. Harrison, R. Spahn, M. Iannacone, E. Downing, and J. Goodall, "Nv: Nessus vulnerability visualization for the web." ACM, 2012, pp. 25–32.