Software Engineering Institute | Carnegie Mellon University

Application Firewalls and Proxies – Introduction and Concept of Operations

Howard F. Lipson

Ken van Wyk

December 2005

ABSTRACT: Practices that could significantly improve application security by integrating knowledge about an application's specific security needs into elements of the IT security infrastructure are often overlooked. This document describes one of the many potential topic areas involving the integration of business applications into a supporting IT security infrastructure. Application firewalls attempt to use application-specific knowledge to improve the perimeter defense that the security infrastructure provides.

INTRODUCTION

Providing a safe operating environment for a business-critical application is among the final—and arguably most crucial—steps in the assembly and integration process. Creating and maintaining a secure IT environment for a mission-critical business application is essential not only for the security of the application but also for the viability of the organization that the application supports. This step represents the traditional intersection between the software development and IT teams. And, as one might expect, the critically timed team handoff from application developers to the IT staff is filled with opportunities for making catastrophic errors. Among other things, practices that can significantly improve application security are often overlooked due to myriad reasons, including the respective teams' lack of understanding of the needs and capabilities of their counterparts. For example, practices that could significantly improve application security by integrating knowledge about the application's specific security needs into elements of the IT security infrastructure are seldom applied or even considered.

Application developers often mistakenly feel they have little to contribute to improving the security of the IT environment, which is assumed to be the sole responsibility of the IT staff. Conversely, IT security staff are typically overburdened with protecting against generic security threats (such as viruses and network-protocol-based attacks) and may feel that managing application-specific security risks is the sole responsibility of the application developers. However,

Software Engineering Institute
Carnegie Mellon University
4500 Fifth Avenue
Pittsburgh, PA 15213-2612

Phone: 412-268-5800
Toll-free: 1-888-201-4479

www.sei.cmu.edu

increased communication between application developers and IT security staff as often and as early in the software development life cycle as possible can provide for security elements to be built into both the application and the infrastructure, which would ultimately make the security infrastructure even more secure for the application. These security elements might be, for example, identifying critical application-specific information assets in need of increased monitoring by the infrastructure, or providing the higher order application-specific knowledge needed to dramatically improve traffic analysis and identify more classes of troublesome traffic before they reach the application. This shared responsibility provides defense in depth.

One of the essential objectives of the IT security infrastructure of an organization is to provide an effective perimeter defense for the business applications that the IT infrastructure supports. Of course, for modern, open, highly-distributed applications operating over an unbounded systems environment such as the Internet, a sound perimeter defense is a necessary but not sufficient condition for achieving high levels of security. There are typically multiple perimeters across multiple administrative domains, representing varying levels of trust. It is rare for any of these perimeters to be totally closed during the conduct of an organization's activities. Holes in the perimeter defense must be kept open to allow communication and collaboration with business partners, suppliers, and customers. For example, if port 80 (the standard "listening port" for HTTP servers) were kept closed by an organization's firewall, no web services could be provided.

Historically, the primary element of an organization's perimeter defense has been the firewall (more precisely the network layer firewall), separating the networks within an organization's administrative control from those in the "outside" world. Firewalls are also commonly used to separate various subnetworks within an organization's administrative control from each other. Standing between incoming and outgoing network traffic, the firewall enforces a predefined set of policies, or "rules," that determine what network traffic may and may not pass. The policy decisions are based on each packet's header information (e.g., source and destination addresses, network ports, protocol type, and state flags). "Stateless" firewalls make their decisions for each packet based solely on the information contained in that individual packet. "Stateful" firewalls accumulate information about the packets they see and use that information in an attempt to match incoming and outgoing packets to determine which packets may be part of the same network communications session. By evaluating a packet in the larger context of a network communications session, a stateful firewall has much more complete information than a stateless firewall and can therefore more readily recognize and reject packets that may be part of a network-protocol-based attack.

While traditional firewalls can do a good job of blocking many types of attacks at the network protocol level, they are not effective at protecting against attacks that exploit application-specific vulnerabilities. That is because traditional firewalls only examine a packet's header (i.e., delivery information) and not the data moving between the application and its user(s). Hence, a traditional firewall cannot protect against malformed application content that could, for example, probe for and exploit a buffer overflow in a given application.

Whether stateful or stateless, a network firewall can only make decisions based on traffic analyses at the network level. These firewalls and their rule sets contain no higher level information about the applications that will make use of the packet payloads, and so these firewalls would not recognize or reject packet payloads comprising malformed input that would exploit a vulnerability in a particular application.

This shortcoming of traditional network-level firewalls has been noted by the security product community, as one might expect. One of the first generation of products that have surfaced in response to this shortcoming is the application firewall.

APPLICATION FIREWALLS

In the past several years, so-called "application firewall" products have emerged in the commercial marketplace. The intention of these products is to shore up application-level security, primarily by providing content filtering—in both inputs and outputs—between the application's users and its servers. Almost (if not) all such products available in the commercial or open source space today are designed to work with web-based applications.

The way that web application firewalls function is by interposing themselves between the application server and the user-side client, thereby "intercepting" all user data entering or leaving the application server. The intercepted traffic is examined against various rules in order to attempt to judge whether the data is "good" or "bad." For example, data entered by a user can be examined to see if it contains potentially dangerous characters, such as SQL commands that could lead to an SQL insertion attack. When attacks are detected, the application firewall can take whatever remedial or evasive action its owner deems appropriate, ranging from simply disconnecting the current application session to more sophisticated approaches such as shunting the session to a "honeypot" system that is specially instrumented to gather details of attack methodologies. Running the

application firewall in a separate process and memory space from the application further protects the applications, business logic, and data.

Naturally, in order for an application firewall to be able to make reasonable assessments of the application data, it would need to have some advance knowledge. Some products accomplish this by way of a database of known attacks—either via signatures of actual attack data, behavioral patterns, or some hybrid of both. Other products observe the normal behavior of the application(s) that they are to protect, preferably in a controlled environment, and then attempt to ensure that all "live" data conforms to similar content, size, and structure.

Benefits and Drawbacks

As one might reasonably imagine, there are both benefits and drawbacks to using application firewall technology.

Benefits

Another layer of security

An application firewall provides an additional tier of protection between a web application and its end users. As such, it is conceivable that some novel attacks may be thwarted before they can reach the application server. Moreover, output data can be analyzed and sensitive data intercepted before it is output to the attacker. Isolating security checks in a separate process and memory space also allows the system to isolate failures.

Specialized security knowledge

Most web applications are written by software developers with varying degrees of security-specific knowledge. On the other hand, application firewalls are developed specifically as security devices. It seems a reasonable assumption that more specialized application-specific security knowledge goes into the design of application firewalls than goes into most web applications.

Specialized application knowledge

Typically the elements of an organization's IT infrastructure provide generic security services that are independent of the application services being protected. An application firewall's reason for being is to provide high-quality application-specific security services. As an example, an XML application firewall allows the application to offload activities such as XML document parsing, inspection, and authentication onto the application firewall. If a vendor's market focus on XML application firewalls allows the vendor to bring together a development team with a greater level and concentration of XML and security expertise than the typical web services application development team, then the services provid-

ed by the XML firewall would likely be of higher quality than those that would have been built into the application itself. Moreover, offloading activities such as XML parsing onto separate hardware may improve some aspects of performance, especially if the vendor design team's expertise in XML allows them to produce highly efficient code.

Flexible policy enforcement

Application firewalls allow you to enforce policy on acceptable application behavior from a central location. Changing rule sets for the application firewall is much easier than updating your policy by making changes to the application. Of course, this degree of flexible policy enforcement presupposes that the organization is able to clearly and precisely define its policy to the degree of specificity required for each application. This assumption is often far from what is found in practice in many production data centers. Further, it is a topic that requires a high level of cooperation and coordination between the application developers and the IT security staff. In the case of third party and legacy applications, additional difficulties in defining behavioral policies are also likely to surface. That said, because an application firewall typically sits between the end user and the server environment—no matter how complex—it truly provides the IT security staff with a central policy enforcement "choke point" of sorts.

Detection of some attacks

Since application firewalls are designed specifically to prevent application-level attacks from gaining unauthorized access to the applications they protect, they are also well positioned to detect suspected attacks as they occur and alert operations staff. Thus they offer a degree of utility as application-specific intrusion detection sensors. However, the level of usefulness here depends almost entirely on how thoroughly the application firewall and its operators are able to define the application's normative behavior. This creates yet another situation that requires a great deal of cooperation between the application's developers and the IT security staff.

Logging of application data

An application firewall is in an ideal position to provide event logging of data to and from the application it is protecting. As such, an application firewall can be beneficial at integrating a particular web application's logging into an enterprise-level security monitoring and auditing capability. Application firewalls understand application protocols and data formats better than network devices and can provide audit log services by logging header and body data. For example, it may be desirable to have the application firewall log traffic data that the intrusion

detection system cannot read due to SSL protection on the network communications.

Augmenting third-party and legacy applications

It is often necessary to deploy third-party applications and/or to support legacy applications in production environments. In these cases, application firewalls can effectively augment the operational security capabilities of the applications that they protect. Event logging, as described above, is an excellent example. Or in instances where a third-party vendor does not perform validation, application firewalls can be used to provide this service.

Drawbacks

Configuring

Many of the application firewall products available today require the product to "learn" the application's normal behavior. In order for that behavior to, in fact, be "normal," it is vital that the learning process be done carefully in a controlled test bed environment that adequately emulates the production environment. Failing to do so can result in non-normal (e.g., attack) sessions being deemed "normal." Moreover, the process of "learning" normal behavior (in either emulated or production environments) has been demonstrated by the scientific record of the performance of intrusion detection systems to be beset with theoretical and practical problems. Even if this is done perfectly, however, configuring an application firewall to precisely know the application it protects is problematic at best, due to the sheer volume of information that needs to be defined. For example, every user input field in every single page of the application needs to be properly described to the application firewall in terms such as maximum field size, allowable data types/values, unallowable data types, etc. This drawback is exacerbated if the application firewall is "default deny." (See the "Default deny" bullet item below.) Finally, configuring the application firewall is not a one-time activity but an ongoing maintenance task because of the necessity of keeping the application firewall in sync with the application. Whenever updates to an application change the specification of what is normal user and application behavior, those changes must be simultaneously reflected in the application firewall's rule set.

Single point of failure

Depending on the design and architecture of the overall application system and IT infrastructure, an application firewall could very well be a single point of failure for the services that the application provides.

Performance

Since an application firewall sits in series with the application that it protects, there must always be some performance cost imposed on the application. Even with optimizing data caching and such, connection latencies may come into play to slow the application down somewhat. However, a partially offsetting performance gain may be produced when an application firewall allows the application to offload some significant amount of processing. For example, as mentioned earlier, XML application firewalls allow the application to offload activities such as XML document parsing, inspection, and authentication onto the separate hardware of the XML firewall.

Complexity

Maintaining a web application plus multiple devices, such as a web application firewall and a separate XML firewall, increases the complexity of a system, and the opportunity for misconfiguration, conflicts, and other problems that could limit functionality and availability, or weaken security. However, there have been vendor moves to combine multiple devices into a single appliance.

Passing the buck

It can be easily argued that deploying an application firewall effectively "passes the buck" of application security to another device. Further, since application firewalls require a good amount of time and effort to properly learn the normal behavior of an application, the argument could be made that the application developers should be focusing their efforts on better software development techniques to improve quality and security instead of relying on an application firewall to do the work of the developers. This is a compelling argument, to be sure, and it bears careful consideration. For example, allowing programmers to omit input validation from an application and instead leaving the responsibility for this critical security task to the application firewall would be extremely bad programming practice. On the other hand, configuring the application firewall to perform some input validation for the application in addition to the application's own input validation may provide some redundant and diverse defense in depth.

Blacklisting

Some, but not all, application firewall products function by identifying "known bad" sorts of behavior, as opposed to only allowing "known good" behavior. As such, they inherently implement a blacklist methodology (also known as a "default allow" policy)—one that inevitably fails every time a new attack is discovered.

Default deny

One the other hand, an application firewall that supports a "default deny" policy allows only "known good" behavior as defined in its rule set. This provides a much higher level of security but also presents an extremely difficult challenge for those configuring and maintaining the application firewall rule set. Unless all acceptable behavior is precisely and completely specified (both initially and as the application evolves), legitimate user input and application behavior that is not fully specified in the application firewall's rule set will be blocked, causing to at least some degree a self-imposed denial of service. Ironically, without a substantial configuration and maintenance effort, a "default deny" application firewall may deny the functionality and services you are trying to protect.

Incompatibility

Inconsistencies between the way input data is parsed (or interpreted) by an application firewall and the way an application parses input data could be exploited by an attacker who sends malformed input. For example, application firewalls rule sets and policies could be bypassed by a variety of attacks where the input appears legitimate to the application firewall but is problematic to the application the firewall was meant to protect. If the application firewall is bypassed, the attacker may be able gain unauthorized access to the application, deny service, or create some other security breach.

DECIDING WHETHER TO DEPLOY

The decision of whether to deploy an application firewall should not be made lightly. While there are certainly good reasons to do so, there are certainly bad reasons as well. Application firewalls are no substitute for good, sound programming practices, for example. Relying on an application firewall to protect bad software is doomed to the eventual catastrophic failure of the application. On the other hand, application firewalls do represent a single point for adding security features to existing, legacy, and third-party applications that can be highly beneficial to monitoring and incident handling operations. Moreover, attack signatures for common protocol vulnerabilities relevant to the application domain (e.g., XML protocol vulnerabilities for web services applications) can be maintained and updated by an external security organization (e.g., the application firewall vendor), reducing maintenance effort for the organization deploying the application firewall.

It should be noted that while application firewalls can provide a variety of valuable security services to the application system, from a design point of view each

capability may be mutually exclusive, precluding other capabilities and/or inhibiting other qualities such as performance. For example, if data caching strategies are used to deal with performance issues, then complexity and availability may be adversely affected. Or the application that is protected by the application firewall may be vulnerable to attacks that can evade the application firewall, such as insider attacks. So a holistic viewpoint should be used to assess tradeoffs.

REFERENCES

[Allen 01]

Allen, Julia. The CERT Guide to System and Network Security Practices. Boston, MA: Addison-Wesley, 2001.

[ArmorLogic 08]

ArmorLogic, ApS. Profense Professional, 2008.

[Barracuda 08]

Barracuda Networks Inc. Barracuda Web Site Firewall, 2008.

[Checkpoint 08]

Checkpoint, Inc. VPN-1 Power, 2008.

[Cheswick 03]

Cheswick, William; Bellovin, Steven M.; & Rubin, Aviel D. *Firewalls and Internet Security: Repelling the Wily Hacker, 2nd Edition.* Boston, MA: Addison-Wesley, 2003.

[Citrix 08]

Citrix Systems, Inc. Citrix Application Firewall, 2008.

[Comer 00]

Comer, Douglas E. Internetworking with TCP/IP, Vol. 1: Principles, Protocols, and Architecture, 3rd Edition. New York: Prentice-Hall, 2000.

[eEye 08]

eEye Digital Security. SecureIIS Web Server Security, 2008.

[f5 08]

F5 Networks Inc. BIG-IP Application Security Manager, 2008.

[Imperva 08]

Imperva, Inc. SecureSphere Web Application Firewall, 2008.

[May 04]

May, Chris; Baker, Marie; Gabbard, Derek; Good, Travis; Grimes, Galen;
Holmgren, Mark; Nolan, Richard; Nowak, Robert; & Pennline, Sean. *Advanced Information Assurance Handbook* (CMU/SEI-2004-HB-001). Pittsburgh, PA:
CERT®/CC Training and Education Center, Software Engineering Institute, Carnegie
Mellon University, March 2004, pp. 109-114.

[Microsoft 08]

Microsoft Corporation. Forefront Unified Access Gateway, 2008.

[ModSecurity 08]

ModSecurity. Open Source Web Application Firewall, 2008.

[Protegrity 08]

Protegrity, Inc. Web Application Firewall, 2008.

[Stevens 94]

Stevens, W. Richard. *TCP/IP Illustrated, Volume. 1: The Protocols*. Boston, MA:
Addison-Wesley, 1994.

[Stevens 96]

Stevens, W. Richard. TCP/IP Illustrated, Volume 3: TCP for Transactions, HTTP,
NNTP, and the UNIX® Domain Protocols. Boston, MA: Addison-Wesley, 1996.

APPENDIX – APPLICATION PROXIES

The term *application firewall* has come into vogue rather recently. However, an
application firewall is just a special case of the more general concept of an *application proxy*, which manages the traffic between an application server and its
clients. In contrast to a network layer packet filter or firewall, an application
proxy typically contains lots of higher level information about the application it
is protecting, allowing the proxy to make good application-specific decisions
about incoming and outgoing traffic. This appendix contains additional information about application proxies, as well as an example of an open-source appli-

cation proxy for Internet web browsing called *Squid*. This material in this appendix is an excerpt (with some minor edits) from the *Advanced Information Assurance Handbook* [May 04] produced by the CERT® Education and Training Center.

Proxy Filtering

Filtering is based on specific application data

Done for many common protocols and services (Telnet, FTP, SMTP, HTTP, etc.)

Filtering decisions based on a variety of triggers
- Source/destination addresses
- Contents of the connection (content filtering)

© 2003 Carnegie Mellon University   Module 2: Firewalls and Network Access Controls–Slide 23

Proxy Filtering

An application proxy is an application program that runs on a firewall system between two networks. The host on which the proxy runs does not need to be acting as a router. When a client program establishes a connection through a proxy to a destination service, it first establishes a connection directly to the proxy server program. The client then negotiates with the proxy server to have the proxy establish a connection on behalf of the client between the proxy and the destination service. If this is successful, there are then two connections in place: one between the client and the proxy server and another between the proxy server and the destination service. Once established, the proxy then receives and forwards traffic bidirectionally between the client and service. The proxy makes all connection-establishment and packet-forwarding decisions. Any routing functions that may be active on the host system are irrelevant to the proxy.

As with packet filtering, application proxies are available on both special purpose proxy machines and general purpose computers. Generally speaking, appli-

cation proxies are slower than packet filtering routers, as there is a great deal of processing and storage overhead associated with creating, maintaining, and managing the two connections for every connection made through the proxy. However, in some ways application proxies are inherently more secure than packet filtering routers. They depend on a lot of higher layer intelligence and expend a great deal of resources in offering that security.

The application proxy is configured for a specific service: inspecting all the traffic that it handles as it passes it from one network to another. There are a number of services for which proxying is common—and Internet web browsing traffic is the most common of these. We'll use it in our examples, but the rest of the proxy filters perform very similar functions for their specific services.

SquidGuard

Squid is the most popular and most common open-source web proxy. Squid is designed to run on Linux/UNIX systems. As with all proxies, it is set up to receive http requests from the clients for which it is proxying. After consulting its rules to ensure that (a) the client is making a valid request and (b) the request itself does not violate any of the rules we will discuss later, the proxy makes a request for the web page being requested by the client. It receives the content on the client's behalf, inspects the content (again to make sure that the content of

the response is valid and does not violate any rules), and then sends it on to the client. For performance reasons, Squid can be configured to cache web pages it gets on behalf of its client. Subsequently, if there are additional requests for the same URL, Squid can simply send the cached content. This can cut down on web requests and bandwidth consumption.

We're able to "plug in" technologies for our Squid proxy, which will allow us to do more thorough content filtering on our http traffic. SquidGuard and Dans-Guardian are the two technologies (open source, of course) on which we will focus.

SquidGuard works with Squid to block access to sites by domain, IP address, or even keywords. It is very flexible, allowing you to block and allow access according to the time of day and to define groups within your organization that have different access privileges.

This all works because Squid allows you to define a redirector program that gets to examine each requested web page before Squid goes and gets it. If the redirector determines that a request violates an ACL, Squid will serve up the redirected page.

DansGuardian is a web content filtering proxy for Linux, FreeBSD, OpenBSD, NetBSD, Mac OS X, and Solaris that uses Squid to do all the fetching. It filters using multiple methods. These methods include content phrase filtering, POST limiting filtering, URL and domain filtering, Platform for Internet Content Sharing (PICS) filtering, MIME filtering, and file extension filtering. The content phrase filtering will check for pages that contain profanities and phrases often associated with pornography and other undesirable content. The POST limiting filtering allows you to block or limit web downloads or uploads. The URL and domain filtering is able to handle huge lists and is significantly faster than SquidGuard. The filtering has configurable domain, user, and source IP exception lists. SSL tunneling is supported.

The configurable logging feature produces a log in an easy-to-read format that provides the option of only logging the text-based pages, significantly reducing redundant information such as every image on a page. All parts of DansGuardian are configurable, giving total control over what is filtered to the end user administrator and not some third-party company.

Configuring Squid to Do Content Filtering

We'll use Webmin to manage the access control rules for our web content filter, which will be Squid in this case. Here's an overview of making one access control rule in Squid.

We need to get the Squid proxy server running, along with Webmin. We will assume that Squid is running on this host.

1. Open a Webmin management window and browse to the Servers page.
2. Click on the icon for Squid Proxy Server (see Figure 1).

Figure 1. Squid Proxy Server Icons

3. We'll start by changing the listening port for Squid from the default, TCP Port 3128, to another standard proxy port, TCP Port 8080. To do this, we click on the Ports and Networking icon from the Squid Proxy Server page (see Figure 1).
4. Next, check the radio button for "Listed below" and type 8080 in the dialog box under Port. This causes Squid to listen on port 8080. Click the Save button to save the changes. Clicking Save will bring back the Squid main page.
5. Next we will create an access control list to block some bad web sites. To do this, click on the Access Control icon (see Figure 1).
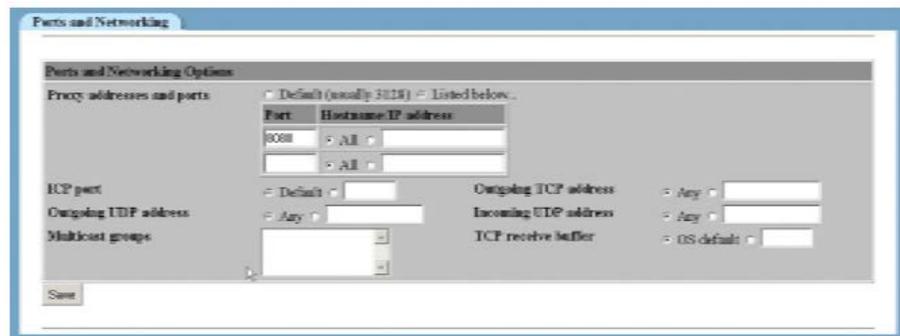
Figure 2. Ports and Networking in Squid Proxy Server

6. From the drop-down box on the lower left of the Access Control page, select "Web Server Address." This will allow a rule to be created that will block a specific web server by its address. Click the Create New ACL button to edit the ACL rules:
7. On the Edit ACL page, enter a name for the ACL rule (BadWebSite in this example). Do not use spaces! Enter the IP address of the web site you want to block (128.2.243.156 in this example) and the Netmask (/32 in this case—to block only that specific IP address). Leave the Failure URL blank

to get the Squid default denial when the ACL is met and the site is blocked. Click Save to save the rule and return to the Access Control main page.
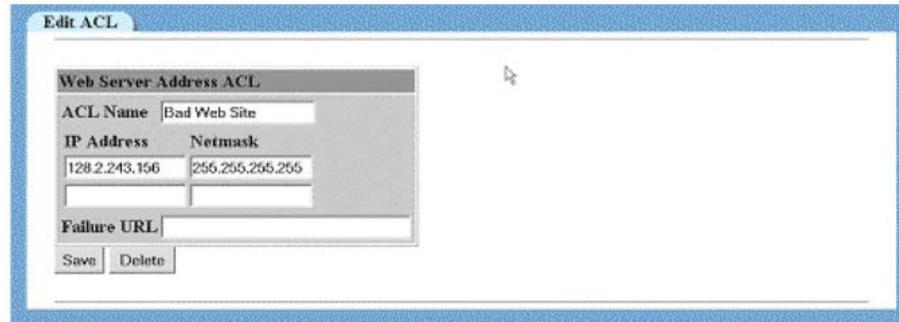
Figure 3. Edit ACL in Squid Proxy Server

8. Now that the ACL is built, we need to apply it. To do that, we'll add a proxy restriction. Click the "Add Proxy Restriction" link under the Proxy Restriction heading on the right of the Access Control main page.
9. Highlight the name of the ACL you created, select the Deny radio button, and click Save to save the proxy restriction and return to the Access Control main page.
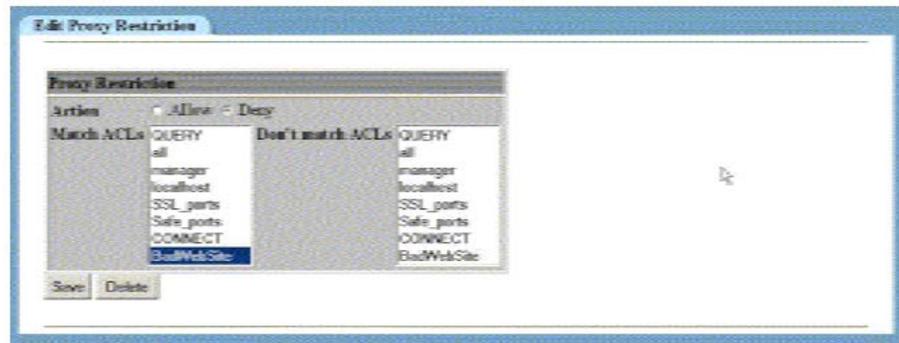
Figure 4. Edit Proxy Restriction in Squid Proxy Server

Squid's default behavior upon installation is to block all outbound traffic, as evidenced by the "Deny All" proxy restriction. You will need to edit this by clicking the Deny link under the Proxy Restriction heading and changing the Allow radio button, ensuring that "all" is highlighted under the Match ACLs text box. Click Save to change this rule.
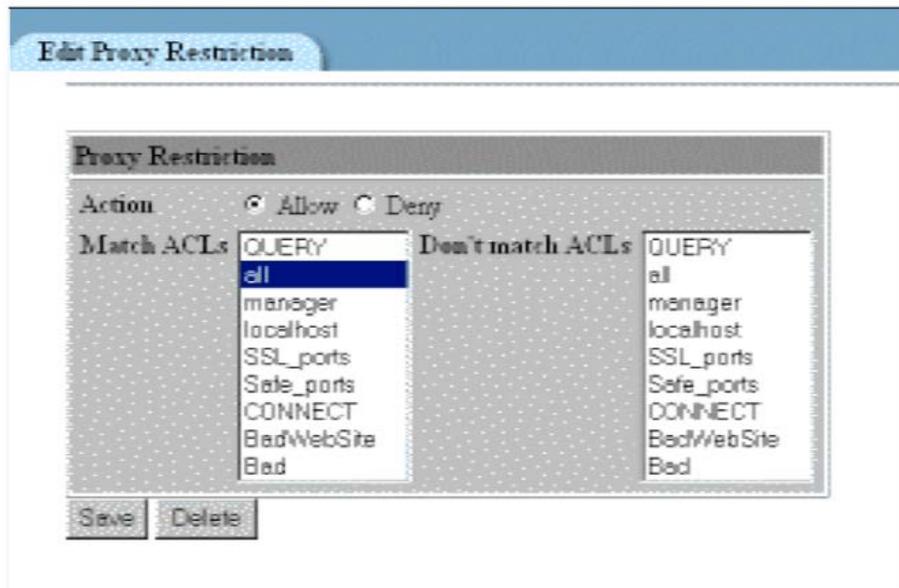
Figure 5. Changing Defaults to Allow Outbound Traffic in Squid Proxy Server

10. You'll also need to ensure that the proxy restrictions are in the right order. Like all packet filters, Squid compares requests to its proxy restriction list in order and makes a filtering decision based on the first match it encounters. Therefore, if your "Allow All" rule is above your filtering rule, all requests will be granted because the filtering rules will never be reached. To move the Allow All rule down to the bottom of the list, click the down arrow to the far right under Move to move the rule to the bottom of the list. This will ensure that all deny rules are tested before the Allow All rule is encountered.
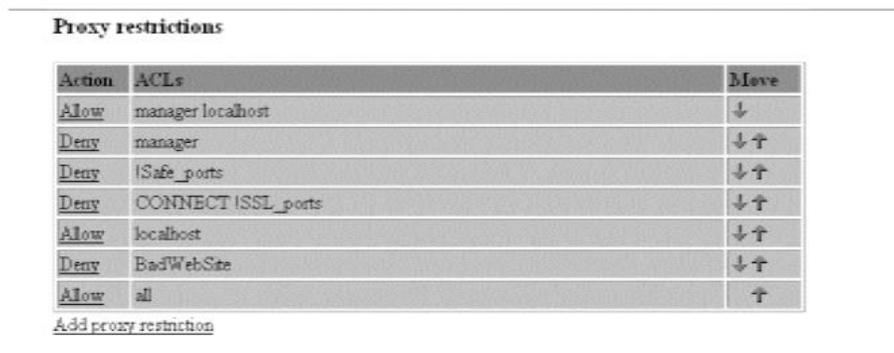
Figure 6. Ordering Proxy Restrictions in Squid Proxy Server

11. Now we should test the proxy rule. We must point a web browser at the Squid system, to port 8080, and try to browse to the IP address we blocked with our rule. If Squid is configured correctly, we should see the Squid-

generated web page shown in Figure 7.
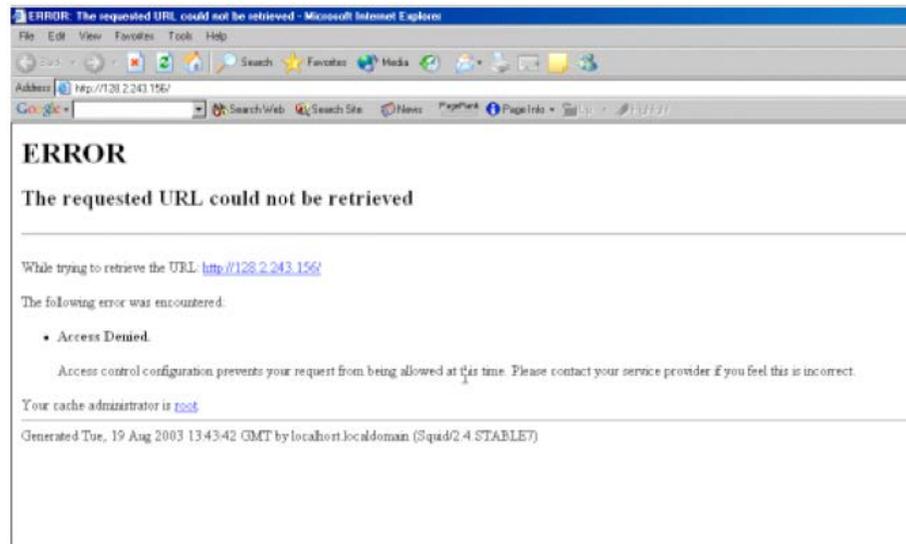
Figure 7. What a Browser Displays for a Blocked IP Address When Squid Proxy Server Is Configured Correctly