Analyzing Evolving Trends of Vulnerabilities in National Vulnerability Database

Mark A. Williams¹, Sumi Dey², Roberto Camacho Barranco¹, Sheikh Motahar Naim¹, M. Shahriar Hossain¹, and Monika Akbar¹
¹Department of Computer Science, ²Department of Computational Science
University of Texas at El Paso
El Paso, Texas, USA
Emails: {mawilliams7, sdey2}@miners.utep.edu, rcamachobarranco@utep.edu, snaim@miners.utep.edu, mhossain@utep.edu, makbar@utep.edu

Abstract—As the world approaches a state of greater dependence on technology, many products face increasing threats from malicious attackers who are attempting to take advantage of vulnerabilities in software design. Most of the known vulnerabilities are already aggregated, stored in text format, and are readily accessible to the public, making such an aggregated database a prime corpus for analysis using data mining methods. A multitude of research efforts have been deployed to analyze individual aspects of such cyber-security corpora to create taxonomies, assess vulnerability impact, and even predict future vulnerabilities. However, minimal effort has been committed to analyze cyber-security corpora to explore correlations between vulnerabilities and study the evolution of a vulnerability from its genesis. In this paper, we propose an integrated data mining framework to automatically lay out how vulnerabilities develop over time and detect the evolution of a specific cyber-security threat. We use (1) a Supervised Topical Evolution Model (STEM), which discovers temporal themes from a text corpus and (2) a diffusion-based storytelling technique that sifts through past vulnerability reports to describe how a current threat evolved. The STEM gives a holistic evolution structure of the vulnerabilities, while diffusion-based storytelling provides the precise genealogy of a specific threat. A considerable series of experiments demonstrate that the proposed framework can discover evolutionary patterns in today's most pressing vulnerabilities with a high degree of precision. As case studies, we explore the development of vulnerabilities in certain products, providing a unique insight into the correspondence between seemingly unrelated vulnerabilities and the impact of that correspondence on overall software security.

Index Terms—cyber-security, vulnerabilities, temporal topic modeling, storytelling

I. INTRODUCTION

A high degree of dependence on software products in all sectors — government, private, and academia — necessitates acquiring a great deal of knowledge regarding software vulnerabilities. Luckily, the growing volume of vulnerabilities are meticulously recorded by many software development companies including operating system providers like Microsoft and Apple. The U.S. Government has an ongoing effort to consolidate all the reported software vulnerabilities in a standard database called National Vulnerability Database (NVD)¹.

¹National Vulnerability Database (NVD), https://nvd.nist.gov

Figure 1: The product susceptibility of ten major software products since 2000 based on STEM model.

Despite containing years' worth of indispensable vulnerability information on every major software product, such a large corpus of structured data remains largely neglected due to a lack of tools and algorithms to support extensive analysis of vulnerabilities. Consequently, there is a considerable gap in knowledge of vulnerability interactions, trends, and evolution as well as overall product susceptibility to vulnerabilities. Understanding the vulnerability trends would pave the pathway of researchers and industry experts to develop more secure systems, mitigate the impact of existing vulnerabilities, as well as guide emerging research in the field of cyber-security.

Consider that a team of industry experts is trying to determine the proportion of company resources that should be allocated to cyber-defense by analyzing how susceptible their products are to vulnerabilities. In this case, a concise theme-based model based on an aggregate of vulnerability data would provide the expert with ample information to aid in making a decision. Figure 1 describes the product susceptibility of ten major software products based on the probability of topics...
over time in the National Vulnerability Database. A brief analysis of this figure reveals, among other trends, the drastic decline of the susceptibility of Internet Explorer in recent years, but a sudden rise in susceptibility for Windows 10 and Edge about three years ago. The team of experts will be able to recommend an area of focus based on a figure similar to Figure 1.

Now, consider the scenario where the team is performing a detailed analysis regarding a Windows 10 vulnerability — e.g., the operating system mishandles library handling allowing users to gain privileges to execute arbitrary codes. Understanding how such a vulnerability evolved despite monitoring multiple earlier versions of the product would allow the team to resolve the issue in the current version as well as in any other related versions or related products. Figure 2 shows how the vulnerability described evolved over time from another issue involving kernel mode drivers that affected an earlier version of Windows. We explain Figure 2 later in Experimental Results as part of Section V-B as a case study.

In this paper, we use two analytical approaches to aid decision making using vulnerability corpora. The first approach utilizes a temporal topical model that builds on top of our previous effort called Supervised Topical Evolution Model (STEM) [10]. Outcomes such as Figure 1 can be generated using STEM. STEM provides a holistic idea about the vulnerability topics in the dataset. The second approach leverages another of our efforts called Diffusion-based storytelling [1], by providing the ability to describe how a specific threat evolved historically. Overall, STEM is the first stage of the analytic process to better understand the trends in vulnerabilities. Diffusion-based storytelling is the second stage which provides finer details of the evolution of a specific threat.

In summary, the contributions of this paper are:

1) We provide a high-level holistic view of cyber-security vulnerabilities using a probabilistic graphical model, STEM, that integrates timestamps, annotations, latent themes, and textual data.
2) Through a variety of experiments on a cyber-security corpus, we demonstrate that diffusion-based storytelling can establish evolutionary narratives describing the propagation of vulnerabilities over time from one product to another and between different versions of the same product.
3) We determine the overall susceptibility of major software products to vulnerabilities within a specified time frame, and explore the distribution of susceptibility within distinct product domains.

II. RELATED WORK

There are existing efforts to implement broader data mining algorithms on cyber-security corpora. In fact, data mining algorithms have been increasingly utilized to facilitate knowledge gathering in the cyber-security domain in recent years. Vulnerability prediction has been addressed in many efforts. For instance, text mining techniques are applied by Scandariato et al. to predict software component vulnerabilities in [15]. Han et al. [5] develop a method to predict the severity of a vulnerability by using a one-layer convolutional neural network. Zhang et. al [18] use data mining techniques to try to predict day zero software vulnerabilities using the NVD.

Researchers have also targeted analysis of vulnerabilities through detection of relationships between vulnerabilities [14], [20]. Lin et al. [9] present an association rule discovery algorithm to find correlations between certain aspects of a cyber-security corpus, the Common Weakness and Enumeration database. Shin et al. [16] discover correlations between vulnerabilities by analyzing code complexity, code churn, and developer activity metrics. Such correlations are extended beyond code content to a vulnerability life cycle in [8]. Topic models are used extensively in [11] to discover frequent vulnerability types and new patterns in cyber-security corpora.

One significant limitation of these works is that they do not take into account the temporal aspect of vulnerabilities. This makes it difficult to determine if the relationship discovered by the work is relevant in the present. Our proposed framework is distinct in this manner because we discover trends in vulnerabilities based on the latent structure of the corpus and by incorporating the temporal feature of vulnerability reports. We then go one step further and study the evolution

---

Figure 2: How a security threat in Windows, related to library-mishandling giving all users the capability of executing arbitrary code, evolved over time. The document ID in NVD is CVE-2015-6128.
of a vulnerability as a series of connected vulnerability reports.

III. PROBLEM AND SOLUTION STATEMENT

We have a set of vulnerability reports \( D = \{d_1, d_2, \ldots, d_\|D\|\} \), which contain entities or words from \( E = \{e_1, e_2, \ldots, e_E\} \). Each document \( d \) has a publication date \( t_d \), and a binary distribution of vulnerability labels specific to the document \( d \) from \( \Lambda_d = [l_1, l_2, \ldots, l_K] \), where \( K \) is the number of vulnerability labels. Additionally, each document \( d \) has a list of affected products distinct to the document \( d \) that is a subset of \( \Lambda_d = [a_1, a_2, \ldots, a_P] \) where \( P \) is the number of products referenced in the NVD dataset.

The proposed framework consists of three stages:

1) Given the vulnerability reports \( D \), compute the appearance probability (i.e., topic probability) of each of the \( K \) vulnerability labels in each timestamp \( t \).

2) Given the vulnerability reports \( D \), compute the product recurrence (i.e., product susceptibility) of each of the \( P \) product labels in each timestamp \( t \).

3) Given a specific vulnerability report \( d \), create a chain of relevant vulnerability reports from the past such that the chain reflects the evolution of the vulnerability reported in \( d \).

IV. METHODOLOGY

The National Vulnerability Database (NVD) that we extensively use in this paper contains 111,060 unique reports. The reports describe vulnerabilities and exposures in software products from as early as 1996. The database contains links to the report-documents published by software development companies such as Apple and Microsoft.

Each document in the NVD consists of a Common Vulnerabilities and Exposures\(^3\) (CVE) entry, which is a high-level vulnerability description. Each CVE entry is associated with products that are affected by the vulnerability described in the entry. CVE entries in the NVD vary in length but they are usually short paragraphs. We leverage the links provided with each entry to enrich the data by augmenting NVD reports with descriptions from the original source.

A. Dataset Preparation

To prepare the dataset for our experiments, we used distinct attributes present in each NVD document (publication date, CVE entry description, etc.) and labeled documents as a collection of these attributes. We then extracted entities from each document description in the dataset by using standard entity detection approaches [1], [7]. We ignored documents with less than four entities and documents that were published before 2000 because they do not contain enough information to be discriminative.

Each NVD document is also associated with a label that serves as a classifier for the vulnerability described in the document and a set of products that are affected by that vulnerability. We have selected the 50 most frequent vulnerability labels and the 50 most affected products in our experiments with STEM. All vulnerability labels and affected products are utilized in the diffusion-based storytelling framework.

B. STEM

Supervised Topical Evolution Model (STEM) is a probabilistic graphical approach for modeling dynamic document collections with explicit labels. Similar to traditional topic models like LDA [2], STEM views each document as a mixture of underlying topics, and each topic as a distribution over the words. Unlike traditional topic models, STEM learns the topics and their evolution over time by guiding its inference procedure through the incorporation of timestamps and label information of the documents. The generative process for the model is as follows.

1) For \( k = 1 \) to \( K \) :
   a) \( \phi_k \sim Dirichlet(\beta) \)
   b) For each document \( d \in D \) :
      i) \( \Lambda_{dk} \sim Bernoulli(\gamma_k) \)
      b) Compute \( L_{d} \) from \( \Lambda_d \)
      c) \( \alpha_d = L_d \times \alpha \)
      d) \( \theta_d \sim Dirichlet(\alpha_d) \)
   e) For each entity \( e_d \in d \) :
      i) \( z_{di} \sim Multinomial(\theta_d) \)
      ii) \( e_{di} \sim Multinomial(\phi_{z_{di}}) \)
      iii) \( t_{di} \sim Beta(\psi_{z_{di}}) \)

A list of all the symbols used in the generative process is provided in Table I.

STEM approximates the posterior probabilities of the hid-

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( D )</td>
<td>Set of all the documents</td>
</tr>
<tr>
<td>( V )</td>
<td>Number of unique entities</td>
</tr>
<tr>
<td>( K )</td>
<td>Number of unique labels</td>
</tr>
<tr>
<td>( N_d )</td>
<td>Number of entities in document ( d )</td>
</tr>
<tr>
<td>( \alpha, \beta )</td>
<td>Dirichlet prior for the document-topics and topic-words distributions, respectively</td>
</tr>
<tr>
<td>( \gamma )</td>
<td>Prior probabilities for the Bernoulli distribution of labels</td>
</tr>
<tr>
<td>( \theta_d )</td>
<td>Multinomial distribution of topics specific to the document ( d )</td>
</tr>
<tr>
<td>( \Lambda_d )</td>
<td>Binary distribution of labels specific to the document ( d )</td>
</tr>
<tr>
<td>( L_{d} )</td>
<td>Projection matrix for labels in document ( d )</td>
</tr>
<tr>
<td>( \phi_z )</td>
<td>Multinomial distribution of words specific to topic ( z )</td>
</tr>
<tr>
<td>( \psi_z )</td>
<td>Beta distribution of time specific to topic ( z )</td>
</tr>
<tr>
<td>( z_{di} )</td>
<td>The topic associated with the ( i )-th token in the document ( d )</td>
</tr>
<tr>
<td>( e_{di} )</td>
<td>The ( i )-th entity in the document ( d )</td>
</tr>
<tr>
<td>( t_{di} )</td>
<td>The timestamp associated with the ( i )-th token in the document ( d )</td>
</tr>
</tbody>
</table>

\(^3\)Common Vulnerabilities and Exposures (CVE), https://cve.mitre.org
\[
p(z_{di}|z_{-di}, e, t, \alpha, \beta, \psi) \propto (n_{z_{di}} + \alpha - 1) \frac{n_{z_{di}v} + \beta - 1}{\sum_{v=1}^{V} (n_{z_{di}v} + \beta)} - 1 \\
\times \frac{\psi_{z_{di}, 1}^1 (1 - t_{di}) \psi_{z_{di}, 2}^2 - 1}{B(\psi_{z_{di}, 1}, \psi_{z_{di}, 2})}
\]
(1)

For the sake of speed and simplicity, \( \psi \) is updated after each Gibbs sample using the method of moments as follows:
\[
\hat{\psi}_{z, 1} = \hat{\bar{t}}_z \left( \frac{\hat{t}_z (1 - \hat{\bar{t}}_z)}{\bar{s}_z^2} - 1 \right)
\hat{\psi}_{z, 2} = (1 - \hat{\bar{t}}_z) \left( \frac{\hat{t}_z (1 - \hat{\bar{t}}_z)}{\bar{s}_z^2} - 1 \right)
\]
(2)

where \( \hat{\bar{t}}_z \) and \( \bar{s}_z \) indicate the mean and standard deviation of all the timestamps belonging to topic \( z \), respectively. Refer to [10] for a more detailed derivation.

C. Diffusion-based story generation

In [1], we presented a diffusion-based framework that mines a large corpus of documents to generate smooth evolving stories given a seed document published recently. The framework runs several preprocessing tasks, including candidate generation which reduces the search space to a few hundred documents. Then, via the optimization of an objective function, a storyline is generated which consists of several story segments delimited by “turning points” which contain coherent documents. The main objective is to help the user understand the succession of events or articles that lead to the seed document.

We assume that we have a set of candidate documents \( D = \{d_1, d_2, \ldots, d_D\} \) which contain entities from \( E = \{e_1, e_2, \ldots, e_E\} \) and a publication date \( t_d \). A turning point in the story \( \tau \) in the story \( \tau \in T \) delimits a set of segments \( S \) of coherent documents such that \( S = \{(\tau_1, \tau_2), (\tau_2, \tau_3), \ldots, (\tau_{|T|-1}, \tau_T)\} \). An optimization routine finds the values for every \( \tau \) as well as the probability that a candidate document is relevant to the segment that includes it.

We formulate the objective function for evolution using four terms: incoherence, similarity, overlap, and uniformity. First, we have two penalties that aim to minimize the incoherence of documents within a segment, while also minimizing the similarity across segments. The expectation is that these terms will promote having highly coherent segments that are independent of each other.

\[
incoherence(s) = \frac{\sum_{i,j}^{|D| \times |D|} w_i \ast w_j \ast \Phi \ast \text{soergel}(d_i, d_j) \ast |t_i - t_j|}{\sum_{i,j}^{|D| \times |D|} w_i \ast w_j \ast \Phi}
\]
(3)

\[
similarity(s) = \frac{\sum_{i,j}^{|D| \times |D|} w_i \ast w_j \ast \Phi \ast e^{-\text{soergel}(d_i, d_j)}}{\sum_{i,j}^{|D| \times |D|} w_i \ast w_j \ast \Phi}
\]
(4)

where \( w_i \in \mathcal{W} \) is the weight for document \( d_i \) and
\[
\Phi = \gamma(t_1, t_L^* \tau_L, t_H^*) \ast (t_j, t_L^* \tau_L, t_H^*)
\]
\[
\phi = \gamma(t_1, t_L^* \tau_L, t_H^*) \ast (1 - \gamma(t_j, t_L^* \tau_L, t_H^*))
\]
(5)

where \( t_{L,H}^* \) are the lower/upper turning points for a particular segment and \( \gamma \) is defined as:
\[
\gamma(t, t_L^*, t_H^*) = \begin{cases} 
\frac{1}{\sqrt{2 \pi \sigma^2}} e^{-\frac{(t-t_L^*)^2}{2\sigma^2}} & \text{if } t \leq t_L^* \\
\frac{1}{\sqrt{2 \pi \sigma^2}} e^{-\frac{(t-t_L^*+\sigma^2)^2}{2\sigma^2}} & \text{if } t_L^* < t < t_H^* \\
\frac{1}{\sqrt{2 \pi \sigma^2}} e^{-\frac{(t-t_H^*)^2}{2\sigma^2}} & \text{if } t_H^* \leq t
\end{cases}
\]
(6)

where \( \sigma \) is the standard deviation of a Gaussian distribution that indicates the degree of membership of a timestamp in a segment.

The final objective function (Eq. 7) is minimized for two or more turning points very close to each other.
\[
\text{overlap} = \left( 1 + \sum_{i,j} \sum_{i,j<i} e^{-\frac{(t_i-t_j)^2}{2\sigma^2}} \right)
\]
(8)

where \( \sigma \) is the standard deviation of a Gaussian distribution that defines the sensitivity of the overlap penalty.

A final term is added to avoid having high uniformity in the distribution of the weight probabilities that indicate if the documents are relevant or not. This penalty will make sure that not all of the weights are set to 0 or 1.
\[
\text{uniformity} = \left( 1 + \sum_{s=1}^{|S|} \left( 1 - \left( \frac{\sum_{s=1}^{|S|} \sum_{s=1}^{|S|} w_s \ast w_t^\top \left( \sqrt{|W_s|} - 1 \right) \right)^2}{|W_s| - 1} \right) \right)
\]
(9)

where \( \Gamma_s \) is a vector of values returned by the membership function (Eq. 7 for the documents that fall in segment \( s \)).

The final objective function (Eq. 10) is minimized for two vectors: \( S \) and \( W \). The elements \( s \in S \) are bounded by the range of the turning points and the elements \( w \in W \) are bounded between \([0, 1]\). We use the quasi-newton limited memory algorithm for bound constrained optimization (L-BFGS-B) [19].

\[
\mathcal{F}(T, W) = \sum_{s=1}^{|S|} \left( \text{incoherence}(s) \ast \text{similarity}(s) \ast \text{overlap} \ast \text{uniformity} \right)
\]
(10)

Given a seed document, Eq. 10 results in a set of documents with a vector of importance values \( \mathcal{W} \), which helps in identifying documents from the past that are highly relevant to the seed document and at the same time represent the evolution of the content of the seed document.
V. EXPERIMENTAL RESULTS

In this section we seek to answer the following questions:  
1) How does Supervised Topical Evolution Model (STEM) perform compared to other topic modeling approaches in explaining trends in comparable vulnerabilities? (Section V-A)  
2) How can an expert analyze the evolution of a specific vulnerability of interest? (Section V-B)  
3) How can shifts in product susceptibility to vulnerabilities be detected and explained? (Section V-C)  
4) How well does STEM perform on the NVD dataset in terms of quality and coherence of topics? (Section V-D)  
5) How well does STEM scale in its parallel execution? (Section V-E)

We implemented two versions of STEM — a sequential version and a parallel one. The parallel version of the STEM model uses 12 processors and is denoted as STEM-P for the rest of this section. The L-BFGS-B optimization algorithm [19] minimizes a GPU-oriented parallel version of the objective function for the diffusion-based storytelling framework.

We conducted a series of quantitative and qualitative experiments using these algorithms to answer the preceding questions. We compared our model with two contemporary graphical models — Topics Over Time (TOT) [17], which models topical evolutions over continuous time, and Labeled LDA (LLDA) [13], which uses labels to guide the model inference process — to determine the accuracy of our methods. We utilized a real-world dataset — a cyber security corpus known as the National Vulnerability Database (NVD) — for our experiments.

It is important to note that in our experiments with STEM using products as topics, we refer to topic probability as product susceptibility. Topic probability and product susceptibility are interchangeable in the context of this paper because they reflect the same concept. For example, if the topic Microsoft Edge had high topic probability in 2011, then the topic was highly susceptible in that year because it appeared frequently in the data at that time.

A. Trends in Vulnerabilities

One of the goals of this paper is to study how we can discover connections and trends in vulnerabilities that will help create more effective mitigation strategies and cyber-defense implementations. Usually, this discovery is achieved by analyzing individual aspects of a vulnerability with limited consideration of temporal factors. In the following experiments, we utilize STEM to discover novel correlations in vulnerabilities, with a particular emphasis on both temporal factors and the interplay between vulnerability dependencies. We also compare STEM results with results obtained by LLDA and TOT. Capability-wise, STEM enables both aspects of LLDA and TOT. LLDA is capable of handling labels in topic modeling but does not include time. TOT includes topics over time but does not take labels into consideration. STEM provides a combination of the advantages of both algorithms — labeled topics over time for a massive text corpus.

In Figure 3, we display the top ten highly probable terms discovered by STEM, LLDA, and TOT for vulnerability labels in the NVD dataset. Although we performed this experiment on the ten most prominent vulnerability labels in the dataset, we have displayed only three in the Figure due to space considerations. The complete results for this experiment and the source code for this paper can be accessed on a website\(^4\) we created. The three listed vulnerability labels are: Permissions and Privileges, Path Traversal, and Resources Errors. Overall, the terms inferred by STEM and LLDA are moderately similar, referring to the fact that STEM subsumes the capability of LLDA.

However, the terms discovered by TOT differ significantly from the other models. The reason of such difference is that TOT does not consider vulnerability labels as its topics while the other models do. The three topics — Topic 0, Topic 14, and Topic 13 — listed in Figure 3 (right) are based on mere overlaps of top terms with vulnerability labels Permissions and


![Figure 3: Terms discovered by STEM, LLDA, and TOT for three vulnerability labels in NVD.](image-url)
Privileges, Path Traversal, and Resources Errors as detected by STEM in Figure 3 (left). Overall, 50 distinct topics were found by TOT. Topic 0, 14 and 13 are chosen for TOT because they have the most resemblance with the three columns of STEM in Figure 3.

Figure 3 offers immense insight into the impact of vulnerabilities as well as how they can be exploited in the wild. For example, the Permissions and Privileges vulnerability must be a common vulnerability for android products since android is related to Permissions and Privileges according to the STEM and LLDA models. Another trend can be found in the Path Traversal vulnerability, which is related to terms xml and xxe, implying that the validation issues associated with this vulnerability are usually the result of XML factors. The final trend we discover is seen in the Resources Errors vulnerability, which is related to the terms denial, service, and crash. The relations given by the models suggest that a Resources Errors vulnerability causes a denial of service, and potentially a crash when it is exploited.

To discover trends in affected products, we performed a similar analysis with STEM and the other models using affected products as topics. Figure 4 demonstrates the top ten terms discovered by STEM, LLDA and TOT for three susceptible software products — Linux Kernel, iPhone OS, and Android — in the NVD dataset. This experiment was originally performed on the ten most affected software products of the dataset, but only three are displayed due to space limitations. The terms inferred by STEM and LLDA are incredibly similar as there is an overlap of 94 percent between the two models. The terms discovered by TOT are once again slightly different compared to the terms discovered by the other models. Topic 13, 9 and 10 are chosen for TOT because they have the most resemblance with the three columns of STEM in Figure 4.

We discover equally interesting trends when we use affected products as topic labels. For instance, in looking at the terms generated by the models for the topic Linux Kernel we discover that memory and crash are related and appear high in the list, indicating vulnerabilities that affect Linux Kernel cause memory issues. Another unique relationship can be seen for Android, whose terms include kernel, qualcomm, and driver for both STEM and LLDA. Unlike for Linux Kernel, these terms describe the individual software components that are affected most by the vulnerabilities. Such information is invaluable for software developers and researchers in narrowing down the scope for system vulnerability analysis. The final relationship we discuss pertaining to Figure 4 relates to the topic iPhone OS, which is related to the terms component and corruption indicating that the result of an exploit in the product iPhone OS is component corruption.

Vulnerabilities are not conventional topics. They evolve and adapt over time, so we further utilized our STEM framework to determine the distribution of vulnerabilities with respect to time. Figure 5 shows the topic evolution of ten vulnerability labels since 2010 based on the STEM model.
be generated using LLDA and TOT because, as stated earlier, LLDA does not support time and TOT does not support labels.

An analysis of Figure 5 reveals some interesting characteristics of vulnerabilities in this decade. First, vulnerabilities appear to have less sudden fluctuations in topic probability compared to affected products as seen in Figure 1 of Section I. However, there are exceptions, such as with the vulnerability Race Conditions which has an abrupt growth in probability followed by stagnation. Figure 5 also reveals that most vulnerabilities had a relatively equal topic probability at the beginning of the decade, but in recent years OS Command Injections, Code, and Session Fixation have become the most prominent vulnerabilities under this topic selection. Meanwhile, Cryptographic Issues, Configuration, and Incorrect Conversion and Cast have had dramatic reductions in probability, implying that software products are becoming less susceptible to these issues.

B. Vulnerability Evolution

In this experiment, we utilize our diffusion-based storytelling framework on individual documents of the NVD dataset. We use the knowledge gained by using STEM on the NVD dataset to choose seed documents for a case study on vulnerability evolution. In Section I, we introduce Figure 1, which chronicles the susceptibility of software products in the 21st century. The figure shows, among other correlations, that there has been a substantial increase in susceptibility for Windows 10 over the past few years. In Figures 2, 6, and 7, we analyze three vulnerabilities using diffusion-based storytelling to establish an evolutionary narrative of vulnerabilities that impact Windows 10. These narratives allow us to determine the factors that contribute to the susceptibility of this product. Similar studies could be implemented for all affected products in the NVD dataset, but for simplicity we chose to only run this experiment on one product. For readability, the document descriptions provided in each figure have been reduced to one sentence.

We select documents over a five-year time segment since the publication of the seed document, and evaluate the quality of a chain using the concept of dispersion coefficient introduced by Hossain et al. in [6]. Dispersion coefficient among a chain of documents containing \( n \) reports is described as:

\[
\psi = 1 - \frac{1}{n - 2} \sum_{i=0}^{n-3} \sum_{j=i+2}^{n-1} \text{disp}(d_i, d_j) \quad (11)
\]

where

\[
\text{disp}(d_i, d_j) = \begin{cases} 
\frac{1}{n+i-j}, & \text{if } \text{soergel}(d_i, d_j) < \theta \\
0, & \text{otherwise}
\end{cases} \quad (12)
\]

The dispersion coefficient of a chain of documents is highest only if consecutive pairs meet a distance threshold, \( \theta \). For our chains, we only choose documents with the highest possible

Figure 6: Evolutionary narrative of CVE-2017-0050, which is related to a kernel permission enforcement vulnerability in many versions of Microsoft Windows. The first four vulnerability reports from left in the chain represent how the last vulnerability report in the chain, CVE-2017-0050, evolved over time.

Figure 7: Evolutionary narrative of CVE-2016-3346, which describes a Windows Permissions Enforcement Elevation of Privilege Vulnerability.
In Figure 6, we display the evolutionary narrative of CVE-2017-0050, a document in the NVD dataset that describes an elevation of privilege vulnerability caused by faulty input validation in Microsoft Windows. The first document in the narrative describes the likely origin of this vulnerability as being a scaling error in Windows kernel-mode drivers that occurred in 2015. The following documents in the narrative describe similar flaws in the Microsoft Windows kernel or its components. The relations between previous CVE’s to the seed CVE, CVE-2017-0050, can be clearly seen. For instance, the document labeled CVE-2016-7216 (4th from left in Figure 6) describes how the kernel API mishandles permissions, which in turn allows for escalation of privilege errors. In the seed document, the kernel does not properly enforce permissions, so although the mishandling of permissions was patched in the kernel, the kernel itself did not enforce these new permissions leading to a denial of service. Note that the documents in the narrative are all related to kernel vulnerabilities, suggesting that vulnerabilities affecting Microsoft Windows components are heavily dependent on previous vulnerabilities that affect the same components.

Figure 7 illustrates the evolutionary narrative of CVE-2016-3346. This particular CVE describes a bug in the loading of a dynamic link library (DLL). The bug leads to elevation of privilege, which causes information disclosure or remote code execution. A possible origin of the vulnerability can be pinpointed to security feature bypass and buffer overflow vulnerabilities in the kernel mode drivers of Microsoft Windows. This implies that the vulnerability documented in CVE-2016-3346 is kernel-based, rather than an inherent DLL issue. Additional documents in the narrative provide further background on the adaptation of the vulnerability, offering a fascinating example that displays the progression of one vulnerability to another. As a whole, the narrative reinforces the concept that vulnerabilities are multifaceted, and they often have dependencies that are not easily visible to experts.

In Figure 2, we present the evolutionary narrative for the document labeled CVE-2015-6218, which describes a high impact vulnerability in Microsoft Windows 10 that causes elevation of privilege and possibly remote code execution because of improper input validation before loading libraries. The first document found by the storytelling algorithm describes a race condition vulnerability in the kernel mode drivers of an earlier version of Microsoft Windows, as Microsoft Windows 10 was released in 2015. The next documents in the narrative describe similar vulnerabilities in the kernel mode drivers of an earlier version of Microsoft Windows. Note that all vulnerabilities in the chain can be exploited via a crafted application. In the final two documents of the narrative, a shift from driver exploitation to library exploitation is seen, but the vulnerability still has the same impact on product security. In summary, this narrative explores the impact of a single vulnerability on the security of multiple products, with a focus on how vulnerabilities in one product contribute to vulnerabilities in others.

Overall, our case study into the susceptibility of Microsoft Windows 10 suggests that vulnerabilities do evolve over time. It also suggests that vulnerability development is based on several factors, such as the dependence of vulnerabilities on previous issues and the cross-product nature of vulnerability adaptation.

C. Shifts in Product Susceptibility

In order to explain shifts in products susceptibilities, we perform an empirical comparison test. Figure 8 (left) shows the topic probability of ten vulnerability labels since 2000 and Figure 8 (middle) shows the product susceptibility of ten products since 2000. Comparing these two figures explains some of the shifts in susceptibility. However, other factors such as product use and product depreciation that have an undeniable impact on susceptibility are not considered here. For instance, Figure 8 (middle) shows that since 2000, certain products, such as Edge and Windows Server, have appeared that completely changed the susceptibility dynamic. However, Safari remains an exception. It becomes susceptible around 2003, when it was released to the public, and it has a high susceptibility until about 2014 when its susceptibility abruptly falls. Similarly, the topic Numeric Errors in Figure 8 (left) grows especially probable in the time range 2003-2014, before abruptly declining as well. Of course, there are multiple reasons for the shift in susceptibility for Safari, but Numeric
Errors vulnerabilities undoubtedly play a part as they are the only vulnerabilities with a similar probability tendency out of all the given topics.

Another shift that can be partially explained using this pragmatic method is the sudden susceptibility of both Microsoft Edge and Windows Server 2012. Although the shifts in susceptibility for these products can be partly credited to the time of their introduction into the product ecosystem, the shift in susceptibility is especially drastic and unlike the natural growth of susceptibility observed thus far. However, an analysis of overall vulnerability distribution offers some explanation for this phenomena. Both the topics NULL Pointer and Data Handling in Figure 8 (left) show similar growth patterns suggesting that a proliferation of these vulnerabilities contributes to an equal if not greater increase in susceptibility for the products Edge and Windows Server 2012. Once again, this assertion stands for these particular vulnerabilities because they are the only ones that have a comparable probability variation during this time period.

Our final experiment using STEM in the realm of product susceptibility is an analysis of shifts that occur when a new software product is introduced to the public. In Figure 8 (right), we assess product susceptibility over ten years since the introduction of iPhone OS. For this experiment, we chose to only study products that were created by Apple, the company that developed iPhone OS, to display the impact of product introduction on a company’s overall product susceptibility. An analysis of this experiment reveals that since the introduction of iPhone OS there has been a steady decline in susceptibility for iTunes, Apple TV, and Safari, but there has been an aggressive growth in susceptibility for iPhone OS. Now, iPhone OS is the most susceptible product created by Apple, but the susceptibility of Mac OS, which remained steady for most of the period, is on the rise as well. This relationship suggests that substantial shifts occur in the product susceptibility domain once a new product enters the market, but it also demonstrates that product susceptibility is volatile even if a product introduction has not occurred, such as in the case of Mac OS.

D. Timestamp Prediction, Coherence, and Perplexity

STEM models the temporal evolution of topics along with word co-occurrence probabilities. To illustrate that the topics generated by STEM can capture time more accurately than other models, we approximated the timestamp of each NVD document from its distribution of topics according to four topic models (STEM, LDA, LLDA, and TOT) as well as a baseline model that predicted timestamps randomly. Support Vector Machine regression was used to predict timestamps for the four topic models, where the feature vectors for regression were the topic distributions in NVD documents. For more information on Support Vector regression refer to [3]. Root Mean Squared Error (RMSE) was used for error measurement and the models were evaluated at various folds (training/test splits). We found that STEM performed as good as TOT, resulting in lower error (RMSE). STEM also had a lower error than LDA and LLDA. The resulting figure is omitted here because of space considerations, but it can be found at the website mentioned in Section V-A.

We use Topic Coherence [12] as a second method to evaluate the quality of the topics generated by STEM. Each generated topic consists of words, and topic coherence is applied to the top \( N \) words from each topic. Higher topic coherence is better. Figure 9 shows the coherence of topics generated by STEM, LLDA, TOT, and LDA for different numbers of topics in the NVD dataset. The figure shows that STEM outperforms all models in terms of topic coherence.

Perplexity is another widely used metric of convergence in topic modeling. It is measured as the likelihood of the inverse of the geometric mean per word. A lower perplexity is an indicator of a better fit to the data. Figure 10 shows the perplexity for STEM, STEM-P, LLDA and TOT on the NVD dataset.
NVD dataset. STEM and the parallel version of STEM (called STEM-P) converge in less iterations than TOT and compete well with LLDA by being very close to it.

**E. Parallel Inference**

In this paper, we draw conclusions on the nature of vulnerabilities that affect software products based on only one cyber-security corpus. In order to discover more intricate relationships in this domain, other corpora may be necessary. Considering this, the scalability of STEM becomes of crucial importance as the model needs to be trained for each new dataset, a process that can be computationally intensive and time-consuming. In this subsection, we perform an experiment on the scalability of STEM, focusing on how the model converges during the inference process. In Figure 11, we compare the convergence times of the STEM with its parallel version, STEM-P, over ten iterations for datasets with 25, 50, 75, and 100 topics. Our analysis reveals that STEM-P is able to converge 30 to 40 percent faster than the regular STEM model, suggesting that the STEM model has high scalability when implemented using a parallel approach.

Figure 11: Convergence time of STEM and STEM-P for different number of topics.

**VI. Conclusion**

This paper proposes an analytic framework that uses a Supervised Topical Evolution Model (STEM) and a diffusion-based storytelling algorithm to discover and study hidden themes and trends in the National Vulnerability Database, a cyber-security corpus describing software vulnerabilities since 1996. A variety of experimental results indicate that our overall framework reveals notable relationships between vulnerabilities, vulnerability interactions, and the susceptibility of software products. We reinforce our results by comparing different models. In the future, we will use STEM in conjunction with diffusion-based storytelling to predict vulnerabilities in software before they occur. We will also use STEM to estimate the future appearance probabilities of vulnerabilities in the NVD.

**VII. Acknowledgments**

This material is based upon work supported by the National Science Foundation under Grant No. HRD-1242122.

**REFERENCES**


