# Which Faults are Security Faults?

Michael Gegick, Tao Xie, Laurie Williams, Pete Rotella
North Carolina State University Department of Computer Science, Cisco Systems, Inc.
{mcgegick}@ncsu.edu, {xie, williams}@csc.ncsu.edu, protella@cisco.com

## Abstract
The subtleties associated with security faults can sometimes be missed by developers and testers. When developers encounter a fault and are unaware of the security implications, they are less likely to report it as a security fault to a security team. Security engineers may know the best remediation for a security fault and have the authority to elevate the priority of that fault. Limited resources (e.g., budget, person-hours) preclude a security team from examining all faults in a database to identify which faults are security-related. Therefore, an automated means to identify which faults in a fault database are security faults can improve the security assurance of the software. We used SAS Enterprise Miner to automate the textual analysis of fault reports of a Cisco software system in a fault database. We created a predictive model based on a neural network that takes as input the textual description of a fault report and assigns a probability that the fault is security-related. Preliminary results indicate that the model correctly predicted 91.4% of the system's security faults. We applied the model to three other different Cisco software systems and showed that 67% of the security faults were correctly predicted to be security-related. The results indicate that the model is very effective for the system that it was trained on, and is moderately effective for other systems; it may require training on security faults specific to other systems to achieve similar performance.

## 1 Introduction
An analysis of all of the faults[1] in a fault database would likely show that security faults (vulnerabilities[2]) are labeled as non-security problems and have not been submitted to a security engineer. While all of the faults in the database are likely to be fixed, the fix may be delayed until after the software is released to the customers possibly because the fault is seemingly innocuous. If a security engineer determines that a fault is actually a vulnerability, then they may elevate the priority to ensure the fault is remediated before release. Additionally, the security engineer may know the best mitigation strategy for a vulnerability and thus the security assurance[3] of the software will likely improve if each vulnerability in the fault database is recognized and mitigated timely and appropriately.

One possible reason why vulnerabilities are labeled as non-security problems in a fault database is the lack of security expertise in the software development team [6]. The subtleties associated with security faults can sometimes be missed by developers and testers. For example, if a tester uses a protocol fuzzer to exploit a buffer overflow in protocol implementation code, then the tester is likely to report the observed failure as a security failure in their fault database. However, a tester may enter "O'Neill" into an unguarded SQL input field and find that the single quote in the string causes the SQL parser to throw an exception. The tester may be unaware that the failure could be a security fault and thus may not label the fault as a security problem.

Limited resources preclude security engineers from finding and fortifying all vulnerabilities in a fault database and so an automated means of identifying which of the faults are security-related would reduce the time spent in looking for the vulnerabilities. *Our research objective is to automate the analysis of developers' and testers' textual descriptions of faults in a fault database to predict which faults are vulnerabilities.* We used SAS Enterprise Miner[4], which includes SAS Text Miner[5] (STM), to analyze the

[1] A fault is an incorrect step, process, or data definition in a computer program [4].
[2] A vulnerability is an instance of a [fault] in the specification, development, or configuration of software such that its execution can violate an [implicit or explicit] security policy [5].
[3] Security assurance provides "confidence that an entity meets its security requirements, based on specific evidence provided by the application of assurance techniques" [2].
[4] http://www.sas.com/technologies/analytics/datamining/miner/

textual descriptions of faults and predict which faults are vulnerabilities. We present preliminary findings in this report and illustrate our current efforts for validating our predictive model.

At this point in our research, it is unclear if the predictive model will accurately identify vulnerabilities versus traditional software faults. The dataset used may include vulnerabilities that have already been publicly disclosed as well as vulnerabilities that are under active investigation. This research has no bearing on the disclosure status of any potential security vulnerability and does not discern between repaired or open faults. Any new vulnerabilities that are identified as a result of this research will be evaluated by the appropriate security team and will be subject to the Cisco vulnerability disclosure policy located on http://www.cisco.com/security. No details of any vulnerability including disclosure status will be communicated in this research.

## 2 Cisco case study

At Cisco, when developers or testers identify a vulnerability, they report the problem to the Cisco Security Evaluation Office (SEO). The SEO tracks Cisco vulnerabilities including those reported externally. The developers and testers can also submit problems to the SEO when they suspect that the fault is security-related. If the SEO confirms their suspicion, then security fortifications begin on that vulnerability. Developers and testers may not report all vulnerabilities to the SEO because they are unaware that the problem they encounter is security-related. Therefore, we train and evaluate a predictive model to detect vulnerabilities that reside in the fault database. We assume that developers and testers always submit obvious security problems to the SEO and we are thus less concerned about these vulnerabilities in our model than the ones that are subtle and can escape the untrained eye.

We investigate if the textual description of vulnerability reports resemble the textual description of vulnerabilities that are not submitted to the SEO. We use SAS Enterprise Miner[6] to automate textual analyses of faults in a database and predict which faults are vulnerabilities. SAS Enterprise Miner includes STM and predictive modeling techniques such as neural networks, memory-based reasoning, regression, and trees that we use in our analyses. Our approach is a dichotomized approach where faults are predicted to be either vulnerable or not vulnerable. We next state our hypotheses.

**$H_0$** – the textual descriptions in vulnerability reports cannot be used to predict which faults, originally classified as non-security faults by a developer or tester, in a fault database are actually vulnerabilities.

**$H_A$** – the textual descriptions in vulnerability reports can be used to predict which faults, originally classified as non-security faults by a developer or tester, in a fault database are actually vulnerabilities.

The Cisco Product Security Incident Response Team (PSIRT) is a security team that handles all vulnerability evaluations at Cisco, which include assigning Common Vulnerability Scoring System (CVSS)[7] scores to vulnerabilities. A CVSS score is a rating between 0.0 and 10.0 where 10.0 has the highest security impact. We are currently working with PSIRT to confirm the results of our predictive model.

## 3 Research approach

### 3.1 Application of the model on known Cisco vulnerabilities

We collected vulnerabilities reported by developers, testers, and those found in the field between the start January 2006 and December 2008 for a Cisco software system that we call System A. The details of System A are not disclosed here due to confidentiality[8]. The vulnerabilities were made available to us by the SEO. We use the descriptions of the vulnerabilities in the Cisco fault database to train our model to detect vulnerabilities that are not submitted to the SEO. At Cisco, all fault reports in the fault database

---

[5] http://www.sas.com/technologies/analytics/datamining/textminer/
[6] All analyses performed with SAS Enterprise Miner 4.3 in SAS 9.1.3 Service Pack 3.
[7] http://www.first.org/cvss/
[8] All security data have been slightly and consistently modified for confidentiality reasons.

have a Description field that provides a textual description of the fault. Our analysis is applied exclusively to that data field.

In our analysis, the unique word count from the Description fields from all the fault reports summed to 1,431. We created a security keyword subset of the 1,431 words to train our predictive model. We manually removed words that did not have any relationship to software security. For example, we removed words such as developer names, prepositions, and conjunctions. Our final security keyword subset has a word count of 318 and is referred to as a *start list* by STM. STM uses only the words in the start list during the text mining analysis. Words that do not exist in the start list, but are in the fault report description are not considered by the analysis. The specific words in the start list are not disclosed here due to confidentiality. We also included a synonym list, based on the start list, in the analysis to enhance the text mining analysis. To determine the efficacy of our start list and synonym list, we compare the results of using them to the results of using the STM default *stop list*. A stop list is a data set of words that STM excludes from the text mining analysis. The stop list is not specific to security, but is available for any type of STM analysis. The stop list includes prepositions and conjunctions. Additional details of the SAS default stop list are SAS confidential and thus not reported. STM does not contain a default start list.

We collected the same number of non-security faults (as the number of vulnerabilities) in System A to include in the analysis to determine if STM can distinguish between vulnerabilities and non-security faults. We experimented with different predictive techniques including neural networks, memory-based reasoning, regression, and trees. We also consider two options with STM: singular value decomposition (SVD) and roll-up terms. STM parses textual data into a term-document frequency matrix. Each entry in the term-document frequency matrix represents the number of times a term appears in a fault report. With SVD, STM determines the best least square fit to a matrix, given a number of dimensions [7]. SAS documentation [7] recommends between 30 and 200 dimensions for predictive applications. The roll-up term option uses the highest weighted terms in the document collection and has variables for each document for the weighted frequencies of each term in the fault reports [7]. The roll-up terms technique works best when documents are short [7]. The word count for System A fault reports ranges 40 to 2,021 and so SVD is likely the better candidate for our analyses.

We partitioned the data set that contained security faults and non-security faults for System A into a 60%, 20%, 20% training, validation, and testing ratios, respectively. The training and validation partitions are used to train and evaluate the model and the remaining 20% is used to test the model on data that the model had not "seen" before. The values of the ratios were suggested by STM documentation [7]. According to Hastie et al. [3], defining the ratios of training, validation, and test sets should be determined by the signal-to-noise ratio. In our setting, the signal corresponds to the reported vulnerabilities in System A. The noise corresponds to fault reports that are flagged by the model as vulnerabilities, but are actually non-security problems. In our research context of applying text mining, we cannot be certain if any of the non-security faults are actually vulnerabilities ahead of time (which is the problem to be addressed by our approach). Therefore, we cannot determine the optimal training, validation, testing ratios because we do not know our noise-to-signal ratio. We therefore show additional results of 50%-20%-30% and 40%-20%-40% ratios to investigate how our model performs with these other ratios.

We provide measures of the accuracy to indicate the overall success of the model, precision to indicate how high percentage of non-security fault reports are false positives, and the recall to represent how well the model correctly classified vulnerable reports. We now define the following terms for our measures:

**True positive (TP)** – a vulnerability report correctly classified as a vulnerability by the model.
**False positive (FP)** – a fault report misclassified as a vulnerability by the model.
**False negative (FN)** – a vulnerability report misclassified as a non-security problem by the model.
**True negative (TN)** – a fault report correctly classified as a non-security problem by the model.

**Accuracy** - (TP + TN)/(TP + FP + TN + FN)
**Precision** - (TP)/(TP+FP)
**Recall** - (TP)/(TP+FN)

### 3.2 Application of the model to the Cisco fault database

We next apply the model (trained with the techniques in Section 3.1) to the faults associated with System A in the fault database that have not been submitted to the SEO and thus have not been scrutinized by either the SEO or PSIRT teams. The faults that we examine were submitted to the fault database between January and June 2007. We will submit a sample of faults that were evaluated by the model to the PSIRT team for CVSS scoring. The PSIRT will not be given the model-generated probability that a fault is vulnerability as the probability could bias their CVSS score.

We envision the PSIRT using STM by ranking the probability that the fault is a vulnerability in descending order. The higher probabilities may show that characteristics of the fault report make the problem a vulnerability; lower probabilities may indicate security characteristics are not present. The PSIRT will start their analyses at the top of the ranked list of fault reports and continue to downward until a threshold where false positives become too frequent that time spent analyzing the model out is wasteful. Therefore, we will submit the top ranked fault reports to the PSIRT, but we also must submit fault reports with low probabilities to the PSIRT to determine whether the model assigned low probabilities to vulnerabilities. Based on the PSIRT analyses, we can estimate a threshold for which they can use when the model is put into practice.

### 3.3 Application of the model to three different Cisco software systems

We also used the predictive model trained from the data set in System A to predict which faults were security faults in three other Cisco software systems denoted as System B, System C, and System D. These software systems use different technologies than System A. Details of these systems are not disclosed due to confidentiality. The vulnerabilities given to us by the SEO were identified between January 2003 and December 2008.

## 4 Results

### 4.1 Model performance on known Cisco vulnerabilities

We found that the neural network produced slightly better results than memory-based reasoning, and that both performed much better than the regression and trees predictive techniques. We also found that singular value decomposition (SVD) worked better than the "Roll up terms" technique available in STM. We tried 30, 50, 100, 150, and 200 dimensions in our SVD analyses and found that the lowest false negative rates with 200 dimensions and so we choose 200 dimensions for our model.

Preliminary results show that STM can identify 91.4% of the vulnerabilities reported in System A as shown in Table 2. These results are observed in the test set (20% of the data set). The percentages reported in Table 1 are decided by a threshold available in STM. STM allows us to choose a probability that decides which faults are vulnerabilities. In our setting, we use a threshold of 35% where any fault that had a probability of 35% or greater was decided to be a vulnerability. We choose that threshold because the threshold minimized the false negative rate while maintaining an acceptable true positive rate. It is unclear whether the recall rate decreases due to the larger test set size or due to the less training data set size or some combination of both factors. We do suspect that the more data we use to train, validate, and test the model, the better the performance of the model. Security vulnerabilities are few in number, and as we use more vulnerabilities with STM, we suspect we could improve the success of our model.

**Table 2. Model evaluation on System A with custom start list and synonym list.**

| Training-validation-test ratio | Accuracy | Precision | Recall |
|---|---|---|---|
| 60-20-20 | 78.6% | 74.0% | 91.4% |
| 50-20-30 | 50.2% | 55.9% | 65.5% |
| 40-20-40 | 58.9% | 57.5% | 69.0% |

In Table 3, we show the results of performing the same analysis, but with the SAS default stop list. The results indicate that the SAS stop list has higher overall accuracy, precision, and recall in most cases, but not as well as the 60%-20%-20% ratio in Table 1. Therefore, we choose our start list and synonym list for the analysis with the PSIRT described in Section 3.2. Additionally, as we find more vulnerability reports,

we can further train the model and enhance the start list and synonym list, which we could afford for better results than the SAS default stop list. The accuracy and precision formulas include the false positive rate in the formulas. At this stage of our research, we are uncertain whether false positives are really false positives because no inspection was conducted yet on these fault reports to determine whether they were security-related.

**Table 3. Model evaluation on System A with the SAS default stop list.**

| Training-validation-test ratio | Accuracy | Precision | Recall |
|---|---|---|---|
| 60-20-20 | 77.9% | 73.8% | 83.8% |
| 50-20-30 | 77.2% | 72.5% | 86.8% |
| 40-20-40 | 75.6% | 72.3% | 81.1% |

We observed that the smallest test set has the highest recall or ability to correctly classify fault reports as security-related. As the test set increases in size -- at the cost of a smaller training set -- the ability to correctly classify vulnerabilities decreases to near 70%. At this point in our case study, we are concerned only about the false negative rate (100%-recall rate). The false positives that influence the precision and accuracy may not really be false positives as we describe in Section 5. We suspect that a percentage of the false positives are actually true positives. In these cases, developers and testers did not know that such a fault was security-related and did not submit the fault to the SEO. Decreasing the false negative rate may be achievable by training the model on larger data sets and enhancing the start and synonyms lists.

### 4.2 Model performance on the Cisco fault database

The analysis (described in Section 3.2) is currently in progress and the results will be reported soon when they are available. We show initial findings and outline our next steps.

The probability distribution of fault reports generated by our model for a six-month period of System A is shown in Figure 1. According to Figure 1, most of the faults in the database are not security-related, but a small percentage have a high probability of having characteristics in the textual description that resemble vulnerabilities. Previous studies have shown that vulnerabilities are rare events [1] and thus this distribution is consistent with that observation.

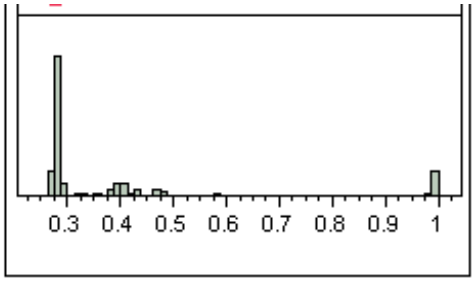


**Figure 1. The probability distribution from our model. The x-axis represents the probability that a fault is a vulnerability. The y-axis represents the count (not disclosed due to confidentiality) of fault reports.**

We ranked the fault reports in the descending order of probability that the fault is a vulnerability. Since our sample size is large and we want to determine which probabilities are most likely to indicate a vulnerability, we divide the fault reports into quartiles where each quartile contains approximately the same number of fault reports. The fourth quartile will contain faults with the highest probability and the first quartile will contain fault reports least likely to be security-related. The range of probabilities for each quartile is shown in Table 1.

**Table 1. Probabilities for fault reports by quartile.**

| Quartile | Probability that fault report is security-related |
|---|---|
| Q4 | 45.8%-99.9% |
| Q3 | 28.5%-45.7% |
| Q2 | 28.1%-28.5% |
| Q1 | 24.1%-28.1% |

An equal sample from each quartile will be given to the PSIRT team. The sample size was calculated based on approximately 140 person hours that we estimated the PSIRT would require to assign CVSS scores to the fault reports. The sample size of faults from each quartile is confidential and thus not disclosed. By sampling each quartile, we gain estimation of how many vulnerabilities may exist in each quartile. If the same number of vulnerabilities are in each quartile, then the probability ranking is of little value. We observed that the fourth quartile has a large range (45.8%-99.9%) and thus we selected the fault reports with the highest probabilities from each quartile. By sampling the fault reports from the fourth quartile we suspect we will have a better true positive rate according to Figure 1. Since an equal sample of the highest probabilities from each quartile are selected, we believe this sample is a fair choice. We will use a two sample t-test to compare the average CVSS score between each quartile. Upon the release of the model into true automation at Cisco, we can advise the PSIRT team to start at probability 100% and keep looking down the list to some probability, x, to find the vulnerabilities. Looking beyond x will result in too many false positives and thus wasted time on analyses. As reported in Section 4.1, the threshold for known vulnerabilities is 35%. The PSIRT evaluation will give us insight if the threshold is consistent with fault reports that were not originally labeled as security problems by developers and testers.

To determine the efficacy of the predictive model, we will take a random sample from faults returned by PSIRT and compare the average CVSS score of those faults to an equal number of vulnerabilities in the each quartile of the model. If the count of vulnerabilities and average CVSS score of those in the fourth quartile are significantly higher than from a random sample, then we will accept the model as a valid means of identifying vulnerabilities in the fault database.

One challenge for the model is that the start list and synonym list used for the System A analysis consisted of keywords such as “vulnerability”, “attacker”, and “exploit.” The textual description in the faults that we will analyze that are not reported to be security problems, but really are security problems are unlikely to have such security-related diction.

### 4.3 Model performance on three different Cisco systems

We applied the predictive model based on System A vulnerabilities to three different Cisco systems that we call Systems B, C, and D. The threshold for our analysis was kept at 35% as was done in the System A analysis. In addition, the training-validation-test ratio was maintained at 60%-20%-20%, respectively. The model correctly predicted 65% of the vulnerabilities from these three systems. We do not know the ratio of non-security faults to security faults because we do not know the true ratio of vulnerabilities to non-security faults. Therefore, we have estimated the ratio at 1:1, 1:50, and 1:100 to provide some guidance for our model. The accuracy, precision, and recall rates are provided in Table 4. We suspect that the reason for the lower recall rate for Systems B, C, and D is that the style of writing fault description by different developers and testers, and that the vulnerabilities in the three systems may be different than the vulnerabilities in System A. The low precision is due to many false positives (non-security fault reports) above the 35% threshold. The sample size in this analysis is much larger than the analysis in Section 3.1. Also, as explained in Section 3.1, some percentage of the false positives could actually be vulnerabilities. More analyses are required to confirm these results.

**Table 4. Evaluation of the model on Systems B, C, and D.**

| Vulnerability-to-non-security-fault ratio | Accuracy | Precision | Recall |
|---|---|---|---|
| 1:1 | 69.6% | 71.3% | 67.0% |
| 1:50 | 67.4% | 3.0% | 49.5% |
| 1:100 | 73.3% | 1.9% | 49.5% |

## 5 Conclusions and Future Work

We have evaluated a predictive model on a Cisco software system to determine if the textual descriptions in vulnerability reports can be used to predict if other fault reports (that are known vulnerabilities) are likely to be vulnerabilities. The results indicate that the model is useful when the model is applied to the system it was trained on, but is less accurate on three software systems that it was not trained on. The vulnerabilities of the three other systems may be different than the first system, which may hinder the predictive power of the model. Based on these results, we are applying the predictive model to samples of the Cisco fault database to determine if fault reports that were not previously labeled as vulnerabilities are indeed security-related.

In our future work, we will train the model on the vulnerability reports from the three other Cisco software systems in an effort to increase the model's accuracy. This training will include adding the security-related keywords from the text descriptions of the vulnerability reports. Also, we need to determine if a keyword search (e.g., a grep) can perform as well as Text Miner. Currently, our start list includes 318 words and thus a keyword search for 318 words may not be as robust as STM. Lastly, based on the PSIRT feedback, we will have a dataset that shows which fault reports are vulnerabilities and which are non-security problems. We can train our model on these data to determine how the model performs on vulnerabilities reported to the SEO and to a new set of fault reports that have not been examined by the SEO or the PSIRT.

## 6 Acknowledgements

This work is supported by the National Science Foundation under CAREER Grant No. 0346903. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

## 7 References

[1] O. H. Alhazmi, Y. K. Malaiya, and I. Ray, "Measuring, analyzing and predicting vulnerabilities in software systems," *Computers & Security*, vol. 26, no. 3, pp. 219-228, May 2006.
[2] M. Bishop, *Computer Security: Art and Science*, Boston, Addison-Wesley, 2003.
[3] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*, New York, Springer, 2001.
[4] ISO/IEC 24765, "Software and Systems Engineering Vocabulary," 2006.
[5] I. Krsul, "Software Vulnerability Analysis," PhD Thesis in Computer Science at Purdue University, West Lafayette 1998.
[6] G. McGraw, *Software Security: Building Security In*, Boston, Addison-Wesley, 2006.
[7] SAS Institute Inc., "Getting Started with SAS 9.1 Text Miner," Cary, NC, 2004.