



## **NetCache® Technical Advisory:**

**Controlling Peer-to-Peer Traffic**

**Tobin Sears, Network Appliance, May 2006, TR-3333**

### **Executive Summary**

The increasing popularity of peer-to-peer (P2P) networking poses a significant risk to corporate environments. P2P services, such as Kazaa, eDonkey, Morpheus, and Skype allow users to share various forms of media in an unsecure, virtually unrestricted manner. Because firewalls and caching devices alone cannot completely block P2P access, a tiered approach is recommended. In order to effectively control P2P access within an organization, access controls must be implemented at the network, transport, and application layers.

## Table of Contents

1. Introduction .....	3
1.1 P2P Considerations for the Enterprise.....	3
2. A Tiered Approach to Control P2P Traffic .....	4
3. Recommended Process for Controlling P2P .....	5
3.1 Monitor and Analyze Network Traffic .....	5
3.2 Block Supernode and Index Server IP Addresses.....	7
3.3 Block P2P and Spyware-Related Domains, Request Headers, and IP Addresses.....	7
3.4 Implement a Whitelist/Blacklist Access Control Policy.....	10
4. Conclusion .....	11
5. Revision History .....	11

## 1. Introduction

Since its emergence in early 1999, there has been a steady increase in the use of peer-to-peer–based file-sharing programs. The inherent risks associated with P2P usage, however, have necessitated that it be controlled within the business environment. Today’s P2P clients are more intelligent and exponentially more difficult to control than their predecessors. Early P2P clients, such as Napster, relied on a central set of index servers that maintained content and user listings. Subsequent P2P implementations have migrated to a more decentralized architecture, increasing redundancy, performance, and anonymity, but making them nearly impossible to block. Depending on the specific implementation, clients can employ various methods of communicating with peer networks and are often dynamic in nature.

Kazaa, for example, relies on a semistatic group of servers that perform user-based authentication and peer localization. Access to these IP addresses can be blocked at both the network and application layers, but they can change at a moment’s notice. Additionally, peer-to-peer communication and data transfers can use either HTTP, TCP, or UDP over a dynamic port range, making it nearly impossible to block unless a tiered layer 3 to 7 access policy is implemented.

Similarly, Morpheus and other Gnutella-based clients participate in a truly decentralized network architecture, relying on peers for media indexing and search capabilities. Gnutella-based clients receive information about peer or “servant” machines from a group of predetermined servers. When the newly subscribed client receives a servant list, it gains the ability to communicate in a true peer-to-peer fashion. Peer-to-peer communication and data transfers in the Gnutella system can take place using HTTP, TCP or UDP over a dynamic port range.

The latest entry into the P2P list is the Skype application developed by Kazaa, a free program that uses the P2P technology to bring cost-effective and high-quality voice communications over the Internet around the world. Skype introduces new challenges to the administrator, because it is designed to work seamlessly across firewalls and NAT devices and decentralizes subscriber information. In addition to voice communication, it supports instant messaging and file transfer capabilities which opens the use to exchange confidential information or receive links to malicious sites.

Although some architectural and procedural differences exist between the various P2P clients, there are some shared similarities that make their usage in corporate and residential networks questionable.

### 1.1 P2P Considerations for the Enterprise

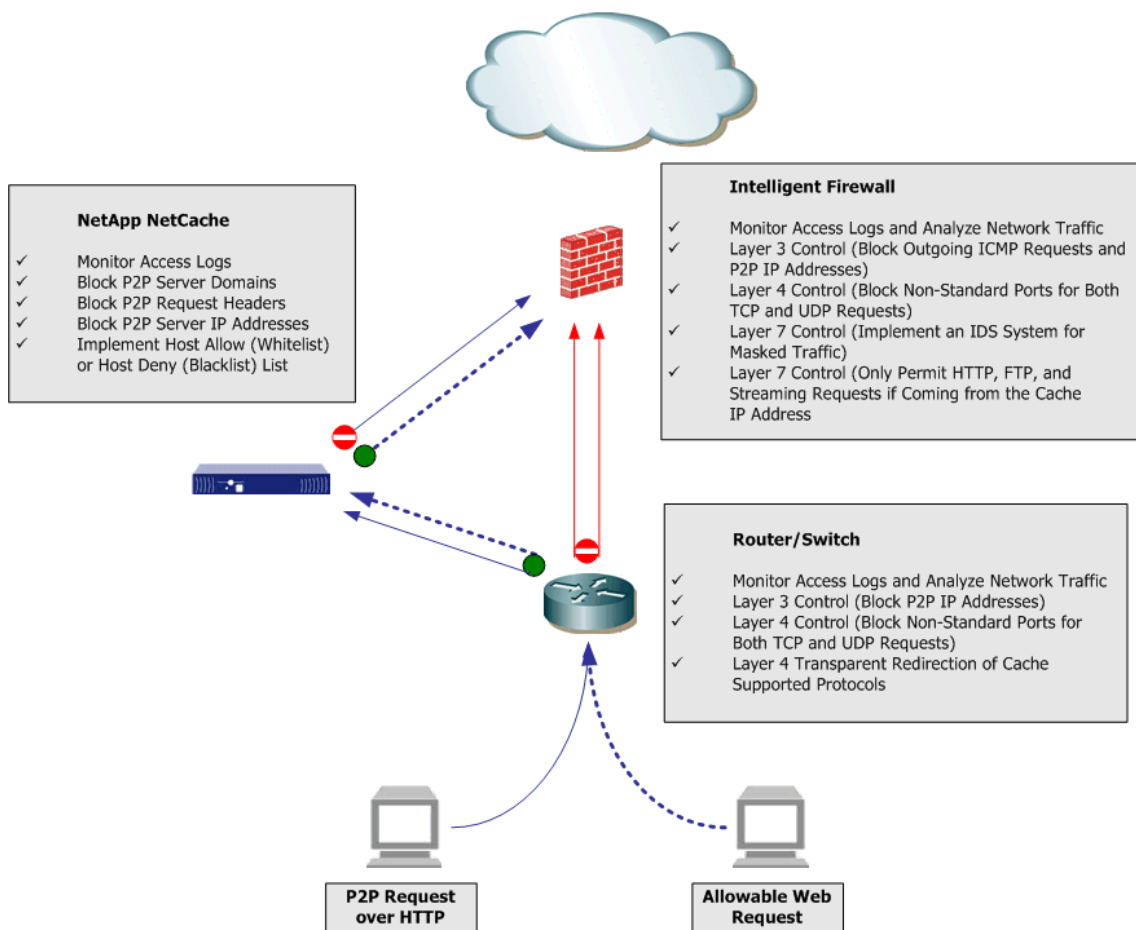
- **Security.** The lack of encryption and other security-related features associated with these applications can be exploited, allowing individuals to distribute almost any type of content throughout the peer-to-peer network unchecked. This includes viruses, worms, and other malicious types of code.
- **Legality.** P2P applications allow users to share copyrighted and legally binding documents such as music and software programs.
- **Privacy.** Users may inadvertently expose sensitive individual and corporate information to the peer network, jeopardizing protected and valuable information.
- **Productivity.** Employees using P2P applications in the workplace typically spend a significant amount of their time searching for and viewing shared media.
- **Network congestion.** P2P applications by definition consume a large amount of bandwidth—the result of users downloading new content and simultaneously sharing with others. An overburdened network can lead to poor application responsiveness and unnecessary downtime for corporate users.

## 2. A Tiered Approach to Control P2P Traffic

Unfortunately, P2P clients don't rely exclusively on HTTP or other application layer protocols for their operation. For P2P clients that rely on HTML for their navigation and interface, as do many of the ad-supported implementations, using a Web cache to block HTTP requests results in an unusable product. However, the effectiveness of this approach depends on the presence of HTML. In many cases, blocking HTTP requests from the client results in an ugly display but still allows users to actively search and download files due to the application's inherent ability to switch between multiple protocols. Similar behavior is observed by blocking specific UDP and TCP ports at the router or firewall—the client will subsequently try a different port and/or use HTTP. Because of the dynamic nature of P2P networking and its added ability to mask P2P traffic by using known service ports, such as ports 25 (SMTP) and 143 (IMAP), some form of network intrusion detection system (IDS) may be required to detect usage. Fortunately, many of today's intelligent firewall products, and NetApp partner products such as the Webwasher® Content Security Management (CSM) suite, offer various forms of application-level inspection technology that can be used to detect and block P2P traffic that does not typically get redirected to a cache.

It's important to note that caching appliances can impose restrictions only on traffic that is configured to pass through them. HTTP, FTP, and RTSP are a few of the more commonly cached protocols. Once requests are redirected to the cache, they can be subjected to a gauntlet of security checks before allowing the request to be fulfilled. Please refer to [NetApp Internet Access and Security \(IAS\) Features Overview<sup>1</sup>](#) for details on how NetCache can be used to consolidate the deployment and administration of multiple security technologies into a complete solution for securing Web access.

As a general rule, network and transport layer requests should be blocked at either the router or the firewall. NetCache appliances can be configured to block specific IP addresses and TCP and UDP port ranges by using the bandwidth management feature, but this simply adds another layer of complexity to the deployment—an L4 device would have to be configured to redirect this traffic to the appliance for a given port range. It's more efficient to block lower level (layers 3 and 4) traffic on pass-through devices that are designed for that purpose. Hence the need to create a tiered access policy.



**Figure 1) Tiered architecture example.**

Figure 1 illustrates a simplified version of a tiered architecture that could be used to provide maximum control over P2P and other unwanted traffic. Because P2P applications are so adaptable, it's necessary to implement access rules at multiple layers.

### 3. Recommended Process for Controlling P2P

#### 3.1 Monitor and Analyze Network Traffic

Because of the dynamic nature of P2P networking, it's necessary to monitor and analyze network traffic on an ongoing basis. Preferred servers and protocols can change between software releases, and taking a proactive approach is the only way to ensure that you are implementing an effective access control policy. Installing the various P2P clients on a test machine can give you vital information about their behavior.

Most P2P clients will issue a series of ping and DNS-related requests that can be used to get the IP addresses of the current set of servers, whereas Skype uses a hostcache file, which contains a list of IP addresses and ports of supernodes it could connect.

The following is a list of entries in a hostcache file of the Skype application installed on one of the test machines:

66.235.181.9:33033

82.41.80.59:1651  
152.2.12.205:3348  
213.113.160.107:34693  
24.209.43.253:32285  
218.131.72.53:40184  
213.187.197.112:4636  
221.113.163.103:1676  
194.54.18.152:38987  
69.118.240.115:2154

Identifying and blocking those IP addresses prevents users from using P2P applications and thus prevents them from using its capabilities. However, be aware that these server IP addresses and associated ports change on a regular basis, so it is necessary to periodically update any associated policies.

The following searchable port databases are useful for tracking P2P related ports on an ongoing basis:

- The Internet Ports Database (<http://www.portsdb.org/bin/portsdb.cgi>)
- Ports at Tantalo.net (<http://ports.tantalo.net/>)
- Port Database at Treachery.net (<http://www.treachery.net/tools/ports/>)

#### **Evaluating Policy Effectiveness by Using the NetCache Appliance**

Cache access logs, in combination with policy statistics, should be used to determine the effectiveness of the currently configured P2P policy. Detailed, customizable logging is available for all of the supported protocols and can be tailored for any third-party log analysis software. Additionally, policy statistics provide a feedback mechanism that can be used to fine tune P2P access controls. Figure 2 shows the policy statistics derived from an access policy designed to block P2P traffic transferred over HTTP.

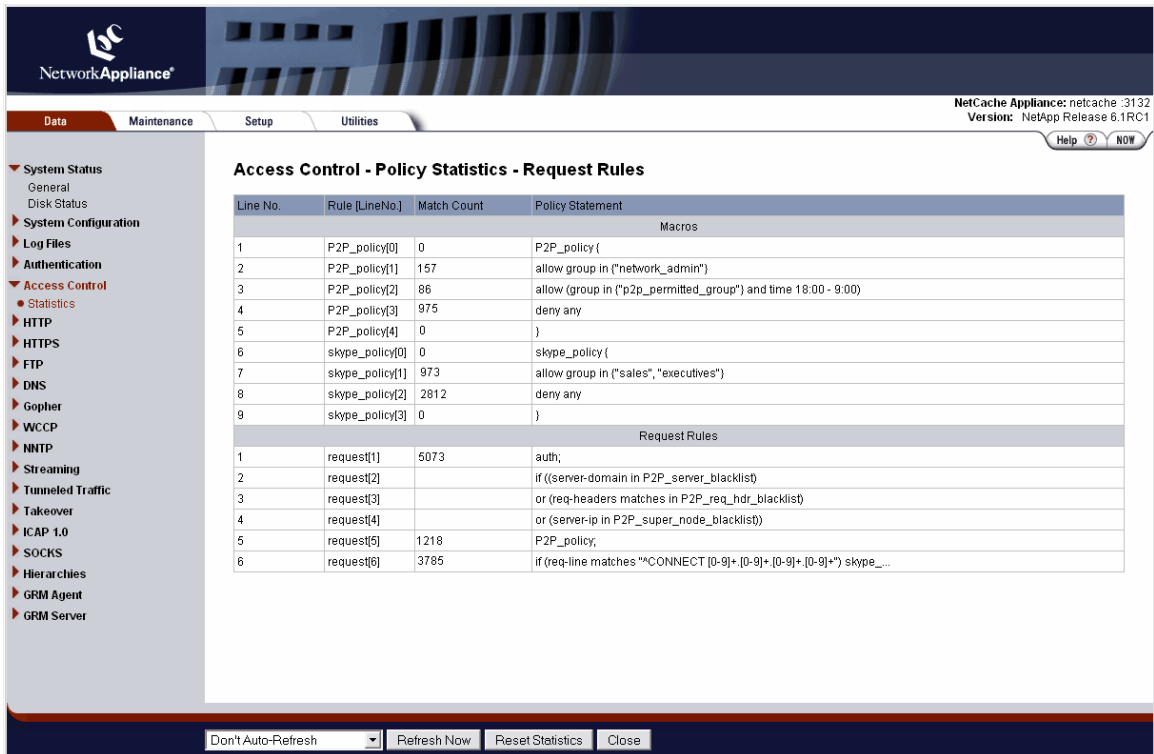


Figure 2) Per rule policy statistics.

Figure 2 illustrates the number of rule matches that occurred for a policy designed to block traffic to P2P servers. Policy statistics are available from both the GUI and the CLI of the NetCache appliance.

### 3.2 Block Supernode and Index Server IP Addresses

Information derived from log and network traffic analysis can be used to block P2P clients. Once the supernode or index server is identified, access control policies (ACPs) can be configured on the router, switch, firewall, and cache to deny all access to those addresses. Depending on the network architecture, it makes sense to configure IP-based (layer 3) ACPs only on the core routers and switches to lessen the burden of managing numerous access lists on multiple devices. As illustrated in Figure 1, all of the HTTP traffic is forced through the cache, and the firewall is configured to accept HTTP requests only from the cache IP address. Traffic passing through the cache not only is exposed to ACPs specifically blocking HTTP access to those IP addresses, but also potentially experiences the added security benefits provided by URL filtering, virus scanning, active content stripping, authentication, monitoring, and logging. IP-based ACPs configured on the firewall catch any traffic that doesn't get redirected to the cache.

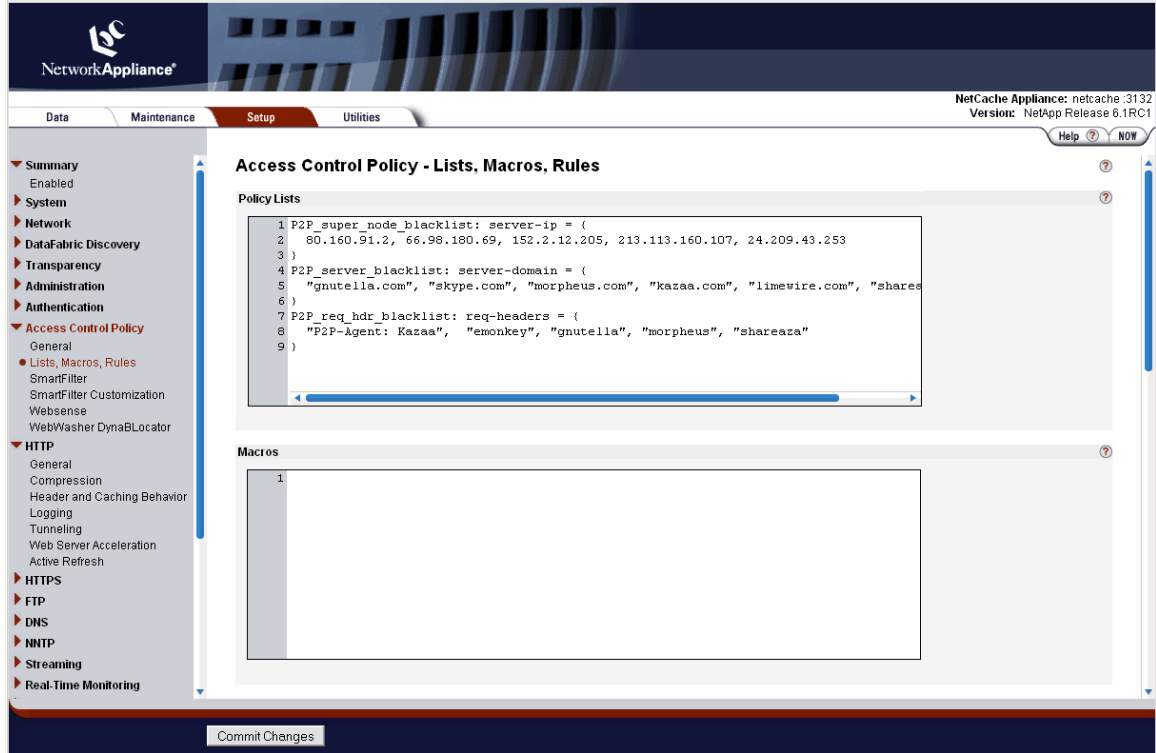
#### Generic Firewall/Router IP Access List Example

Access-list 101 deny ip any host 66.235.181.9 (Deny any IP traffic to destination 66.235.181.9)  
 Access-list permit ip any any (Allow everything else)

### 3.3 Block P2P and Spyware-Related Domains, Request Headers, and IP Addresses

NetCache appliances offer granular control over all of the supported protocols. For controlling P2P traffic transferred over HTTP, access control rules that specifically block P2P-related server domains, IP addresses, and request headers offer a powerful tool to combat its use within the enterprise. Because it can be difficult and time-consuming keeping up with the dynamic nature of P2P networking, your policy can be

further reinforced by enabling URL filtering on the cache. URL filtering vendors such as Webwasher, Websense®, and Secure Computing® provide up-to-date name and numerically based URL lists that add an additional layer of security and control to any deployment.

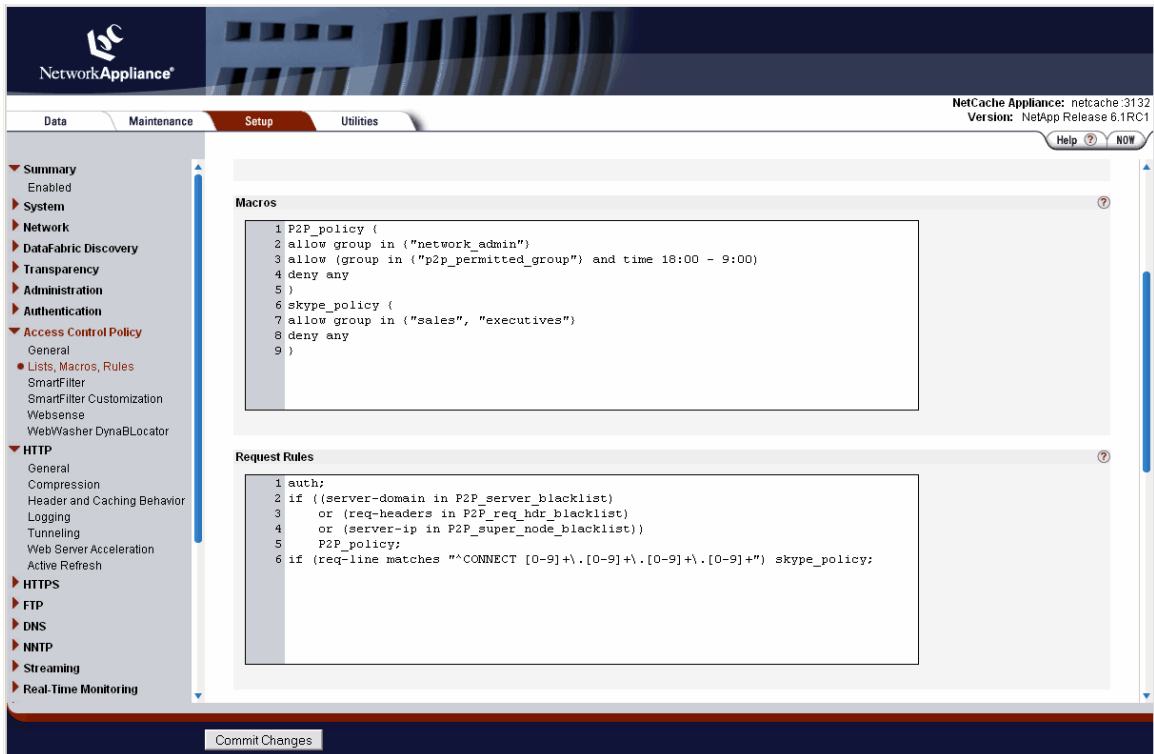


**Figure 3) Defining P2P blocking conditions.**

Figure 3 illustrates how to define lists on NetCache to block HTTP requests for server domains and request headers related to the most common P2P applications. ACP configuration is available from both the GUI and the CLI of the NetCache appliance.

### Blocking Skype

Skype initially attempts to connect directly to the supernodes from the hostcache entries. If that fails mainly due to the firewall blocking them, Skype uses the proxy server and attempts to tunnel the traffic over HTTP. So blocking the HTTP CONNECT method would deny users the use of Skype but also denies access to some legitimate sites, including e-banking, e-commerce, etc. using HTTPS. From analyzing the packet traces it was observed that the Skype client always specifies the supernode by the numeric IP address in the HTTP CONNECT request, so using a rule to block all CONNECT requests using the IP address effectively blocks all Skype login attempts and still allows other CONNECT requests using fully qualified domain names.



**Figure 4) Blocking P2P server domains and request headers.**

Figure 4 illustrates how to configure macros and request rules on NetCache to translate the business policy that says who is allowed to use P2P and Skype services.

### Sample Access Control Policy for Blocking P2P Traffic, Including Skype

The following NetCache access control policy blocks embedded HTML in the P2P applications, rendering it virtually unusable. This information would be entered into the ACP configuration of NetCache via the GUI or the CLI.

```

config.policy.lists = \\
    P2P_super_node_blacklist: server-ip = {
        80.160.91.2, 66.98.180.69, 152.2.12.205, 213.113.160.107, 24.209.43.253
    }
    P2P_server_blacklist: server-domain = {
        "gnutella.com", "skype.com", "morpheus.com", "kazaa.com", "limewire.com",
        "shareaza.com"
    }
    P2P_req_hdr_blacklist: req-headers = {
        "P2P-Agent: Kazaa", "emonkey", "gnutella", "morpheus", "shareaza"
    }
\\

```

```

config.policy.macros = \\
    P2P_policy {
        allow group in {"network_admin"}
        allow (group in {"p2p_permitted_group"} and time 18:00 - 9:00)
        deny any
    }
    skype_policy {
        allow group in {"sales", "executives"}
        deny any
    }
\\
config.policy.rules.request = \\
    auth;
    if ((server-domain in P2P_server_blacklist)
        or (req-headers in P2P_req_hdr_blacklist)
        or (server-ip in P2P_super_node_blacklist))
        P2P_policy;
    if (req-line matches "^CONNECT [0-9]+\.[0-9]+\.[0-9]+\.[0-9]+")
        skype_policy;
    deny any;
\\

```

### 3.4 Implement a Whitelist/Blacklist Access Control Policy

All of the previous examples focus on explicitly blocking variables associated with P2P applications. However, an alternative to explicitly denying, or “blacklisting,” client access to specific sites involves implementing a “whitelist.” A whitelist, or allow list, allows client access only to designated sites. All requests for sites that are not specified in the whitelist are categorically denied.

#### Whitelist Example (from the NetCache CLI)

```

config.policy.lists = \\
domain_whitelist: server-domain = {
    "mycompany.com", "myintranet.com", "mybank.com"
}
config.policy.rules.request = \\
    auth;
    allow server-domain in domain_whitelist;
    deny any;
\\

```

## 4. Conclusion

Single-facet security solutions alone can't adequately address today's adaptive threats. To effectively control intelligent new technologies, such as peer-to-peer networking, a comprehensive solution is required. This involves implementing access control policies at multiple layers within the network and a proper understanding of the application in question. A tiered security architecture offers tighter control and a higher level of protection against technology designed to evade detection.

## 5. Revision History

Date	Name	Description
April, 2006	Arun Raman	Update
October, 2004	Tobin Sears	Creation

<sup>1</sup> <http://www.netapp.com/library/tr/3236.pdf>

