

# A Threat-Rigidity Analysis of the Apache Software Foundation's Response to Reported Server Security Issues

By

Yoav Shapira

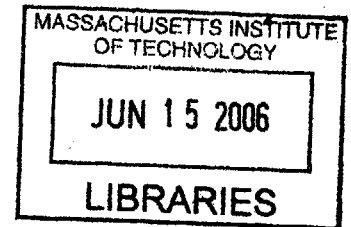
Submitted to the System Design and Management Program  
In Partial Fulfillment of the Requirements for the Degree of

Master of Science in Engineering and Management

At the

Massachusetts Institute of Technology  
February 2006

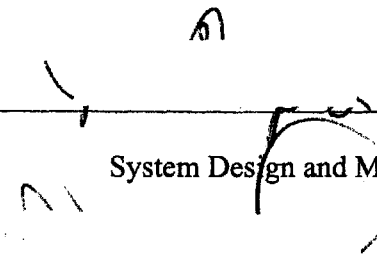
© 2006 Yoav Shapira  
All rights reserved




ARCHIVES

The author hereby grants to MIT permission to reproduce and to distribute publicly paper and electronic copies of this thesis document in whole or in part.

Signature of Author

  
Yoav Shapira  
System Design and Management Program  
February 2006

Certified by

  
Eric A. von Hippel  
Professor of Management, Sloan School of Management

Certified by

  
Patrick Hale  
Director, System Design and Management Program

# **Abstract**

## **A Threat-Rigidity Analysis of the Apache Software Foundation's Response to Reported Server Security Issues**

By

Yoav Shapira

Submitted to the System Design and Management Program in Partial Fulfillment of the Requirements for the Degree of Master of Science in Engineering and Management

There exists a broad body of literature documenting organizational responses to competitive threats, including those responses which fit into the threat-rigidity hypothesis. The purpose of this thesis is to investigate how a novel organizational form, the open-source software development community known as the Apache Software Foundation, responds to a specific type of threat: security issues reported to exist in its software products.

An analysis of publicly available data from the Apache Software Foundation is conducted, the security issue handling process is described in detail, and an analysis on security issue origin, severity, and resolution is provided. Special attention is given to communication along the issue resolution process, as the threat-rigidity hypothesis predicts a reduction in the flow of information across the organization. The results show that this organization defies some central predictions of the hypothesis: there is little reduction in information flow, little or no centralization in decision-making, and no loss of group-level focus.

The research results are framed within the literature of user-led innovation and organizational behavior. The implications for traditional software development organizations are discussed, and recommendations for further research are provided.

Thesis Supervisor:  
Eric A. von Hippel, Professor of Management  
Sloan School of Management

## **Acknowledgements**

I am fortunate to have had several inspirational figures over the past years, most of them providing me with support and encouragement on a regular basis without being aware of the fact.

I would like to thank Professor Eric A. von Hippel for providing guidance, encouragement, and knowledge not only on matters relevant to this thesis, but on scientific research and academic life in general. I was fortunate to find an advisor who shares my fascination with this topic.

My teachers have played a significant role in this thesis by sharing their vast experience and knowledge in diverse areas. I would like to thank professors Thomas Allen, Michael Cusumano, Ralph Katz, Nancy Leveson, and David Simchi-Levi of MIT, as well as professors John Brackett and Richard Vidale of Boston University. I would especially like to thank Professor James Utterback of MIT for teaching me entirely new ways of looking at technologies, industries, and innovation, and Dr. Karim Lakhani of MIT for providing me with plentiful ideas and boundless energy from the very first draft proposal for this thesis.

Many friends have helped me with this thesis more than they know. I would like to thank my fellow students for their insightful comments in and out of class, and my fellow hackers out in cyberspace for their dedication and spirit. I thank the Apache security team and the Foundation in general for being available and helpful.

Finally, I would like to thank my wife Allison and my parents Oz and Etty. Their support and love are truly unconditional and unlimited. Without them, this thesis and my academic work would not have been possible. Their encouragement helped me conquer all my challenges, and I can only hope to provide them, as well as my future children, with the same level of support when it is needed.

## Table of Contents

Abstract.....	2
Acknowledgements.....	3
Table of Contents.....	4
List of Figures.....	5
List of Tables .....	6
Chapter 1: Introduction.....	7
1.1    The Apache Software Foundation .....	9
1.2    Expected Findings.....	11
1.3    Thesis Structure .....	13
Chapter 2: Literature Review and Research Questions .....	14
2.1    The Threat-Rigidity Hypothesis .....	15
2.2    Restriction in Information Processing .....	17
2.3    Constriction of Control .....	19
2.4    Open-Source Software Development .....	21
2.5    Summary of Research Questions .....	24
Chapter 3: Research Methodology.....	25
3.1    Security Issue Databases.....	26
3.1.1    Common Vulnerabilities and Exposures (CVE).....	26
3.1.2    SecurityFocus and BugTraq.....	28
3.1.3    Computer Emergency Response Team (CERT).....	30
3.2    Sample Selection: Why Apache?.....	34
3.3    Apache Security Team Information.....	35
3.3.1    What is the Core Product Team? .....	37
3.4    Source Code Repositories.....	38
3.5    Product Release Announcements.....	41
Chapter 4: Results and Analysis .....	42
4.1    The Apache Security Process.....	44
4.2    Security Issue Severity Classification.....	53
4.3    User Participation.....	61
Chapter 5: Conclusions and Further Work .....	75
5.1    Threat-rigidity? .....	77
5.2    Limitations of this Research .....	80
5.3    Further Work.....	82
Bibliography .....	84
Appendix A: Original Thesis Proposal.....	88
Appendix B: Raw Security Issue Record.....	90
Appendix C: Security Record Processing Scripts.....	124



## List of Figures

Figure 1 - Apache Development Process, Adapted from LinuxCare 2000 .....	10
Figure 2 - Key Elements of the Threat-Rigidity Hypothesis .....	16
Figure 3 - Typical CVE Entry.....	27
Figure 4 - Typical SecurityFocus Entry.....	29
Figure 5 - SecurityFocus External References.....	30
Figure 6 - Typical Apache-Related CERT Advisory.....	32
Figure 7 - Parts of the Revision History for a Typical CERT Advisory.....	33
Figure 8 - Apache Security Team Records (Abbreviated Sample).....	36
Figure 9 - Apache Web Server Commit Log Example: Fixing CVE CAN-2005-2088 ...	38
Figure 10 - Sample Change Log showing Fix for CVE-2004-0811 .....	39
Figure 11 - Product Release Announcement Example .....	41
Figure 12 - Apache Security Process .....	50
Figure 13 - Security Issues by Severity .....	56
Figure 14 - Average Resolution Time by Severity .....	57
Figure 15 - Apache 2.0 Average Time between Releases .....	58
Figure 16 - Apache 1.3 Average Time between Releases .....	60
Figure 17 - Sources of Reported Security Issues .....	61
Figure 18 - Breakdown of External Sources of Issue .....	63
Figure 19 - Issues Reported by External Reporters Histogram .....	65
Figure 20 - Security Analyst Issue Report Sample Page 1 .....	66
Figure 21 - Security Analyst Issue Report Sample Page 2 .....	67
Figure 22 - Independent Hacker Bug Report (Excerpt 1).....	68
Figure 23 - Independent Hacker Bug Report (Excerpt 2).....	69
Figure 24 - "Normal User" Issue Report Sample.....	70
Figure 25 - Breakdown of Sources of Issue Fixes .....	72
Figure 26 - Issues Fixed per Fixer .....	73

## **List of Tables**

Table 1 – Apache Security Issues, 2000 – 2005.....	55
Table 2 – Apache 2.0 Release History.....	58
Table 3 – Apache 1.3 Release History.....	59
Table 4 – Top Fixers versus Top Reporters.....	73

## Chapter 1: Introduction

*“Software is usually accompanied by documentation in the form of big fat scary manuals that nobody ever reads. In fact, for the past five years most of the manuals shipped with software products have actually been copies of Stephen King’s The Stand with new covers pasted on.”*

*–Dave Barry*

For better or worse, competition is the primary focus for most companies and managers in the world. It is an existing fact in most markets, and a weight for consideration in nearly all product development processes. After all, if one company’s product is inferior to a competitor’s offering, that company will most likely sell fewer units, make less money, and deliver less value to its shareholders. That, at least, is the prevailing thought in many organizations, and numerous strategies have been formulated and studied in order to defeat the competition in one manner or another.

Researchers in the academic fields of psychology, organizational behavior, strategy, and management have been conducting studies of competition for centuries. A broad and rich body of literature exists on the topic, and specific parts of it are reviewed later in this thesis. Within this body of work is a segment focusing on organizational responses to competitive threats: how do companies behave when a competitor’s product is shown to be superior, or when a newcomer threatens to corner a lucrative market segment? How do individuals or groups scramble to address the threat? What is the role of managers in redirecting strategy – do they just add stress? These questions and more are of interest to researchers in the combined areas of organizational behavior, psychology, and technology management.

One hypothesis in this area is that of threat-rigidity. Originally posed by Staw, Sandelands, and Dutton (Staw et al. 1981), it makes several observations and assertions regarding organizational responses to competitive threats. This hypothesis is discussed at length as part of the Literature Review chapter of this thesis. It has been both empirically and theoretically confirmed in a range of studies spanning diverse industries and

contexts. However, the vast majority of researchers surveyed traditional organizational structures: the for-profit company competing in an open market. The main contribution of this thesis is the survey of a novel organizational form, the free and open-source software development community, in light of the threat-rigidity hypothesis.

There has also been much research in recent years on open-source communities such as the Apache Software Foundation. This research is valuable and presents numerous interesting findings, but rarely do these publications describe operational aspects where time is of the essence, such as responding to a competitive threat. Indeed, one motto of many open-source software organizations is that “the product is ready when it’s ready” (Chaven 2003). Working according to a schedule is an exceedingly rare practice in the free and open-source software development world. Accordingly, the study of how this type of organization reacts when time is of the essence has not been widely pursued. Moreover, these organizations typically rely heavily on volunteers, some of which have not been involved in the product’s development, to help fix bugs and contribute new software. Are these volunteers effective in helping address competitive threats, or is the reliance on a volunteer work force a point of weakness when speed is of the essence?

As explained above, the purpose of this thesis is to explore how a relatively novel organizational form, the loosely-organized, geographically-distributed, non-profit open-source software development group, handles competitive threats. The key research questions include:

- How does the Apache Software Foundation handle reported security issues?
- Does the issue-handling process lead to restrictions in information flow and centralized decision-making, as predicted by the threat-rigidity hypothesis?
- Does such an organizational structure present advantages and benefits in responding to competitive threats?
- What is the role of volunteers in scrambling to address competitive threats?
- Can we apply lessons from this organizational structure to traditional ones?
- How can this research inform the broader fields of organizational behavior and management of innovation?

## **1.1 The Apache Software Foundation**

The Apache Software Foundation is a non-profit organization that consists of a community of software developers and the technical infrastructure required to support them (Coar 2005). The Foundation is one of the oldest and most eminent free and open-source development communities in the world, established in 1995 and providing more than 100 different products at no cost. The Apache Software Foundation was chosen as the focus of study for this thesis due to several reasons discussed below and further in the Sample Selection section of Chapter 3.

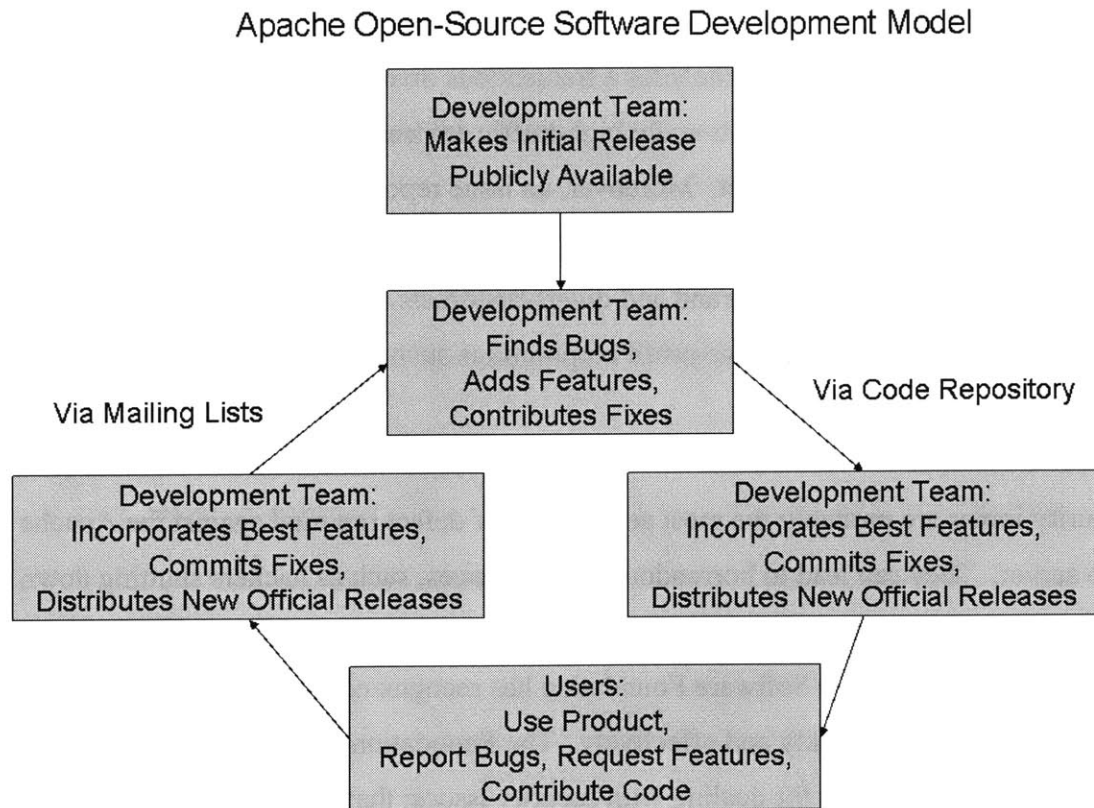
The Foundation's main product, the Apache HTTPD web server, is the dominant product in its market segment. Nearly 70% of the Internet's web sites are powered by the Apache server (Netcraft 2005). As this includes a tremendous diversity of server environments, mission-critical systems, and extremely high-traffic environments, users are quick to report any issue with the product. Moreover, an issue reported by one user is likely to affect millions of others. If untreated or improperly treated, these issues can cause serious damage to the Apache brand and divert customers to other products. The product faces strong competition from for-profit corporations such as Microsoft and Sun Microsystems.

Security issues are probably the most severe type of defect reported against the Apache web server. They can lead to horrendous consequences, such as hackers shutting down web sites or stealing financial and health care information about millions of customers. Accordingly, the Apache Software Foundation has recognized the importance of dealing with reported issues quickly and effectively. The Foundation has implemented a special and well-defined process for dealing with security issues; that process is discussed in detail in the Research Methodology chapter of this thesis.

In addition, much of the Apache Software Foundation's processes and data are publicly available. It is an open and accepting community, and direct contact with the people who respond to security issues is possible. Such contact, which was part of this thesis, can substantially improve the researcher's understanding of the relevant processes and issues.

Furthermore, because of the Foundation's age and the product's adoption rate, the sample size of security issues is larger than in many other open-source communities. This enabled a more statistically-significant analysis of the organization's response to competitive threats.

The following diagram illustrates the open-source software development process. While it originates in a book about the Linux operating system, the same process is applied in the Apache Software Foundation, with one relevant difference: "project management" and the "development team" are one and the same.



**Figure 1 - Apache Development Process, Adapted from LinuxCare 2000**

## **1.2 Expected Findings**

As the above section explains, the Apache Software Foundation is a volunteer-based community of software developers. It is not a traditional software development organization where employees are paid for their time and are therefore expected to act according to management directive. Indeed, there is no “management” per se in the Foundation, as no one member has the authority to direct another. Given the structure of the Foundation and the threat-rigidity hypothesis, we would make several educated assumptions about the security issue resolution process.

First, we would expect the burden of fixing security issues to divide roughly equally among the many different Foundation volunteers working on the project. We would not expect one or two members to fix a majority of security issues in the long-term, although over any shorter time period, for example a few months, one member may take an active leadership role in addressing security issues. To test this assumption, we will attempt to track the average number of issues addressed per developer: it should be close to one.

We would also expect that more serious issues would be addressed more quickly. While this makes common sense from a user perspective, it is worth noting that traditional software development organizations frequently bundle multiple fixes into one bigger release to help alleviate the burden of deploying the new release to many customers. In the open-source software development world, the “release early, release often” (Raymond 1999) mantra is common practice, and we would expect the Apache Software Foundation to issue new product versions faster than normal when security issues are involved. In order to test this assumption, we will analyze the time taken for new product releases overall and separately for those releases driven by serious security issues. We would expect the time to release security-motivated product versions to be less than the average.

Because the product we have chosen to examine, the Apache HTTPD server, is typically used by system administrators, third party integration engineers, and other technologically-advanced or “lead” users, we would expect these users to report a large percentage of security issues. Specifically, we would expect users to report a much larger

proportion of security issues than the team discovers by itself. An analysis of security issues sources will be conducted in order to determine the relative proportion of serious security issues reported by different types of users.

The threat-rigidity hypothesis predicts that organizations facing a competitive threat will react by centralizing decision-making, often resulting in a few senior individuals in management making technical decisions that are typically best left to the engineers most familiar with the project. At the Apache Software Foundation, the engineers are their own managers, so we would expect the normal decision-making process to stay in place, without management interruption. This assumption is difficult to test empirically, but we will conduct a detailed analysis of the security issue handling process and discuss opportunities for management interruption.

Finally, the threat-rigidity hypothesis also predicts a restriction in the flow of information across the organization during times of competitive threat. Because the Foundation espouses an open culture with publicly-accessible communications forums, we expect this facet of the hypothesis not to hold true. The detailed analysis of the security issue resolution process will attempt to address this aspect of the threat-rigidity hypothesis in a qualitative manner.



### **1.3 Thesis Structure**

This chapter provides an introduction to the thesis and what questions the research attempts to address. The discussion above describes the Apache Software Foundation, why it was chosen for this research, and the expected findings of this thesis according to the threat-rigidity hypothesis. As noted, there has been significant previous research on the Foundation and other open-source development communities, but the focus of this thesis is unique: it examines the response of this type of organization under stress from a competitive threat, where timely action is of the essence. This thesis is the first work to examine the threat-rigidity hypothesis in relation to this type of an organization.

The rest of the work is structured as follows:

- Chapter 2 is a review of relevant literature intended to establish the context for this work as drawn from previous research in competition, psychology, technology management, and user-led innovation. The focus of the chapter is on the relevant aspects of the threat-rigidity hypothesis, especially centralized decision-making and restrictions in information flow throughout the organization.
- Chapter 3 describes in detail the research methodology and sources of data used in this study, including the Apache security process and publicly available data sources.
- Chapter 4 presents the research results regarding incident handling, organizational response, and participant motivation. It also provides an analysis of the findings.
- Chapter 5 discusses the implications of the findings, draws conclusions, and provides recommendations for further work.

## Chapter 2: Literature Review and Research Questions

*“Given enough eyeballs, all bugs are shallow.”*  
–Eric S. Raymond

This chapter examines the relevant existing literature in order to provide a context for the work in this thesis. There exists a substantial body of work on competition, competitive strategy, and the psychology of individuals and groups responding to threats. There is also a growing collection of research studies on open-source software, user innovation communities, and related business models. This literature review reviews the key findings of interest in these fields, but it does not attempt to provide a detailed view of any one area of research. Moreover, the findings are reviews only as they related to the research objectives in this thesis: further references are made available to the interested reader but are not discussed in depth.

This thesis combines elements from numerous fields, and works will be reviewed with a focus on how they apply to a distributed user innovation community under stress. Relevant theoretical and empirical evidence will be cited as appropriate, with a focus on those aspects of the threat-rigidity hypothesis tested by the Apache Software Foundation. The key research questions are summarized at the end of the chapter.

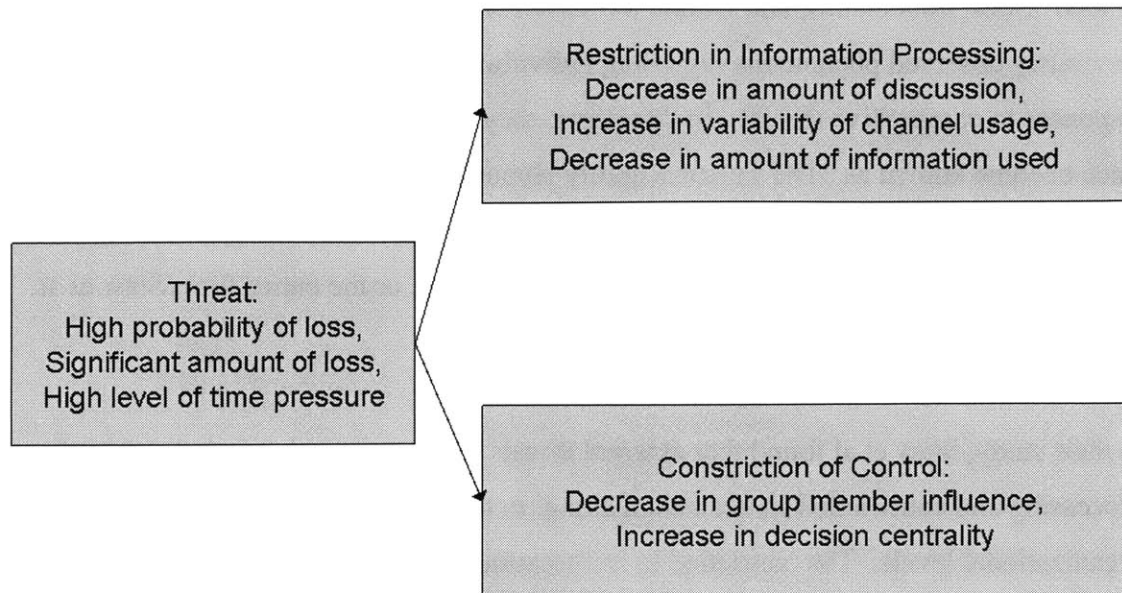
## **2.1 *The Threat-Rigidity Hypothesis***

In 1981, Staw, Sandelands, and Dutton published research combining many of the previously observed phenomena regarding individual, group, and organizational responses to competitive threats. In this paper, they put forth a set of claims that has since become known as “The Threat-Rigidity Hypothesis.” Staw et al define “threat” as “an environmental event that has impending negative or harmful consequences for the entity,” where “the entity” may be an individual, a group, or the entire firm (Staw et al. 1981).

In their study, Staw et al found that external threats lead to restriction in information processing and constriction in decision making, in turn leading to rigid responses at all organizational levels. The restriction in information processing manifests itself in the use of fewer communication channels for discussion and consideration of fewer strategic and tactical alternatives for action. The latter impact is sometimes called “tunnel vision,” and it can be detected by a decline in organizational sensitivity to peripheral cues. The constriction in decision-making is typically manifested via increased involvement from executive management, executives making decisions that are best left to middle managers and staff, less people involved in the decision-making process, and an increased focus on established processes rather than creative problem-solving.

Staw et al further posit that these rigid responses can be adaptive in a positive way for the organization, or maladaptive and leading to failure. They suggest that whether rigidity is good or bad is determined by the degree of environmental change: if the normal, causal links between firm performance and organizational processes are stable, rigidity is adaptive in a positive manner. But if the environment has changed significantly, rigidity is maladaptive and will lead to a deeper crisis.

## The Threat-Rigidity Hypothesis



Reproduced from Gladstein and Reilly (1985), Figure 1

**Figure 2 - Key Elements of the Threat-Rigidity Hypothesis**

In the twenty-five years since its publication, the work of Staw et al. has become one of the central theories of organizational behavior under stress. It has been studied by numerous researchers in diverse fields, including shipbuilding, electronics, government agencies, activist organizations, farm equipment, and more. Selected field studies are covered in subsequent sections of this chapter.

## **2.2 *Restriction in Information Processing***

One of the key predictions of the threat-rigidity hypothesis is a restriction in information flow along different organizational levels: at the individual level where the person ignores information not related to the current threat, the group level where the group lapses into “groupthink” (Janis 1982), the organization level where all focus is given to the current threat at the expense of other issues, and sometimes at the level of organizational partnerships and joint ventures (Staw et al. 1981).

The restriction in information flow is bothersome because it prevents the individual, group, or organizational from making fully-informed decisions. However, the existing literature shows that this restriction is not universal. It may be limited only to organizations small enough for the current threat to be critical (Audia and Greve 2002), or may only occur at the organizational-level instead of at all four levels discussed above (Beckman et al. 2004). For the Apache Software Foundation, we assume no one security threat is critical: the Foundation’s diverse user base and strong reputation prevents any one threat from destroying the organization.

The restriction in information flow seems to occur at the management levels of electronics firms (Hoffi-Hofstetter and Mannheim 1999) as well as non-profit drug treatment centers (D’Aunno and Sutton 1992). The Apache Software Foundation, like electronics firms, operates in a high-technology arena, but like the drug treatment centers relies on volunteers and runs a non-profit operation. Both studies pointed to management-level restriction of information flow as a threat-rigid behavioral response.

Other studies have considered the amount of attention devoted to internal versus external communications when an organization is faced with a competitive threat (D’Aveni and MacMillan 1990). As discussed in the findings of this research, the Apache Software Foundation maintains an open channel of communication with the reporters of security issues, offering to work together to identify and resolve issues as well as test suggested

patches. This equal focus on internal and external communications defies existing field evidence of the threat-rigidity hypothesis (Taylor 2001, Pauwels and Matthyssens 2002).

### **2.3 Constriction of Control**

The threat-rigidity hypothesis predicts that when faced with a competitive threat, organizations and groups will constrict control of operations to a smaller group of individuals than normal for that organization or group (Staw et al. 1981). This is frequently manifested as more senior individuals, including management, take control of low-level decision-making processes, make more technical and engineering decisions than they should, and become less open to alternative ideas, approaches, and strategies. As a side effect, these managers lose track of other activities, having become too involved in low-level details.

This part of the threat-rigidity hypothesis has been observed in the electronics industry (Hoffi-Hofstetter and Mannheim 1999), where the researchers found management to restrict decision-making to itself during a crisis and gradually adapt over time. It has also been observed in the withdrawal of firms from international market operations (Pauwels and Matthyssens 2002), not an area of concern for the Apache Software Foundation because its marginal cost of operating internationally compared to operating only in the United States is virtually zero.

Other authors have observed this restriction in information processing in controlled game situations (Gladstein and Reilly 1985), where the increased stress of having to make numerous decisions at a rapid pace caused rigid behaviors inside and across participant groups. As the research shows, Apache Software Foundation volunteers rarely work on more than one or two security issues at a time, and therefore this field finding does not apply to the Foundation.

Sometimes the restriction in information processing is natural: for example, drug treatment centers needing to reduce personnel necessarily turn only to senior management to make the downsizing decisions (D'Aunno and Sutton 1992). Again, the Foundation has no such decision to make as it is entirely volunteer-driven. Moreover, internal competition for resources within the organization is practically non-existent, as each volunteer does only what he or she wants to do, so the type of internal threat that

leads to rigid behaviors inside the organization (Thomas 1976) is not found at the Foundation.

Finally, some researchers investigated the theory that these threat-rigid behaviors among individuals were a function of stress. One of the most dominant factors of stress among software professionals is fear of professional obsolescence. This fear leads to individually threat-rigid responses, such as focusing only on the emergent issue at hand and ignoring alternative technologies or solutions proposed by others, over long periods of time (Rajeswari and Anantharaman 2003). These individual manifestations of restriction in information processing are not limited to the engineers but also appear at the management level, as evidenced by studies that framed threats honestly or as opportunities for advancement, resulting in different management behaviors (Xie and Wang 2003). In the Apache Software Foundation, where there are no managers and advancement is not a concern, we would not expect the same rigid behavior. In fact, because the Foundation's volunteers are not concerned about being fired, they may learn from these threat situations more than employees in conventional scenarios (Barnett and Pratt 2000), allowing true "double-loop learning" (Argyris and Schon 1978) at both the individual and group levels.



## **2.4 Open-Source Software Development**

The previous sections of this chapter describe the threat-rigidity hypothesis and related research in various fields. While the hypothesis has been considered in diverse scenarios, it has not been studied within the context of an open-source software development community. This section of the chapter provides a review of some relevant literature on the composition, function, and form of these distributed innovation groups. In order to understand the findings of this research, it is important to understand the community-driven nature of open-source software development, the role played by volunteer participants, the openness of the community, and the nature of its users as innovators who are ahead of the mainstream adoption curve.

Open-source software (OSS) is a term that combines technical aspects of software development, practices of project management and group organization, licensing and legal terms for the resulting software, and a philosophical approach to the production and use of software. The focus of this thesis is on group organization, management, and response to threats.

The premise behind OSS is that the source code for software is made publicly available, so that that “when programmers can read, redistribute, and modify the source code for a piece of software, the software evolves. People improve it, people adapt it, and people fix bugs. And this can happen at a speed that, if one is used to the slow pace of conventional software development, seems astonishing” (Open Source Initiative 2005).

Many authors have noted the “necessity is the mother of invention” proverb, and Raymond (1999) relates it to open-source software development by saying that “every good work of software starts by scratching a developer’s personal itch.” Developers of open-source software have numerous motivations, and many have personal or organizational software requirements that they fulfill by suitably enhancing an existing open-source software package.

Among the key motivations are the creative challenge of the work and a feeling of obligation to the project's community (Lakhani and Wolf 2005). Solving security issues can be regarded as a creative activity, especially for the highly capable and experienced programmers who have been involved in the project for a considerable time, as well as for the programmers who wrote the code in which the security vulnerability is revealed.

One of the main arguments against open-source software is the lack of a support department or help desk for users. OSS is supported mainly by volunteers using mailing lists, but there are also commercial organizations providing support and consulting services. The volunteer-driven support system is effective and efficient (Lakhani and von Hippel 2003), leading us to expect that a volunteer-driven security issue resolution system would also be effective. However, as shown by the security process analysis in the following chapters, the handling of security issues is done over a more private forum than normal support, perhaps reducing the amount of community learning in the process.

The users of the Apache web server who also contribute patches, documentation, and feature ideas are a classical example of lead users (von Hippel 1986). Reporting a security issue and suggesting a patch for it are a classical lead user activity, because the solution to a security flaw has high value to those lead users reporting the issue. To this effect, the Apache Software Foundation should go further in providing users with security information, "user innovation toolkits," or configuration options and tools to allow custom security solutions to users (Franke and von Hippel 2002).

Coar (2005) notes that the Apache Software Foundation's goal is for each project to develop a community of people with a common interest in that project. The Foundation simply "allows code creation to occur within this community." Management of the community is decentralized and delegated to the community itself, which usually does not have a set schedule for releases or other activities. Security issues, however, are unique in that they should be handled urgently, and there may be negative consequences to not resolving them quickly. The consideration of such issues is missing from much of

the literature on open-source software development, and it is the central research question of this thesis.

## **2.5 Summary of Research Questions**

The following is a summary of the primary research questions that formed the basis of this thesis.

- What aspects of the threat-rigidity hypothesis are shown to be accurate in an open-source software development organization?
- If any aspects of the threat-rigidity hypothesis are shown to be inaccurate within this type of organization, why is this so?
- How does the lack of formal management in the Apache Software Foundation impact the organization's response to competitive threats?
- How does the lack of a schedule or project manager impact the efficiency of the issue resolution process?
- What aspects of the open-source software development organization help with threat-rigidity, and which, if any, can be applied in a traditional organization?
- Does the open-source organization behave differently under competitive threats than it does during normal development?
- What role do the reporters of security issues play in their resolution? Do they report the issue and disappear? How often do they solve the problem or provide the code fix?
- What is the process used by the Apache Software Foundation to handle reported security issues? Is it effective and efficient? How does it compare to processes used by traditional organizations?

## Chapter 3: Research Methodology

*“The idea is to try to give all the information to help others to judge the value of your contribution; not just the information that leads to judgment in one particular direction or another.”*

*– Richard P. Feynman*

This chapter describes the sources of data used in this study and the research approaches used to obtain and characterize the data. There is a perception that within free and open-source software development communities such as the Apache Software Foundation, all data is publicly available. That perception is largely true, but security issues, due to their sensitive nature and potentially dangerous consequences, are one of the rare exceptions. These issues are typically made public only following resolution: in fact, the way most users find out about an existing security issue is when the product team announces a new software version that addresses the issue.

Fortunately for this research, however, once a fix for the issue is made available, much of the relevant history is released into the public domain as well. Moreover, during the past several years, centralized security issue tracking bodies and mechanisms have been established. These mechanisms are described in more detail below. The Apache Software Foundation and other vendors typically fully cooperate with these tracking bodies and comply with naming conventions, making cross-referencing of security issues possible.

The major challenge in gathering data for this study involved the notion of timing, which was central to this thesis. While it was fairly easy to confirm the date of public announcement for a given issue, it was frequently difficult or impossible to ascertain the original date when the issue was reported to Apache and therefore the time span for resolving the issue. The approach used to reconstruct this information is also described below.

In the spirit of the Feynman quote above, this thesis relies on publicly-available data sources. The data used is easy to independently analyze or verify.

### **3.1 Security Issue Databases**

There are several organizations that track security issues for multiple products in publicly-available databases on the Internet. This section describes three of the leading such databases, all of which were used to gather security issue data for the purposes of this thesis. There are several other relevant repositories on- and off-line, but their contents are nearly always included in one or more of these three databases.

For a given security issue, the following sources present some redundant information, which allows for cross-referencing and data validation. However, each source typically contributes at least one unique piece of information which is difficult or impossible to obtain elsewhere, and so a thorough examination of any specific security issue typically includes data from all of the sources detailed below.

#### **3.1.1 Common Vulnerabilities and Exposures (CVE)**

The CVE database is maintained by MITRE, a not-for-profit organization specializing in information technology, systems engineering, and related research.<sup>1</sup> The CVE database is primarily a naming and numbering scheme for security issues affecting the public. This includes bugs in numerous products, including the Apache web server, reported by many organizations and individuals.

The CVE database is not a solution center but simply a resource for uniquely naming and identifying issues for discussion and resolution. Many other data sources, including the Apache Software Foundation's own public announcements, use the CVE numbering scheme when discussing security issues. Even those organizations that assign their own number to the issue usually include the CVE number for cross-referencing purposes. Similarly, the CVE entry includes numbers used by other organizations as applicable.

---

<sup>1</sup> Please see <http://www.cve.mitre.org/about/> for more information about the CVE database, and <http://www.mitre.org/about/index.html> for more information about MITRE.

Issues are reported to CVE either by the person who identified them, by the entity affected, or by third parties. Thus, an Apache web server vulnerability may be reported by a hacker who discovered it, by the Apache Security Foundation, or by a user of the web server.

A sample CVE entry is shown below. This entry, number 913 for the year 2000, is for a typical Apache web server vulnerability. Note the cross-reference identification numbers for multiple other sources:

The screenshot shows a Mozilla Firefox browser window with the address bar displaying `http://www.cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2000-0913`. The page title is "CVE-2000-0913 - Mozilla Firefox". The website header features the "Common Vulnerabilities and Exposures" logo and the tagline "The Standard for Information Security Vulnerability Names". A navigation bar includes links for Home, Get CVE, About CVE, News and Events, Editorial Board, Advisory Council, and Compatible Products. The main content area displays the entry "CVE-2000-0913" with the version "CVE Version: 20040901". A text box on the right states: "Additional information is available from the National Vulnerability Database (also sponsored by US-CERT)". Below this, a paragraph explains that this is an entry on the CVE list, which standardizes names for security problems, and it was reviewed and accepted by the CVE Editorial Board. A table provides details about the entry:

Name	CVE-2000-0913
Status	Entry
Description	mod_rewrite in Apache 1.3.12 and earlier allows remote attackers to read arbitrary files if a RewriteRule directive is expanded to include a filename whose name contains a regular expression.

Below the table, a "References" section lists several identifiers:

- BUGTRAQ:20000929 Security vulnerability in Apache mod\_rewrite
- MANDRAKE:MDKSA-2000:060
- REDHAT:RHSA-2000:088
- REDHAT:RHSA-2000:095
- CALDERA:CSSA-2000-035.0
- HP:HPSBUX0010-126
- BUGTRAQ:20001011 Conectiva Linux Security Announcement - apache
- BID:1728

The browser window status bar at the bottom shows "Done".

Figure 3 - Typical CVE Entry

### **3.1.2 SecurityFocus and BugTraq**

SecurityFocus is a web site owned and operated by Symantec, Inc. with the aim of providing a central and exhaustive security vulnerability repository.<sup>2</sup> The site and its data are provided free of charge to the public. Like CVE, SecurityFocus contains data on products from numerous vendors, and information on issues reported by many individuals and organizations, including third parties not related to the product in question.

An essential part of the site is the BugTraq mailing list and its searchable archives. The BugTraq mailing list is one of the earliest sources of security information on the Internet, dating back to early Usenet adoption. It is still a central discussion point for security issues, with tens of thousands of subscribers and wide circulation. The BugTraq archives provide a treasure trove of security issue information, including data on who reported the issue and when, who researched the issue and how, and details of the issue resolution.

While the BugTraq mailing list originally assigned its own identifiers for security issues, that practice has ceased. Most of the current BugTraq issues use CVE numbers, and the archives are searchable by CVE numbers as well.

The screenshot below shows a typical SecurityFocus entry for an Apache web server vulnerability. Note the CVE cross-reference identification number and the links to discussion pages for the issue.

---

<sup>2</sup> Please see <http://www.securityfocus.com/about> for more details about the site.



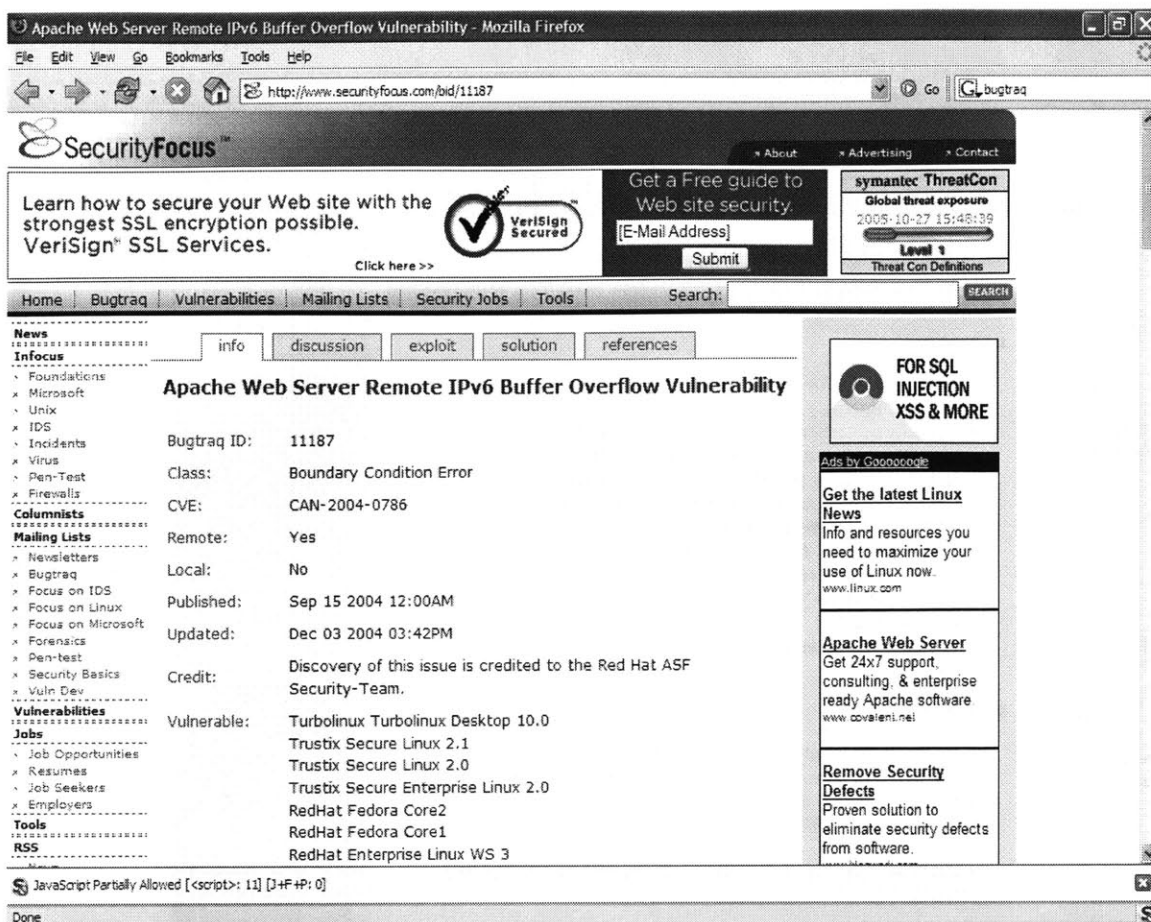


Figure 4 - Typical SecurityFocus Entry

The following screen shot shows the “references” tab from the above screen. It illustrates Security Focus’s excellent cross-referencing database. In this case, which is fairly typical for serious and confirmed security vulnerabilities, vendors other than Apache have released updates to their products which incorporate the Apache server. SecurityFocus provides links to that information and directs the user to the appropriate vendor site. In this case, note for example the updated Linux distributions from Red Hat, Fedora, TurboLinux, and other vendors.

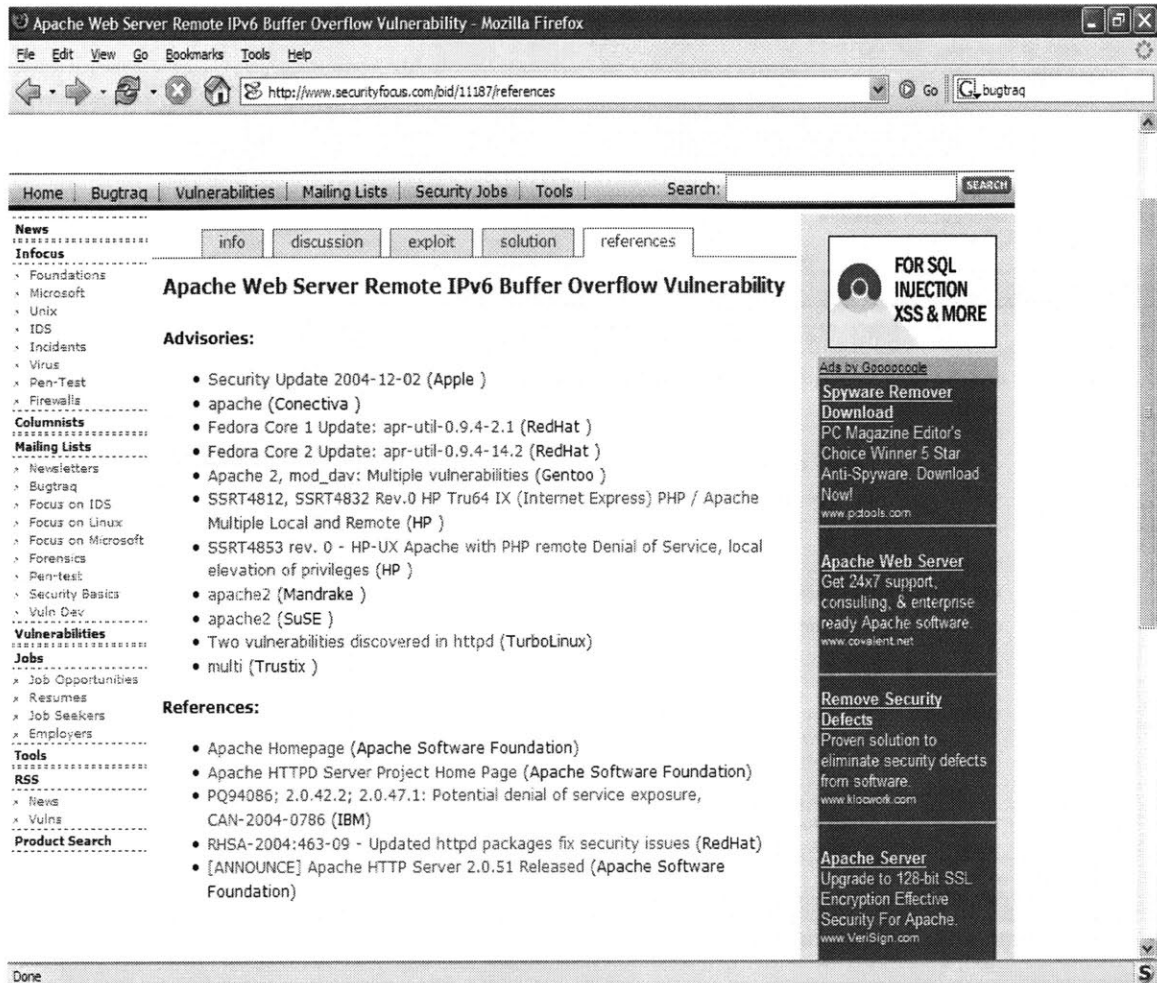


Figure 5 - SecurityFocus External References

### 3.1.3 Computer Emergency Response Team (CERT)

The CERT coordination center was established in 1988 by Carnegie Mellon University's Software Engineering Institute (SEI). Its aim is to provide an information center regarding security vulnerabilities of public interest and to foster research in this area.<sup>3</sup>

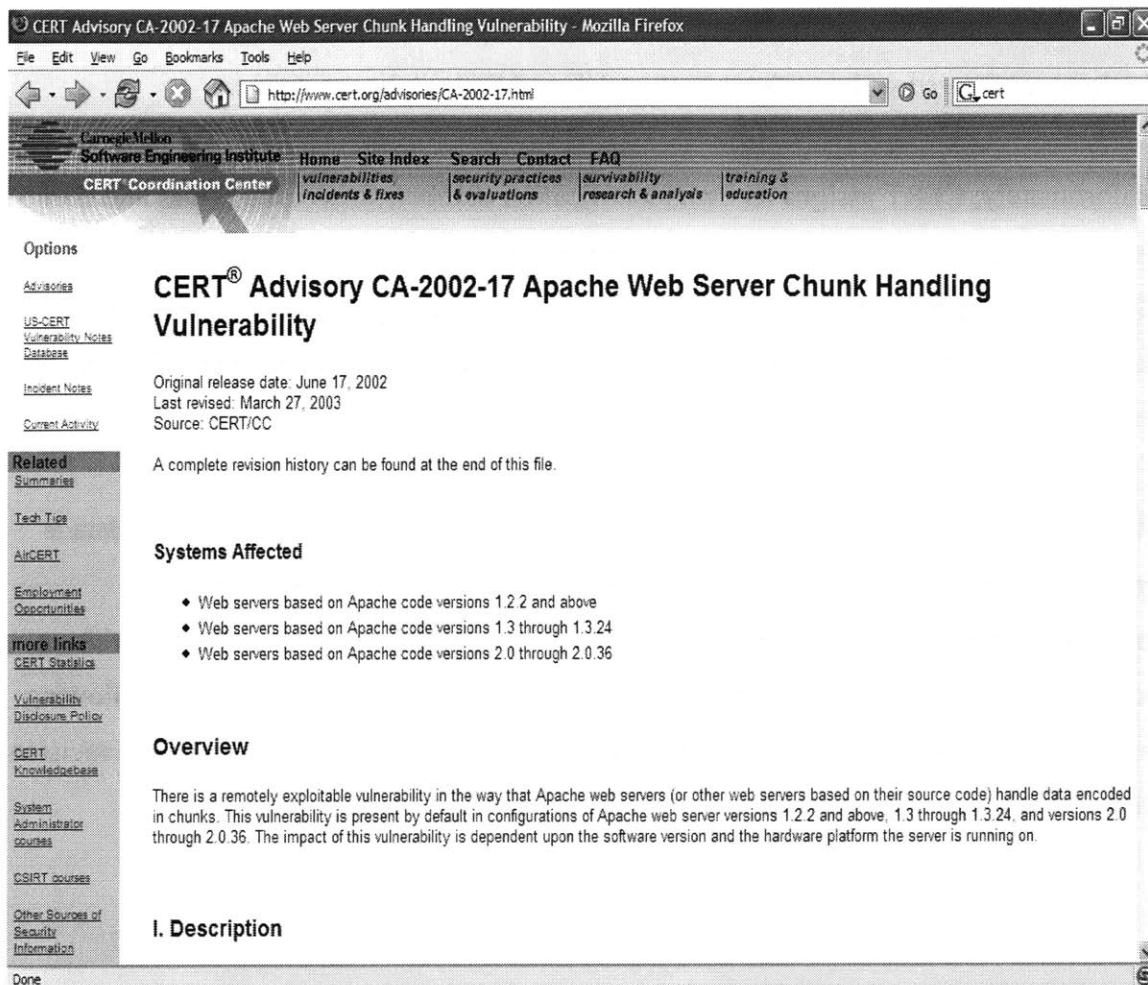
Over time, the CERT center has focused more on tracking consumer-centric security issues such as flaws in the Windows operating system or Internet Explorer web browser. It contains relatively little information on server products such as the Apache web server.

<sup>3</sup> Please see [http://www.cert.org/meet\\_cert/meetcertcc.html](http://www.cert.org/meet_cert/meetcertcc.html) for more background information about CERT.

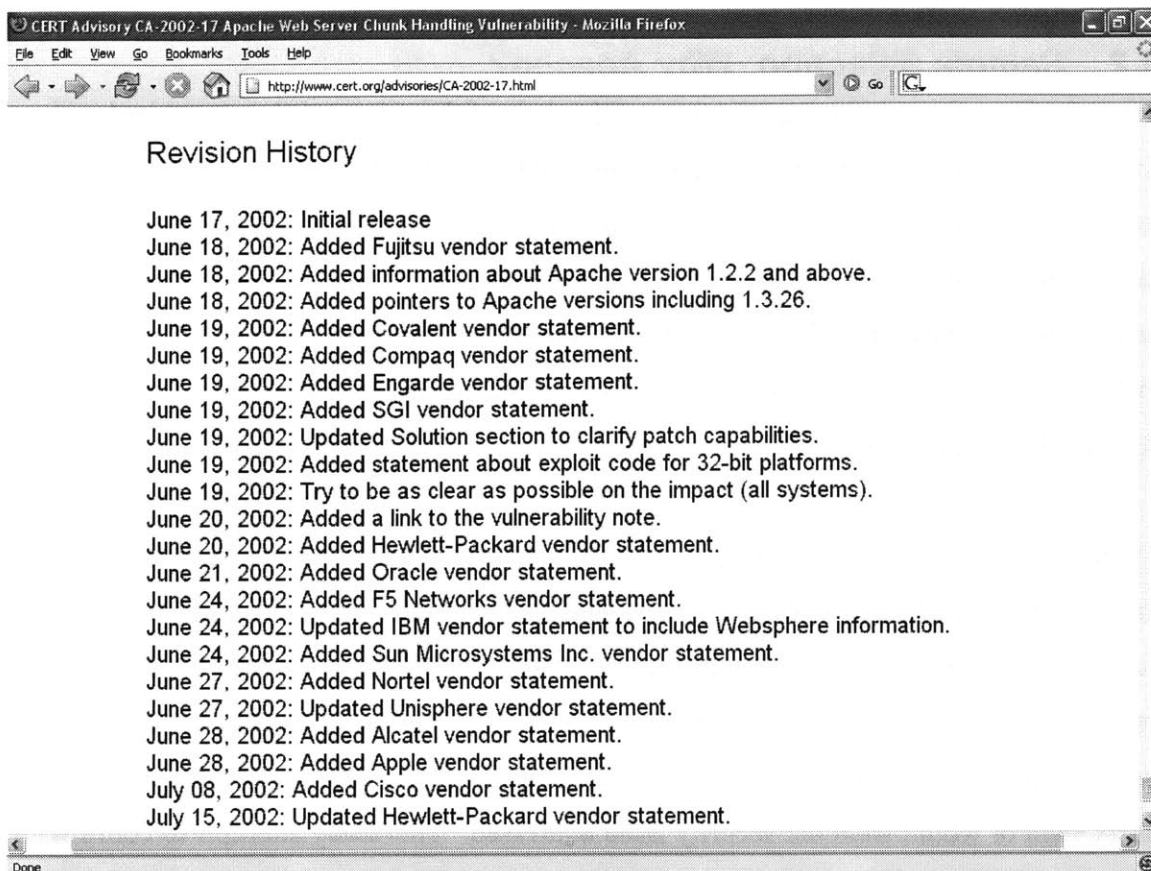
For example, as of this writing there were only 12 Apache-related advisories in the CERT database, compared with hundreds in the CVE and SecurityFocus indices.

Unfortunately, CERT assigns its own identification numbers to security issues and does not automatically cross-reference CVE numbers. However, because issues tracked by CERT are independently confirmed and of wide interest, they always exist in the CVE databases with their proper CVE identification; the researcher must manually link the two sources of data.

However, where the CERT center contains data on a specific security issue, that data is typically of high quality. More importantly, the CERT center is meticulous in maintaining and publishing the revision history for any issues it tracks. This provides a valuable timeline in security issue evolution for researchers. For example, the following screenshots illustrate a typical Apache-related CERT issue. Note specifically the revision history in the second screenshot:



**Figure 6 - Typical Apache-Related CERT Advisory**



**Figure 7 - Parts of the Revision History for a Typical CERT Advisory**

### **3.2 Sample Selection: Why Apache?**

Several other open-source organizations with successful products that were considered for this thesis include the Sendmail electronic messages transfer agent, the MySQL relational database system, the PHP scripting language, and the Linux operating system. In fact, one interesting extension to this thesis would be to conduct similar research at any one or more of these organizations. The Apache web server was chosen as the focus of this thesis because of its infrastructure-critical nature and the availability of data from the Apache Software Foundation.

There has been considerable research on lead users (von Hippel 1986) and their role as innovators. Von Hippel describes the two essential characteristics of lead users: their concerns foreshadow general demand, and they expect to obtain high benefits from the solution to their needs. It is intuitive to see that the first characteristic holds true for people reporting Apache security issues: their concerns echo general demand because all users want or would want the security issue addressed. This study attempts to confirm the second characteristic as well: reporters of security issues obtain benefits both by having the issue fixed for their own servers and by receiving praise or recognition in the community. Lead users and the relevant literature were discussed at length in Chapter 2.

The Apache Software Foundation lies at one extreme of the open-source software development world. It is among the most mature OSS organizations, at more than ten years old, more than one thousand committers (persons with the technical privileges to directly modify the software), and thousands of other contributors and users. Its processes are accordingly different from the many smaller open-source groups, including projects consisting of a handful (or less) of developers on SourceForge.net or Tigris.org. These projects are fascinating and worthy of analysis, but there are significant operational and organizational differences between them and the Apache Software Foundation. Other large and mature groups, such as the Linux project or the Mozilla Foundation, would present a more direct analog for comparative studies.

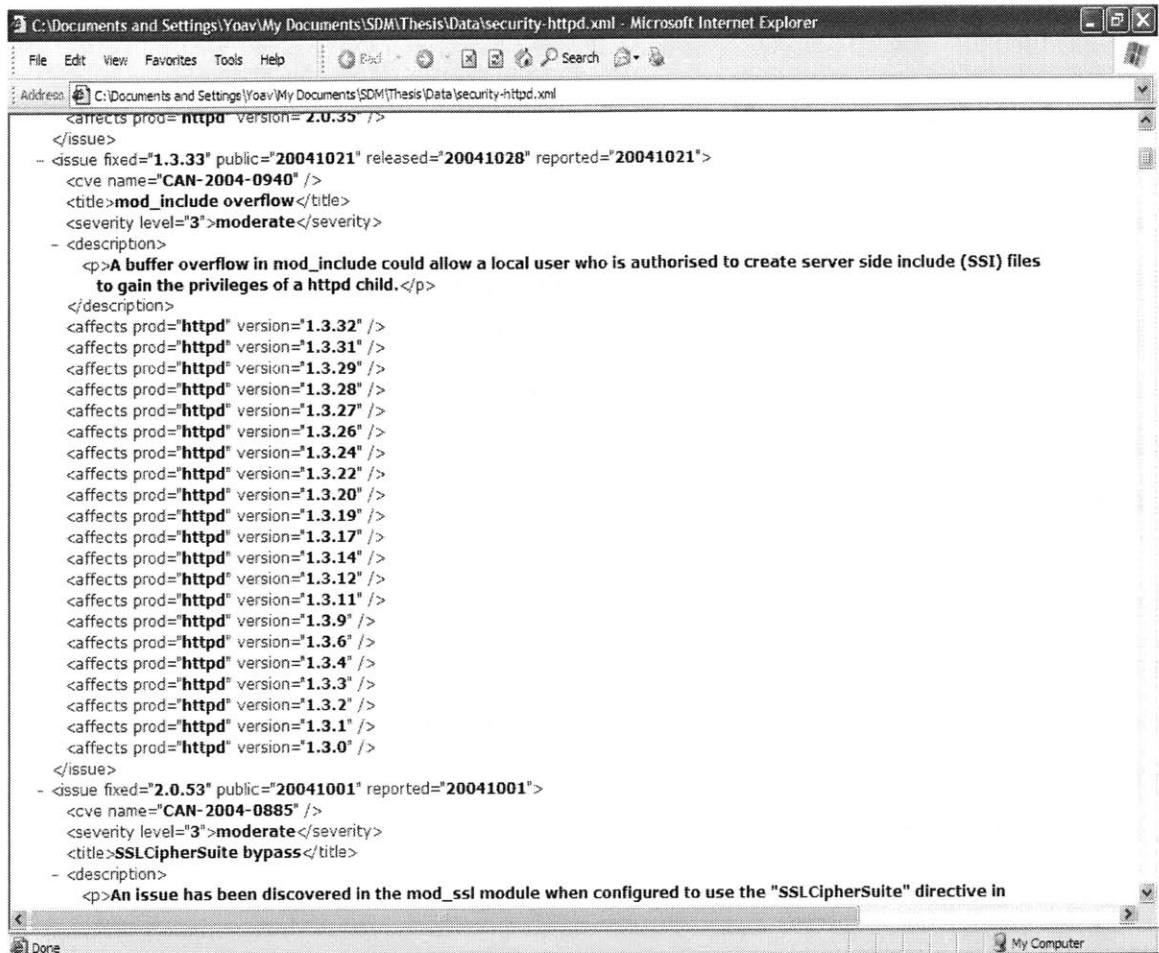
### **3.3 Apache Security Team Information**

As part of this research, the author initiated contact with members of the Apache Software Foundation's Security Team. This team consists of volunteers who participate as developers on several Apache products, including the web server. The Foundation has approximately one thousand committers, people with the technical access privileges to directly add source code to the software, but only six serve on this Security Team.<sup>4</sup> The role of this team and the Foundation's security issue handling process in general are discussed at length in Chapter 4 of this thesis.

The team was a significant source of information for this research. It provided a computerized record of security incidents including reporting and resolution dates as well as issue severity. This record is one of the best sources of Apache-specific security information. A snippet of it is shown on the next page, and the complete record is included in the Appendix.

---

<sup>4</sup> As of this writing, the six committers are Harmeet Bedi ([hbedi@apache.org](mailto:hbedi@apache.org)), Mark J. Cox ([mjc@apache.org](mailto:mjc@apache.org)), Lars Eilebrecht ([lars@apache.org](mailto:lars@apache.org)), G.W. Haywood ([ged@apache.org](mailto:ged@apache.org)), Ben Laurie ([ben@apache.org](mailto:ben@apache.org)), and Marc Slemko ([marc@apache.org](mailto:marc@apache.org)), identified on the Apache Software Foundation's committer list page at <http://people.apache.org/~jim/committers.html>.



**Figure 8 - Apache Security Team Records (Abbreviated Sample)**

As part of this thesis, a computer program was written to analyze the raw log records and convert them into a format suitable for reading and analysis by Microsoft Excel. The code for this program, written in the Java programming language, is included in the Appendix.

While these files are an excellent data source, they are incomplete and sometimes inaccurate. As part of this research, the author assisted the Apache Security Team in correcting and completing this file using his findings from other data sources.



### **3.3.1 What is the Core Product Team?**

As part of this research, contributors to the Apache web server product split into the “core product team” and external contributors. The core product team is a dynamic group of people within this project, like all other Apache projects and most open-source software development communities in general. While various groupings can be used to describe the core product team, this thesis adopts two fairly simple criteria.

At any given time, the core product team members at any given time are those who are already Apache committers, people with the technical privileges to submit source code directly into the product code repository, as listed in (Jagielski 2005). In addition, to ensure a person was otherwise active on the product, he or she must have made at least one other code contribution during the year preceding the security incident in question in order to be considered part of the core product team. These contributions can be verified by accessing the Foundation’s source code repositories as described in the following section.

Other contributors who help in resolving security issues are referred to as “3<sup>rd</sup> party” or “external contributors” as appropriate. They were not part of the core product team at the time of the specific security issue in question. They may have become part of the core product team since then, due to additional contributors and the normal dynamics of the open-source software development community. Conversely, they may have contributed only one fix and since disappeared entirely. It would be interesting to research whether contributors to security issues are more likely to become core committers over time than normal contributors, but this question is left as further work following this thesis.

### 3.4 Source Code Repositories

Open-source software development organizations, including the Apache Software Foundation, use publicly-accessible repositories to hold their software source code. The Apache Software Foundation uses two such systems: the Concurrent Versions System (CVS) and Subversion. Some of the data for this research came from the records held in these source code control systems, which record every change to the software. Specifically, developers frequently comment on changes they are making, and in the case of security issues, they sometimes identify the issue they are working on. The screenshot below illustrates one such example.

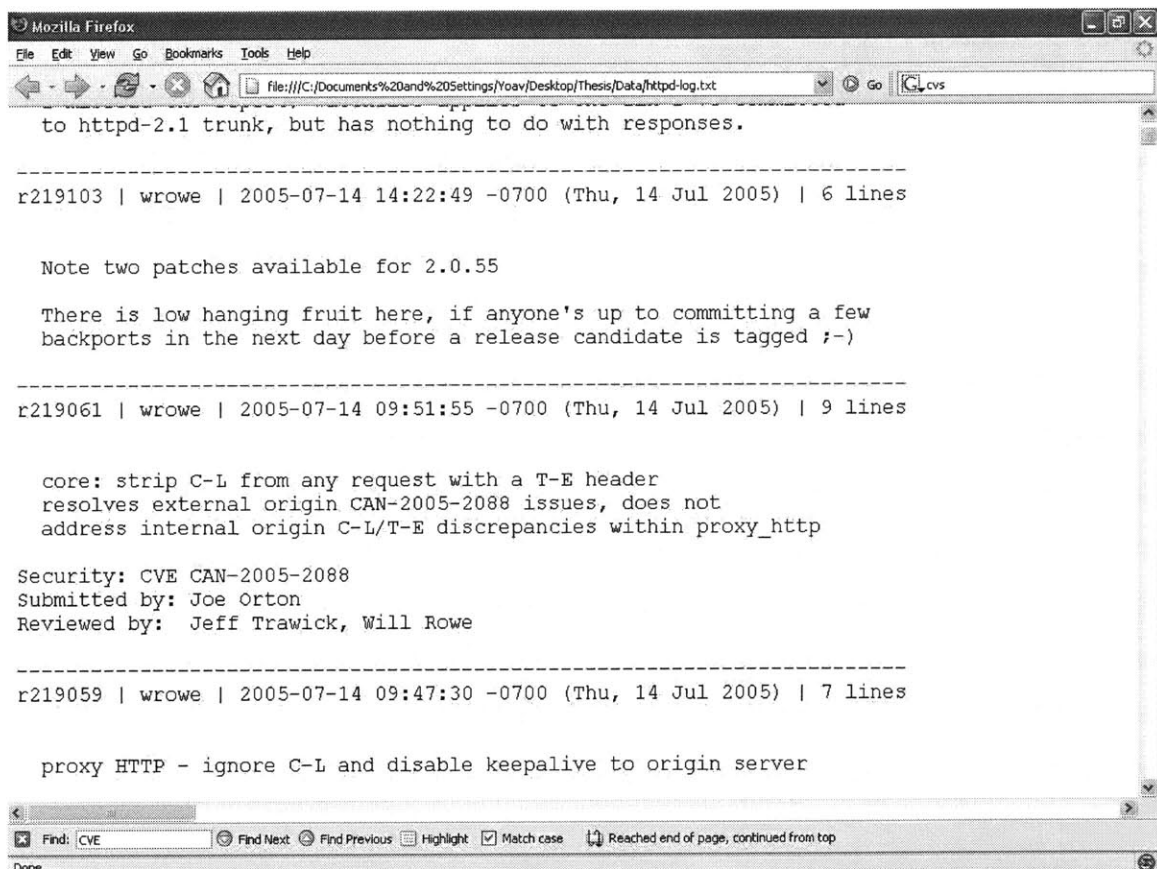


Figure 9 - Apache Web Server Commit Log Example: Fixing CVE CAN-2005-2088

CVS is one of the original open-source code repository systems and is the most widely used by open-sourced projects (Price and Ximbiot 2005). Between 1995 and 2002, it was the only system used by the Foundation.

Subversion is a newer source code control system intended to replace CVS. It contains virtually all the same features as CVS and more convenience tools for administrators and users alike (Tigris 2005). The Foundation took care to preserve historical records when migrating from CVS to Subversion, making this research easier than it would have been otherwise.

Also included in the source code repositories are change logs for the Apache web server product. These files contain a list of major new features and changes in each product version and sometimes contain further information as to who reported or fixed the issue. An example section of such a file is shown below.

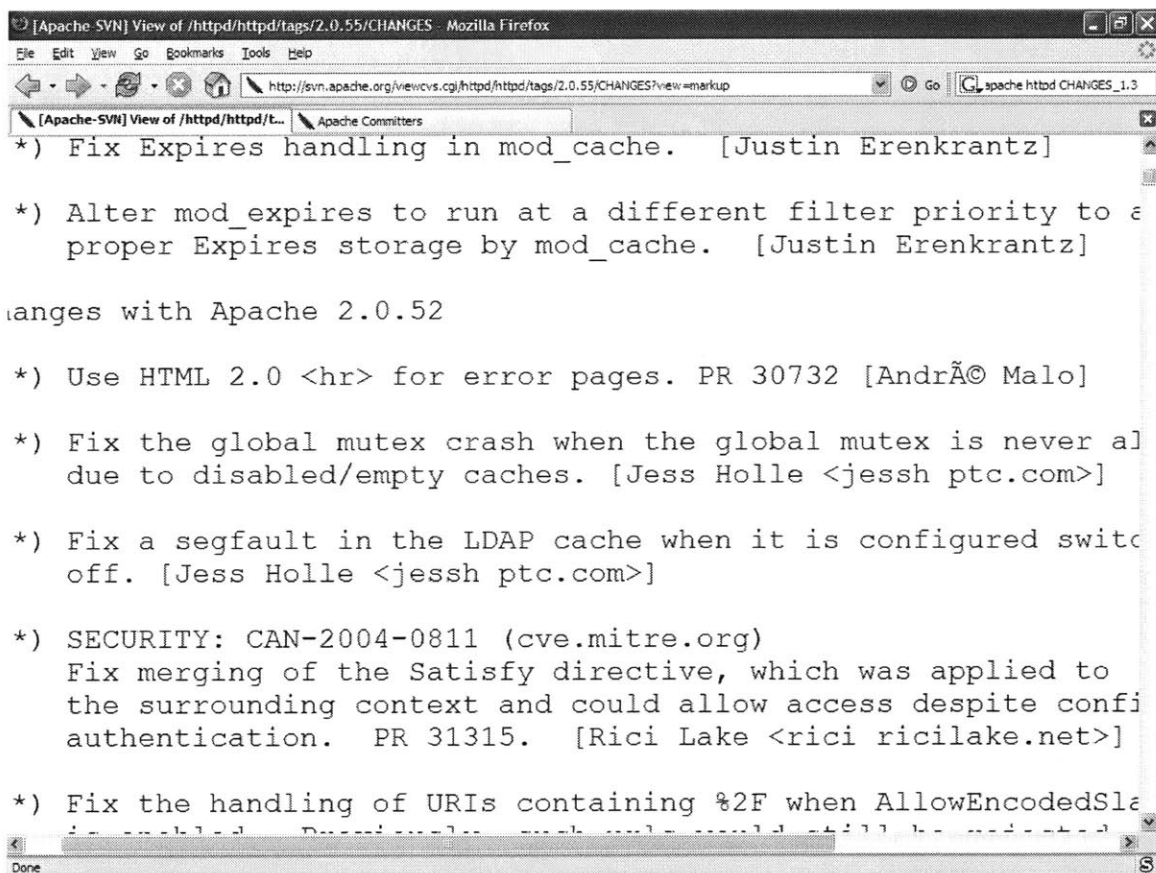


Figure 10 - Sample Change Log showing Fix for CVE-2004-0811

In the previous screen shot, the change log shows that Rici Lake fixed this security issue. Further investigation and cross-referenced data sources such as the CVE database showed that Rici Lake was also the original reporter of this issue. Therefore we conclude that this is one of the cases showing a user-led issue discovery and resolution process.

### 3.5 Product Release Announcements

Every time a new version of the Apache HTTP server product is released to the public, an announcement is made on the relevant mailing list. This announcement lists major new features added and significant bugs fixed. Particular attention is paid to any security issues that may have been addressed in this release. These release announcements, a sample of which is shown below, constitute the best source for establishing the level of user involvement in fixing a specific security issue.

For example, the screenshot below shows that the security issue identified as CVE-2003-0245 was addressed in Apache version 2.0.46, the issue was originally reported by David Endler (an employee of iDefense Labs, a computer security research and consulting firm), and the fix was contributed by Joe Orton (an employee of RedHat, a Linux solutions provider).

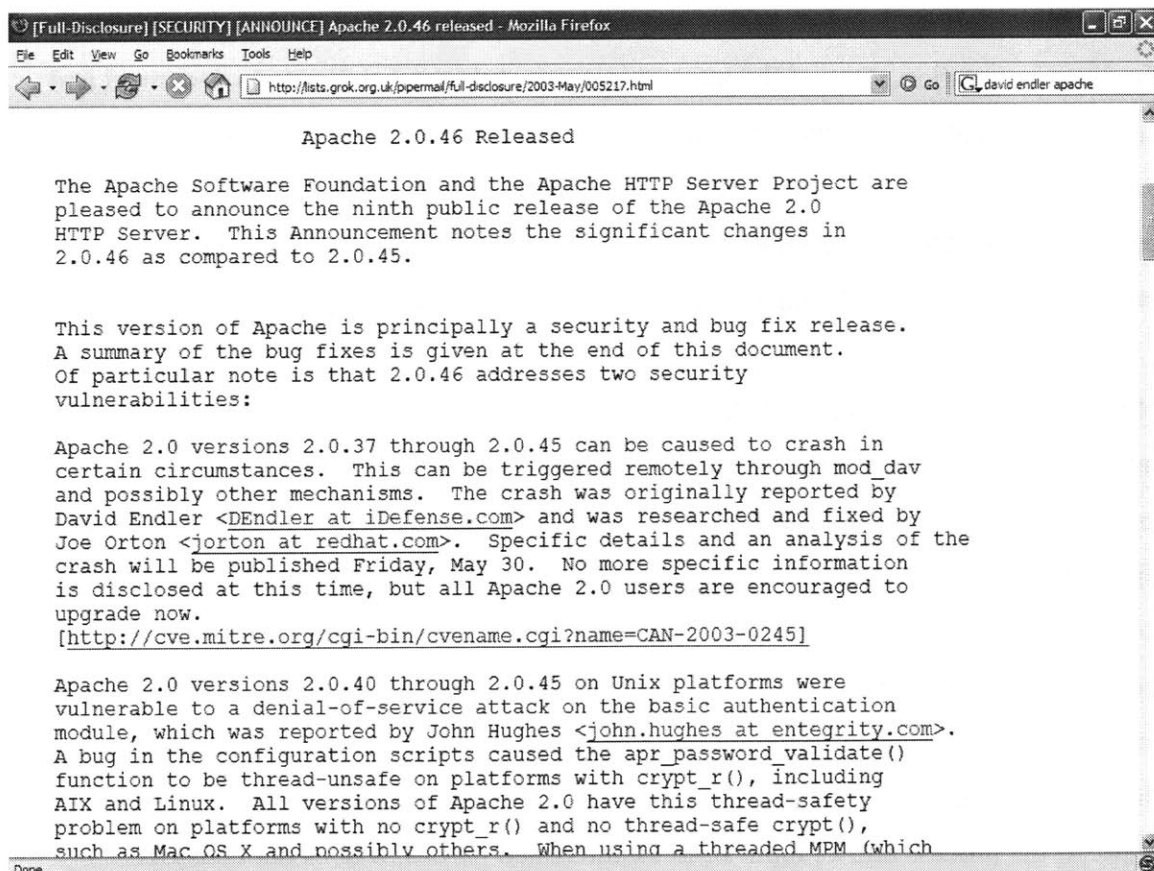


Figure 11 - Product Release Announcement Example

## Chapter 4: Results and Analysis

*“However beautiful the strategy, you should occasionally look at the results.”*  
– Sir Winston Churchill

This chapter presents both the data collected during the research and an analysis of that data. The data collected is fairly voluminous, so summaries or excerpts will be used as appropriate; additional detailed data is available in the Appendix. The results are presented in light of the theoretical assumptions of the threat-rigidity hypothesis, discussed in the previous chapters, with a focus on answering the research questions of this thesis.

The findings are divided into the following sections. First, the Apache Software Foundation’s security issue resolution process is discussed in detail. The process includes multiple optional steps that may be followed depending on the specific details of the issue in question. One of the main objectives of this research was to document and analyze the Apache security issue handling process in order to understand its benefits as well as its shortcomings. Specifically, the question is whether the process contributes to, prevents, or is natural with respect to the restriction of information flow and centralization of decision-making aspects of the threat-rigidity hypothesis.

Next, the types of issues reported, their sources, the actions of the team, and participant involvement in the process are explored. Because the server is so widely used in diverse and heterogeneous environments, we expect a broad range of types of issues and some diversity in how these issues are found initially. It will be interesting to see how much work is performed by the reporter of the issue in finding, discussing, and solving it, and how much of the work is done by other volunteers. This participation of issue reporters as well as third parties in resolving security issues speaks to the centralization of decision-making during this stressful process as well as to the restriction in information flow. If the process allows input from all interested parties, it does not constrict information flow.

The frequency of new product releases will also be examined, along with the relationship between the release frequency and the magnitude of security issues addressed in the new release. It will be shown that new product versions which address serious security issues do tend to arrive faster than average for a new release, but not much faster. This observation confirms that the release process, including its community consensus requirements, is not violated or modified for security-driven product releases, thereby ensuring that decision-making is not centralized to those working on the security issue.

As previously noted, the majority of the Apache Software Foundation's data is publicly available. Some of the data, such as new product release dates and what product version addresses what security issue, is easy to find and validate. Other data, such as the severity classification of a security issue, is more subjective, but the ratings assigned by the Foundation itself will always be used to ensure data consistency. In fact, there was at least one case where this research revealed an inconsistency in the Security Team's data: these inconsistencies were reported to the Team, addressed by the Foundation, and the updated data was used in this research.

## **4.1 The Apache Security Process**

This section describes the process used by the Apache Software Foundation to handle reports of security issues related to its products. While virtually every for-profit software organization has a process in place for handling security reports, not all open-source communities do. The Foundation has developed and refined its process over the past decade, partially in anticipation of widening product usage and increasing consequences for security issues, and partially in response to past accidents and mishandling thereof. It is important to understand this process in order to analyze its efficiency and efficacy in dealing with security issues, restriction of information flow, centralization of decision-making, and other threat-rigid behaviors.

The original process for handling security issues at the Foundation was neither well-defined nor well-documented. The initial group of contributors was small, consisting of only eight engineers, and each contributor was familiar with a large percentage of the source code. Accordingly, each person could help in analyzing security issues and fixing them quickly. However, as the group of contributors grew along with the size of the code base, the need for a better process became apparent. The current process was established in the late 1990s by several of the original founders of the Foundation. It has been proven adequate over the past eight years, especially in the face of rapid Foundation growth and product diversification.

The security process is different from most other Apache processes, and indeed from most other open-source development processes, in two key ways. First, it is more centralized than other processes: there is one Apache Security Committee (ASC) responsible for coordinating all Apache security responses. The membership of this committee, listed previously in this thesis, consists of senior core members who have typically been involved with the Foundation for several years. The members tend to have an understanding of numerous Apache products, not just their own. This is again different from most open-source processes, whereby contributors are primarily involved with one product that they use routinely.



The other key difference between the security process and other processes at Apache is its privacy concerns. Because the information discussed during this process presents a potential security exploit, which would impact millions of web sites, it is kept private. The discussions take place on lists which are not open to the public, and the information is made available to the public only once a fix has been made. While the original process was more open, it also facilitated rapid exploits of reported security issues in the form of hackers attacking vulnerable systems before the administrators could update the Apache product, so the revised process is significantly more closed. However, an emphasis is placed on including all relevant stakeholders in the decision-making process: the original reporter, the entire security team, the entire product team, and sometimes key third parties such as those working for external threat databases.

The following is a detailed description of the process, which has been proven adequate in the face of the Apache products' explosive user base growth as well as the Foundation's diversifying product line. The stage numbers in the description correspond to the numbers in the process flow diagram following the detailed description below.

Stage Number	Stage Description
1	<p>A possible security issues is reported to the Apache Software Foundation via an electronic message to the designated address, <a href="mailto:security@apache.org">security@apache.org</a>. This address is monitored by all members of the Apache Security Committee (ASC), and its contents or archives are not open to detailed public review.</p> <p>The report may come from a user stumbling upon the issue, a paid consultant conducting a product security audit, or even another Apache developer who happened to notice suspect code or behavior while working on another section of the product. Section 4.2 of this thesis examines the distribution of security issue sources.</p> <p>Please note that there is no standard template or web page for the reporter to</p>

	<p>fill out. The reports come in as free-form electronic mail messages. While this can be construed as a weakness in the process, the average technical competency level of issue reporters is high enough that he or she includes much of the relevant information a-priori without a need for further prompting. Once an issue is made public, it is typically entered into the official product issue tracking software, where detailed information about the affected operating systems and other details are recorded.</p>
2	<p>The ASC acknowledges the issue report by sending an electronic message back to the reporter. The ASC requests that the issue be kept private until further notice, in order to allow for further diagnosis and repair efforts as necessary. If any relevant information is missing, such as what product version the reporter was using, then the ASC requests that this information be submitted as well.</p>
3	<p>The ASC rejects any obviously inappropriate issues. This includes spam messages to the <a href="mailto:security@apache.org">security@apache.org</a> address, messages not related to security issues, messages related to issues which have been publicly addressed in the past, and messages not related to Apache products at all.</p> <p>These messages are silently ignored, and the reporter is usually not provided any feedback. While this may seem unprofessional, it has been deemed necessary in order to not overwhelm the volunteers on the ASC.</p>
4	<p>Once the issue is accepted by the ASC as relevant, further diagnosis and triage are conducted. The ASC may request additional details from the issue reporter. The primary goal of the ASC at this stage is to identify the relevant product or product section and therefore the relevant developer team to contact. The ASC is not expected to fix security issues by itself, but it is expected to conduct sufficient triage to locate the correct developers for fixing the issue.</p> <p>Once the relevant product is identified, the ASC forwards all the available information on the issue, including the original issue reporter's identity and</p>

	<p>contact information, to the product team.</p> <p>For some products, the forwarding address is the Project Management Committee (PMC), while other products have dedicated security addresses, such as <a href="mailto:security@tomcat.apache.org">security@tomcat.apache.org</a>. Each product team must provide the ASC with a forwarding address and is accountable for its own security issue resolution.</p> <p>Once the issue is reported to the product team, that team conducts an in-depth analysis using all available information. The team attempts to reproduce the issue reported and may ask the original reporter for more information. The product team makes one of three possible decisions: the issue is invalid, the issue has been addressed already in new versions of the software, or the issue is valid and further work is necessary.</p>
5	<p>One possible decision for the project team is to reject the issue as having been addressed already in versions of the software that followed the reporter's version. This happens when the user is using an old version, or when a fix was independently made a short time before the issue was reported to the product team. This is the least frequent of the three possible product team decisions described in stages 5, 6, and 7.</p> <p>When this happens, the original reporter is encouraged to update to the latest version as soon as possible, but typically no further action is taken. If the issue had been previously leaked to the public, then an announcement is made with the same message, urging everyone to update their product versions.</p>
6	<p>The product team rejects the issue as invalid. As mentioned above, the ASC has some knowledge of most Apache products, but it is not always detailed operational knowledge, and it may be slightly out of date. Accordingly, issues occasionally slip through the ASC that the product team is able to dismiss as invalid.</p>

	<p>One common reason for an issue to be declared as invalid stems from inappropriate usage of the product. For example, a user reporting a security issue caused by using a 3<sup>rd</sup> party extension to an Apache product would see that issue rejected as inappropriate, and the product team would suggest that the user contact the 3<sup>rd</sup> party library provider so that it may fix the product. If the extension is widely used, the product team may announce this issue on the public mailing list and it may also assist the 3<sup>rd</sup> party provider in addressing this issue. However, the official Apache process would be over at this point.</p>
7	<p>If the product team verifies that the issue is valid and that it has not been fixed in other versions, work proceeds towards fixing the issue. Again, the issue reporter may be contacted for more details, but usually the team knows the code well enough to proceed without further assistance.</p> <p>If particular developers are associated with the section of the product where the issue is reported, these developers are often contacted by the product team and asked to assist with the process. Sometimes these developers have been inactive for a prolonged period of time, but as the original authors of parts of the code, their opinion is respected and they are also the most qualified to fix the issue.</p> <p>Frequently, it is during this step that these more veteran, but less involved, core team members become temporarily involved in order to help fix the issue.</p> <p>The product team notifies the ASC of its plan and progress.</p>
8	<p>The ASC continues to remain in contact with the issue reporter, updating him or her on the progress to date and continued plans for addressing the issue. At this point, the ASC may contact external parties that track security issues,</p>

	such as the Computer Emergency Response Team (CERT) at Carnegie Mellon University, <sup>5</sup> or the Common Vulnerabilities and Exposures (CVE) initiative at MITRE. <sup>6</sup> These organizations assign a case number to the security issue for further tracking, communication, and archival purposes.
9	In parallel to the ASC activities in stage 8 above, the product team works on fixing the issue. Once a code fix is available, it is first tested by the product team. If the fix is found satisfactory, the product team notifies the ASC of the results. The fix is also committed into the project's source code repository, where lead users who track the repository may download and use it before any formal announcement is made.
10	The ASC or the product team may request the original reporter to help with verifying the fix. The ASC may also advise the same third parties (CERT and CVE) of the progress to date, indicating a fix is available. This is an important communications measure, because many organizations that use Apache products rely on these third parties to disseminate security updates to them.
11	After updating the source code repository and notifying the ASC, the product team issues a new official release of the product. The release is announced on public mailing lists, with special emphasis given to the security issues addressed. Credit is given to the issue reporter as appropriate, and users are encouraged to update to the latest version as soon as possible.

---

<sup>5</sup> For more details on CERT, please see Chapter 3.

<sup>6</sup> For more details on CVE, please see Chapter 3.

# The Apache Security Process

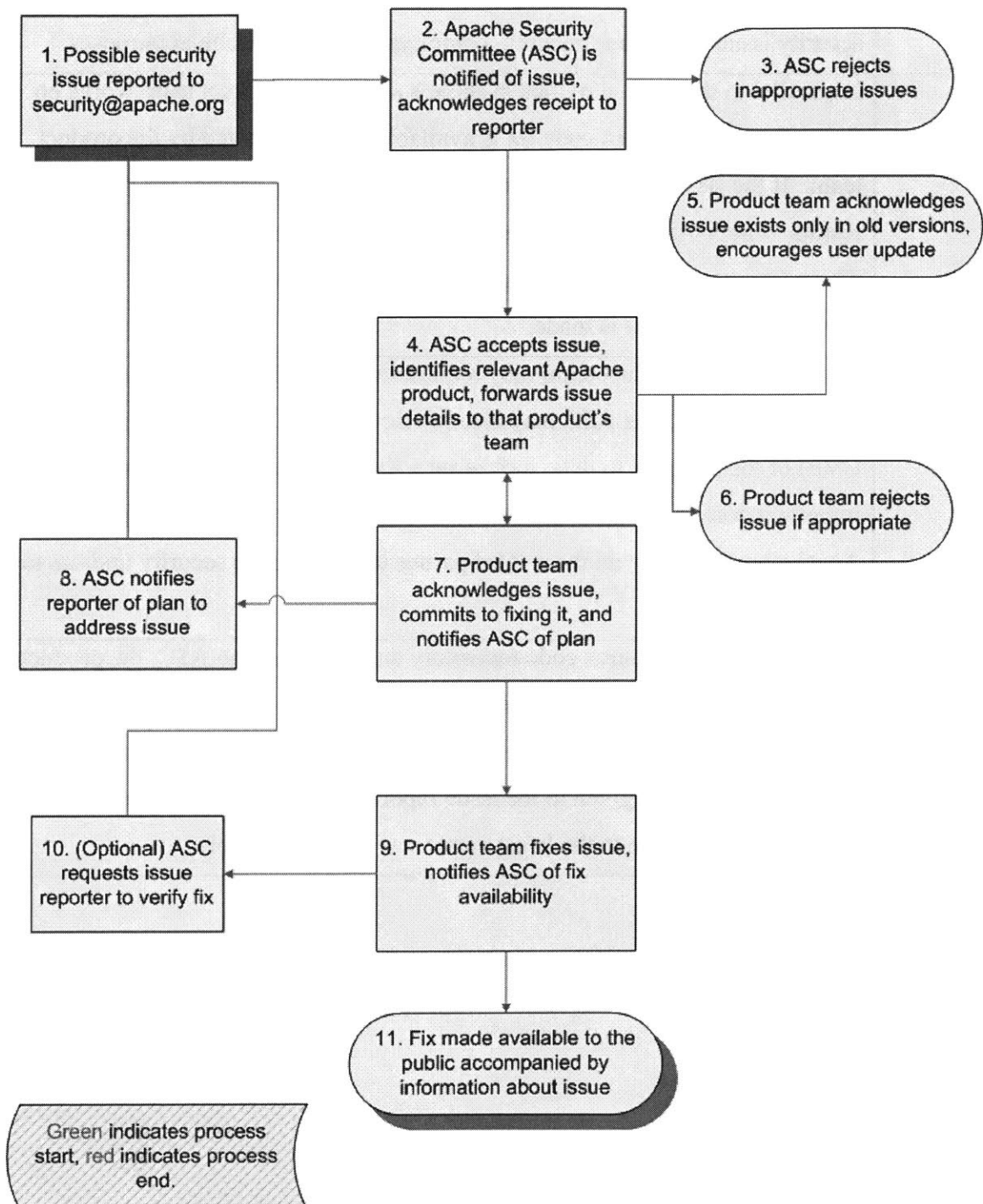


Figure 12 - Apache Security Process

The process is fairly efficient. Step 2, notification of the security team and an acknowledgement to the issue reporter, is virtually always completed within a few hours of the reporter's message to the team.

Steps 3 and 4 typically require a day or two for analysis and triage. The team is careful to assess each issue thoroughly prior to rejecting it. The process for identifying the relevant Apache product is fairly straightforward, as most of the issues are reported against the web server itself, and the remaining issues are evenly split among less than a handful of other projects that can have server-level security issues.

There is further variance in steps 5, 6, and 7, depending on the project team availability and the complexity of the issue. These steps can take a few hours, typically in the case where the issue is invalid, or a few days, where the issue requires deeper analysis or is difficult to reproduce. Cases that require further information from the issue reporter take longer.

Step 9, actually fixing the issue, also has high variance, taking anywhere from a couple of hours to a week. Some fixes are as trivial as correcting the documentation or removing a misleading comment in the code, but others involve significant code changes to key data structures in multiple modules of the software. Accordingly, testing these patches can also be a significant effort, sometimes requiring as much work as the fixing effort itself.

Finally, a new product release is issued containing the fixes as appropriate. For critical or important issues, a release is typically made as soon as the fix is available. For less important releases, the code is fixed in the repository, but no release may be made until additional features and fixes are implemented. In these cases, users are free to download and build the code from the repository, but there is no official product release made. The cases of users downloading specific patches as opposed to official releases are not included in the analysis for this thesis, as they are assumed to be a negligible fraction of the overall product adoption.

During the entire process, the original reporter and entire product team are kept in the loop. While the Security Team steers the process, it does not make technical decisions on behalf of the product team, and these decisions are made using the normal voting and consensus guidelines of the Apache Software Foundation. Accordingly, the process does not lead to excessive centralization of decision-making, and this aspect of the threat-rigidity hypothesis does not hold in this context.

However, when looking at the restriction of information flow aspect, we observed that virtually all others Foundation processes are completely open to the public. The security issue handling process is not open to the public, and although it is closed for good reasons, it still represents a restriction in information flow. This restriction forfeits many of the benefits of the open-source software development community, such as having more people look at every code fix or issue, but the Foundation believes this sacrifice is necessary in order to mitigate the impact of a security issue becoming publicly known prematurely. Therefore, although there technically is a restriction in information flow compared to normal Foundation processes, we do not find it to be the same type of restriction predicted by the threat-rigidity hypothesis.



## **4.2 Security Issue Severity Classification**

There have been more than 1,000 security issues reported to the Apache Security Team, judging from the archives of the [security@apache.org](mailto:security@apache.org) mailing list. Only a small portion (55, or less than 5.5%) is verified as a legitimate issue and receives a tracking number such as the one provided by the CVE database described in Chapter 3. The mailing list archives, which are private and cannot be published as part of this thesis, contain a large amount of spam messages: 98% according to a recent estimate conveyed in private discussions with one Security Team member. The team sifts through these messages without comment, focusing instead on the actual or apparent issues reported.

Like other software organizations, the Foundation has developed a classification scheme for security issues according to their severity. The classification serves two purposes: it focuses the attention of developers should simultaneous issues arise and helps convey the significance of issues to users. The latter is especially important when issues become publicly known before a fix is available for them, and when a fix is available but requires a substantial upgrade effort. Some server administrators will not upgrade their installation unless an issue falls in the “Critical” or “Important” categories. For our purposes, the classification serves to divide security issues into “serious” or not and for statistical analysis purposes: the effects of threat-rigidity should be most pronounced for the most serious security issues and less evident for less serious issues.

The Apache Software Foundation’s security classification is repeated below (Cox 2004):

Severity Category	Description
Critical	A vulnerability rated with a Critical impact is one which could potentially be exploited by a remote attacker to get Apache to execute arbitrary code (either as the user the server is running as, or root). These are the sorts of vulnerabilities that could be exploited automatically by worms.
Important	A vulnerability rated as Important impact is one which could

	result in the compromise of data or availability of the server. For the Apache web server this includes issues that allow an easy remote denial of service (something that is out of proportion to the attack or with a lasting consequence), access to arbitrary files outside of the document root, or access to files that should be otherwise prevented by limits or authentication.
Moderate	A vulnerability is likely to be rated as Moderate if there is significant mitigation to make the issue less of an impact. This might be because the flaw does not affect likely configurations, or it is a configuration that isn't widely used, or where a remote user must be authenticated in order to exploit the issue. Flaws that allow Apache to serve directory listings instead of index files are included here, as are flaws that might crash an Apache child process in Apache 1.3.
Low	All other security flaws are classed as a Low impact. This rating is used for issues that are believed to be extremely hard to exploit, or where an exploit gives minimal consequences.

In this research, we use the classification in two ways: to categorize security issues and to categorize product versions as “security-driven” or not. A “security-driven” product release is defined as one addressing at least one issue of critical or important severity.

The complete record of security issues that have been given a CVE tracking number is below.<sup>7</sup>

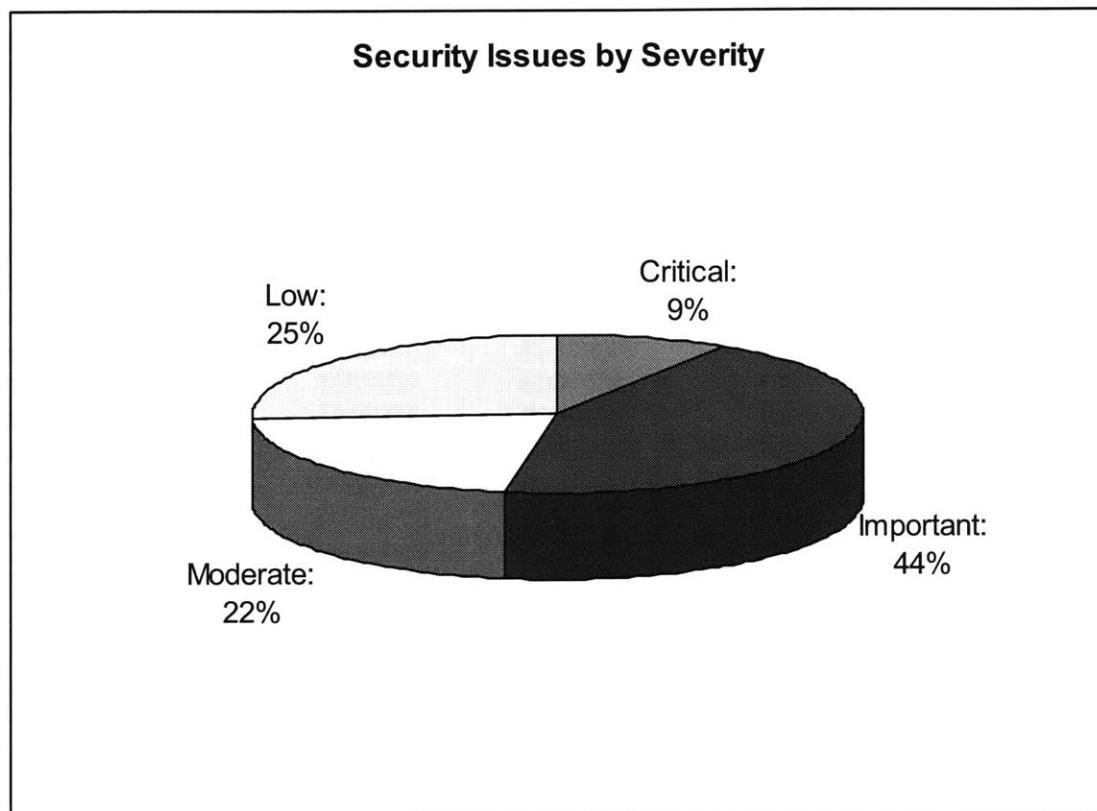
Issue ID	Date Reported	Date Made Public	Date Resolved	Days to Resolution	Severity
CVE-2005-2970		9/19/2005	10/14/2005	25	Moderate
CVE-2005-2728	7/7/2005	7/7/2005	10/14/2005	99	Moderate
CVE-2005-2700	8/30/2005	8/30/2005	10/14/2005	45	Important
CVE-2005-2491		8/1/2005	10/14/2005	74	Low
CVE-2005-2088		6/11/2005	10/14/2005	125	Moderate
CVE-2005-1268	5/26/2005	5/26/2005	10/14/2005	141	Critical
CVE-2004-1834	3/2/2004	3/20/2004	2/8/2005	343	Low
CVE-2004-0942	10/28/2004	11/1/2004	2/8/2005	103	Important
CVE-2004-0940	10/21/2004	10/21/2004	10/28/2004	7	Moderate

<sup>7</sup> See Chapter 3 for an explanation of the CVE system.

CVE-2004-0885	10/1/2004	10/1/2004	2/8/2005	130	Moderate
CVE-2004-0811	9/18/2004	9/18/2004	9/28/2004	10	Important
CVE-2004-0786	8/25/2004	9/15/2004	9/15/2004	21	Critical
CVE-2004-0751	7/7/2004	7/7/2004	9/15/2004	70	Low
CVE-2004-0748	7/7/2004	7/7/2004	9/15/2004	70	Important
CVE-2004-0747	8/5/2004	9/15/2004	9/15/2004	41	Low
CVE-2004-0493	6/13/2004	7/1/2004	7/1/2004	18	Important
CVE-2004-0809	9/12/2004	9/12/2004	9/15/2004	3	Low
CVE-2004-0492	6/8/2003	6/10/2003	10/20/2004	500	Moderate
CVE-2004-0488		5/17/2004	7/1/2004	45	Low
CVE-2004-0174	2/25/2004	3/18/2004	5/12/2004	77	Important
CVE-2004-0113	2/20/2004	2/20/2004	3/19/2004	28	Important
CVE-2003-0993	10/15/2003	10/15/2003	5/12/2004	210	Important
CVE-2003-0987	12/18/2003	12/18/2003	5/12/2004	146	Low
CVE-2003-0789	10/3/2003	10/27/2003	10/27/2003	24	Moderate
CVE-2003-0542	8/4/2003	10/27/2003	10/27/2003	84	Low
CVE-2003-0460	7/4/2003	7/18/2003	7/18/2003	14	Important
CVE-2003-0254	6/25/2003	7/9/2003	7/9/2003	14	Moderate
CVE-2003-0253	6/25/2003	7/9/2003	7/9/2003	14	Important
CVE-2003-0245	4/9/2003	5/28/2003	5/28/2003	49	Critical
CVE-2003-0192	4/30/2003	7/9/2003	7/9/2003	70	Low
CVE-2003-0189	4/25/2003	5/28/2003	5/28/2003	33	Important
CVE-2003-0134	3/31/2003	4/2/2003	4/2/2003	2	Important
CVE-2003-0132	3/31/2003	4/2/2003	4/2/2003	2	Important
CVE-2003-0083	2/24/2003	2/24/2003	4/2/2003	37	Low
CVE-2003-0020	2/24/2003	2/24/2003	3/19/2004	389	Low
CVE-2003-0017	11/15/2002	1/20/2003	1/20/2003	66	Important
CVE-2003-0016	12/4/2002	1/20/2003	1/20/2003	47	Critical
CVE-2002-1593		9/19/2002	9/24/2002	5	Moderate
CVE-2002-1592		4/22/2002	5/8/2002	16	Low
CVE-2002-1156		9/25/2002	10/3/2002	8	Moderate
CVE-2002-0843	9/23/2002	10/3/2002	10/3/2002	10	Important
CVE-2002-0840	9/20/2002	10/2/2002	10/3/2002	13	Low
CVE-2002-0839	11/11/2001	10/3/2002	10/3/2002	326	Important
CVE-2002-0661	8/7/2002	8/9/2002	8/9/2002	2	Important
CVE-2002-0654	7/5/2002	8/9/2002	8/9/2002	35	Low
CVE-2002-0392	5/27/2002	6/17/2002	6/18/2002	22	Critical
CVE-2002-0061	2/13/2002	3/22/2002	3/22/2002	37	Critical
CVE-2001-1342		4/5/2001	5/22/2001	47	Important
CVE-2001-0925		3/12/2001	5/22/2001	71	Important
CVE-2001-0731		7/9/2001	10/12/2001	95	Important
CVE-2001-0730		9/28/2001	10/12/2001	14	Moderate
CVE-2001-0729	9/18/2001	9/28/2001	10/12/2001	24	Important
CVE-2000-1205			2/25/2000		Important
CVE-2000-1204			10/13/2000		Important
CVE-2000-0913	9/22/2000	9/29/2000	10/13/2000	14	Important
CVE-2000-0505		5/31/2000	10/13/2000	135	Moderate

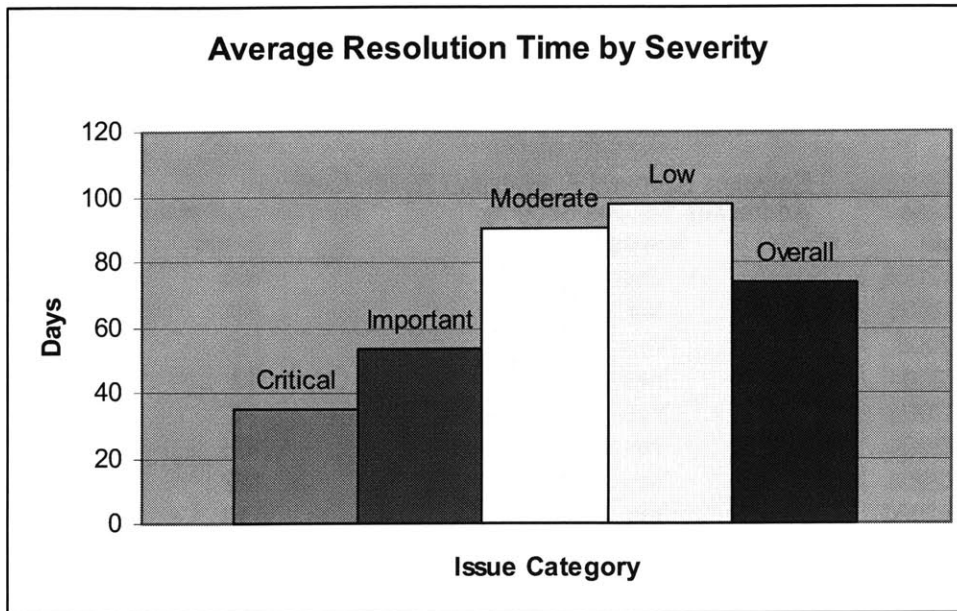
**Table 1 – Apache Security Issues, 2000-2005**

Of the 55 issues in the security records five (9.1%) are classified as critical, twenty four (43.6%) as important, twelve (21.8%) as moderate, and fourteen (25.5%) as low. This is fairly remarkable considering the product team itself is responsible for categorizing security issues: one might think that the team, having developed the buggy code itself, would down-play the significance of issues. Instead, more than 52% of issues are in the top two most serious categories.



**Figure 13 - Security Issues by Severity**

When examining the average resolution time from when the issue is reported (either privately to [security@apache.org](mailto:security@apache.org) or via public announcements) to when a release is made that addresses the issue, one sees that the response is faster when the issue severity is higher, as one would hope.



**Figure 14 - Average Resolution Time by Severity**

The average time to address critical issues is 35.2 days, important issues 53.38 days, moderate issues 90.5 days, and low issues 97.6 days, for an overall average resolution time of 74.0 days. It is interesting to note that the average resolution time for moderate- and low-severity issues is fairly similar: this reflects the fact that the low-priority issues are frequently rolled into another product version, whereas critical or important issues typically have a product release made specifically to address them once a fix is available.

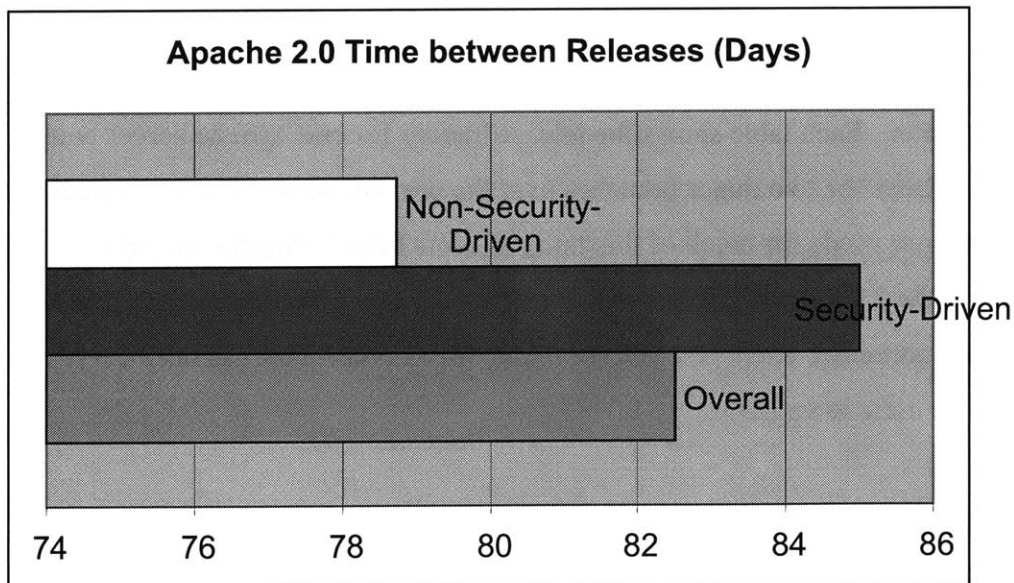
However, one might also expect security-driven product releases to be issued more frequently than standard releases. This does not appear to be the case, as shown in the following two tables. Each table shows the release history for one Apache server branch: 1.3 and 2.0 have been the two major branches over the past six years. The 1.3 branch has been in maintenance mode for much of this time, as users have been encouraged to migrate towards the 2.0 branch. However, when important security issues arise in a 1.3 version, they are addressed and a new release made; the product team has been very reluctant to force users to upgrade.

The following charts include releases intended for General Availability or a wide-scope beta test. They do not include releases that were tagged in the source code control

repository but never made available to the public, or releases that were only available for a few days and recalled due to severe issues.

<u>Apache 2.0 Releases (General Availability / Stable Quality)</u>			
Version	Release Date	Addresses Serious Security Issues	Time Since Previous Release
2.0.55	10/14/2005	Yes	180
2.0.54	4/17/2005	No	68
2.0.53	2/8/2005	Yes	133
2.0.52	9/28/2004	Yes	13
2.0.51	9/15/2004	Yes	77
2.0.50	6/30/2004	Yes	103
2.0.49	3/19/2004	Yes	142
2.0.48	10/29/2003	No	112
2.0.47	7/9/2003	Yes	42
2.0.46	5/28/2003	Yes	56
2.0.45	4/2/2003	Yes	71
2.0.44	1/21/2003	Yes	110
2.0.43	10/3/2002	No	9
2.0.42	9/24/2002	No	46
2.0.40	8/9/2002	Yes	52
2.0.39	6/18/2002	Yes	41
2.0.36	5/8/2002	No	32
2.0.35	4/6/2002	No	48
2.0.32	2/17/2002	No	93
2.0.28	11/16/2001	No	222
2.0.16	4/8/2001	No	N/A

**Table 2 – Apache 2.0 Release History**



**Figure 15 - Apache 2.0 Average Time between Releases**

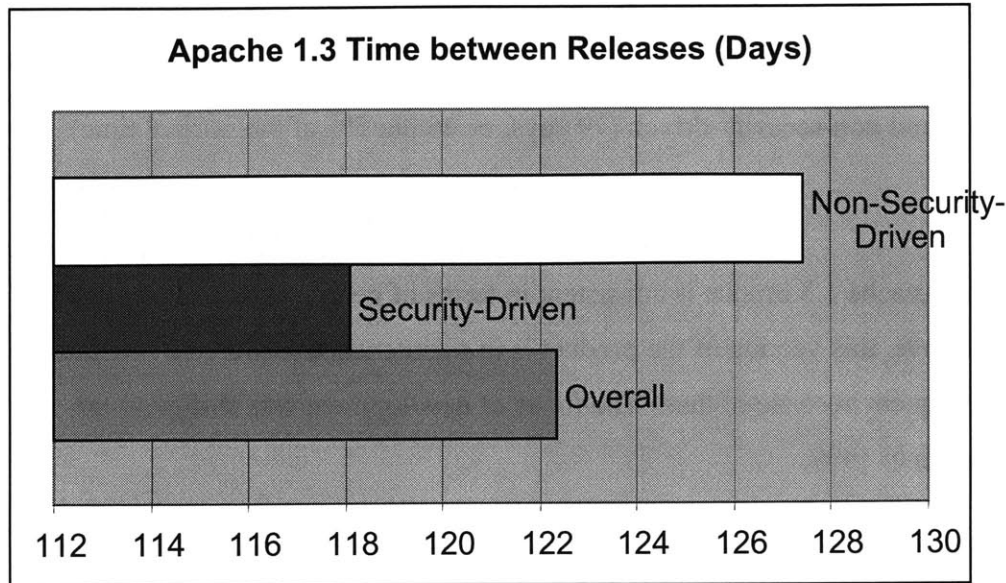
As the graph shows, there is a fairly small variance between the overall average time between releases, 83 days, and the averages for security-driven (85 days, or within 3% of the normal time) and non-security-driven (79 days, or within 5% of the normal time) releases.

The data for the Apache 1.3 branch is consistent in terms of percentages and variance. As mentioned above, this version of the product is in maintenance mode, and releases are generally less frequent because of that. The focus of development was shifted to the Apache 2.0 branch in 1999.

As with the other tables in this work, the time since previous release is measured in days.

Apache 1.3 Releases (General Availability / Stable Quality)			
Version	Release Date	Addresses Serious Security Issues	Time Since Previous Release
1.3.34	10/18/2005	No	354
1.3.33	10/29/2004	No	171
1.3.31	5/11/2004	Yes	195
1.3.29	10/29/2003	No	103
1.3.28	7/18/2003	Yes	288
1.3.27	10/3/2002	Yes	107
1.3.26	6/18/2002	Yes	88
1.3.24	3/22/2002	Yes	57
1.3.23	1/24/2002	No	104
1.3.22	10/12/2001	Yes	143
1.3.20	5/22/2001	Yes	83
1.3.19	2/28/2001	Yes	33
1.3.17	1/26/2001	No	105
1.3.14	10/13/2000	Yes	231
1.3.12	2/25/2000	Yes	35
1.3.11	1/21/2000	No	154
1.3.9	8/20/1999	No	148
1.3.6	3/25/1999	No	73
1.3.4	1/11/1999	Yes	94
1.3.3	10/9/1998	No	16
1.3.2	9/23/1998	Yes	63
1.3.1	7/22/1998	No	46
1.3.0	6/6/1998	No	N/A

**Table 3 – Apache 1.3 Release History**



**Figure 16 - Apache 1.3 Average Time between Releases**

In this branch of the software, the situation is similar to the 2.0 branch; the overall average time between releases is 122 days. The average for security-driven releases is 118 days, within 4% of the normal time, and the average for non-security-driven releases is 127 days, also within 4% of the normal time.

This suggests that security issues do not cause a panic or emergency situation within the Foundation. Issues are reported, fixes are made, and the product is released when the fix is ready. Moreover, the data shows that the normal processes of code validation, testing, and release by team consensus are all still followed for security-driven releases as well as normal releases. This negates the threat-rigidity hypothesis' prediction that due to centralized decision-making, security-driven releases should emerge significantly faster than the average release.

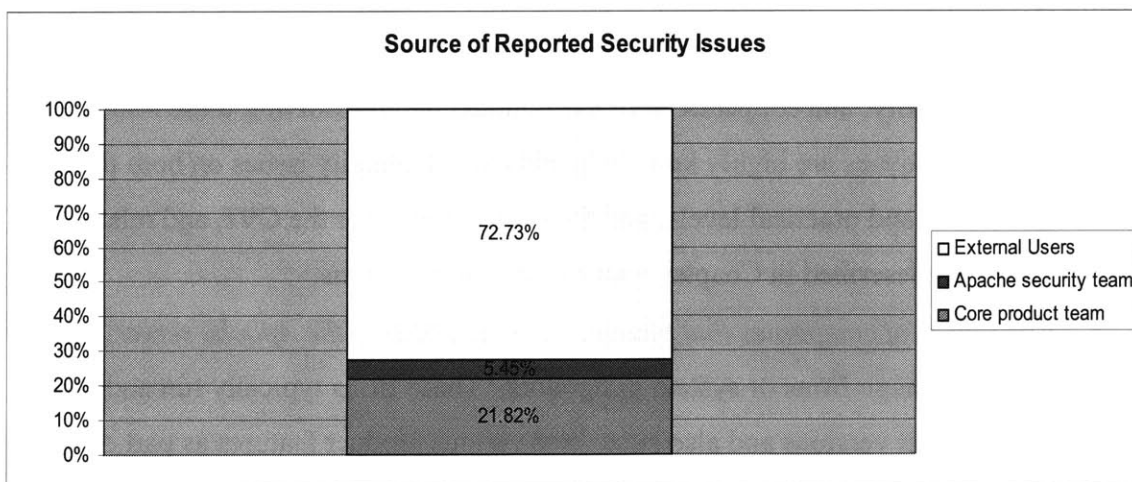


### 4.3 User Participation

As part of the research questions motivated by the user-led innovation nature of open-source software development, we examine the level of involvement of users in the security issue identification and handling process.

Users of the Apache HTTP web server are technically-savvy individuals: they download, install, and configure web servers, sometimes for mission-critical applications in complex environments. Because the product is popular and frequently used to serve content to the public from a company's front page, it is also independently tested and audited for security issues by users and third party consultants alike.

Unfortunately, identifying the reporters of security issues is not always easy or even possible. Issues are sometimes reported anonymously. Other times the team fails to disclose the identity of the reporter, whether intentionally in response to the reporter's request or unintentionally by an error of omission.



**Figure 17 - Sources of Reported Security Issues**

Out of the 55 issues that received a CVE identifier, the majority (more than 70%) were reported by external users who were not part of the Apache Software Foundation in any way. The core product team found and reported most of the remaining issues, accounting for approximately 20% of the total. The Apache Security Team itself reported two issues, possibly in the process of testing or verifying a different report.

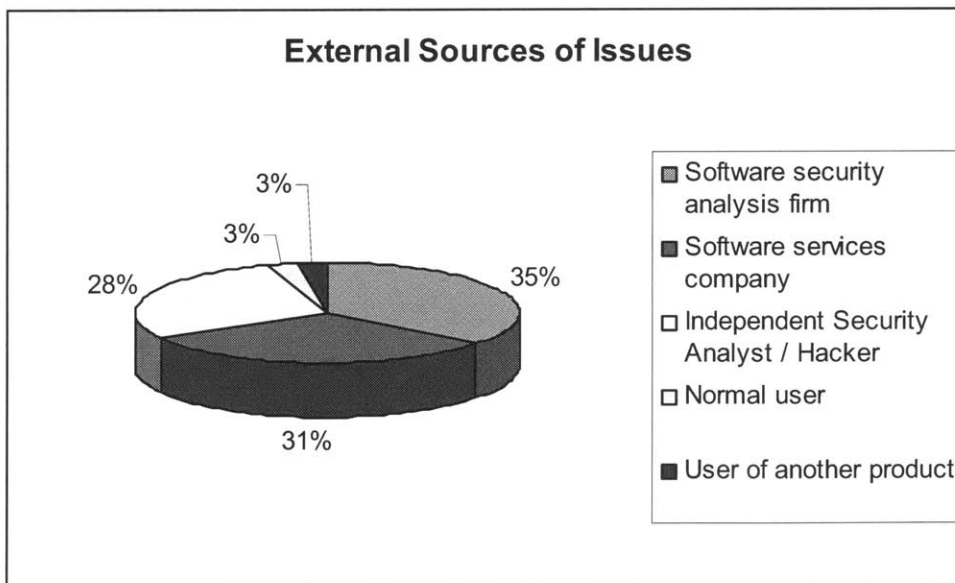
While the 20% figure may seem small, it is actually significant. One must assume that the core product team stumbles upon many real or possible security issues during regular development and testing. There is nothing to compel any team member to report these issues; a fix could be made silently and included in the next product version silently as well. However, it appears that the individuals working in the core product team have a high ethical motivation, and they choose to report in public those security issues that become visible to them, even before any users encounter the problem. Perhaps this is an indication of the team's awareness of the level of stress associated with fixing these issues once they are publicly known, or perhaps it is addressing a simple need: as mentioned previously, these developers are also strong and frequent users of the product.

The users within the "External" or non-Apache category can be broken down further according to the type of employment or activity they were doing when they found the security issue. For this analysis, they are divided into 5 groups:

- *Security analysts who are paid to conduct audits of products such as the Apache server.* These companies, such as iDefense Labs or Next Generation Security Software, routinely inspect new product releases for security issues, develop tools to test security, and cooperate with the Foundation on resolving these issues. Their employees are highly knowledgeable about security issues on both the theoretical and practical levels, and they tend to monitor the CVE and related databases described in Chapter 3 above on a regular basis.
- *Employees of companies that provide services utilizing the Apache server, such as web site design firms or system integrators.* These firms typically run audits of new product versions and also stress-test various product features as part of their normal operations. They are not, however, as dedicated to finding and reporting security issues as the analysts in the first category. Moreover, they are generally not as knowledgeable as the specialists in the security analysis firms described above.
- *Independent security analysts and "hackers."* These are people who report issues independently of any organization; they do not appear to be users of the product,

but rather hackers or independent consultants providing security assessments. For them, finding and reporting issues is partially a matter of pride, a way to establish credibility in the technical arena, or perhaps a way to draw the attention of and gain employment with security analysis firms. They are geographically dispersed: this research found issues reported by hackers in Russia, England, Italy, China, and the United States.

- *Normal users who just happen to run across an issue while using the product.* Again, “normal users” of the Apache web server are typically experienced technical professionals who have set up the server for their organization. They are webmasters or server administrators, and they typically conduct a limited set of tests when deploying new product versions.
- *Users of another product.* Because the Apache server is a central infrastructure component for many applications, sometimes users note bad behavior in those applications and trace it to the server. This is a fairly difficult debugging effort, so this category is typically very small.



**Figure 18 - Breakdown of External Sources of Issue**

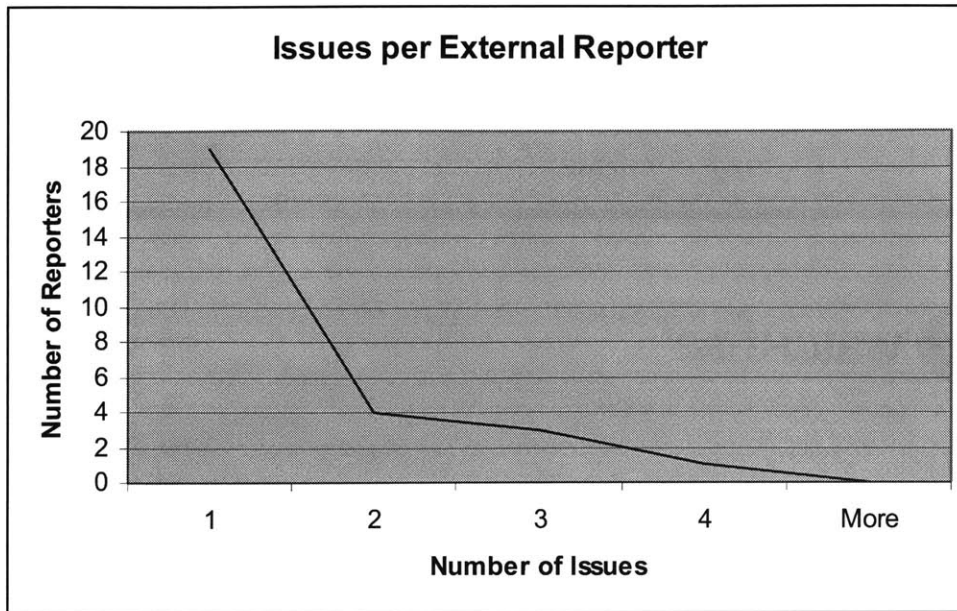
As the breakdown diagram above shows, software security firms are the leading reporters of security issues in the Apache web server, with approximately 35% of the issues. This is not surprising: in recent years, the number of these firms and their level of

sophistication have greatly increased. They employ various automated tools as well as experienced analysts with complete knowledge of historically buggy areas in the product. These companies conduct some audits on their own, and other times they are hired by a third party that relies on the product to independently test it.

The next leading sources of issues are software services companies and independent hackers. While neither typically has the resources (time or money) to spend on security auditing that the dedicated analysis firms do, both of these groups have strong motivations to find and report issues. Software services companies provide consulting and support services to clients that use the Apache product: it is in their financial interest to resolve these issues and do so quickly, lest they lose clients to a competing platform. Independent consultants and hackers have a different interest in finding and reporting issues: they want to gain recognition and credibility, which typically leads to employment with the security analysis firms.

This level of involvement from external users is encouraging in light of the threat-rigidity hypothesis, because it shows that the team does not restrict information flow to itself. Rather, input from external users is welcomed and taken into full consideration.

Looking at the statistics for external reporters, and counting all reports from one security analysis firm as one source even if they come from different analysts, one sees the following:



**Figure 19 - Issues Reported by External Reporters Histogram**

Most reporters only report one issue: 19 of them. In fact, the median for issues reported per reporter is 1.0, although the average is 1.48. Four people have reported two issues each, three people three issues each, and one person has reported four issues: that “person” is actually a security analysis firm, iDefense Labs.

It is somewhat surprising that “normal users” only reported a small minority of issues. There are two complementary explanations for this observation. The first is that the Apache product is technically mature and extremely well tested by millions of users over the past decade; accordingly, serious security issues are not only rare but they only surface in highly unusual configurations or environments. By definition, only a few users run these configurations, so the opportunity for these bugs to arise is low. This explanation is partially corroborated by the nature of recent security issues: they tend to be highly specialized configurations set up by security analysts.

For an illustration of three types of reports, consider the following issues. First, a report from a security analysis firm, Watchfire, regarding a theoretical security issue called “HTTP Request Smuggling.” The report was submitted to [security@apache.org](mailto:security@apache.org) as requested, and the normal process was followed to fix the server before the report was

made public. The following two screenshots are just the cover sheet and table of contents of a 23-page analysis complete with examples and tests (Linhart et al 2005).

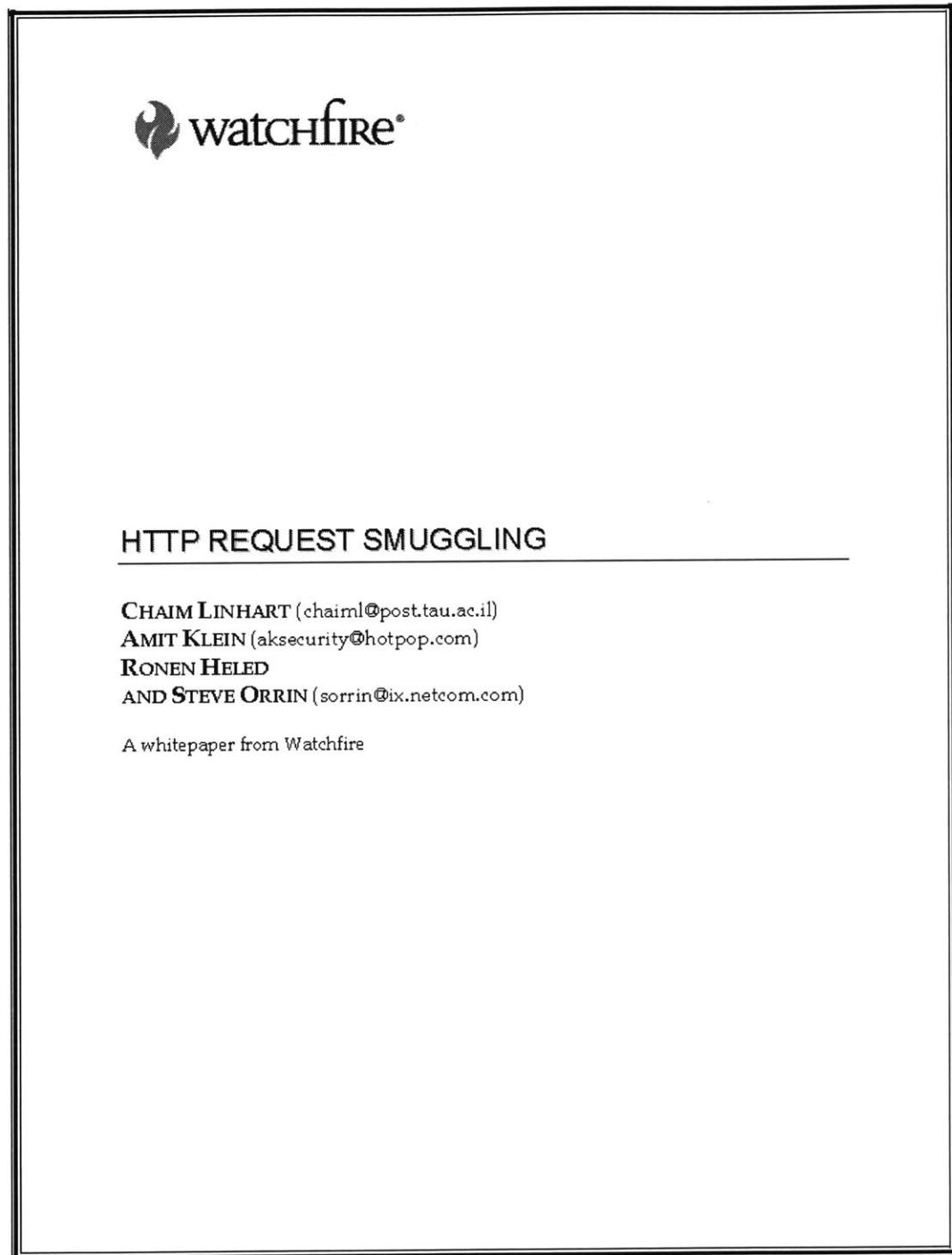


Figure 20 - Security Analyst Issue Report Sample Page 1

## TABLE OF CONTENTS

<b>Abstract .....</b>	<b>1</b>
<b>Executive Summary .....</b>	<b>1</b>
<b>What is HTTP Request Smuggling?.....</b>	<b>2</b>
<b>What damage can HRS inflict?.....</b>	<b>2</b>
<b>Example #1: Web Cache Poisoning .....</b>	<b>4</b>
<b>Example #2: Firewall/IPS/IDS evasion .....</b>	<b>5</b>
<b>Example #3: Forward vs. backward HRS .....</b>	<b>7</b>
<b>Example #4: Request Hijacking .....</b>	<b>9</b>
<b>Example #5: Request Credential Hijacking.....</b>	<b>10</b>
<b>HRS techniques.....</b>	<b>10</b>
<b>Protecting your site against HRS .....</b>	<b>19</b>
<b>Squid .....</b>	<b>19</b>
<b>Check Point FW-1.....</b>	<b>19</b>
<b>Final note regarding solutions.....</b>	<b>19</b>
<b>About Watchfire .....</b>	<b>20</b>
<b>References.....</b>	<b>21</b>

Copyright © 2005 Watchfire Corporation. All Rights Reserved. Watchfire, WebCPO, WebXM, WebQA, Watchfire Enterprise Solution, WebXACT, Linkbot, Macrobot, Metabot, Bobby, Sanctum, AppScan, the Sanctum Logo, the Bobby Logo and the Flame Logo are trademarks or registered trademarks of Watchfire Corporation. GómezPro is a trademark of Gómez, Inc., used under license. All other products, company names, and logos are trademarks or registered trademarks of their respective owners.

Except as expressly agreed by Watchfire in writing, Watchfire makes no representation about the suitability and/or accuracy of the information published in this whitepaper. In no event shall Watchfire be liable for any direct, indirect, incidental, special or consequential damages, or damages for loss of profits, revenue, data or use, incurred by you or any third party, arising from your access to, or use of, the information published in this whitepaper, for a particular purpose.

[www.watchfire.com](http://www.watchfire.com)

**Figure 21 - Security Analyst Issue Report Sample Page 2**

Next, consider an issue report from an independent hacker: the report is sent via electronic mail directly to the BugTraq mailing list discussed in Chapter 3 and not to the

Apache Security Team as requested. The analyst provides a script built from scratch to demonstrate the vulnerability. The report shows a level of technical maturity but contains spelling errors and other less professional indicators.

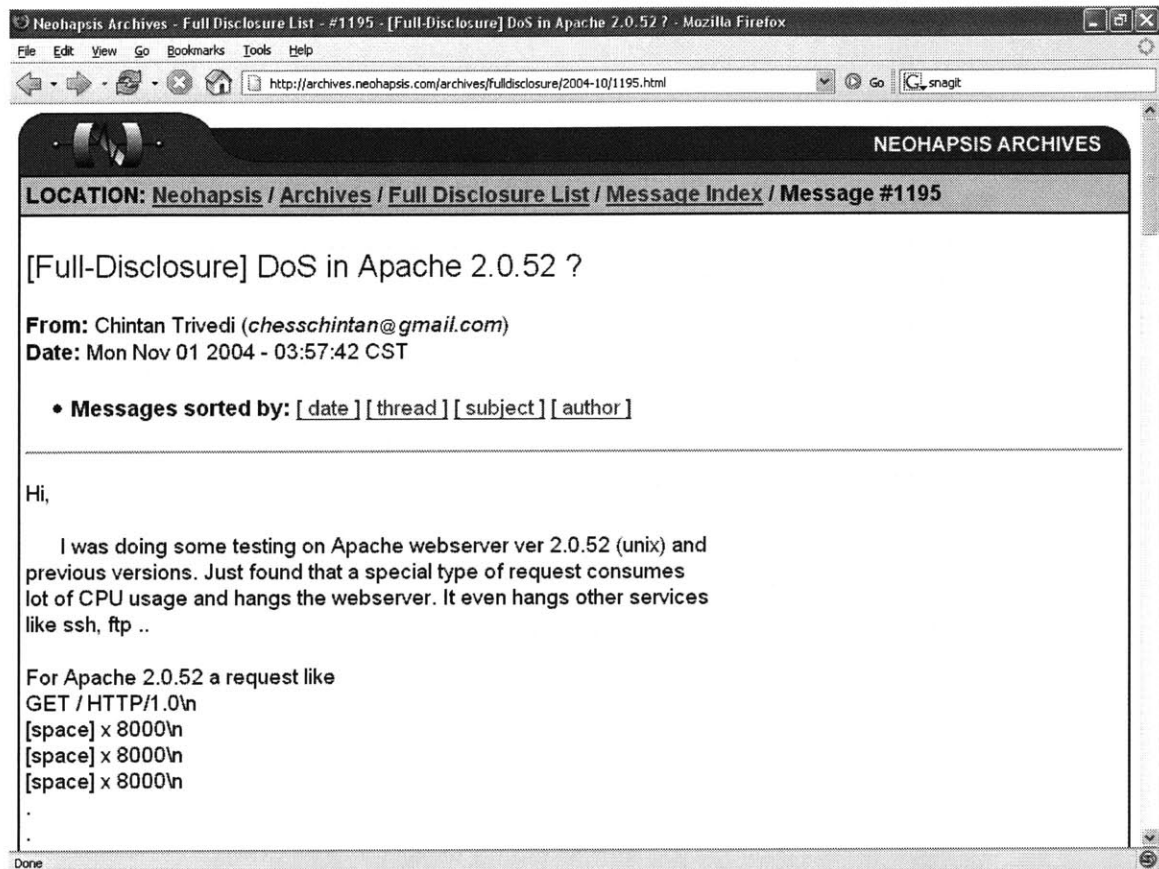


Figure 22 - Independent Hacker Bug Report (Excerpt 1)



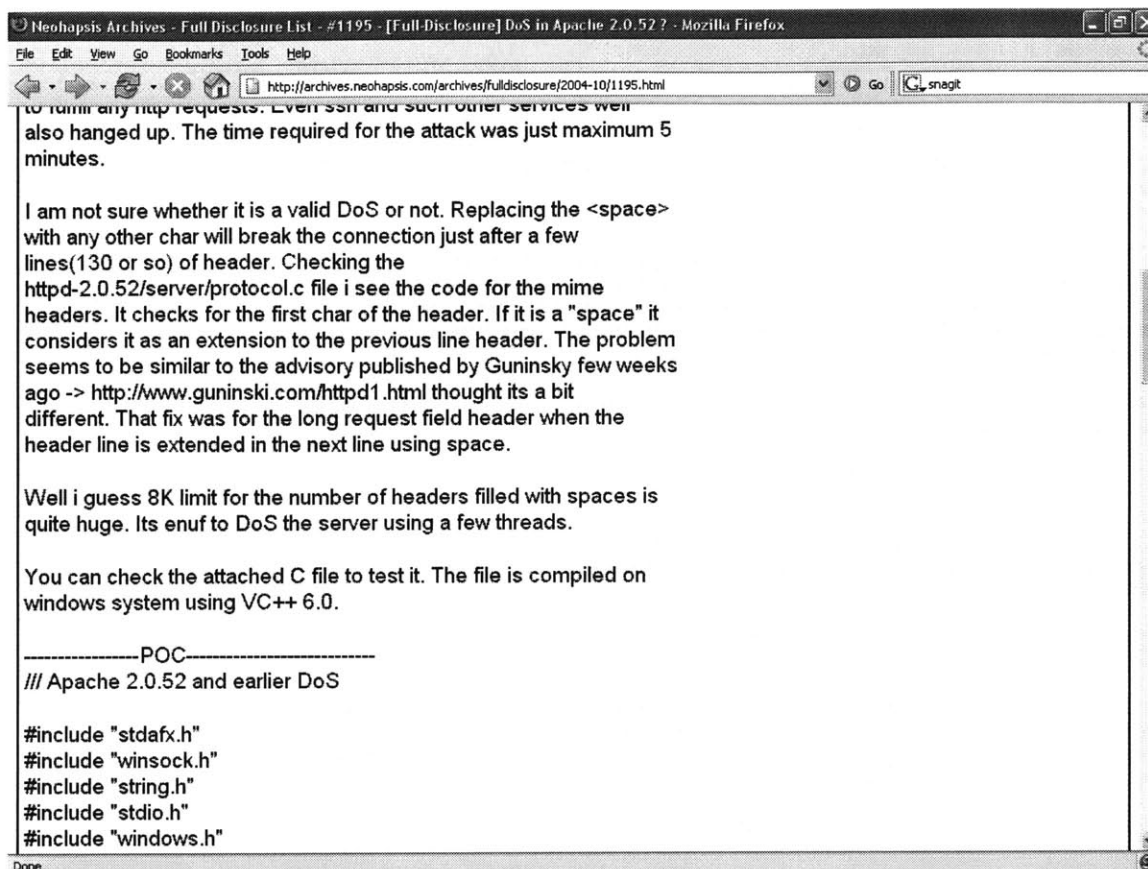
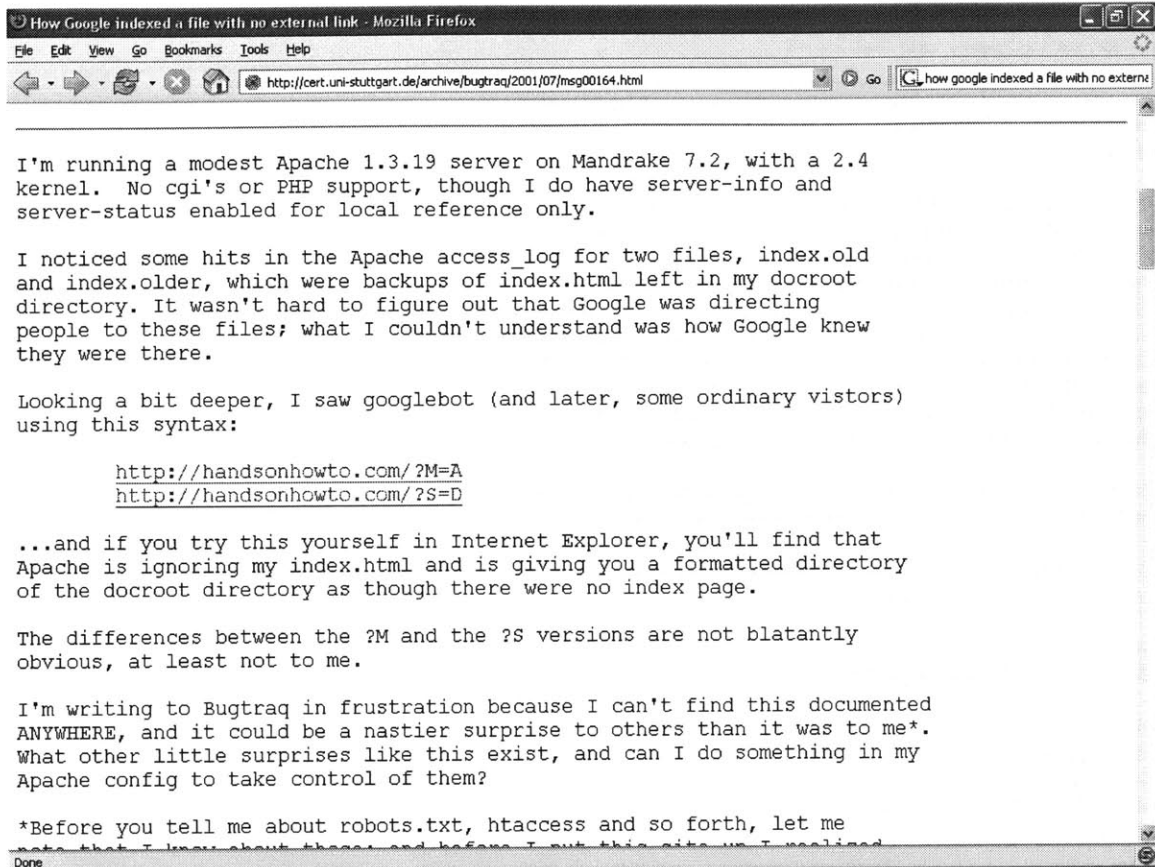


Figure 23 - Independent Hacker Bug Report (Excerpt 2)

Finally, consider a possible issue reported by a “normal user” who is actually a web site administrator. He noticed strange requests in his site log file and realized Google was providing search results for a page that had no external links and therefore should not have been visible to the outside world. That prompted him to investigate the issue and ask for advice on the mailing list. Further discussion ensued on the list, where workarounds and tests were devised, and all parties collaborated with the Apache Security Team to resolve the issue.



**Figure 24 - "Normal User" Issue Report Sample**

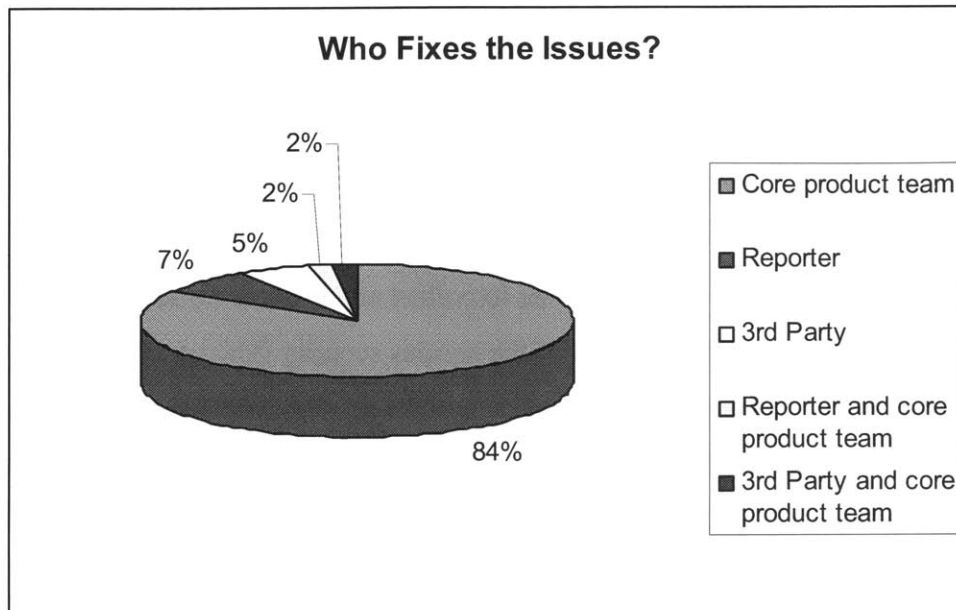
Note that the user is aware that this might be a more serious issue to others and easily admits that he is not sure if this is a security flaw. Instead, he asks for help, better documentation, and commentary. In subsequent discussions, this user helped test the fixes proposed by the development team, and he was properly credited in the resulting CVE database record and Apache release announcements.

The second explanation for the relative minority of serious security issues reported by normal users is that these users actually report many of their issues directly to the team; these issues are real but not significant enough to merit widespread distribution or a CVE number. This explanation is also partially corroborated when one examines the release notes for any particular Apache version: references are frequently made to "possible, but unlikely or insignificant" security issues reported by a particular user and addressed by the core team. Because these issues are judged (by both the reporter and the team) to not present a widespread security risk, they are not passed through the normal security

process described above. The public is made aware of them only when the release is made. History has shown this process to be true: over the past five years, there have been no such issues that were judged insignificant by the team and then became a widespread risk resulting in CVE / CERT advisories.

Nonetheless, one extension to this work would be to collect and analyze these small user reports and compare their composition to that of the serious security issues included in this research. It would also be beneficial to contact these users who reported an issue, whether it received a CVE tracking number or not, and survey them regarding the effort they expended in finding and reporting the issue.

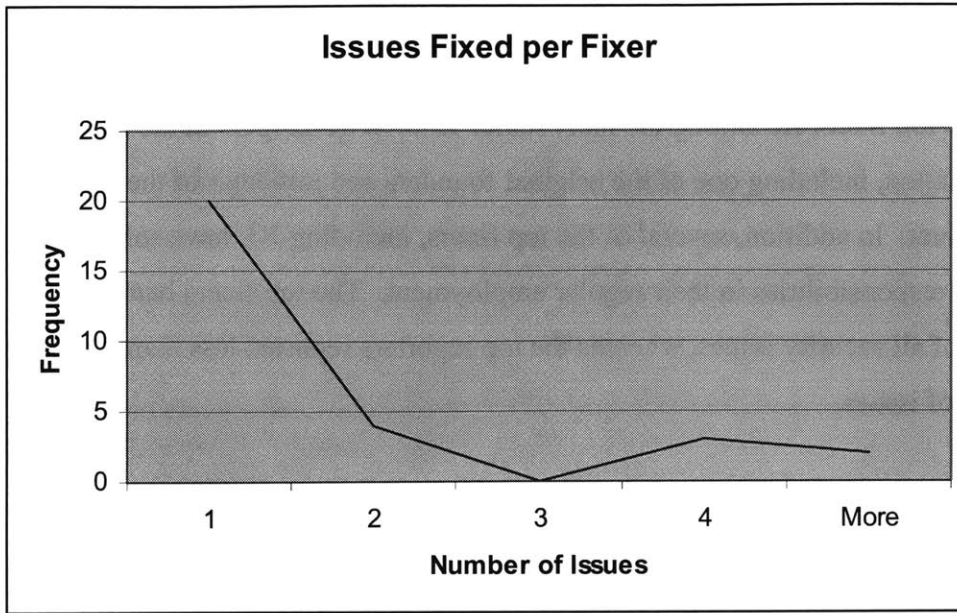
Next, consider user participation in actually fixing issues. As the following data shows, users participate in fixing issues in several ways. Sometimes the person who reported the issue submits a fix, either at the time of reporting or after further discussions. Other times, the person drops out of sight after reporting the issue and the core product team comes up with a fix by itself. Perhaps most interesting, sometimes a third party is responsible for the fix: these may be people contacted discreetly by the Apache Security Team, or they may be individuals who stumbled upon the same security issue and are reporting it after the original report.



**Figure 25 - Breakdown of Sources of Issue Fixes**

As the above diagram shows, the vast majority of issues are fixed by the product team. The probable reason for this difference is that the Apache web server product is a complex and mature software package with many components, layers, and other complicating issues. Properly identifying the relevant code in order to fix it, and then fixing the code without introducing regression defects, is a very difficult task; it is beyond the capability of most users.

If this explanation is true, one might argue that correcting security issues is beyond the capability of most Apache developers as well, and therefore we should see a small minority of developers who fix more than their share of issues. The following tables shed light on this theory.



**Figure 26 - Issues Fixed per Fixer**

The above table shows that most fixers only fix one issue: indeed, the median number of issues fixed per fixer is 1.0. However, the average is 1.72, 16% higher than the 1.48 issues reported per reporter. Furthermore, the percentage of fixers who fix more than one issue is 31.03%, slightly greater than the 29.63% of reporters who report more than one issue.

The difference really becomes evident in the top fixers versus the top reporters breakdown. An examination of people who report or fix four issues or more shows the following:

Top Reporters	Issues Reported	Top Fixers	Issues Fixed
DE	4	JO	8
GG	3	WR	5
MM	3	JT	5
AL	3	JJ	4
		AM	4
		BS	4
Total:	13		30
Percentage of Issues:	23.21%		53.57%

**Table 4: Top Fixers versus Top Reporters**

As the table shows, the top fixers addressed significantly more issues than the top reporters, and there are more of them. An examination of the source code repository shows that these top fixers are among the most senior tenured developers in the Apache Software Foundation, including one of the original founders and initiators of the Apache web server product. In addition, several of the top fixers, including JO, have software security-related responsibilities in their regular employment. The top fixers handled more than half of all security issues, whereas the top reporters reported less than a quarter of the same set of issues.

Again, the plurality of issue reporters shows that the security issue handling process is open to external input and does not feature excessive restrictions in information flow. Although many security issues are fixed by the top fixers, each fix, independent of its source, is still validated by the product team, and each product version is approved by the usual consensus-based Foundation voting procedures. This helps prevent centralized decision-making by just one or two developers, even if they are top fixers or otherwise senior members of the team. The overall process is efficient and does not suffer from threat-rigid behaviors.

## Chapter 5: Conclusions and Further Work

*“All of the most significant open source communities have some centralized ‘Cathedral’ elements – look at the way Linus controls what goes into the Linux kernel, or the way Larry Wall controls what goes into the design of Perl. But the most successful open source communities surround that cathedral with a bazaar that is significantly open.”*  
– Tim O’Reilly (referring to Raymond 1999)

This chapter presents conclusions from the research, notes the limitations of this thesis, and makes recommendations for further work. In considering the conclusions and scope of this study, it is helpful to remember the context and time frame of the research. The work was conducted over approximately seven months while the researcher was engaged in full-time study at the Massachusetts Institute of Technology. While virtually all the data gathered for this research is publicly available, it is highly fragmented and its assembly is a time-consuming and challenging task. Furthermore, it is only in the past several years that centralized security issue tracking databases have begun thoroughly cross-referencing and linking entries. Information on issues reported in the year 2000 or earlier is more difficult to gather and cross-check. In addition, while the current Security Team members have been active for the past several years, the security process before the year 2000 was very different, the team members were different, and little public data exists regarding issues from that time period.

The main goal of this research was to examine how one example of an open-source software development organization staffed by volunteers, the Apache Software Foundation, responds to one type of competitive threat: the threat of losing users and brand quality due to a security flaw in the Apache HTTP web server product. Because there are competing products and no licensing or contractual costs for users when switching web server products, this type of threat is presented as a potentially significant source of threat. The intent of the research was to then examine the response to the threat, specifically whether the behavioral responses at the personal, group, and organizational levels matched those predicted by Staw et al. (1981) in their threat-rigidity hypothesis.

This examination of the Apache Software Foundation was done in the context of previous work on user-led innovation (von Hippel 1988). Finding and fixing security issues in this complex software product is a challenging and creative activity. Professional analysts and independent hackers alike go to significant lengths to create unusual testing scenarios, write analyses of discovered security flaws, and often provide test scripts or patches to the source code that addresses the flaws. They are a significant part of the security handling process, as predicted by the literature on user innovation.



## **5.1 Threat-rigidity?**

The threat-rigidity hypothesis, described in detail in Chapter 2, predicts that when a security issue arises in a public forum, the Apache Software Foundation and its members would respond in a rigid manner. More specifically, the hypothesis suggests the Foundation would restrict information flow regarding the security issue to only a selected few individuals, and that those individuals would be among the more senior members of the organization. Further, the hypothesis predicts that there would be a constriction in decision-making, meaning that decisions regarding the security issue would be made by fewer people and that opposing opinions would be censored. The data collected during this research shows that most of these predictions do not hold in the Apache Software Foundation's organization.

On its face, the Foundation's process for dealing with reported security issues does involve less people and therefore restricts the flow of information. Usually, most Apache decisions are discussed on public mailing lists open to anyone. Accordingly, any other forum would represent a reduction in the number of people involved in the discussion. However, while the Apache Security Team represents a small minority of the Foundation's developers, they usually open the discussion to all the committers on the project, even those who have not been involved for a while or who have not been involved in the part of the product that is under discussion. In numerical terms, this expands the discussion from the six or so members of the Security Team to the more than fifty product committers (Apache Software Foundation 2005).

Moreover, because the Foundation is composed of volunteers who may work with other organizations, the Security Team and product teams are unusually diverse. For example, the Security Team includes employees of a secure web site hosting company, a Linux provider, a cryptography research laboratory, and a secure data center provider. The variance of employment and therefore usage scenarios among the product team members is even greater. This leads to a significantly more heterogeneous team working on security issue resolution than the equivalent team in most traditional organizations. Even

if we accept that information flow is restricted within this team, the diversity of use-cases included in the discussion is large, and the information that does flow represents many organizations, many usage scenarios, and broad interests. In this environment, there is no clear dominant opinion, and no executive decision-maker, so the solution that is finally adopted really suits most, if not all, of the participants.

The lack of management is another way in which the Apache Software Foundation does not meet the threat-rigidity hypothesis. While the Foundation does have a formal hierarchy that culminates in a Board of Directors, this hierarchy does not manifest itself in security issue discussions. A scan of the [security@apache.org](mailto:security@apache.org) archive reveals very few messages from directors of the Foundation, and an examination of those message shows that they are specific to the security issue under discussion. They are not an attempt to force decisions or dictate policy from further up in the organizational hierarchy.

The Foundation's guidelines for certifying a product as ready for release ensure that all the product team members have an equal vote. Any individual member can veto a product release if he or she thinks the security issue is not adequately addressed or for any other reason (Coar 2005). This individual veto power ensures an honest and complete discussion of every issue and prevents any one or few individuals from exercising undue power or making decisions without the group's consent. This includes situations where the Security Team thinks an issue has been adequately addressed, but an individual product team member disagrees: discussion and fixing must continue until this individual is convinced the issue has been properly addressed. Thus, in this regard, the Foundation's decision-making guidelines ensure that no constriction will take place even during stressful situations, such as the ones posed by security issues.

Another aspect of threat-rigidity that is frequently noted and discussed in the Literature Review above is that of the shifting allocation of resources. Frequently, organizations will scramble to meet competitive threats by re-assigning key employees and shifting other resources from their standard work towards addressing the imminent threat. This

happens to a small extent within the Foundation: much of the finding and analysis of issues is done by external parties, such as security analysts or independent hackers. The Security Team is composed of volunteers with specific interest in fixing these issues; it appears that these individuals are enthusiastic to the point that security flaws attract their interest more than whatever current work they are pursuing. In other words, they do not need to be re-assigned by a manager: they re-assign themselves to work on these security issues. Once re-assigned, they do not face the full brunt of dealing with an issue by themselves: they have the entire Security Team, product team, and issue reporters with whom to work. Those individuals who are not interested keep working in their own areas: they are not forced by management to stop their work stream and address the security issue, because there is no management. Accordingly, the “ripping” effect where talented engineers are constantly re-assigned to fight fires in the organization does not occur in the Foundation.

Several key predictions of the threat-rigidity hypothesis do not hold in the context of the Apache Software Foundation’s handling of security issues, a type of competitive threat. They do not hold because of a combination of a volunteer-based, user-driven organization that lacks formal management, and a security issue handling process designed to incorporate the lead users from beginning to end, harnessing their creative resources to address the problems they themselves report. The evidence shows that the Foundation makes good use of these volunteers, decides how to handle issues in an open and participatory manner, resolves issues in a fairly timely fashion, and communicates the findings to the public adequately.

## **5.2 *Limitations of this Research***

the Apache Software Foundation is a large and mature open-source software development organization. Its processes have been refined and tested over the past decade, resulting in a stable and self-sustaining community. While some other open-source development groups have attained this level of maturity, it is still the exception: numerous other open-source software development communities are younger, less mature, and less tested. Accordingly, the findings of this research may only apply to a certain subset of open-source communities.

The Apache web server product is among the most successful open-source software products ever developed. With a market share of 70% and over 50,000,000 customers (Netcraft 2005), it presents a unique research opportunity. On the one hand, the product is used in tremendously heterogeneous environments with diverse usage scenarios. On the other hand, its central place in the Internet's infrastructure has led to increased attention on security, security handling, and related research, both within the Foundation and outside. While some other open-source products are similarly widely used, the lessons drawn from this project may be limited to mature, well-developed, extremely-well-tested projects that attract a community of analysts and security firms.

Security issues are only one type of a competitive threat. One can easily visualize other threats in the technical realm, such as a competing product introducing a feature that Apache does not have and trumpeting this feature in press releases, or in the organizational realm, such as a large portion of Apache's volunteers leaving the organization for whatever reason. Neither of these competitive threats is considered in this research: they may cause the same type of behavior in the Foundation, or they may bring about a different response. This research is not a general study of competitive threats to open-source organizations.

This research focused on those security issues serious enough to merit widespread publication, product fixes, and database tracking numbers such as those assigned by the CVE and CERT organizations described in Chapter 3. There are many other security

issues reported to the Apache Software Foundation that do not meet these criteria: they are judged to be insignificant, they merit a change to the documentation rather than a code fix, or they are deemed so obscure that the reporter is asked to change his or her environment. These issues are harder to track and analyze, especially given the privacy of the [security@apache.org](mailto:security@apache.org) mailing list archives.

Finally, one must note that there are artificial constraints on the security issue handling process imposed on the Apache Software Foundation by external organizations, such as security analysis firms or central coordinating databases. Sometimes these organizations report an issue but ask the Foundation to hold off on addressing the issue until further information is discovered or a user's permission to disclose certain data is obtained. In these cases, the time to issue resolution and the time to product release may be artificially lengthened, and some of the statistics presented in Chapter 4 may be inaccurate. This thesis assumes that such occurrences are rare, and that when they do happen the delay is minimal in comparison with the actual time spent resolving the security issue.

### **5.3 Further Work**

The conclusions and limitations sections above identify some opportunities for further work. As noted, the Apache Software Foundation is somewhat unusual within the open-source world due to its maturity and development processes. Research in other open-source software development organizations, such as the Mozilla Foundation, the Eclipse Project, or the GNU Project would be worthwhile: these are also large organizations with high visibility and high product usage rates. In addition, research into smaller groups, such as some of the team projects on SourceForge.net, would be insightful. Such research could analyze these organizations and compare them to Apache.

Another vein of research to build on this work should involve commercial, for-profit software development organizations. A comparison of how these organizations handle security issues, and how they respond to competitive threats in general, would be an interesting and valuable contribution to the field. For example, one might compare Microsoft's handling of security issues in its Internet Information Server (IIS), a direct competitor to the Apache web server product, to the Foundation's handling analyzed in this work. On its face, it appears that the threat-rigidity hypothesis should hold stronger in a for-profit organization. The security incident data is typically difficult to obtain from such organizations, but nonetheless it presents a fascinating research opportunity.

There is much work to be done even within the Apache Software Foundation's organization. As noted above, this thesis focused on security issues serious enough to merit widespread attention. There are many more issues that do not meet these criteria; analyzing them, and the organization's handling of them, would present an interesting research topic. Along similar lines, analyzing the organizational response to other types of competitive threats, such as a competitor coming up with a much-hyped new feature or a competitor claiming to be compliant with some new specification not yet implemented by the Apache web server, would be a good follow-up research project.

Another type of possible research work in this area involves the composition, consistency, and completeness of the security issue tracking databases themselves. As

previously noted, they contain fragmented information, and one must labor to join and cross-reference the data on any given security issue. Perhaps another type of research database should be created. New ways are needed for cross-referencing this security data such that the relevant information remains intact and available for many years.

Finally, and perhaps most fundamentally, there is room for further research on what constitutes a competitive threat. Is there any type of threat that can make an organization like the Apache Software Foundation drastically alter the way it operates? How do these volunteer communities fit into the organizational behavior and strategy literature? These are interesting questions for further work and analysis.

## Bibliography

Apache Software Foundation (2005). Apache web server product team:  
<http://httpd.apache.org/contributors/>. Apache Security Team:  
<http://people.apache.org/~jim/committers.html>.

Argyris, C. and D. Schon (1978). Organizational Learning, Addison-Wesley, Reading, MA.

Audia, P.G. and H.R. Greve (2002). "Performance, Firm Size, and Factory Expansion in the Shipbuilding Industry." Management Science, accepted for publication in 2005.

Barnett, C.K. and M.G. Pratt (2000). "From threat-rigidity to flexibility: Toward a learning model of autogenic crisis in organizations." Journal of Organizational Change Management 13(1): 74-88.

Beckman, C.M., Haunschild, P.R., and D.J. Phillips (2004). "Friends or Strangers? Firm-Specific Uncertainty, Market Uncertainty, and Network Partner Selection." Organization Science 15(3): 259-275.

Chavez, R. (2003). "One More Post About ApacheCon 2003." O'Reilly Developer Weblogs, O'Reilly Media, Inc. <http://www.oreillynet.com/pub/wlg/3978>.

Christensen, C. (1997). The Innovator's Dilemma. Harvard Business School Press, Cambridge, MA.

Coar, K. (2005). "Software Development the Apache Way." Linux Magazine, May 2005: <http://www.linux-mag.com/content/view/2015/2304/>

Cox, M.J. (2004). "Apache Security Levels." Apache Week, October 2004.  
<http://www.apacheweek.com/features/security-levels>

D'Aveni, R.A. and I.C. MacMillan (1990). "Crisis and the content of managerial communications: a study of the focus of attention of top managers in surviving and failing firms." Administrative Science Quarterly, 35: 634-657.

D'Aunno, T. and R.I. Sutton (1992). "The Responses of Drug Abuse Treatment Organizations to Financial Adversity: A Partial Test of the Threat-Rigidity Thesis." Journal of Management 18(1): 117-131.

Edmondson A.C. (1999). "Psychological Safety and Learning Behavior in Work Teams." Administrative Science Quarterly 44: 350-383.



- Franke, N. and E. von Hippel (2002). "Satisfying Heterogeneous User Needs Via Innovation Toolkits: The Case of Apache Security Software." MIT Sloan School of Management Working Paper 4341-02.
- Gilbert, C.G. (2002). "Beyond Resource Allocation: Towards a Process Model of Response to Disruptive Change." Harvard Business School Working Paper 03-018.
- Gladstein, D.L. and N.P. Reilly (1985). "Group decision making under threat: The tycoon game." Academy of Management Journal, 28: 613-627.
- Hoffi-Hofstetter, H. and B. Mannheim (1999). "Managers' Coping Resources, Perceived Organizational Patterns, and Responses during Organizational Recovery from Decline." Journal of Organizational Behavior 20(5): 665-685.
- Jagielski, J. (2005). "Apache Committers" Dynamically-updated web page at <http://people.apache.org/~jim/committers.html>.
- Janis, I.L. (1982). Groupthink: Psychological studies of policy decisions and fiascoes. Houghton-Mifflin, Boston, MA.
- Kahneman, D. and A. Tversky (1979). "Prospect Theory: An Analysis of Decision Under Risk." Econometrica 47: 263-292.
- Lakhani, K.R. and R.G. Wolf (2005). "Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects." In Perspectives on Free and Open Source Software, MIT Press, Cambridge, MA, USA.
- Lakhani, K.R. (1999). "Sustaining the Virtual Commons: End User Support for Apache Web Server Software on the Usenet." Masters Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA.
- Lakhani, K.R. and E. von Hippel (2003). "How open source software works: "free" user-to-user assistance." Research Policy 32: 923-943.
- Linhart, C., Klein, A., Heled, R., and S. Orrin (2005). "HTTP Request Smuggling" Watchfire, Inc. whitepaper: <https://www.watchfire.com/securearea/whitepapers.aspx?id=12>.
- LinuxCare (2000). "Demystifying Open Source: How Open Source Software Development Works." <http://www.linuxcare.com>
- March, J.G. and Z. Shapira (1987). "Managerial perspectives on risk and risk taking." Management Science 33: 1404-1418.

Mitroff, I.I., Pearson, C., and T.C. Pauchant (1992). "Crisis management and strategic management: similarities, differences, and challenges." In Advances in Strategic Management, 8: 235-260. JAI Press, Greenwich, CT.

Netcraft (2005). "October 2005 Web Server Survey" Netcraft.com:  
[http://news.netcraft.com/archives/2005/10/04/october\\_2005\\_web\\_server\\_survey.html](http://news.netcraft.com/archives/2005/10/04/october_2005_web_server_survey.html).

Open Source Initiative (2005). <http://www.opensource.org/>

Ozcan, S. (2005). "Examining Slack-Innovation Relationship: Longitudinal Evidence from US Farm Equipment Industry (1966-2001)." Copenhagen Business School Working Paper.

Pauwels, P. and P. Matthyssens (2002). "The Dynamics of International Market Withdrawal." In State of the Art of Research in International Marketing, Edwar Elgar Publishing, Cheltenham, UK.

Price, D. R. and Ximbiot (2005). "Concurrent Versions System."  
<http://www.nongnu.org/cvs/>

Rajeswari, K.S. and R.N. Anantharaman (2003). "Development of an Instrument to Measure Stress among Software Professionals: Factory Analytic Study." ACM SIGMIS Conference 2003, Philadelphia, PA, USA.

Raymond, E.S. (1999). The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary. O'Reilly and Associates, Inc., Sebastopol, CA, USA.

Rosenblatt, Z. and B. Mannheim (1996). "Organizational Response to Decline in the Israeli Electronics Industry." Organization Studies 19(6): 953-984.

Schein, E.H. (1987). Process Consultation, Vol. II Addison-Wesley, Reading, MA.

Schein, E.H. (1993). "How can organizations learn faster? The challenge of entering the green room." Sloan Management Review Winter: 85-92.

Sigglekow, N. and D.A. Levinthal (2003). "Temporarily Divide to Conquer: Centralized, Decentralized, and Reintegrated Organizational Approaches to Exploration and Adaptation." Organization Science 14(6): 650-669.

Staw, B.M., Sandelands, L.E. and Dutton, J.E. (1981). "Threat-Rigidity Effects in Organizational Behavior: A Multilevel Analysis." Administrative Science Quarterly 26:501-524.

Taylor, M., Kent, M.L. and White, W.J. (2001). "How activity organizations are using the Internet to build relationships." Public Relations Review 27: 263-284.

Tigris (2005). "What Is Subversion?" <http://subversion.tigris.org/>

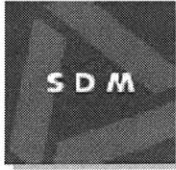
Thomas, K. (1976). "Conflict and Conflict Management." In M.D. Dunnette (Ed.) Handbook of industrial and organizational psychology: 889-936. Rand McNally, Chicago, IL.

Xie, X.F. and X.T. Wang (2003). "Risk Perception and Risky Choice: Situational, Informational, and Dispositional Effects." Asian Journal of Social Psychology 6: 117-132.

von Hippel E. (1986). "Lead Users: A Source of Novel Product Concepts." Management Science 32(7): 791-805.

von Hippel E. (1988). The Sources of Innovation. New York, Oxford University Press.

## ***Appendix A: Original Thesis Proposal***



Below is the original proposal for this thesis. It is included here for historical reference and completeness only.

### **SDM Thesis Proposal Form**

Student Name: Yoav Shapira

MIT I.D. Number: 966257984

Student's E-Mail Address: yoavsh@sloan.mit.edu

Today's Date: 3 March 2005

Thesis Completion Date: January 2006

Thesis Supervisor: Professor Eric von Hippel

e-mail Address: evhippel@mit.edu

Thesis Reader: Dr. Karim Lakhani

e-mail Address: lakhani@mit.edu

Thesis Title: Threat-Fluidity: How Open-Source Communities Defy Classical Threat-Rigidity Patterns

#### Motivation:

There is a significant body of literature suggesting that traditional organizations respond rigidly to threats: there are numerous social and economic factors which make the organization retrench its stance and focus on its core competencies and established procedures. Often times, this is not the optimal behavior, and yet even well-managed organizations succumb to it. The open-source community model, however, is a relatively new organizational structure that I believe exhibits the opposite behavior: threat-fluidity. Unlike traditional organizational structures, the open-source community reacts to threats with an increased level of productivity and efficiency.

The area of open-source user innovation communities holds particular fascination to me as both a practitioner and a researcher. I am intrigued by its new organizational forms, and how they can be applied to traditional organizations in order to improve productivity and quality. I also hope that this thesis will form an important part of Professor von Hippel and Dr. Lakhani's ongoing research efforts.

#### Thesis Statement & Primary Research Objectives:

I contend that open-source software development communities differ from traditional organizations in how they respond to threats: instead of retrenching and slowing down, these communities exhibit increased efficiency and effectiveness in responding to the threat.

The primary research objective of this thesis is to prove the above point. The thesis will include a review of relevant literature on threat-rigidity and open-source community

models. I will collect and analyze data on how both classical and open-source software development organization respond to two types of threats: critical flaws being found in their products, and new technical specifications to which the products must conform. Finally, I will attempt to point out those features of open-source organizations that can be transferred into a more traditional environment, if any, the technical means for doing so, and under what circumstances such reorganizations are beneficial.

Engineering and Management Content:

This thesis will combine engineering and management content by examining both the social and technical aspects of open-source communities and traditional software development organizations as they apply to thread-rigidity.

Much of the existing literature has a sociological focus on organizational behavior. But with the critical and growing importance of software to practically all organizations today, a work of research in this area which specifically focuses on the practices of software development should prove valuable to both engineers and managers.

Research Methods & Approaches:

I plan to employ a combination of classical and open-source-specific research methods. The literature review and statistical analyses will follow well-established scientific standards. To analyze open-source communities, I will take advantage of the public history of changes to source code as well as the publicly available mailing list archives for the products covered in the thesis.

In addition, I hope to take advantage of my network of open-source developers and conduct a number of interviews or surveys as part of the thesis research. I would like to correlate their views with those of other innovating lead users found in Professor von Hippel's work.

Timeline:

1. Thesis initiation, organization, and planning: Spring 2005.
2. Thesis research, initial writing: Summer 2005.
3. Research concluded and first draft complete: October 2005.
4. Final draft completed, thesis submitted: December 2005.

Signatures:

SDM Fellow: \_\_\_\_\_ Date: \_\_\_\_\_

Thesis Supervisor: \_\_\_\_\_ Date: \_\_\_\_\_

Thesis Reader: \_\_\_\_\_ Date: \_\_\_\_\_

Company Sponsor: \_\_\_\_\_ Date: \_\_\_\_\_

## Appendix B: Raw Security Issue Record

The following is the raw record of security issues, in XML format, as records by the Apache HTTP web server product team. The record is available in the source code repository at

<http://svn.apache.org/viewcvcs.cgi/httpd/site/trunk/xdocs/security/vulnerabilities-httpd.xml?view=markup>.

```
= <security updated="20051101">
= <issue fixed="2.0.55" released="20051014">
  <cve name="CVE-2005-2970" />
  <severity level="3">moderate</severity>
  <title>Worker MPM memory leak</title>
= <description>
  <p>A memory leak in the worker MPM would allow remote attackers to cause
    a denial of service (memory consumption) via aborted connections, which
    prevents the memory for the transaction pool from being reused for other
    connections.</p>
  </description>
  <affects prod="httpd" version="2.0.54" />
  <affects prod="httpd" version="2.0.53" />
  <affects prod="httpd" version="2.0.52" />
  <affects prod="httpd" version="2.0.51" />
  <affects prod="httpd" version="2.0.50" />
  <affects prod="httpd" version="2.0.49" />
  <affects prod="httpd" version="2.0.48" />
  <affects prod="httpd" version="2.0.47" />
  <affects prod="httpd" version="2.0.46" />
  <affects prod="httpd" version="2.0.45" />
  <affects prod="httpd" version="2.0.44" />
  <affects prod="httpd" version="2.0.43" />
  <affects prod="httpd" version="2.0.42" />
  <affects prod="httpd" version="2.0.40" />
  <affects prod="httpd" version="2.0.39" />
  <affects prod="httpd" version="2.0.37" />
  <affects prod="httpd" version="2.0.36" />
  - <!--
    bad code was added 20020428 therefore after 2.0.35
  -->
  </issue>
= <issue fixed="2.0.55" public="20050707" reported="20050707"
  released="20051014">
  <cve name="CVE-2005-2728" />
  <severity level="3">moderate</severity>
  <title>Byterange filter DoS</title>
= <description>
  <p>A flaw in the byterange filter would cause some responses to be buffered
    into memory. If a server has a dynamic resource such as a CGI script or PHP
```

**script which generates a large amount of data, an attacker could send carefully crafted requests in order to consume resources, potentially leading to a Denial of Service.**

```
</description>
<affects prod="httpd" version="2.0.54" />
<affects prod="httpd" version="2.0.53" />
<affects prod="httpd" version="2.0.52" />
<affects prod="httpd" version="2.0.51" />
<affects prod="httpd" version="2.0.50" />
<affects prod="httpd" version="2.0.49" />
<affects prod="httpd" version="2.0.48" />
<affects prod="httpd" version="2.0.47" />
<affects prod="httpd" version="2.0.46" />
<affects prod="httpd" version="2.0.45" />
<affects prod="httpd" version="2.0.44" />
<affects prod="httpd" version="2.0.43" />
<affects prod="httpd" version="2.0.42" />
<affects prod="httpd" version="2.0.40" />
<affects prod="httpd" version="2.0.39" />
<affects prod="httpd" version="2.0.37" />
<affects prod="httpd" version="2.0.36" />
<affects prod="httpd" version="2.0.35" />
</issue>
- <issue fixed="2.0.55" public="20050830" reported="20050830"
  released="20051014">
  <cve name="CVE-2005-2700" />
  <severity level="2">important</severity>
  <title>SSLVerifyClient bypass</title>
- <description>
  <p>A flaw in the mod_ssl handling of the "SSLVerifyClient" directive. This flaw would occur if a virtual host has been configured using "SSLVerifyClient optional" and further a directive "SSLVerifyClient required" is set for a specific location. For servers configured in this fashion, an attacker may be able to access resources that should otherwise be protected, by not supplying a client certificate when connecting.
```

```

<affects prod="httpd" version="2.0.39" />
<affects prod="httpd" version="2.0.37" />
<affects prod="httpd" version="2.0.36" />
<affects prod="httpd" version="2.0.35" />
</issue>
- <issue fixed="2.0.55" public="20050801" released="20051014">
  <cve name="CVE-2005-2491" />
  <severity level="4">low</severity>
  <title>PCRE overflow</title>
  - <description>
    <p>An integer overflow flaw was found in PCRE, a Perl-compatible regular
      expression library included within httpd. A local user who has the ability to
      create .htaccess files could create a maliciously crafted regular expression
      in such as way that they could gain the privileges of a httpd child.</p>
    </description>
    <affects prod="httpd" version="2.0.54" />
    <affects prod="httpd" version="2.0.53" />
    <affects prod="httpd" version="2.0.52" />
    <affects prod="httpd" version="2.0.51" />
    <affects prod="httpd" version="2.0.50" />
    <affects prod="httpd" version="2.0.49" />
    <affects prod="httpd" version="2.0.48" />
    <affects prod="httpd" version="2.0.47" />
    <affects prod="httpd" version="2.0.46" />
    <affects prod="httpd" version="2.0.45" />
    <affects prod="httpd" version="2.0.44" />
    <affects prod="httpd" version="2.0.43" />
    <affects prod="httpd" version="2.0.42" />
    <affects prod="httpd" version="2.0.40" />
    <affects prod="httpd" version="2.0.39" />
    <affects prod="httpd" version="2.0.37" />
    <affects prod="httpd" version="2.0.36" />
    <affects prod="httpd" version="2.0.35" />
    </issue>
  - <issue fixed="2.0.55" public="20050611" released="20051014">
    <cve name="CVE-2005-2088" />
    <severity level="3">moderate</severity>
    <title>HTTP Request Spoofing</title>
    - <description>
      <p>A flaw occurred when using the Apache server as a HTTP proxy. A remote
        attacker could send a HTTP request with both a "Transfer-Encoding:
        chunked" header and a Content-Length header, causing Apache to
        incorrectly handle and forward the body of the request in a way that causes
        the receiving server to process it as a separate HTTP request. This could
        allow the bypass of web application firewall protection or lead to cross-site
        scripting (XSS) attacks.</p>
      </description>
      <affects prod="httpd" version="2.0.54" />
      <affects prod="httpd" version="2.0.53" />
      <affects prod="httpd" version="2.0.52" />

```



```

<affects prod="httpd" version="2.0.51" />
<affects prod="httpd" version="2.0.50" />
<affects prod="httpd" version="2.0.49" />
<affects prod="httpd" version="2.0.48" />
<affects prod="httpd" version="2.0.47" />
<affects prod="httpd" version="2.0.46" />
<affects prod="httpd" version="2.0.45" />
<affects prod="httpd" version="2.0.44" />
<affects prod="httpd" version="2.0.43" />
<affects prod="httpd" version="2.0.42" />
<affects prod="httpd" version="2.0.40" />
<affects prod="httpd" version="2.0.39" />
<affects prod="httpd" version="2.0.37" />
<affects prod="httpd" version="2.0.36" />
<affects prod="httpd" version="2.0.35" />
</issue>
- <issue fixed="2.0.55" public="20050608" released="20051014">
  <cve name="CVE-2005-1268" />
  <severity level="4">low</severity>
  <title>Malicious CRL off-by-one</title>
- <description>
  <p>An off-by-one stack overflow was discovered in the mod_ssl CRL
    verification callback. In order to exploit this issue the Apache server would
    need to be configured to use a malicious certificate revocation list
    (CRL)</p>
  </description>
  <affects prod="httpd" version="2.0.54" />
  <affects prod="httpd" version="2.0.53" />
  <affects prod="httpd" version="2.0.52" />
  <affects prod="httpd" version="2.0.51" />
  <affects prod="httpd" version="2.0.50" />
  <affects prod="httpd" version="2.0.49" />
  <affects prod="httpd" version="2.0.48" />
  <affects prod="httpd" version="2.0.47" />
  <affects prod="httpd" version="2.0.46" />
  <affects prod="httpd" version="2.0.45" />
  <affects prod="httpd" version="2.0.44" />
  <affects prod="httpd" version="2.0.43" />
  <affects prod="httpd" version="2.0.42" />
  <affects prod="httpd" version="2.0.40" />
  <affects prod="httpd" version="2.0.39" />
  <affects prod="httpd" version="2.0.37" />
  <affects prod="httpd" version="2.0.36" />
  <affects prod="httpd" version="2.0.35" />
  </issue>
- <issue fixed="2.0.53" public="20041101" released="20050208"
  reported="20041028">
  <cve name="CVE-2004-0942" />
  <severity level="2">important</severity>

```

```

<title>Memory consumption DoS</title>
= <description>
<p>An issue was discovered where the field length limit was not enforced for
certain malicious requests. This could allow a remote attacker who is able to
send large amounts of data to a server the ability to cause Apache children
to consume proportional amounts of memory, leading to a denial of
service.</p>
</description>
<affects prod="httpd" version="2.0.52" />
<affects prod="httpd" version="2.0.51" />
<affects prod="httpd" version="2.0.50" />
<affects prod="httpd" version="2.0.49" />
<affects prod="httpd" version="2.0.48" />
<affects prod="httpd" version="2.0.47" />
<affects prod="httpd" version="2.0.46" />
<affects prod="httpd" version="2.0.45" />
<affects prod="httpd" version="2.0.44" />
<affects prod="httpd" version="2.0.43" />
<affects prod="httpd" version="2.0.42" />
<affects prod="httpd" version="2.0.40" />
<affects prod="httpd" version="2.0.39" />
<affects prod="httpd" version="2.0.37" />
<affects prod="httpd" version="2.0.36" />
<affects prod="httpd" version="2.0.35" />
</issue>
= <issue fixed="1.3.33" public="20041021" released="20041028"
reported="20041021">
<cve name="CVE-2004-0940" />
<title>mod_include overflow</title>
<severity level="3">moderate</severity>
= <description>
<p>A buffer overflow in mod_include could allow a local user who is
authorised to create server side include (SSI) files to gain the privileges of a
httpd child.</p>
</description>
<affects prod="httpd" version="1.3.32" />
<affects prod="httpd" version="1.3.31" />
<affects prod="httpd" version="1.3.29" />
<affects prod="httpd" version="1.3.28" />
<affects prod="httpd" version="1.3.27" />
<affects prod="httpd" version="1.3.26" />
<affects prod="httpd" version="1.3.24" />
<affects prod="httpd" version="1.3.22" />
<affects prod="httpd" version="1.3.20" />
<affects prod="httpd" version="1.3.19" />
<affects prod="httpd" version="1.3.17" />
<affects prod="httpd" version="1.3.14" />
<affects prod="httpd" version="1.3.12" />
<affects prod="httpd" version="1.3.11" />
<affects prod="httpd" version="1.3.9" />

```

```

<affects prod="httpd" version="1.3.6" />
<affects prod="httpd" version="1.3.4" />
<affects prod="httpd" version="1.3.3" />
<affects prod="httpd" version="1.3.2" />
<affects prod="httpd" version="1.3.1" />
<affects prod="httpd" version="1.3.0" />
</issue>
- <issue fixed="2.0.53" public="20041001" reported="20041001"
  released="20050208">
  <cve name="CVE-2004-0885" />
  <severity level="3">moderate</severity>
  <title>SSLCipherSuite bypass</title>
- <description>
  <p>An issue has been discovered in the mod_ssl module when configured to
    use the "SSLCipherSuite" directive in directory or location context. If a
    particular location context has been configured to require a specific set of
    cipher suites, then a client will be able to access that location using any
    cipher suite allowed by the virtual host configuration.</p>
  </description>
  <affects prod="httpd" version="2.0.52" />
  <affects prod="httpd" version="2.0.51" />
  <affects prod="httpd" version="2.0.50" />
  <affects prod="httpd" version="2.0.49" />
  <affects prod="httpd" version="2.0.48" />
  <affects prod="httpd" version="2.0.47" />
  <affects prod="httpd" version="2.0.46" />
  <affects prod="httpd" version="2.0.45" />
  <affects prod="httpd" version="2.0.44" />
  <affects prod="httpd" version="2.0.43" />
  <affects prod="httpd" version="2.0.42" />
  <affects prod="httpd" version="2.0.40" />
  <affects prod="httpd" version="2.0.39" />
  <affects prod="httpd" version="2.0.37" />
  <affects prod="httpd" version="2.0.36" />
  <affects prod="httpd" version="2.0.35" />
  </issue>
- <issue fixed="2.0.53" public="20040320" reported="20040302"
  released="20050208">
  <cve name="CVE-2004-1834" />
  <severity level="4">low</severity>
  <title>mod_disk_cache stores sensitive headers</title>
- <description>
  <p>The experimental mod_disk_cache module stored client authentication
    credentials for cached objects such as proxy authentication credentials and
    Basic Authentication passwords on disk.</p>
  </description>
  <affects prod="httpd" version="2.0.52" />
  <affects prod="httpd" version="2.0.51" />
  <affects prod="httpd" version="2.0.50" />
  <affects prod="httpd" version="2.0.49" />

```

```

<affects prod="httpd" version="2.0.48" />
<affects prod="httpd" version="2.0.47" />
<affects prod="httpd" version="2.0.46" />
<affects prod="httpd" version="2.0.45" />
<affects prod="httpd" version="2.0.44" />
<affects prod="httpd" version="2.0.43" />
<affects prod="httpd" version="2.0.42" />
<affects prod="httpd" version="2.0.40" />
<affects prod="httpd" version="2.0.39" />
<affects prod="httpd" version="2.0.37" />
<affects prod="httpd" version="2.0.36" />
<affects prod="httpd" version="2.0.35" />
</issue>
= <issue fixed="2.0.52" released="20040928" public="20040918"
  reported="20040918">
  <cve name="CVE-2004-0811" />
  <title>Basic authentication bypass</title>
  <severity level="2">important</severity>
= <description>
  <p>A flaw in Apache 2.0.51 (only) broke the merging of the Satisfy directive
    which could result in access being granted to resources despite any
    configured authentication</p>
  </description>
  <affects prod="httpd" version="2.0.51" />
  </issue>
= <issue fixed="2.0.51" public="20040915" released="20040915"
  reported="20040825">
  <cve name="CVE-2004-0786" />
  <title>IPv6 URI parsing heap overflow</title>
  <severity level="1">critical</severity>
= <description>
  <p>Testing using the Codenomicon HTTP Test Tool performed by the Apache
    Software Foundation security group and Red Hat uncovered an input
    validation issue in the IPv6 URI parsing routines in the apr-util library. If a
    remote attacker sent a request including a carefully crafted URI, an httpd
    child process could be made to crash. On some BSD systems it is believed
    this flaw may be able to lead to remote code execution.</p>
  </description>
  <affects prod="httpd" version="2.0.50" />
  <affects prod="httpd" version="2.0.49" />
  <affects prod="httpd" version="2.0.48" />
  <affects prod="httpd" version="2.0.47" />
  <affects prod="httpd" version="2.0.46" />
  <affects prod="httpd" version="2.0.45" />
  <affects prod="httpd" version="2.0.44" />
  <affects prod="httpd" version="2.0.43" />
  <affects prod="httpd" version="2.0.42" />
  <affects prod="httpd" version="2.0.40" />
  <affects prod="httpd" version="2.0.39" />
  <affects prod="httpd" version="2.0.37" />

```

```

<affects prod="httpd" version="2.0.36" />
<affects prod="httpd" version="2.0.35" />
</issue>
- <issue fixed="2.0.51" public="20040915" released="20040915"
  reported="20040805">
  <cve name="CVE-2004-0747" />
  <severity level="4">low</severity>
  <title>Environment variable expansion flaw</title>
- <description>
  <p>The Swedish IT Incident Centre (SITIC) reported a buffer overflow in the
    expansion of environment variables during configuration file parsing. This
    issue could allow a local user to gain the privileges of a httpd child if a
    server can be forced to parse a carefully crafted .htaccess file written by a
    local user.</p>
  </description>
  <affects prod="httpd" version="2.0.50" />
  <affects prod="httpd" version="2.0.49" />
  <affects prod="httpd" version="2.0.48" />
  <affects prod="httpd" version="2.0.47" />
  <affects prod="httpd" version="2.0.46" />
  <affects prod="httpd" version="2.0.45" />
  <affects prod="httpd" version="2.0.44" />
  <affects prod="httpd" version="2.0.43" />
  <affects prod="httpd" version="2.0.42" />
  <affects prod="httpd" version="2.0.40" />
  <affects prod="httpd" version="2.0.39" />
  <affects prod="httpd" version="2.0.37" />
  <affects prod="httpd" version="2.0.36" />
  <affects prod="httpd" version="2.0.35" />
  </issue>
- <issue fixed="2.0.51" released="20040915" public="20040707"
  reported="20040707">
  <cve name="CVE-2004-0751" />
  <severity level="4">low</severity>
  <title>Malicious SSL proxy can cause crash</title>
- <description>
  <p>An issue was discovered in the mod_ssl module in Apache 2.0.44-2.0.50
    which could be triggered if the server is configured to allow proxying to a
    remote SSL server. A malicious remote SSL server could force an httpd child
    process to crash by sending a carefully crafted response header. This issue
    is not believed to allow execution of arbitrary code and will only result in a
    denial of service where a threaded process model is in use.</p>
  </description>
  <affects prod="httpd" version="2.0.50" />
  <affects prod="httpd" version="2.0.49" />
  <affects prod="httpd" version="2.0.48" />
  <affects prod="httpd" version="2.0.47" />
  <affects prod="httpd" version="2.0.46" />
  <affects prod="httpd" version="2.0.45" />
  <affects prod="httpd" version="2.0.44" />

```

```

    </issue>
- <issue fixed="2.0.51" released="20040915" public="20040707"
  reported="20040707">
  <cve name="CVE-2004-0748" />
  <severity level="2">important</severity>
  <title>SSL connection infinite loop</title>
- <description>
  <p>An issue was discovered in the mod_ssl module in Apache 2.0. A remote
    attacker who forces an SSL connection to be aborted in a particular state
    may cause an Apache child process to enter an infinite loop, consuming CPU
    resources.</p>
  </description>
  <affects prod="httpd" version="2.0.50" />
  <maybeaffects prod="httpd" version="2.0.49" />
  <maybeaffects prod="httpd" version="2.0.48" />
  <maybeaffects prod="httpd" version="2.0.47" />
  <maybeaffects prod="httpd" version="2.0.46" />
  <maybeaffects prod="httpd" version="2.0.45" />
  <maybeaffects prod="httpd" version="2.0.44" />
  <maybeaffects prod="httpd" version="2.0.43" />
  <maybeaffects prod="httpd" version="2.0.42" />
  <maybeaffects prod="httpd" version="2.0.40" />
  <maybeaffects prod="httpd" version="2.0.39" />
  <maybeaffects prod="httpd" version="2.0.37" />
  <maybeaffects prod="httpd" version="2.0.36" />
  <maybeaffects prod="httpd" version="2.0.35" />
  </issue>
- <issue fixed="2.0.51" public="20040912" reported="20040912"
  released="20040915">
  <cve name="CVE-2004-0809" />
  <title>WebDAV remote crash</title>
  <severity level="4">low</severity>
- <description>
  <p>An issue was discovered in the mod_dav module which could be triggered
    for a location where WebDAV authoring access has been configured. A
    malicious remote client which is authorized to use the LOCK method could
    force an httpd child process to crash by sending a particular sequence of
    LOCK requests. This issue does not allow execution of arbitrary code. and
    will only result in a denial of service where a threaded process model is in
    use.</p>
  </description>
  <affects prod="httpd" version="2.0.50" />
  <affects prod="httpd" version="2.0.49" />
  <affects prod="httpd" version="2.0.48" />
  <affects prod="httpd" version="2.0.47" />
  <affects prod="httpd" version="2.0.46" />
  <affects prod="httpd" version="2.0.45" />
  <affects prod="httpd" version="2.0.44" />
  <affects prod="httpd" version="2.0.43" />
  <affects prod="httpd" version="2.0.42" />

```

```

<affects prod="httpd" version="2.0.40" />
<affects prod="httpd" version="2.0.39" />
<affects prod="httpd" version="2.0.37" />
<affects prod="httpd" version="2.0.36" />
<affects prod="httpd" version="2.0.35" />
</issue>
- <issue fixed="2.0.50" released="20040701" reported="20040613"
  public="20040701">
  <cve name="CVE-2004-0493" />
  <title>Header parsing memory leak</title>
  <severity level="2">important</severity>
- <description>
  <p>A memory leak in parsing of HTTP headers which can be triggered
    remotely may allow a denial of service attack due to excessive memory
    consumption.</p>
  </description>
  <affects prod="httpd" version="2.0.49" />
  <maybeaffects prod="httpd" version="2.0.48" />
  <maybeaffects prod="httpd" version="2.0.47" />
  <maybeaffects prod="httpd" version="2.0.46" />
  <maybeaffects prod="httpd" version="2.0.45" />
  <maybeaffects prod="httpd" version="2.0.44" />
  <maybeaffects prod="httpd" version="2.0.43" />
  <maybeaffects prod="httpd" version="2.0.42" />
  <maybeaffects prod="httpd" version="2.0.40" />
  <maybeaffects prod="httpd" version="2.0.39" />
  <maybeaffects prod="httpd" version="2.0.37" />
  <maybeaffects prod="httpd" version="2.0.36" />
  <maybeaffects prod="httpd" version="2.0.35" />
  </issue>
- <issue fixed="2.0.50" released="20040701" public="20040517">
  <cve name="CVE-2004-0488" />
  <severity level="4">low</severity>
  <title>FakeBasicAuth overflow</title>
- <description>
  <p>A buffer overflow in the mod_ssl FakeBasicAuth code could be exploited by
    an attacker using a (trusted) client certificate with a subject DN field which
    exceeds 6K in length.</p>
  </description>
  <affects prod="httpd" version="2.0.49" />
  <affects prod="httpd" version="2.0.48" />
  <affects prod="httpd" version="2.0.47" />
  <affects prod="httpd" version="2.0.46" />
  <affects prod="httpd" version="2.0.45" />
  <affects prod="httpd" version="2.0.44" />
  <affects prod="httpd" version="2.0.43" />
  <affects prod="httpd" version="2.0.42" />
  <affects prod="httpd" version="2.0.40" />
  <affects prod="httpd" version="2.0.39" />

```

```

<affects prod="httpd" version="2.0.37" />
<affects prod="httpd" version="2.0.36" />
<affects prod="httpd" version="2.0.35" />
</issue>
- <issue fixed="1.3.32" public="20030610" released="20041020"
  reported="20030608">
  <cve name="CVE-2004-0492" />
  <severity level="3">moderate</severity>
  <title>mod_proxy buffer overflow</title>
- <description>
  <p>A buffer overflow was found in the Apache proxy module, mod_proxy,
    which can be triggered by receiving an invalid Content-Length header. In
    order to exploit this issue an attacker would need to get an Apache
    installation that was configured as a proxy to connect to a malicious site.
    This would cause the Apache child processing the request to crash, although
    this does not represent a significant Denial of Service attack as requests will
    continue to be handled by other Apache child processes. This issue may lead
    to remote arbitrary code execution on some BSD platforms.</p>
  </description>
  <affects prod="httpd" version="1.3.31" />
  <affects prod="httpd" version="1.3.29" />
  <affects prod="httpd" version="1.3.28" />
  <affects prod="httpd" version="1.3.27" />
  <affects prod="httpd" version="1.3.26" />
  </issue>
- <issue fixed="1.3.31" public="20030224" released="20040512"
  reported="20030224">
  <cve name="CVE-2003-0020" />
  <title>Error log escape filtering</title>
  <severity level="4">low</severity>
- <description>
  <p>Apache does not filter terminal escape sequences from error logs, which
    could make it easier for attackers to insert those sequences into terminal
    emulators containing vulnerabilities related to escape sequences.</p>
  </description>
  <affects prod="httpd" version="1.3.29" />
  <affects prod="httpd" version="1.3.28" />
  <affects prod="httpd" version="1.3.27" />
  <affects prod="httpd" version="1.3.26" />
  <affects prod="httpd" version="1.3.24" />
  <affects prod="httpd" version="1.3.22" />
  <affects prod="httpd" version="1.3.20" />
  <affects prod="httpd" version="1.3.19" />
  <affects prod="httpd" version="1.3.17" />
  <affects prod="httpd" version="1.3.14" />
  <affects prod="httpd" version="1.3.12" />
  <affects prod="httpd" version="1.3.11" />
  <affects prod="httpd" version="1.3.9" />
  <affects prod="httpd" version="1.3.6" />
  <affects prod="httpd" version="1.3.4" />

```



```

<affects prod="httpd" version="1.3.3" />
<affects prod="httpd" version="1.3.2" />
<affects prod="httpd" version="1.3.1" />
<affects prod="httpd" version="1.3.0" />
</issue>
- <issue fixed="1.3.31" public="20031218" released="20040512"
  reported="20031218">
  <cve name="CVE-2003-0987" />
  <severity level="4">low</severity>
  <title>mod_digest nonce checking</title>
- <description>
  <p>mod_digest does not properly verify the nonce of a client response by
    using a AuthNonce secret. This could allow a malicious user who is able to
    sniff network traffic to conduct a replay attack against a website using
    Digest protection. Note that mod_digest implements an older version of the
    MD5 Digest Authentication specification which is known not to work with
    modern browsers. This issue does not affect mod_auth_digest.</p>
  </description>
  <affects prod="httpd" version="1.3.29" />
  <affects prod="httpd" version="1.3.28" />
  <affects prod="httpd" version="1.3.27" />
  <affects prod="httpd" version="1.3.26" />
  <affects prod="httpd" version="1.3.24" />
  <affects prod="httpd" version="1.3.22" />
  <affects prod="httpd" version="1.3.20" />
  <affects prod="httpd" version="1.3.19" />
  <affects prod="httpd" version="1.3.17" />
  <affects prod="httpd" version="1.3.14" />
  <affects prod="httpd" version="1.3.12" />
  <affects prod="httpd" version="1.3.11" />
  <affects prod="httpd" version="1.3.9" />
  <affects prod="httpd" version="1.3.6" />
  <affects prod="httpd" version="1.3.4" />
  <affects prod="httpd" version="1.3.3" />
  <affects prod="httpd" version="1.3.2" />
  <affects prod="httpd" version="1.3.1" />
  <affects prod="httpd" version="1.3.0" />
  </issue>
- <issue fixed="1.3.31" public="20040318" released="20040512"
  reported="20040225">
  <cve name="CVE-2004-0174" />
  <severity level="2">important</severity>
  <title>listening socket starvation</title>
- <description>
  <p>A starvation issue on listening sockets occurs when a short-lived
    connection on a rarely-accessed listening socket will cause a child to hold
    the accept mutex and block out new connections until another connection
    arrives on that rarely-accessed listening socket. This issue is known to
    affect some versions of AIX, Solaris, and Tru64; it is known to not affect
    FreeBSD or Linux.</p>

```

```

    </description>
    <affects prod="httpd" version="1.3.29" />
    <maybeaffects prod="httpd" version="1.3.28" />
    <maybeaffects prod="httpd" version="1.3.27" />
    <maybeaffects prod="httpd" version="1.3.26" />
    <maybeaffects prod="httpd" version="1.3.24" />
    <maybeaffects prod="httpd" version="1.3.22" />
    <maybeaffects prod="httpd" version="1.3.20" />
    <maybeaffects prod="httpd" version="1.3.19" />
    <maybeaffects prod="httpd" version="1.3.17" />
    <maybeaffects prod="httpd" version="1.3.14" />
    <maybeaffects prod="httpd" version="1.3.12" />
    <maybeaffects prod="httpd" version="1.3.11" />
    <maybeaffects prod="httpd" version="1.3.9" />
    <maybeaffects prod="httpd" version="1.3.6" />
    <maybeaffects prod="httpd" version="1.3.4" />
    <maybeaffects prod="httpd" version="1.3.3" />
    <maybeaffects prod="httpd" version="1.3.2" />
    <maybeaffects prod="httpd" version="1.3.1" />
    <maybeaffects prod="httpd" version="1.3.0" />
    </issue>
- <issue fixed="2.0.49" public="20040318" released="20040319"
    reported="20040225">
    <cve name="CVE-2004-0174" />
    <severity level="2">important</severity>
    <title>listening socket starvation</title>
- <description>
    <p>A starvation issue on listening sockets occurs when a short-lived
    connection on a rarely-accessed listening socket will cause a child to hold
    the accept mutex and block out new connections until another connection
    arrives on that rarely-accessed listening socket. This issue is known to
    affect some versions of AIX, Solaris, and Tru64; it is known to not affect
    FreeBSD or Linux.</p>
    </description>
    <affects prod="httpd" version="2.0.48" />
    <affects prod="httpd" version="2.0.47" />
    <affects prod="httpd" version="2.0.46" />
    <affects prod="httpd" version="2.0.45" />
    <affects prod="httpd" version="2.0.44" />
    <affects prod="httpd" version="2.0.43" />
    <affects prod="httpd" version="2.0.42" />
    <affects prod="httpd" version="2.0.40" />
    <affects prod="httpd" version="2.0.39" />
    <affects prod="httpd" version="2.0.37" />
    <affects prod="httpd" version="2.0.36" />
    <affects prod="httpd" version="2.0.35" />
    </issue>
- <issue fixed="1.3.31" public="20031015" released="20040512"
    reported="20031015">
    <cve name="CVE-2003-0993" />

```

```

<title>Allow/Deny parsing on big-endian 64-bit platforms</title>
<severity level="2">important</severity>
- <description>
  <p>A bug in the parsing of Allow/Deny rules using IP addresses without a
    netmask on big-endian 64-bit platforms causes the rules to fail to
    match.</p>
</description>
<affects prod="httpd" version="1.3.29" />
<affects prod="httpd" version="1.3.28" />
<affects prod="httpd" version="1.3.27" />
<affects prod="httpd" version="1.3.26" />
<affects prod="httpd" version="1.3.24" />
<affects prod="httpd" version="1.3.22" />
<affects prod="httpd" version="1.3.20" />
<affects prod="httpd" version="1.3.19" />
<affects prod="httpd" version="1.3.17" />
<affects prod="httpd" version="1.3.14" />
<affects prod="httpd" version="1.3.12" />
<affects prod="httpd" version="1.3.11" />
<affects prod="httpd" version="1.3.9" />
<affects prod="httpd" version="1.3.6" />
<affects prod="httpd" version="1.3.4" />
<affects prod="httpd" version="1.3.3" />
<affects prod="httpd" version="1.3.2" />
<affects prod="httpd" version="1.3.1" />
<affects prod="httpd" version="1.3.0" />
</issue>
- <issue fixed="2.0.49" public="20040220" released="20040319"
  reported="20040220">
  <cve name="CVE-2004-0113" />
  <severity level="2">important</severity>
  <title>mod_ssl memory leak</title>
- <description>
  <p>A memory leak in mod_ssl allows a remote denial of service attack against
    an SSL-enabled server by sending plain HTTP requests to the SSL port.</p>
</description>
<affects prod="httpd" version="2.0.48" />
<affects prod="httpd" version="2.0.47" />
<affects prod="httpd" version="2.0.46" />
<affects prod="httpd" version="2.0.45" />
<affects prod="httpd" version="2.0.44" />
<affects prod="httpd" version="2.0.43" />
<affects prod="httpd" version="2.0.42" />
<affects prod="httpd" version="2.0.40" />
<affects prod="httpd" version="2.0.39" />
<affects prod="httpd" version="2.0.37" />
<affects prod="httpd" version="2.0.36" />
<affects prod="httpd" version="2.0.35" />
</issue>

```

```

- <issue fixed="2.0.49" public="20030224" released="20040319"
  reported="20030224">
  <cve name="CVE-2003-0020" />
  <severity level="4">low</severity>
  <title>Error log escape filtering</title>
- <description>
  <p>Apache does not filter terminal escape sequences from error logs, which
    could make it easier for attackers to insert those sequences into terminal
    emulators containing vulnerabilities related to escape sequences.</p>
  </description>
  <affects prod="httpd" version="2.0.48" />
  <affects prod="httpd" version="2.0.47" />
  <affects prod="httpd" version="2.0.46" />
  <affects prod="httpd" version="2.0.45" />
  <affects prod="httpd" version="2.0.44" />
  <affects prod="httpd" version="2.0.43" />
  <affects prod="httpd" version="2.0.42" />
  <affects prod="httpd" version="2.0.40" />
  <affects prod="httpd" version="2.0.39" />
  <affects prod="httpd" version="2.0.37" />
  <affects prod="httpd" version="2.0.36" />
  <affects prod="httpd" version="2.0.35" />
  </issue>
- <issue fixed="2.0.48" public="20031027" released="20031027"
  reported="20031003">
  <cve name="CVE-2003-0789" />
  <title>CGI output information leak</title>
  <severity level="3">moderate</severity>
- <description>
  <p>A bug in mod_cgid mishandling of CGI redirect paths can result in CGI
    output going to the wrong client when a threaded MPM is used.</p>
  </description>
  <affects prod="httpd" version="2.0.47" />
  <affects prod="httpd" version="2.0.46" />
  <affects prod="httpd" version="2.0.45" />
  <affects prod="httpd" version="2.0.44" />
  <affects prod="httpd" version="2.0.43" />
  <affects prod="httpd" version="2.0.42" />
  <affects prod="httpd" version="2.0.40" />
  <affects prod="httpd" version="2.0.39" />
  <affects prod="httpd" version="2.0.37" />
  <affects prod="httpd" version="2.0.36" />
  <affects prod="httpd" version="2.0.35" />
  </issue>
- <issue fixed="1.3.29" public="20031027" released="20031027"
  reported="20030804">
  <cve name="CVE-2003-0542" />
  <severity level="4">low</severity>
  <title>Local configuration regular expression overflow</title>
- <description>

```

```

<p>By using a regular expression with more than 9 captures a buffer overflow can occur in mod_alias or mod_rewrite. To exploit this an attacker would need to be able to create a carefully crafted configuration file (.htaccess or httpd.conf)</p>
</description>
<affects prod="httpd" version="1.3.28" />
<affects prod="httpd" version="1.3.27" />
<affects prod="httpd" version="1.3.26" />
<affects prod="httpd" version="1.3.24" />
<affects prod="httpd" version="1.3.22" />
<affects prod="httpd" version="1.3.20" />
<affects prod="httpd" version="1.3.19" />
<affects prod="httpd" version="1.3.17" />
<affects prod="httpd" version="1.3.14" />
<affects prod="httpd" version="1.3.12" />
<affects prod="httpd" version="1.3.11" />
<affects prod="httpd" version="1.3.9" />
<affects prod="httpd" version="1.3.6" />
<affects prod="httpd" version="1.3.4" />
<affects prod="httpd" version="1.3.3" />
<affects prod="httpd" version="1.3.2" />
<affects prod="httpd" version="1.3.1" />
<affects prod="httpd" version="1.3.0" />
</issue>
- <issue fixed="2.0.48" public="20031027" released="20031027"
  reported="20030804">
  <cve name="CVE-2003-0542" />
  <severity level="4">low</severity>
  <title>Local configuration regular expression overflow</title>
- <description>
  <p>By using a regular expression with more than 9 captures a buffer overflow can occur in mod_alias or mod_rewrite. To exploit this an attacker would need to be able to create a carefully crafted configuration file (.htaccess or httpd.conf)</p>
  </description>
  <affects prod="httpd" version="2.0.47" />
  <affects prod="httpd" version="2.0.46" />
  <affects prod="httpd" version="2.0.45" />
  <affects prod="httpd" version="2.0.44" />
  <affects prod="httpd" version="2.0.43" />
  <affects prod="httpd" version="2.0.42" />
  <affects prod="httpd" version="2.0.40" />
  <affects prod="httpd" version="2.0.39" />
  <affects prod="httpd" version="2.0.37" />
  <affects prod="httpd" version="2.0.36" />
  <affects prod="httpd" version="2.0.35" />
  </issue>
- <issue fixed="1.3.28" public="20030718" released="20030718"
  reported="20030704">
  <cve name="CVE-2003-0460" />

```

```

<severity level="2">important</severity>
<title>RotateLogs DoS</title>
- <description>
  <p>The rotatelogs support program on Win32 and OS/2 would quit logging
    and exit if it received special control characters such as 0x1A.</p>
</description>
<affects prod="httpd" version="1.3.27" />
<maybeaffects prod="httpd" version="1.3.26" />
<maybeaffects prod="httpd" version="1.3.24" />
<maybeaffects prod="httpd" version="1.3.22" />
<maybeaffects prod="httpd" version="1.3.20" />
<maybeaffects prod="httpd" version="1.3.19" />
<maybeaffects prod="httpd" version="1.3.17" />
<maybeaffects prod="httpd" version="1.3.14" />
<maybeaffects prod="httpd" version="1.3.12" />
<maybeaffects prod="httpd" version="1.3.11" />
<maybeaffects prod="httpd" version="1.3.9" />
<maybeaffects prod="httpd" version="1.3.6" />
<maybeaffects prod="httpd" version="1.3.4" />
<maybeaffects prod="httpd" version="1.3.3" />
<maybeaffects prod="httpd" version="1.3.2" />
<maybeaffects prod="httpd" version="1.3.1" />
<maybeaffects prod="httpd" version="1.3.0" />
</issue>
- <issue fixed="2.0.47" public="20030709" released="20030709"
  reported="20030625">
  <cve name="CVE-2003-0254" />
  <severity level="3">moderate</severity>
  <title>Remote DoS via IPv6 ftp proxy</title>
- <description>
  <p>When a client requests that proxy ftp connect to a ftp server with IPv6
    address, and the proxy is unable to create an IPv6 socket, an infinite loop
    occurs causing a remote Denial of Service.</p>
</description>
<affects prod="httpd" version="2.0.46" />
<affects prod="httpd" version="2.0.45" />
<affects prod="httpd" version="2.0.44" />
<affects prod="httpd" version="2.0.43" />
<affects prod="httpd" version="2.0.42" />
<affects prod="httpd" version="2.0.40" />
<affects prod="httpd" version="2.0.39" />
<affects prod="httpd" version="2.0.37" />
<affects prod="httpd" version="2.0.36" />
<affects prod="httpd" version="2.0.35" />
</issue>
- <issue fixed="2.0.47" public="20030709" released="20030709"
  reported="20030625">
  <cve name="CVE-2003-0253" />
  <severity level="2">important</severity>

```

```

<title>Remote DoS with multiple Listen directives</title>
- <description>
  <p>In a server with multiple listening sockets a certain error returned by
    accept() on a rarely access port can cause a temporary denial of service,
    due to a bug in the prefork MPM.</p>
</description>
<affects prod="httpd" version="2.0.46" />
<affects prod="httpd" version="2.0.45" />
<affects prod="httpd" version="2.0.44" />
<affects prod="httpd" version="2.0.43" />
<affects prod="httpd" version="2.0.42" />
<affects prod="httpd" version="2.0.40" />
<affects prod="httpd" version="2.0.39" />
<affects prod="httpd" version="2.0.37" />
<affects prod="httpd" version="2.0.36" />
<affects prod="httpd" version="2.0.35" />
</issue>
- <issue fixed="2.0.47" public="20030709" released="20030709"
  reported="20030430">
  <cve name="CVE-2003-0192" />
  <title>mod_ssl renegotiation issue</title>
  <severity level="4">low</severity>
- <description>
  <p>A bug in the optional renegotiation code in mod_ssl included with Apache
    httpd can cause cipher suite restrictions to be ignored. This is triggered if
    optional renegotiation is used (SSLOptions +OptRenegotiate) along with
    verification of client certificates and a change to the cipher suite over the
    renegotiation.</p>
</description>
<affects prod="httpd" version="2.0.46" />
<affects prod="httpd" version="2.0.45" />
<affects prod="httpd" version="2.0.44" />
<affects prod="httpd" version="2.0.43" />
<affects prod="httpd" version="2.0.42" />
<affects prod="httpd" version="2.0.40" />
<affects prod="httpd" version="2.0.39" />
<affects prod="httpd" version="2.0.37" />
<affects prod="httpd" version="2.0.36" />
<affects prod="httpd" version="2.0.35" />
</issue>
- <issue fixed="2.0.46" public="20030528" released="20030528"
  reported="20030409">
  <cve name="CVE-2003-0245" />
  <severity level="1">critical</severity>
  <title>APR remote crash</title>
- <description>
  <p>A vulnerability in the apr_psprintf function in the Apache Portable Runtime
    (APR) library allows remote attackers to cause a denial of service (crash)
    and possibly execute arbitrary code via long strings, as demonstrated using
    XML objects to mod_dav, and possibly other vectors.</p>

```

```

    </description>
    <affects prod="httpd" version="2.0.45" />
    <affects prod="httpd" version="2.0.44" />
    <affects prod="httpd" version="2.0.43" />
    <affects prod="httpd" version="2.0.42" />
    <affects prod="httpd" version="2.0.40" />
    <affects prod="httpd" version="2.0.39" />
    <affects prod="httpd" version="2.0.37" />
    </issue>
- <issue fixed="2.0.46" public="20030528" released="20030528"
    reported="20030425">
    <cve name="CVE-2003-0189" />
    <severity level="2">important</severity>
    <title>Basic Authentication DoS</title>
- <description>
    <p>A build system problem in Apache 2.0.40 through 2.0.45 allows remote
        attackers to cause a denial of access to authenticated content when a
        threaded server is used.</p>
    </description>
    <affects prod="httpd" version="2.0.45" />
    <affects prod="httpd" version="2.0.44" />
    <affects prod="httpd" version="2.0.43" />
    <affects prod="httpd" version="2.0.42" />
    <affects prod="httpd" version="2.0.40" />
    </issue>
- <issue fixed="2.0.46" public="20040402" released="20040402">
    <cve name="CVE-2003-0134" />
    <severity level="2">important</severity>
    <title>OS2 device name DoS</title>
- <description>
    <p>Apache on OS2 up to and including Apache 2.0.45 have a Denial of Service
        vulnerability caused by device names.</p>
    </description>
    <affects prod="httpd" version="2.0.45" />
    <maybeffects prod="httpd" version="2.0.44" />
    <maybeffects prod="httpd" version="2.0.43" />
    <maybeffects prod="httpd" version="2.0.42" />
    <maybeffects prod="httpd" version="2.0.40" />
    <maybeffects prod="httpd" version="2.0.39" />
    <maybeffects prod="httpd" version="2.0.37" />
    <maybeffects prod="httpd" version="2.0.36" />
    <maybeffects prod="httpd" version="2.0.35" />
    </issue>
- <issue fixed="2.0.46" released="20040402" public="20030224"
    reported="20030224">
    <cve name="CVE-2003-0083" />
    <severity level="4">low</severity>
    <title>Filtered escape sequences</title>
- <description>

```



```

<p>Apache did not filter terminal escape sequences from its access logs,
which could make it easier for attackers to insert those sequences into
terminal emulators containing vulnerabilities related to escape
sequences.</p>
</description>
<affects prod="httpd" version="2.0.45" />
<affects prod="httpd" version="2.0.44" />
<affects prod="httpd" version="2.0.43" />
<affects prod="httpd" version="2.0.42" />
<affects prod="httpd" version="2.0.40" />
<affects prod="httpd" version="2.0.39" />
<affects prod="httpd" version="2.0.37" />
<affects prod="httpd" version="2.0.36" />
<affects prod="httpd" version="2.0.35" />
</issue>
- <issue fixed="2.0.45" public="20040402" released="20040402">
  <cve name="CVE-2003-0132" />
  <severity level="2">important</severity>
  <title>Line feed memory leak DoS</title>
- <description>
  <p>Apache 2.0 versions before Apache 2.0.45 had a significant Denial of
Service vulnerability. Remote attackers could cause a denial of service
(memory consumption) via large chunks of linefeed characters, which
causes Apache to allocate 80 bytes for each linefeed.</p>
</description>
<affects prod="httpd" version="2.0.44" />
<affects prod="httpd" version="2.0.43" />
<affects prod="httpd" version="2.0.42" />
<affects prod="httpd" version="2.0.40" />
<affects prod="httpd" version="2.0.39" />
<affects prod="httpd" version="2.0.37" />
<affects prod="httpd" version="2.0.36" />
<affects prod="httpd" version="2.0.35" />
</issue>
- <issue fixed="2.0.44" public="20030120" released="20030120"
  reported="20021204">
  <cve name="CVE-2003-0016" />
  <severity level="1">critical</severity>
  <flaw type="msdos-device" />
  <title>MS-DOS device name filtering</title>
- <description>
  <p>On Windows platforms Apache did not correctly filter MS-DOS device
names which could lead to denial of service attacks or remote code
execution.</p>
</description>
<affects prod="httpd" version="2.0.43" />
<maybeaffects prod="httpd" version="2.0.42" />
<maybeaffects prod="httpd" version="2.0.40" />
<maybeaffects prod="httpd" version="2.0.39" />
<maybeaffects prod="httpd" version="2.0.37" />

```

```

<maybeaffects prod="httpd" version="2.0.36" />
<maybeaffects prod="httpd" version="2.0.35" />
</issue>
= <issue fixed="2.0.44" public="20030120" released="20030120"
    reported="20021115">
    <cve name="CVE-2003-0017" />
    <flaw type="unk" />
    <severity level="2">important</severity>
    <title>Apache can serve unexpected files</title>
= <description>
    <p>On Windows platforms Apache could be forced to serve unexpected files
        by appending illegal characters such as '<' to the request URL</p>
    </description>
    <affects prod="httpd" version="2.0.43" />
    <maybeaffects prod="httpd" version="2.0.42" />
    <maybeaffects prod="httpd" version="2.0.40" />
    <maybeaffects prod="httpd" version="2.0.39" />
    <maybeaffects prod="httpd" version="2.0.37" />
    <maybeaffects prod="httpd" version="2.0.36" />
    <maybeaffects prod="httpd" version="2.0.35" />
    </issue>
= <issue fixed="1.3.27" public="20021003" released="20021003"
    reported="20020923">
    <cve name="CVE-2002-0843" />
    <severity level="2">important</severity>
    <flaw type="buf" />
    <title>Buffer overflows in ab utility</title>
= <description>
    <p>Buffer overflows in the benchmarking utility ab could be exploited if ab is
        run against a malicious server</p>
    </description>
    <affects prod="httpd" version="1.3.26" />
    <affects prod="httpd" version="1.3.24" />
    <affects prod="httpd" version="1.3.22" />
    <affects prod="httpd" version="1.3.20" />
    <affects prod="httpd" version="1.3.19" />
    <affects prod="httpd" version="1.3.17" />
    <affects prod="httpd" version="1.3.14" />
    <affects prod="httpd" version="1.3.12" />
    <affects prod="httpd" version="1.3.11" />
    <affects prod="httpd" version="1.3.9" />
    <affects prod="httpd" version="1.3.6" />
    <affects prod="httpd" version="1.3.4" />
    <affects prod="httpd" version="1.3.3" />
    <affects prod="httpd" version="1.3.2" />
    <affects prod="httpd" version="1.3.1" />
    <affects prod="httpd" version="1.3.0" />
    </issue>

```

```

- <issue fixed="1.3.27" public="20021003" released="20021003"
  reported="20011111">
  <cve name="CVE-2002-0839" />
  <severity level="2">important</severity>
  <flaw type="perm" />
  <title>Shared memory permissions lead to local privilege escalation</title>
- <description>
  <p>The permissions of the shared memory used for the scoreboard allows an
    attacker who can execute under the Apache UID to send a signal to any
    process as root or cause a local denial of service attack.</p>
  </description>
  <affects prod="httpd" version="1.3.26" />
  <affects prod="httpd" version="1.3.24" />
  <affects prod="httpd" version="1.3.22" />
  <affects prod="httpd" version="1.3.20" />
  <affects prod="httpd" version="1.3.19" />
  <affects prod="httpd" version="1.3.17" />
  <affects prod="httpd" version="1.3.14" />
  <affects prod="httpd" version="1.3.12" />
  <affects prod="httpd" version="1.3.11" />
  <affects prod="httpd" version="1.3.9" />
  <affects prod="httpd" version="1.3.6" />
  <affects prod="httpd" version="1.3.4" />
  <affects prod="httpd" version="1.3.3" />
  <affects prod="httpd" version="1.3.2" />
  <affects prod="httpd" version="1.3.1" />
  <affects prod="httpd" version="1.3.0" />
  </issue>
- <issue fixed="2.0.43" public="20021002" released="20021003"
  reported="20020920">
  <cve name="CVE-2002-0840" />
  <flaw type="css" />
  <severity level="4">low</severity>
  <title>Error page XSS using wildcard DNS</title>
- <description>
  <p>Cross-site scripting (XSS) vulnerability in the default error page of Apache
    2.0 before 2.0.43, and 1.3.x up to 1.3.26, when UseCanonicalName is "Off"
    and support for wildcard DNS is present, allows remote attackers to execute
    script as other web page visitors via the Host: header.</p>
  </description>
  <affects prod="httpd" version="2.0.42" />
  <affects prod="httpd" version="2.0.40" />
  <affects prod="httpd" version="2.0.39" />
  <affects prod="httpd" version="2.0.37" />
  <affects prod="httpd" version="2.0.36" />
  <affects prod="httpd" version="2.0.35" />
  </issue>
- <issue fixed="2.0.43" released="20021003">
  <cve name="CVE-2002-1156" />
  <flaw type="unk" />

```

```

<severity level="3">moderate</severity>
<title>CGI scripts source revealed using WebDAV</title>
- <description>
  <p>In Apache 2.0.42 only, for a location where both WebDAV and CGI were
    enabled, a POST request to a CGI script would reveal the CGI source to a
    remote user.</p>
</description>
<affects prod="httpd" version="2.0.42" />
</issue>
- <issue fixed="2.0.42" public="20020919" released="20020924">
  <cve name="CVE-2002-1593" />
  <severity level="3">moderate</severity>
  <title>mod_dav crash</title>
- <description>
  <p>A flaw was found in handling of versioning hooks in mod_dav. An attacker
    could send a carefully crafted request in such a way to cause the child
    process handling the connection to crash. This issue will only result in a
    denial of service where a threaded process model is in use.</p>
</description>
<affects prod="httpd" version="2.0.40" />
<affects prod="httpd" version="2.0.39" />
<affects prod="httpd" version="2.0.37" />
<affects prod="httpd" version="2.0.36" />
<affects prod="httpd" version="2.0.35" />
</issue>
- <issue fixed="1.3.27" public="20021002" released="20021003"
  reported="20020920">
  <cve name="CVE-2002-0840" />
  <severity level="4">low</severity>
  <title>Error page XSS using wildcard DNS</title>
  <flaw type="css" />
- <description>
  <p>Cross-site scripting (XSS) vulnerability in the default error page of Apache
    2.0 before 2.0.43, and 1.3.x up to 1.3.26, when UseCanonicalName is "Off"
    and support for wildcard DNS is present, allows remote attackers to execute
    script as other web page visitors via the Host: header.</p>
</description>
<affects prod="httpd" version="1.3.26" />
<affects prod="httpd" version="1.3.24" />
<affects prod="httpd" version="1.3.22" />
<affects prod="httpd" version="1.3.20" />
<affects prod="httpd" version="1.3.19" />
<affects prod="httpd" version="1.3.17" />
<affects prod="httpd" version="1.3.14" />
<affects prod="httpd" version="1.3.12" />
<affects prod="httpd" version="1.3.11" />
<affects prod="httpd" version="1.3.9" />
<affects prod="httpd" version="1.3.6" />
<affects prod="httpd" version="1.3.4" />
<affects prod="httpd" version="1.3.3" />

```

```

<affects prod="httpd" version="1.3.2" />
<affects prod="httpd" version="1.3.1" />
<affects prod="httpd" version="1.3.0" />
</issue>
- <issue fixed="2.0.40" public="20020809" released="20020809"
  reported="20020807">
  <title>Path vulnerability</title>
  <severity level="2">important</severity>
  <flaw type="priv" />
- <description>
  <p>Certain URIs would bypass security and allow users to invoke or access
    any file depending on the system configuration. Affects Windows, OS2,
    Netware and Cygwin platforms only.</p>
  </description>
  <os>win32</os>
  <os>netware</os>
  <os>os2</os>
  <os>cygwin</os>
  <cve name="CVE-2002-0661" />
  <affects prod="httpd" version="2.0.39" />
  <affects prod="httpd" version="2.0.37" />
  <affects prod="httpd" version="2.0.36" />
  <affects prod="httpd" version="2.0.35" />
  </issue>
- <issue fixed="2.0.40" public="20020809" released="20020809"
  reported="20020705">
  <title>Path revealing exposures</title>
  <severity level="4">low</severity>
  <flaw type="unk" />
- <description>
  <p>A path-revealing exposure was present in multiview type map negotiation
    (such as the default error documents) where a module would report the full
    path of the typemapped .var file when multiple documents or no documents
    could be served. Additionally a path-revealing exposure in cgi/cgid when
    Apache fails to invoke a script. The modules would report "couldn't create
    child process /path-to-script/script.pl" revealing the full path of the
    script.</p>
  </description>
  <cve name="CVE-2002-0654" />
  <affects prod="httpd" version="2.0.39" />
  <maybeaffects prod="httpd" version="2.0.37" />
  <maybeaffects prod="httpd" version="2.0.36" />
  <maybeaffects prod="httpd" version="2.0.35" />
  </issue>
- <issue fixed="2.0.37" public="20020617" released="20020618"
  reported="20020527">
  <title>Apache Chunked encoding vulnerability</title>
  <severity level="1">critical</severity>
  <flaw type="buf" />
- <description>

```

```

<p>Malicious requests can cause various effects ranging from a relatively harmless increase in system resources through to denial of service attacks and in some cases the ability to execute arbitrary remote code.</p>
</description>
<cve name="CVE-2002-0392" />
<affects prod="httpd" version="2.0.36" />
<affects prod="httpd" version="2.0.35" />
</issue>
- <issue fixed="2.0.36" public="20020422" released="20020508">
  <cve name="CVE-2002-1592" />
  <severity issue="4">low</severity>
  <title>Warning messages could be displayed to users</title>
- <description>
  <p>In some cases warning messages could get returned to end users in addition to being recorded in the error log. This could reveal the path to a CGI script for example, a minor security exposure.</p>
  </description>
  <affects prod="httpd" version="2.0.35" />
  </issue>
- <issue fixed="1.3.26" public="20020617" released="20020618"
  reported="20020527">
  <title>Apache Chunked encoding vulnerability</title>
  <severity level="1">critical</severity>
  <flaw type="buf" />
- <description>
  <p>Requests to all versions of Apache 1.3 can cause various effects ranging from a relatively harmless increase in system resources through to denial of service attacks and in some cases the ability to be remotely exploited.</p>
  </description>
  <cve name="CVE-2002-0392" />
  <affects prod="httpd" version="1.3.24" />
  <affects prod="httpd" version="1.3.22" />
  <affects prod="httpd" version="1.3.20" />
  <affects prod="httpd" version="1.3.19" />
  <affects prod="httpd" version="1.3.17" />
  <affects prod="httpd" version="1.3.14" />
  <affects prod="httpd" version="1.3.12" />
  <affects prod="httpd" version="1.3.11" />
  <affects prod="httpd" version="1.3.9" />
  <affects prod="httpd" version="1.3.6" />
  <affects prod="httpd" version="1.3.4" />
  <affects prod="httpd" version="1.3.3" />
  <affects prod="httpd" version="1.3.2" />
  <affects prod="httpd" version="1.3.1" />
  <affects prod="httpd" version="1.3.0" />
  </issue>
- <issue fixed="1.3.26" released="20020618" reported="20030224"
  public="20030224">
  <cve name="CVE-2003-0083" />
  <severity level="4">low</severity>

```

```

<title>Filtered escape sequences</title>
= <description>
<p>Apache does not filter terminal escape sequences from its access logs,
which could make it easier for attackers to insert those sequences into
terminal emulators containing vulnerabilities related to escape
sequences,</p>
</description>
<affects prod="httpd" version="1.3.24" />
<affects prod="httpd" version="1.3.22" />
<affects prod="httpd" version="1.3.20" />
<affects prod="httpd" version="1.3.19" />
<affects prod="httpd" version="1.3.17" />
<affects prod="httpd" version="1.3.14" />
<affects prod="httpd" version="1.3.12" />
<affects prod="httpd" version="1.3.11" />
<affects prod="httpd" version="1.3.9" />
<affects prod="httpd" version="1.3.6" />
<affects prod="httpd" version="1.3.4" />
<affects prod="httpd" version="1.3.3" />
<affects prod="httpd" version="1.3.2" />
<affects prod="httpd" version="1.3.1" />
<affects prod="httpd" version="1.3.0" />
</issue>
= <issue fixed="1.3.24" released="20020322" reported="20020213">
<severity level="1">critical</severity>
<title>Win32 Apache Remote command execution</title>
<os>win32</os>
<cve name="CVE-2002-0061" />
<flaw type="metachar" />
= <description>
<p>Apache for Win32 before 1.3.24 and 2.0.34-beta allows remote attackers
to execute arbitrary commands via parameters passed to batch file CGI
scripts.</p>
</description>
<affects prod="httpd" version="1.3.22" />
<maybeffects prod="httpd" version="1.3.20" />
<maybeffects prod="httpd" version="1.3.19" />
<maybeffects prod="httpd" version="1.3.17" />
<maybeffects prod="httpd" version="1.3.14" />
<maybeffects prod="httpd" version="1.3.12" />
<maybeffects prod="httpd" version="1.3.11" />
<maybeffects prod="httpd" version="1.3.9" />
<maybeffects prod="httpd" version="1.3.6" />
<maybeffects prod="httpd" version="1.3.4" />
<maybeffects prod="httpd" version="1.3.3" />
<maybeffects prod="httpd" version="1.3.2" />
<maybeffects prod="httpd" version="1.3.1" />
<maybeffects prod="httpd" version="1.3.0" />
</issue>

```

```

- <issue fixed="1.3.22" released="20011012" public="20010928"
  reported="20010918">
  <title>Requests can cause directory listing to be displayed</title>
  <severity level="2">important</severity>
  <flaw type="unk" />
- <description>
  <p>A vulnerability was found in the Win32 port of Apache 1.3.20. A client
    submitting a very long URI could cause a directory listing to be returned
    rather than the default index page.</p>
  </description>
  <os>win32</os>
  <cve name="CVE-2001-0729" />
  <affects prod="httpd" version="1.3.20" />
  </issue>
- <issue fixed="1.3.22" released="20011012" public="20010928">
  <severity level="3">moderate</severity>
  <title>split-logfile can cause arbitrary log files to be written to</title>
- <description>
- <p>
  A vulnerability was found in the
  <samp>split-logfile</samp>
  support program. A request with a specially crafted
  <samp>Host:</samp>
  header could allow any file with a
  <samp>.log</samp>
  extension on the system to be written to.
  </p>
  </description>
  <os>all</os>
  <cve name="CVE-2001-0730" />
  <flaw type="dot" />
  <bug pr="7848" />
  <affects prod="httpd" version="1.3.20" />
  <affects prod="httpd" version="1.3.19" />
  <affects prod="httpd" version="1.3.17" />
  <affects prod="httpd" version="1.3.14" />
  <affects prod="httpd" version="1.3.12" />
  <affects prod="httpd" version="1.3.11" />
  <affects prod="httpd" version="1.3.9" />
  <affects prod="httpd" version="1.3.6" />
  <affects prod="httpd" version="1.3.4" />
  <affects prod="httpd" version="1.3.3" />
  <affects prod="httpd" version="1.3.2" />
  <affects prod="httpd" version="1.3.1" />
  <affects prod="httpd" version="1.3.0" />
  </issue>
- <issue fixed="1.3.22" released="20011012" public="20010709">
  - <!--
    * BUGTRAQ:20010709 How Google indexed a file with no external link

```



```

*
URL:http://www.securityfocus.com/archive/1/20010709214744.A28765@brassc
annon.net
* CONFIRM:http://www.apacheweek.com/issues/01-10-05#security
* BID:3009
* URL:http://www.securityfocus.com/bid/3009
-->
<severity level="2">important</severity>
<title>Multiviews can cause a directory listing to be displayed</title>
= <description>
= <p>
=   A vulnerability was found when
<directive>Multiviews</directive>
are used to negotiate the directory index. In some configurations,
requesting a URI with a
<samp>QUERY_STRING</samp>
of
<samp>M=D</samp>
could return a directory listing rather than the expected index page.
</p>
</description>
<cve name="CVE-2001-0731" />
<flaw type="other" />
<affects prod="httpd" version="1.3.20" />
<maybeaffects prod="httpd" version="1.3.19" />
<maybeaffects prod="httpd" version="1.3.17" />
<maybeaffects prod="httpd" version="1.3.14" />
<maybeaffects prod="httpd" version="1.3.12" />
<maybeaffects prod="httpd" version="1.3.11" />
<maybeaffects prod="httpd" version="1.3.9" />
<maybeaffects prod="httpd" version="1.3.6" />
<maybeaffects prod="httpd" version="1.3.4" />
<maybeaffects prod="httpd" version="1.3.3" />
<maybeaffects prod="httpd" version="1.3.2" />
<maybeaffects prod="httpd" version="1.3.1" />
<maybeaffects prod="httpd" version="1.3.0" />
</issue>
= <issue fixed="1.3.20" released="20010522">
<title>Denial of service attack on Win32 and OS2</title>
<cve name="CVE-2001-1342" />
<severity level="2">important</severity>
<flaw type="dos-malform" />
= <description>
= <p>A vulnerability was found in the Win32 and OS2 ports of Apache 1.3. A
client submitting a carefully constructed URI could cause a General
Protection Fault in a child process, bringing up a message box which would
have to be cleared by the operator to resume operation. This vulnerability
introduced no identified means to compromise the server other than
introducing a possible denial of service.</p>
</description>

```

```

- <!--
  http://www.securiteam.com/windowsntfocus/5IP0Q0K4AU.html
-->
<os>win32</os>
<os>os2</os>
<affects prod="httpd" version="1.3.20" />
<maybeaffects prod="httpd" version="1.3.19" />
<maybeaffects prod="httpd" version="1.3.17" />
<maybeaffects prod="httpd" version="1.3.14" />
<maybeaffects prod="httpd" version="1.3.12" />
<maybeaffects prod="httpd" version="1.3.11" />
<maybeaffects prod="httpd" version="1.3.9" />
<maybeaffects prod="httpd" version="1.3.6" />
<maybeaffects prod="httpd" version="1.3.4" />
<maybeaffects prod="httpd" version="1.3.3" />
<maybeaffects prod="httpd" version="1.3.2" />
<maybeaffects prod="httpd" version="1.3.1" />
<maybeaffects prod="httpd" version="1.3.0" />
</issue>
= <issue fixed="1.3.19" released="20010228">
  - <!--
    apcore 200102
  -->
  <title>Requests can cause directory listing to be displayed</title>
  <severity level="2">important</severity>
= <description>
= <p>
  The default installation can lead
  <samp>mod_negotiation</samp>
  and
  <samp>mod_dir</samp>
  or
  <samp>mod_autoindex</samp>
  to display a directory listing instead of the multiview index.html file if a very
  long path was created artificially by using many slashes.
  </p>
  </description>
= <resolution>
  <p>From Apache 1.3.19 a 403 (Forbidden) response is given.</p>
  </resolution>
= <exploit>
  <p>In order to exploit this bug the server has to have "Options +MultiViews"
  enabled and be using multiviews to determine which document to send as a
  directory index. By constructing a GET request with the right number of
  trailing slashes the directory index will be displayed instead of the default
  document index. The number of trailing slashes required depends on the
  directory requested, where the full path is around the OS file limit, usually
  1024 characters. With too few trailing slashes the index.html file will be
  displayed, with too many a 403 (forbidden) response will be given.</p>
  </exploit>

```

```

<cve name="CVE-2001-0925" />
<flaw type="unk" />
<affects prod="httpd" version="1.3.17" />
<affects prod="httpd" version="1.3.14" />
<affects prod="httpd" version="1.3.12" />
<affects prod="httpd" version="1.3.11" />
  </issue>
- <!--
  1.3.19
  * NetWare is a case insensitive file system so all directory and file
    names are now compared in a case insensitive manner to avoid security
    holes.
  -->
= <issue fixed="1.3.14" released="20001013" public="20000929">
- <!--
  RHSA-2000:088-04
  -->
<cve name="CVE-2000-0913" public="20000929" />
<severity level="2">important</severity>
<title>Rewrite rules that include references allow access to any file</title>
= <description>
= <p>
=   The Rewrite module,
<samp>mod_rewrite</samp>
  , can allow access to any file on the web server. The vulnerability occurs
  only with certain specific cases of using regular expression references in
<samp>RewriteRule</samp>
  directives: If the destination of a
<samp>RewriteRule</samp>
  contains regular expression references then an attacker will be able to
  access any file on the server.
  </p>
</description>
<exploit>RewriteRule /test/(.*) /usr/local/data/test-stuff/\$1 RewriteRule
  /more-icons/(.*) /icons/\$1</exploit>
<os>all</os>
<affects prod="httpd" version="1.3.12" />
<maybeaffects prod="httpd" version="1.3.11" />
<maybeaffects prod="httpd" version="1.3.9" />
<maybeaffects prod="httpd" version="1.3.6" />
<maybeaffects prod="httpd" version="1.3.4" />
<maybeaffects prod="httpd" version="1.3.3" />
<maybeaffects prod="httpd" version="1.3.2" />
<maybeaffects prod="httpd" version="1.3.1" />
<maybeaffects prod="httpd" version="1.3.0" />
  </issue>
- <!--
  i don't think this one actually exists, I looked
    through the cvsweb and think it's a duplicate
  -->

```

```

- <issue fixed="1.3.14" released="20001013">
- <severity level="2">important</severity>
- <title>Mass virtual hosting can display CGI source</title>
- <description>
- <p>
-   A security problem for users of the mass virtual hosting module,
-   <samp>mod_vhost_alias</samp>
-   , causes the source to a CGI to be sent if the
-   <samp>cgi-bin</samp>
-   directory is under the document root. However, it is not normal to have your
-   cgi-bin directory under a document root.
-   </p>
-   </description>
-   <os>all</os>
-   <cve name="CVE-2000-1204" />
-   <flaw type="unk" />
-   <affects prod="httpd" version="1.3.12" />
-   <affects prod="httpd" version="1.3.11" />
-   <affects prod="httpd" version="1.3.9" />
-   </issue>
- <issue fixed="1.3.14" released="20001013">
- <cve name="CVE-2000-0505" />
- <severity level="3">moderate</severity>
- <title>Requests can cause directory listing to be displayed on NT</title>
- <description>
- <p>A security hole on Apache for Windows allows a user to view the listing of
-   a directory instead of the default HTML page by sending a carefully
-   constructed request.</p>
-   </description>
-   <os>win32</os>
-   <flaw type="unk" />
-   <affects prod="httpd" os="win32" version="1.3.12" />
-   <maybeffects prod="httpd" version="1.3.11" />
-   <maybeffects prod="httpd" version="1.3.9" />
-   <maybeffects prod="httpd" version="1.3.6" />
-   <maybeffects prod="httpd" version="1.3.4" />
-   <maybeffects prod="httpd" version="1.3.3" />
-   <maybeffects prod="httpd" version="1.3.2" />
-   <maybeffects prod="httpd" version="1.3.1" />
-   <maybeffects prod="httpd" version="1.3.0" />
-   </issue>
- <issue fixed="1.3.12" released="20000225">
- <severity level="2">important</severity>
- <title>Cross-site scripting can reveal private session information</title>
- <description>
- <p>Apache was vulnerable to cross site scripting issues. It was shown that
-   malicious HTML tags can be embedded in client web requests if the server or
-   script handling the request does not carefully encode all information
-   displayed to the user. Using these vulnerabilities attackers could, for

```

**example, obtain copies of your private cookies used to authenticate you to other sites.**

```
</description>
<cve name="CVE-2000-1205" />
<flaw type="css" />
<os>all</os>
<affects prod="httpd" version="1.3.11" />
<affects prod="httpd" version="1.3.9" />
<affects prod="httpd" version="1.3.6" />
<affects prod="httpd" version="1.3.4" />
<affects prod="httpd" version="1.3.3" />
<affects prod="httpd" version="1.3.2" />
<affects prod="httpd" version="1.3.1" />
<affects prod="httpd" version="1.3.0" />
</issue>
- <issue fixed="1.3.11" released="20000121">
- <severity level="3">moderate</severity>
- <title>Mass virtual hosting security issue</title>
- <description>
- <p>
  A security problem can occur for sites using mass name-based virtual
  hosting (using the new
  <samp>mod_vhost_alias</samp>
  module) or with special
  <samp>mod_rewrite</samp>
  rules.
  - <!--
    Makes sure vhost alias can only be alnum, - or .
    -->
  </p>
</description>
<os>all</os>
<cve name="CVE-2000-1206" />
<flaw type="unk" />
<affects prod="httpd" version="1.3.9" />
- <!--
  mod_rewrite stuff only below
  -->
<maybeaffects prod="httpd" version="1.3.6" />
<maybeaffects prod="httpd" version="1.3.4" />
<maybeaffects prod="httpd" version="1.3.3" />
<maybeaffects prod="httpd" version="1.3.2" />
<maybeaffects prod="httpd" version="1.3.1" />
<maybeaffects prod="httpd" version="1.3.0" />
</issue>
- <issue fixed="1.3.4" released="19990111">
- <severity level="2">important</severity>
- <title>Denial of service attack on Win32</title>
- <description>
```

```

<p>There have been a number of important security fixes to Apache on
Windows. The most important is that there is much better protection
against people trying to access special DOS device names (such as
"nul").</p>
</description>
<os>win32</os>
<affects prod="httpd" version="1.3.3" />
<affects prod="httpd" version="1.3.2" />
<affects prod="httpd" version="1.3.1" />
<affects prod="httpd" version="1.3.0" />
</issue>
= <issue fixed="1.3.2" released="19980923">
  <cve name="CVE-1999-1199" />
  <severity level="2">important</severity>
  <flaw type="memleak" />
  <title>Multiple header Denial of Service vulnerability</title>
= <description>
  = <p>A serious problem exists when a client sends a large number of headers
    with the same header name. Apache uses up memory faster than the
    amount of memory required to simply store the received data itself. That is,
    memory use increases faster and faster as more headers are received,
    rather than increasing at a constant rate. This makes a denial of service
    attack based on this method more effective than methods which cause
    Apache to use memory at a constant rate, since the attacker has to send
    less data.</p>
  </description>
  <os>all</os>
  <affects prod="httpd" version="1.3.1" />
  <affects prod="httpd" version="1.3.0" />
  </issue>
= <issue fixed="1.3.2" released="19980923">
  <title>Denial of service attacks</title>
  <severity level="2">important</severity>
= <description>
= <p>
  Apache 1.3.2 has better protection against denial of service attacks. These
  are when people make excessive requests to the server to try and prevent
  other people using it. In 1.3.2 there are several new directives which can
  limit the size of requests (these directives all start with the word
  <SAMP>Limit</SAMP>
).
  </p>
  </description>
  <flaw type="msdos-device" />
  <os>all</os>
  <affects prod="httpd" version="1.3.1" />
  <affects prod="httpd" version="1.3.0" />
  </issue>
- <!--

```

- \* Avoid denial of service attacks if a configuration file (such as a .htaccess file) is a device file, by refusing to open device files apart from /dev/null which is still valid (1.3.0)
- \* Correctly handle over-long lines in configuration files (1.3.0)
  - \* Fix denial of service attack by sending requests with lots of slashes in them (1.3.0)
  - \* Deny access to directories if a .htaccess file in that directory cannot be read (1.3.0)

-->  
</security>

## ***Appendix C: Security Record Processing Scripts***

The following software was written to process the raw security records above. It is a collection of Java classes aimed to enable the reader to reproduce these results as well as to analyze updated security records that will be published following the release of this thesis.

```
import java.util.Date;

final class Issue {
    private Date released;
    private Date reported;
    private Date pub;
    private String severity;
    private String cve;
    private String releaseFixed;

    void setCve(final String theCve) {
        cve = theCve;
    }

    String getCve() {
        return cve;
    }

    void setReleaseFixed(final String fix) {
        releaseFixed = fix;
    }

    void setReleased(final Date rel) {
        released = rel;
    }

    void setReported(final Date rep) {
        reported = rep;
    }

    void setPub(final Date pubDate) {
        pub = pubDate;
    }

    void setSeverity(final String sev) {
        severity = sev;
    }

    String getSeverity() {
        return severity;
    }

    boolean isSerious() {
        if(getSeverity() == null) {
```



```

        return false;
    } else if(getSeverity().equals("critical")) {
        return true;
    } else if(getSeverity().equals("important")) {
        return true;
    } else {
        return false;
    }
}

private Date findEarliest() {
    if((reported != null) && (pub == null)) {
        return reported;
    } else if((reported == null) && (pub != null)) {
        return pub;
    } else if((reported == null) && (pub == null)) {
        return released;
    } else if(reported.before(pub)) {
        return reported;
    } else {
        return pub;
    }
}

/**
 * Returns the number of days it took to resolve this issue.
 *
 * @return The number of days
 */
double getDaysToResolve() {
    Date start = findEarliest();
    Date finish = released;

    return findDifferenceInDays(start, finish);
}

/**
 * Finds the difference between two dates, in days.
 *
 * @param date1 The first date (can't be null)
 * @param date2 The second date (can't be null)
 * @return The difference between date1 and date2, in days
 */
static double findDifferenceInDays(final Date date1, final Date
date2) {
    if((date1 == null) || (date2 == null)) {
        throw new IllegalArgumentException("Args cannot be null.");
    }

    long millis = Math.abs(date2.getTime() - date1.getTime());

    double seconds = millis / 1000;
    double minutes = seconds / 60;
    double hours = minutes / 60;
    double days = hours / 24;

    return days;
}

```

```

    }

    public String toString() {
        StringBuffer buffer = new StringBuffer();

        buffer.append("Issue: ");
        buffer.append("CVE: " + cve);
        buffer.append(" / ");
        buffer.append("Release fixed: " + releaseFixed);
        buffer.append(" / ");
        buffer.append("released: " + released);
        buffer.append(" / ");
        buffer.append("reported: " + reported);
        buffer.append(" / ");
        buffer.append("public: " + pub);
        buffer.append(" / ");
        buffer.append("severity: " + severity);

        return buffer.toString();
    }
}
import java.util.ArrayList;
import java.util.Date;
import java.util.Iterator;
import java.util.List;

/**
 * Entry point to the issue analyzer package.
 */
public final class IssueAnalyzer {
    /**
     * The parser.
     */
    private IssueParser parser = null;

    /**
     * Constructor.
     */
    IssueAnalyzer() {
    }

    /**
     * Parses the issues file.
     *
     * @param path (Optional) The path to vulnerabilities-httpd.xml
     * @throws Exception If an error occurs
     */
    private void parse(final String path) throws Exception {
        parser = new IssueParser();

        if(path != null) {
            parser.parse(path);
        } else {
            parser.parse("vulnerabilities-httpd.xml");
        }
    }
}

```

```

        System.out.println("Done parsing, got " +
parser.getIssues().size() + " issues.");
    }

/**
 * Command-line entry point. One argument is optional: the path to
 * the vulnerabilities-httpd file.
 *
 * @param args Command-line arguments
 * @throws Exception If an error occurs
 */
public static void main(String[] args) throws Exception {
    IssueAnalyzer ia = new IssueAnalyzer();

    if((args != null) && (args.length > 0)) {
        ia.parse(args[0]);
    } else {
        ia.parse(null);
    }

    ia.calculateAverageResolutionTime();
    ia.countCveNames();

    ia.calculateAverageIssuesPerRelease();
    ia.calculateAverageReleaseTimes();
}

/**
 * Returns the security-driven releases.
 *
 * @return The security-driven releases
 */
private List getSecurityDrivenReleases() {
    List combined = parser.getReleases();

    List driven = new ArrayList();
    Iterator iter = combined.iterator();
    Release rel = null;

    while(iter.hasNext()) {
        rel = (Release) iter.next();
        if(rel.isSecurityDriven()) {
            driven.add(rel);
        }
    }

    System.out.println("Security-driven releases: " + driven);

    return driven;
}

/**
 * Calculates the average release times, for security-driven and
 * all releases.
 *
 * @throws Exception If an error occurs
 */

```

```

        private void calculateAverageReleaseTimes() throws Exception {
            List releases = parser.getReleases13();
            double avg = calculateAverageReleaseTime(releases);
            System.out.println("Average release time for 1.3 releases: " +
avg);

            releases = parser.getReleases20();
            avg = calculateAverageReleaseTime(releases);
            System.out.println("Average release time for 2.0 releases: " +
avg);

            List driven = getSecurityDrivenReleases();
            avg = calculateAverageReleaseTime(releases);
            System.out.println("Average release time for security-driven
releases: " + avg);
        }

        /**
         * Calculates the average release time for releases on the given
list.
         *
         * @param releases The releases
         * @return The average (in days)
         * @throws Exception If an error occurs
         */
        private double calculateAverageReleaseTime(final List releases)
throws Exception {
            if((releases == null) || (releases.size() < 1)) {
                return 0.0;
            }

            Iterator iter = releases.iterator();

            Release release = null;
            String version = null;
            String nextVersion = null;
            Date date = null;
            Date nextDate = null;
            double releaseTime = 0.0;
            List releaseTimes = new ArrayList();

            ReleaseDates releaseDates = ReleaseDates.getInstance(null);

            while(iter.hasNext()) {
                release = (Release) iter.next();
                version = release.getVersion();
                date = releaseDates.getDate(version);

                nextVersion = releaseDates.getNextVersion(version);
                if(nextVersion != null) {
                    nextDate = releaseDates.getDate(nextVersion);
                } else {
                    nextDate = null;
                }

                if((date != null) && (nextDate != null)) {

```

```

        releaseTime = Issue.findDifferenceInDays(date,
nextDate);
        if(releaseTime > 0) {
            releaseTimes.add(new Double(releaseTime));
        }
    }

    iter = releaseTimes.iterator();
    double sum = 0.0;
    while(iter.hasNext()) {
        sum += ((Double) iter.next()).doubleValue();
    }

    return sum / releaseTimes.size();
}

/**
 * Calculates the average number of issues per release, for 1.3,
2.0, and combined.
 *
 * @throws Exception If an error occurs
 */
private void calculateAverageIssuesPerRelease() throws Exception {
    List r13 = parser.getReleases13();
    double avg13 = calculateAverageIssuesPerRelease(r13);
    System.out.println("1.3 average issues per release: " + avg13);

    List r20 = parser.getReleases20();
    double avg20 = calculateAverageIssuesPerRelease(r20);
    System.out.println("2.0 average issues per release: " + avg20);

    List combined = parser.getReleases();
    double avgCombined =
calculateAverageIssuesPerRelease(combined);
    System.out.println("Combined average issues per release: " +
avgCombined);
}

/**
 * Calculates the average number of issues per release in the given
list.
 *
 * @param releases The releases
 * @return The average
 */
private double calculateAverageIssuesPerRelease(final List
releases) {
    if((releases == null) || (releases.size() < 1)) {
        return 0.0;
    }

    Iterator iter = releases.iterator();
    Release rel = null;
    double sum = 0.0;
    int n = 0;

```

```

        while(iter.hasNext()) {
            rel = (Release) iter.next();

            sum += rel.getIssueCount(false);
            n++;
        }

        return sum / n;
    }

    /**
     * Counts the issues that have CVE names.
     */
    private void countCveNames() {
        List issues = parser.getIssues();
        if((issues == null) || (issues.size() < 1)) {
            return;
        }

        Iterator iter = issues.iterator();
        Issue issue = null;
        String name = null;
        int names = 0;
        int n = 0;

        while(iter.hasNext()) {
            issue = (Issue) iter.next();
            name = issue.getCve();
            if((name != null) && (name.trim().length() > 0)) {
                names++;
            }
            n++;
            name = null;
        }

        System.out.println(names + " issues have CVE names (" + (100 *
names / n) + "%)");
    }

    /**
     * Calculates the average resolution time by issue severity
     * for all levels of severity.
     *
     * @throws Exception If an error occurs
     */
    private void calculateAverageResolutionTime() throws Exception {
        List issues = parser.getIssues();
        if((issues == null) || (issues.size() < 1)) {
            return;
        }

        calculateAverageResolutionTime(issues, null);
        calculateAverageResolutionTime(issues, "critical");
        calculateAverageResolutionTime(issues, "important");
        calculateAverageResolutionTime(issues, "moderate");
        calculateAverageResolutionTime(issues, "low");
    }

```

```

    /**
     * Calculates the average resolution time for the issues in the
given
     * list that match the given severity level.
     *
     * @param issues The list of issues
     * @param severity The severity level to match
     * @throws Exception If an error occurs
     */
    private void calculateAverageResolutionTime(final List issues,
final String severity) throws Exception {
        if((issues == null) || (issues.size() < 1)) {
            return;
        }

        Iterator iter = issues.iterator();
        Issue issue = null;
        int n = 0;
        double sum = 0;

        while(iter.hasNext()) {
            issue = (Issue) iter.next();

            if((severity != null) && (!
(severity.equals(issue.getSeverity())))) {
                // Skip
            } else {
                n++;
                sum += issue.getDaysToResolve();
            }
        }

        double average = sum / n;
        System.out.println("Average resolution time: " + average + ",
using severity filter: " + severity +
                        " and " + n + " issues passing this
filter.");
    }
}

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Iterator;
import java.util.List;
import javax.xml.parsers.SAXParserFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.parsers.SAXParser;

final class IssueParser extends DefaultHandler {
    /**
     * The issue currently being parsed.
     */
    private Issue issue;

```

```

/**
 * The release currently being parsed.
 */
private Release release;

/**
 * All the issues.
 */
private static List issues;

/**
 * The 1.3 releases.
 */
private static List releases13;

/**
 * The 2.0 releases.
 */
private static List releases20;

/**
 * All the releases.
 */
private static List releases;

/**
 * The date format: yyyyMMdd.
 */
private SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMdd");

/**
 * The release dates.
 */
private ReleaseDates releaseDates;

/**
 * Constructor.
 *
 * @throws Exception If an error occurs
 */
IssueParser() throws Exception {
    issue = null;
    release = null;
    issues = new ArrayList();
    releases13 = new ArrayList();
    releases20 = new ArrayList();
    releases = new ArrayList();
    sdf = new SimpleDateFormat("yyyyMMdd");
    releaseDates = ReleaseDates.getInstance(null);
}

/**
 * Parses the file.
 *
 * @param path (Optional) path to vulnerabilities-httpd.xml
 * @throws Exception If any error occurs

```



```

    */
    void parse(final String path) throws Exception {
        DefaultHandler handler = new IssueParser();
        SAXParserFactory factory = SAXParserFactory.newInstance();
        SAXParser saxParser = factory.newSAXParser();

        if(path != null) {
            saxParser.parse(path, handler);
        } else {
            saxParser.parse("vulnerabilities-httpd.xml", handler);
        }
    }

    /**
     * Returns a list of all the issues.
     *
     * @return All the issues
     */
    List getIssues() {
        return issues;
    }

    /**
     * Returns a list of all the releases.
     *
     * @return All the releases
     */
    List getReleases() {
        return releases;
    }

    /**
     * Returns all the 1.3 releases.
     * May return an empty list.
     *
     * @return The 1.3 releases
     */
    List getReleases13() {
        return releases13;
    }

    /**
     * Returns all the 2.0 releases.
     * May return an empty list.
     *
     * @return The 2.0 releases
     */
    List getReleases20() {
        return releases20;
    }

    /**
     * Adds the current release to the proper list.
     */
    private void addRelease() {
        if(release == null) {

```

```

        throw new IllegalArgumentException("Can't add null
release.");
    }

    releases.add(release);

    String version = release.getVersion();
    if((version == null) || (version.trim().length() < 1)) {
        throw new IllegalArgumentException("Can't add release
without a version.");
    } else if(version.startsWith("1.3")) {
        releases13.add(release);
    } else if(version.startsWith("2.0")) {
        releases20.add(release);
    } else {
        throw new IllegalArgumentException("Unknown branch: " +
version);
    }
}

/**
 * Returns the XML element name. This is a convenience method
 * added to handle both namespace-aware and namespace-unaware
 * SAX parser implementations. Preference is given to the local
 * name over the qualified one.
 *
 * @param sName The local name
 * @param qName The qualified name
 * @return The name
 */
private String getName(final String sName, final String qName) {
    String name = sName;
    if((name == null) || (name.trim().length() < 1)) {
        name = qName;
    }

    return name;
}

/**
 * Parses the given string into a Date object.
 *
 * @param yyyyymmdd The date string
 * @return The Date object, or null if an error occurred
 */
private Date getDate(final String yyyyymmdd) {
    try {
        return sdf.parse(yyyyymmdd);
    } catch (Exception e) {
        e.printStackTrace();
        return null;
    }
}

/**
 * Translates the integer severity level (from 1-4)
 * into the string one, from critical to low.

```

```

*
* @param level 1-4
* @return The string level
*/
private String getSeverityLevel(final String level) {
    if((level == null) || (level.trim().length() < 1)) {
        throw new IllegalArgumentException("Arg can't be null or
empty.");
    } else if(level.equals("1")) {
        return "critical";
    } else if(level.equals("2")) {
        return "important";
    } else if(level.equals("3")) {
        return "moderate";
    } else if(level.equals("4")) {
        return "low";
    } else {
        throw new IllegalArgumentException("Unknown severity level:
" + level);
    }
}

/**
 * Called by the SAX parser implementation for each element.
 */
public void startElement(String namespaceURI, String sName, String
qName, Attributes attrs)
    throws SAXException {
    String name = getName(sName, qName);

    if(name.equals("issue")) {
        issue = new Issue();

        for(int i = 0; i < attrs.getLength(); i++) {
            String aName = getName(attrs.getLocalName(i),
attrs.getQName(i));
            if(aName.equals("reported")) {
                issue.setReported(getDate(attrs.getValue(i)));
            } else if(aName.equals("released")) {
                issue.setReleased(getDate(attrs.getValue(i)));
            } else if(aName.equals("public")) {
                issue.setPub(getDate(attrs.getValue(i)));
            } else if(aName.equals("fixed")) {
                String version = attrs.getValue(i);

                issue.setReleaseFixed(version);

                if(release == null) {
                    release = new Release();
                    release.setVersion(version);
                } else if(release.getVersion().equals(version)) {
                    // Do nothing
                } else {
                    addRelease();

                    release = new Release();
                    release.setVersion(version);
                }
            }
        }
    }
}

```

```

        }

        if(release.getDate() == null) {
            System.out.println("Setting date for " +
release.getVersion() +
                                " to " +
releaseDates.getDate(release.getVersion()));
release.setDate(releaseDates.getDate(release.getVersion()));
        }
    }
    } else if(name.equals("severity")) {
        for(int i = 0; i < attrs.getLength(); i++) {
            String aName = getName(attrs.getLocalName(i),
attrs.getQName(i));
            if(aName.equals("level")) {

issue.setSeverity(getSeverityLevel(attrs.getValue(i)));
            }
        }
    } else if(name.equals("cve")) {
        for(int i = 0; i < attrs.getLength(); i++) {
            String aName = getName(attrs.getLocalName(i),
attrs.getQName(i));
            if(aName.equals("name")) {
                issue.setCve(attrs.getValue(i));
            }
        }
    }
}

/**
 * Called by the SAXParser implementation at the end of each
element.
 */
public void endElement(String namespaceURI, String sName, String
qName) throws SAXException {
    String name = getName(sName, qName);

    if(name.equals("issue")) {
        issues.add(issue);
        release.addIssue(issue);
    }
}

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

final class Release {
    private String version;
    private List issues;
    private int seriousIssues;
    private Date date;

    Release() {

```

```

        issues = new ArrayList();
        seriousIssues = 0;
    }

    void setVersion(final String v) {
        version = v;
    }

    String getVersion() {
        return version;
    }

    void setDate(final Date d) {
        date = d;
    }

    Date getDate() {
        return date;
    }

    void addIssue(final Issue issue) {
        issues.add(issue);

        if(issue.isSerious()) {
            seriousIssues++;
        }
    }

    int getIssueCount(boolean onlySerious) {
        if(onlySerious) {
            return seriousIssues;
        } else {
            return issues.size();
        }
    }

    boolean isSecurityDriven() {
        return getIssueCount(true) > 0;
    }

    public String toString() {
        StringBuffer buffer = new StringBuffer();

        buffer.append("Release " + version + " has " +
getIssueCount(false) + " issues, " +
        getIssueCount(true) + " of which are serious.");

        return buffer.toString();
    }
}

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.HashMap;

```

```

import java.util.Map;
import java.util.StringTokenizer;

/**
 * Holds information about release dates.
 *
 * *** Note that this class relies on the ordering of the source file:
 * *** It assumes all the 2.0 releases are first, in descending order,
 * *** followed by all the 1.3 releases, also in descending order.
 */
final class ReleaseDates {
    /**
     * The instance of this singleton.
     */
    private static ReleaseDates instance = null;

    /**
     * The release dates for the 2.0 branch, keyed by version.
     */
    private Map release20Dates;

    /**
     * The release dates for the 1.3 branch, keyed by version.
     */
    private Map release13Dates;

    /**
     * Map of release name to next release name.
     * The value for 2.0.55 is null, for 2.0.54 is 2.0.55, etc.
     */
    private Map nextReleases;

    /**
     * The date formatter.
     */
    private SimpleDateFormat sdf;

    /**
     * Constructor.
     *
     * @param path (Optional) path to releaseDates.txt file
     */
    private ReleaseDates(final String path) throws Exception {
        release20Dates = new HashMap();
        release13Dates = new HashMap();
        nextReleases = new HashMap();
        sdf = new SimpleDateFormat("yyyyMMdd");

        if(path != null) {
            readFile(path);
        } else {
            readFile("releaseDates.txt");
        }
    }

    /**
     * Returns the instance of this singleton.

```

```

    *
    * @param path (Optional) path to releaseDates.txt file
    * @return ReleaseDates
    * @throws Exception If an error occurs
    */
    static ReleaseDates getInstance(final String path) throws Exception
    {
        if(instance == null) {
            instance = new ReleaseDates(path);
        }

        return instance;
    }

    /**
     * Reads the file, populates the release dates map.
     *
     * @param path The file path
     */
    private void readFile(final String path) throws Exception {
        BufferedReader reader = new BufferedReader(new FileReader(new
        File(path)));

        String line = null;
        String nextVersion = null;
        String version = null;
        String yyyyymmdd = null;
        Date date = null;
        StringTokenizer st = null;

        while((line = reader.readLine()) != null) {
            st = new StringTokenizer(line);
            while(st.hasMoreTokens()) {
                version = st.nextToken().trim();
                yyyyymmdd = st.nextToken().trim();
            }

            date = sdf.parse(yyyyymmdd);

            if(version.startsWith("1.3")) {
                release13Dates.put(version, date);
            } else {
                release20Dates.put(version, date);
            }

            nextReleases.put(version, nextVersion);
            nextVersion = version;
        }

        reader.close();
    }

    /**
     * Returns the next version name for the given version name.
     * May return null.
     *
     * @param version The version, e.g. 2.0.54

```

```

    * @return The next version name, e.g. 2.0.55, or null
    */
String getNextVersion(final String version) {
    return (String) nextReleases.get(version);
}

/**
 * Returns the release date for the given release.
 *
 * @param version The version
 * @return The date (if found, null otherwise)
 */
Date getDate(final String version) {
    if((version == null) || (version.trim().length() < 1)) {
        return null;
    } else if(version.startsWith("1.3")) {
        return (Date) release13Dates.get(version);
    } else {
        return (Date) release20Dates.get(version);
    }
}
}

```



