Global Information Assurance Certification Paper

Kris Monroe
SANS Security Essentials
GSEC Practical Assignment, Version 1.4b, option2
December 30, 2003

Is Our Network Perimeter for our Web-based Application Vulnerable? A Case Study

Abstract

I was the senior member of the Network Support department for a software development company that had created a Web-based application. Network Support had rapidly configured the new servers, the network, application load balancers, and firewalls in order for the development team to go live quickly. While we invested a fair amount of time and attention on hardening the operating system and the firewalls, we came to a point where we asked the questions: Is our network perimeter vulnerable? How can we tell?

During my GSEC reading and Mentor led class, I learned about Network Security Scanners like Nessus[1]. Around the same time I learned of a free trial of QualysGuard by way of Network Solutions and later a SANS webcast. Surely these products would outline whether we had any vulnerabilities. Upon running my first Network Perimeter Scan with QualysGuard I was surprised at the number of medium to critical vulnerabilities listed. I immediately started researching how to patch these vulnerabilities using the solutions suggested by the QualysGuard scan results as well as from Web research. The scan results and the actions we took to resolve the outlined vulnerabilities are discussed below. We ran a follow-up scan to verify that our solutions and workarounds fixed the vulnerabilities. At that time I came to the conclusion that vulnerability scanners are a vital tool for those hosting Web-based applications, as well as for flushing out vulnerabilities in default network and software configurations.

Before Snapshot

The security issue presented in this paper is the investigation of the vulnerability of our network perimeter for our Web-based application.

There were no security policies in place for our company at the start of this venture. We had two objectives: to reduce the risk of our application's network perimeter being compromised, and to protect the integrity and confidentiality of our customers' data and communications.

Hardware and Software Environment

Our network environment consisted of F5 Networks Big-IP 1000 Application Switches for front-end load balancing, some Watchguard V-Class firewalls and Big-IP 5000 Application Switches for back-end application load balancing in a

[1] Nessus

configuration referred to as a “firewall sandwich” (load balancers as the bread and firewalls as the meat of the sandwich). See Figure 3 at the end of this paper for a network diagram that I created to show our environment. Only HTTP and HTTPS traffic is forwarded through to our application load balancers. Our SSL certificate was proxied by these same back-end application load balancers. Behind the application load balancers we had a number of IBM HTTP Servers with IBM WebSphere application server software and finally IBM DB2 database servers.

IBM HTTP Web Server settings were more or less in their default state.

The Load Balancers were initially configured by our Value Added Reseller (VAR). The Big-IP devices are stated to be very secure in their default configuration. The back-end application load balancers were configured to proxy SSL. This meant SSL was spoken over the Internet to the load balancers but the load balancers spoke HTTP to the Web servers.

Firewalls were configured to allow only HTTP and HTTPS Internet requests to pass through to the back-end Big-IPs.

During

I signed up for a free trial of QualysGuard, allowing us to scan the one IP address of our Web application.

The Qualys website gave the following description of QualysGuard:

> *QualysGuard is an on-demand security audit service delivered over the Web that enables organizations to effectively manage their vulnerabilities and maintain control over their network security with centralized reports and one-click links to verified remedies. QualysGuard provides comprehensive reports on vulnerabilities including severity levels, time to fix estimates and impact on business, plus trend analysis on security issues.*[2]

I asked for and received permission from our CEO/President to run vulnerability scans outside of normal business hours. This request would be the start for a future Security Policy regarding Vulnerability Scanning. I scheduled the scan for 8:00pm, well after the 5:00pm end of business day, in case the scan caused any problems. There was little chance that the scan would interrupt service to our customers (because we didn’t have any).

Upon return to work the next day I had email from Qualys stating that the scan had been run. I was prompted to log into the Qualysguard website to view the results. QualysGuard provided a detailed Scan Report as well as an option for an

[2] QualysGuard

Executive Report. The Executive Report summarized the scan results and included the total number of vulnerabilities found, an overall trend, number by severity, and vulnerabilities by severity over time (to hopefully show a reduction in vulnerabilities from when the first scan is run). I provided our CEO/President with the Executive Report to show that we indeed had some vulnerabilities and that we needed to take some actions to mitigate them.

Around this time I also installed and configured Nessus on a Linux server and ran vulnerability scans in order to compare against the QualysGuard results.

Vulnerability Scan Results

QualysGuard categorizes Vulnerabilities into severity levels from Severity 5, or Urgent, to Severity 1, or Minimal. Severity 1 items typically refer to information disclosure like Traceroute, WHOIS and Open TCP Services List. While you may be able to reduce the number of Severity 1's, there will always be some form of information disclosure if you are allowing the Internet to connect to one of your services, such as a Web server.

Our first scan with QualysGuard resulted in 1 Severity 4 (Critical), 5 Severity 3 (Serious), 2 Severity 2 (Medium), and 18 Severity 1 (Minimal) for a total of 26 items. Of those, 23 were designated as vulnerabilities.

Some duplicate vulnerabilities were reported, because both port 80 and 443 were allowed (one for each port). Taking these duplicates into account, the results work out to be 1 Severity 4 (Critical), 3 Severity 3 (Serious), 1 Severity 2 (Medium), and 16 Severity 1 (Minimal) for a total of 21 vulnerabilities.

QualysGuard also reported the vulnerabilities in 5 categories, those being Web server, TCP/IP, Information gathering, General remote services and Firewall. Output from our first scan is summarized in the charts shown in Figures 1 and 2.

Summary of Vulnerabilities

Vulnerabilities Total:	26	Security Risk:	4.0

by Severity

Severity	Vulnerabilities
5	0
4	1
3	5
2	2
1	18

5 Biggest Categories

Category	Vulnerabilities
Web server	13
TCP/IP	5
Information gathering	5
General remote services	2
Firewall	1

Figure 1: *QualysGuard Report - Summary of Vulnerabilities.*

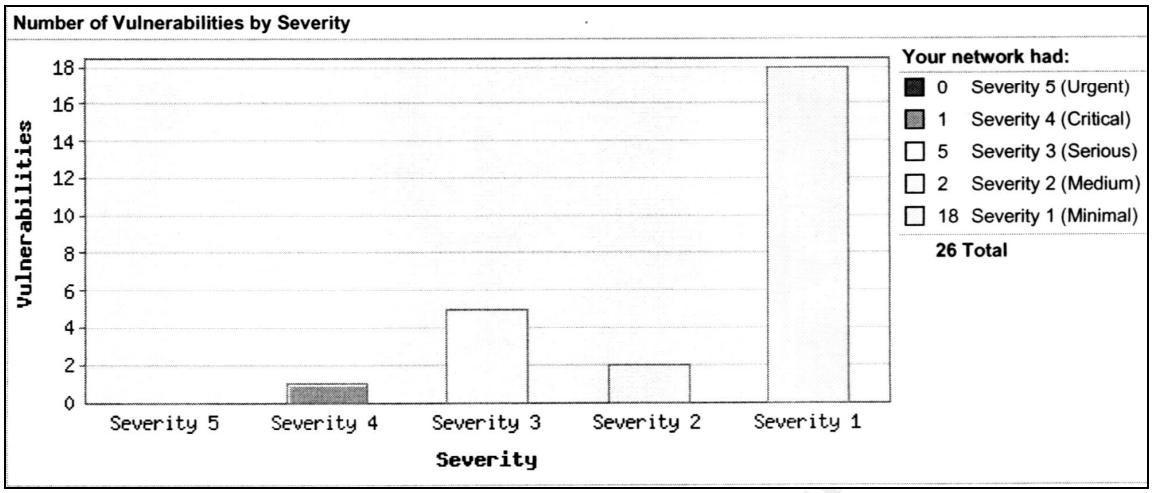

Figure 2: *QualysGuard Report - Number of Vulnerabilities by Severity*.

In the following sections, I address the vulnerabilities discovered, from greatest to least severe. I describe each vulnerability, evaluate the risk, and offer a solution (if any).

I discuss Severity 1 items selectively, as most are just information gathering results, such as output of Traceroute, WHOIS, and SSL Certificate Information.

Severity 4 (Critical) - SSL Server Has SSlv2 Enabled

Vulnerability

QualysGuard reported that SSLv2 was enabled and there are known flaws in the SSLv2 protocol.

I had not known of this vulnerability prior to this scan. SSLv2 was allowed by default by the Big-IP 5000 SSL Proxy, probably for backwards compatibility.

For examples of man-in-the-middle attacks see SSL Man-in-the-Middle Attacks http://www.sans.org/rr/paper.php?id=480.[3]

Risk

A man-in-the-middle attacker could force the communication to a less secure level at which point they could attempt to crack the weaker encryption and have a chance at reading secure communications (including passwords) or modifying messages/data.

A summary of SSLv2's security problems can be viewed in a report entitled Investigations about SSL at the following URL: http://www.eucybervote.org/Reports/MSI-WP2-D7V1-V1.0-02.htm [4]:

[3] Burkholder, Feb 1, 2002
[4] Investigations about SSL

Solution

Disable SSLv2. The fix for this is outlined in the Severity 3 (Severe) - SSL Server Supports Weak Encryption section, which begins on page 6.

Severity 3 (Severe) - Listing of Scripts in the Scripts Directory

Vulnerability

Files in our scripts directory were listable or browseable.

Risk

Anyone could acquire a list of the CGI scripts on our server by simply browsing the scripts directory. With this information an attacker could begin further attacks on susceptible scripts or gather important information by viewing the scripts themselves.

Solution

Disable directory listing of the scripts directory on the IBM HTTP Server

Our HTTP Server had a virtual directory for the CGI scripts directory, there was no index.html file in that directory, and indexing was not turned off, and therefore the directory was browseable. As an immediate quick fix a very basic index.html file was added to the scripts directory. If attackers tried to browse the scripts directory now, they simply received the index.html file. Later **Options –Indexes** was added to the IBM HTTP Server configuration file as well as in any virtual directory sections.

Severity 3 (Severe) - Web Server HTTP Trace Method Support Cross Site Tracing

Vulnerability

It was detected that our IBM HTTP Server supported the HTTP TRACE method.

Most HTTP Servers support this method as it is part of the HTTP specification and is intended for debugging and connection trace analysis.

HTTP Servers with this method in conjunction with various weaknesses in browsers may be subject to cross-site-scripting attacks.

Risk

This vulnerability could allow an attacker to trick an end user into revealing sensitive information that is intended only for our specific Web site.

The announcement of the discovery of this vulnerability may be viewed at the following URL:

http://www.whitehatsec.com/press_releases/WH-PR-20030120.pdf [5]

Further discussion on this vulnerability may be viewed at the following URL:
http://archives.neohapsis.com/archives/vulnwatch/2003-q1/0035.html [6]

Solution

Our IBM HTTP Web Server is based on Apache and has support for a Rewrite module that allows HTTP requests to be handled in a specific way. The QualysGuard report suggested we add the following lines for each virtual host in our configuration file:

```
RewriteEngine on
RewriteCond %{REQUEST_METHOD} ^TRACE
RewriteRule .* - [F]
```

The RewriteCond line outlined did not work in our particular environment. However, a Nessus report offered an alternative RewriteCond line that did work in our environment:

```
RewriteCond %{REQUEST_METHOD} ^(TRACE|TRACK)
```

After we added this line to our configuration file, QualysGuard no longer reported the vulnerability.

Severity 3 (Severe) - SSL Server Supports Weak Encryption

Vulnerability

It was reported that our SSL configuration supported weak or LOW encryption.

SSL encryption ciphers are classified based on encryption key length as follows:
HIGH – key length larger than 128 bits
MEDIUM – key length equal to 128 bits
LOW – key length smaller than 128 bits

Messages encrypted with LOW encryption ciphers are easy to decrypt.

Risk

This vulnerability could be exploited to decrypt secure communications without authorization.

Solution

Disable support for LOW encryption ciphers.
QualysGuard and Nessus both suggest the following configuration for Apache/mod_ssl:

[5] WhiteHat Security, Inc.
[6] Rain Forest Puppy

SSLProtocol: -ALL +SSLv3 +TLSv1
SSLCipherSuite: ALL:!aNULL:!eNULL:!LOW:!EXP:RC4+RSA:+HIGH:+MEDIUM

The recommended SSLProtocol setting above disables ALL protocols and then specifically includes only SSL v3 and TLS v1 ciphers.

The SSLCipherSuite setting above contains a cipher string preceded by !, -, or +. "!" permanently deletes a cipher from the list, "-" deletes a cipher from the list but allows it to be added again later, and "+" adds a cipher into the list and in the current location.

Analysis of the outlined cipher string permanently deletes aNull, eNULL, LOW and EXP. These ciphers are explained on the OpenSSL Documents Web site as follows:

> *aNull – the cipher suites offering no authentication. This is currently the anonymous DH algorithms. These cipher suites are vulnerable to a "man in the middle" attack and so their use is normally discouraged.*
>
> *eNULL – the "NULL" ciphers offering no encryption. Because these offer no encryption at all and are a security risk they are disabled unless explicitly included.*
>
> *LOW – "low" encryption cipher suites, currently those using 64 or 56 bit encryption algorithms but excluding export cipher suites.*
>
> *EXP – export encryption algorithms, including 40 and 56 bits algorithms.*[7]

Our SSL is proxied by our Big-IP 5000's, which use OpenSSL and not Apache/mod_ssl. I had to read the product manual on how to make these **SSLProtococol** and **SSLCipherSuite** changes on our load balancers, as attempts to modify the configuration files with a text editor failed. Page 7-41 of the Big-IP® Reference Guide[8] has a section entitled "Specifying SSL Ciphers". In this section the Reference Guide outlines that you can use the **bigpipe proxy** command or the Configuration utility to set SSL Ciphers. Using the Web-based Configuration utility, I navigated to our Proxy Advanced Properties page and checked the SSLv2 box under "Client-side Connections Do Not Use These SSL Versions". This disables SSLv2.

On page 7-65 of the Big-IP® Reference Guide there is a section entitled "Configuring invalid protocols". In this section the Reference Guide outlines that you can use the Configuration utility or the command line to specify invalid SSL protocols. On the same Proxy Advanced Properties page mentioned above for the disabling of SSLv2, I entered

[7] OpenSSL
[8] F5 Networks, p. 7-41, 7-65

ALL:!aNULL:!eNULL:!LOW:!EXP:RC4+RSA:+HIGH:+MEDIUM in the “Client Cipher List String” box.

Armed with the knowledge that the Big-IPs use OpenSSL, I signed up for and started watching for any vulnerabilities reported in the SANS Critical Vulnerability Analysis, the Security Alert Consensus, and similar email alerts. I also started scanning the IP addresses of the Big-IPs with Nessus, as these IPs were different than the Web-application IP that QualysGuard was configured to scan. From the time I started my GSEC class to the time of writing this paper, two OpenSSL vulnerabilities were reported. F5 Networks came out with two fix packs addressing each of the vulnerabilities. Because our Web application relied on SSL, I applied the fix packs within 24 hours of availability.

Severity 2 (Medium) - Web Directories Listable

Vulnerability

The Web server has some listable/browseable directories.

Risk

Like the Listing of Scripts in the Scripts Directory vulnerability above, anyone could acquire a list of the files on our server by simply browsing the vulnerable directories. With this information an attacker could begin further attacks by gathering important information by viewing the files themselves.

Solution

Disable directory browsing or listing for all directories.

To implement a quick fix for this vulnerability, I added an index.html file to all directories that did not already have one. To create a long-term solution, I added **Options –Indexes** to the IBM HTTP Server configuration file as well as in any virtual directory sections. This was implemented after testing in the development environment to ensure this directive would not break functionality in our Web application.

Severity 1 (Minimal) - Predictable IP ID Field

Vulnerability

It was reported that our host used non-random IP ID values. The next value of the ip_id field could easily be predicted.

Risk

This may be used for port scanning, or an attacker could attempt a denial of service attack, such as resetting the TCP/IP connection of a legitimate user.

Solution

No effective patches for the Windows platform exist. However, there are existing factors that mitigate this risk. A busy set of IBM HTTP Servers should generate enough traffic to make IP ID values difficult to determine. Big-IP load balancing also randomizes which HTTP Server gets contacted. WebSphere clustering load balances the application, which also randomizes which server responds. Session affinity is not retained, so any request to the application could go to any one of a number of HTTP/WebSphere servers.

Severity 1 (Minimal) - Apache Server Address Disclosure

Vulnerability

Our IBM HTTP Server was reported as having a vulnerability that could result in disclosing the server's address when the requested URL does not contain the trailing slash.

This vulnerability is dependant on the configuration of the server. If the **ServerName** directive is not configured to return the appropriate host name, or is configured to return the internal IP address, a remote user could reveal the internal IP or internal server name. Additionally, if the **UseCanonicalName** feature is turned on (which is the default), this will assist in the disclosure of internal host information.

Risk

Exploitation of this vulnerability could lead to the disclosure of information such as the server's internal network address. This information could aid in further attacks.

Solution

There is no patch at this time.

The workaround used was to comment out (turn off) both the Web server **ServerName** and **UseCanonicalName**.

With the **UseCanonical** setting off, the HTTP Server uses the hostname:port that the client supplied.

In our environment this workaround redirects http://www.domain.extension/images incorrectly to http://www.domain.extension:forwarded_port/images/, which fails because the forwarded port is available only between the load balancers and HTTP Servers and not through the Internet.

With this configuration the internal server name (and/or IP address) is not revealed.

Severity 1 (Minimal) - Apache Web Server ETag Header Information Disclosure Weakness

Vulnerability

It was reported that our IBM HTTP Server could disclose ETag header information that could include inode information.

Risk

With this vulnerability there is the risk that inode information could be disclosed and aid in providing information for additional attacks. Apache on Windows 2000 was stated as vulnerable on the Apache website. I could not find any example as to how this would affect IBM HTTP Server in a Windows 2000 environment. An example of an exploit that was given was that NFS uses inode numbers to generate file handles, but we weren't using NFS. A Webopdedia definition of inode states "inodes are data structures that contain information about files in Unix file systems"[9]. We also were not using Unix.

Solution

I added the "FileETag –Inode" in the IBM HTTP Server configuration files to be on the safe side. This vulnerability was no longer being reported as present once this was completed.

Severity 1 (Minimal) - Information Gathered - Web Server Version

Vulnerability

The full Web Server version was being displayed.

Results displayed:

Server Type:	Apache
Server Version:	IBM_HTTP_SERVER/1.3.19.4 Apache/1.3.20
Server Banner:	IBM_HTTP_SERVER/1.3.19.4 Apache/1.3.20

Risk

With this knowledge attackers could formulate an attack based upon known vulnerabilities in a particular HTTP Server version.

Solution

I changed ServerTokens Minimal to ServerTokens Prod.

With this change, HTTP Server will send Server: IBM_HTTP_SERVER rather than Server: IBM_HTTP_SERVER/1.3.19.4 Apache/1.3.20**.**

[9] Webopedia

A bogus server header can help you avoid tipping your hand as to what Web server you are really running. I was not able to find information on doing this with IBM HTTP Server. I did find a Newsgroup posting on the Web where Ajai Khattri modified source code and recompiled Apache 1.3.26 to set a bogus server header.[10] This is not something I felt comfortable trying on IBM HTTP Server. HTTP Server is already a modified version of Apache and there is tight integration between HTTP Server and WebSphere. To put it simply, I did not want to run the risk of breaking things just to change the server header.

Severity 1 (Minimal) - Information Gathered - Open TCP Services List

Vulnerability

Listing of Open TCP Services could be done using a port scanner.

QualysGuard reported the following two open TCP services:

> Port: 80
> IANA Assigned Ports/Services: www
> Service Detected: HTTP
>
> Port: 443
> IANA Assigned Ports/Services: HTTP protocol over TLS/SSL
> Service Detected: HTTP over SSL

Risk

Unauthorized users can exploit this information to test vulnerabilities in each of the open services.

Solution

Shut down any unknown or unused services.

Since we are serving up both HTTP (Port 80) and HTTPS (Port 443), we expected these ports to be reported as open. Note that by design these are the only two TCP Services available.

After Snapshot

After addressing all the vulnerabilities identified by the initial QualysGuard scan, we ran another scan to evaluate the effectiveness of our solutions and workarounds. This time, the scan reported zero vulnerabilities of severity level 5, 4, 3 and 2. The number of Severity 1 vulnerabilities was reduced from 18 to 15, including one duplicate item (again due to both ports 80 and 443 being used).

[10] Khattri, Sep 18, 2002

I was not 100% confident that Qualys found all the vulnerabilities we may have had with our Network Perimeter. To verify the results, I used Nessus in conjunction with QualysGuard. Nessus scans reported the same initial vulnerabilities found by QualysGuard, and confirmed that these vulnerabilities were eliminated after fixes were applied. This helped back up the belief that our overall risk was now reduced. A copy of the Nessus "after" report, as well as a QualysGuard Executive Report version, was given to our CEO/President to show the progress made.

Improved Security?
The changes that were made on the network perimeter resulted in:

Improved configuration of Big-IPs SSL settings
By making the SSL configuration changes to disable weak or LOW encryption and disabling the less secure SSLv2, we increased protection of our potential customers' confidentiality as well as the integrity of their data by reducing the chance of an attacker reading secure communications (including passwords) or modifying messages/data.

Improved configuration of IBM HTTP Server settings
By configuring the IBM HTTP Server to prevent browsing of the scripts and other directories, and by preventing HTTP Server address disclosure, we now reveal less information to potential attackers. By disabling the HTTP Trace method we helped protect our end users from revealing sensitive information that was intended only for our specific web site.

Added improvements of Security in general were:

Improved awareness and development of Security policy
Because of this exercise and the GSEC course, my department started working on Security policies regarding Vulnerability Scanning. Even though not directly related to the Network Perimeter for our Web-based application, other Security policies were started, including a strong password policy, P2P software policy, and more.

Improved security awareness by upper management
By providing the Executive Report version of the QualysGuard scan to our CEO/President, I was able to make management more aware of security issues and solutions. The Executive Report included simple graphs and charts showing that we had vulnerabilities and that we had reduced the number of these vulnerabilities over time.

Improved Vulnerability and Security awareness
By signing up for and reading vulnerability lists like SANS Critical Vulnerability Analysis, the Security Alert Consensus, and similar email alerts, I increased my awareness of vulnerabilities that might affect our environment.

Improved level of patching
The vulnerabilities revealed by the QualysGuard and Nessus scans alerted me to other vulnerabilities that could affect our equipment and software. When a vulnerability was present and a patch was available, I would schedule downtime in order to apply the patch in a timely fashion.

Summary

It is clear that vulnerability scanners are vital tools for those hosting Web-based applications, as well as for flushing out vulnerabilities in default network device and software configurations that are exposed to the Internet (such as our Big-IPs and IBM HTTP Servers). Had it not been for these scanning tools, I would not have known that we were vulnerable and our network perimeter would have remained at risk.

In our particular case, I recommended to upper management that we pay for the QualysGuard service after the free trial. Third party confirmation that our network perimeter was secure against compromise would be an excellent selling point for our application and as such could add to the Confidentiality, Integrity and Availability of not only our application but also of our potential customers' data. Unfortunately, upper management did not approve the expense, due in part to budgetary constraints. Future vulnerability scanning was continued using Nessus in order to keep a handle on vulnerabilities and to reduce the risk of our network perimeter being breached. Ideally, I would have liked to have used both QualysGuard and Nessus in case one tool found something overlooked by the other. I actually saved time in finding the solution for the HTTP Trace Method Support Cross Site Tracing vulnerability by using both tools rather than just QualysGuard. As I mentioned earlier QualysGuard suggested a rewrite condition that did not work in my environment whereas the Nessus one did.

As a Network Security professional I believe that using network scanning programs, be they commercial or open source, are mandatory tools in today's hostile networking environments.

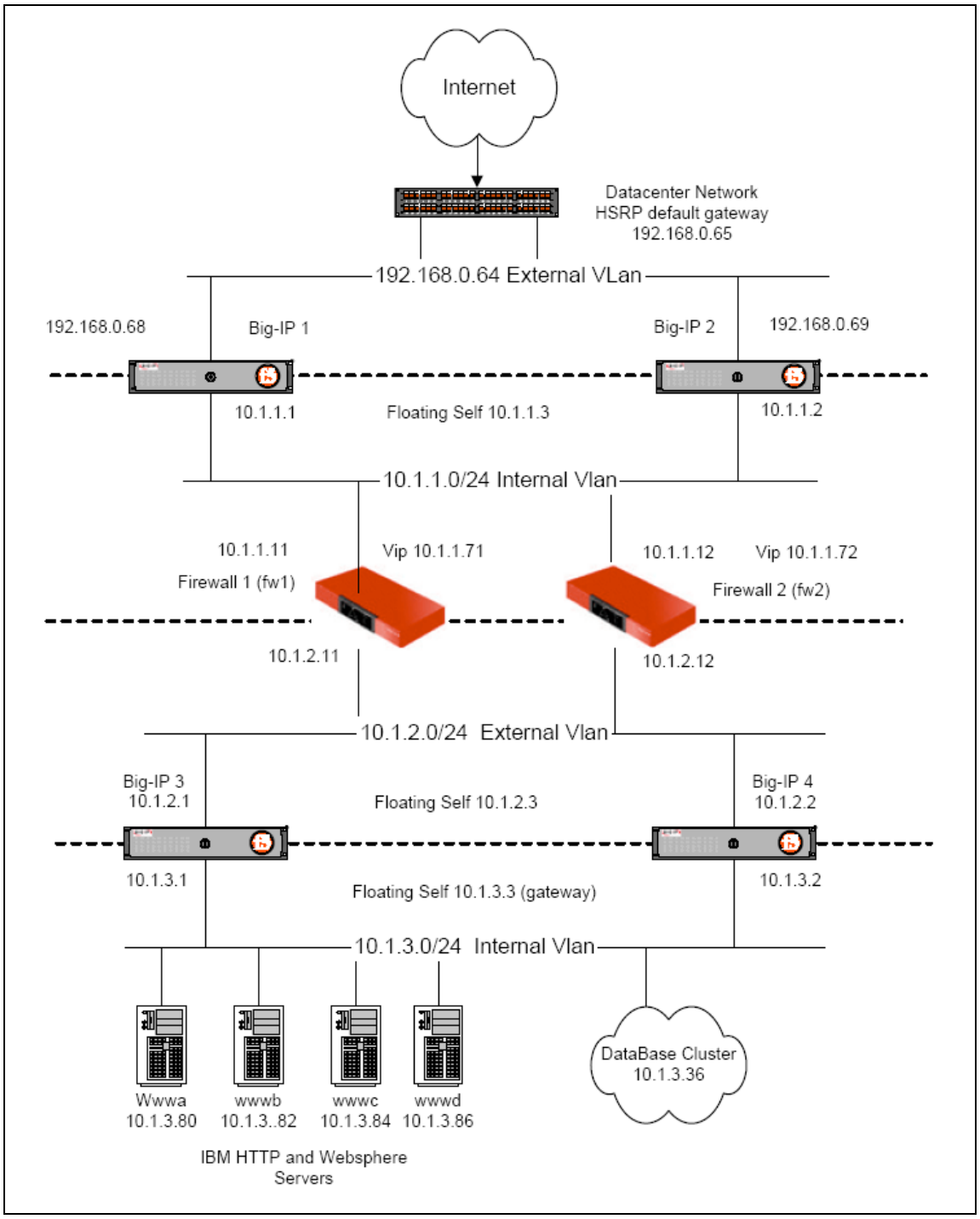

Figure 3: *Network Diagram*.

References

1. Nessus URL: http://www.nessus.org/
2. QualysGuard URL: http://www.qualys.com/?page=services/qg (November 1, 2003)
3. Burkholder, Peter. “SSL Man-in-the-Middle Attacks”. February 1, 2002. URL: http://www.sans.org/rr/paper.php?id=480
4. “Investigations about SSL”. URL: http://www.eucybervote.org/Reports/MSI-WP2-D7V1-V1.0-02.htm (November 9, 2003)
5. WhiteHat Security, Inc. “WhiteHat Discovers serious Security Flaw Affecting All Web Servers Worldwide”. January 20, 2003. URL: http://www.whitehatsec.com/press_releases/WH-PR-20030120.pdf
6. Rain Forest Puppy (rfp_at_vulnwatch.org), “administrivia: cross-site tracing”, Jan 22 2003. URL: http://archives.neohapsis.com/archives/vulnwatch/2003-q1/0035.html
7. OpenSSL Documents, Ciphers URL: http://www.openssl.org/docs/apps/ciphers.html
8. Big-IP® Reference Guide, version 4.5, F5 Networks, Inc., 2002. 7-41, 7-65
9. Webopedia - definition of *inode* URL: http://www.webopedia.com/TERM/i/inode.html

Newsgroup Posting

10. Ajai Khattri. “Re: Linux Slapper Worm” Online posting. Newsgroup muc.lists.bugtraq. September 18, 2002