Security Trend Analysis with CVE Topic Models

Stephan Neuhaus
Università degli Studi di Trento
Trento, Italy
Stephan.Neuhaus@disi.unitn.it

Thomas Zimmermann
Microsoft Research
Redmond, WA, USA
tzimmer@microsoft.com

***Abstract*—We study the vulnerability reports in the Common Vulnerability and Exposures (CVE) database by using topic models on their description texts to find prevalent vulnerability types and new trends semi-automatically. In our study of the 39,393 unique CVEs until the end of 2009, we identify the following trends, given here in the form of a weather forecast:**
PHP: declining, with occasional SQL injection.
Buffer Overflows: flattening out after decline.
Format Strings: in steep decline.
SQL Injection and XSS: remaining strong, and rising.
Cross-Site Request Forgery: a sleeping giant perhaps, stirring.
Application Servers: rising steeply.

***Keywords*-security; trends; machine learning**

I. Introduction

Security is often viewed as an arms race between crackers, who try to exploit flaws in deployed systems, and security people, who wish to make that impossible. It is therefore desirable to know of emerging trends in security in order to be able to think about countermeasures before these emerging trends become large-scale problems.

One large and publicly available source of vulnerability reports is the Common Vulnerabilities and Exposures database (CVE) hosted at MITRE[1]. According to the CVE's FAQ[2], "CVE is a list of information security vulnerabilities and exposures that aims to provide common names for publicly known problems. The goal of CVE is to make it easier to share data across separate vulnerability capabilities (tools, repositories, and services) with this 'common enumeration.'" This seems largely to be true: vulnerability databases often contain CVE identifiers whenever they reference a vulnerability that is also in the CVE.

The only large trend study on publicly available data has been done on the CVE in 2007 [1]. This study relied on a manual classification of entries, helped by a classification system called Common Weakness Enumeration (CWE) [3]. The CWE aims to be a complete dictionary for software weaknesses.

One problem with the CVE is that the CWE classification system is far too detailed: there are simply too many CWEs to choose from. Therefore, a small set of 19 CWEs was chosen for the user interface to avoid overwhelming the person making the CVE entry. Steven Christey from MITRE told us in an email of August 2009: "[MITRE] worked with the NVD team at NIST to come up with a set of identifiers that was fairly small but still gave broad coverage for most vulnerabilities. (You don't want to have to train people to map 700+ CWE names, nor do you want to write the user interface for building up a menu of that size ;-)".

This will lead to coarse-grained classification and consequently to loss of information. Also, keeping the list fixed will cause emerging trends to be buried in related but irrelevant classifiers. The same problem arises when using supervised learning techniques, since the topics are fixed in advance and are not allowed to emerge by themselves. In addition, even though the classification system is already coarse-grained, many CVE entries do not have any CWE classification at all. That holds especially for the earlier entries. Therefore, to analyze trends for CVE data, there is a lot of manual work involved; work that will include the exercise of discretion and good taste.

In this paper, we follow a different approach, not based on CWEs or any other fixed classification system. Instead we use Latent Dirichlet Allocation (LDA), an *unsupervised learning technique*, on the description texts of CVE entries in order to come up with our own classification system, called a *topic model*. This allows us to identify prevalent topics, but also *emerging trends*, in an automated fashion.

Our contributions to the analysis of corpora of vulnerability data in general and to the CVE in particular are:

- We have conducted the *first independent study* on the whole body of the CVE database outside of MITRE.
- Since we used a completely different methodology than the original study, we offer an independent way to *validate* the results periodically published by MITRE. While most of our findings agree with theirs, the independence in methodology gives a greater confidence in the results.
- We offer a mostly *automated approach* to analysing large corpora of vulnerability texts such as the CVE. Topic models will automatically find the prevalent topics in the CVE and researchers are not limited to manual and potentially error-prone labeling, for example by CWEs or supervised learning techniques. Overall, the

[1]http://cve.mitre.org/
[2]http://cve.mitre.org/about/faqs.html
[3]http://cwe.mitre.org/

topics identified by our automated analysis are in good agreement with the manual CWE classification in the CVE database.
- Related to the previous point, this work can be the basis of a *recommendation system* that suggests possible topics to people entering new CVEs into the CVE database. This could solve the problem of the relative coarse-grainedness of CVE classifications.
- We show how topic analysis helps to identify emerging trends. We found one trend (vulnerabilities involving application servers) for which we offer evidence of growing importance, but which has gone unnoticed in the original MITRE study.

The remainder of this paper is organised as follows: First, we describe our methodology (Section II). Then we present our results (Section III) and describe potential threats to validity of this study (Section IV). We finish the paper with a discussion of related work (Section V), as well as conclusions and future work (Section VI). We also provide a step-by-step guide on how to replicate our study (Appendix A).

This paper contains some large tables, which we present at the end of the paper rather than in-line.

II. Methodology

A. Data Gathering and Corpus Preparation

An annotated version of the CVE database is offered by NIST, and there known as the National Vulnerability Database (NVD)[4]. MITRE creates the CVE descriptions and adds the relevant references. The NVD team at NIST then receives this basic information from MITRE (and through the public CVE site). NIST then adds other information such as structured product names and versions, and also maps the entries to CWE names.

NIST offers files named *nvdcve-2.0*-year*.xml* for download, where *year* is a number from 2002 to 2010 inclusive. The file for 2002 contains entries from 1988 through 2002, and the file for 2010 is necessarily incomplete. Since there is usually a period of consolidation when a new CVE entry (called a *candidate*) might be changed or rejected, we only looked at entries that were published up to 2009, inclusive.

I order to build timelines of topics, we need to know when the problem described in a CVE was discovered (first identified as a problem), disclosed (made available to the developers or the public), or published (entered into the CVE corpus). This is a notoriously difficult problem, since discovery or disclosure dates are by their very nature hard to come by. For our purposes, we need to date a CVE entry only to within a year, but while each CVE entry contains a date field called `published-datetime`, this field cannot be used to date a CVE reliably, according to information from MITRE. Therefore, we use the disclosure date of the corresponding OSVDB[5] entry. The OSVDB is an open vulnerability database that is comparable to CVE in scope.

Table I
List of stop words

&apos	and	does	into	or	then	whether
&quot	are	e	is	p	there	which
+	as	ff	it	s	these	while
++	at	for	k	some	they	who
?	attack	from	kei	ss	thi	whose
??	be	g	later	such	to	with
a	been	get	mc	t	up	within
about	by	go	no	than	us	without
all	can	ha	not	that	v	x
allow	dn	has	o	the	via	
also	do	if	of	their	wa	
an	doe	in	on	them	when	

aix 1	execut 1	ibm 1
arbitrari 1	file_comp 1	**overflow** 1
buffer 1	for 1	rcp 1
code 1	function 1	remot 1

Figure 1. Stemmed word list for CVE 2002-1621

From each CVE entry, we extracted the CVE ID (a unique identifier) and the summary text. We then subjected the words in the summary text to the Porter stemming algorithm [2], which attempts to find word stems so that for example the words 'programming', 'programmed', and 'programs' would all be mapped to their stem 'program'. Next, we removed a number of stop words, i.e., words that are so common that they do not help differentiate between CVEs; see Table I.[6] Finally we counted the occurrence of each stem in the summary text. After these operations, we therefore had, for each CVE, a list of word stems and the number of times they occurred in that CVE. For example, Figure 1 shows the list for CVE 2002-1621, clearly a buffer overflow.

B. Topic Models

We subjected these stemmed CVEs to Latent Dirichlet Allocation (LDA), an unsupervised learning technique [3].

In the LDA model, a CVE is generated by first picking a distribution over topics. Given this distribution, the topic of each specific word is picked, and finally, words are generated given their topics.

For example, the LDA model of the CVE data has topics for "buffer overflow" and "directory traversal". The topic for buffer overflow will contain words like 'buffer', 'overflow', 'code' and 'execut' with high probability, whereas the topic for directory traversal will have words like 'directori', 'dot', and 'travers'. Words like 'vulner' will appear in both topics.

[4]http://nvd,nist.gov

[5]http://www.osvdb.org

[6]Seemingly nonsensical stop words like 'a', 'e', 'g', 'k', or 'mc' come about by breaking words like "e.g.", "a.k.a.", or "MC/ServiceGuard" at word separators like the period '.' or the slash '/'.

In LDA, documents are considered to be bags of words, and words are considered to be independent given the topics (i.e., word order is irrelevant). When LDA is run, the various distributions are learned using Bayesian inference. In effect, the generative model just described is run backwards and the distributions are updated from their priors using the actual documents.

We let LDA identify 40 topics, starting from random topic assignments.[7] If there are m unique stemmed words in the entire CVE corpus, and if there are n CVEs, the result of LDA contains, among other things, a file that assigns the word with index k ($1 \le k \le m$) in document c ($1 \le c \le n$) a topic z ($1 \le z \le 40$), meaning that word k in document c is about topic z. Note that the same word can be assigned to different topics in different documents. Another result of the LDA software is a list of the most frequent words in a topic, together with their frequencies.

First, we define the probability that a given document c is about a given topic z:

$$\hat{p}(z \mid c) = \sum_{\substack{w \in c \\ w \text{ is about } z}} \frac{\#\text{occurrences of } w \text{ in } c}{\#\text{words in } c}. \tag{1}$$

This formula can lead to fractional (or fuzzy) assignments. For example, CVE 1999-1471 is about the passwd program; the description reads “Buffer overflow in passwd in BSD based operating systems 4.3 and earlier allows local users to gain root privileges by specifying a long shell or GECOS field.” After stemming and removing stop words, 18 words remain, 12 of which (‘bsd’, ‘earlier’, ‘gain’, ‘gecos’, ‘local’, ‘oper’, ‘passwd’, ‘privileg’, ‘specifi’, ‘system’, ‘user’, and ‘root’) are assigned to topic 40 (manually labeled “Privilege Escalation”), four (‘buffer’, ‘long’, ‘overflow’, and ‘shell’) to topic 12 (“Buffer Overflow”), and one each (‘base’ and ‘field’) to topics 34 (“Linux”) and 30 (“Message Boards”), respectively. The assignment (about 67% privilege escalation, 22% buffer overflow, 11% wrong assignments) is quite accurate.

We also give an indication of the *overall importance* of a topic, which is given by

$$\hat{p}(z) = \frac{1}{\#\text{CVEs}} \sum_{c} \hat{p}(z \mid c). \tag{2}$$

LDA cannot leave a topic unassigned; therefore, when LDA is run on a corpus as large as the CVE with a comparatively large number of topics, it can happen that some of the topics are really the same and should be combined. This topic equivalence can be detected through a large overlap in the set of words that appear in different topics. For example, the most common words in topic 8 are ‘overflow’, ‘buffer’, ‘code’, ‘execut’, and ‘arbitrari’, whereas the most common words in topic 12 are ‘command’, ‘execut’, ‘arbitrari’, ‘buffer’, and ‘overflow’. When k topics $z_1, \ldots, z_k$ are combined into a topic z, a that topic will have the joint probability

$$\hat{p}(z) = \sum_{1 \le j \le k} \hat{p}(z_j), \tag{3}$$

where $\hat{p}(z_j)$ is given by Equation (2).

One result of the LDA software is a list of the most frequent words in a topic, together with their probabilities. Again, when k topics are combined, a word w will have the joint probability

$$\hat{p}(w \mid z) = \sum_{1 \le j \le k} \frac{\hat{p}(w \mid z_j)}{\hat{p}(z_j)} \hat{p}(z), \tag{4}$$

where $\hat{p}(z)$ is given by Equation (3).

C. Trend Analysis

LDA does not measure changes over time. One alternative is to use Dynamic Topic Models [5], which would change the topic distributions and the word distributions within a topic from one year to the next. Topics over Time [6] works the other way around, assuming that a CVE has a time stamp depending on a topic-specific beta distribution.

Both approaches have restrictions, however: dynamic topic models penalise large changes in word distributions for any particular topic from one year to the next, whereas topics-over-time distributions are rather inflexible.

Therefore, we follow Hall et al. [7] and study only the post hoc empirical probabilities that a randomly selected CVE in a given year y is about topic z.

$$\begin{aligned} \hat{p}(z \mid y) &= \sum_{c:t(c)=y} \hat{p}(z \mid c)\, \hat{p}(c \mid y) \\ &= \sum_{c:t(c)=y} \hat{p}(z \mid c) \frac{1}{\#\text{CVEs in year } y} \\ &= \frac{1}{\#\text{CVEs in year } y} \sum_{c:t(c)=y} \hat{p}(z \mid c), \end{aligned} \tag{5}$$

where $\hat{p}(z \mid c)$ is given by Equation (1) and $t(c)$ is the year in which CVE c was published. This formula is simple to compute, since $\hat{p}(z \mid c)$ is a result of the LDA model estimation process.

Plotting $\hat{p}(z \mid y)$ against y will then give an indication about the *relative importance* of topic z in year y: if $\hat{p}(z_1 \mid y)$ is twice as high a $\hat{p}(z_2 \mid y)$, then twice as many CVEs were about z_1 than about z_2 in year y.

In addition to a graphical display, it is also interesting to know how much a topic has changed in importance, both since measurements began and in the last year. In general, we compute the change from year y to year $y + n$ as the *average annual change*:

[7] Automatically finding the best number of topics when doing unsupervised learning is at this point an unsolved problem. It has been solved for supervised learning [4], and there are some ideas for unsupervised learning, but at the time of writing, there is no solid theory. The number 40 seemed to be a good value with good topic separation.

	Actually is x	Is not x
Classified as x	true positive (*TP*)	false positive (*FP*)
Classified as not x	false negative (*FN*)	true negative (*TN*)

$$\textit{precision} = \frac{\#\textit{TP}}{\#\textit{TP} + \#\textit{FP}} \quad \textit{recall} = \frac{\#\textit{TP}}{\#\textit{TP} + \#\textit{FN}}$$

Figure 2. Assessing the quality of a classifier.

$$\textit{change} = \Big(\frac{\hat{p}(z \mid y+n)}{\hat{p}(z \mid y)}\Big)^{1/n} - 1, \tag{6}$$

The intuition behind this is that this number will give the relative change that the initial level of importance will have to undergo every year to arrive at the final level of importance. For example, if this number is 0.1, then that means that the importance of the topic has risen by 10% per year on average.

D. Causes and Impacts

During exploratory analysis of the CVE corpus we noticed a curious feature, namely that most (72.9%) were of the form "x allows remote attackers to y" or similar forms. For example, CVE 2008-0895 reads, "BEA WebLogic Portal 10.0 and 9.2 through MP1, when an administrator deletes a single instance of a content portlet, removes entitlement policies for other content portlets, which *allows attackers to* bypass intended access restrictions." This separates the description of the CVE into two parts, the first of which describes the *cause* of the problem and the second the *impact*. For those CVEs for which we found such a division, we subjected them to the same LDA process in the hope of finding out trends in both causes and impacts.

E. Alignment with CWEs

One problem is that topic models, being an unsupervised learning technique, do not give indications of goodness-of-fit, so we usually have no quantitative indication on how well the classification works. In this case, however, many CVEs do have an independent classification in the form of CWE assignments, so we can compare the CWE value with the topic that was assigned by running LDA. We will discuss the question of what it means for LDA and CWE topics to be in agreement below, in Section III-E.

Usually, the quality of a classifier is determined by using the false-positive and false-negative rates, or equivalently the related measures of *precision* and *recall*. High precision values mean that classifications are often correct, and high recall values mean that many documents will be correctly classified; see Figure 2. We cannot use precision and recall directly, however, since our topic assignment based on LDA is *probabilistic*. Remember that a document can be about multiple topics, which is captured by the probability $\hat{p}(z \mid c)$. We therefore need to transform our probabilistic topic assignments into crisp ones, where each document is assigned exactly one topic. We do this for a given document c by taking the topic z for which $\hat{p}(z \mid c)$ is largest.

We compute precision and recall values separately for each topic. This helps us identify topics that are aligned particularly well with CWEs. Let $cwe(c)$ be the CWE value for a document c and let $\hat{z}(v)$ be a mapping that maps a CWE v onto a topic. (For simplicity, we assume that $\hat{z}$ is single-valued.) Then we count for each document c with topic z:

- a *true positive for* z if $z = \hat{z}\big(cwe(c)\big)$;
- a *false positive for* z **and** a *false negative for* $\hat{z}\big(cwe(c)\big)$ if $z \neq \hat{z}\big(cwe(c)\big)$.

The intuition is that if the CWE topic and the LDA topic are the same for a document, i.e., $z = \hat{z}\big(cwe(c)\big)$, then that should count as a true positive. If they are not, however, it should be counted as a false positive for the LDA topic z and as a false negative for the true CWE topic $\hat{z}\big(cwe(c)\big)$. We do not count true negatives because they are not needed for the computation of precision and recall (see Figure 2).

III. Results

A. Overview

The earliest CVE has an OSVDB disclosure date of August 1, 1982 (CVE 1999-0531, a candidate that is now rejected). By contrast, the CVE *with the earliest* `published-datetime` *field in the CVE database* is CVE 1999-0095, reporting the sendmail DEBUG hole that was exploited by the Morris worm [8]). The latest are from December 31, 2009, comprising a total of 39,749 entries. After removing duplicates, 39,393 unique CVEs remained.[8]

After removing stop words and stemming, the CVE summaries could be as short as a single word (for example for CVE 1999-0657, for which the entire summary reads "WinGate is being used", which after removing stop words and stemming reduces simply to "wingat"), but also as long as 99 words (for CVE 2007-0018, which describes fairly completely a stack-based buffer overflow in an ActiveX control). The distribution is shown in Figure 3 (left). It is clearly a unimodal distribution, having a median of 18 words, a mean of 18.85 words, and a standard deviation of 6.49 words. But it is not a symmetric distribution (skewness 1.51, kurtosis 5.91)[9].

The right part of Figure 3 shows how the CVE database grew over the years. Apparently, the number of CVE entries peaked in 2006 with 6,885 submissions in that year, and has been slightly declining in 2007 (to 6,393) and more drastically in 2009 (by a massive third to 4,446). We assume that the people creating CVE entries have a large backlog, and that over time, more CVE entries that are published in

[8]We have checked 53 duplicate CVE IDs manually to confirm that they were indeed duplicate entries having identical fields and not different entries with the same CVE ID.

[9]Skewness and kurtosis have been computed with methods compatible with SAS and SPSS.

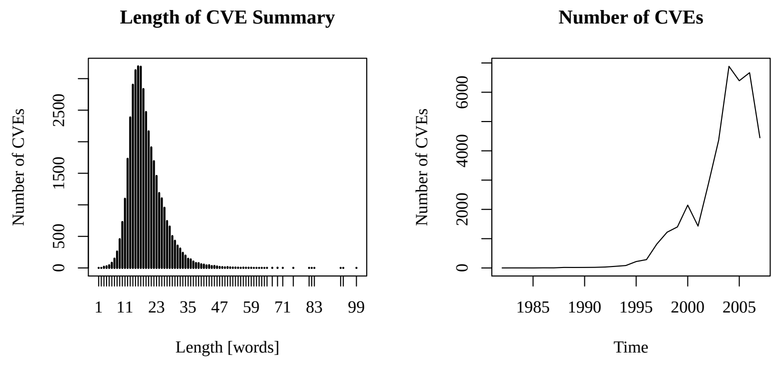

Figure 3. Length of CVE summary text after stemming and stop word removal (left) and number of CVEs issued per year between 1988 and 2009 (right).

2010 will turn out to have been disclosed in 2009, so that this number will probably rise.

B. Topic Models on the Entire Corpus

Combining topics, we arrived at 28 unique topics. Topics for which we could not find a good name are simply labeled "Topic x", where x is the topic number produced by LDA.

The first main result is a table with all 28 unique topics (Table IV). The first column contains the topic name z. The second column contains a sparkline, showing the trend of that topic. Sparklines are wordlike summaries of data, invented by Edward Tufte [9]. This particular version of sparklines shows a diamond at the beginning and end of the sparkline, and also a diamond at the maximum value. The remaining columns contain relative frequency $\hat{p}(w \mid z)$ of that topic in the years from 2000 to 2009, according to Equation (4).

The table contains all the expected topics such as buffer overflows, format string vulnerabilities, SQL injection, cross-site scripting, cross-site request forgery and so on, and roughly in the order that we would intuitively expect them to be from following the security press.

However, there are also a few surprises:

- There are topics just for Linux kernel issues, Microsoft Office and Microsoft Windows. The reason is that with partial/fuzzy assignments, LDA will not try to find mutually exclusive topics, since a document can be partly about one topic and partly about another.
- The proportion of resource management issues is very high. This is due to the combination of denial-of-service topics with more traditional resource management topics such as memory leaks. This was done because manual inspection of respective CVEs showed that these topics were not really separable: a CVE that had words assigned to resource management issues usually also had words assigned to a denial-of-service topic.
- CVE Management issues (disputed entries and rejects) make up almost 3% of the CVE corpus. We will return to this topic when discussing the changes in topic importance below.
- The word "PHP" appears in four of the 28 topics, SQL injection and cross-site scripting, implicating PHP in these vulnerabilities. However, the proportion of PHP-specific vulnerabilities has been declining since 2007, especially for arbitrary code executions due to PHP. One reason might be that support for PHP 4 was finally discontinued in August 2007, forcing web site operators to upgrade to the more secure PHP 5. See also Section III-C below.
- For the topic labeled "Cross-Site Request Forgery", the strings "cross-sit" and "forgeri" appear only beyond the 10 most frequent words (stemming reduces "cross-site" to "cross-sit" and "forgery" to "forgeri").[10]

C. Trend Analysis

The second main result is a graphical display of $\hat{p}(z|y)$ for the 28 topics since 2000 (the sparklines in Table IV). The main findings are:

- Well-known vulnerabilities like buffer overflows and format strings are declining, though perhaps not as steeply as one would wish. It seems that buffer overflows are harder to exploit now than they were in the past, and therefore that effective protective measures have finally found their way into operating systems, compilers, and libraries.
- Resource management issues are also generally declining. This puts the high percentage of this topic in Table IV into perspective: the situation was bad, but it is getting better.
- Privilege escalation and link resolution issues are also going down, perhaps indicating better compartmentalisation and secure-by-default configurations.
- Perhaps surprisingly, the importance of exploits allowing arbitrary code execution is also going down slightly in recent years (but see the section on causes and impacts below).

[10]Due to space constraints, we cannot show the most frequent words in this paper. Interested readers are invited to peruse the Replication Guide in Appendix A to get a copy of the data and scripts or to contact the authors.

- SQL injection and cross-site scripting have dents in their growth curves of the last few years.

The third main result is Table II, which shows more quantitatively the change in importance for the 28 unique topics identified by LDA from 2000 to 2009 and from 2008 to 2009, according to Equation (6).

The table is sorted in decreasing order of combined change[11]; the "since 2000" column can be used to see whether the change from 2008 to 2009 was in accordance with or against the general trend since 2000. For example, application server-related vulnerabilities have risen by 20.6% annually since 2000, and by 83.2% from 2008 to 2009.

The comparatively unimportant Topic 35 appears relatively high up the list because it rose from a mere 0.33% in 2008 to 1.5% in 2009. We believe this to be a random fluctuation, not a new trend.

Above we mentioned that incomplete or disputed CVE entries make up about 3% of CVE entries. Looking now at the changes in importance, we can see that this quality has been eroding since 2000, but apparently some quality measures have taken effect. In fact, when one looks at the plot of that topic in Table IV, one can see that after a sharp, almost exponential increase until 2006, the topic has been falling.

D. Causes and Impacts

The CVE corpus contains 28,699 unique entries of the form "x, allows attackers to y" or similar forms, 72.9% of all unique entries.

Tables V and VI show causes impacts respectively, analogous to Table IV. What can be seen from the table is that the four most frequent causes are responsible for two thirds of all CVEs; after the top four, topics fall off sharply in importance. PHP is identified as one of the four major causes, making it a prime candidate for improvement. For the impact data, the situation is even more clear-cut: arbitrary script and code executions make up almost 58% of all impacts. This suggests that research on the prevention of such vulnerabilities could have major impact (no pun intended). While it is true that adding denial of service and information leaks would make the figure rise to 92%, preventing such attacks is in our opinion more difficult than preventing arbitrary code execution.

When looking at the most frequent words in the "cause" topics, one disappointing element is the appearance of the word "unspecifie[d]" in various topics. Looking at the corresponding CVEs, one finds that the phrase "unspecified vulnerabilities" or "unspecified vectors" is responsible for this. For example, CVE 2007-0114 reads, "Sun Java System Content Delivery Server 5.0 and 5.0 PU1 allows remote attackers to obtain sensitive information regarding 'content details' via unspecified vectors." Entries like these simply mean that the source cannot or will not disclose the actual cause of the vulnerability. That will in turn mean that the CVE entry will be incomplete, rendering it less useful.

There is an apparent discrepancy in the cause/impact data versus the entire data as seen in Table IV. For example, in Table V, we see that 19% of all CVEs are about cross-site scripting, whereas the corresponding column sum in Table IV gives only about 9.7%. The reason for this effect is that sometimes a given cause may allow different impacts. For example, a buffer overflow may at one time allow the execution of arbitrary code, at another it may lead to a crash and therefore to a denial of service. So LDA might assign a document to either the cause (buffer overflow) or to the impact topic (arbitrary code or denial of service), or partly to both, depending on which part dominates in the description. As soon as the document is split into cause and impact, however, assignments can be made more clearly.

Tables V and VI show graphs of the unique cause and impact topics analogous to Table IV. For both data sets, the important information is that the graphs follow the same general curves as the corresponding ones in Figure IV. For example, cross-site scripting and SQL injection are rising, whereas buffer overflows, denial of service and privilege escalation are falling and PHP seems to have had its peak in 2006 or 2007.

E. Alignment with CWEs

Next, we looked at how well the 28 topics found by LDA aligned with the 19 pre-assigned CWEs that are available to someone who enters a new CWE. To do that, we first needed to map LDA topics to CWEs. We ended up with the assignment shown in Table III, but not all topics were assigned a CWE or vice versa. The reason for this is that an LDA topic might not coincide naturally with an available CWE or only with an CWE that is not used in the NVD. For example, Topic 23 was labeled "Privilege Escalation", and CWE 269 (improper privilege management) exists for just such a case. But this CWE is not one of the 19 CWEs offered for CVE entries. This is not a problem for LDA, since partial assignments are possible ("this document is 50% about topic 1 and 50% about topic 2"), but the CVE does not allow such partial assignments. That said, we were able to map 12 of the 24 LDA topics directly to CWEs.[12]

Table III shows the precision and recall values according to Figure 2. These show that the performance of LDA is very good when it comes to standard categories like cross-site scripting, directory traversals, link resolution or SQL injection (precision of 80% or more and recall of

[11]We used a weighted geometric average, weighing the change since 2000 ten times as high as the change since 2008.

[12]Of the 19 CWEs used in the NVD, we could not map seven CWEs to LDA topics; they are: CWE 16 (configuration/insecure defaults), CWE 20 (improper input validation), CWE 78 (OS command injection), CWE 189 (numeric errors), CWE 255 (credentials management), CWE 310 (cryptographic issues), and CWE 362 (race conditions).

Table II
TOPICS SORTED ACCORDING TO THEIR CHANGE IN IMPORTANCE SINCE 2008. THE COLUMN MARKED "SINCE 2000" CONTAINS THE AVERAGE CHANGE PER YEAR SINCE 2000 IN PERCENT, AND THE COLUMN MARKED "SINCE 2008' CONTAINS THE CHANGE FROM 2008 TO 2009 IN PERCENT. LONG-TERM CHANGES WEIGH MORE THAN SHORT-TERM CHANGES.

Name	since 2000	since 2008	Name	since 2000	since 2008
Cross-Site Scripting	+46.6	+3.8	Firewalls	−0.7	+44.1
SQL Injection	+45.7	−39.6	Resource Management	−1.4	+36.1
Arbitrary Code (PHP)	+37.7	−23.0	Linux	−2.6	+50.0
PHP	+29.9	−8.7	Arbitrary Code	+2.6	−16.1
Application Servers	+20.6	+83.2	Directory Traversal	+1.9	−19.6
Topic 35	+15.6	−8.7	Format String	+1.9	−23.1
Microsoft Office	+7.9	+37.6	Buffer Overflow	−4.3	+37.7
Mozilla	+5.3	+68.8	Message Boards	+0.4	−22.7
Information Leak	+10.6	−5.7	Topic 17	−10.4	+45.2
Microsoft Windows	−2.6	+139.0	Credentials Management	−7.4	−12.8
Topic 7	+1.0	+65.6	Arbitrary Code (IE)	−11.2	+5.1
Java	+7.0	−13.1	Cryptography	−10.6	−2.4
CVE Issues	+5.7	−9.5	Privilege Escalation	−17.8	−8.5
Cross-Site Request Forgery	+5.5	−17.6	Link Resolution	−14.5	−51.3

Table III
PRECISION AND RECALL FOR MAPPABLE CVES. THE FIRST COLUMNS ARE FOR PRECISION ('P') AND RECALL ('R').

P[%]	R[%]	LDA Topic Name	CWE	CWE Name
97.8	94.6	SQL Injection	89	Improper Sanitization of Special Elements used in an SQL Command ('SQL Injection')
98.1	85.4	Cross-Site Scripting	79	Failure to Preserve Web Page Structure ('Cross-site Scripting')
93.1	85.6	Directory Traversal	22	Path Traversal
57.6	80.1	Link Resolution	59	Improper Link Resolution Before File Access ('Link Following')
51.8	75.3	Format String	134	Uncontrolled Format String
60.1	57.6	Buffer Overflow	119	Failure to Constrain Operations within the Bounds of a Memory Buffer
29.7	49.3	Resource Management	399	Resource Management Errors
24.9	54.5	Cross-Site Request Forgery	352	Cross-Site Request Forgery (CSRF)
33.1	18.6	Information Leak	200	Information Leak (Information Disclosure)
28.0	18.0	Cryptography	310	Cryptographic Issues
12.1	38.7	Credentials Management	255	Credentials Management
14.2	8.7	Arbitrary Code	94	Failure to Control Generation of Code ('Code Injection')

80% or more). Other categories fare much worse, among them the buffer overflow. This is surprising at first glance, since buffer overflow reports ought to have the two words "buffer" and "overflow" occurring somewhere. However, this need not be the case; for example, CVE 2008-0090 reads, "A certain ActiveX control in npUpload.dll in DivX Player 6.6.0 allows remote attackers to cause a denial of service (Internet Explorer 7 crash) via a long argument to the SetPassword method.", so possible CWEs would include 20 (input validation), 399 (resource management), or 255 (credentials management), yet the CVE was assigned CWE 119 (buffer overflow), something that is not apparent from the description.

False positives and false negatives in the assignment of LDA topics to CWEs can exist for multiple reasons.

1) In practice a CVE can be about multiple CWEs; while LDA accounts for this with its probabilistic assignment, the actual CWE assignment of in the NVD database does not.
2) Ambiguous entries in the CVE database. For example if a buffer overflow allows the injection of arbitrary code, should the CVE entry be classified as CWE 119 (buffer overflow) or CWE 94 (code injection)?
3) The assignment of a CWE to a CVE is to some degree arbitrary, too, as the example for CVE 2008-0090 and buffer overflow shows.

Overall, we believe that our approach shows that the current CWE assignment system is too strict and too opaque. It should be possible to tag CVEs using a more flexible system, one that allows to assign more than one tag.

IV. THREATS TO VALIDITY

We let LDA seed topics randomly. On the one hand, this is precisely the point of using an unsupervised learning technique. On the other hand, random topic seeding could yield different topics on different runs. While not having conducted a systematic investigation, we ran LDA a number of times with different random seeds and, apart from different topic numbers, got the same topics that we report in this study.

The CVE database is uneven when it comes to the quality of the vulnerability descriptions, ranging from meaningless entries such as "WinGate is being used" (CVE 1999-0657;

see above) to complete analyses of the problem, including the root cause. This could lead LDA to skew its analysis towards those CVEs that are better reported.

Different vendors may have different terminologies when describing vulnerabilities. This may lead to essentially identical vulnerabilities being assigned different topics. This threat is to some extent mitigated by manually assigning labels to topics and combining topics.

Different vendors may also have different disclosure strategies, which could skew the trends we publish. There is evidence for this because December 31 has an unusually high number of CVEs: 7.7% of all CVEs (3051) were published on that day (according to their OSVDB disclosure dates), where publishing CVEs more regularly would give only 1/365, or 0.28%,. (Other end-of-month dates do not have such a disproportionately high publishing frequency). This threat is mitigated by aggregating CVEs by year, while at the same time introducing the threat that trends within a year or trends spanning adjacent years will be obscured.

It is always dangerous to make predictions, especially about the future.[13] The mere fact that we are seeing trends does not mean that we can extrapolate them into the future. However, through our knowledge of software security, we are confident that the trends we see are real, and that they will indeed continue in the ways we have described.

V. Related Work

The study that is closest to this work is of course the paper by Christey et al., also on CVE trends [1]. The main differences to our study are:

1) They analysed the data up to and including 2007, whereas we analysed the data up to 2009.
2) They used manual classification of CVE entries, also using information that is not public, even though it is not clear from the report text *how* they categorised the CVEs and exactly what non-public information was being used. The categories appear associated with CWE numbers, but some of the CVE numbers used in the report are not in the 19 CWE numbers that are provided for CVE entry, such as CWE 415 (double-free vulnerability). A request for clarification by email seems to indicate largely manual classification based on indicator words, phrases in the text (like manual topic models), or non-public information.
3) We did not differentiate between open vs. closed source or between different operating systems.

Mainly, our findings agree with this report: buffer overflows are still high on the list, but in decline when compared to web application attacks like cross-site scripting or SQL injection; a low number of causes are responsible for the majority of CVE entries; and decline in link following and directory traversal; regular appearance of information leaks.

[13]Variously attributed to Niels Bohr, Groucho Marx, and Yogi Berra.

However, we disagree on the interpretation of cross-site request forgery. Christey et al. state that it is a sleeping giant (emphasis on “sleeping”) and use the low prevalence of 0.1% in 2006 as an argument. We measure 1.8% prevalence in 2006, and also the growth rate, while negative in the last year, is by no means negligible. Given the ease with which CSRF might be exploited, we should definitely keep an eye on this sleeping giant, lest he wake up.

We seem also to have uncovered a category of attacks on application servers that seems to be rising faster than any other attack category. This category does not appear in the CWEs and hence is missing from the MITRE study.

Another large-scale study is regularly done by Microsoft; the most recent example is the Microsoft Security Intelligence Report for H1 2009 [10]. The report uses a vast array of data sources, ranging from online sources like Bing and Windows Live Hotmail to programs collecting data such as the Malicious Software Removal Tool or various filters in Internet Explorer. Overall, the report tends to focus more on malware and browser exploits.

Woo et al. characterise Web browser vulnerabilities in order to build a vulnerability discovery model [11].

Both Li et al. [12] and Ozment et al. [13] studied in 2006 how the numbers of defect and security issues evolved over time. While Li et al. reported an increase, Ozment reported a decrease in the rate at which new vulnerabilities are reported; for our data, we observed an increase in vulnerabilities until 2006 and since then a decrease. Li et al. used supervised techniques such as Support Vector Machines and Naive Bayes to classify software defects; for security vulnerabilities they used manual classification. In contrast our work uses LDA, an automated and unsupervised learning technique for vulnerability data.

Coverity used its static analysis tools to scan a large number of open-source programs and concluded that the quality of open source software is increasing [14]. We have not yet tried to separate our CVE data into entries pertaining either to open source or to closed source software.

Eric Rescorla looked at the related problem of vulnerability discovery and, by modeling the discovery process and estimating model parameters, concludes that finding vulnerabilities is often without a clear effect on a its lifetime [15]. Also, he noticed that vulnerability data is very heterogeneous. His conclusion was therefore that vulnerability discovery should be de-emphasized the quality of gathered data be increased.

VI. Conclusions and Future Work

We studied the Common Vulnerability and Exposures from the National Vulnerability Database by using topic models on their description texts. Our results include the following.

- Eliminating cross-site scripting, SQL injection, and buffer overflows, and making PHP more secure will

eliminate the majority of all CVEs.
- Application server vulnerabilities have massive growth rates and are probably the "next big thing" for years to come.
- Cross-site request forgery is indeed a sleeping giant, and it is probably already stirring.
- Related to methodology, the possibility to allow that an entry is partly about one topic and partly about another allows much better classification.

Our contribution is not intended to replace manual methods, but rather to complement them: our findings by and large support the analysis done by MITRE [1]. By providing a methodically completely independent way, we gain mutually supporting evidence that the trends that we are seing are actually real. Also, while the analysis in this paper is only a snapshot in time, we have shown that our method is feasible and leads to good results. Since our method is mostly automated, it can thus be easily repeated. To facilitate replication, we provide an archive file with all the data and scripts (see Appendix A).

Another important point is that this work is not another "Top-n vulnerabilities"-type study such as the CWE/SANS Top 25 Programming Errors[14] ot he OWASP Top 10 Project[15]. The main difference is that Top-n lists are awareness instruments based on consensus, not necessarily on actual data, like our study.

Future work will include replication on other corpora like the SANS Consensus Security Alerts [16], the OSVDB, or Bugtraq[16]. Many of these corpora have fewer but longer documents, so one question that could be investigated is whether the same trends emerge. In addition, we plan to distinguish between open-source and closed-source projects. For open-source projects CVEs often link to specific bug report, which could help to obtain the actual change that fixed the vulnerability. Thus another data source that we plan to tap into are bug databases and version archives.

VII. Acknowledgments

Stephan Neuhaus was supported by the European Commission under contract No. 216917, project EU-FP7-IST-IP-MASTER.

The authors thank the reviewers for their constructive and helpful comments.

References

[1] S. M. Christey and R. A. Martin, "Vulnerability type distributions in CVE," http://cwe.mitre.org/documents/vuln-trends/index.html, May 2007.

[2] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 2003, no. 3, pp. 993–1022, January 2003.

[4] D. Blei and J. McAuliffe, "Supervised topic models," in *Advances in Neural Information Processing Systems 20*, J. Platt, D. Koller, Y. Singer, and S. Roweis, Eds. Cambridge, MA: MIT Press, 2008, pp. 121–128.

[5] D. M. Blei and J. D. Lafferty, "Dynamic topic models," in *ICML '06: Proceedings of the 23rd international conference on Machine learning*. New York, NY, USA: ACM, 2006, pp. 113–120.

[6] X. Wang and A. McCallum, "Topics over time: a non-markov continuous-time model of topical trends," in *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 2006, pp. 424–433.

[7] D. Hall, D. Jurafsky, and C. Manning, "Studying the history of ideas using topic models," in *Proceedings from the EMNLP 2008: Conference on Empirical Methods in Natural Language Processing*, October 2008, pp. 363–371.

[8] E. H. Spafford, "The internet worm program: An analysis," Purdue University, West Lafayette, IN 47907-2004, Purdue Technical Report CSD-TR-823, 1988.

[9] E. Tufte, *Beautiful Evidence*. Graphics Press, 2006.

[10] R. Boscovich *et al.*, "Microsoft security intelligence report volume 7: January through June 2009," Microsoft, Inc., Tech. Rep., 2009.

[11] S.-W. Woo, O. H. Alhazmi, and Y. K. Malaiya, "An analysis of the vulnerability discovery process in web browsers," in *Proceedings of the 10th IASTED International Conference on Software Engineering and Applications*, Nov. 2006, pp. 172–177.

[12] Z. Li, L. Tan, X. Wang, S. Lu, Y. Zhou, and C. Zhai, "Have things changed now?: an empirical study of bug characteristics in modern open source software," in *ASID '06: Proceedings of the Workshop on Architectural and System Support for Improving Software Dependability*. ACM, 2006, pp. 25–33.

[13] A. Ozment and S. E. Schechter, "Milk or wine: does software security improve with age?" in *USENIX-SS'06: Proceedings of the 15th USENIX Security Symposium*. Berkeley, CA, USA: USENIX Association, 2006, pp. 93–104.

[14] Coverity, Inc., "Coverity scan open source report 2009," Coverity, Inc., Tech. Rep., 2009.

[15] E. Rescorla, "Is finding security holes a good idea?" in *Third Annual Workshop on Economics of Information Security*, Jul. 2006, http://www.dtc.umn.edu/weis2004/rescorla.pdf.

[16] SANS, "@Risk: The consensus security alert," http://www.sans.org/newsletters/risk/, September 2009.

[17] (2009, Nov.). [Online]. Available: http://www.cs.princeton.edu/~blei/lda-c/

[14]http://www.sans.org/top25-programming-errors/
[15]http://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
[16]http://www.securityfocus.com/archive/1

[18] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2008.

[19] E. Dimitriadou, K. Hornik, F. Leisch, D. Meyer, and A. Weingessel, *e1071: Misc Functions of the Department of Statistics (e1071), TU Wien*, 2009.

[20] S. Neuhaus and T. Zimmermann, "Security trend analysis with CVE topic models," Department of Computer Science, University of Calgary, Tech. Rep. 2010-970-19, Aug. 2010.

Appendix A. Replication Guide

LDA is a general technique that can be used to analyse topics for all kinds of corpora. Also, the techniques to extract graphs like Figure IV are similar for all corpora. We therefore give a step-by-step guide how to replicate our results on the CVE data or on other corpora.

1) *Gather data.* Gather a corpus of n documents describing vulnerabilities. Each document must carry a time stamp. Also choose the number of topics N.
2) *Stem words, apply stop words (optional).* Subject the words in the documents to a stemming algorithm and to a list of stop words.
3) *Compute vocabulary and word counts.* The result of the previous step(s) is a set of n documents containing a total of m unique words. Enumerate the unique words from 1 to m and build a matrix $\langle m_{jk} \rangle$ where m_{jk} is the number of occurrences of word k in document j. This matrix will generally be sparse.
4) *Run LDA.* Now run the LDA software on $\langle m_{jk} \rangle$. We used David Blei's LDA implementation for C [17]. The result will be a matrix $\langle z_{jl} \rangle$ ($1 \leq j \leq n$ and $1 \leq l \leq N$) where z_{jl} is the topic that was assigned to word l in document j.
5) *Compute post-hoc probabilities.* Compute the post-hoc probabilities using Equations (1), (2), (5) and (6).
6) *Join topics (optional).* If you find that some of the topics produced by LDA are really equal, join them using Equations (3) and (4). The order of this and the previous step can be interchanged; in this case, you will have to devise a mapping from original topics to joint topics and adjust $\langle z_{jl} \rangle$ accordingly. The result will however be the same.
7) *Plot results.* At this point, you have for each unique topic its overall importance, its average annual change, its development in importance during the relevant time period, and a list of the most frequent words in the topic. We used R [18] and the e1071 package [19] to make plots like Figure 3, and a number of custom Perl scripts to create tables like Table IV. We used the sparklines feature of Microsoft Excel 2010 to create the sparklines of Tables IV–VI

To facilitate replication of this study, an archive file with all the data and scripts is available in a technical report [20] at http://hdl.handle.net/1880/48066.

Table IV
Relative importance, all 28 topics. Topics are ordered alphabetically, followed by unnamed topics. The y axes are all equal, to facilitate comparison.

Topic	Trend	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009
Application Servers		1%	1%	1%	1%	1%	1%	3%	3%	3%	5%
Arbitrary Code		5%	4%	5%	5%	5%	5%	6%	7%	7%	6%
Arbitrary Code (IE)		3%	2%	2%	2%	2%	1%	1%	1%	1%	1%
Arbitrary Code (PHP)		0%	1%	1%	2%	2%	3%	11%	8%	3%	2%
Buffer Overflow		19%	18%	17%	18%	15%	10%	8%	10%	8%	11%
CVE Issues		1%	2%	2%	2%	2%	3%	4%	4%	2%	2%
Credentials Management		5%	6%	6%	5%	4%	4%	3%	3%	4%	3%
Cross-Site Request Forgery		2%	2%	2%	2%	2%	3%	2%	3%	3%	3%
Cross-Site Scripting		0%	1%	6%	6%	8%	13%	13%	9%	10%	10%
Cryptography		2%	2%	2%	2%	1%	2%	1%	1%	1%	1%
Directory Traversal		6%	8%	5%	4%	4%	4%	4%	4%	5%	4%
Firewalls		1%	2%	2%	2%	2%	1%	1%	2%	2%	2%
Format String		2%	3%	2%	2%	3%	2%	1%	1%	1%	1%
Information Leak		2%	2%	3%	3%	3%	3%	5%	4%	4%	3%
Java		3%	1%	2%	2%	2%	2%	2%	2%	3%	3%
Link Resolution		6%	6%	4%	4%	4%	3%	1%	2%	3%	1%
Linux		2%	2%	2%	3%	3%	3%	2%	3%	2%	3%
Message Boards		2%	1%	2%	1%	2%	2%	2%	1%	1%	1%
Microsoft Office		1%	1%	1%	1%	1%	1%	1%	1%	2%	2%
Microsoft Windows		4%	3%	3%	3%	2%	1%	1%	1%	1%	2%
Mozilla		1%	2%	2%	2%	3%	2%	2%	2%	2%	3%
PHP		0%	0%	1%	1%	1%	3%	4%	3%	4%	4%
Privilege Escalation		12%	10%	8%	8%	5%	4%	3%	3%	3%	2%
Resource Management		14%	12%	12%	13%	13%	9%	6%	8%	7%	10%
SQL Injection		1%	1%	2%	2%	4%	11%	10%	8%	17%	10%
N/A		1%	1%	1%	1%	1%	1%	1%	1%	1%	1%
N/A		4%	5%	5%	3%	3%	2%	2%	2%	1%	2%
N/A		0%	0%	1%	1%	2%	1%	2%	2%	2%	2%

Table V
Relative importance of discovered causes.

Topic	Trend	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009
ActiveX		1%	2%	2%	2%	2%	2%	1%	4%	3%	2%
Application Servers		3%	2%	4%	3%	2%	1%	1%	1%	1%	3%
Buffer Overflow		17%	14%	14%	17%	13%	8%	6%	8%	7%	10%
Cisco IOS		3%	4%	5%	3%	3%	2%	1%	2%	1%	2%
Credentials Management		4%	4%	3%	2%	2%	1%	1%	2%	2%	2%
Cross-Site Request Forgery		1%	1%	1%	1%	1%	1%	1%	2%	3%	3%
Cross-Site Scripting		11%	9%	15%	14%	17%	21%	22%	16%	18%	17%
Directory Traversal		6%	12%	7%	6%	5%	5%	6%	6%	7%	5%
Format String		2%	2%	2%	3%	3%	2%	1%	2%	1%	2%
Internet Explorer		2%	3%	3%	4%	3%	3%	4%	3%	2%	2%
Java		1%	1%	1%	2%	2%	2%	2%	2%	2%	2%
Microsoft Office		2%	2%	2%	1%	1%	1%	1%	1%	1%	2%
Mozilla		2%	2%	2%	2%	2%	2%	1%	1%	2%	4%
PHP		5%	6%	6%	8%	8%	10%	19%	17%	10%	8%
SQL Injection		10%	8%	11%	12%	15%	23%	21%	19%	28%	21%
Security Appliances		1%	3%	2%	3%	2%	1%	1%	1%	1%	1%
N/A		4%	2%	2%	1%	2%	2%	1%	2%	1%	1%
N/A		5%	5%	4%	3%	3%	2%	1%	2%	1%	2%
N/A		4%	3%	2%	3%	3%	1%	1%	1%	1%	2%
N/A		2%	3%	3%	2%	2%	2%	1%	2%	2%	2%
N/A		2%	1%	1%	1%	2%	1%	1%	1%	1%	2%
N/A		4%	3%	3%	3%	3%	2%	2%	2%	2%	3%
N/A		4%	2%	3%	2%	2%	2%	2%	2%	2%	2%
N/A		3%	4%	2%	2%	2%	2%	1%	2%	1%	1%

Table VI
Relative importance of discovered impacts.

Topic	Trend	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009
Arbitrary Code		15%	18%	20%	25%	22%	17%	23%	24%	17%	24%
Arbitrary Script		17%	17%	20%	21%	24%	39%	37%	31%	43%	35%
CVE Dispute		1%	2%	2%	1%	2%	3%	5%	5%	3%	5%
Denial of Service		30%	23%	21%	22%	21%	13%	8%	12%	9%	11%
Information Leak		22%	23%	18%	13%	14%	13%	14%	16%	17%	15%
Information leak		1%	1%	3%	4%	3%	4%	3%	2%	1%	1%
Privilege Escalation		11%	12%	11%	10%	10%	8%	6%	7%	8%	7%
Resource Abuse		2%	2%	2%	1%	1%	1%	1%	1%	1%	1%
N/A		1%	2%	2%	2%	1%	1%	1%	1%	1%	0%
N/A		0%	0%	0%	0%	0%	0%	1%	1%	0%	0%
N/A		0%	0%	0%	0%	0%	1%	1%	1%	1%	1%
N/A		0%	0%	0%	0%	0%	0%	1%	1%	0%	0%