Association for Information Systems
AIS Electronic Library (AISeL)

AMCIS 2004 Proceedings

Americas Conference on Information Systems (AMCIS)

December 2004

A Method for Mining XML Data Describing Open-Source and Proprietary Software Vulnerabilities

J. Art Gowan
University of North Carolina at Wilmington

Richard Mathieu
Saint Louis University

Jie Chen
Saint Louis University

A Method for Mining XML Data Describing Open-Source and Proprietary Software Vulnerabilities

J. Art Gowan, Jr.
University of North Carolina at Wilmington
gowanj@uncw.edu

Richard G. Mathieu
Saint Louis University
mathieur@slu.edu

Jie Chen
Saint Louis University
chenj3@slu.edu

ABSTRACT

There is an ongoing debate about the magnitude and characteristics of software vulnerabilities found in open-source versus proprietary software. A software vulnerability is defined as a flaw in a piece of software that has been discovered and exploited. Once publicized, typically by an electronic announcement, the vulnerability is posted to one of several competing online security information services. The software vendor will then respond with a scripted patch or remediation. Data related to software vulnerabilities is available publicly (i.e., CERT, SecurityTracker, Mitre, and SANS) and is often stored in an XML format. However, XML presents several challenges for text mining software. The purpose of this paper is to describe a method for mining XML data that sufficiently addresses these concerns in the context of software vulnerability research. Preliminary results are presented.

Keywords

Text mining, data mining, XML, security, software vulnerabilities, open-source software

INTRODUCTION

The cover story of *e-Week*, September 30th, 2002, titled "*Open Source: A False Sense of Security?*" (Fisher, 2002) starts by stating "Advocates claim open platforms have a leg up on big vendors' products, but recent flaws cast doubt." The author poses the ongoing question: "So who is right? Does patent-protected development behind closed doors produce more secure software? Or does the collaborative, open-source community, where thousands of smart, independent developers are poised to spot and fix security problems?"(p.20). Just below that cover story is an article "*Microsoft Puts Meat Behind Security Push*". The debate goes on and descriptive statistics like, average days to respond to a security vulnerability with a patch, or frequency of patches, comparing Red Hat (open-source) to Microsoft (closed-source), continue to be reported, but no true empirical results are being reported that objectively address the question of the cover story. The following reports an effort to address these issues through empirical research which involves the use of text mining, a specialized application of data mining.

Software Vulnerabilities

A software vulnerability taxonomy is emerging and is beginning to be more widely accepted. The following was primarily taken from Browne, Arbaugh, McHugh, and Fithen, (2000) and CERT Coordination Center (CERT/CC)'s FAQs (2002). A **flaw** is a required precondition in a piece of software that allows it to be exploited. Once **discovered** and **exploited**, the flaw becomes a **vulnerability**. The process starts with a **probe.** The probe may be performed by a hacker, or the vendor, or a third party who is for some reason analyzing a piece of software. Once discovered, exploitation means that the flaw allows the software to be manipulated or used in an unintended fashion that could lead to damage to the installed system. Depending upon who discovers the flaw, it may or may not be reported prior to an **incident** in which the vulnerability is used to carry out an **attack** or **intrusion.** Once **publicized** (**disclosure**), typically by electronic announcement followed by written announcements in industry and professional publications, the vulnerability is posted to one of several competing online security information services (SecurityTracker, Mitre, SANS and CERT). The vendor will then respond with a **scripted patch** or **remediation.** The time span from initial announcement so that the public is made aware to the posting of a patch or

remediation is the **recess**. Once the patch or remediation proves to be successful, the vulnerability may become **passé**, and eventually **dies**.

Arbaugh, Fithen, and McHugh (2000) performed a number of Windows case studies that showed how systems often remain vulnerable long after patches are made available. They proposed a "life-cycle" model which suggests that intrusions increase once a vulnerability is discovered, at an increasing rate until a patch or fix is completed, but due to a lag in implementation, the rate of intrusions continues to rise and then drop off once fixes are put into place. They concluded that most intrusions could be prevented with better systems management through more timely deployment of updates and patches. They also concluded that in the long term, prevention, through structured approaches, is preferred, never allowing the vulnerability to exist. Browne, et al. (2000) used regression analysis, using a square root transformation on time, which resulted in good predictive power for the accumulation of vulnerabilities across a number of different cases. They concluded "that the models will aid in predicting the severity of subsequent vulnerability exploitations, based upon the rate of early incident reports." Browne, et al. (2000) did not perform any analysis to test this model. Instead, they focused on the mathematical modeling of the duration of recess as a function of frequency of reported incidents.

Open versus Closed Systems Software: An Ongoing Debate

For over a decade there has been an ongoing debate whether open source system software improves or diminishes system security. By restricting and controlling the release of the source code, the difficulty of the hacker's search for flaws is increased significantly, but when an incident occurs and vendor becomes aware of the vulnerability, the resources to solve the problem are limited to those available to the vendor. Reavis (2000) compared recess duration of a total of 100 advisories in 1999 issued by Red Hat, Microsoft and Sun Microsystems. Red Hat is a major provider of Linux, an open source system, as compared to Microsoft and Sun, both of which operate in a closed source environment. The averages days of recess was 11.23 for Red Hat, compared to 16.10 and 89.50 for Microsoft and Sun respectively. No statistical analyses were performed to indicate statistically significant differences, but the general support was in favor of open source code. The assumption is that the number of people who could analyze the problem and suggest a solution is almost limitless, including the hackers themselves. As the status of the patches and their effectiveness are reported, the hackers lose interest and move onto the next vulnerability. Therefore, many more programmers and specialists peruse the code for flaws in open systems than any closed system vendor could ever afford to consult.

Witten, Landwher, and Caloyannides (2001) note that there is substantial debate in the popular literature over the merits of open source, but most is anecdotal and very few empirical studies with hard data and rigorous analysis exists. They also note that "development of proper metrics for system security is a vast topic-often addressed but rarely with satisfying results…"(p.59). They note the recess metric does appear to be widely considered a relevant issue as recess statistics similar to those reported in Reavis (2000) are now published on Security Portals Web site. In addition, they make note that 1.) all flaws are not equally bad (relating to vulnerability severity), 2.) taking the time to fix many flaws may be better than fixing fewer faster, 3.) controlling for differences between operating systems should be considered, and 4.) there may be differences in effectiveness between proposed fixes, especially in that there would be multiple approaches proposed to open system vulnerabilities.

Data Stores for Vulnerability Research

Fortunately, several data stores exist for software vulnerability research. Different vulnerability data exists at CERT (*www.cert.org*), SecurityTracker (*www.securitytracker.com*), Mitre (*cve.mitre.org*), and SANS (*www.sans.org*). The SANS (SysAdmin, Audit, Network, Security) Institute and the National Infrastructure Protection Center (NIPC) at the FBI keep an updated set of documents summarizing the Ten Most Critical Internet Security Vulnerabilities. The SANS Top Twenty is actually two Top Ten lists: the ten most commonly exploited vulnerable services in Windows and the ten most commonly exploited vulnerable services in UNIX and Linux. Although there are thousands of security incidents each year affecting these operating systems, the overwhelming majority of successful attacks target one or more of these twenty vulnerable services. The Top Twenty is a consensus list of vulnerabilities that require immediate remediation. It is the result of a process that brought together dozens of leading security experts. The Common Vulnerabilities and Exposures (CVE) hosted at Mitre is a list of standardized names for vulnerabilities and other information security exposures. CVE aims to standardize the names for all publicly known vulnerabilities and security exposures. The goal of CVE is to make it easier to share data across separate vulnerability databases and security tools. While CVE may make it easier to search for information in other databases, CVE should not be considered as a vulnerability database on its own merit.

The CERT Coordination Center, a federally funded research and development center operated by Carnegie Mellon University, publishes information on a wide variety of vulnerabilities. Descriptions of these vulnerabilities are available from their web site in a searchable database format, and are published as "CERT Vulnerability Notes". One can search or browse Vulnerability Notes by several key fields, including name, vulnerability ID number, CVE name, date updated, date public, or metric. The CERT/CC Vulnerability Notes database is cross-referenced with the Common Vulnerabilities and Exposures (CVE) catalog. SecurityTracker is a service that helps organizations track the latest security vulnerabilities. This service monitors a wide variety of Internet sources for reports of new vulnerabilities in Internet software, hardware, and/or services, and provides subscribers with a timely and reliable source for vulnerability notification. Vulnerabilities are categorized by *vendor*, *operating system*, *cause* of vulnerability, primary *impact* of vulnerability, and other parameters. The SecurityTracker web site can be searched for specific vulnerabilities. The basic SecurityTracker vulnerability feed is stored in an XML format.

Text Mining: Methods and Processes

Some of the basic concepts and technologies being used to implement text mining applications is not new and is part of the field of knowledge management and knowledge discovery by data. Content analysis is a research method developed in the 1960's (verify & provide reference) used to make inferences from text about the content of the message, as well as the author and potentially the audience (Weber, 1990). The primary objective is to extract generalized meaning of paragraphs or papers by classifying word use. For example, a political scientist might classify speeches into categories such as "economic", "political" or "welfare" focused. Keywords were analyzed based upon the use of specialized dictionaries or keyword lists developed by experts. The primary objective of text mining is different, but some of the analytical steps in the process, like the development of word-frequency-lists, are similar.

The field of text mining is still in its infancy, but when looking across products and related research, there is a common theme emerging with respect to the general process of text mining and nomenclature. Some packages focus on specific parts of the process while others are more comprehensive in nature. Some are extensible automated yet allow human intervention while others may require more manual operation. The following are stages of the text mining process:

I. Information retrieval (IR) includes tools to search and identify unstructured and semi-structured sources of data to be mined. More traditional definitions of IR include human directed searches while more recent applications include agents that can perform automated searches for specified topics or concepts.

II. Text preparation or pre-processing provides tools similar to those found at the back-end of data warehouses for "cleansing" the data. It may include a process of breaking the text into conceptual chunks (like chapters, sections, paragraphs). A substantial amount of text can be removed through the use of exclusion rules and stub/stop lists. Stub or stop lists include words such as "a", "the", "it", "to", "of", etc. that can be removed, which can also be edited and updated by users for specific applications. Stemming algorithms are used to develop rule sets to address the issue of terms that are essentially the same (e.g. "selling", "sales" and "sell"), so they can be reduced to the stem word. The opposite issue may arise where a concept is a combination of words, which must be maintained as a string and not separated. The issue of case must also be addressed. Often, there will be no loss of information if all text is reduced to lower case, but there is a need to recognized special text objects, like proper names. Other special issues arise when the content may contain special strings like e-mail addresses or URL's which must be treated more like proper nouns and should not be stemmed or parsed. Other cleansing may be required or translated depending upon the specific language, format, and requirements.

III. Text processing is at the heart of the text mining process. It utilizes different algorithms to identify categories concepts, called categorization or classification, and then clustering algorithms to establish the relationships between the categories. The most critical element of the text mining process lies in data categorization and clustering process and algorithms used to carry out those processes. The initial conceptual search often requires the use of natural language processing technologies. Category management tools often allow for subjective human intervention for the deletion, addition or reorganization of categories or the refinement of the results. Some tools offer both multilingual and cross-lingual capabilities. Clustering tools can also be used to not only identify category relationships but also carry out analysis of categories and clusters across time or different sets of text, adding additional dimensions, similar to that used in data mining to drill down through similar data across time or sources. Certain tools can also profile individuals (employees, suppliers, customers) and organizations and utilize this information in the clustering process, again to help identify patters and relationships between categories. The categorization/clustering tools may also be used interactively by users to manually query a data set (text) for specific concepts, if desired.

The product of the text processing is a taxonomy which reveals the concepts and their relationships. The taxonomy is essentially a semantic interpretation of the text. At this point structure has been extracted from the unstructured or semi-structured data. Depending upon the clustering tools used, certain metrics may be provided to identify the size or importance of each concept and the relative strength of the relationships (e.g. cohesion and distinctness). Multiple taxonomies are often generated across different sets of text and based upon different internal dictionaries and can ultimately be compared and analyzed.

This text processing stage is visually modeled by SPSS in Figure 1. SPSS's product LexiQuest Mine using linguistics techniques to understand and extract the underlying concepts, count and classify each, then models the detected relationships (SPSS, 2003).

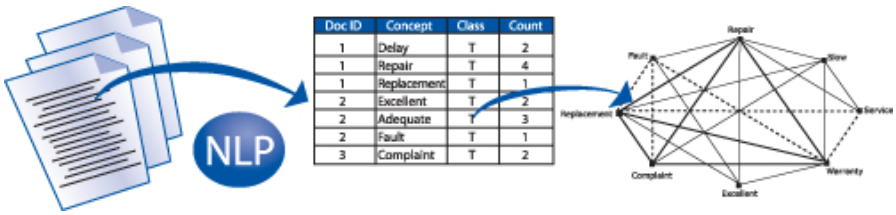

Figure 1

IV. Text analysis involves the statistical analysis of the taxonomies as well as their visualization, typically through a GUI interface. The analysis may be standardized and static or interactive, depending upon the tool. Of ten, text mining tools are a part of a data mining package and at this stage, the data mining analytical tools are used to analyze and provide statistical and visual models of the results. SPSS's visual model is created through the use of it data mining package, Clementine. SPSS describes text mining as a data source for data mining. (SPSS, 2003)

Text Mining and XML: Problems and Opportunities

The software vulnerability data used in this research is distributed in an XML format. However, the XML format presents two distinct challenges for traditional text mining software. First, XML data may contain a mixture of categorical, numeric and free-form text and numeric elements. Secondly, the non-symmetrical, tree-like structures permitted by XML must be effectively processed by text-mining software. Edmonds (2002) shows that when XML data is "tablularized", useful knowledge is potentially lost, and that the presence or absence of data in a tree is something to be mined. Text mining tree-based structures requires new analytical methods.

RESEARCH OBJECTIVES

This research has two primary objectives. First**,** this research will propose a method for text-mining XML data that describes software vulnerabilities. Second, it will present preliminary results that show the practicality of the developed method. The overall purpose is to develop a method that will permit the scientific examination of software vulnerability reports for both open-source and proprietary software to determine significant trends and associations. Data was collected in an XML format from the SecurityTracker vulnerability database across a period of four years (2000-2003). Separate reported vulnerabilities into "Open Source Vulnerabilities (i.e. Linux, Apache, Tomcat, etc.", "Proprietary System Vulnerabilities (i.e. MS Windows, Oracle RDBMS, etc.), and Hybrid Vulnerabilities (i.e. Netscape). Text transformation software was developed to convert the XML structures into a flat file format so that traditional text-mining software analysis could be performed. More specifically the text-mining tools (text-analysis and link charts) in PolyAnalyst 4.6 (from Megaputer Intelligence) were used. The appropriateness of this approach is discussed for finding trends and significant differences between different groups of software vulnerabilities.

Method for Mining XML Data

The following steps explain how PolyAnalyst 4.6 was used as a research tool to analyze XML-based software vulnerability data. Our original data comes from *Securitytracker.com*. The vulnerability data is listed by date in which every record includes several attributes including *cause, impact name, impact text, target software name, underlying operating system name, vendor name, source message,* and *description*. As a preliminary test, we analyzed more 1,000 vulnerability reports

between January 2003 to May 2003. In this case, we are especially interested in using PolyAnalyst 4.6 to analyze the links between the attributes of vulnerability such as cause and impact, cause and solution, etc. So in this report we focus on how to organize the data for link chart and link analysis, how to use link chart and link analysis function and how to make the chart and analysis report available for output. The whole process can be separated into the following steps:

Step 1 Data Conversion: XML to Flat File

PolyAnalyst 4.6 can load several different kinds of flat files like text files, Excel files and SAS data files. However, it does not directly support XML file structures. A software program, written in Visual Basic, was developed to convert xml files, and their corresponding (one-to-many relationships) into CSV files. Attributes with multiple categorical data values were transformed into single rows with multiple values placed as text in a single attribute.

Step 2 Data Importation and Segmentation

In PolyAnalyst 4.6, we set up a new project and loaded the CSV data. We could not load all the five CSV files in one time and had to use "Append" function to add the data. An important thing is that the same fields should be selected or the data could not be added because the application would suppose that they were not the same form of data and block the conversion process. Vulnerability records were separated into one of three categories: "Open Source Vulnerabilities" (i.e. Linux, Apache, Tomcat, etc., "Proprietary System Vulnerabilities" (i.e. MS Windows, Oracle RDBMS, etc.), or "Hybrid Vulnerabilities" (i.e. Netscape). This was accomplished in part by analyzing the license types for each software product through the web site *Freshmeat.org*.

Step 3 Logic Design and Text Analysis

Next it is important to determine the vulnerability attributes that are to be studied. First we selected two vulnerability attributes: cause and impact. Cause is defined by securitytracker.com to be "the primary cause of the vulnerability (such as authentication errors) is listed to enable you to quickly see what kind of error created the vulnerability." Impact is defined as "recent vulnerabilities resulting in a common type of impact (e.g., denial of service) are listed by those types." *Securitytracker.com* lists 10 causes (access control error, authentication error, boundary error, configuration error, exception handling error, input validation error, randomization error, resource error, state error and not specified) and 14 impacts (denial of service via local system, denial of service via network, disclosure of authentication information, disclosure of system information, disclosure of user information, execution of arbitrary code via local system, execution of arbitrary code via network, modification of authentication information, modification of system information, modification of user information, root access via local system, root access via network, user access via local system and user access via network).

We selected the "World" dataset and launched the Text Analysis function. In the Text Analysis window, we checked "find collocations", "include verbs", "remove identical leaves", "discard equivalent rules" and "include word generalizations" and uncheck all the other items. We set the "Thematic Context" as "Technology", "Maximum generalization level" as 0.5, "Min support" as 1.564% (default) and "Max support" as 50% (default). Then we right clicked the dataset field of "causename". In the process name part, we input "cause" as we like. Under the folder of "report" in the toggle workspace window, the process of "cause" ran several minutes and generated a report. The report showed us 10 categories of cause and count and percentage of record for each category. Under the folder of "rules", the rules of the 10 categories of cause were automatically generated with the process of the report. These 10 categories are almost the same as the 10 categories we found in securitytracker.com. The only difference is the category of "not specified" was not there and the category of "proof; validation" appeared. We deleted the category of "proof; validation" and tried to generate a new rule for "not specified". We checked all the 9 categories left. Then we right clicked the folder of rule to launch the create new rule function. In the new rule window, we select "not" as the logical word and use "or" to link the 9 categories and we named this rule as "cause_not specified". To check whether we was right, we applied the rule to the "World" dataset. Then under the dataset folder, we selected the "World" dataset and split the dataset to equal intervals by the rule of "cause_not specified". Two datasets were generated. One was "Cause_not specified_0", which contained all the records with cause other than "not specified". The other dataset was "Cause_not specified_1", which contained all the records with cause of "not specified". We were satisfied with the result and applied all the other 9 categories to the "World" dataset for link chart and link analysis.

Next we performed a Text Analysis for impact. It was more difficult than cause because it was hard to get the categories as we wanted. The procedures were almost the same as what we did to cause. In the Text Analysis window, we made some small changes. We unchecked "remove identical leaves" and checked "Consolidate text cells". We ran the report of "impacts" and got part of the differentiated categories as we wanted. To perform link chart and link analysis later, we had to

come up with the categories we wanted. If we knew SRL language, we could have used it to create new rules. Unfortunately we did not have the knowledge. So we did text analysis on some general categories. By the way, as we had to perform text analysis on the general categories, we did text analysis by having selected different setting to realize it. For example, we apply the rule of "authentication" to the "World" dataset and split "World" by the rule. Then we did text analysis on the dataset of "authentication_1" and we got the rule of "modification of authentication". In such kind of way, we got all the rules for impact as we wanted. And we applied all these rules to the "World" dataset". We also applied all the rules of general categories to the "World" dataset because later we would do link chart and link analysis both between cause and differentiated categories of impact and between cause an general categories of impact.

Step 4 Link Chart

A Link Chart was created by selecting the items we were interested in. We adjusted the correlation to change the chart. If we increased the correlation, fewer links were presented and it tended to highlight those most important links. If we decreased the correlation, we saw more links from the chart. The degree of correlation was determined by our requirements. We launched "create new link chart" from the menu of "create object" and named the new link chart as "cause_impact_detail". Then we selected the 10 causes as antecedent attributes and the 14 differentiated categories of impacts as consequent attributes by right clicking them. After we clicked ok, a new folder named "illustration and graphs" appeared in the toggle workspace window and the chart of "cause_impact_detail" showed up. In one side of the chart sat all the causes and in the other side sat all the differentiated categories of impact. There were bunches of lines between all the attributes. Blue line represents negative relations while red one represents positive relationship. If we click the line, the data set matching this link will jump out and stay at the upper left corner of the screen. And there will be a headline showing us the detail of the correlation and p-value.

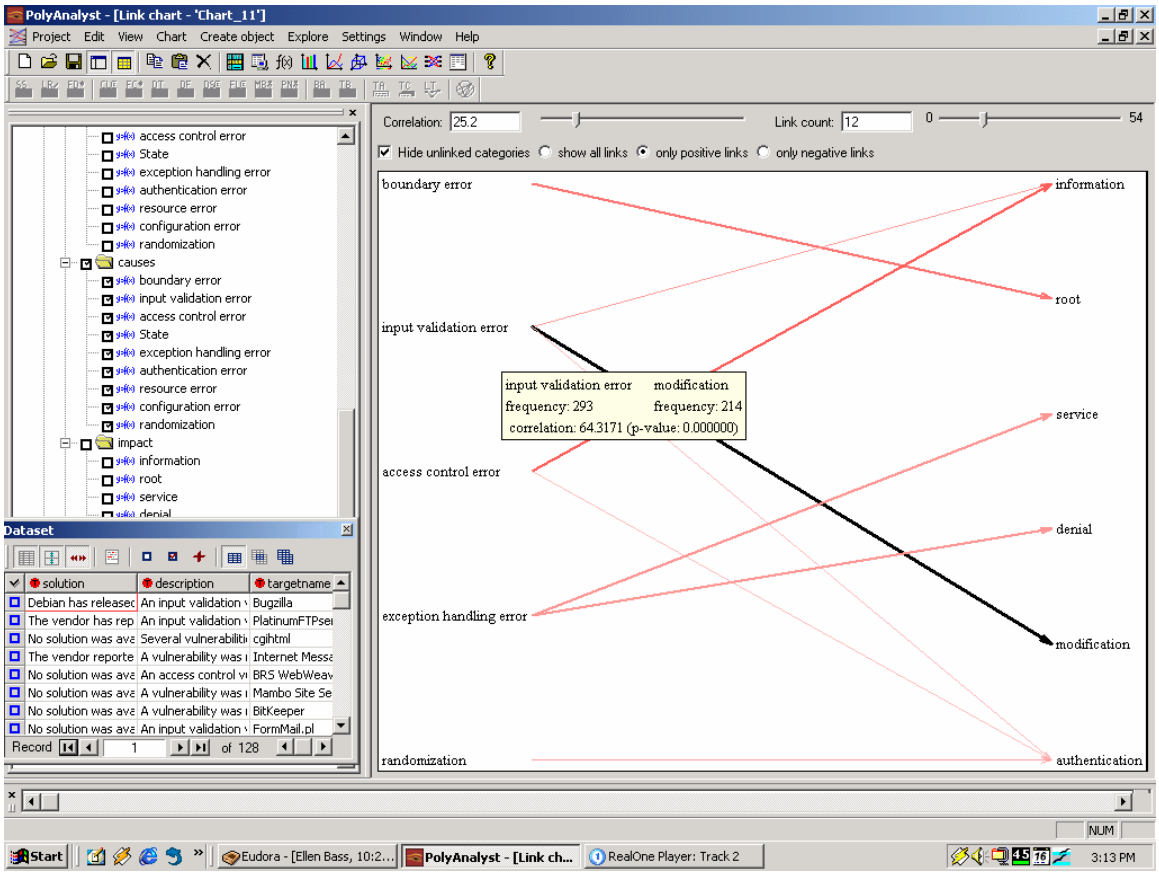

Figure 2: Link Chart from Cause to Impact showing only positive correlations and with a correlation greater than 25.0.

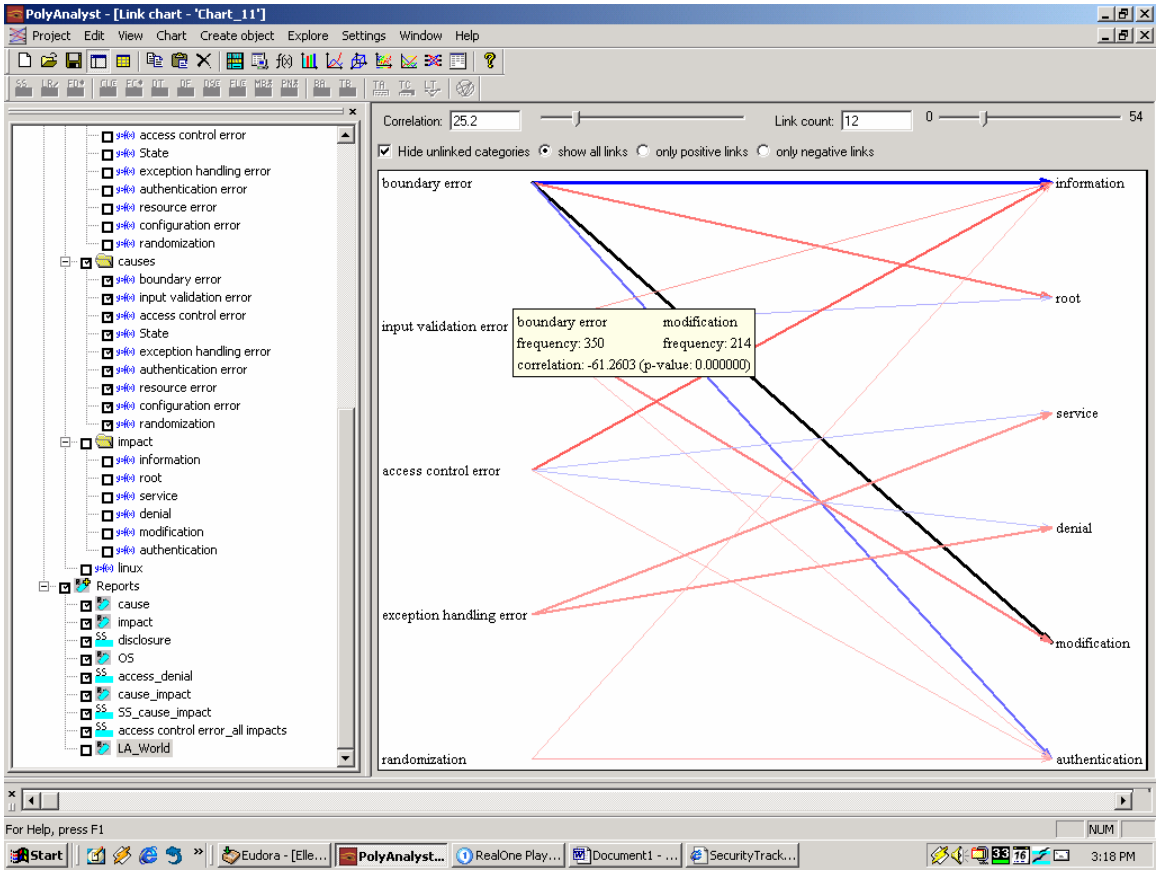

Figure 3: Link Chart from Cause to Impact showing positive and negative correlations with a correlation greater than 25

Step 5 Link Analysis

In order to get better vision effect and analysis, we used the link analysis function in PolyAnalyst 4.6. It provides more formats and an intelligent wizard to facilitate analysis. Everything is done with a slight click. The features of Link Analysis and Link Chart are almost the same. Link Analysis just has more powerful graph functions than Link Chart. The graph is not generated in "illustration and graphs" section but "report" section. There are two parts for each report. "Text" part lists all the names of the attributes and "Correlation" part shows the graph.

CONCLUSION

The proposed method presented in this paper is used to successfully converts XML vulnerability data into a flat file format, segments data into groups associated with open source, proprietary and hybrid software vulnerabilities, permits vulnerability attributes to be selected for analysis, and allows link charts and link analysis to be conducted for trend and association analysis. It appears that PolyAnalyst 4.6 is a good tool to analyze semi-structured web resources. It may not be an ideal tool to find the Boolean and relations between words and sentences, but it does allow the user to analyze data according to well-designed logic and summarized statistics.

REFERENCES

1. Arbaugh, W., Fithen, W. and McHugh J. (2000) Windows of Vulnerability: A Case Study Analysis. *IEEE Computer*, December, 52-59.

2. Browne, H., Arbaugh, W., McHugh, J. and Fithen, W. (2000) A Trend Analysis of Exploitations. Department of Defense Technical paper CS-TR-4200, UMIACS-TR-2000-76, 1-22.

3. CERT/CC, Carnegie-Melon University (2002) Vulnerability Note Field Descriptions. Available at http://www.kb.cert.org/vuls/html/fieldhelp#metric.

4. Edmonds, A. (2002) On data mining tree structured data in XML, Recent Advances in Soft Computing, Scientio Incorporated, Nottingham, UK.

5. Fisher, D. (2002) Open Source: A False Sense of Security? *eWeek, **19**(39) 20-22.*

6. Reavis, J. (2000) Linux vs. Microsoft: Who Solves Security Issues Faster? *SecurityPortal.com* cover story, available at: http://web.archive.org/web/20010818152843/ securityportal.com/cover/coverstory20000117.html.

7. SPSS (2003) Combine text mining with data mining to turn your information into business intelligence, LexiQuest Mine, http://www.spss.com/spssbi/lexiclem.

8. Weber, R. (1990) Basic Content Analysis, Sage Publications, Newbury Park, CA.

9. Witten, B., Landwher, C. and Caloyannides M. (2001) Does Open Source Improve System Security?. *IEEE Software*, September/October 2001, 57-61.