2001

The development of a database taxonomy of vulnerabilities to support the study of denial of service attacks

Thomas Winfred Richardson
Iowa State University

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

The development of a database taxonomy of vulnerabilities to support the study of denial of service attacks

by

Thomas Winfred Richardson

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Engineering

Major Professor: James. A. Davis

Iowa State University

Ames, Iowa

2001

Graduate College

Iowa State University

This is to certify that the doctoral dissertation of

Thomas Winfred Richardson

has met the dissertation requirements of Iowa State University

Signature was redacted for privacy.

Major Professor

Signature was redacted for privacy.

For the Major Program

Signature was redacted for privacy.

For the Graduate College

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I gratefully acknowledge the assistance given to me by the professors, staff, and other students at Iowa State University over the past few years; especially my major professor, Dr. Jim Davis. It has been great knowing and working with you all while doing this research.

I am very thankful for the opportunity that Rockwell-Collins has given me to do graduate work here at Iowa State, starting with my own Information Technology group and Larry Bricker (a great boss), and extending throughout the entire organization. Whether I was test-flying cockpit displays in Kansas, taking corporate jets into high-speed descents through the Rocky Mountains, or troubleshooting aircraft systems in central China, I've always felt that I have had great support from the wonderful people at Collins Avionics. The headquarters may be in Iowa but, to my friends and colleagues around the world, especially my old friend Martin Lin in the Rockwell-Collins Beijing office, thanks to you all.

Last but certainly not the least, I would like to thank my family for their patience, support, and understanding over the past few years while I've tried to perform two full-time jobs. To Dana, Milana & Eric, you are the best support team ever.

ABSTRACT

As computer networks continue to proliferate, the world's dependence on a secure communication infrastructure is of prime importance. Disruption of service through Denial of Service (DoS) attacks can result in great financial loss for Internet-based companies and major inconveniences for users of Internet services. The purpose of this two-year study was to study and understand network denial of service attacks so that methods may be developed to detect and prevent them.

Initially, the researcher constructed a database of system and network exploits that revealed the underlying vulnerabilities in the software or protocols they attack. The database was populated with exploits posted at popular reporting sites such as Rootshell, Bugtraq, Security Focus. To encourage the use of a common vulnerability taxonomy and to facilitate sharing of data, parts of the classification scheme proposed by Krsul (1998) in his research were included, as well as developing a taxonomy tree based on the current research.

Sifting through the reports and categorizing the attacks has been a challenging experience; and creating categories that are unambiguous, repeatable, and exhaustive has proven to be a difficult task. The results were two to three methods of classification that are useful for developing categories of vulnerabilities.

The next phase of the project was to look for any clustering of attacks based on these vulnerability categories, and to determine if effective countermeasures can be deployed against them. Although past history is no guarantee of future exploit activity, it is hoped that the countermeasures proposed based on these 630 exploits will remain valid for future DoS attacks. Toward this goal, the research made use of data mining software packages to plot the various categories of attacks so that the interrelationships could be more easily discovered and studied. A sampling of the database plots, an interpretation of the plotted data, and the countermeasures proposed for the vulnerability categories developed as part of the database creation are presented in this research.

1. PROBLEM STATEMENT, HYPOTHESIS, AND APPROACH

1.1. Problem Statement

Despite the efforts of many researchers, there lacks a universally accepted method of categorizing attacks against networked computers. In the absence of a framework, software vendors address network attacks on a case-by-case basis and in a reactive fashion. There is also confusion and a lack of standardization in the terminology utilized; the terms exploit and vulnerability are often used incorrectly and interchangeably. For the purposes of clarity, vulnerability is defined as being a weakness in a computer or networking system, and an exploit is the intentional act of compromising said system. The vulnerability is the path by which the compromise is achieved, but the act is carried out or performed to exploit.

Application and operating system vulnerabilities are discovered in networked computers and in network components on a daily basis. A large percentage of these attacks are posted on public Websites; and, while the sharing of vulnerability information is generally laudable for system administrators, it could lead to abuse by malicious elements of the computer user population. It extremely difficult to disseminate vulnerability information to systems administrators, who are the people who need to know this information, without sharing the same data to those who would use it in a malicious manner.

At the outset of this research effort, two primary goals were established. The first was to develop a complete and robust taxonomy for the classification of these vulnerabilities. The second was to use this taxonomy to derive a set of countermeasures that could be deployed against selected classes of vulnerabilities. If a database taxonomy that identifies classes of vulnerabilities could be created and effective countermeasures suggested, it might bring order to the current chaotic situation. Vendors and implementers could then focus on deploying countermeasures against entire classes of vulnerabilities, rather than merely patching each new vulnerability as it is discovered and disseminated.

The research was conducted based on two hypotheses:

1. A useful taxonomy could be built that organizes Web-based vulnerabilities into an objective, deterministic, and repeatable structure; and

2. The taxonomy will lead to classes or sets of countermeasures that could be deployed to address these vulnerabilities.

1.2. Rationale and Method

In the initial survey of existing literature, several schemes are proposed for classifying computer vulnerabilities. However, these existing database taxonomies fully defined the characteristics of remotely exploitable vulnerabilities, it was determined that a more complete taxonomy would be necessary. Previous efforts at classification usually did not discuss fully discuss all aspects of remotely exploitable and brute force attacks. Some taxonomies focus on locally exploitable vulnerabilities while ignoring remotely exploitable attacks, and most do not mention brute force attacks at. Brute force attacks are those which consume inordinate amounts of computer or network resources by either flooding a network with useless packets or by requesting large quantities of computer processing time. Such attacks usually do not violate any specifications nor are they reliant on coding errors; they merely consume large quantities of resources. These types of attacks tend to be simple to launch, yet they can render a target network of systems useless due to the quantity of data. A simple analogy might be the situation where a company's telephone number is erroneously posted in a newspaper advertising free pizza. With hundreds of people calling the business in error, very few of the regular customers or colleagues can get through to conduct business. The phone and phone system are working fine, but legitimate customers have little change of getting through due to the barrage of useless traffic.

This researcher not only expected to more fully describe the characteristics of remotely exploitable vulnerabilities and brute force attacks as a primary goal, but also create a taxonomy that has multiple levels of classification or definition. It was envisioned that the structure would include multiple levels of abstraction, providing ever-increasing levels of detail in the lower and more detailed classification layers. These layers of abstraction are referred to as the flowdown structure (i.e., leads to a more definitive or detailed level).

Since the database would also contain important metrics such as the type of operating system (OS), the date of the vulnerability posting, and the type of service exploited, it is hoped that important statistical results could be gathered from the data collected. Parameters

such as OS type, service exploited, and date could be analyzed for trend information, as well. These would be two secondary goals of the research.

As mentioned previously, the second primary goal is that the taxonomy will provide insight into creating sets of countermeasures that can be deployed for given categories of vulnerabilities. Even if the matching of vulnerability to a countermeasure were indefinite or even ambiguous, it would be better than no classification at all. Such a result has the potential for simplifying the lives of computer security professionals worldwide as they seek to protect imperfect systems in a threat environment that seems to grow more hostile every day.

Previous work in this area was reviewed in anticipation of incorporating schema structures from these prior studies. Then, a new taxonomy was built to achieve the two primary goals.

1.3. Criteria for Successful Conclusion

It is a simple matter to state the success criteria of these two primary goals: 1) classify the exploits encountered in a consistent manner; and 1) suggest classes of countermeasures for every major class of vulnerability defined.

Important considerations are the consistency and completeness of this categorization scheme and the subsequent countermeasure suggestions. The goal for the categorization was to place each exploit into the taxonomy in a reasonable category that is consistent with other entries. Since the categorization effort would span many months, it would need to be invariable over time and independent of the researcher performing the task. In other words, an exploit placed into the database taxonomy late in the research period would be grouped with similar exploits that had been determined early in the research effort and across the span of the research period, regardless of the person entering the data.

The completeness of the categorization effort would need to be to the level that would support the development of effective countermeasure categories, so the level of detail could remain flexible as long as it is consistent. As for the secondary goals of gathering trend information and publishing database statistics, these would depend entirely on the contents of the database. Any existing trend indications would become apparent as the database becomes

populated, and the composition of the database could easily be calculated using simple statistical analysis.

1.4. Dissertation Organization

The remainder of this dissertation is organized in the following manner. Chapter 2 expands on the previous research in this area and establishes a taxonomy for vulnerabilities that meets the established goals. Chapter 3 details the countermeasures that are effective against attacks seeking to exploit these vulnerabilities, and Chapter 4 presents the conclusions and establishes the relationship between classes of countermeasures and the underlying vulnerabilities. Finally, Chapter 5 presents the recommendations for future studies.

CHAPTER 2. A TAXONOMY FOR VULNERABILITIES

2.1. A Survey of Previous Research

Shortly after the commencement of this study, a literature review of related research in the area of vulnerability database taxonomies was conducted. The relevant peer-reviewed papers on the topic are discussed as follows.

Landwehr et al. (1994) [1] wrote one of the first papers to specifically discuss Denial of Service (DoS) attacks, and were also among the first to propose a taxonomy for vulnerability databases. Their work focuses on operating system faults and provided insight into potential classification schemes, with categorizations based on time of occurrence, genesis, and location. There is an attempt to provide ever-increasing levels of detailed categorization, but with only two to three levels of detail. Although the categories are somewhat ambiguous, a major outcome of the research is an attempt at devising categories for security vulnerabilities, and the realization of the difficulty in partitioning vulnerabilities into mutually exclusive categories in a consistent manner.

Bishop (1995) [2] built on the taxonomies proposed by Landwehr et al. [1], further describing techniques that are useful for finding vulnerabilities, describing the vulnerabilities in a form useful to Intrusion Detection (IDS) mechanisms, and presenting techniques to inhibit or eliminate exploitation of these vulnerabilities. Bishop's paper provides a summary of previous work, which leads to a new taxonomy consisting of six axes, or categories. These categories include: 1) the nature of the flaw, 2) the time of introduction, 3) the exploitation domain of the vulnerability, 4) the effected domain, 5) the minimum number of components needed to exploit the vulnerability, and 6) the source of the identification of the vulnerability. Other points include the assertion that every successful attack exploits a vulnerability, and that the detection and prevention work in this area is at an embryonic stage. Another major point presented by Bishop [2] is that the testing and abstraction techniques are crucial to effective detection and prevention of the exploits. The techniques presented for categorizing, plotting, and analyzing countermeasures are a step in that direction.

Similar to Bishop [2], Aslam et al. (1996) [3] proposed to understand security faults by categorizing them. Aslam et al. suggested two major categories, termed Coding faults and

Emergent faults. Emergent faults are defined as being faults that are a result of the improper installation of the software, where the software performs exactly according to the specification but still causes a fault. In contrast, Coding faults are composed of faults introduced during software development. Their research shows that it is still exceedingly difficult to develop categories that are unambiguous and consistent.

Other papers by Bishop and Heberlein (1996) [4], Bishop (1996) [5], and Bishop and Bailey (1996) [6] expand on these themes. Bishop [6] proposed a unique taxonomy concept similar to that used in the biological world, namely using the categories of Kingdom, Phylum, Class, Order, Family, and Genus. The main goal is to have a categorization scheme that is unique from other security flaws, using a hierarchical taxonomy similar to that used in the biological world, which culminates in the final two categories where the vulnerability is the Genus and the attack is the Species. Bishop contends that such a categorization scheme should allow groupings of vulnerabilities, which, in turn, suggests where efforts should be deployed to reduce or eliminate flaws.

Krsul (1997; 1998) [7, 8] asserts that vulnerability analysis is not a well-understood process, but further states that there should be a way to collect vulnerability information in a reasonable manner, classify this information in an effective manner, store it in a consistent format, and process it with analysis tools that lead to preventative measures. Krsul [7] provides a definition of a vulnerability database, an attack, and a vulnerable state, and even the beginning of a taxonomic scheme involving coding faults and emergent faults, as detailed in Aslam et al. [3] in 1995. Most of these recent papers on computer vulnerabilities reference two earlier works from the 1970s by Abbot et al. (1976) [9], and Bisbey and Hollingsworth (1978) [10], but very little of this work is directly relevant to the current research. Another paper by Krsul et al. (1998) [11] offers further insight into the difficulty of vulnerability analysis. In addition, the proceedings of two workshops at Purdue, in 1997 [12] and 1999 [13], the second of which was attended by ISU researchers, did little to help establish a universally accepted taxonomy.

The two general categories proposed in Aslam et al. (1996) [3] were adopted for the taxonomy developed in this research. Coding Faults roughly translate into the Implementation Faults category, whereas Emergent Faults translate into the Specification

Weaknesses category. Neither category covers the case of an attacker consuming inordinate amounts of computer or network resources in a brute-force fashion. This element is also missing in the other taxonomies.

As a result, in addition to these two major categories published in previous work, the ISU taxonomy also contains a category, called Brute Force, in recognition of the truly unique methods used in many DoS attacks that saturate a network or a host computer's ability to respond to legitimate requests. In most brute force attacks, the attacker hammers away at a service or protocol hoping to deny service to legitimate users either by keeping the system so busy that it has no time to service other users or by causing the operating system to crash. In either case, the legitimate user is left without access to computer or network resources.

A common theme observed during the early research phase is that, although several taxonomies were proposed and presented, none directly addressed the problem of describing all the characteristics of remotely exploitable Denial of Service attacks. In particular, the brute-force aspect of computer attacks was missing, and most taxonomies had only one level of classification and no further refinement or definition. In addition, there was an urgent need to create a taxonomy that could lend insight into potential countermeasures.

Based on the shortcomings of the previous work, the current research focused on developing a database that contains only remotely exploitable vulnerabilities, an easy-to-use taxonomy with increasing levels in detail in a flowdown structure, and a taxonomy that provides some ideas and insight into Denial of Service (DoS) countermeasures. This work can develop a structure, sort attacks and exploits into such a classification scheme and then, as a separate goal, derive classes of countermeasures for these groups of categories.

2.2. History and Evolution of the Database Taxonomy

The focus of the current research was on remotely exploitable attacks that are meant to deny service, and to sort these vulnerabilities into categories of ever-increasing refinement. The intent was to restrict the database population to attacks that were remotely exploitable and of DoS in nature. However, after examining the early database entries, it was found that there were very few attacks carried out solely for the purpose of DoS. Thus, if this

restriction were to remain in force, the database would consist of only 150-170 attacks at the most, making it only about 25% of its current size.

Consequently, the definition of DoS attacks was widened to include vulnerabilities that provide root access to the system (i.e., allowing the perpetrator to bypass system security), using the philosophy that root access would deny legitimate users access to their own systems, which by most definitions is considered a DoS attack. In addition, database categories were added that resulted in unauthorized access and buffer overflow attacks to be quantified. This helped to more fully populate the database, as these two attack categories seem to constitute the majority of remotely generated attacks posted to vulnerability Websites. To meet the research requirements, the database taxonomy acquired three distinct subsections: 1) the original flowdown structure; 2) the categories previously defined by Krsul [8]; and 3) a category termed Mechanisms. The subsections are described as follows.

2.2.1. Flowdown Structure

In subdividing the database structure, three general categorizations were proposed, two of which were suggested by Aslam et al. (1996) [3], Bishop and Bailey (1996) [6], and Krsul (1998) [8], as discussed previously, and one category that was added by the present researcher to accommodate the nature of Brute Force attacks. Furthermore, unlike most previous taxonomies, subcategories for each of these three main categories were added to more fully refine the categorization. The three major categories are: 1) Specification Weakness; 2) Implementation Weakness; and 3) Brute Force. These subcategories experienced a great deal of metamorphism and evolution in the early days of the research, but were eventually refined sufficiently to allow subdivision of exploits quickly and easily. This meant modifying some subcategories and eliminating others, while constantly seeking the right level of detail. Too much detail leads to sparsely populated categories whereas too few categories lead to overpopulation of categories. Nevertheless, after this refinement period, ending in the summer of 1999, it was a relatively simple process to determine suitable bins with the right level of detail for each attack encountered. Researchers new to the project would probably need some amount of training because the categories tend to be ambiguous

to the uninitiated. After this initial training phase, researchers should have no problem determining consistent classifications.

2.2.2. Work of Krusl

The thesis written by Krsul (1997) [7] was particularly relevant to the ISU research effort. Although it was not specific to DoS attacks and lacked a concept of brute force attacks, it contained some interesting characteristics and useful sections. Krusl's major contributions were in the development of new taxonomic characters which included categorization parameters. Application of the subcategories were inconsistent at first, but became more consistent with the memory aid developed here at ISU that helped determine suitable categories for posted exploits. This memory aid was utilized by fabricating an English-language mnemonic, for example, in the sections titled: objects affected, effect on objects, method or mechanism used, and input type. At first glance, the last two items (method or mechanism used, and input type) seem to be redundant, but the categories listed under these two sections are different and, thus, not easily confused. When the suitable noun or verb from the category list is substituted in these four sections, the result is a short, English-language sentence that describes the basic components of the exploit, using these four action words. This was a great help to the schema development, and resulted in consistent categorizations of the exploits. These various action words are shown in Appendix A. Categories that were added in the current research can be easily distinguished from the previously developed categories by noting the ISSL (Information Systems Security Laboratory) prefix.

2.2.3. Mechanisms

From the original categorization scheme and this four-part sentence, yet a third categorization scheme was developed that contained from one to three words that describe the underlying vulnerability or weakness, called Mechanisms. This resulted in six major categories, the 6th having four underlying subcategories. The hypothesis, that it is possible to use this Mechanism scheme to develop successful countermeasure categories, turned out to be the case. The development of the mechanism category is more fully explained in

subsequent sections. Examples of the mechanism classification category are shown in Appendix B, with the entire vulnerability database evolution process shown in Appendix C.

In summary, presently there are three categorization schemas embedded within the current vulnerability database structure; 1) the original structure; three major categories further broken down into subcategories as a flow down, tree-like structure similar to the biological classification scheme; 2) the categories derived from the Krsul classification schema [7], with enhancements from the current research that form a simple sentence describing the vulnerability; and 3) the two or three word summary of six categories, called Mechanisms, which sought to answer the *How* piece of the puzzle.

2.3. Plotting the Data in Search of Groupings

Initially in this research, one of the essential assumptions was that data in an instantiated taxonomy could be visualized and plotted in a way that provides insight into countermeasures. It was theorized that a single countermeasure or group of countermeasures could be identified that would be effective against attacks whose data points are grouped or clustered in a vulnerability plot. The software program, called Mineset, was used to produce 3-D plots of database categories, looking for trends and groupings of the data. Later, individual screenshots were combined to create movies of these plots by creating sequences of plots and observing how the clusters grew and shrunk during the three-year period of the study, from March 1997 until February 2000.

The categories most frequently plotted included items such as: 1) the vulnerability category, using several of the three major categorization schemes; 2) the service being used at the time (i.e., FTP, HTTP, messaging, Telnet); 3) the date that the exploit was first posted; 4) the operating system (OS) being exploited which is usually shown as a color on our graph; and 5) whether or not the vulnerability was in the OS kernel, a System Program, a User Program, or a System Utility.

Although it is hard to see the three-dimensional plots rendered on a two-dimensional media such as a sheet of paper, several of these plots are included in this and subsequent sections. The value of the plot is derived mainly when it is on a computer screen, where the object can be rotated and examined from all angles, exploring the relationships between the

various clusters. By using Mineset, it was also possible to plot the data in such a manner as to view the temporal nature of the database and watch the clusters grow and shrink over time. This was done using the slider function, where the data plotted are only for that month, and then the plot slides into the next month's data as a seamless image. Despite this capability, no distinguishable patterns in the data were noticed that would lead to any firm conclusions concerning predictability of a certain style or type of attack.

Early in the graphing stage of this project, several parameters were examined in a two-dimensional manner prior to adding the complexities of three-dimensional graphs and plots. Using some of the more basic parameters stored in the database, simple plots, such as the one depicted in Figure 1, were created.

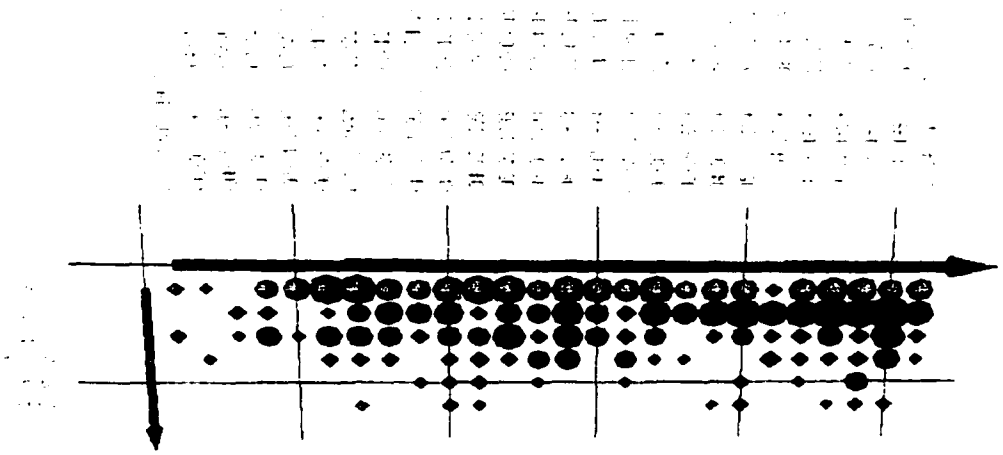


Figure 1. The operating system plotted as a function of time

The plot shown in Figure 1 represents the Operating System (OS) versus time, and shows the relative popularity of the various OS versions victimized during that time period; the larger the cluster, the more incidents of that type in the given time period. Most attacks are carried out against either a version of Microsoft Windows (e.g., W95, W98, or NT) or against one of the many different versions of Unix. There are about 10 different major versions of Unix software and countless updates of each version, but they can easily be

categorized under the generic category of Unix. For visual effect, color was also used to distinguish between the various OS versions. Clusters shown in yellow are the various versions of Unix, and clusters shown in green are the various versions of Windows. Yellow and green were not random selections, since an attack that affected both operating systems would then appear as blue. Finally, red was used to denote attacks against network devices such as firewalls, routers, and other appliances that had their own operating system. This color scheme is used in all of the other 2D and 3D plots, adding an additional dimension to the data already presented. Another example of a 2D plot is shown in Figure 2.

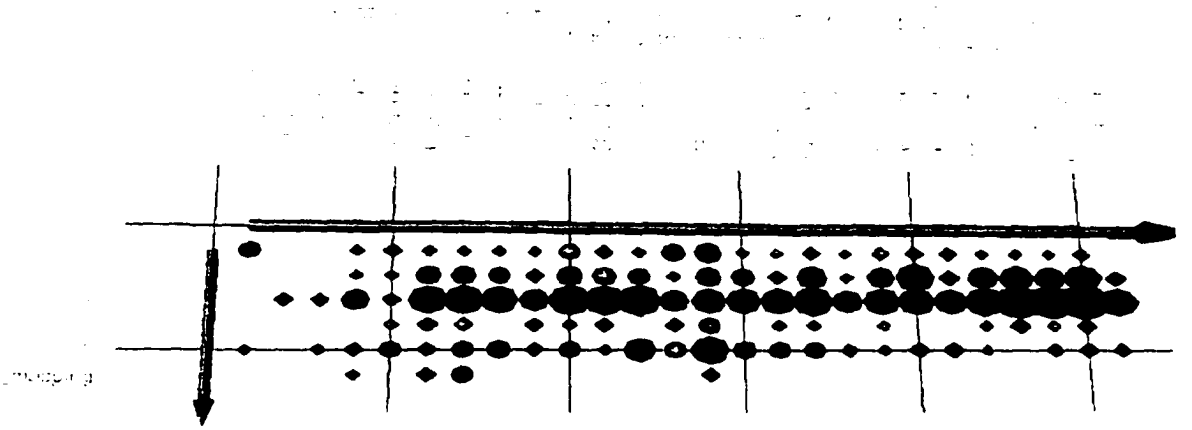


Figure 2. Object_Affected versus Time

Figure 2 shows how the parameter *object_affected* plots out against time, with the bulk of the attacks being against the software itself instead of tying up the network or consuming CPU cycles like many Brute Force attacks. These types of attacks are represented here in the CPU and Network categories. However, the majority of the attacks are not DoS in nature; rather they are attacks that seek privileged access to a system. These types of attacks are represented in the Files or Software categories.

Figure 3 shows how the various services rank in terms of attack targets, with the bulk being in the form of HTTP or messaging attacks, where *msg* equates to a messaging attack like email, IRC chat, etc. *Network utilities* is a popular category, but was also somewhat overused if the posting was not specific as the nature of the service being victimized. Figure 3

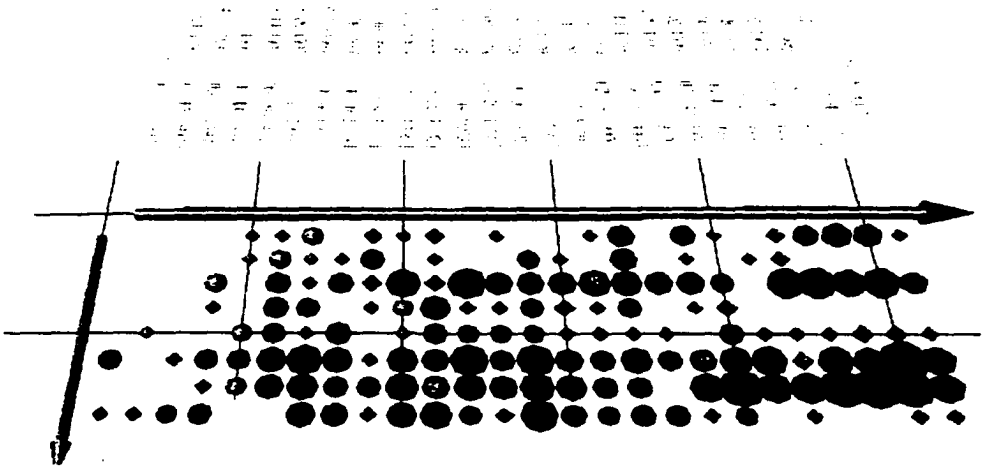


Figure 3. Service versus Date

shows a steady increase, over time, in the number of attacks, as witnessed by the size of the clusters in the most recent timeframes. The plotting results indicate that, over time, all forms of attacks seem to be increasing in frequency, and this matches the observations of many others in this field of research.

Most plots of the parameters of interest, when plotted against time on the X-axis, show an increase in frequency as time increases (moves to more recent months), as witnessed by the increase in size of the clusters of that parameter. The larger the cluster size, the greater the frequency of attacks of that variety occurring in that time period. Plotting these various categories led to insight into the *Mechanism* category. Plots of object_affected and effect_on_object categories, particularly when both are plotted in relation to time as the third parameter (see Figure 4), show clusters that align themselves along similar parameters, which is termed *stringing*.

Figures 3 and 4 provide examples of stringing, which indicates that many of these attacks are related. If object_affected was the *who* parameter, and effect_on_object was the *what* parameter, it seemed reasonable to try and answer the *how* question. A slightly different version of the same plot is shown in Appendix D. The results provided the foundation of the Mechanism category, which sought to answer the *how* question. The development of Mechanism category was uniquely a product of this research effort.

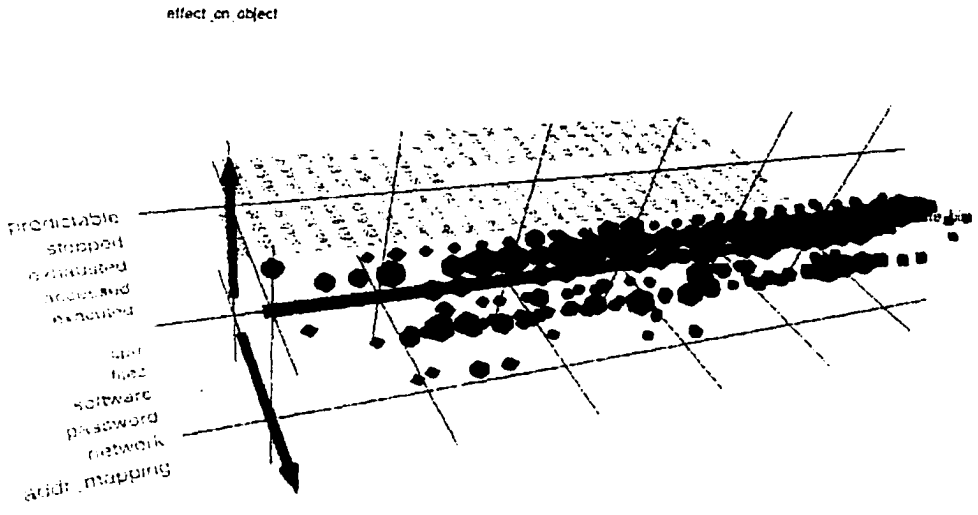


Figure 4. Object_Affected versus Effect_on_Object versus Date

In Figure 5, the Mechanism category is plotted against time. One can see from this that the categories of *Buffer Overflow* and *Poor Authentication* or *Access Control* show steady growth over the period, as to be expected from the database statistics. It should be noticed that no category has died out in recent months; they all showed intermittent activity over the research period. The mechanism category is discussed more completely in Section 2.8, and a more complex 3D plot is included in Section 2.9.

Part of the research effort included validating this taxonomy structure against other database structures, in terms of content, using the current groupings and categories to confirm that the taxonomy in the database is a true representation of the attacks that exist. The Mitre Corporation has posted their vulnerability database online at http://cve.mitre.org. A comparison of the two databases is given in the next section. Permission to examine the database maintained by CERT (Computer Emergency Response Team) was obtained, but the data were not usable to this research effort because they did not have a temporal aspect. This does not mean that future collaboration efforts would not yield highly useful data; thus, it might provide an opportunity for future research.

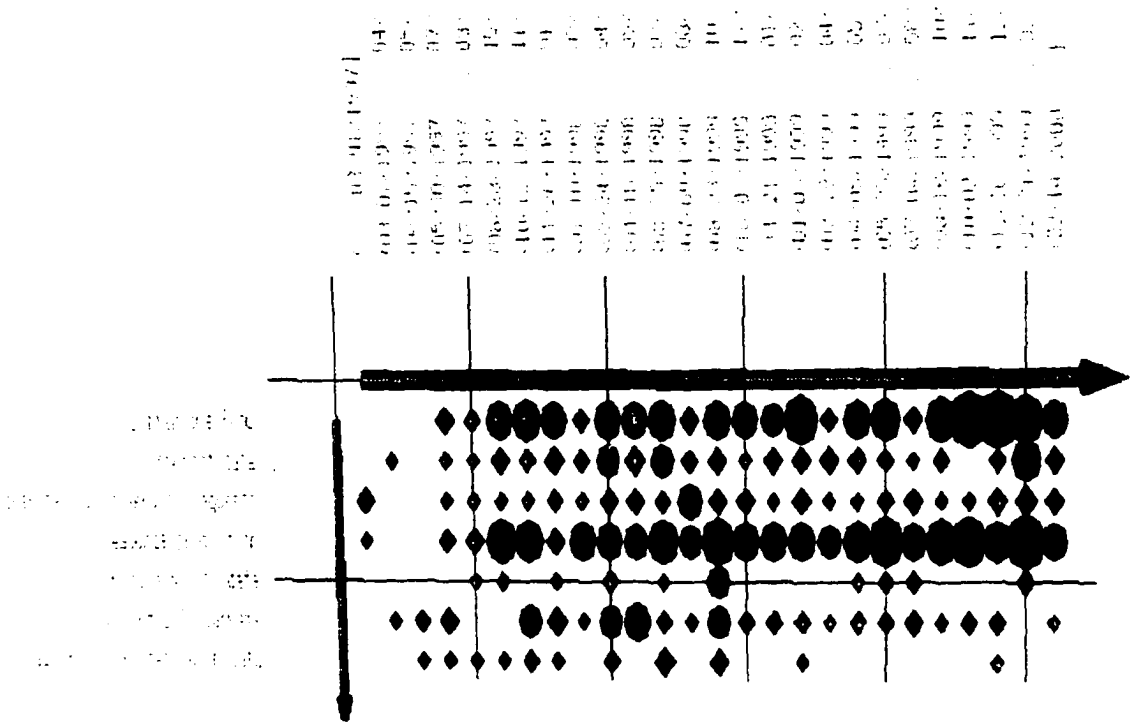


Figure 5. Mechanism versus Date

2.4. Comparing the ISU Taxonomy Structure with Other Available Databases

Temporarily leaving behind the structural considerations, it was important to determine that the contents of the ISU database were a representation of the attack population; therefore, other we other databases were actively sought for comparison purposes. Data from CERT (Computer Emergency Response Team) and an online database being compiled by the MITRE Corporation were examined for consistency with the research database.

The ISU database consists of over 630 entries gathered from computer security sites that span the dates of March 1997 through February 2000. The focus remained on remotely exploitable attacks that fit the expanded definition of a DoS attack, as detailed previously. The metrics for this database are given in Figure 6 and 7.

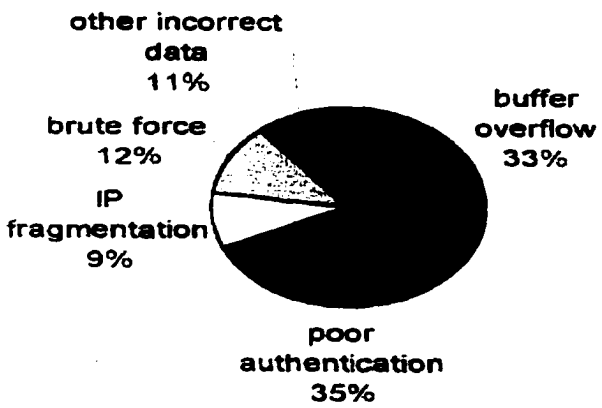


Figure 6. Classes of ISU database entries

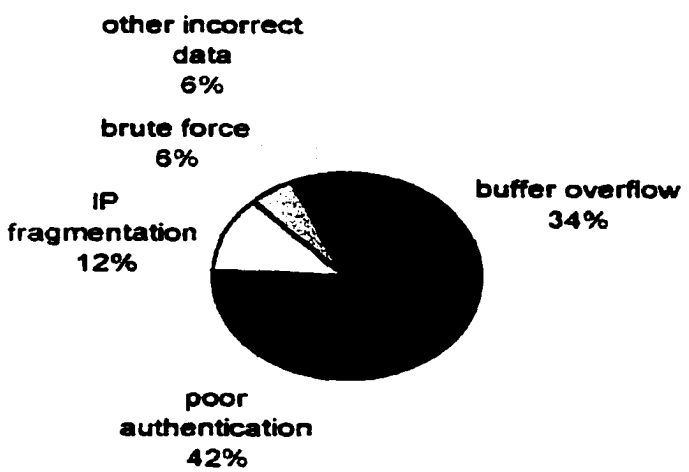


Figure 7. Classes of Mitre CVE database entries

Of the 630 entries in the ISU database, nearly one-third were buffer overflow attacks (201 out of 630), and slightly more than one-third of the entries were poor authentication or access check attacks (215 out of 630) (see Table 1). The remaining attacks were distributed among a variety of categories, including two Brute Force categories. This compares favorably with a survey of the contents of the Mitre Website database, a site that had about 382 entries in early June 2000. The CVE database is an attempt to develop a common vocabulary in the computer vulnerability field, and assigns numbers to commonly found vulnerabilities. Note that at Mitre Website, the CVE database uses different categories. The results were obtained by applying classification schema in this research to the attacks listed in the Mitre database, to allow comparison of content.

The contents of the two databases are quite similar, which provided confirmation that the ISU database contents mirror the attack or exploit population of the Mitre database, which is assumed to be representative of the real world.

Table 1. The ISU database compared to the Mitre CVE database

	Database			
	ISU ISSL (n=630)		Mitre CVE (n=382)	
Mechanism categories	Attacks	Percentage	Attacks	Percentage
Buffer overflow	201	32	128	33
Poor authentication	233	37	162	42
IP fragmentation	57	9	45	12
Brute force	73	12	23	6
Other incorrect data	66	10	24	6
Total	630	100	382	100

In the early days of database structure development and plotting, it was found that too many categories were almost as inconvenient as having too few. While some of the database categories gave a great deal of choice as to parameter categorization, it was determined that it was best to go with a category having only 5-7 choices. If the database continues to grow, it

is expected that these additional 5-7 categorization options will become useful in describing the vulnerabilities being posted.

As mentioned previously, permission to examine the contents of the CERT database was secured in the early Spring of 2000, but in the end, it was found the data were not appropriate for use in this research. One of the possibilities for a future study might be to categorize future databases using the taxonomy and database structure developed in this research. That study could be used to validate the relevance of the taxonomy in this study.

2.5. The Present Status of the Taxonomy Structure and Future Plans

Currently, the database contains 630 total exploits that span three years: March of 1997 through February of 2000. Thus far, the exploits have been easily placed into the defined categories, but this is no guarantee for the future. The real test of relevancy would be whether or not the taxonomy categories remain relevant for exploits posted in the future.

A limitation of the database is that it is comprised of singular entries of a new exploit. When an exploit is posted on one of the online sites, it ends up being just one entry into the database; there is no indication of how popular that form of attack is and how often it may be used in the subsequent days, weeks, and months after its posting. From this, it is not possible to obtain enough data to predict the direction of future attacks. In other words, there is no *temporal data* to use in this effort. It is believed that if we can obtain the desired temporal information, perhaps models can be built to show how the attack actively migrated, ebbed and flowed as the exploit was first found and published, spread throughout the underground, and then was subsequently replaced after a new technique was found or software patches posted for the exploited weakness.

2.6. Database Category Structure

The tree-like schema of vulnerability categorization is shown in Appendix E, with the various layers of subclassification described on the subsequent pages. The three main categories, from which all others descend, are Specification Weakness, Implementation Weakness, and Brute Force. The first two were derived from classifications proposed in earlier papers, while the third is original to this work. Under the Specification Weakness

category are the subcategories of *Incomplete Definition*, *Ambiguous Specification*, and *No Error Recovery Defined*. Incomplete definition is further broken down as shown in Appendix E. The Implementation Weakness category has the largest quantities of subcategories, with the flowdown structure shown. Subcategories allow for the improper handling of both correct and incorrect data, which cover most of the buffer overflow-style attacks. The Poor Authentication and Access Check covers the situations where it is thought that the software should do more to limit unauthorized access to sensitive data. This has proven to be a very popular category, and contains more than one-third of the exploit entries. In Appendix A, the various categories in the Krsul scheme are shown [8], along with the memory shorthand to assist in classification efforts. More details on these various categories are included in Section 2.7. The parameters used most often are shown in boldface type in Appendix A. Mineset has a utility that can perform statistical analysis on these database entries, and can easily show the category contents and the percentages thereof. This data follows in Figure 9-12.

The following three attacks are given as examples of how to use this taxonomic structure. 1) In an HTTP flooding attack, where an attacker repeatedly requests a page from a Webserver, the requests come so quickly that no other user can get access to the server's resources. An attack such as this would fall into the Brute Force category, and then flow down to the *Overwhelm with Service Requests* subcategory. 2) The *Smurf* attack, which sends a large quantity of data to a hapless victim, is also a Brute Force attack that falls into the *Overwhelm with too much Data* subcategory. 3) The *IP fragmentation* attacks that were so popular a few years ago are an example of an attack that fits into the Implementation Weakness category, and then flows down into the *Improper Error Recovery for Incorrect Data* and *IP Fragmentation* subcategories. With experience, database maintainers should be able to fit any posted exploit into this database taxonomy with a high degree of both accuracy and repeatability.

2.7. Database Statistics

Figure 8, 9, and 10 show some of the statistics gathered on the ISU database contents, with other parameters shown in Appendix D in the form of a 2D or 3D plot. Figure 8 indicates that Unix and Windows systems are victimized almost equally, however, this may be

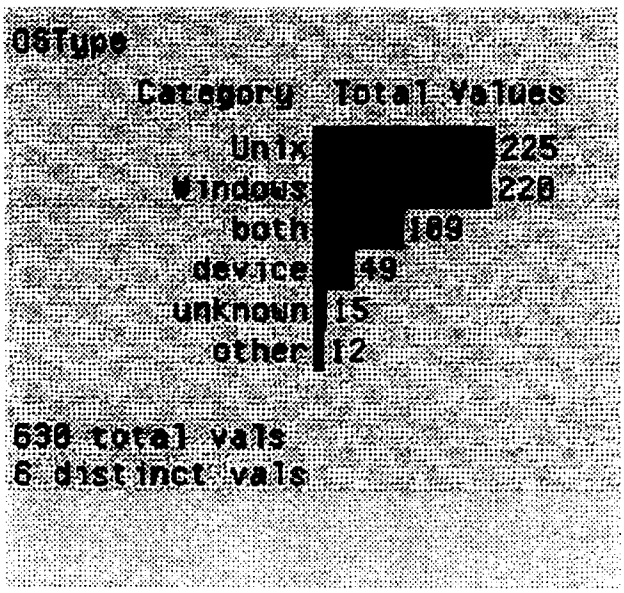


Figure 8. Victim OS

somewhat unfair due to the many different versions of Unix. Many attacks, particularly the Brute Force ones, victimize the network and not the computer's operating system. Most of these end up in the OS category called *Both*. Such attacks show up on the plots colored with blue, signifying a combination of green for Windows and yellow for Unix. Figure 9 shows the

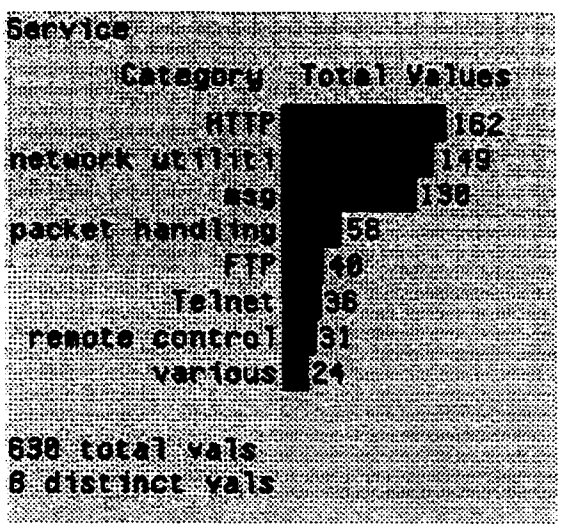


Figure 9. Service Exploited

services being exploited; one can see where HTTP and the messaging types of formats (msg) are some of the most often abused. Figure 9 clearly shows the dominance of the Web attacks on the HTTP protocol, accounting for 25% of the reported total, while the number of attacks in the non-specific *various* category was rather small.

Figure 10 shows the object being attacked or impacted by the particular exploit. Figure 10 shows that most attacks are against specific application software packages, and not merely network protocols. This indicates that it is important for software vendors to test their products against such possibilities.

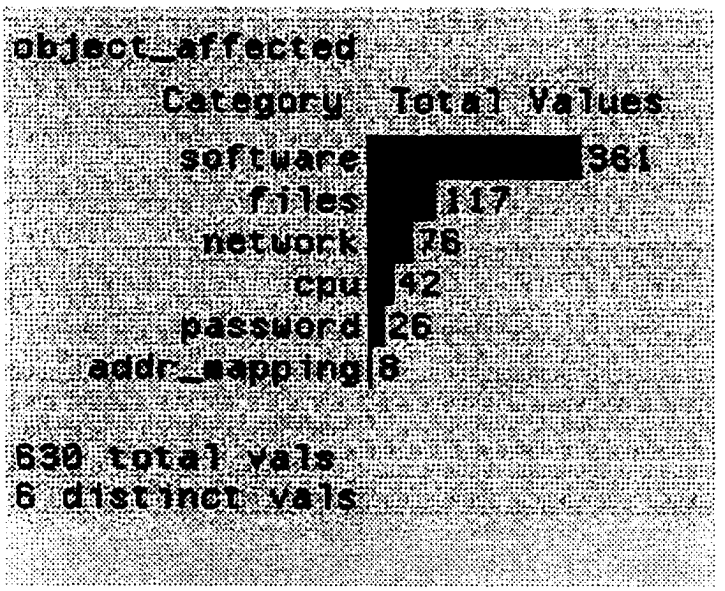


Figure 10. Object_Affected

2.8. Categorization Process

The categorization process underwent a great deal of modification during the first year of gathering exploits. These items, as they were gathered from various public Websites, were *binned* in several formats in order to capture the essence of the vulnerability. Of particular interest were how each of the attacks worked, the services being attacked, and what operating systems and programs the attacks were effective against.

As mentioned previously, the desire was to have at least one flowdown structure where the postings could be categorized in ever increasing degrees of detail, much like the biological categorization scheme. Most previous taxonomies usually included only one level of classification, or were ambiguous and non-repeatable. Care must be taken to control the granularity detail in the level of segmentation, so that the data is not spread too thin and meaningful trends go unnoticed. Categorization categories defined by Krsul (1997) [8] were used, along with enhancements from the current research that covered remotely exploitable and brute force types of vulnerabilities.

It was also noticed that the categories in Krsul [8] were not the optimum way to sort the data in a manner from which countermeasure categories could be defined. As a result, new *Mechanism* categories were developed in the current research. As shown in Figure 4, the stringing effect of the plot led to consistent WHO and WHAT categories, the only missing piece being the HOW part. The six Mechanism categories provided that missing piece and the correct granularity of classification for identification of countermeasure categories. As a contrast of the different formats, both of these categorization methods are shown in Appendix E. The flowdown structure is on the left, with solid connection lines between the various flowdown elements, and the Mechanism categories are shown immediately right of the center, aligned with their typical flowdown counterparts. Some of the categories used in Krull's [7] *Nature of Method* category are shown along the right-hand edge and are further detailed as part of Appendix E. This connection to the earlier Krsul research is provided as a convenient reference.

Figure 11 shows the relative composition of the database when sorted by the Mechanism category. It should be noted that these Mechanism categories closely resemble the major weaknesses most often identified with attacks that lead to a Denial of Service. Not only are the flooding or connection attempt attacks represented in the *overwhelm* subcategories, but also the additional types of attacks that can lead to a DoS. The include such vulnerabilities such as the Poor Resource Protection attacks referred to as Poor Authentication or Access Control attacks, the Buffer Overflow, and incorrect data/malformed IP packet attacks.

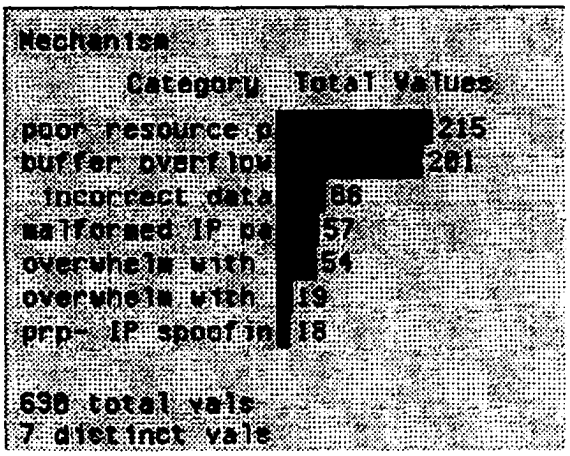


Figure 11. Composition of the Mechanism category

2.8.1. The Flowdown Structure

It should be noted that the exploit or vulnerability first has to be sorted into one of three categories (see Appendix E). A determination is made as to whether it is a Specification Weakness, a Brute Force attack, or some form of Implementation Weakness. As previously mentioned, the Specification Weakness category had its origin from previous works in the form of the Emergent fault, and the Implementation Weakness was called the Coding Fault in previous papers. The Brute Force category was added to cover the types of attacks that overwhelm a target with connection requests or with data packets, as well as the ones that reset a connection prematurely or which lead to a condition where no error recovery scheme is available.

2.8.2. Specification Weakness

The Specification Weakness category covers the case where the specification is either incomplete or ambiguous. For the purposes of this research, most of the attacks that relied on IP source address spoofing as an integral part of the attack were included in the *Incomplete Definition* subcategory. The logic applied was that if the current Internet IP specification, IPv4, had included some form of address verification or authentication, attacks such as the Smurf attack would not be possible. For an attack to be placed in this category, address spoofing must be one of the main reasons that the attack succeeds. Attacks such as DNS

Poisoning, Smurf, and some authentication weaknesses are placed in this category. There are few attacks currently in the Ambiguous Specification subcategory, however the advent of Ipv6 should help populate this section.

2.8.3. Brute Force

The Brute Force category covers the type of attacks that rely on massive amounts of data packets or connection requests to overwhelm a victim. IP source address spoofing is commonly used with this type of attack, but only to hide the true source of the packets and avoid the consequences of discovery rather than carry out the attack. The actual flooding works with or without address spoofing. This category also covers the types of attacks that are allowed to reset valid TCP connections, although one could argue that the real cause is a lack of authentication capability and, therefore, it is a specification weakness. For a variety of practical considerations, it was decided to leave the *reset* type of attacks under the Brute Force category. Some attacks, such as the attacks that seek to fill a logging space with error messages, are binned into the *Depository Filled* subcategory under *No Error Recovery Utilized*. Including attacks that fill up log space or consume other system resources is considered a Brute Force attack because such actions prevent normal processing from occurring.

2.8.4. Implementation Weakness

The Implementation Weakness category includes items such as poor authentication and buffer overflow style attacks, and it tends to be the largest category. Each of these types of attacks account for about one-third of the items in the database. This category also has the most complex flowdown structure, including subcategories that cover: 1) Improper Throttling of Data, which include Buffer Overflows; and 2) Improper Error Recovery for Incorrect data, or *wrong data*. The former refers to the quantity of data provided whereas the later refers to the type of data being provided. Examples of wrong data include characters that crash a program, because they were not expected as input, or cases where the fragmentation of an IP packet is done in such a way that the re-assembly process is impossible and the computer OS

or program crashes. These subcategories are missing from most prior taxonomies that were examined.

Also under this classification category are the Poor Authentication or Access Check weaknesses, as well as the rare cases of correct data being handled improperly. The subcategories deserve more description, which is provided in the examples that follow. Figure 12 shows the distribution of these three main categories in the flowdown structure:

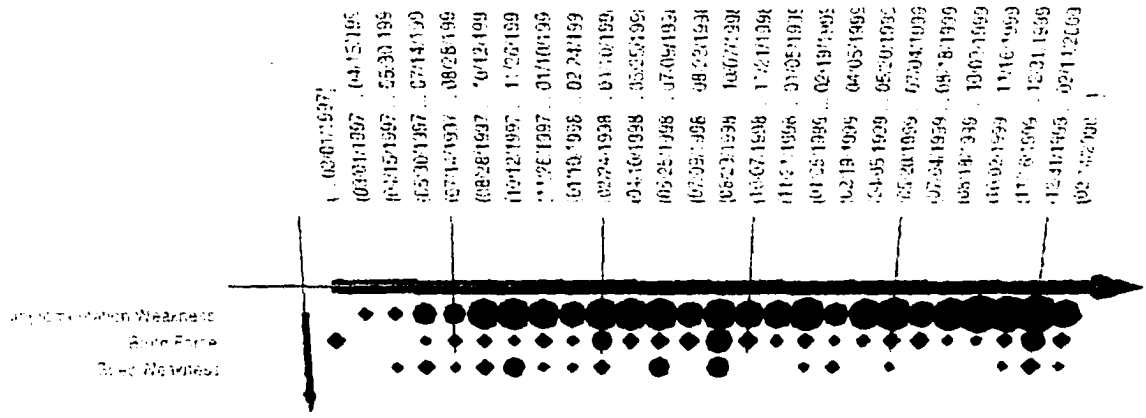


Figure 12. Vulnerability Type versus Date

The following paragraphs include sample classifications and the logic used to place a particular database item into this category, particularly for the most complex category, Implementation Weakness. The order of discussion is the same as the order shown in Appendix E.

2.8.5. Classification Examples

The attacks that rely on IP spoofing as an integral part of the attack itself, and not just as a form of disguise, are listed under the Specification Weakness category and Incomplete Definition subcategory. These include the Smurf attack and all Smurf variations, the DNS Spoofing attacks that replace URL-to-IP address mappings, and predictable TCP sequence numbers.

Attacks that exhaust resources with either packet flooding attempts, massive connection requests, reset connections, or filling up log space are referred to as Brute Force attacks. These attacks are most often thought of as DoS and are the hardest to defend against. Most of the new Distributed Denial of Service attacks (DDoS) are of this variety as well. The Synflood attack is an example of *Overwhelm a Target with too Many Service Requests*, while the various flooding attacks, such as *ICQflood*, are binned into the *Overwhelm a Target with Too Much Data* subcategory. Some attacks, such as *brkill* and *TCP_reset*, set up TCP connections and then hit the server with a RESET command, causing the computer to refuse connections for a set period of time (usually two seconds). One could argue that this is more of a Spec Weakness fault because the specification does not guard against it, but the intentional setup and reset cycle makes it a prime Brute Force candidate. In addition, the intentional filling up of log space with error messages, when correctly generated, is an example of the *No Error Recovery Utilized* subcategory.

The Implementation Weakness category is the most popular because it contains buffer overflow and poor authentication weaknesses. It is also the most complex. The Improper Throttling of Data subcategory includes attacks that rely on sending more data which is then expected by the destination computer. Such is the case with most buffer overflow attacks, whereas the Improper Error Recovery for Incorrect Data subcategory is the result of *wrong data* being input. The IP fragmentation attacks, such as *Winnuke*, and input of special characters that the software does not know how to handle, are examples of this subcategory. The Poor Authentication and Access Check subcategory includes any attack that exploits weaknesses in the authentication, and has proven to be a very popular category. One could argue that almost any attack is successful because of an authentication or access weakness, but then the database would only contain one category in which every attack was binned! Clearly, these other categories help spread the distribution.

In the initial stages of subcategory determination, a subcategory called *Improper Handling of Correct Data* was defined, which has proven to be a lightly used category. One attack that fits this category is the *testtrack_dos* entry, which sets up a telnet connection and then promptly disconnects it without entering any data. Although it is not a normal situation,

it should not generate a fault. When this occurs, the CPU utilization on the computer goes to 100%, which clearly should not be the result of such an innocent sequence of events.

2.9. Three-Dimensional Plots Showing Clustering

Some of the simpler, two-dimensional plots were then used to determine suitable three-dimensional plots that show the more popular attack modes. The same parameter plotted in Figure 12 was used also in the 3D plot of the entire flowdown structure depicted in Figure 13.

As shown in Figure 13, some of the more popular attack modes are clearly visible as clusters: namely Buffer Overflow (3), Poor Authentication or Access Check (2), and IP Fragmentation-style attacks (1). Because of the clustering effect of these parameters in this plot, they also became subcategories in the Mechanism scheme, which was introduced in Figure 5.

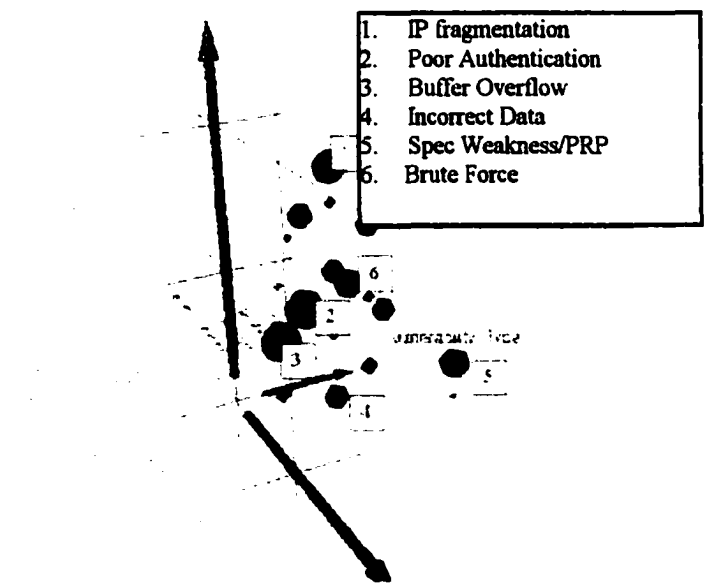


Figure 13. Vulnerability_Type versus Catalog_Type versus Attack_Type, the flowdown structure

Figure 14 shows a complex plot that involves three unrelated categories, yet the clusters give hints as to the most popular types of attacks or, expressed another way, the most common types of vulnerabilities. The data represented in the plots shown in Figure 13 and 14 led us to use the Mechanism category as the relevant category for the development categories of countermeasures, as explained in the next section.

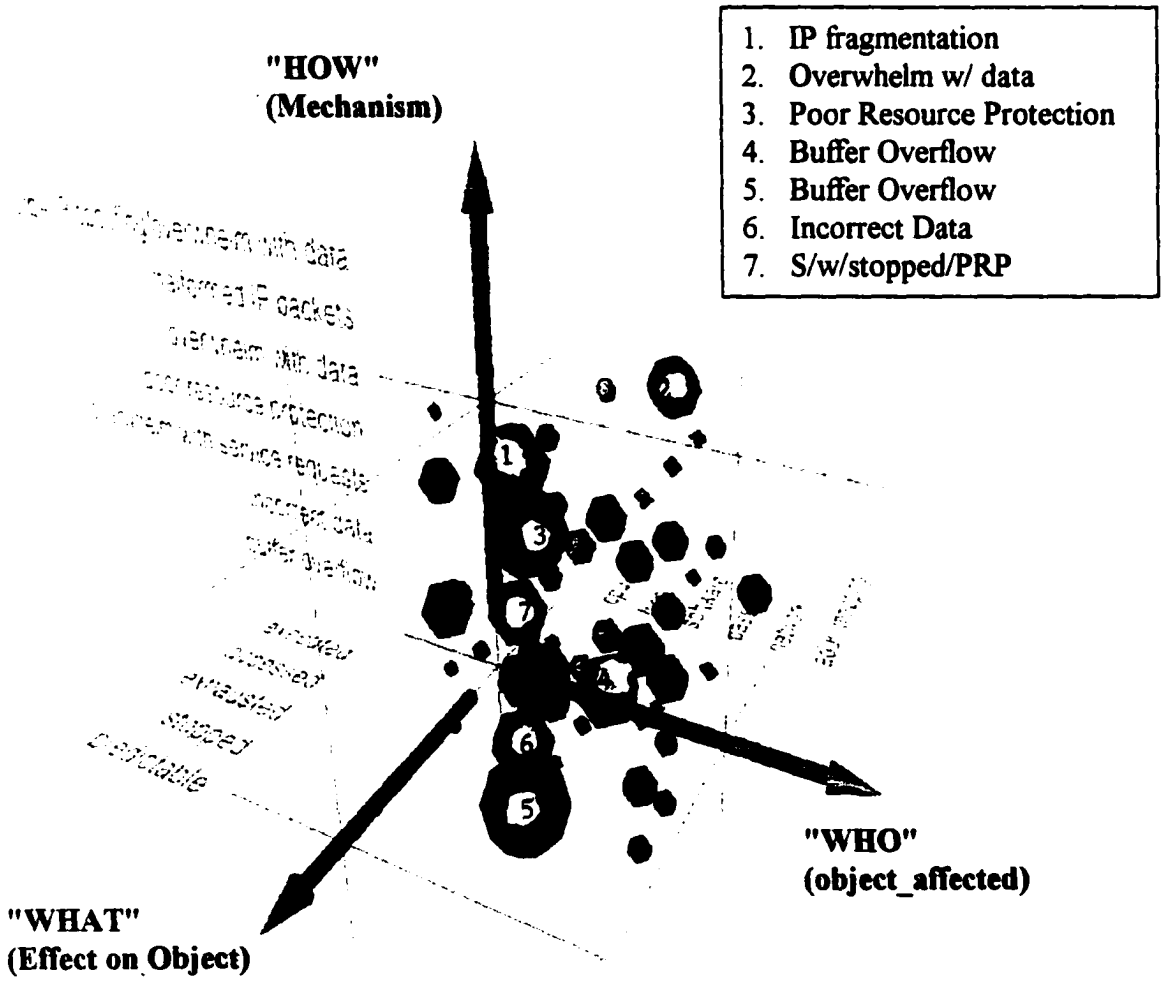


Figure 14. The Who/What/How plot showing popular categories

The three axes of this plot are comprised of: object_affected on the X axis, the mechanism category on the Y axis, and the effect_on_object parameter on the Z axis. This is referred to as the Who/What/How plot. It was plots such as this which led to confirmation that the Mechanism category is an appropriate one to use for deriving sets of countermeasures for these vulnerabilities.

From the preceding discussion, it can be seen that the flowdown taxonomy structure contained previously existing definitions for two of the three major categories, and subcategories were added for all three. Enhancements to the Krusl classification scheme led to an easy method of determining the proper categories for a given exploit, using the method shown in Appendix A. Finally, plots such as those shown in Figure 13 and 14 confirmed that the Mechanism category was the best one from which to derive classes of countermeasures.

2.10. Conclusion

The previous discussion explained how some of the ideas from earlier taxonomy classification works were incorporated into a new database structure that includes the concepts of brute force and other denial of service attacks. This new taxonomy has a means to provide ever-increasing levels of detail to the categorization process, similar to the biological classification scheme. It also provides multiple facets of classification. Some of the categories were uniquely developed in this research and others were borrowed from previous research. The database taxonomy performs superbly to sort vulnerabilities into consistent categories and subcategories. As the research progressed, the database steadily grew in size and diversity. A comparison of this database with another publicly-accessible database confirmed the assumption that this database was representative of the vulnerability population. During the long months of plotting efforts, it was discovered that certain taxonomy categories not only accurately explained *Who* and *What* the exploit attacked, but also the newly developed Mechanism category provided the *How* piece of the puzzle. For the first time vulnerabilities are classified in a multifaceted and systematic manner. Judging by the resulting clusters in Figures 13 and 14, there is a convincing argument that classes of countermeasures can, indeed, be created for these various Mechanism categories. These classes of countermeasures are the focus of the next chapter.

3. OVERVIEW OF COUNTERMEASURE TECHNIQUES

Chapter 3 explores the countermeasures currently available to the networked world as well as proposed future countermeasures under development. The objective is to survey the spectrum of relevant countermeasures in order to provide a context for the solutions proposed by this process. The chapter begins by introducing the concept of a Survivable System and discussing the elements that make up such a system (section 3.1). Techniques used to handle large amounts of legitimate traffic at busy Websites can also be used to mitigate the effects of many DoS and DDoS attacks, these techniques are discussed in section 3.2.

Sections 3.3 and 3.4 discuss other presently available countermeasure techniques that are often deployed and countermeasures being prepared for future deployment, respectively. Section 3.5 discusses the attacker versus victim imbalance in detail, showing why DoS and DDoS attacks are so simple to carry out and so damaging in effect. The relationship between countermeasures and the exploits that were classified in Chapter 2 will be established in Chapter 4.

3.1. The Definition of Survivable Systems

3.1.1. Characteristics of Survivable Systems

The goal of each Systems Administrator is to: 1) provide reliable access to mission-critical systems, networks, and applications, and 2) place controls on these electronically interconnected networks as to deter unauthorized access to the network. These controls are frequently detailed in a site Security Policy, which defines the rights and responsibilities of users and defines what the acceptable use of the system. If the easiest way to break into the network is to employ Social Engineering schemes, then the Systems Administrator and, by extension, the Security Policy, have performed their function adequately. Social Engineering refers to the efforts employed to get someone from within the organization, a help desk, for example, to grant access to a person who is not part of that organization. If there are easier routes than Social Engineering, namely, calling up administrators and help desk personnel, and

trying to get usernames or passwords from them, then more security preparation must be done.

Prior to the efforts that developed commercial firewall products that defended network gateways [14-23], research was conducted at the Software Engineering Institute at Carnegie Mellon University (CMU) that helped define the goals of these new products. CMU refers to this concept as one of Survivable Systems, and has published papers on this concept [24-27]. The premise offered in these papers is that Survivable Systems is an emerging discipline worthy of consideration when deploying network security techniques.

The rapid increase in the use of the Internet for business applications has caused a paradigm shift in the way these companies operate. CMU refers to this as the shift from bounded networks with central control to the unbounded networks with no centralized control [24]. The definition of a bounded network is one that is under direct control with known components; conversely, an unbounded network has limitless bounds and the administrator has no control over the widely dispersed devices. A prime example, of course, is the Internet. This paradigm shift is a reflection of the change in computing environments that have occurred over the past 5-10 years, and the contention is that these networks now need to take on the aspects of a Survivable System. Survivability is normally defined as, "the capability of a system to fulfill its mission, in a timely manner, in the presence of attacks, failures, or accidents" [24, 25].

The probability of malicious attacks increase dramatically as more and more organizations connect their internal networks to public networks such as the Internet. Even worse, it is often difficult to determine if network activity is legitimate or malicious in nature, and to determine if equipment or network failures are the result of normal random failure or of a malicious attack. This is certainly the atmosphere that exists today. It is very difficult to determine if a Website has slow response because of a DoS attack or because of a sudden increase in legitimate activity. Because of these uncertainties, it is best to prepare for the worst and to configure the network to withstand the worst case scenario, a full-fledged network attack by an intelligent adversary. In addition, it is important to note that it is vital for the mission to survive, and not just a particular component or subsystem [24, 25].

A key characteristic of a survivable system is the ability to deliver their intended service in the face of attack, failure, or accident. As one might imagine, this implies a great deal of redundancy and fault tolerance of hardware, software, and networking ability. CMU contends that survivable systems must exhibit at least 4 key properties [24]. These are:

1. Resistance to Attacks;
2. Recognition of attacks and extent of the threat or damage;
3. Recovery of essential services after the attack; and
4. An adaptation and evolution to reduce the effectiveness of future attacks.

In the past, Computer Security has been a binary term that suggests that the computer or network system is either safe or compromised. Firewall techniques try to stop an intruder at the gate, but an intruder that makes it past this barrier compromises the entire system. The history of computer security efforts seem to indicate that most of the work has gone into Property #1, the resistance to attack, instead of how to control the damage once the firewall is breached. The primary focus was on the hardening of firewalls, implementing both packet filtering and application level proxies, or anything to keep the would-be intruder out; nevertheless, very little work delved into Properties 2, 3, and 4. In the past, the barbarians were either kept outside the gate, or they got inside to do significant damage.

The first step, and the one most often implemented, is to add fault tolerance to a network or system that has many single points of failure by adding redundant hardware that replaces a failed or compromised unit. If one border router is a single point of failure, two or three parallel routers reduce this risk substantially. The same is true for a firewall, a Webserver, or any device that performs a needed function that has no replacement if that device fails. To build a survivable system, it takes more than a mere redundancy of components; it takes the concept of dissimilar hardware to fully implement this concept. It does little good to deploy two parallel routers if both are susceptible to the same attacks. Thus, for true redundancy and fault tolerance, Router #1 should be from vendor ABC, while Router #2, which uses a different hardware and software operating system, is obtained from Company XYZ. This greatly reduces the risk of a single vulnerability to be able to compromise both router devices.

3.1.2. Parallels in the Aviation Industry

The concept survivable systems has been used for many years in the Aviation industry, which is probably the premier example of an industry that is very much interested in survivable systems. Especially in recent years, one will rarely find an aircraft accident where just one failure caused the loss of an airplane. Even accidents frequently labeled "pilot error" are mitigated by the requirement that even the smallest passenger aircraft must have two pilots. Because of this requirement, there is faith that one pilot will notice and correct any error made by the other pilot. Due to sophisticated warning devices, even bad weather cannot be considered a single-point failure; a weather-related accident represents a failure of more than one system or process. As a last resort, the decision to fly must be made by both pilots as well as arrive or depart from an area experiencing inclement weather.

The concept of dissimilar hardware was present, even in the earliest digital aircraft autopilots. Not only were the servo motor commands that moved aircraft surfaces driven from two different hardware channels, but also the digital processing that generated the commands to the motor were conducted along at least two parallel paths that utilized different microprocessors. For example, one channel may involve signal processing using an 8080 microprocessor from Intel while the other channel may use a 6800 microprocessor from Motorola. This greatly reduced the chance that a single power interrupt or power surge could induce the same failure in both sources of processing and cause the autopilot to do something unpredicted or unexpected. This has proven to be a very successful concept, and today's aircraft avionics equipment is much safer because of it.

Of course, with only two channels, if one side's processing yields a solution different from the other, it is difficult to tell which side is correct. In the case of a two-channel autopilot, the system architecture solves the problem by merely disconnecting the autopilot and forcing the human pilot to take over and hand-fly the aircraft. On the larger commercial aircraft, it is common as well as customary to have autopilots with three channels. In this case, with a failure in only one channel, the system can use two-out-of-three voting to continue safe operation with two channels still functioning. This necessitates heightened vigilance by the

human pilot, but safe operation can continue. There are many parallels between philosophies routinely used in the aviation world and the philosophy of a Survivable System.

3.1.3. The Current State of Survivable Systems, and Potential for Improvement

As mentioned previously, much of the work in the computer/network security field has taken the path of hardening the systems or components, but very little has been done in terms of automatically determining what to do once an intrusion is underway or how to prevent future attacks.

Organizations such as the Computer Emergency Response Team (CERT), CIAC, Security Focus [28-31] and others provide a much-needed service in documenting known attacks and publishing advice on how to counteract them. These published documents are of great use because most people use commercially available hardware or software solutions for network security (COTS, i.e, Commercial, off the shelf), and any exploit discovered for vendor XYZ is likely to be effective against a similar device from vendor XYZ. Since vulnerabilities and exploit scripts are routinely posted to publicly available newsgroups and Websites, at least one exploit will attempted against a number of targets. For this reason dissimilar hardware solutions can be of great service to protect the network, ensuring that vulnerabilities posted to a public resource do not cause the collapse of an entire network protection scheme.

Groups such as CERT also provide another useful function, that of advice and guidance once an organization's network is under attack or has suffered such an attack. Some commercial products contain the intelligence to take proactive steps in the face of an intrusion attempt; but, if these measures are not in place, little that can be done to secure a network once it is under attack, short of pulling the plug on the network connection. Most firewall products secure networks as passive, filter-only devices, but it is hoped that most will obtain enhancements that enable them to play a greater role in a survivable system by the addition of intrusion detection and dynamic-response capabilities [24, 25]. Adding the last three properties of survivable systems (i.e., Recognition of attacks and extent of the threat or damage; Recovery of essential services after the attack; and Adaptation and evolution to reduce the effectiveness of future attacks [24] will be the goal of creating survivable systems

of the future. The actual deployment steps will depend only on the organization's desire to protect its data and its budget for such attack countermeasures.

The purpose of mentioning the Survivable Systems concept and expounding on the concepts used in the aviation industry was an attempt to emphasize the concept of a flexible response to computer security. It is also important to note that economic as well as security issues drive countermeasure techniques. One could build the world's safest aircraft or the most secure computer network; but if it is not cost effective, no one will either build or buy it. In the commercial world, most decisions come down to a cost/benefit analysis; and Survivable Systems and Flexible Response must use the same considerations to balance the conflicting factors.

3.2. Handling Heavy Loads of Legitimate Traffic

If a computer or a network suddenly becomes very busy, it is difficult to determine if the lackluster performance is due to a DoS attack or to an increase of legitimate traffic. Companies that chose to run television advertisements during popular timeslots or during popular events have learned to make their Websites more robust in anticipation of the increased Internet traffic to their sites. Over the years, organizations have learned to use techniques to help facilitate the handling of this increased traffic load. Section 3.2.1 discusses redundancy, failover, and load balancing, while section 3.2.2 covers the various Quality of Service, or QoS, techniques.

3.2.1. Using Redundant Hardware, Failover, and Load Balancing

One common DoS attack is to overwhelm the networking devices by shear volume of data or shear volume of connection attempts. In the early days, routers, firewalls, and even Webservers were known to fail regularly, whenever unusual volumes of traffic were experienced. Systems are much more robust today, but it is still a good idea to construct networks with dual redundancy in the routers and firewalls, and perhaps with triple redundancy, or better yet, in the Webservers. Networking devices such as routers and firewalls can be installed as pairs and programmed to automatically track the state of the

connections being handled by the other device. In this manner, if one router fails, the other one can pick up the traffic and not lose any of the present connections.

In addition to redundant hardware such as routers, firewalls, etc, another networking device that has gained popularity in recent years is called the Load Balancer [16, 19-22]. This device typically sits in front of a Webserver farm and handles the connections to the Webservers from the outside, or untrusted, network. This device makes it appear that there is only one Webserver at one IP address, but in reality it is parsing out the connections to a multitude of Webservers to handle periods of high traffic. This can be a vital component for efficient operation at busy Websites.

Not only does this device help the Website to survive under periods of heavy loading, but more sophisticated load balancers also monitor the time that a given connection is left open. The device can then shut down inactive connections or connections where the *SYN* command has been received from a Web browser and a *SYN - ACK* command given, but where the third and final piece of the handshake has not been received. This is a direct countermeasure to one of the more common types of DoS attacks, the Synflood attack. In this type of attack, the attacker opens a large number of connections to a Webserver by sending the first packet of a three-way handshaking protocol, but never sends the third handshake. Because of network latencies, many servers will leave this connection open for long periods of time in hopes of finally receiving the third transmission. Attackers can take advantage of this in order to fill up the connection table with half-open connections in order to squeeze out legitimate users or to get the server to crash. In either case, a DoS results [32-37].

A sophisticated load balancer not only parses out all of these connection attempts across a multitude of servers, both legitimate and malicious connections, but can also monitor and then close these half-open states to free up room for legitimate users. In some cases, the device can recognize specific address ranges that look suspicious and can dynamically discontinue these connections for the period of time that it feels that it is under attack by a malicious source, even if source address spoofing is used. There are many commercial products available in this category [19, 22, 23, 34, 36], and these devices serve as important

countermeasures against the *Overwhelm with Service Requests* category of DoS attacks that are so common today.

3.2.2. Quality of Service (QoS) Techniques

In the current networking methods, a best-effort model of packet delivery is followed. Delivery service levels degrade gracefully as more and more network traffic is added, but many time-sensitive applications cannot withstand these increasing delays, referred as network latencies. These applications include voice-over IP, teleconferencing, video and other forms of multimedia. Quality of Service (QoS) methods manage the available bandwidth like the scarce resource that it has become [38-45]. The current bandwidth situation, across the Internet and within many enterprise domains, is much akin to the situation involving other renewable resources. For example, in the early days of tree logging in the United States, trees were cut without much consideration of the consequences; but, as the land became more crowded and trees became scarcer, it was recognized that conservation techniques had to be implemented.

Many protocols, such as Email and FTP, are not severely impacted by network latencies. However, because the arrival time of the packet(s) can be important, many of the more current multimedia protocols are impacted. Several bandwidth management techniques have been developed in recent years that allow network managers to allocate and control the bandwidth that their networks spend on any specific application. Many of these are discussed by the company QoSforum.com (1998) [41], and require that some decisions be made of the network, the traffic, and the methods of control. The goal of any QoS method is to achieve an improvement the current method of best effort delivery of IP packets. These techniques represent various levels of effectiveness and complexity, as well as cost factors. QuoSforum.com [41] outlines these major techniques as follows:

QoS methods are generally classified into two types: Integrated Services protocols such as Resource Reservation, and Prioritization methods such as Differentiated Services. In addition, these techniques can be applied on a "per flow" basis, roughly akin to "per connection", or they can be applied to an aggregate of flows, applying to many connections. The major focus is placed on the two main methods: Integrated Services, and Prioritization. A

key consideration would be implementation of these QoS methods, without supplying an attacker with a way to launch a DoS attack under the auspices of a QoS method.

The Resource Reservation protocol, usually abbreviated as RSVP as a familiar acronym for ReSerVation Protocol, is an Integrated Services method that apportions network resources in a manner that resembles an end-to-end virtual circuit, allocating bandwidth in accordance with a bandwidth management policy. This protocol reserves portions of bandwidth for specific applications or traffic categories.

The Differential Services protocol, usually abbreviated as DiffServ, is a Prioritization method that provides a way to prioritize the network traffic, and is the QoS method that is most applicable to preventing Denial of Service attacks. In practice, both RSVP and DiffServ are widely used as complementary techniques, so both will be covered in some detail. To provide a more comprehensive review of QoS in general, a summary of other methods is provided in the next two sections, followed by a more thorough discussion of the RSVP and DiffServ protocols.

3.2.2.1. Multi Protocol Label Switching (MPLS). In addition to the two main techniques of RSVP and DiffServ, Multi Protocol Labeling Switching (MPLS) can be used to establish routing control through a network. The MPLS protocol marks traffic at the network's ingress points with a 20-bit label that is used to determine the next router in the path, establishing bandwidth pipes of fixed bandwidth. MPLS does simplify the routing process, but is applicable only to routers that recognize this protocol. It is not used to defeat DoS attacks since any increased traffic workload, due to the bogus traffic, will be routed along with the remaining normal traffic.

3.2.2.2. Subnet Bandwidth Management (SBM). Subnet Bandwidth Management, or SBM, exists at layer two of the OSI protocol stack, and encompasses IEEE standards that assist in the delivery of time-critical traffic. Using the IEEE 802.1p, 802.1Q, and 802.1D standards in this effort, SBM gives QoS features to the Ethernet layer of LANs. SBM at the Ethernet layer closely resembles RSVP processing in a router, and can be classified as Centralized or Distributed depending on the location of the required bandwidth allocater

(BA). A 3-bit field determines the eight SBM levels, and even this implied simplicity can become very complex. Since it functions at layer 2, SBM is typically used only within a given LAN and does not improve end-to-end network performance. The ISU vulnerability database assumes that any DoS attack originates from outside a given domain-space, therefore, this particular method is not of immediate interest.

3.2.2.3. ReSerVation Protocol (RSVP). As mentioned previously, RSVP, or ReSerVation Protocol, establishes a virtual pipeline from source to destination that sets aside a bandwidth for an application, or a set of applications, that are totally reserved for only that application; thus emulating the concept of a dedicated telephone line. It is the most complex of all the QoS methods, and requires all the routers in the path to be RSVP-enabled. RSVP is a convenient acronym, because it clearly denotes what the application does, via APIs. It must request and receive bandwidth allocations from each of the intervening routers, and hold these resources throughout the communication period. When finished, the setup period must be reversed by the teardown process, or the connection will eventually time out. A detailed description is provided in [41].

RSVP is a form of the Integrated Services method of QoS, and is quite complex in that the applications desiring to establish a QoS environment must use APIs to initiate the reservation requests, track the status, set up the session, and then tear down the connection when finished. This is quite time-consuming; and since non RSVP-enabled routers are included in the router chain, there is a weak link where the best efforts of Ipv4 must be used for that particular link. Like the other previously mentioned protocols, this protocol does not have a great potential for alleviating DoS attacks based on high volumes of Network traffic or from large amounts of connection requests. The DiffServ method, discussed below, has more promise for DoS attack alleviation and is frequently used for this purpose. It has several advantages over the RSVP method, not the least is the ability to deploy the concept in only one network border device, rather than an entire series of enhanced routers.

3.2.2.4. Differential Services (DiffServ). A version of the prioritization method, the DiffServ protocol assumes the existence of some sort of Service Level Agreement (SLA)

between bordering networks that it seeks to enforce. DiffServ will mark and prioritize traffic based on the service, protocol, time of day, source and destination addresses, and a number of other parameters, including the ToS bits in the IPv4 protocol header. It can designate a certain percentage of bandwidth to be given for a certain protocol or traffic type, and then discard the excess traffic if so programmed. This makes it especially useful for throttling several DoS attacks that rely on volume or quantity of data, such as the attacks listed in the *overwhelm with data* or *overwhelm with service requests* categories. The DiffServ protocol complements the RSVP protocol in that RSVP is typically deployed at the outer edge of the network and DiffServ is typically deployed at the core, or center, of the network.

The concepts of DiffServ are most readily implemented in routers, but Internet routers are beyond the control of the network manager. Therefore, the most convenient place to establish these levels of bandwidth usage is in some sort of bandwidth broker at the border of the enterprise network. These devices can be stand-alone devices, but are more commonly deployed as part of a firewall or VPN that serves as the point of entry to the enterprise domain.

One such product is Checkpoint's Floodgate-1 bandwidth control software, which is deployed as part of the Firewall-1 and VPN-1 products. Floodgate-1 is a prime example of a commercial implementation of the DiffServ method, using weighted fair queuing to establish traffic priority so that higher priority traffic moves quickly across the network interface, while assuring that even lower priority traffic is not starved for bandwidth.

The Floodgate-1 module is deployed at the same traffic control location as the firewall, avoiding duplication of the stateful inspection process since the firewall module is operating in this mode for security purposes as well. By monitoring and controlling the traffic based on the stateful inspection process being performed on traffic for security purposes, the traffic can be inspected once and the information used across several applications. Many commercial firewalls already use stateful inspection [45], therefore, the bandwidth management process does not add much to the processing overhead. In addition, by monitoring the policy at Internet/intranet access points, network congestion can be alleviated.

In the weighted priority scheme used by Checkpoint in Floodgate-1, types of network traffic can be given more priority by giving these classes more weight. For example, outgoing HTTPS traffic, which is the protocol used for encrypted Ecommerce traffic, can be given five times the weight as incoming HTTP traffic, which is probably being used by internal users for Web surfing. On the other hand, HTTPS can be given a weight that is two times the level given to incoming FTP traffic used in file downloads.

While providing limits, guarantees, and priorities, *weighting factors* can be assigned to the traffic based on a number of parameters, including IP source/destinations, groups of users, application, traffic direction, time of day, and other parameters. This ensures that high priority traffic reaches its destination and provides a means to discriminate against protocols and traffic types that could be DoS in nature.

To understand how this is applicable as a countermeasure to DoS attacks, one has to realize that if desirable traffic can be identified and given a guarantee, or even a priority over what becomes a flood of bogus traffic from random sources, then it is not possible for the bogus traffic to steal large portions of bandwidth. While here will be some performance degradation, the bogus traffic should not be able to consume all the bandwidth space, which would be the case in the current best effort IP traffic delivery method. For example, this method would be very effective in reducing the effects of a Smurf attack, because bandwidth limitations could be placed on the percentage of traffic of that nature, namely ICMP responses. Other TCP and UDP flooding attacks could be controlled as well. Thus, QoS mechanisms can be considered effective countermeasures for both the *Overwhelm with Data* and the *Overwhelm with Service Requests* categories.

3.2.2.5. Limitations. The remaining weakness in this area of using QoS methods to alleviate the consequences of DoS attacks is in the area of equal service requirements, namely in the realm of initial HTTP connections from random and unknown IP addresses. How does the system distinguish between a legitimate connection attempt and a bogus connection attempt? In the QoS prioritization scheme, they would be of equal weight, or priority, and there is the chance that a flood of bogus connection attempts could squeeze out connection attempts of legitimate users. Known classes of users or users from known address space could

be given either a positive or negative priority in terms of Web connections, but anonymous Web connection attempts, both legitimate and bogus, would have the same priority level. This is where other DoS countermeasures may have to be considered and deployed, because QoS techniques will have reached the limit of their effectiveness.

3.3. Present Techniques for Countering Attacks that are Deliberate in Nature

There are two general types of attacks that are deliberate in nature. The first type includes the Buffer Overflow, Poor Authentication, and IP fragmentation attacks, and the second type includes flooding-style attacks. Flooding attacks rely on a large volume of requests for service or overwhelming the recipient with large volumes of data, whereas the first type of attack relies on the inability of the receiving software to know how to handle misconfigured commands or packets. The particular style of the first group is not usually remembered when considering DoS attacks, yet these attacks account for the majority of attacks categorized in the database. These are considered in section 3.3.1, and the flooding-style attacks in 3.3.2

3.3.1. Countermeasures for Buffer Overflow, Poor Authentication, and IP Fragmentation Attacks

The following subsections detail the limited countermeasures available for the "non-flooding" style of DoS attacks. This style of attack is the direct result of the quality of the data received, rather than the quantity of the data received as in the flooding attacks. These are categories created for the present research's database schema that include a large quantity of DoS attacks and vulnerabilities, but are usually not considered when analyzing the flooding style of DoS or DDoS attacks that are so popular in the media today.

3.3.1.1. Countermeasures for Buffer Overflows. The most famous example of a buffer overflow attack was used by the November 1988 Internet worm, and involved the finger service on Unix machines. This buffer overflow replaced the Unix program being run with a command interpreter, or shell, which was then used to transfer a new program that executed a new copy of the worm program, which would seek out a new victim. This form of attack was not used very often until late 1996, when Aleph One published his now-famous

paper, "Smashing the Stack for Fun and Profit" [46]. Since its publication, buffer overflow attacks have represented a sizeable percentage of all security exploits. In the ISU database, they represent more than one-third of the exploit entries.

Most buffer overflows are a two-step process. One must first get the exploit code into the program's address space, and then get the program to jump to that new code. More detail on how this is accomplished is provided in Cowan et al. (1999) [47]: Section 2.1 (Ways to Arrange for Suitable code to be in the Program's Address Space; and 2.2 (Ways to Cause the Program to Jump to the Attacker's Code).

Other than convincing code writers to conduct bounds checking on user-supplied input at the time they write the code, there are a few techniques, none of which are simple, to patch buffer overflow problems after the fact.

Cowan et al. [47] proposes four defenses against buffer overflow attacks. The first involves bounds checking at the outset, or what is termed the Brute Force method. The second method, or Operating Systems approach, makes the storage areas for buffers non-executable, preventing the code-injection phase that was referred to previously. The Direct Compiler mode, the third method, performs bounds checks on all array accesses, and the fourth one, Indirect Compiler approach, performs integrity checks on the code pointers as an add-on to the compiler function.

These techniques require the software writer to either do it right the first time (the first defense method), change the operating system (second defense), or have the compiler check the code after the fact (third), as dictated by method 3 and 4. Every option, except the first, requires changes either to the compiler or to the OS; both of these changes would prove to be slow to implement. Of the four options presented, the 4th method seems the most logical, since it is merely an add-on to the gcc compiler, and is readily available today in the form of commercial products such as *StackGuard*, along with a sister product called *Pointguard*. While these products will not prevent every conceivable type of buffer overflow attack, they offer some help. It is anticipated that these recently introduced products, if commercially successful, will improve in quality and scope much like firewalls, routers, QoS devices, and Intrusion Detection Systems have improved over the years.

Other than these methods, there is little that systems administrators can do other than keep their software current with the latest patches so that the systems are protected against known vulnerabilities. This requires constant vigilance and must be considered as part of the price of operating in today's networked world.

3.3.1.2. Countermeasures for Poor Authentication or Access DoS Attacks. The *Poor Authentication or Access Check* type of DoS is comprised of greater than one-third of the database but, because of the limitations inherent in the IPv4 protocol, there is not much that can be done to prevent this type of attack. If authentication is not built into the protocol or application, there is little that can be done besides keeping the software up to date, in terms of patches, and to implement IPSec or manual authentication wherever practical. In the future, IPv6 holds great promise for the ability to mutually authenticate parties, assuming they want to be authenticated, however, widespread implementation of IPv6 is considered to be at least two years away. Because the authentication protocols are optional rather than mandatory, it remains to be seen whether IPv6 becomes an effective countermeasure.

3.3.1.3. Countermeasures for IP Fragmentation or other *Wrong Data* attacks. As mentioned in the previous section, IPv6 also holds promise as an effective countermeasure for IP fragmentation attacks, since IPv6 allows for an end-to-end negotiation for if, and how, packets get fragmented. Currently, it is important to keep networking software patched and to keep firewall and IDS software up to date, since many modern firewalls and IDS products can protect against known IP fragmentation attacks. The problem remains with the unknown attacks, which tend to be discovered and propagated at an alarming rate. Much like the buffer overflow situation, this creates an arms race, of sorts, between malicious users and the administrators who are tasked with protecting these networks and systems. In this scenario the first victim of a new exploit, by definition, is helpless. Nevertheless, it is hoped that the software vendor will patch the vulnerability quickly, thus enabling the remaining networked population to gain immunity from the newly discovered hole.

3.3.2. Countermeasure Considerations for Flooding Attacks

Unlike the case of the "quality of data" attacks, the flooding attacks rely on overwhelming the victim with an inordinate amount of requests to service or by sending large quantities of data packets to process. Both scenarios rely on the quantity of data or requests, as opposed to the quality of the data as discussed in the previous sections.

3.3.2.1. Current Countermeasures for DoS and DDoS Attacks that Overwhelm with Data or Service Requests. For countermeasures, it is useful to narrow the field of consideration from the six mechanism categories to the two categories dealing with *too much data* or *too many service requests*, which are the heart of most DoS and DDoS attacks. These attacks seem to be the ones most actively reported by the media. For the purposes of this discussion, these two types of attacks are considered together, since the countermeasures deployed are similar for both.

An example of the *too many service requests* category of attack is the Synflood attack, which has been discussed in the previous sections. Another example might be an attack that generates a lot of entries into an error log, filling up disk space, or consuming processor time.

The most notorious example of the *too much data* category of attack is the Smurf attack [48-50]. In this attack, the ultimate victim is inundated with traffic that they did not request. Actually, there are multiple victims: the victims that receive a flood of packets, and the subnetwork that is used to generate this flood. By using a false, or spoofed, source address, the attacker sends an ICMP broadcast message to the broadcast address of a subnetwork that may have hundreds of host computers. If the subnetwork's router allows this type of message, i.e., broadcast messages are passed from the layer 3 router to the layer 2 subnet, then this broadcast message tells every host listening to send a reply back to the source address in the broadcast message. The effect of that single short broadcast message, asking if "anyone is here?" solicits a reply from every host, back to the victim, that says, "yes, I'm here". This results in a flood of traffic back to the victim, who is left to handle the flood of responses. The subnet that is used in this manner is typically referred to as the bounce site, and there are sites on the Internet that list subnets that allow this type of broadcast message. Attackers have no trouble finding bounce sites to use, even years after this type of attack

became common knowledge. Most, if not all, routers have a way to turn off this broadcast message and negate the effectiveness of the attack, but there are still large lists of suitable bounce sites.

The Distributed DoS (DDoS) attacks of February 2000 spawned a great volume of information dealing with DDoS in particular, and on DoS attacks and computer security issues in general. These various RFCs, CERT advisories, workshop minutes, and white papers can be placed into three distinct categories in terms of reader usefulness: 1) Practical tips and current countermeasures; 2) future techniques under development; and 3) nice things to know. The current research focused on the first two.

3.3.2.2. The Distributed Denial of Service Attacks (DDoS). Partially through the current research effort, a new and disturbing method of launching DoS attacks was been identified. In a variation of the old methods, these new DDoS variants launched the same old DoS attacks from hundreds, or thousands, of compromised network-based hosts instead of from only one source. The victim is virtually helpless in the face of the onslaught of network traffic destined to the network connection. The packets come from a variety of hosts, usually with many spoofed source addresses. A discussion of DoS attacks would not be complete without considering these Distributed Denial of Service (DDoS) attacks which have been in the news freqently, particularly since the February 2000 attacks that crippled several high-profile Internet sites such as Yahoo!, CNN, Zdnet, and eBay. At the time of the attack, the majority of the mainline media treated these attacks as something new, although the Computer Security world had been aware of this threat since August of 1999, when a similar attack was launched against a major university shutting it down for several days. CERT held at least one workshop on this issue in November of 1999 [51], and issued advisories on the topic months prior to the highly publicized February attacks [52, 53]. The publicity surrounding these attacks certainly put DoS attacks, and Computer Security in general, high on the radar screens of many Corporate Security Officers, ISP operators, and the general public.

Despite the warnings from CERT, the topic did not receive widespread attention until the first week of February 2000, when attacks crippled well-known public Webservers. The

DDoS attacks are discussed in detail in [54-71], however, a brief description of this insidious attack is as provided as follows.

An intruder plants zombie subroutines on Internet-based hosts through well-known and published vulnerabilities. These subroutines hide themselves among the other processes, and open a port to listen for the command, from the master controller, to launch the attack. The command is given by a single packet that tells these processes the IP address of the victim, as well as the type and duration of the attack. After that, a flood of packets or connection requests start up and hundreds, possibly thousands of these packet-generator subroutines saturate the hapless victim, usually filling the network pipe and eliminating any chance for legitimate traffic to getting through.

When considering appropriate countermeasures, it makes a difference as to which particular aspect of this attack is being defended against; appropriate defenses against the deluge of packets by the flood victim or appropriate defenses against becoming a zombie site launching the flood of packets.

3.3.2.3. *Flood Victim* Protection. Since there is little that a victim can do to shut off the flood of packets heading towards the networking address space, efforts have focused on ways to stop spoofed packets or identify the true source of the bogus packets. Trying to find the true source has typically involved some form of traceback mechanism, which is complicated by the various administrative domains that the packets flow through, from attacker to victim, as they cross the Internet. As a bare minimum, the administrator of the victim subnetwork should immediately contact the upstream ISP and attempt to get the bogus traffic stopped and the source(s) traced. In addition to these steps, the following paragraphs offer other techniques to counter such traffic.

It is a laborious process to trace packets back to their true source, since the packet only has information concerning the previous hop that the packet took on its way through the network. Tracing the packet backwards involves the personal attention of systems administrators from each of the domains that the packet travels through until the source router is determined. This tracing process may be feasible in an attack that is launched from one source, but it quickly becomes futile in the face of DDoS attacks from many sources. Not only

are the source addresses spoofed, but these packets may also originate from hundreds or thousands of hosts that are themselves victims of compromise. Tracking the source, or sources, of these packets is difficult at best, even with advent of automated tools that once existed but are currently unavailable.

In the Fall of 1997, MCI announced a new tool called *DosTracker* [72] that had the ability to track DoS attacks across a network. There were two caveats: 1) the network was under a single administrative control (e.g., a large ISP); and 2) it was comprised of Cisco routers. Several press releases could be located during the October 1997 timeframe that reference the URL to go and download this tool, but these Websites no longer exist. There is no further reference to *DosTracker* at the MCI Website, nor at any current security Websites. Perhaps these two caveats were enough to kill the usability of the tool; and, therefore, the tool had no hope of living up to its hype. Any similar tool that is developed in future will need to interoperate with various router vendors and across various administrative domains to overcome the limitations of MCI's DOSTracker.

Instead, most of the recent literature recommend several methods of mitigating these attacks. The following techniques or groups of techniques are discussed in subsequent paragraphs:

1. Ingress/Egress filtering;
2. Traffic shaping, bandwidth control, and other QoS mechanisms;
3. Spreading the site out over several IP addresses and use load balancing hardware; and
4. Using intermediate hardware and proprietary techniques developed to determine bogus from legitimate traffic

The universal application of Ingress or Egress filtering to all routers that serve as onramps to the Internet would prevent spoofed traffic from propagating very far [56, 71]. Whether it is called Ingress or Egress filtering, the process is the same depending on the point of view of the observer. Egress refers to the traffic leaving the local area network (LAN), whereas ingress refers to the traffic that is entering the ISP's domainspace on or near the Internet backbone. In both cases, the filtering being advocated involves ensuring that the packets leaving the LAN have a source address that is reasonable for that domain space. Since

the Postcard Analogy is used so often to describe the path of a packet through the Internet, one can carry that analogy a step further to describe the filtering being proposed.

The flow of packets through the Internet is frequently likened to that of a postcard through any government's postal service. Postcards, like networked packets, have destination addresses to tell postal workers where to deliver them, and the source address lets the recipient send a response back to the sender. Of course, the source address is not crucial to the delivery of the postcard; therefore, its validity is not usually checked closely by postal workers. After the postcard enters the postal system, it is nearly impossible to ensure the validity of the source address. The only time to ensure this is when it enters the system, i.e., when the postcard is being picked up by the postal worker. It is only here that the validity of the source address can be confirmed, and only if the postman takes the time to do so.

Currently, most routers are not set up to do this type of filtering; most routers at the Internet's onramps blindly pass along traffic, allowing packets with invalid source addresses to enter the system unchecked. Another consideration is that this added task would overload some older routers currently fielded, severely impacting their ability to perform. Although such filtering rules can be added manually, this was not the default condition on routers delivered from vendors. Router manufacturers are just now starting to ship products that have this type of filtering enabled as the default condition, which requires the source domain space to be entered as part of the router's configuration. The status of the RFC that promotes this type of filtering, RFC 2827 [73], which obsoletes RFC 2267 [74], has been upgraded to the status of "Best Current Practice", titled BCP 38.

This BCP does a lot to prevent source address spoofing, but it takes time to reprogram for the proper configurations to propagate throughout the entire fielded router population. Although it does not totally prevent DoS attacks, it performs two valuable functions: 1) the allows the attacking packets to be traced to their true source, removing the ability to remain anonymous; and 2) it discards any packet that does not have a reasonable source address (i.e., packet spoofing). It is easy to see that if this filtering were widespread, the ability to transmit forged packets would become more difficult.

At first glance, it would appear that a local systems administrator would have little motivation to employ this type of filtering. After all, this action does nothing to prevent a host in the domain from being a victim of a DoS attack; the onslaught of packets will come toward the network with or without this filtering. However, if employed, this action can prevent the local domain from be accused of originating the attacking DoS packets, which is of great value. In addition, if such filtering is in place, the local (LAN) domain space becomes a much less desirable place for attackers to plant their zombie subroutines to attack others. Traffic shaping or QoS techniques also limit certain types of traffic from overwhelming the LAN's internal network bandwidth, but the packets will still fill the network pipe to the LAN until dropped at the outside interface of the firewall. Therefore, it is important that network administrators develop a close relationship with their Internet Service Provider (ISP) so that if their network comes under attack, they know whom to call to turn off the flood of packets. This procedure needs to be established well in advance of the anticipated need; searching for the correct name and phone number of the ISP contact should not be done, for the first time, in the middle of an attack.

Closely related to QoS techniques [19, 38-45, 75] are the techniques that can be used to mitigate the effects of a Synflood attack; namely, increasing the number of simultaneous TCP connections that can supported, reducing the period of time that a half-open connection will wait for the third handshake, and employing intermediate devices that can serve as a proxy for the connection request [34, 36]. By completing the connection on behalf of the server, the intermediate device, typically a router, verifies the legitimacy of the request before passing the connection over to the server for handling.

Most Network IDS will recognize DOS/DDoS attacks and send a warning to an administrator, but some are taking even more proactive steps, including using network or router configuration to block these attacks. Techniques involving these more proactive steps are considered in a separate section.

Other methods used to mitigate attacks involve spreading the site out over several IP addresses [61], across multiple Web-hosting facilities, and using multiple ISPs to ensure connectivity. Similar to redundancy concepts used in other industries, this technique can

prevent a single failure of a hardware device, DNS server, Web-hosting site, or ISP connection to completely block access to Web-based resources. Hardware devices such as Load Balancers can help handle the large amounts of connection requests, some legitimate and some bogus, that can flood a Website in a DoS or DDoS attack.

Some routers use proprietary methods to help determine if the source of the packet seems legitimate. Cisco (2000) uses a method called Unicast Reverse Path Forwarding [76] to determine if the packet arrives on a router interface that is consistent with it's source address. The method is not conclusive, and the closer the router is to the source of the attacking packets the more effective the technique; but it is a proven method only if the limitations are respected. Another Cisco (2000) innovation that helps alleviate a Synflood attack was mentioned in a previous paragraph. It is their TCP Intercept feature [34, 36]. This feature allows the router to act as a proxy for the subsequent Webserver and to verify or authenticate that the three-way handshake can be completed before allowing a TCP SYN packet to proceed onward. This prevents bogus connection requests from reaching the server and tying up resources. Similar features from other vendors are needed, and are most likely under development.

Other than the methods mentioned here, there are few ways to provide relief from DDoS attacks and most ordinary DoS attacks. As discussed in a previous section, MCI developed a product called DoSTracker to help trace DoS attacks in progress across the MCI network [72]. Nevertheless, like a phone trace, the attack had to stay in progress until the source was determined. In addition, it was entirely possible that the offending traffic went through routers that did not supply the information to continue the traceback to the source, thus the DoSTracker tool was of limited use. Reports exist of a tool that was used to illegally hack into routers and gather this information during the Gulf war, allowing one to quickly and easily gather the traceback information without spending a lot of time trying to get this information verbally from systems administrators. Since this involved illegal access to routers not under the operator's control, it is not a practical technique. More reliable, robust, and ethical methods of traceback are sorely needed.

With no form of magic-silver-bullet solution to DoS/DDoS attacks, the network under attack must do something to limit the barrage of traffic heading its way. Proposals that allow downstream LAN administrators access to the ISP's router configuration for automatic shutoff of this bogus traffic have been presented, but have received a lukewarm reception, at best. Strongly authenticating the originator of such a request, determining that the traffic flow to the victim network is, indeed, excessive, and allowing such router configuration changes by a customer seems to be beyond the confidence level of most ISP managers. This idea may be practical when Public Key Infrastructure (PKI) systems have been in place and have proven themselves reliable for mutual authentication. For now, that day seems far away. In theory, it is possible to allow certain levels of control to an upstream router by a downstream system administrator, strongly authenticated with limited access control, but no examples of this could be found.

3.3.2.4. Preventing a Network from Becoming a DDoS Attack Site

Several factors should be considered to prevent a domainspace under local control from becoming a launch site of DDoS attacks. Two of the countermeasures are the same as those listed in the previous section, egress filtering and bandwidth limits. These techniques can be used to ensure that no packets leave the local domain space without a legitimate source address, and that the traffic that does go out is not part of a flooding attack aimed at a network.

In addition, firewalls and other network hosts should follow normal security safeguards such as staying patched, being limited on having only the traffic and protocols needed to perform their function active on the machine, and hardening public machines outside the firewall. In addition, quality Intrusion Detection Systems should be run on both the network and the hosts to capture evidence of DDoS processes and other malicious activity. As with anti-virus software, IDS software needs to stay current so that new attack signatures are recognized in a timely manner. These and other recommendations are covered by Simple Nomad (2000) [69]. Tools such as *Tripwire* can be used to tell the administrator if files have been modified on a computer, perhaps flagging an intrusion that plants these DDoS zombie tools.

In the model of DDoS attacks, an attacker finds vulnerable computers across the Internet and plants hidden processes on these machines that he/she hopes to use later in an attack on a hapless victim. The attacker's software keeps track of where these processes are hidden, and these processes silently wait for the command packet that tells the process who to attack and how long to continue the attack. These command packets have what is termed "marching orders", which can be transmitted in a variety of ways. At least one of the recently discovered DDoS attack tools uses ICMP replies because many firewalls incorrectly allow ICMP replies to pass from the outside Internet to the inside network, in the assumption that the reply was in response to an ICMP query.

There are other legitimate arguments to the philosophy of limiting most ICMP messages; not only can they can be used to carry these command packets, but they can also give the attacker information about the hosts on the inside of the network [69]. As a general rule, it is good policy to disallow any type of traffic that is not needed by the local network.

To recap the discussion on the various forms of currently available flood victim countermeasures, the following paragraphs summarize the strengths and weaknesses, without the full detail provided in the previous paragraphs.

3.3.2.5. Summary of Useful Techniques to Prevent Flooding. To stop the flood of incoming packets or connections, contact the upstream ISP to shut off the flood of bogus packets and to initiate the traceback process. This assumes that the bogus traffic can be separated from the legitimate traffic, either by protocol, service, or origin. Also, it assumes that the ISP can be contacted in a timely manner and that they have the necessary technical staff to start the traceback process. If the upstream ISP is willing and able to perform this function, the downstream target officially notifies them of the problem and may successfully put the onus on them to find the source of the bogus packets. Unfortunately, more often than not, an ISP contact may be hard to find or disinterested if their own customers are not impacted. Some Internet Service Providers will not initiate action unless law enforcement agencies are involved.

For a variety of reasons, it is important to perform egress filtering on all traffic that leaves the local network. It is assumed that the subnetwork boundary is easy to identify, and

that the subnetwork uses a finite range of IP addresses that they are easily demarcated from the rest of the Internet. This will also prevent packets from leaving the network without a reasonable source address, and make the LAN domainspace a less desirable location for DDoS zombie subroutines to be planted, and eliminate the subnetwork as a source of bogus packets. Unfortunately, this filtering may place an additional workload on the routers, which may impact network performance.

One can also rely on the firewall to do QoS or bandwidth filtering or rate limits on the incoming or outgoing traffic streams; but, as before, the legitimate traffic has to be distinguishable from the bogus traffic. Most modern firewalls will perform these functions, yet the bogus traffic can still fill the network pipe, even if it does not breach the interior network. Again, it may be hard to separate the traffic and allow only legitimate traffic into the network.

Certain unique features offered by various network product vendors can help significantly. Some vendors, such as Cisco, have proprietary means of distinguishing bogus traffic, such as Unicast Reverse Path Forwarding [76], and ways to verify the source of a TCP SYN packet before passing this on to the server, called *TCP Intercept* [36]. This assumes, of course, that Cisco routers are widely deployed in the network, or that other router vendors have similar capabilities. These features are useful only if properly configured and utilized, and the results interpreted correctly.

For Synflood type attacks, one can increase the number of simultaneous connections allowed, decrease the timeout period, and drop older connections when new ones arrive. These techniques are simple and effective with today's improved devices and processor power. On the other hand, one can use load-balancing hardware to spread connection requests across several servers. Many busy sites need to implement this just to handle the legitimate traffic, and the additional service capability allows for capacity to handle legitimate and bogus traffic with ease. This is such a useful configuration that many servers are routinely deployed in a Web farm arrangement, allowing for ample redundancy and capacity. The only downside to these techniques is the cost of the additional hardware.

Using the various QoS mechanisms available is also a very effective defense. By giving priority to certain types of legitimate traffic, this, by definition, limits the capability of bogus

traffic to occupy resources. Of course, each router in the path must support the QoS protocol being used, or else the result is the standard IPv4 best effort delivery method.

Finally, one can utilize multiple ISP connections and IP addresses to help mitigate the effects of a DoS/DDoS attack. For Synflood attacks against a Webserver, a redirect page can be set up to send users to a different server [61] and the DNS entry can be changed for the IP under attack to send the URL to a different IP address. Most attacks will not follow the new DNS mapping to the new IP address, but legitimate users will once the change filters through the DNS system [61]. Multiple ISP connections may also provide a path for legitimate users if another ISP is getting hit with a DoS attack. Although more complex and not infallible, these techniques offer another option in the face of a packet or connection flood. Unfortunately, the increased cost and complexity may be prohibitive, and sophisticated attackers may follow legitimate users to the new IP address that successfully counters the countermeasure.

To find the source of the attacking packets, manual traceback of packets to the true source is possible; but automatic methods are sorely needed. If an IP address is spoofed, the lack of authentication methods in the IPv4 protocol makes the true source difficult to determine. IP traceback is easy in theory, but tracing the packets through the various domains requires help from each administrative contact in order to get the interface of the previous hop; a time consuming process that must be completed while the attack is in progress. In addition, in a DDoS, there may be hundreds, if not thousands, of sources of these packets. They were also compromised by the original attacker without the local systems administrator's knowledge or permission. Following are some of the various countermeasures currently being deployed to prevent a network from becoming a site that hosts these zombie subroutines.

One of the more popular countermeasures involves running Network-based and Host-based Intrusion Detection Systems (NIDS and HIDS) software to detect any file alterations. Assuming that the attack has been seen before and the signature categorized, many, if not most, of these IDS tools now look for evidence of DDoS daemons on the LAN or on the individual hosts. *Tripwire* is a good example of a program for detecting altered files. There is an added cost; not just the initial purchase price, but for periodic upgrades and for the manpower needed to examine the logs produced by these products. One thing that should be

done on all networks is to shut off traffic and protocols that are not needed by the local network. This can limit the ability of DDoS masters to communicate with the daemons, or agents, under their control. Network administrators need to be very suspicious of ICMP and UDP traffic, particularly if the traffic is unsolicited. This traffic should either be shut off completely, or limited to legitimate responses and inquiries that originate from the LAN. Such measures result in increased complexity and the possible interference with legitimate network traffic.

As an ongoing countermeasure, it is important to stay updated with the latest software; this keeps the software current and patches holes that can be communicated to the hacker community. This countermeasure is not effective against the first target of an exploit, but will help protect potential victims after the attack has been seen, reported, and a vendor patch is installed. Of course, this adds the cost of constant vigilance to the network administrator's workload.

Limiting the rate at which certain traffic can exit the subnetwork and do egress filtering on outgoing traffic will keep the outgoing traffic from flooding other networks; which, in turn, makes the LAN more secure. This may require router upgrades, increasing the cost, and may lead to possible interference with legitimate traffic flow, but it makes the subnetwork a much less desirable place to plant Zombie subroutines. In addition, file-monitoring programs such as *Tripwire* can tell if new files have been added or old files modified. These programs can easily detect Trojan Horse-type programs added to a host; but like other software add-ons, add cost, complexity, and daily monitoring to the administrator's workload. Additional ideas for future countermeasures are included in the next section.

3.4. Future Techniques for Countering Attacks that are Deliberate in Nature

These subsections outline the techniques that are under development for countering the attacks and the attack categories, as detailed previously. They contain both general and specific countermeasure techniques for all six of the mechanism categories defined in the schema. There is hope that techniques currently under development can help alleviate certain classes of vulnerabilities, including the *Poor Authentication and Access Control* and the

DDoS problem, but only time will tell. There are six such promising techniques referenced in the literature, and this research effort proposes two other techniques.

1. IPv6, via IPSec
2. Centertrack, a way to route suspicious traffic that determines the ingress point
3&4. Two methods of IP traceback, both of which add ink marks to a traffic stream
5. Host identity payload (HIP)
6. The Intrusion Protection System, an outgrowth of IDS
7. A proposed *postmarking* method
8. The integration of IDS/IPS techniques with automatic traceback and reporting

3.4.1. Countering Buffer Overflow, Poor Authentication, and IP Fragment Attacks

3.4.1.1. Countermeasures for Buffer Overflows. It will be difficult for border devices such as routers and firewalls to utilize the same techniques employed in commercial products such as Stackguard and PointGuard, because these techniques require specific knowledge of the operating system of the receiving device. Therefore, techniques will have to be developed and deployed on each individual server that operates on a public network such as the Internet. Perhaps the stateful inspection process will become intelligent enough to recognize buffer overflow attempts, but unfortunately that day is not near. For now, each server being protected should have these special software packages installed to protect against buffer overflow attacks.

3.4.1.2. Countermeasures for Poor Authentication DoS Attacks Using IPv6. The ultimate implementation of IPv6, which uses IPSec, holds some promise for the ease of authentication and authorization for TCP/IP connections between two parties that wish to authenticate to each other [77-95]. Web-based transactions such as Internet banking, electronic commerce and communication between business partners will no doubt take advantage of these extra features [96-105], however, this will involve the knowledge and permission of all parties in the communication. Within IPv6, there is the capability for packet authentication to identify the source of the packet down to the microprocessor serial number

or Ethernet/machine address. Full details of these features are included in several references [78-81, 90, 91, 95]. IPv6 should also help to eliminate some of the problems in the *Poor Authentication and Access Control* category of vulnerabilities. Care must be taken to ensure the intense computational requirements of authentication do not result in a DoS within themselves. Historically, it has been easy to twist a complex authentication scheme into a process that resulted in more trouble than it was worth.

Following the massive protest that arose after the enhanced features of the new Intel chip became public knowledge, it is unlikely that the full capabilities of these authentication methods will be implemented on the public Internet. This means that it still will be possible to surf the Web anonymously, and that the problem of IP Spoofing will not be completely resolved. IPv6 will only help the authentication and authorization between willing parties, and the malicious user will probably still be able to hide his/her identity on the Web [92].

Even after IPv6 is widely used, attacks that rely on a false or fake source addresses will still be possible, however, the additional features of source identification and the QoS features of IPv6 will offer some assistance in determining packet origin. This will offer increased levels of QoS priority if the traffic is valid and the user is authorized, over and above the normal net traffic noise. Of course, authentication would be attempted only between willing parties. With this limitation, it appears that IPv6 will not be a magic bullet to stop DoS and DDoS attacks.

3.4.1.3. Countermeasures for IP Fragmentation and Other *Wrong Data* Attacks.

With the previous section's assumption that IPv6 will not be universally utilized to authenticate communicating parties, all forms of DoS attacks are far more likely to occur; particularly buffer overflow and IP fragmentation-style attacks. This leaves either Intrusion Detection Systems, individual TCP/IP stack kernels, or firewalls to recognize such attempts and deny the offending packets further travel. Reliance on any of these is likely to be a futile exercise, because keeping the device's software up-to-date with patches and updates requires constant vigilance on the part of the computer or network owner. The section on Intrusion Detection Systems (IDS) discusses this arms race in greater detail.

3.4.2. Countermeasure Considerations for Flooding Attacks

The prospects for effective countermeasures for flooding-style attacks is slightly better, but mainly from the standpoint of being able to trace the source or sources of these attacks. With the current protocols in place, there is little that can be done to stop a flooding attack except to drop the malicious packets once they are recognized. Practically every technology under consideration is in the realm of ability to recognize and trace the source of flooding attacks instead of preventing them, as detailed below. The main complication is that there are two competing protocols for tracing IP packet streams being proposed by two different, but influential, groups. The lone exception to these packet tracing schemes is the proposed HIP (Host Identity Payload) protocol, detailed in section 3.4.2.1.

3.4.2.1. Future Countermeasures for DoS Attacks that Overwhelm with Data or Service Requests. The techniques detailed below represent the most promising prospects in terms of countering flooding-style attacks. Most techniques simplify the effort necessary to trace the source or sources of DoS and DDoS attacks, while others such as the HIP protocol attempt to shift the workload from the receiving server to the requesting party. Tracing techniques such as Centertrack, two methods of IP Tracing called traceback and itrace, and Postmarking are discussed first, followed by the HIP protocol.

3.4.2.1.1. Centertrack. The previous section on current countermeasures mentioned the concept of allowing limited ability of a victim network to dynamically reconfigure upstream routers, including routers at ISPs or in other domains of control, to prevent a DDoS or DoS victim from having to suffer through a barrage of bogus network traffic. One method under current consideration has a slightly different approach. Robert Stone (1999) of UUNET [106] has proposed a concept called CenterTrack that employs an overlaying network of tracking routers that examine and trace suspicious traffic at the borders of an ISP's domain. These tracking routers, poised only at the edge of the domain, reroute suspicious traffic from the edge routers and can easily determine the ingress router by observing the exact tunnel in which the packets arrive. This concept works well at the border of a large domain, such as a large ISP, but expanding the concept to the entire Internet would

be very complex. This method was introduced at the October 1999 NANOG meeting, and the paper was presented again at the August 2000 USENIX conference in Denver.

3.4.2.1.2. IP Packet Traceback. Various methods to allow IP packet traceback are being considered. This section details two competing methods that have influential groups within the IETF backing them. One scheme [107] would add routing information to suspicious packet streams to allow the recipient to diagnose the source of the packet stream. This occurs by adding router IP addresses to random packets destined for a singular IP address, each one having a randomly selected router's IP address placed in the packet. Once the packets are inspected at the receiving device, the various router IP addresses can be recovered and the exact path through the network determined.

Another traceback method, called *itrace* [108], recognizes the signs of a packet flooding attack, and then adds packets to the stream with information on the various routers traversed. This allows the exact route to be recreated at the receiving end. To reduce router overhead and workload, it is important not to add too many additional packets to the datastream. Probabilistic methods should be used to determine the rate at which additional packets are generated to mark the data, and create the minimum number of packets to allow the receiving end to recreate the path via router IP addresses and timestamps. Clearly, some means involving authentication should be developed to prevent attackers from sending spoofed packets and confusing the results. This technique, as well as the one described previously [107], show great promise and both are being considered by the IETF community. It remains to be seen which method will gain momentum and achieve the critical mass necessary to become accepted in the marketplace.

3.4.2.1.3. Postmarking. Another method considered by ISU researchers, that is not seen elsewhere in the literature, requires a basic understanding of the OSI network levels. In the OSI stack, there are seven levels of abstraction, starting at level one, the physical link or wire level, and progressing up to layer seven, the application level. The next level above level one is level two, or the datalink level. In most commercial networks, this is usually Ethernet, ATM, or the Frame Relay protocols. Above level two is the networking level, or IP in most

commercial networks, including the Internet. The software modifications include provisions that if a router is the device that converts network level-two data into level three data, then an additional piece of information is added to the IP datagram's header called a *postmark*. Similar in concept to the postmark that the U.S. Postal service stamps on each piece of mail that enters its network, this postmark is a 32-bit word that identifies the network entry point by the router's IP address. As long as no one is able to replace or modify this additional data without detection, the postmark identifies the entry point of the packet by specific IP address. The most complex part of this scheme is ensuring that this postmark is added only at the conversion point from layer two to layer three by the router performing the conversion, or else this piece of data can be easily replaced or spoofed by an attacker, eliminating its usefulness. The most stringent requirement of this countermeasure is that it should be difficult (ideally impossible) for the attacker to use this technique to gain an undeserved status of authenticity. Perhaps the second most important aspect is to stay compatible with existing hardware or software; in other words, the improvement must cause no harm.

The additional information that would be added to the IP header is shown in Figure 15. Essentially, an extra 32 bits is added to the header to identify the router, by IP address, that serves as the entry point for the packet as it transitioned from the layer two (Ethernet) level to the layer three (IP) layer.

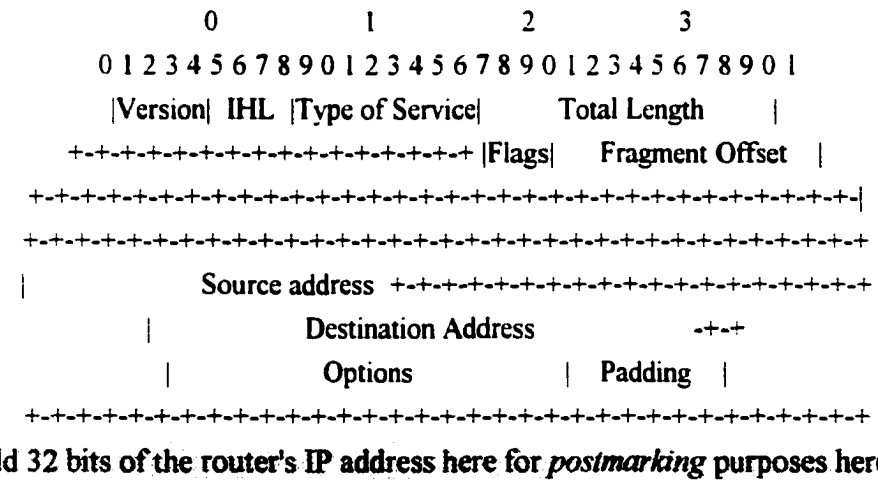


Figure 15. Example of an Internet datagram header (Tick marks represent bit positions)

Several protocol and routing network changes have the most potential to be used as countermeasures against DoS and DDoS attacks. It is difficult to predict at this time which ones will get fully developed and fielded. Certainly, one should follow the progress of these techniques as they progress down the standards track.

3.4.2.1.4. Host Identity Payload (HIP). Many applications used across the Internet utilize the client/server model, which currently places more workload on the server than it does the client, making DoS attacks easier. One proposed countermeasure involves changing the workload balance of the client/server relationship to shift the workload more toward the requesting party (i.e., the client in this model), and shifting the workload away from the receiving party, typically a Web-based server.

This new protocol [109] changes the three-way handshake to a four-way handshake, which not only makes more work for the requestor but also invokes a short-lived key exchange protocol that mutually authenticates the two parties. Like all authentication schemes, care must be used to prevent the authentication process from becoming a DoS in itself. The whole idea is that the initial stages of the four-way handshake are quickly and easily accomplished, with the completion of each stage earning more and more trust between the server and the initiating party until both are satisfied as to the legitimacy of the request.

Quick key exchange and timely authentication methods are important for this protocol to be useful, but this adds complexity. Like many techniques, the complexity of the algorithm could be used against the victim. Section 3.5.2 discusses the concept of Jiu-jitsu attacks and on DoS efficiencies in general.

3.4.3. Integrated Responses to Complex Attacks

The previous sections have shown how difficult and potentially expensive it is to successfully counter DoS and DDoS attacks; all previously discussed techniques rely on efforts to build and deploy stand-alone solutions to mitigate the effects of these attacks. The processing power and PKI authentication schemes available today make it possible to envision more integrated responses, responses that can be coordinated across LAN boundaries and across equipment types. The subsequent sections discuss these possibilities.

3.4.3.1. The Intrusion Protection System, an Outgrowth of IDS. Several papers propose Intrusion Detection Systems (IDS) that take a more complex role than simple intrusion detection. The present generation IDS products offer a choice of Host-based IDS (HIDS) or Network-based IDS (NIDS). Both types tend to be reactive in nature, but are currently most useful for providing a warning that something is occurring and providing forensic data after an attack; nevertheless, few offer any proactive measures to prevent attacks. Each style of IDS has its advantages, but an HIDS is preferable to take a more active role in actually preventing attacks such as buffer-overflow attacks. This stems from the inherent weaknesses of the NIDS approach; namely, it is difficult for an NIDS to identify an attack at the network level, and even more difficult to stop one. HIDS are installed on every machine in the network, and can use the machines OS to track all the service requests being handled by the host.

By watching the service requests, any attempt to access system resources is detected, and dangerous actions can be stopped. This converts the Intrusion Detection System into a more proactive Intrusion Prevention System, or IPS [110]. Using this approach, precise actions can be taken against an attacker, thus stopping the effects before they have a chance to compromise the host. Such an IPS offers protection even for buffer-overflow attacks by intercepting the exploit code before it has a chance to be run on the machine. At least one company is already marketing such a system and, if successful, it is expected that more companies will follow suit. This does little for DoS or DDoS, except provide a warning, but some IDS systems will eventually take more proactive measures if programmed to do so. Some systems can dynamically reconfigure the firewall to block objectionable packets, and they can also automatically notify the upstream network provider.

Another approach recently suggested [111] uses a network of interlocking subnetworks whereby the IDS on each subnetwork tracks intrusions across member domains in a cooperative fashion, and takes actions to prevent attacks not only at the local level but also at the group or neighborhood level. If all subnetworks are eventually joined in this manner, suspicious activities can be traced back to their origin; or, if this fails, the system can at least notify human administrators and/or take corrective action on its own. This action

could terminate suspicious activities or reconfigure the firewalls to block damaging packet floods. The system, called IDIP (Infrastructure for Intrusion Detection and Response), is an application level program that provides automated intrusion response across multiple, but related and cooperating domains.

Several vendors are participating in this research effort. Network Associates (NAI) is supplying the firewall and *Cybercop* IDS, Linux is supplying routers, SCC is supplying another firewall type, and ISS is supplying their IDS, as well. A few smaller companies are donating their specialized software and other assets into a true cooperative venture.

The obvious limitation of this scheme is the scalability—whether enough vendors can be convinced to make their products IDIP-compatible and whether users will trust and use the full capabilities of the system remains to be seen. All of the intervening domains must be a part of this IDIP network in order to have the capability to trace an intruder from end to end, a limitation that has defeated previous prevention efforts of this type.

Similar schemes are proposed in [112-114], each involving some special hardware or software capability to authenticate the traffic that the network receives using IPSec security associations [112], active network nodes [113], or protects routing infrastructures using IDS approaches [114]. Each proposal requires widespread, if not universal, distribution onto special hardware or deploys special software, but it is presently not clear if any of these techniques will gain acceptance.

Another example of a more proactive IDS system is currently in the beta-test phase at CERT. It is a system that uses an open-source IDS system called *Snort*. The CERT organization has worked with this IDS vendor to incorporate automatic notification not only to the local administrator when an attack is detected, but also to alert CERT electronically. This IDS allows more rapid and accurate reporting of the problem by using electronic means instead of human interfaces. This project, or technique, is called AirCERT.

3.4.3.2. Router Control via SNMP Using IDS, AirCERT, and Unicast Reverse Path Forwarding Mechanisms. Router control might be accomplished by using various tracking and traceback mechanisms to automate the traceback function to determine the source(s) of bogus traffic streams. Where previous, but manual, traceback methods were

effective against single attackers, an automated approach may be the only hope of finding multiple attackers as in cases involving DDoS attack agents. Such an idea is not farfetched; various pieces of it exist now in a variety of products. This automatic traceback incorporates the heuristic capability of Intrusion Detection Systems, the traceback capability demonstrated by various router products, the traceback schemes under consideration by the IETF, and the reporting capability of the AIRCert technique described in the previous section.

In this proposed scenario, the various router products have the means to determine if a flooding or connection-type flood attacks were in progress by recognizing large amounts of traffic to a particular IP address, or IP address range. This is done either by proprietary software means or by way of the techniques used in modern IDS products. Once a router is aware that a flooding or connection-type attack is passing through its interface, it uses the same techniques utilized by Cisco's Unicast Reverse Path Forwarding to determine which interface the traffic is arriving on, and then determines the next upstream router. An SNMP message is sent to that upstream router, alerting it of the traffic and asking for a similar trace. A separate SNMP message is generated to CERT, or some other central location, alerting them as to the existence of a packet-flooding attack and providing details such as the router's own IP address, the timestamp, the destination of the packet flood, etc. This automated reporting employs the same concept used in AirCERT, a joining of the IDS and CERT alerting functions.

When the upstream router receives the SNMP warning from the downstream router, or perhaps detects the flooding attack on its own, the process starts anew, tracing the source or sources of these packets to the edge of the network. If additional routers are not responding with SNMP packets being transmitted to CERT, it can be assumed that the edge boundary has been reached. Nevertheless, in the early stages of this system's deployment, there will be numerous edges because few of the routers in the path will be capable of this function. However, as deployment spreads there will be fewer holes and better coverage. The various techniques described thus far exist in current products, in isolation, and this proposal merely merges them into one process that relies on SNMP v3 messages to carry data and

commands between the various routing elements and report malicious activity to a third party monitoring site such as CERT.

Figure 16 depicts a greatly-simplified model of the Internet, showing the path taken by a flooding attack that emerges from a connection to the edge router A, and progresses through transit routers D, F and P, arriving at edge router Z, where the traffic finally enters a company LAN to saturate a victim host. The automatic traceback mechanism described in the previous section relies on the ability of various routers to detect the flooding attack, either on their own or by virtue of a connecting router that alerts them to the condition, and then sends the interface information onward to other routers and locations.

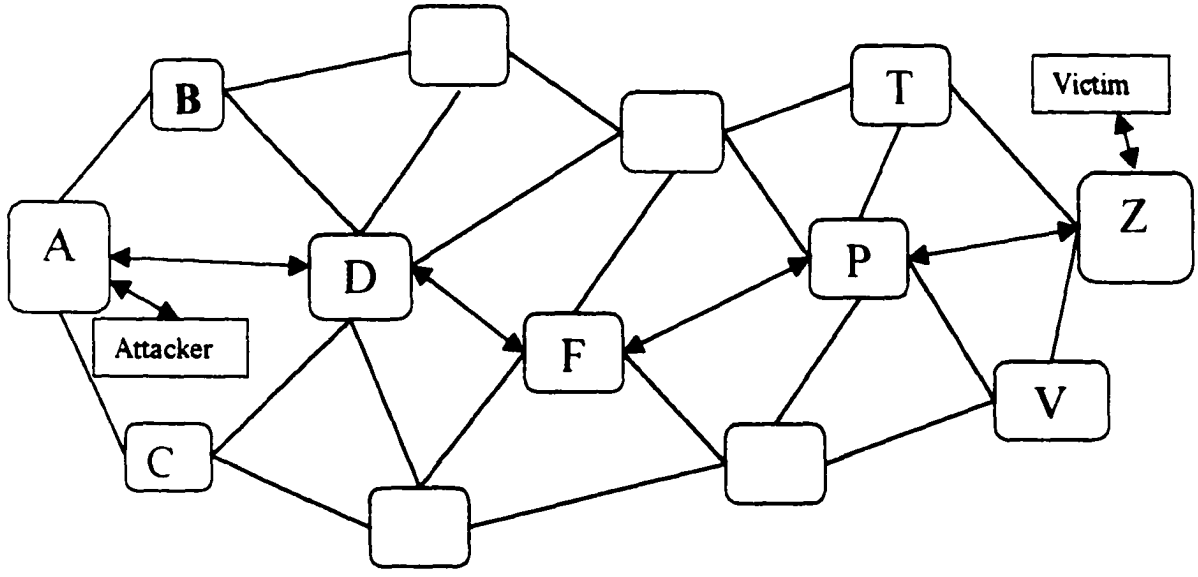


Figure 16. Attack and traceback paths from attacker to victim

In the case of the proposed traceback methods of Savage (2000) [107] and Bellovin (2000) [108], the timestamp information is sent to the victim to aid forensic efforts to locate the originating router. This is done either by inserting additional information into selected packets or by generating additional packets and sending them on to the victim. The algorithms need to insert the data on a fairly low frequency basis so as not to add to the packet flooding problem. In the model of the AIRCert scheme, traffic information is sent to an interested third party, such as the CERT group, for analysis.

In the proposed technique described in the previous section, newly-defined SNMP messages send information about the flooding attack up and down the attack chain, asking each router to identify the previous router of origin until the originating edge router is determined. Although time-consuming and tedious for human operators, network routers could easily be programmed to perform such techniques; nevertheless, care must be taken to avoid adding an inordinate amount of network traffic that will contribute to the router workload.

In a simple DoS attack, with one attacker and one victim, the traceback process can identify the one specific path that most packets will take through the Internet from edge router A to edge router Z. Since there is only one attacker, it is anticipated that the flooding attack will not cause much of a problem as to disrupt the traffic flow such that packets are routed around this congestion, and the routers will have the bandwidth to absorb the flooding traffic.

In a DDoS attack, edge router Z is likely to see flooding traffic on all three of the interfaces shown, since traffic will be originating from its interfaces with P, T, and V. Reporting this fact is not likely to be of much interest, but tracing backwards to the point of being able to detect and locate an edge router will be of great interest. From this point, the ISP being used as the on ramp should be able to identify the corporate LAN, College campus, or DSL/cable modem connection being used as a zombie packet generator.

In addition to this traceback method, elements of the other proposed schemes could be implemented, as well, sending pertinent data both to the victim's IP address and to CERT, if desired, to aid in the forensics effort, whether it is after-the-fact or in real-time. This requires an IETF group to be tasked with determining the format of the SNMP messages, the method by which a flooding attack is determined, whether the messages are sent to adjacent routers, the victim host(s), and a central CERT monitoring site, as well as knowing how an edge router is pinpointed.

At this point, the only accomplishment is to have found other victims, such as compromised hosts on other networks, but this information can be used to locate the master-controller computers and, perhaps, the intruders that initially set up the DDoS network. This automatic traceback function merely automates a process that is presently performed

manually, with great resource demands, and is likely to produce a long list of originating edge routers and/or originating hosts for later analysis. These would be the zombie, or soldier, units originally compromised by the intruder. Nevertheless, it is hoped that firewall logs or other LAN devices could be used to locate the master controllers once the zombie hosts are identified. In addition, owners of these compromised hosts should be informed so that they can start regaining control of their own machines.

Key concepts that must to be employed to make this automatic traceback functional are: 1) keeping the processing requirements of this analysis to a minimum; 2) keeping the additional packets generated to a minimum; and 3) having ironclad and spoof-proof authentication in place so that the attackers cannot flood the network with false information and negate the valid forensics information. This necessitates a foolproof method of authentication, which would probably rely on some method of digital certificate use, typically referred to as PKI, or Public Key Infrastructure. Bellovin (2000) [108] admits this is a difficult problem that he will not broach.

3.5. The Attacker versus Victim Imbalance

Denial of Service attacks are likely in any venue where it is possible to cause a victim a great deal of work in comparison to the workload required from the attacker. This imbalance, or ratio, becomes an important measure of the effectiveness of a DoS attack. In some of the early DoS attacks, this victim/attacker workload ratio was not very high, and the effectiveness of any attack could be limited if the attacker was connecting to the network via some low-bandwidth connection, such as a phone line. With traditional flooding types of attacks, the rate of the connection requests or echo requests were limited by slow-speed connections, so the resulting resource consumption was also limited. However, more recent DoS and DDoS have a much higher amplification factor, and very little workload on the part of the attacker can result in a tremendous resource drain on the recipient, or victim.

3.5.1. The Amplification Factor

The amplification factor is a ratio between the workload that the victim experiences vis-a-vis the effort that the attacker expends to create the victim workload. Early DoS attacks

were of the nature of a Pingflood or Synflood attack where this ratio was effectively 1:1, meaning that an ICMP echo request packet sent by the attacker resulted in an ICMP echo reply from the victim. Thus it is easy to see that the effectiveness of one attacker on one victim could be relatively minor, especially with a slow Internet connection for the attacker. Unfortunately, more recent attacks have increased this ratio substantially.

The Smurf attack is a prime example. One echo request packet sent to a subnetwork can result in hundreds or thousands of responses. A constant barrage of pings sent via a high-speed Internet connection can easily absorb all the resources of a network pipe, even if the traffic is dropped at the firewall. Current DDoS attacks achieve an even higher amplification factor by bringing multiple attacking sources to bear against a lone victim; a few packets sent by the master controller can set hundreds or thousands of attacking sources loose upon a single victim.

Table 2 shows some typical DoS/DDoS flooding or connection attempt attacks, the date of first appearance, and their corresponding amplification factors. The amplification factors were quite low in the early days of the Internet, but have risen quickly since their modest beginnings. The Distributed Denial Service attacks can be shown to have an amplification factor that approaches infinity, since a few packets from the master controller set loose a packetstorm on the victim from thousands of sources.

3.5.2. Jiu-jitsu DoS Attacks and Effects

By their very nature, networking protocols and computer algorithms can be accidentally or maliciously used to consume resources over and above what it takes in terms of starting the process or protocol. Everyday examples include the unusually long wait-times that typical computers have in terms of waiting for the third piece of the TCP three-way handshake, the Smurf Network DoS attack, and the heavy computational requirements of SSL encryption.

In the early days of the Internet, it was not uncommon for Webservers to have as few as 5-10 slots available for use in the early phases of the TCP connection. This resulted in five, or less half-open connections that were still in the process of getting the third piece of the three-way handshake. This coupled with the unusually long wait times that were allocated

Table 2. A comparison of amplification factors

Date	Name of attack	Amplification factor
3/92	ICMP packetstrom	1:1
2/93	Pingflood	roughly 1:1
2/96	Echo/chargen	infinite; once the attacker sends a chargen to an echo reply, it is a perpetual motion machine until reboot
9/96	Synflood	roughly 1:1, but 1/2 open connections could linger and create an effective 30:1 or 50:1 ratio
1/98	Smurf	100:1 to 1000:1 or more, depending on number of subnetworked hosts
2/99	DDoS	infinite; few packets sent to captured machines unleash an endless barrage

to receive a handshake, typically 75-90 seconds, it is not difficult to see how this situation was easily abused. An attacker merely had to send 5-10 connection requests to a Webserver and then never complete the connection; therefore, the server was unable to handle any other requests from legitimate users. This is known today as a Synflood attack, and it is still one of the hardest attacks to defend against. Synflood is a prime example of the Jiu-jitsu attack; by using the strength of the protocol against itself, a small effort on behalf of the attacker nets a major impact to the victim system or network.

Modern servers take precautions against Synflood abuse; namely, they have greater connection capability by allocating more memory. They are able to drop the open connections more quickly as well as routinely drop older half-open connections in favor of new connection attempts. In addition, other techniques such as load balancing are deployed at larger Websites that send connection requests to a server farm, not just one or two machines. This spreads the traffic over a large number of servers to provide a better chance of servicing all incoming connection requests, some of which may be malicious while many of are legitimate. The difficulty is in distinguishing the bogus/malicious connection from the legitimate one.

The Smurf attack is another example whereby one small action on behalf of the attacker creates a huge workload for the victim. Section 3.3.2.1 details this attack. When such imbalances exist, it is an attacker's paradise.

The last example of Jiu-jitsu attacks involves SSL encryption, which is computationally intensive to the point of almost being a Denial of Service (DoS) attack even under normal operating conditions. The high numbers of bits that are used in the encryption keys consume a large amount of processing power, so much so that normal operation of the Webserver can be adversely impacted by as few as 10-12 simultaneous SSL sessions. SSL can cause the server's CPU to run at nearly full capacity, even with just a few connections.

In a recent performance test [115], a Pentium II, 333 MHz processor running NT and Netscape's server could process 1069 connections per second fetching a Webpage that had a 2493 byte Web object, but could only process 12 such SSL connections per second using 128 bit SSL encryption. At a busy Website, it is easy to imagine a scenario where this would be a problem. Once again, the Jiu-jitsu attack sets up a large number of such connections, causing the server to become overloaded to the point of crashing or to the point of locking out legitimate users.

Developers of future products, processes, protocols, or algorithms must guard against the imbalance of instigation versus computation. If this balance is not considered, new DoS attacks are sure to be developed against the system. The Host Identity Payload (HIP) discussed in the previous section is an attempt to prevent this imbalance. In the hope of preventing DoS attacks, its intent is to shift the workload from the server to the requestor, removing this potential for abuse. It is hoped that Jiu-jitsu attacks can be removed from the networking world in such a manner, and remain in the martial arts training hall where they belong.

3.6. Summary

This section presented and discussed potentially effective countermeasures that are either presently available or should be available in the future. Chapter 4 explores possible arrangements between classes of vulnerabilities and their requisite countermeasures, using the Mechanisms category developed in the taxonomy created in this study as a method of linkage.

4. THE RELATION OF COUNTERMEASURES TO EXPLOITS

This chapter details the mapping of currently available countermeasures to exploit classes, best current practices, and the mapping of future countermeasures to the presently defined exploit classes. The current trends of increased authentication and layered access to networks are also discussed.

4.1. Mapping Currently Available Countermeasures to Exploit Classes

One of goals of creating this database was to group exploits and the underlying vulnerabilities in such a manner that groups of countermeasures might be deployed against these exploit categories. In other words, for a given category of vulnerability, there might well be specific categories of countermeasures that could be applied to help alleviate the problem or, at least, mitigate the effects of an attack against these vulnerabilities.

Toward this end, specific countermeasures that are effective against certain vulnerability categories have been identified, and listed in a table using the *Mechanisms* vulnerability category as a guiding parameter (see table 3). The degree of effectiveness is variable, but it is clear that there are at least some measures that can, and should, be taken against all vulnerability categories. Table 3 and 4 show which countermeasures are effective against any particular style of DoS attack, using the Mechanism category of the taxonomy as a baseline parameter. In addition, there is a subsequent commentary to explain the logic behind the entries in these tables.

Many commercial firewall products contain stateful inspection software that can protect against known IP fragmentation problems, known Incorrect Data problems, and some Buffer Overflow issues. Firewall solutions are not as complete or robust as the anti-virus or IDS solutions, primarily because they are not updated nearly so often. Nevertheless firewall solutions help.

The load balancing solution, as shown in Table 3, is only effective against the types of attacks that overwhelm the victim with service requests, and is usually implemented in the form of a load balancing device feeding a Webserver farm. Bandwidth control, QoS mechanisms, and dynamic router reconfiguration are effective against both data flooding and

connection requests, but use different techniques to do so. The IDS/IPS products shown as the last subject in Table 3 are quite effective if kept up-to-date.

Much like the situation in the anti-virus world, the application software must be kept current if all known attacks are to be defended against. Of course, this model means that any new attack that is different enough to go undetected by the IDS/IPS software is going to be uncategorized and, therefore, a surprise to the victim until the vendor releases a patch to identify and thwart the new threat. It seems that the anti-virus and IDS worlds are in a constant arms race with attackers, with anti-virus vendors constantly lagging behind the most recently discovered vulnerabilities.

The Backtracking/tracing column in Table 3 is there as a countermeasure only in terms of being able to locate and cut off the source of the attacking packets or connections. It does not stop the attack, but it is hoped that the threat of discovery acts as a deterrent to prevent the attacks or to limit attacks to a short duration.

Table 3. Current countermeasure matrix for vulnerability categories

	column 1	column 2	column 3	column 4	column 5	column 6	column 7
	Stateful Inspection Firewalls	Load Balancing	Band-width Control	Other QoS Mech	Dynamic Router reconfig	Back-Tracking/ Tracing	IDS/ IPS
Overwhelm w/data			X	X	X	X	X
Overwhelm w/ Service request		X	X	X	X	X	X
Buffer Overflow	X	allied with code review/code overhaul and vigilance					X
IP Fragmentation	X						X
"incorrect data"	X						X
For poor authentication/access, it is possible to use IPSec protocol if both parties participate, but is otherwise dependent on the techniques invoked by the application to allow access. The application could be a proxy application to prevent direct access.							

4.2. Best Practices for Current Countermeasures

If at all possible, most systems administrators prefer to use the proxy model when it is necessary to permit an outside party access an interior network or to protected assets within an interior network. Web-based applications are well suited for this, and by placing Webservers outside the protected domainspace and then having these servers make calls to protected databases inside the firewall, critical data can be protected. This works well for Web-based, or HTTP applications; but for other types of access it is best to invoke proxy servers to act on behalf of the outside party. Once the requesting party is sufficiently authenticated and authorized to obtain the requested data, these proxy servers will fetch and return the data to the requesting party. In this manner, users coming in from outside the firewall never really have direct access to the interior network; the proxy acts on their behalf. A limitation of this model is that proxy services are not available for every application needed, thus direct access is sometimes unavoidable. In these cases, it is necessary to implement this strong authentication and authorization on the users who seek access, and to audit the actions taken while inside the network. Strong authentication is defined as having at least two separate forms of authentication, typically implemented as *something one has*, such as a token ID or digital certificate, and *something one knows*, such as a password.

Considering the exploit population, there is good reason that Poor Authentication or Access Control is such a popular attack category; if people feel they are anonymous, then they are emboldened to try things that they would not ordinarily attempt. The anonymity offered by the Internet allows malicious users to try to gain access to systems and networks that they normally would ignore if authentication and authorization were required.

For this reason, products are now being developed that include an authentication method that is exercised very early in the communication process. Financial systems at banks are a prime example. A visitor to a bank or financial Website cannot get very far into that system without presenting authentication credentials, or the communication is halted. Access to financial, medical, or personal data simply should not be allowed without valid authentication.

Currently, products are being developed that enter this authentication data automatically, but they are not yet widely deployed. Netegrity markets a product called

Siteminder that actually intercepts the HTTP request for a Webpage, and then determines if that Webpage is a protected asset. If it is protected, it challenges the user for authentication. The actual authentication method can be one of several choices. The authentication can be accomplished using usernames and passwords, hardware or software tokens, or digital certificates; and future growth will most likely include biometric methods.

Products such as Siteminder also have a self-registration function that gathers a small amount of data on the user, including the user's e-mail address, and then e-mails a username/password combination to a new visitor to allow for further access. This represents more work for the visitor, of course, but removes some of the anonymity from typical client/server communications, since the users must supply a valid email address, one that belongs to them, in order to gain their username and password. However, since these credentials are necessary to access protected Webpages, most users will go forward with this process if they deem it to be a worthwhile exercise. Such schemes are almost certain to be used in the healthcare, financial, and insurance markets.

Much like the proposed Host Identity Payload protocol, the idea is to put more work onto the requesting party, and remove some of the presently allowed anonymity from Web-based transactions. However, this additional workload must not be onerous, or users will not bother to use it. Products such as Siteminder do a very good job of authenticating a new visitor in the early phases of communication, and then not challenging them for credentials again for the remainder of the session. Future products will seek even more secure and less burdensome methods of authentication.

As another common practice, users on the inside of the network never really have to directly access Webpages on the outside Internet; an HTTP proxy server fetches the Webpage for them and delivers it to their browsers. In this manner, Webservers do not have a chance to place cookies directly on users' browsers, nor do they have a chance to obtain the users' IP address, so anonymity is preserved and direct access is prohibited.

Most organizations that have data within their domainspace to protect utilize several countermeasure techniques to adequately protect the data from malicious users. As a protection against known Incorrect Data, IP Fragmentation, and Buffer Overflow attacks, most find that the stateful inspection firewall of column 1 is useful. In addition, most

industry-standard IDS software protect against known attacks, and work hard to keep abreast of new exploits (Column 7). For the attacks that seek to overwhelm the victim with either data or service requests, at least one of the techniques listed in columns 2-4 are implemented, and backtracking/tracing (column 6) is sometimes attempted, depending on the disposition of the systems administrator. Although column 5 is currently possible, it is rarely attempted due to the large amounts of false positive indications from current IDS products. As a minimum, a product from column 1 and column 7, plus one product from either columns 2-4 are implemented along with applicable proxy servers as a total protection package.

Recently, many organizations have been implementing another best practice method of protection, and that is of a "layered" approach to security and access. Early, in the days when organizations were first connecting to the Internet, a single firewall would typically protect a domainspace. This firewall would demarcate the external (public) Internet from the internal (private) Intranet, creating a very digital situation. A user was either in or out. As the sophistication of attacks increased, this model has been improved upon by adding multiple layers of protection, defining various levels access by outside parties, with ever increasing sophistication of authentication and authorization allowing access to data of increased sensitivity. Using this model, an organization is better prepared to detect attackers in the outer layers and stop them before they can penetrate too far into the network. With the previous model, once the attackers successfully conquered the first firewall they were inside the network, free to access or destroy any data they wished. As a minimum, most organizations are deploying a border router at the edge of their domainspace and then an application proxy or stateful inspection firewall farther inside their network that protects their internal Intranet. The space between the border router and the main firewall is sometimes called "the DMZ", and the external-facing Web, FTP, and DNS servers are typically placed here. This should be considered a minimal configuration; extra layers defining specific Extranets or Intranets may be needed to provide sufficient layers of protection.

4.3. Mapping Future Countermeasures to Exploit Classes

Table 4 shows the various products or techniques under development, and the likely effectiveness against the six mechanism categories that can be used for vulnerability classification.

Table 4. Future Countermeasure Matrix for given categories of vulnerability

	column 1	column 2	column 3	column 4	column 5	column 6
	Ipv6 Authentication	Centertrack	other IP traceback methods	Host Identity Payload	Post-marking	IPS + traceback
Overwhelm w/ data	?	X	X		X	X
Overwhelm w/ Service requests	?	X	X	X	X	X
Buffer Overflow	?			X		?
IP Fragmentation	?			X		?
Other "incorrect data" attacks	?			X		?
Poor authentication/ Access	X			X		?

Although most of these techniques will not replace existing ones, in most cases they will make the goals of the current techniques easier to achieve. It is hoped that IPv6, shown in column 1, will be a boon to authentication. Columns 2, 3, 5 and 6 simplify the goal of tracing attack sources, while column 4 reduces the ability to carry out the overwhelm with service request style of attacks by shifting most of the workload to the requesting party, not the party that services the request. This Host Identity Payload (HIP) protocol should eliminate the Jiu Jitsu style of attacks that are so prevalent today, while the improved tracing methods reduce the Overwhelm with Data style of attack.

Question marks in the column for IPv6 authentication are there simply because it is unknown, at this time, how widespread the authentication mechanisms will become. As mentioned earlier, these authentication mechanisms are optional, and many applications are not likely to require them. There is a solid X in the row for Poor Authentication or Access Check because it is assumed that if authentication is desired, the protocol will be used.

The Centertrack, traceback, and Postmarking schemes will be effective against the *Overwhelm* attacks, which use data flooding or connection flooding. These proposed

techniques are superior to current methods because there is no requirement for the flood to be in progress while the traceback occurs; the identity of the source edge router can be determined post mortem.

The Host Identity payload, or HIP, is effective against so many styles of attacks simply because it shifts the workload burden from the recipient, which is typically a server of some sort, to the connecting device, a client somewhere. It can do nothing if a massive flood of packets comes toward the receiving IP address, but should be effective against all others, assuming that no Jiu Jitsu attacks are developed against its connection protocol.

The IPS + traceback scheme described in this research is effective only against the flooding style of attacks, although it is recognized that a true IPS system can do much more, as noted in Table 3. Whether or not IDS/IPS vendors can develop systems that interoperate with each other will determine the effectiveness of these products across domain boundaries.

4.4. Research Summary

Until the categories described in this paper were fully developed, entering the data from an exploit posting was as much an art as a science. Some items were straightforward, such as date of posting and the OS involved, but other items took a great deal of thought. After the first year, the database categories stabilized sufficiently and this categorization effort became much more consistent, mainly due to the shorthand tool phrase developed that inserted nouns and verbs from the Purdue categories of the same name. Referring to Appendix A, it can be seen that the words in the various columns can be selected such that a short English-language phrase is formed that does a fair job of describing the exploit or vulnerability. This phrase is listed in several places, but goes, "The <4.3.1> is <4.3.2> by way of <4.3.3> via <4.3.4>". This made the categorization effort far more repeatable than it had been.

As the database and the taxonomy were being built, a great deal of time was spent analyzing the data for plots that were meaningful and useful. Looking at the data from a variety of ways was both challenging and time-consuming, but the Mineset program proved very useful for analysis. The plots were examined in order to determine the best parameters to plot in order to get strings or clusters, and then derive further categories in which to place the data.

Such is the case with the Mechanisms category. This category was developed to simplify the Purdue category called *Methods*, which was felt to be both cumbersome and vague. This Mechanism category was meant to answer the How question, and in conjunction with the Purdue Object_affected and Effect_on_object categories, was an effective category to use to find clusters on three-dimensional plots. The clusters shown in three-dimensional plots such as those shown in Figure 13 and 14 demonstrate that the various exploits can be binned into meaningful categories.

The next step, also quite time-consuming, was to research all of the known countermeasures being deployed, as well as the ones still under development, in order to determine if these countermeasures could be grouped by the items in the Mechanism category. The previous sections testify to this effort, and it appears that the fit is quite solid.

One of the early goals was to see if a model could be developed that displayed how various attack methods ebbed and flowed in popularity in the attack community, much like a Lava lamp of the 1970s; but this has not been proven with the data available. What gets input into the database are the initial postings of an exploit or of a vulnerability; there is no hint as to how often these attacks are attempted on actual victims. The Smurf attack, for example, is probably utilized hundreds or thousands of times a day, yet it may access just one entry into the database. It is much like having data for the first derivative of a mathematical equation, which leads into insights of the original equation. However. without the quantity factor being known, one must insert a "plus constant" into the formula that represents the integration of the first derivative formula, just to cover the unknowns. Many people in the Computer Security field feel that certain types of exploits come in and out of vogue, much like the lava lamp. Nevertheless, without temporal information such a model cannot be proven or demonstrated.

The process in place and the categorization scheme utilized for these 630 database entries is now robust and repeatable, and the categories developed have been proven effective in determining classes of countermeasures that can be deployed against the various attacks, as shown in Table 3 and 4. To this end, it is reasonable to conclude that the research undertaking has been a great success.

5. PROSPECTS FOR FUTURE STUDY

This study resulted in the construction of a database of more than 630 entries that span three years and created a categorization scheme that, after considerable refining, has proven to be a suitable and consistent method for categorizing the underlying vulnerability that makes an exploit possible. The categories developed can be as detailed as necessary, from many bins to few bins, depending on researcher needs. It combines aspects of several different schemes, most of which have their roots in earlier research studies.

With training and calibration, a person familiar with computer security issues can select an exploit posted at *Security Focus* (www.securityfocus.com), *Rootshell* (www.rootshell.com) or *Safer Magazine* (www.safermag.org) and have the item fully described and entered into the database in a matter of a few minutes. One of the early premises of this research was that gathering a long-term history of these exploits could be of great value in preventing newly released software from suffering similar attacks. The classes of countermeasures identified include not only presently available products and techniques, but also ideas that could be considered for future use.

Unlike the situation in the aviation industry, many people feel that the relative obscurity of published computer security faults and oversight methods contribute to the problem of newly released software having the same flaws that have plagued the industry for many years. When a commercial aircraft crashes, teams of investigators and media people converge at the accident site to painstakingly recreate the sequence of events that caused the accident. The results are then published and government agencies develop policies to prevent future occurrences. The computer software industry lacks a similar function today, and it is unlikely that one will be developed in the near future. It is only when customers demand a truly secure product that these flaws, which have been propagated over and over through the years, might get corrected.

APPENDIX A: PURDUE CLASSIFICATION SCHEMA (MODIFIED)

Shorthand Notation: "The <4.3.1> has been <4.3.2> using <4.3.3> via <4.3.4> "

<4.3.1> device	**<4.3.2> object**	**<4.3.3> method**	**<4.3.4>**
CPU: CPU time	**crash(ed)**	**ISSL_brute_Force**	**Netdata**
OS: Operating system	**exhausted**	**incprot (auth/permission)**	Store
Netport: network port	bound	**ISSL_incorr_imp_error (fragmented/offset)**	
Packets: network packets	exported	proxy	
User files	mounted	incorr_imp (environment)	
System files	closed	**special characters**	
System names	**terminated**	**dot .. dot (/../../)**	
User program	executed	configuration error	
System info	replaced	inappropriate capability	
Shell command	**changed**	inherit_priveledges	
Password	**read**	mod_name	
Stack code	appended	back ticks (\\)	
Stack data	created	hidden_mount	
Stack return	**displayed**	verify_fail	
Static data	**predictable**	modifyng_environment	
Public files	**changed_owner**	relative_paths	
System program	changed_permissi on	system_call	
Outfiles	loaded	infinite_loop	
Directory	presented	core_dump	
Partition	debugged	incprot (cgi-bin)	
Heap code	locked	**ISSL_improper_data (buffer overflows & other wrong data)**	
Heap data	cleartexted		
Webpages	not_logged		
Websession			
Email			
Names			
A_net_connections			
Issl_net_services			

Bolded entries denote very popular categories

APPENDIX B THE *MECHANISM* CLASSIFICATION CATEGORY

From plotting the various categories in the database, the most revealing graphs came from plotting the *Mechanism* category.

It is also important to note the complexity and ambiguity of ANY taxonomy; and the *Smurf-like* exploits are a type that demonstrates this perfectly. While the Smurf attack is on that relies on IP spoofing for success, its real damage is in the massive amount of ICMP reply packets that it bombards a victim host with. The database reflects this by creating a combination category of *IP spoofing/overwhelm with data*. And with a Smurf attack, there are actually **two** victims; the network that receives the request to broadcast many ICMP reply packets, and the *return* address where all of these packets are sent; the true victim at a spoofed IP address, not the source of the packet. But here are six preliminary categories of *Mechanism* classification:

1) Buffer overflows
2) IP Fragmentation attacks
3) Other *incorrect data* attacks
4) Overwhelm with Service requests
5) Overwhelm with data
6) Poor authentication or access control (broken down into 4 subcategories)

From these six categories, several general countermeasures can be proposed.

The first three categories are the easiest to deal with; programmers just need to account for (and restrict against) the user attempt to put either the **wrong** data or **too much** data into a form or process. In addition, many Stateful Inspection-type firewalls will screen for many IP Fragmentation attacks and similarly well-known attacks. There is promising work in the area of compiler design and Intrusion Detection Systems (IDS) that may help prevent many types of buffer overflow and incorrect data attacks in the future. Keep software patched and up to date.

For the *Overwhelm with Service Requests* type of attack (Category 4),

- For firewall devices that maintain status of connections, use the *circular buffer* concept where older (stale) connections are dropped in favor of new connection requests- some of which are hoped to be legitimate. Limit the amount of time that a half-open TCP/IP connection can stay in that state.

- Use multiple servers and traffic distributors or load balancers like Cisco's *Local Director* to keep traffic from bombarding a single server to keep bottlenecks from forming and to share the load. Cisco's *TCP Intercept* function insures that the three-way handshake completes before passing the connection to the server. *Quality of Service* (QoS) schemes and bandwidth management and rate limit devices help, as well. Some routers use proprietary methods to traceback suspicious packets. Contact the upstream ISP.

- Future techniques may include *Packet Fingerprinting* to identify (and drop) what is believed to be bogus packets and improved tracing procedures to locate the packet source

For the *Overwhelm with Data* category (Category 5) refers to the Smurf/UDPstorm type of attacks. Same countermeasures as for Category 4 , except for load balancers. But also.

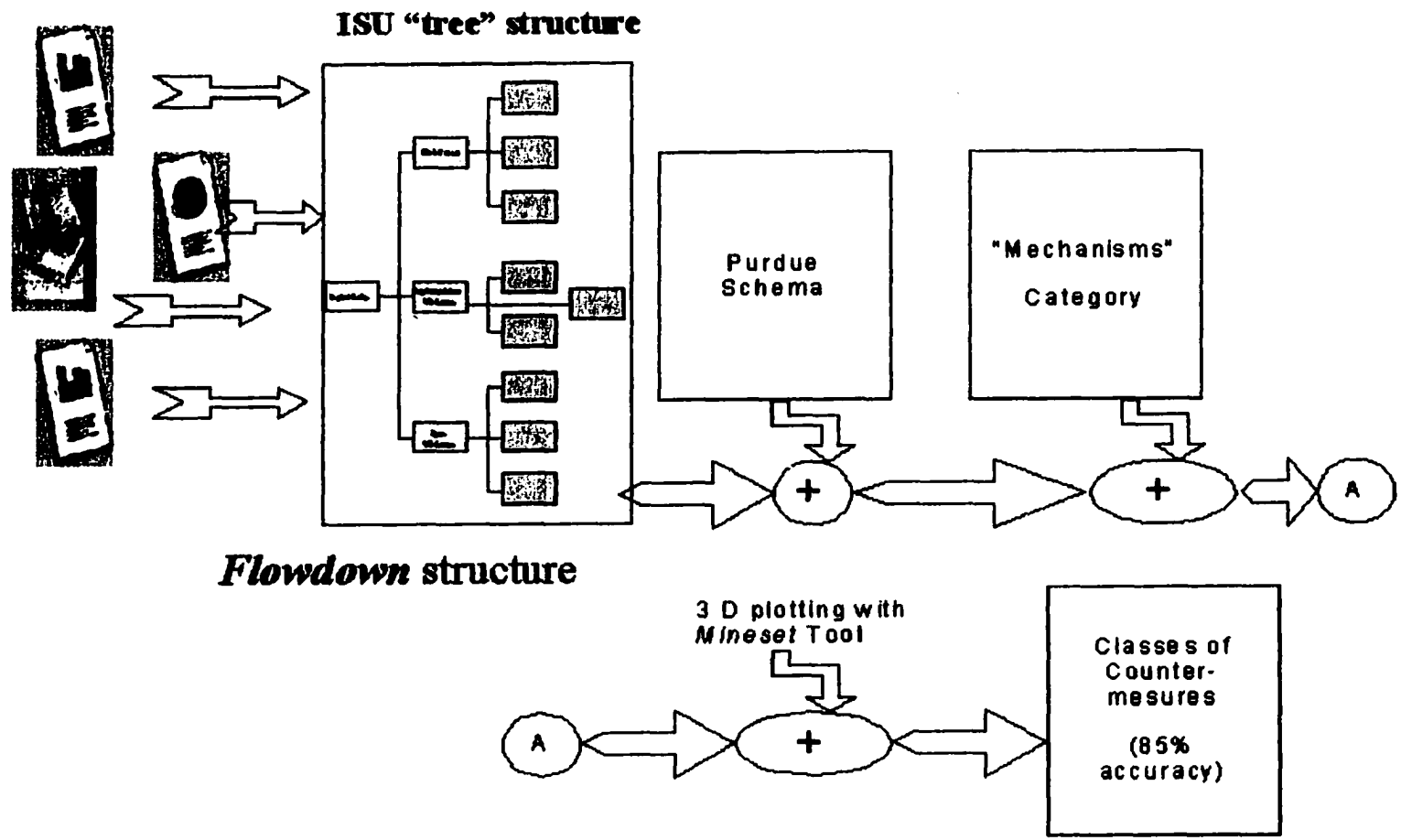
- To prevent the local network from becoming a source of *IP-spoofed* attacks, allow traffic to leave the network only if it has a legitimate source address, commonly called egress filtering. Turn off *IP Directed Broadcasts* on network devices under local jurisdiction to eliminate the possibility of becoming a Smurf bounce site.

For the *Poor Resource Protection/Access Control* (Category 6), this represents the hardest category of all. IPv6 may hold promise for authenticating traffic origin, but depends on the widespread availability of IPv6. With IPv4, there is no ironclad way to authenticate the source of a packet without extraordinary methods being deployed. This category is further broken down into these 4 subcategories, with the examples shown below:

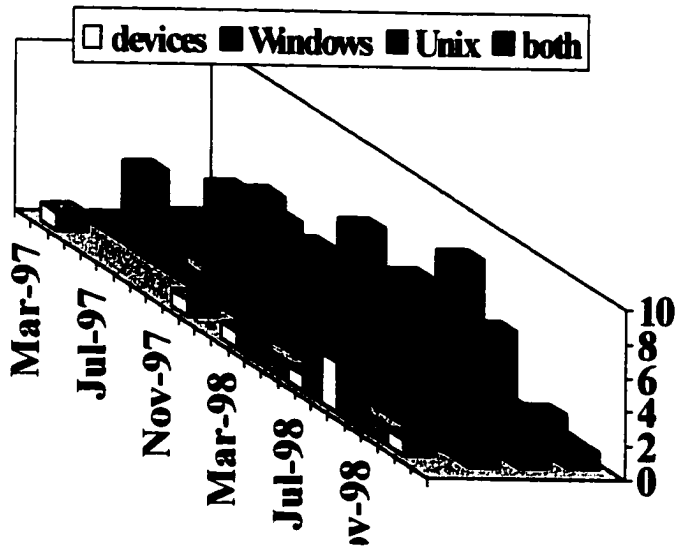
6.1	Poor Authentication Scheme	riptrace.c, htmlscript.txt
6.2	IP Spoofing	redirect, ICQhijack
6.3	Data Poisoning	rip.c, dns_redirect.c
6.4	Other misc. protection shortcomings	perliis.txt, cisco_remote_ios

APPENDIX C VULNERABILITY DATABASE EVOLUTION PROCESS

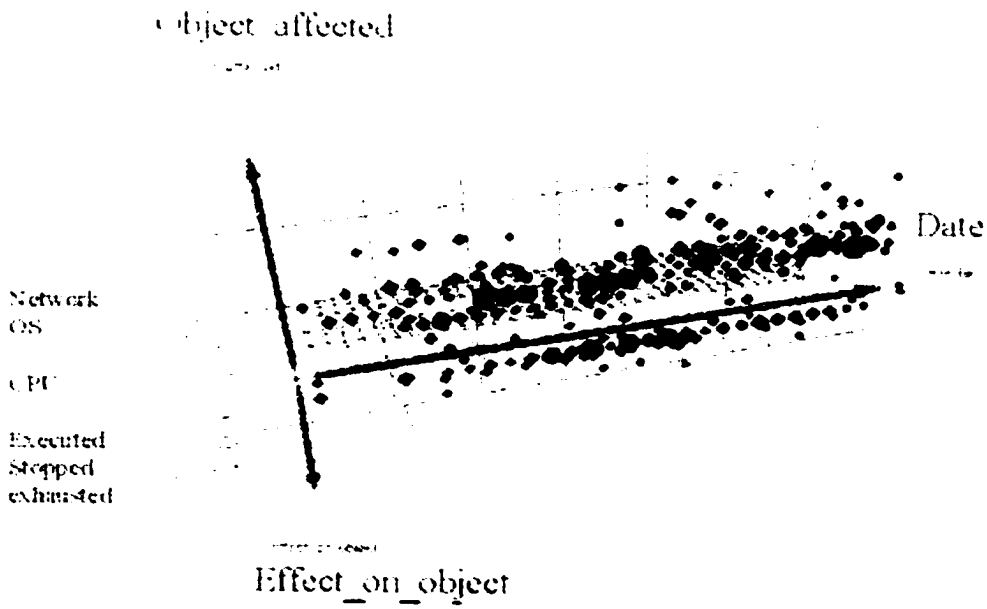
Previous Taxonomies



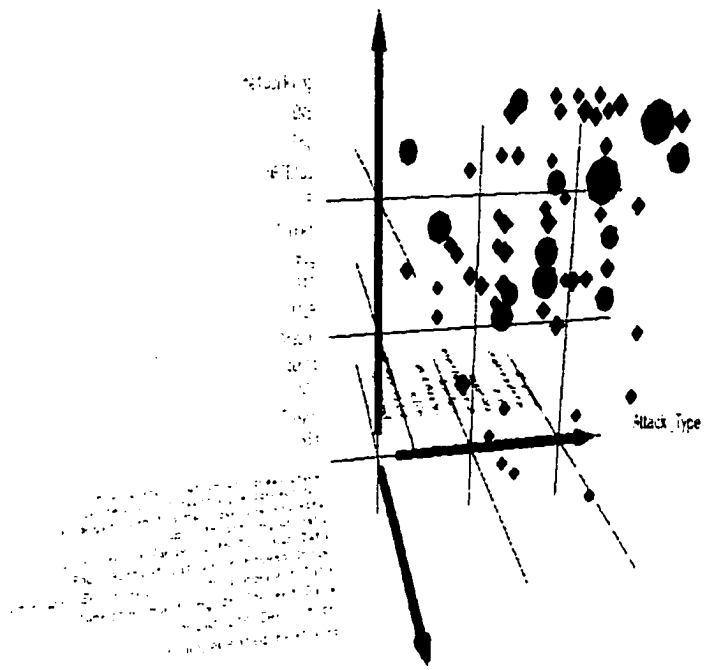
APPENDIX D: VARIOUS 3D PLOTS



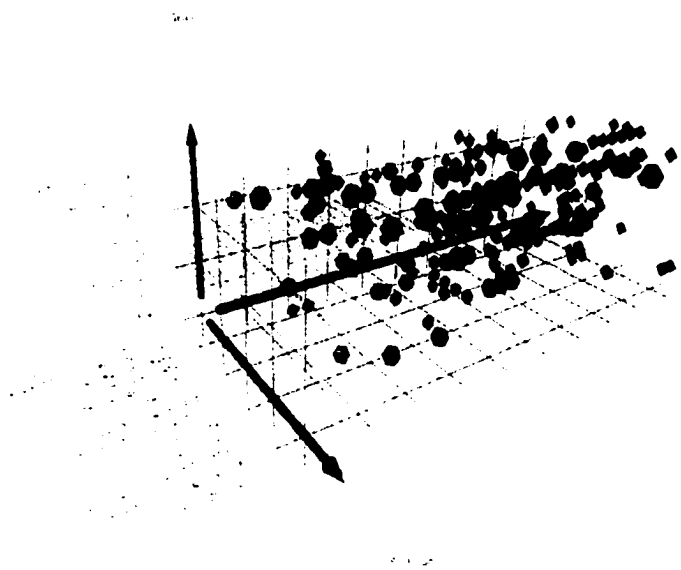
The First 3D Plot Using *Powerpoint* as a plotting tool



Another View of a 3D Plot Showing Stringing

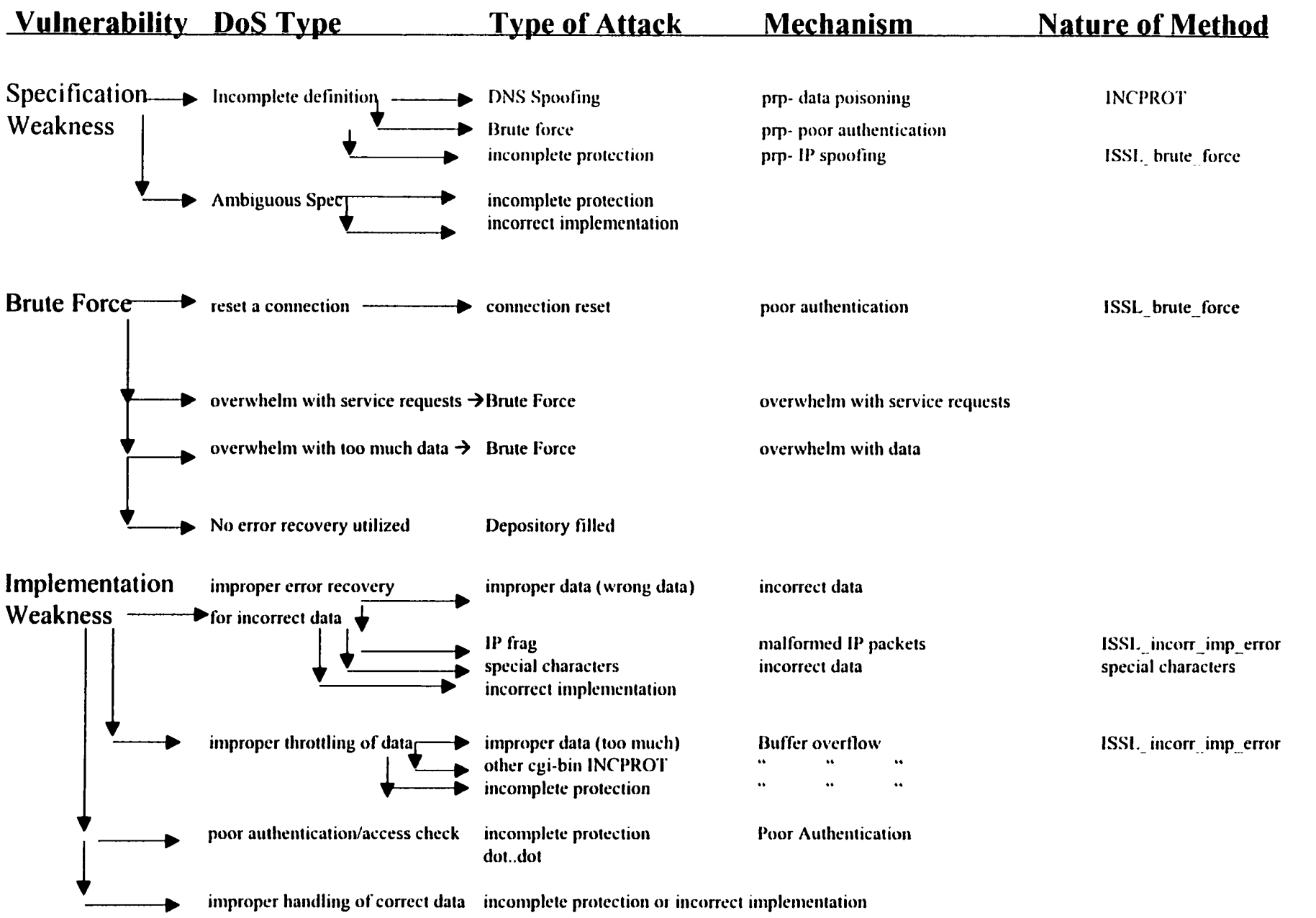


An early plot of 3D data (Attack_type vs Service vs Mechanism)



A poor choice of plotting categories shows no stringing at all, only a confused mess

APPENDIX E: VULNERABILITY CATEGORIES


Vulnerability
DoS Type
Type of Attack
Mechanism
Nature of Method
Specification Weakness
Incomplete definition
DNS Spoofing
prp- data poisoning
INCPROT
Brute force
prp- poor authentication
incomplete protection
prp- IP spoofing
ISSL_brute_force
Ambiguous Spec
incomplete protection
incorrect implementation
Brute Force
reset a connection
connection reset
poor authentication
ISSL_brute_force
overwhelm with service requests
Brute Force
overwhelm with service requests
overwhelm with too much data
Brute Force
overwhelm with data
No error recovery utilized
Depository filled
Implementation Weakness
improper error recovery for incorrect data
improper data (wrong data)
incorrect data
IP frag
malformed IP packets
ISSL_incorr_imp_error
special characters
incorrect data
special characters
incorrect implementation
improper throttling of data
improper data (too much)
Buffer overflow
ISSL_incorr_imp_error
other cgi-bin INCPROT
"
"
"
incomplete protection
"
"
"
poor authentication/access check
incomplete protection
dot..dot
Poor Authentication
improper handling of correct data
incomplete protection or incorrect implementation

BIBLIOGRAPHY

[1] Landwehr, C.E., Bull, A.R., McDermott, J.P., and Choi, W.S. (1994). *A Taxonomy of Computer Program Security Flaws, with Examples.* ACM Computing Surveys, 26-3.

[2] Bishop, M. (1995) *A Taxonomy of UNIX System and Network Vulnerabilities* Report CSE-95-10. UC Davis, Department of Computer Science.

[3] Aslam, T., Krsul, I., and Spafford, E. (1996) *Use of a Taxonomy of Security Faults.* Technical Report TR-96-051, Purdue University, COAST Laboratory.

[4] Bishop, M., and Heberlein, L.T. (1996). *An Isolated Network for Research.* Proceedings of the 19th National Information Systems Security Conference: pp. 349-360.

[5] Bishop, M. (1996). *A series of slides on the UC Davis Vulnerabilities Project.* NISSC Panel on Vulnerabilities Data. [On-line]. Available: www.cs.ucdavis.edu

[6] Bishop, M., and Bailey, D. (1996). *A Critical Analysis of Vulnerability Taxonomies, CSE-96-11.* UC Davis, Department of Computer Science.

[7] Krsul, I. (1997). *Computer Vulnerability Analysis: Thesis Proposal.* Technical Report CSD-TR-97-026, Purdue University, Dept of Computer Science, COAST Laboratory.

[8] Krsul, I. (1998). *Software Vulnerability Analysis*, Ph.D. thesis, Purdue University.

[9] Abbot, R.P. et al. (1976). *Security Analysis and Enhancements of Computer Operating Systems.* NBSIR 76-1041, Gaithersburg, MD, National Bureau of Standards, ICST.

[10] Bisbey II, R., and Hollingsworth, D. (1978). *Protection Analysis Project Final Report.* ISI/RR-78-13, DTIC AD A056816, USC/Information Sciences Institute.

[11] Krsul, I., Spafford, E., and Tripunitara, M. (1998). *Computer Vulnerability Analysis.* Purdue University, Coast Laboratory Publication.

[12] Spafford, E. (1997). *Common System Vulnerabilities*, Proceedings of the Workshop on Future Directions in Computer Misuse and Anomaly Detection. pp. 34-37.

[13] Meunier, P.C., and Spafford, E.H., (1999). Final Report of the 2nd Workshop on Research with Security Vulnerability Databases, Purdue University.

[14] Hancock, B. (1999). *Defending RADIUS Servers from Network Attacks with CyberwallPLUS.* Waltham, MA, Network-1 Security Solutions.

[15] The Aberdeen Group. (1999). Cyberwalls: *Network Security for an Interconnected World.* [On-line]. Available: http://www.aberdeen.com/cgi-bin/rf.cgi?doc_id=01991262

[16] IBM Whitepaper. (1997). *Local Area Network Directions.* IBM Website. [On-line]. Available: http://www.networking.ibm.com/lns/lns0c01.html

[17] Power, R. (1999). *Computer Security Issues & Trends survey.* San Francisco, CSI Computer Security Institute.

[18] NAI Whitepaper. (1999). *Adaptive Proxy Firewalls: The Next Generation Firewall Architecture.* NAI Website. [On-line]. Available: http://www.nai.com

[19] Checkpoint Corporation. (1998). *Products & Solutions: Checkpoint Cyberattack Defense System.* Checkpoint Website. [On-line]. Available: http://www.checkpoint.com/cyberdefense/index.html

[20] Checkpoint Corporation. (2000, June). Product Brochures, Israel, Checkpoint, Inc.

[21] Checkpoint Corporation. (1999). *Top 10 Challenges to Securing Your Network.* Checkpoint Website. [On-line]. Available: http://www.checkpoint.com

[22] Cisco Corporation. (2000, May). *Cisco LocalDirector*. Cisco Whitepaper. [On-line]. Available online: http://www.cisco.com/univercd/cc/td/doc/pcat/175.htm

[23] Cisco Corporation. (1996). *CiscoFusion Architecture*, Cisco Whitepaper. [On-line]. Available: http://www.cisco.com/warp/public/538/2.html

[24] Ellison, R.J. et al. (1999). *Survivable System: An Emerging Discipline*. Pittsburg, PA, Carnegie Mellon University, CERT Coordination Center.

[25] Fisher, D.A., & Lipson, H.F. (1999). *Emergent Algorithms- A new method for enhancing survivability in unbounded systems*. Proceedings of the Hawaii Intl. Conference on System Sciences, Maui.

[26] Linger, R.C. et al. (1999). *Requirements Definition for a Survivable Network System*. Pittsburg, PA, Carnegie Mellon University, CERT Coordination Center.

[27] Ellison, R.J. et al. (1998). *A Case Study in Survivable Networks Systems Analysis*. Carnegie Mellon University Tech Report CMU/SEI-98-tr-014.

[28] Computer Incident Advisory Capability (CIAC). (1998). Website Whitepaper. [On-line]. Available: http://ciac.llnl.org

[29] Bugtraq team. (1999). *Advisories concerning Network Denial of Service Attacks*, Security Focus Website Whitepaper. [On-line]. Available: http://www.securityfocus.com/templates/advisory.html

[30] Online *Bugtraq classification scheme*. (1999). [On-line]. Available: http://www.securityfocus.com/vdb/bottom.html?section=help&vid=512

[31] Cert Coordination Center. (1999). *Tech Tips: Denial of Service*. Pittsburgh, PA, Carnegie Mellon Software Engineering Institute.

[32] Kermode, T. (1999). *Denial of Service Attacks on any Internet Server through SYN Flooding*, Website Whitepaper. [On-line]. Available: http://www.zebra.co.uk/tom/writing/flood.htm

[33] Cisco Webteam. (1999). *Defining Strategies to Protect against UDP and TCP Denial of Service Attacks*. Cisco Website Whitepaper. [On-line]. Available: http://www.cisco.com

[34] Cisco Corporation. (1999). *Configuring 'TCP Intercept' to Prevent Denial of Service Attacks*. Cisco Website Whitepaper. [On-line]. Available: http://www.cisco.com

[35] Schuba, C.L. et al, (1999). *Analysis of a Denial of Service Attack on TCP*. Purdue University. Dept of Computer Science. COAST Laboratory.

[36] Cisco Corporation. (1999). *TCP Intercept*. A product description published at the Cisco Website. [On-line]. Available: http://www.cisco.com

[37] Hancock, B. (1998). *Defending Servers from External and Internal Attack*. Technical Paper. Waltham, MA, Network-1 Security Solutions, Inc.

[38] IBM. (1999). *MQSeries Gets Buy-in for QVC's Intranet Applications*. Hampshire, U.K., IBM Product Brochure.

[39] McGuire, A. (1999). *Providing of Service for Online Trading*. Whitepaper. Waltham, MA, Network-1 Security Solutions, Inc.

[40] McGuire, A. (1999). *Securing Microsoft Exchange Servers*. Whitepaper. Waltham, MA, Network-1 Security Solutions, Inc.

[41] QoSforum.com Group. (1999). *QoS Protocols and Architectures*. Stardust Website Whitepaper. [On-line]. Available: http://www.stardust.com

[42] Allot Communications. (1999). *Policy-Based Network Architecture*. Whitepaper. Los Gatos, CA, Allot Communications.

[43] Seaman, M., and Klessig, R. (1999). *Going the Distance with QoS*. Tech Tutorial. [On-line]. Available: http://www.data.com/ise/990207/distance.html

[44] Allot. (1999) *Multi-level Traffic Management- a Technical Overview*. Allot Website Whitepaper. [On-line]. Available: http://www.allot.com/technology/architecture.htm

[45] McConnell, J. (1997). *Whitepaper on Enterprise Traffic Management*. Boulder, CO, McConnell Consulting, Inc.

[46] Cowan, C. et al, (1999). *Buffer Overflows: Attacks and Defenses for the Vulnerability of the Decade*. Oregon State College Website Whitepaper. [On-line]. Available: http://www.cse.ogi.edu/DISC/projects/immunix

[47] Farrow, R. (1999). *Blocking Buffer Overflow Attacks*. Network.Magazine.com. [On-line]. Available: http://www.networkMagazine.com/Magazine/archive/1999/11/9911def.htm

[48] Devhead Webteam. (1998, Dec.). *Chart of DoS Attacks*, Zdnet Online Magazine. [On-line]. Available: www.zdnet.com/devhead/stories/prtfriendly/0,4558,2172788,00.html

[49] Sennie, D. (1999). *Changing the Default for Directed Broadcasts in Routers*. RFC 2644, Network Working Group. [On-line]. Available: www.ietf.org/rfc/rfc2644.txt

[50] CERT Coordination Center. (1998). CERT Advisory CA-98-01, *Denial of Service Tools*. Pittsburg, PA, Carnegie Mellon University, Software Engineering Institute.

[51] CERT Coordination Center. (1999). *Results of the Distributed-Systems Intruder Tools Workshop, November 2-4, 1999*. Pittsburgh, PA, Carnegie Mellon University, Software Engineering Institute.

[52] CERT Coordination Center. (1999). CERT Advisory CA-99-17, *Denial of Service Tools*. Pittsburgh, PA, Carnegie Mellon University Software Engineering Institute.

[53] CERT Coordination Center. (2000). CERT Advisory CA-2000-01, *Denial of Service Developments*. Pittsburgh, PA, Carnegie Mellon University, Software Engineering Institute.

[54] Todd, B. (2000, February 18). *Distributed Denial of Service Attacks*, Opensource Website Whitepaper. [On-line]. Available: http://www.opensourcefirewall.com/ddos_whitepaper_copy.html

[55] ICSA Whitepaper. (2000). *DDoS Attacks: Bringing Down the Web*. ICSA Website. Available online: http://www.icsa.net/html/communities/ddos/index.shtml . Feb 29, 2000.

[56] SANS Group. (2000). *Consensus Roadmap for Defeating Distributed Denial of Service Attacks*. SANS Website. [On-line]. Available: http://www.sans.org/ddos_roadmap.htm

[57] CERT Coordination Center. (2000). CERT Advisory CA-2000-01, *Denial of Service Developments*. [On-line]. Available: http://www.cert.org/advisories/CA-2000-01.html

[58] ICSA Organization. (2000). *DDoS: The New Denial of Service Threat*. [On-line]. Available: http://www.icsa.net/html/communities/ddos/ts_customers.shtml

[59] Watchguard Technologies. (2000). *Distributed Denial of Service: A Whitepaper*. [On-line]. Available: www.watchguard.com

[60] Dittrich, D. (1999). *Descriptions of DDoS attacks*. University of Washington. [On-line]. Available: http://staff.washington.edu/dittrich/misc/trinoo.analysis

[61] Cohen, F. (2000). *Managing network Security-- Countering DCAs*. Livermore, CA, Fred Cohen & Associates.

[62] COTSE Organization. (1999). *Anatomy of an Attack*. COTSE Website Whitepaper. [On-line]. Available: http://members.cotse.com/fulcrum/attacks.html

[63] Cisco Corporation. (1999). *Characterizing and Tracing Packet Floods Using Cisco Routers*. Cisco Website Whitepaper. [On-line]. Available: http://www.cisco.com/warp/public/707/22.html

[64] Craig, R. (2000). *Defending against Distributed Denial of Service Attacks*. Whitepaper. Fredricksburg, VA, Net-Centric Consulting Group.

[65] Blackcode.com Security Services. (2000). *Distributed Denial of Service Attacks*.

[66] Oak Ridge National Laboratory. (2000). *Backtracking Spoofed Packets*.

[67] Dittrich, D., and Mixter. (1999). *Analysis of Trinoo, Tribal Flood Network, and TFN2K*. University of Washington. [On-line]. Available: staff.washington.edu/dittrich/misc/tfn.analysis

[68] Barlow, J., and Thrower, W. (2000). *TFN2K-- An Analysis*. Packetstorm Website Whitepaper. [On-line]. Available: http://packetstorm.security.com/distributed/tfn2k_analysis.htm

[69] Nomad, S. (2000). *Strategies for Defeating Distributed Attacks*. NMRC Website Whitepaper. [On-line]. Available: http:// www.nmrc.org

[70] Cisco Corporation. (1999). *Characterizing and Tracing packet Floods Using Cisco Routers*. Cisco Whitepaper. [On-line]. Available: http://www.cisco.com

[71] Ferguson, P. and Senie, D. (2000). *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing, RFC 2827*. Obsoletes RFC 2827. [On-line]. Available: www.ietf.org/rfc/rfc2827.txt

[72] Techmall Website. (1997). *MCI Helps Internet Community Combat Hacking with Free Distribution of Innovative Network Security Tool*. [On-line]. Available: http://www8.techmall.com/techdocs/NP97109-3.html

[73] WiredNews, Inc. (2000). *DosTracker*. WiredNews.com Website Whitepaper. [On-line]. Available: http://www.wirednews.com/news/technology/0.1282,9506,00.html

[74] Ferguson, P., and Senie, D. (1998). *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing, RFC 2267*. [On-line]. Available: www.ietf.org/rfc/rfc2267.txt

[75] Cisco Corporation. (1999). *Committed Access Rate*. Cisco Website Whitepaper. [On-line]. Available: http://www.cisco.com/warp/public/732/Tech/car/index.html

[76] Cisco Corporation. (2000). *Unicast Reverse Path Forwarding*, Cisco Website Whitepaper. [On-line]. Available: www.cisco.com/univercd/cc/td/doc/product/software/ios111/cc111/uni_rpf.htm

[77] Masica, K. (2000). Understanding the IP Security Protocol. *Internet Security Advisor Magazine*.

[78] Srisuresh, P. & Sanchez, L. (1999). *Policy Framework for IP Security* (draft-ietf-ipsec-policy-framework-00.txt). IPSec Working Group Website Whitepaper. [On-line]. Available: http://www.ietf.org/Internet-drafts/draft-ietf-ipsec-policy-framework-00.txt

[79] Loshin, *P.* (1998, October). IP: The Next Generation. *Information Security Magazine*.

[80] IBM Corporation. (1998). *IPSec Security Technology*. IBM Website Whitepaper. [On-line]. Available: http://www.software.ibm.com/enetwork/technology/roadmap/ipsec2.html

[81] Warfield, M.H. (1999). *Observations from the Eighth Layer*. Linux World Website Whitepaper. [On-line]. Available: http://www.linuxworld.net/linuxworld/lw-1999-01-ramparts_p.html

[82] Aventail Corporation. (1999). *IPSec vs SSL.* Aventail Website Whitepaper. [On-line]. Available: http://www.aventail.com/index.phtml/solutions/tech_briefs/ipsec_ssl.phtml

[83] Harrison, A. (1999, September. Internet Protocol Security, *ComputerWorld Magazine.*

[84] Checkpoint Corporation. (1998). *VPN Security Components.* Checkpoint Website Whitepaper. [On-line]. Available: http://www.checkpoint.com/products/vpn1/vpnwp.html

[85] E-Lock Technologies Corporation. (1998). *VPN/Using IPSec/ISAKMP.* E-Lock Website Whitepaper. [On-line]. Available: http://www.e-lock.com/white-p/IPSEC.HTM

[86] Cabletron Corporation. (1999). *IP Security (IPSec).* Cabletron Website Whitepaper. [On-line]. Available: http://www.cabletron.com/vpn/VPNipsec.htm

[87] VPN Technologies Corporation. (1999). *VPN Technologies.* VPN technologies Website Whitepaper. [On-line]. Available: http://www.vpn.outer.net.html/technologies.html

[88] Martin, S. (1999). *Authentication and Privacy in IPv4 and IPv6.* 3Com Website whitepaper. [On-line]. Available: http://www.3com.com/enterprise/vpn/ipsec.html

[89] Kessler, G.C. (1998). IPv6: *The Next generation Internet.* Hill.com Website Whitepaper. [On-line]. Available: http://www.hill.com/library/IPv6_exp.html

[90] Derfler, F.J. (1998, February). IPv6: Will Good Sense Prevail? *PC Magazine.*

[91] Cobb, S. (1999, Spring). Sec Specs. *Security Advisor Magazine.*

[92] Taschek, J. (2000, January 17). Prepare to Send via IPv6. *Smart Reseller Magazine.*

[93] DeJesus,E. (1999, April). IPSec Goes Mainstream. *Information Security Magazine.*

[94] Microsoft Corporation. (1999). *Step-by-Step Guide to Internet Protocol Security (IPSec)*. Microsoft Website Whitepaper. [On-line]. Available: http://www.microsoft.com/WINDOWS2000/library/planning/sec.../ipsecsteps.asp

[95] Deering, S., and Hinden, R. (1999, December). *Internet Protocol, Version 6 (IPv6) Specification*, RFC 2460 (obsoletes RFC 1883). [On-line]. Available: www.ietf.org/rfc/rfc2460.txt.

[96] Sabo, D. (1999, August). *Electronic Commerce Barriers Survey results*. Ernst & Young, LLP Website Whitepaper. [On-line]. Available: www.itaa.org/software/research/indpulse/bartext.htm

[97] Null, C. (1999, February). Ecommerce to Go. *Network World Magazine*.

[98] Cisco Corporation. (1999, July). *The Global networked Business: A Model for Success*. Cisco Webpage Whitepaper. [On-line]. Available: http://www.cisco.com/warp/public/756/gnb_gen/gnb_wp.htm

[99] U.S. Government Publication. (1999). *Building out the Internet*. Ecommerce Webpage Whitepaper. [On-line]. Available: http://www.ecommerce.gov/building.htm

[100] IBM Web team. (1998). *Building Trust into E-business*, IBM Website. [On-line]. Available: www.ibm.com

[101] Sahlin, J.P. (1999, February). *Outsourcing Electronic Commerce*. Usi Report. [On-line]. Available: www.usi.net/pdf/outsoucing/ecommerce.pdf

[102] U.S. Government Bureau. (1998, November). *US Govt Working Group on Electronics Commerce Annual Report*. Washington, DC.

[103] Margherio, L. (1999). *The Emerging Digital Economy*. Whitepaper. Washington, DC, U.S. Dept. of Commerce.

[104] Business Objects Group. (1998, October). Putting the "E" in Business Intelligence. *Business Objects Newsletter*.

[105] IBM team. (1998). *Network Computing Framework: Architecture* Workbook. IBM Website Whitepaper. [On-line]. Available: www.ibm.com

[106] Stone, R. (1999). *Centertrack: An IP Overlay for Tracing DoS Floods*. UUNET Whitepaper. [On-line]. Available: www.nanog.org/mtg-9910/robert.html

[107] Savage S. et al. (2000). *Practical Network Support for IP Traceback*. Technical Report UW-CSE-00-02-01. University of Washington, Seattle, Dept. of Computer Science & Engineering.

[108] Bellovin, S. (2000). *ICMP Traceback Messages*. Internet Draft Whitepaper. [On-line]. Available: draft-bellovin-itrace-00.txt

[109] Moskowitz, R. (2000). *Host Identity Payload Architecture*. Internet Draft Whitepaper. [On-line]. Available: draft-moskoitz-hip-arch-01.txt .

[110] Hollander, Y., and Agostini, R. (2000, Sept./Oct.). Stop Hacker Attacks at the OS Level. *Internet Security Advisor Magazine*.

[111] Schnackenberg, D. et al. (2000, January). *Infrastructure for Intrusion Detection and Response*. Proceedings of the DARPA Information Survivability Conference & Exposition, Hilton Head, SC.

[112] Wu, S.F. (2000). *Project DECIDUOUS* (Decentralized Source Identification For Network-based Intrusions). University of North Carolina, Chapel Hill.

[113] Van, V.C. (1997). *A Defense Against Address Spoofing Using Active Networks*. Master's thesis. Cambridge, M.I.T., Dept. of Electrical Engineering and Computer Science.

[114] Cheung, S., and Levitt, K. (1997). *Protecting Routing Infrastructures from Denial of Service Using Cooperative Intrusion Detection*. Proceedings of the New Security Paradigms Workshop. Sept 23-26, Cumbria, UK.

[115] Croll, A.A. (1999, October). SSL Crunch Time. *Information Security Magazine*.