Global Information Assurance Certification Paper

GIAC Certification Practical

Robert M. Gannon
GSEC Version 1.4b Option 2 – Case Study
November, 2003

Circumventing Access Control Lists by Transparent Proxy – A Case Study

Abstract:

Open proxy servers are a well-known security problem that threatens the security of the Internet by making it possible for unscrupulous people to mask their identity while performing any number of malicious acts. Reading about the open proxy problem, one would come to believe that a serious configuration error such as lack of access controls is necessary for a proxy to be abused.

This paper describes a method used in an actual case to circumvent seemingly adequate access controls by using the transparent caching mechanism of the WCCP protocol to abuse an otherwise protected network for the purposes of sending spam and connecting anonymously to unsavory sites. We will also see how this vulnerability could have been used to bypass a firewall and gain access to a protected network.

Background Information:

A strong security policy includes multiple, redundant methods for controlling access to computing resources. This concept, known as Defense-in-Depth, is neither new, nor exclusively tied to the topic of computer security. Examples of the concept of defense-in-depth can be found in writings on both computer security topics and nuclear plant safety. http://securityresponse.symantec.com/avcenter/security/Content/security.articles/defense.in.depth.html) and (http://www.nei.org/index.asp?catnum=2&catid=55)

Relying exclusively on Access Control Lists or any single mechanism is insufficient to protect critical infrastructure. In this paper, we will see how access controls were completely bypassed due to a vulnerability associated with transparent web caching.

Conventions:

For the purposes of this discussion, the actual IP addresses of the described network have been changed. In all cases, the actual IP addresses have been altered so that the first three octets are now represented by RFC 1918 (private) addresses. All addresses will be represented in this paper as 192.168.10.XXX. For more information on RFC 1918, see (http://www.ietf.org/rfc/rfc1918.txt?number=1918). The company involved will be

referred to as Unwitting Accomplice, Inc. (UAI). Any similarity between this and an actual company name is purely coincidental.

Description of the Target Network:

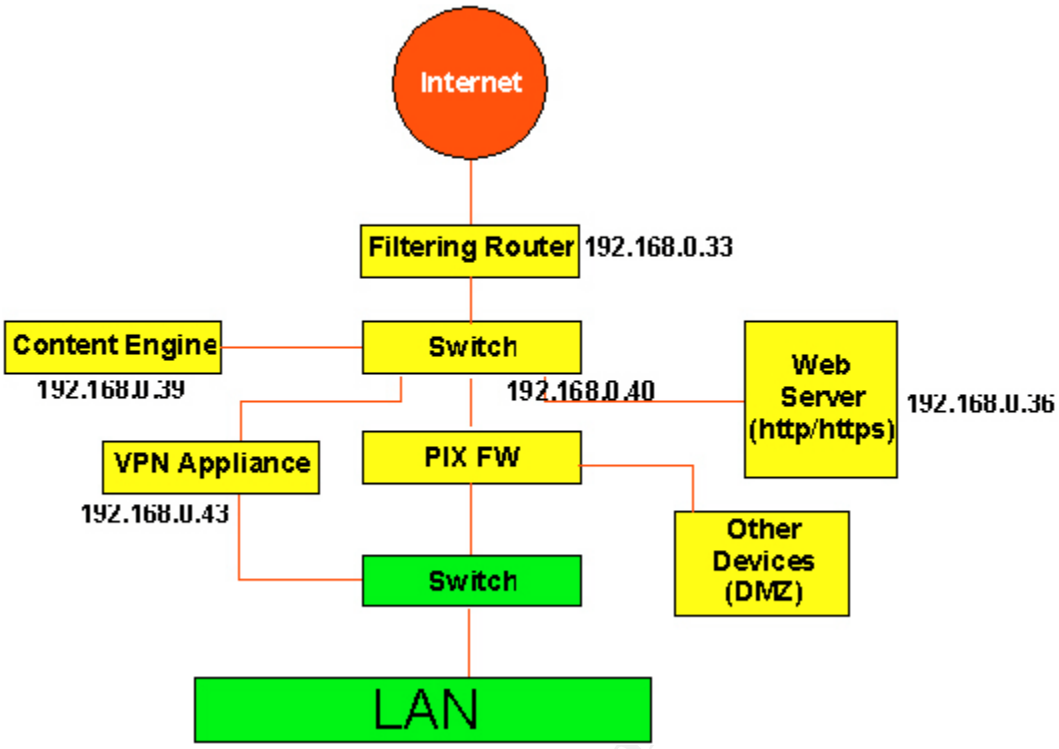
UAI's Internet connection is through a Cisco router connected to an ISP T1. The router is configured to allow access only to those services which require access from outside the network. Outbound connections use dynamic access controls - Context Based Access Controls or CBAC to allow return traffic. Cisco describes CBAC as follows:

> CBAC intelligently filters TCP and UDP packets based on application-layer protocol session information. You can configure CBAC to permit specified TCP and UDP traffic through a firewall only when the connection is initiated from within the network you want to protect. CBAC can inspect traffic for sessions that originate from either side of the firewall, and CBAC can be used for intranet, extranet, and Internet perimeters of your network. …
> ….CBAC inspects traffic that travels through the firewall to discover and manage state information for TCP and UDP sessions. This state information is used to create temporary openings in the firewall's access lists to allow return traffic and additional data connections for permissible sessions.
> (http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fsecur_c/ftrafwl/scfcbac.htm#1000932)

The router is also equipped with the IOS Firewall Feature Set (FFS), and Cisco IOS Intrusion Detection System. Logging is to an internal CiscoWorks server. Behind the filtering router is a Cisco PIX 515 firewall, a web server listening on ports 80 (http) and 443 (https), a third-party VPN appliance and a Cisco Content Engine 507. The segment of the network between the filtering router and the PIX serves as a De-Militarized Zone (DMZ). Other devices are connected to a third PIX interface to form another DMZ. This second DMZ is protected by both the Access Control Lists on the router and by the PIX rules. With this configuration, most of the network is invisible from the outside. We will sometimes refer to the Content Engine as simply "the cache".

As the following partial diagram shows, the Content Engine has a single interface connected to the portion of the network between the border router and the PIX. Packets do not pass through the cache in the same manner as through a router or a switch. The router ACL does not permit any access to the cache from the Internet. The cache was installed primarily to improve user perceived performance when loading web pages and is intended to be used primarily by clients on the protected network. As we'll see, the cache can also deliver pages to outside clients if needed.

Network Diagram:



The router and content engine communicate through the Web Cache Communication Protocol (WCCP). When a request for a web page arrives at either router interface, the router transparently passes the request to the cache. If the cache has a valid copy of the page on the local hard disk, the request is filled from the cache. This eliminates the need for the request to be forwarded to the destination web server. In the case of a low bandwidth connection between the client and the web server, there can be a considerable improvement in the response time as the page is served from the local cache over the local, high-speed network connection. The performance improvement is most dramatic when multiple clients request the same page. The page is retrieved only once from the web server and repeatedly delivered to multiple clients from the cache. If the cache engine does not contain a local copy of the requested page it will contact the web server on behalf of the client, retrieve the page, store it locally, and deliver it to the client. The next time any client requests the page it will be served from the local cache.

> If the cache contains the referenced URL it is checked for freshness by comparing with the "Expires:" date field of the content, if it exists, or by some locally defined freshness factor. Stale objects are revalidated with the server, and if the server revalidates the content, the object is remarked as fresh. Fresh objects are delivered to the client as a cache hit. If the cache does not have a local copy of the URL, or the object is stale, this is a cache miss. In this case the cache acts as an agent for the client, opens its own session to the server named in the URL, and attempts a direct transfer to the cache.
> (http://www.cisco.com/warp/public/759/ipj_2-3/ipj_2-3_webcaching.html)

An important point to note is the behavior of the cache when a requested page is not available locally. In such cases, as mentioned above, the cache will retrieve

the web page on behalf of the client. In this instance, the cache is acting as a proxy. It is this proxy behavior that forms the basis of the vulnerability discussed below. It is important to note that even though we are primarily discussing http proxy, other protocols can be redirected to the Content Engine as well.

Packets between the router and the Content Engine are encapsulated with the Generic Routing Encapsulation protocol (RFC 1701, 1702, 2784 and 3147). Packets from the original source are repackaged within a GRE packet and passed between the two devices.
This encapsulation of one packet within another creates a "tunnel" between the two devices. This is similar to the behavior of VPN protocols. Clients are not aware of the existence of the cache and are not explicitly configured to use it. The caching is completely transparent to the two endpoints of the original connection.

Let's look at an example of transparent caching in action:

1. Client #1 inside the network requests a page from a web server outside the corporate network. Request leaves the client with a destination address of the web server.
2. Request arrives at the border router destined for outside the network.
3. Router transparently directs the request to the cache.
4. The cache does not have a valid, local copy of the page.
5. Cache contacts the web server directly and retrieves the page.
6. Page is delivered back to the router for delivery to Client #1.
7. Router sends the page to the requesting client. (Client #1)
8. Client #2 requests the same page.
9. The router forwards request to the cache.
10. The cache now has a valid local copy of the page.
11. Cache sends page to router for delivery to Client #2.
12. Router sends the page to the requesting client (Client #2)

From the standpoint of both Client #1 and Client #2, the page was retrieved directly from the web server. The only difference likely noticed is that Client #2 experienced much faster page retrieval since the entire transaction took place on the local network.

Security Considerations:

1. Content Engine

The default implementation of WCCP at the time of the installation did not place restrictions on transparent connections. In other words, the WCCP enabled router will gladly communicate with any WCCP web cache. This opens up the possibility of a malicious attacker connecting a cache that contains altered copies

of web pages. Requests to legitimate web sites can be redirected to sites of the attackers choice.

> Although this process is not enabled by default, and takes place only if a user specifically configures the router to enable WCCP, there is no authentication in WCCP itself. A router configured to support Cache Engines will treat any host that sends it valid WCCP hello packets as a cache engine, and may divert HTTP traffic to that host. This means that it is possible for malicious users to divert web traffic passing through such a router, even though they may not have either physical or configuration access to the router.
> **(http://www.cisco.com/warp/public/770/wccpauth-pub.shtml)**

Access controls can be applied to the Content Engine to limit the types of traffic that can be redirected. The default install is not configured with the highest possible security.

> The default configuration of the proxy feature can be abused to open a TCP connection to any reachable destination IP address and hide the true IP source address of the connection. This behavior has been implicated in a variety of undesirable and possibly illegal activities such as transmitting unsolicited commercial e-mail, unauthorized network scanning, and denial of service attacks. **(http://www.cisco.com/warp/public/707/transparentcache-tcp-relay-vuln-pub.shtml)**

2. VPN Appliance

The VPN appliance is a third-party product that has passed strenuous penetration tests and holds several government certifications for security. Unfortunately, the appliance is a proprietary "black box" and much of the configuration and or functionality is not readily apparent to the end user. Such configurations MAY be reasonably secure but it is very difficult to verify. An article on this subject appeared in a recent "Security Advisor" column: (http://www.infoworld.com/article/03/10/31/43secadvise_1.html)

When the VPN appliance was installed several years ago, it was believed to be at least as well hardened as the firewall so no additional access controls were placed in front of it. The appliance acts as an endpoint for an IPSEC VPN tunnel and implements standard ISAKMP and ESP.

3. Web Server

The web server is an IBM HTTP Server running on OS/400. The only access allowed through the router ACL is to ports 80 and 443 (http and https). The web server is constantly the object of unsuccessful attack attempts. Server logs routinely show blocked attempts to exploit IIS, Apache and other known vulnerabilities even though it is not an IIS or Apache server. Such mismatched attempts are the footprint of automated attacks and/or "script kiddies".

4. General

The network access controls are configured according to the principles of "least access". Only those ports needing access from the Internet are open. All others are blocked. ICMP and SNMP are not available during normal operation. ICMP

may be occasionally enabled for troubleshooting purposes. The router is kept up to date with the latest stable release of the IOS. The IDS and Firewall Feature Set are configured to help prevent Denial of Service and other common attacks. Together, the IDS and FFS form a somewhat limited Intrusion Prevention System.

The Cisco IOS IDS (Intrusion Detection System) checks for a fairly limited number of probes/attacks. It does a good job of detecting certain denial of service attacks such as SYN flood. We routinely see log entries where the IOS IDS has alarmed on and dropped excessive half-open connections, etc. It also helps to block SMTP related attacks by recognizing invalid SMTP commands and dropping the connections. It does not recognize a number of application level attacks that use otherwise legitimate protocols and are thus allowed by the router filters (HTTP, for example). In general, if an attack comes through an allowed port, access lists are not sufficient to detect, identify and/or stop them. The IDS improves on this by recognizing a small number of these attacks. A big drawback in the IDS is that it is not easily customizable and relies on signatures provided by the vendor. If an attacker probes or attacks through an allowed protocol such as http, the IOS IDS will not alarm or block except in very limited cases where the attack is very well known and has been known for a long time. IIS attacks routinely come through unnoticed for example.

Proactive Protection Steps:

With the network setup as noted above, we were confident that we had a reasonably secure configuration. “Reasonably” meaning that some access was necessarily allowed but was limited to those hosts, ports and protocols essential to conducting business. No network connected to the public Internet is absolutely secure. Keep in mind that the cache engine was thought to be a non-threatening device as installed. It was placed in a protected section of the network behind a filtering router with IDS and Firewall Feature Set enabled. Access to the cache was blocked by router Access Control List entries.

Even with a fairly high confidence level in the configuration, we took some additional steps to ensure our security:

1. Conducted multiple, exhaustive ports scans of the network from outside (with permission) to determine which hosts and ports were “visible” from the Internet. These scans were done with full knowledge of the internal topology of the network. Scans revealed exactly what was expected. The VPN server and web server listening on the expected ports. There was no response from the cache engine on any port and no ICMP (ping) or SNMP responses from any device. The company e-mail server also responded on the SMTP port (25) as expected. The mail server uses Network Address Translation behind the PIX and is not shown on the diagram above.

2. Identified and disabled open SMTP relays. The default installed configuration of the mail server allowed relaying from outside, un-trusted hosts.

3. Contracted with an outside security vendor for a penetration test. No unexpected vulnerabilities were found. To be fair, the vendor was given minimal information about the internal design of the network. Our aim was to see what information was publicly available about the network and to see if their tools could find any openings that our internal scans had missed.

4. Searched the Common Vulnerabilities and Exposures database (cve.mitre.org) for known vulnerabilities. The searches were done for both the VPN server and the web server. No entries were found for either. Additional searches were done at CERT/CC (http://www.cert.org/nav/index_red.html). Remember that the VPN server was a proprietary "black box" and not widely deployed.

5. Conducted a "Google" search for any information relating to our company and/or our network. Some interesting entries were found. This lead to a policy update on revealing information on technical newsgroups, etc. None of these postings have any relation to the incident described in this paper.

6. Twice daily checks of 9 different security websites to keep informed of newly discovered vulnerabilities, viruses, worms, Trojan horses, etc.

7. Subscribe to SANS, CERT, Cisco and Microsoft security bulletins.

8. All planned network changes verified by two network engineers before implementation. This helps to reduce configuration errors.

Obviously, the above steps do not guarantee perfect security. They did, however, leave us fairly confident that there were no gaping security holes through the network border.

The Problem:

When users began to complain about excessive delays and slow Internet connections, my first thought was that, like past reports, the cause would be traced to high usage within the company. One IT staff member downloading a service pack can put a considerable and noticeable load on the circuit during peak usage times. Before implementing egress filtering, particularly web filtering, slowdowns were often found to be attributable to a user listening to Internet radio or downloading music files. We will skip for now the potential liability from having a user download pirated music through the corporate Internet connection. While such slowdowns are a nuisance, it was generally easy to find and fix the problem simply by educating the users. The cache engine was installed to speed up users' Internet experience and it can help to alleviate the effects of temporary

activity spikes on the Internet connection. Ironically, as we'll see later, its existence on the network eventually had the opposite effect. At the time of the initial complaints, I was not overly concerned. We had seen this before.

To help track the problem, I opened a console session on the border router to see, in real time, the traffic entering and exiting our network. As soon as the terminal session became active, it was apparent that we had a more significant problem. E-mail messages, each with a unique "From" and "To" address were exiting our network at an alarming rate.

Even during peak usage, logging information displayed on the router console scrolls at a rate that allows a quick perusal of network activity. This time the lines were scrolling so fast that the only way to read the messages was by checking the router logs on the syslog server. We were aware of the problem of open e-mail relays and had checked the configuration of our servers. The surprise here was that the source address of the e-mails leaving the network was 192.168.0.39 – the cache engine! A glance at the e-mail subject lines told me that Unwitting Accomplice, Inc. had become an unwitting participant in a large-scale SPAM operation. The cache engine does NOT run an SMTP server. My curiosity was instantly aroused. How could this be happening? Furthermore, how did someone discover the existence of the cache engine when I, with full knowledge of the network topology, could not "see" it from outside the network? The quest for answers to those questions eventually led me to learn quite a lot about a little publicized security problem. The research led me to information about a number of potential vulnerabilities related to open proxy servers. The first step in the process though, had to be to stop the abuse of our resources so I set the questions aside temporarily and proceeded to do some testing on the live network.

Since the Content Engine was not essential to the operation of the network, my first step was to simply unplug it and watch the router console. When the cache was unplugged, the e-mails immediately stopped. This confirmed that the cache engine was somehow an integral part of the incident. Now on to figuring out how it was happening and how to stop it without removing the cache from our network.

I wanted to discover in more detail the exact method that was being employed to exploit the cache engine as a SPAM relay. To do this, I installed the Ethereal® protocol analyzer (http://www.ethereal.com) on the segment attached to the router's internal interface and set it to capture all packets entering and leaving that interface. When the "sniffer" was setup, I plugged the cache back in and watched the router console. Within about 1 minute the rogue e-mail traffic resumed. I captured about 10,000 packets and unplugged the cache again. Looking through the Ethereal capture file left me more confused than ever. The destination address on the original incoming packets was that of the VPN appliance and the source of the original outgoing packets was the same. The

packets were being encapsulated with GRE and bounced through the cache engine.

1. Packet enters the router with destination address of the VPN server (192.168.0.43 port 80)
2. Packet is wrapped in a GRE packet and sent to the cache (192.168.0.39)
3. Packet is sent back to the router (192.168.0.33) with destination address of the SPAM victim and source address of the cache.
4. Inside the GRE packet from the cache to the router is a packet with the source address of the VPN server even though the return traffic did not originate from the VPN server. It came from a host outside of our network. To the recipient of the unwanted mail, it would look like it came from our VPN server.

Since I did not yet understand the WCCP protocol or how it was being exploited, I decided to involve Cisco's Product Security Incident Response Team (PSIRT). They were very responsive and asked for additional information. In response to their request I collected and sent them:

1. The configuration of the Content Engine
2. The configuration of the router including access control lists.
3. The Ethereal .cap (capture) file
4. A description and simple diagram of the network.
5. Syslog entries from the router

The Cisco PSIRT promptly responded with a security advisory from May of 2002 that described the problem and identified a solution. In the default configuration for the version of the cache engine software we were running, any TCP connection could be diverted through the cache, hiding the originating IP address. The PSIRT was aware of instances where this behavior had been exploited for the anonymous delivery of SPAM and for other illicit activities. (http://www.cisco.com/warp/public/707/transparentcache-tcp-relay-vuln-pub.shtml) This is the page referenced above that describes the ability of attackers to subvert the cache for their own purposes. At this time, I had not read the advisory yet.

We applied the workaround described in the advisory and reconnected the cache to the network. Monitoring over a period of several days appeared to indicate that the problem had been resolved. The incident appeared to be over at this point. A known vulnerability that had escaped our attention had been exploited, the solution was readily available and the workaround was easy to implement. We had survived this incident relatively unscathed and were fairly confident that our security posture had once again reached an acceptable level. Still, I continued to monitor the circuit for unusual behavior. We also began evaluating installation of a more robust intrusion detection system that could alarm when traffic leaves the network from unexpected sources.

After an incident-free period of approximately two weeks, I once again began to see unusual traffic passing through the router from the cache engine. Another Ethereal capture and subsequent analysis of the traffic revealed that the destination of the unwanted packets was no longer the VPN server; it was now the web server. The attackers had circumvented the cache access lists by sending the traffic to an allowed port (443). We had stopped the abuse of the VPN server address but had still allowed proxy of https traffic. Since the web server does listen on port 443 for https connections, we must allow that traffic into the network for outside clients to access the server. This created a very similar situation to the one described above. Packets directed to the https server were transparently redirected to the cache and the cache obediently acted as a proxy for the outside client. To take advantage of this opening, the attackers could use a variety of methods. One is the http CONNECT method.

Http CONNECT allows connections through a proxy to ports other than http/https. Since the purpose of https is to secure communications, the proxy is not allowed to see inside the connection request. Because of this, a request to an SSL enabled web server can potentially be redirected to any other port such as SMTP. Another possible use of CONNECT is anonymously accessing an ICQ server to hide the identity and location of the logged in user.

> The HTTP CONNECT method, as well as other HTTP methods and FTP commands, can be abused to establish arbitrary TCP connections through vulnerable proxy services. An attacker could use a vulnerable proxy service on one network as an intermediary to scan or connect to TCP services on another network. In a more severe case, an attacker may be able to establish a connection from a public network, such as the Internet, through a vulnerable proxy service to an internal network. (http://www.kb.cert.org/vuls/id/150227)

After reading the CERT/CC Vulnerability Note I wondered if any of the unusual traffic I had seen was due to abuses other than SPAM. Another look through the Ethereal captures taken during the initial investigation two weeks prior revealed the following transaction channeled through our network:

```
CONNECT login.icq.com:443 HTTP/1.0
HTTP/1.0 200 Connection Established
```

Looking at the individual packets revealed the following sequence:

1. Packet 1154: From attacker/abuser to the VPN server. This was the above CONNECT request.
2. Packet 1156: From cache engine (proxy) to login.icq.com (64.12.161.153)
3. Packet 1159: From VPN server (apparently) to attacker/abuser. (Connection Established response)

The packets in between (1153,1155, 1157 and 1158) were TCP three-way handshakes (SYN, SYN ACK, ACK) between the hosts. Note that the VPN server did NOT connect to the ICQ server, the cache engine on behalf of the

abuser made the connection. The response returned to the attacker appeared to come from the VPN server's address. The VPN server was not involved in this transaction other than to provide an open address behind the filtering router. Once the router allowed this packet as a valid connection, the cache engine took over as a proxy and made the request on behalf of the VPN server.
After the cache engine was reconfigured according to the Cisco Security Notice mentioned before, it took the attackers only a short time to switch their tactics and use the SSL enabled web server as the legitimate destination. Subsequent Ethereal trace analysis showed traffic similar to the ICQ example above with the only difference being the initial destination address of the first packets. Instead of aiming at the VPN server, they had now chosen the web server listening on port 443 at 192.168.0.36. The proxy was still configured to allow proxy of https at this point. It was not necessary for the traffic to ever reach the web server for the attacker to use the proxy for his/her own purposes. It was only necessary that the router access control lists allow the connection. Once again – there was no "open" path from the outside world to the cache engine. The filtering router blocked all access to the cache. Once again, unplugging the cache stopped the unwanted traffic.

Further analysis of the logs showed access to other services outside our network. One example was access to a pornographic website:

```
GET http://cgi.sexswap.com/adswap.dll?showad?accountnumber=8213&PageNumber=0
HTTP/1.1
Referrer: http://hgamecgs.n3.net/
Accept: */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 4.0; Windows 95)
Pragma: no-cache
X-Forwarded-For: 202.66.36.156
Host: cgi.sexswap.com
```

Looking at the above capture, note the "X-Forwarded-For" address. This belongs to an ISP in China and was probably dynamically assigned to the abuser. Often attackers use a chain of proxies to connect to the final destination so it can be impossible to trace back to the origin.

Seeing the ability of an outside attacker to "hijack" our cache to use it as an anonymous proxy for whatever activity he/she would like to hide, I started more research about the general subject of open proxies. I found that open proxies are a much larger problem than I suspected and that they are used for a variety of illegal or questionable purposes. The single largest use appears to be Unsolicited Commercial Email or SPAM but they can be exploited for any purpose when the perpetrator wishes to keep their identity secret.

Some years ago, spammers discovered that they could exploit open mail relays to send mass quantities of SPAM utilizing somebody else's mail server. This not

only gave them free bandwidth but it also transferred the potential liability to the owner of the open mail relay. To combat this problem, lists of open mail relays began to appear and companies began checking incoming e-mail and rejecting it if it came from a known open relay. These lists, known as RBLs or Relay Black Lists are maintained by a variety of organizations such as SpamCop (http://www.spamcop.com). If your company runs an open mail relay, you run the risk of being listed on a RBL, which could cause legitimate mail from your company to be rejected by the recipient. As the problem of open mail relays became widely known, many companies, including UAI, took steps to secure their mail servers against relaying. The disappearance of many open mail relays caused the purveyors of SPAM to look for different methods of anonymously delivering their "product". Their solution is open proxies. There are now listings of open proxies accessible to both spammers and those trying to eliminate the problem. One list contains over 900,000 entries along with some very good additional information on the open proxy problem. Below is an excerpt from the introduction to this list. ***Warning:*** *This page is very large and takes quite some time to load even over a cable modem or T1 connection.*

> Please note that open proxies are **NOT** open SMTP relays; open proxies are a different (and far more serious) problem.
> Open proxies have four key characteristics that make them of exceptional interest to spammers and hacker/crackers:
> (i) Open proxies allow network connections from anyone, to anywhere, on arbitrary ports and with arbitrary protocols.
> (ii) Open proxy servers allow a single individual to launch attacks or send traffic from multiple provider-diverse sources at the same time, thereby complicating the process of blocking spam (or firewalling an attack).
> (iii) Connections made via open proxies are often non-accountable, since the open proxy server may be doing no logging, or if logging **is** being done, logs may be unavailable to those investigating network incidents.
> (iv) Unlike spam sent via an open SMTP relay, spam sent via an open proxy server can be constructed so as to have arbitrary Received: headers, thereby inhibiting efforts at backtracking spam to its true origin. (http://www.uoregon.edu/~joe/open-proxies-used-to-send-spam.html)

At this point in the investigation, I had learned that attackers were able to locate a transparent proxy from outside the network. They had been able to do so even when it was "invisible" to the outside world and they had been able to exploit it for not only spam but for other anonymous connections as well. From reading the above www.uoregon.edu web page, I had leaned that open proxies are a huge problem contributing to not only spam but to other illicit activities. I had learned a lot about the potential for our network being used by others for their own purposes. It was apparent to me that we would have eventually appeared on the open proxy lists, which could have impacted our ability to send legitimate e-mail. All of these abuses and potential problems were serious, but the real significance of the problem occurred when I read an article about Adrian Lamo. The link to the article is provided on the page referenced above. Dr. St Sauver's page (from the University of Oregon, above) is an excellent resource for all kinds of information on open proxies. Many of the referenced links in this paper came to my attention directly or indirectly from this page.

The Lamo article appeared long before his recent arrest for the NY Times intrusion. Lamo has been credited with finding and exploiting security holes in the networks of other large companies. One of his favorite tactics is the exploitation of open proxy servers. The complete story is available at the following site: (http://www.technewsworld.com/perl/story/31548.html)

From reading the Lamo articles and seeing his methods, I began to wonder if such a method could be used against our network through our Content Engine. My earlier concern had been concentrated on stopping the use of our Internet bandwidth for connecting anonymously to other sites. Now here was a new concern: Could someone "see" inside our network by connecting through our transparent cache? This was a whole new level of potential problem that had not occurred to me prior to my research. To test the plausibility of someone actually breaking in via the proxy, I arranged a test with a co-worker.

I worked from home through a cable connection to the Internet. I set my web browser proxy settings to use our router address and port 443. Talking on the phone to my co-worker, I had him reconnect the cache to the network in the same configuration as before. I attempted to access various addresses behind our border router. These were addresses that were blocked by access control lists and invisible during port scans of the network. After only a couple of attempts I hit pay dirt. Attempting to connect to the switch at 192.168.0.40, I was presented with the login screen! The switch was running an http server for management purposes and open to connections from inside the border filtering router. Due to the router ACLs, it should have been impossible for anyone to connect from outside the network but the login screen was definitely that of the internal switch. Since I did not want to type the username and password over the Internet, I closed the connection and had my co-worker disconnect the cache. I had seen enough to know that we could not leave the current setup in place. Fortunately, the switch has a strong password. Imagine if it had been set to a default, simple or blank password. I had already seen that obscurity would not protect it. The attackers had found the cache, I'm sure they could have found the switch as well. A close look at the switch configuration and logs verified that it was unlikely that someone had already successfully accessed the switch but over time it is hard to be confident that the password alone would have been sufficient to protect it.

Accessing the switch has far reaching ramifications. The ability of someone to log into the switch has obvious and not so obvious security implications. Obviously, an attacker could modify the switch configuration to institute a denial of service attack. Shutting down ports and/or changing the passwords could break our Internet connection and make restoring it difficult. The not so obvious implications are due to the switch position within the network. True, there is a PIX firewall behind the switch which would make attacks against the inside LAN difficult but there is NO filter between the switch and the web server. The switch

could have easily been used as a “launching pad” for attempts at the web server. Since the traffic would originate from the zone between the router and PIX, none of the traffic would be logged. An attacker could present himself to the web server as a connection from the inside network.

Summary:

To fully understand how the cache was abused, you must realize that the existence of the VPN server or web server were not necessary for the successful exploitation of the cache vulnerability. In fact, both the VPN and web server could be shutdown and unplugged from the network without stopping the abuse. The only thing needed to exploit the cache is the existence of an access control list allowing access to the addresses and ports inside the network. **The internal devices needn’t even exist!**

There were a number of factors that combined to contribute to the weakness in the defense of the network:

1. The cache engine was installed and setup by a vendor using the default “out-of-the-box” configuration. The purpose of the cache was to improve user-perceived Internet access speed from inside the network. It was not clear that the router would redirect traffic to the cache via WCCP even if the request originated from the Internet. The company employees responsible for maintaining the network were not familiar with the internal workings of the cache and did not realize that it worked as a proxy server. They only knew that it used the WCCP protocol and that the border router blocked access to it. The assumption being that it was only accessible from inside the corporate network.

2. The VPN server was another “black box” highly rated for security. This server does not listen for port 80 (http) connections on its outside interface. Because of this, little need for blocking port 80 was apparent. Only after fully learning about transparent caching via WCCP does it become obvious that a listening web server is not essential to exploiting a caching vulnerability. The filtering router has no way of knowing if the VPN server actually listens to port 80, it is enough that such traffic is allowed by the router access control lists. Once the packets are allowed through the ACLs, WCCP redirects them to the cache and they never arrive at the VPN server. Since the cache does not have a valid copy of the requested resource, it retrieves it from the Internet regardless of whether the initial request comes from inside the corporate LAN or from out on the public Internet.

3. After applying the workaround described in the Cisco security bulletin, the cache stopped proxying one type of connection only to have the spammers discover a listening https server. (allowed in the new cache

ACL per the bulletin) The cache ACL after applying the workaround was simply:

```
https destination-port allow 443
https destination-port deny all
```

This allowed the attackers to funnel their requests through the cache by directing them at port 443. Both the filtering router and the cache permitted this port.

4. The switch at 192.168.0.40 was running a web server for administration. The fact that requests to port 80 on the switch were not allowed by the border router led us to believe that the switch http server was not reachable from the Internet. Since frequent administration of the switch is not required, defense-in-depth concepts require us to shut down this http server in case the border defenses are breached. This is an unnecessary service listening for connections. It is also vulnerable to potential attacks from inside the network.

5. The fact that the switch is behind the first layer of defense at the network border means that it has unrestricted access to other devices in the same zone between the border router and the PIX. This could have led to an attack from the switch on the web server at 192.168.0.36 or the VPN server at 192.168.9.43. Since the attack would be launched from behind the router and in front of the PIX, none of the attack traffic would pass through either device and no alarms would be raised and no access control list violations would be logged. Had the switch been compromised, an attacker could have altered the configuration or even locked us out of it.

6. The fact that the cache engine is not directly visible from the Internet did not prevent discovery and misuse. This is not the typical proxy server listening on well-known proxy ports such as 8080 or 3128. Even when the network was port scanned by both insiders with full knowledge of the network topology and by outside security specialists, the cache was not seen from the Internet. This did not stop spammers and others from finding it and using it for their own purposes.

7. CVE searches for vulnerabilities related to the VPN server returned no results. This helped to foster the false sense of security about allowing access to it from outside. It is true that the vulnerability exploited was not specifically related to the VPN server but ultimately it was the fact that it was a valid destination through the filtering router that led to misuse of the cache. Had the router dropped the packets at ingress to the network, the cache would never have received the requests and would not have relayed the connections to outside hosts.

8. Open proxies are not a new problem as referenced in articles earlier in this paper. Many companies have begun to check incoming connections to see if they are coming from an open proxy and reject those that are. A common, although questionable method is to do a reverse scan of the source address to check for open proxies running at the source. If a service is found to be listening on a well-known proxy server port such as 8080 or 3128 the connection is refused. For a very good (though lengthy) discussion of the problem of open proxies, look at: (http://www.uoregon.edu/~joe/proxies/open-proxy-problem.pdf).

9. In our case, scanning these well-known ports would reveal nothing yet we still had an open, exploitable proxy. Fortunately, our network never wound up on any open proxy list that I could find.

Preventative Measures:

To help improve our security posture, a small number of changes were made to the network:

- All outbound connections now travel through a filter that requires authentication. Only legitimate users can connect to the Internet and all access is logged.
- The router access controls for the VPN server have been changed to allow only VPN connections.
- The Content Engine has been removed from the network pending further testing.
- We are investigating a more robust Intrusion Detection/Prevention System.

Conclusions:

Through researching unusual traffic on our network, I was able to learn about a known but little publicized problem undermining the security of the Internet. Despite controlling access to the network with a double filtering scheme designed to make it very difficult for malicious traffic to enter the network, we were vulnerable to a number of potential security incidents. Traffic entering our network must pass through both a filtering router with firewall capabilities and intrusion detection and a PIX firewall. This paper described a method whereby the first line of defense was completely circumvented, in one case due to excessively open access (to the VPN server) and in the other by necessary access (the web server). Had the PIX not been in place, it is reasonable to assume that access to the internal network would have been possible or even trivial.

Even though access to the internal LAN was blocked by the firewall, there are risks associated with allowing your Internet connection to be hijacked by an

outsider. Since the traffic appeared to be coming from one of our IP addresses, we could have been held liable for attacks originating from outside and channeled through our cache. We also could have found our network added to a blacklist undermining our ability to send legitimate traffic. A smaller but significant side effect was the loss of bandwidth available for our own business purposes. Remember that the first hint of trouble was reports of slow Internet connections from our users.

In this example, there were cases where concepts of defense-in-depth prevented further exploitation and there were cases where a lack of defense-in-depth allowed the incident to occur. Some steps to help prevent any further problems are:

1. Never let a vendor connect a device to your network without a full understanding of exactly what that device does, how it is configured and how it operates. (cache)

2. Even if you believe a device to be safely hidden behind a firewall or filter, don't run unnecessary services on the device (switch in this case violated defense-in-depth) The strong password on the switch kept it temporarily protected. (successful defense-in-depth)

3. It is not absolutely necessary for a device to be "visible" for it to be found and exploited by unscrupulous people.

4. Never rely on only one method of access control. In our case, the PIX prevented further intrusion to the inside network. (successful defense-in-depth)

5. Monitor activity on systems even if you believe your security posture to be satisfactory. New methods of circumventing controls are discovered constantly. In our case, quickly removing the problem probably kept us off the blacklists. I'm sure it would have been a matter of time before we were listed on them.

6. Scans of our network did not identify this vulnerability. Count on attackers to have better tools than you AND your security vendor.

7. Since this traffic was destined for a valid address and port, the simple IDS in the router would not see it as anything to be concerned about. The web server logs show no evidence of the activity either. The connection was limited to the router, cache engine, the attacker and the final destination. The PIX never saw this traffic either. Only by monitoring the traffic passing outbound through the router showed any anomalies. Egress filtering may have helped mitigate the problem. The router access controls should be set to allow SMTP traffic only from

the mail server for example. Unfortunately, since the traffic is encapsulated within a GRE tunnel, it is not clear that ACLs would have blocked it. Further testing is needed.

8. Filtering access to the VPN server would have alleviated some of this problem. Even though there is no reason to believe the server itself is vulnerable to attack, the fact that it was accessible through the filter contributed to the problem. The only access required is UDP port 500 (isakmp) and IP type 50 (esp). There was no need to allow http access (port 80). Blocking the VPN server would not have stopped abuse of the web server address however.

9. Check into any reports from end users indicating unusual activity on your network. In this case, the complaints of slow Internet access were the first alert that something unusual was happening.

10. Even subscribing to multiple security mailing lists and checking security websites will not alert you to a potential problem if you are not completely aware of the operation of all devices on your network. A major security announcement about open proxy servers may not have caught our attention since we did not have a full understanding of the proxy nature of the WCCP cache. This problem, though not new, did not hit our "radar screen".

I have to wonder how many other transparent caches are configured in a similar fashion without the users being aware of it. Judging from the 900,000 open proxies on the aforementioned lists, probably quite a few.

References:

Symantec Corporation, Defense in Depth Benefits URL: http://securityresponse.symantec.com/avcenter/security/Content/security.articles/defense.in.depth.html (11/20/2003)

Nuclear Energy Institute, Plant Safety: Defense in Depth, URL: http://www.nei.org/index.asp?catnum=2&catid=55 (11/19/2003)

Internet Engineering Task Force - Network Working Group, Address Allocation for Private Internets URL: http://www.ietf.org/rfc/rfc1918.txt?number=1918 (11/25/2003)

Cisco Systems, Configuring Context-Based Access Control, URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fsecur_c/ftrafwl/scfcbac.htm#1000932 (11/01/2003)

Huston, Geoff, Web Caching The Internet Protocol Journal, URL: (http://www.cisco.com/warp/public/759/ipj_2-3/ipj_2-3_webcaching.html) (11/25/2003)

Cisco Systems, Cisco Security Advisory: Cisco Web Cache Control Protocol Router Vulnerability, May 13, 1998, URL: http://www.cisco.com/warp/public/770/wccpauth-pub.shtml (7/26/2003)

Cisco Systems, Cisco Security Advisory: Transparent Cache Engine and Content Engine TCP Relay Vulnerability, May 28, 2002, URL:(http://www.cisco.com/warp/public/707/transparentcache-tcp-relay-vuln-pub.shtml (11/25/2003)

CERT/CC – Carnegie Mellon University Software Engineering Institute, Vulnerabilities, Incidents and Fixes, 11/25/2003 URL: http://www.cert.org/nav/index_red.html (11/25/2003)

Open Source, The Ethereal Network Analyzer, 11/14/2003 URL: http://www.ethereal.com (11/20/2003)

CERT/CC – Carnegie Mellon University Software Engineering Institute, CERT/CC Vulnerability Note VU#150227 - Multiple vendors' HTTP proxy default configurations allow arbitrary TCP connections, 9/23/2003, URL: http://www.kb.cert.org/vuls/id/150227 (11/25/2003)

St Sauver, Dr. Joseph, Open Proxies Used to Send Spam, URL: http://www.uoregon.edu/~joe/open-proxies-used-to-send-spam.html) (11/25/2003)

Lyman, Jay, TechNewsWorld, 'Helpful Hacker' Adrian Lamo Faces Federal Charges, 9/10/2003, URL: http://www.technewsworld.com/perl/story/31548.html (11/15/2003)

St Sauver, Dr. Joseph, The Open Proxy Problem, Presentation to the Internet2 Members Meeting, 4/09/2003, URL: http://www.uoregon.edu/~joe/proxies/open-proxy-problem.pdf (11/25/2003)

Last Updated: July 28th, 2017

Upcoming Training

SANS Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UT	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Chicago 2017	Chicago, IL	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
Virginia Beach 2017 - SEC401: Security Essentials Bootcamp Style	Virginia Beach, VA	Aug 21, 2017 - Aug 26, 2017	vLive
Community SANS Pasadena SEC401 @ NASA	Pasadena, CA	Aug 23, 2017 - Aug 30, 2017	Community SANS
Mentor Session - SEC401	Minneapolis, MN	Aug 29, 2017 - Oct 10, 2017	Mentor
SANS San Francisco Fall 2017	San Francisco, CA	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS Tampa - Clearwater 2017	Clearwater, FL	Sep 05, 2017 - Sep 10, 2017	Live Event
Mentor Session - SEC401	Edmonton, AB	Sep 06, 2017 - Oct 18, 2017	Mentor
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
Mentor Session - SEC401	Ventura, CA	Sep 11, 2017 - Oct 12, 2017	Mentor
Community SANS Albany SEC401	Albany, NY	Sep 11, 2017 - Sep 16, 2017	Community SANS
Community SANS Dallas SEC401	Dallas, TX	Sep 18, 2017 - Sep 23, 2017	Community SANS
Community SANS Columbia SEC401	Columbia, MD	Sep 18, 2017 - Sep 23, 2017	Community SANS
SANS Copenhagen 2017	Copenhagen, Denmark	Sep 25, 2017 - Sep 30, 2017	Live Event
Community SANS Boise SEC401	Boise, ID	Sep 25, 2017 - Sep 30, 2017	Community SANS
Baltimore Fall 2017 - SEC401: Security Essentials Bootcamp Style	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	vLive
Community SANS New York SEC401	New York, NY	Sep 25, 2017 - Sep 30, 2017	Community SANS
Rocky Mountain Fall 2017	Denver, CO	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS London September 2017	London, United Kingdom	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS Baltimore Fall 2017	Baltimore, MD	Sep 25, 2017 - Sep 30, 2017	Live Event
SANS DFIR Prague 2017	Prague, Czech Republic	Oct 02, 2017 - Oct 08, 2017	Live Event
Community SANS Charleston SEC401	Charleston, SC	Oct 02, 2017 - Oct 07, 2017	Community SANS
Community SANS Sacramento SEC401	Sacramento, CA	Oct 02, 2017 - Oct 07, 2017	Community SANS
Mentor Session - SEC401	Arlington, VA	Oct 04, 2017 - Nov 15, 2017	Mentor
SANS October Singapore 2017	Singapore, Singapore	Oct 09, 2017 - Oct 28, 2017	Live Event
Community SANS Indianapolis SEC401	Indianapolis, IN	Oct 09, 2017 - Oct 14, 2017	Community SANS