# BIG-IP® New Features Guide for version 4.5 PTF-04

version 4.5 PTF-04

MAN-0077-00

# Legal Notices

## Copyright

Copyright 2000-2003, F5 Networks, Inc. All rights reserved.

F5 Networks, Inc. (F5) believes the information it furnishes to be accurate and reliable. However, F5 assumes no responsibility for the use of this information, nor any infringement of patents or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent, copyright, or other intellectual property right of F5 except as specifically described herein. F5 reserves the right to change specifications at any time without notice.

## Trademarks

F5, F5 Networks, the F5 logo, BIG-IP, 3-DNS, iControl, GLOBAL-SITE, SEE-IT, EDGE-FX, FireGuard, Internet Control Architecture, IP Application Switch, Packet Velocity, iRules, and SYN Check are registered trademarks or trademarks of F5 Networks, Inc. in the U.S. and certain other countries. All other product and company names are registered trademarks or trademarks of their respective holders. F5 trademarks may not be used in connection with any product or service except as permitted in writing by F5.

## Patents

This product protected by U.S. Patent 6,374,300; Pending U.S. Patent 20020040400. Other patents pending.

## Export Regulation Notice

This product may include cryptographic software. Under the Export Administration Act, the United States government may consider it a criminal offense to export this product from the United States.

## Export Warning

This is a Class A product. In a domestic environment this product may cause radio interference, in which case the user may be required to take adequate measures.

## FCC Compliance

This equipment generates, uses, and may emit radio frequency energy. The equipment has been type tested and found to comply with the limits for a Class A digital device pursuant to Part 15 of FCC rules, which are designed to provide reasonable protection against such radio frequency interference.

Operation of this equipment in a residential area may cause interference, in which case the user at his own expense will be required to take whatever measures may be required to correct the interference.

Any modifications to this device, unless expressly approved by the manufacturer, can void the user's authority to operate this equipment under part 15 of the FCC rules.

## Canadian Regulatory Compliance

This class A digital apparatus complies with Canadian I CES-003.

## Standards Compliance

The product conforms to ANSI/UL Std 1950 and Certified to CAN/CSA Std. C22.2 No. 950.

## Acknowledgments

This product includes software developed by the University of California, Berkeley and its contributors.

This product includes software developed by the Computer Systems Engineering Group at the Lawrence Berkeley Laboratory.

This product includes software developed by the NetBSD Foundation, Inc. and its contributors.

This product includes software developed by Christopher G. Demetriou for the NetBSD Project.

This product includes software developed by Adam Glass.

This product includes software developed by Christian E. Hopps.

This product includes software developed by Dean Huxley.

This product includes software developed by John Kohl.

This product includes software developed by Paul Kranenburg.

This product includes software developed by Terrence R. Lambert.

This product includes software developed by Philip A. Nelson.

This product includes software developed by Herb Peyerl.

This product includes software developed by Jochen Pohl for the NetBSD Project.

This product includes software developed by Chris Provenzano.

This product includes software developed by Theo de Raadt.

This product includes software developed by David Muir Sharnoff.

This product includes software developed by SigmaSoft, Th. Lockert.

This product includes software developed for the NetBSD Project by Jason R. Thorpe.

This product includes software developed by Jason R. Thorpe for And Communications, http://www.and.com.

This product includes software developed for the NetBSD Project by Frank Van der Linden.

This product includes software developed for the NetBSD Project by John M. Vinopal.

This product includes software developed by Christos Zoulas.

This product includes software developed by Charles Hannum.

This product includes software developed by Charles Hannum, by the University of Vermont and State Agricultural College and Garrett A. Wollman, by William F. Jolitz, and by the University of California, Berkeley, Lawrence Berkeley Laboratory, and its contributors.

This product includes software developed by the University of Vermont and State Agricultural College and Garrett A. Wollman.

In the following statement, "This software" refers to the Mitsumi CD-ROM driver:  This software was developed by Holger Veit and Brian Moore for use with "386BSD" and similar operating systems. "Similar operating systems" includes mainly non-profit oriented systems for research and education, including but not restricted to "NetBSD," "FreeBSD," "Mach" (by CMU).

In the following statement, "This software" refers to the parallel port driver:  This software is a component of "386BSD" developed by William F. Jolitz, TeleMuse.

This product includes software developed by the Apache Group for use in the Apache HTTP server project (http://www.apache.org/).

This product includes software developed by Darren Reed. (© 1993-1998 by Darren Reed).

This product includes software licensed from Richard H. Porter under the GNU Library General Public License (© 1998, Red Hat Software),  www.gnu.org/copyleft/lgpl.html.

This product includes the standard version of Perl software licensed under the Perl Artistic License (© 1997, 1998 Tom Christiansen and Nathan Torkington). All rights reserved. You may find the most current standard version of Perl at http://www.perl.com.

# Table of Contents

## 1

## Introduction to this Guide

## 2

## Online Certificate Status Protocol for the BIG-IP System

## 3

## Certificate Header Format in Client Requests

## 4

## Preventing SYN Flood Attacks

## 5

## Logging Hardware Failures

1

# Introduction to this Guide

# What’s new in BIG-IP version 4.5 PTF-04

The BIG-IP version 4.5 PTF-04 release includes a number of new features. Some of these features enhance the security of your BIG-IP system, by improving user authentication and thwarting denial-of-service attacks. Other features enhance system performance and ease the task of system administration.

The new features of this release are documented either in this guide or in the release note for BIG-IP version 4.5 PTF-04. For a complete list of the new features in this release, see the release note.

The new features documented in this guide are:

- **Online Certificate Status Protocol for the BIG-IP system**
  A significant feature in this release is support for the Online Certificate Status Protocol (OCSP). ***OCSP*** provides an alternative to a certificate revocation list (CRL), which is used during certificate verification to determine whether an SSL certificate presented by a client has been revoked. Because CRLs are updated only at regular intervals, the information in a CRL can sometimes be outdated at the time that it is checked. Using OCSP instead of a CRL eliminates this problem by ensuring that the revocation status of a client certificate is always current. For more information, see Chapter 2, *Online Certificate Status Protocol for the BIG-IP System*.

- **New format for the SSLClientCertSerialNumber header**
  Another enhancement to the SSL proxy, this change to the **SSLCLientCertSerialNumber** header gives users who write rules based on certificate serial numbers the ability to write to a consistent format, regardless of the length of the serial number. For more information, see Chapter 3, *Certificate Header Format in Client Requests*.

- **SYN Check**
  The new SYN Check feature mitigates a particular type of denial-of-service attack known as a SYN flood. A ***SYN flood*** is an attack against a system for the purpose of exhausting that system’s resources. For more information, see Chapter 4, *Preventing SYN Flood Attacks*.

- The **system_check** script
  The **system_check** script is useful for displaying and logging hardware failures. For more information, see Chapter 5, *Logging Hardware Failures*.

# Using this guide

Before using this guide, it is helpful to understand how the guide relates to other BIG-IP documentation. It is also helpful to understand the stylistic conventions that appear throughout the text.

## Scope of this guide

This guide documents only those new features that are included in the BIG-IP version 4.5 PTF-04 release. You should therefore use this guide in confunction with the complete set of product documentation that applies to the BIG-IP version 4.5 release.

The BIG-IP version 4.5 documentation set comprises these documents:

- **Platform Guide**
  This guide includes information about the BIG-IP unit. It also contains important environmental warnings.
- **BIG-IP Solutions Guide**
  This guide provides examples of common load balancing solutions.
- **BIG-IP Reference Guide**
  This guide provides detailed configuration information for the BIG-IP system. It also provides syntax information for **bigpipe** commands, other command line utilities, configuration files, system utilities, and monitoring and administration information.
- **Link Controller Solutions Guide**
  This guide provides examples of common link load balancing solutions using the Link Controller.
- **BIG-IP e-Commerce Guide (optional)**
  This guide provides detailed configuration information for BIG-IP e-Commerce Controller systems.
- **Release notes**
  Release notes for BIG-IP version 4.5 are available from the product web server home page, and are also available on the technical support site. The release notes contain the latest information for BIG-IP version 4.5, including a list of new features and enhancements, a list of fixes, and, in some cases, a list of known issues.
- **Online help**
  You can find help online in three different locations:
  - The web server on the product has PDF versions of the guides included in the Administrator Kit.
  - The web-based Configuration utility has online help for each screen. Simply click the **Help** button.

- Individual **bigpipe** commands have online help, including command syntax and examples, in standard UNIX man page format. Simply type the command followed by the word **help**, and the BIG-IP system displays the syntax and usage associated with the command.

# Stylistic conventions

To help you easily identify and understand important information, our documentation uses the stylistic conventions described below.

## Identifying new terms

To help you identify sections where a term is defined, the term itself is shown in bold italic text. For example, a ***virtual server*** is a specific combination of a virtual address and virtual port, associated with a content site that is managed by a BIG-IP system or other type of host server.

## Identifying references to objects, names, and commands

We apply bold text to a variety of items to help you easily pick them out of a block of text. These items include web addresses, IP addresses, utility names, and portions of commands, such as variables and keywords. For example, with the **bigpipe pool <pool_name> show** command, you can specify a specific pool to show by specifying a pool name for the **<pool_name>** variable.

## Identifying references to other documents

We use italic text to denote a reference to another document. In references where we provide the name of a book as well as a specific chapter or section in the book, we show the book name in bold, italic text, and the chapter/section name in italic text to help quickly differentiate the two. For example, you can find information about load balancing methods in the ***BIG-IP Reference Guide***, Chapter *4, Pools.*

## Identifying command syntax

We show complete commands in bold Courier text. Note that we do not include the corresponding screen prompt, unless the command is shown in a figure that depicts an entire command line screen. For example, the following command shows the configuration of the specified pool name:

```
bigpipe pool <pool_name> show
```

or

```
b pool <pool_name> show
```

Table 1.1 explains additional special conventions used in command line syntax.

| Item in text | Description |
|---|---|
| \ | Indicates that the command continues on the following line, and that users should type the entire command without typing a line break. |
| < > | Identifies a user-defined parameter. For example, if the command has **<your name>**, type in your name, but do not include the brackets. |
| \| | Separates parts of a command. |
| [ ] | Indicates that syntax inside the brackets is optional. |
| ... | Indicates that you can type a series of items. |

***Table 1.1*** *Command line syntax conventions*

2

# Online Certificate Status Protocol for the BIG-IP System

# Introducing OCSP

***Online Certificate Status Protocol (OCSP)*** is an industry-standard protocol that offers an alternative to a certificate revocation list (CRL) when using public-key technology. A ***CRL*** is a list of revoked client certificates, which a server system can check during the process of verifying a client certificate.

With this release, the BIG-IP system supports both CRLs and the OCSP protocol, for use with its SSL proxy feature.

For more information on CRLs, see the ***BIG-IP Reference Guide***.

## The limitations of CRLs

When presented with a client certificate, the BIG-IP system sometimes needs to assess the revocation state of that certificate before accepting the certificate and forwarding the connection to a target server. The standard method of assessing revocation status is a CRL, which is stored in a separate CRL file on each machine in your configuration. Although CRLs are considered to be a standard way of checking revocation status of SSL certificates, a CRL is updated only at fixed intervals, thus presenting a risk that the information in the CRL is outdated at the time that the status check occurs.

Also, having to store a separate CRL file on each machine presents other limitations:

- All CRL files must be kept in sync.
- Having a separate CRL file on each machine poses a security risk.
- Multiple CRL files cannot be administered from a central location.

## The benefits of OCSP

OCSP ensures that the BIG-IP system always obtains real-time revocation status during the certificate verification process.

OCSP is based on a client/server model. A BIG-IP SSL proxy acts as the OCSP client, and the OCSP server runs on an external system. (The OCSP server is a third-party software application and is therefore not included in the BIG-IP system.)

The external system, known as an OCSP ***responder***, sends certificate revocation status to the SSL proxy. Until the SSL proxy receives this status from the OCSP responder, the proxy blocks the connection. If the OCSP responder rejects the certificate, the proxy denies the connection.

Figure 2.1 shows a basic OCSP implementation in a BIG-IP system configuration.

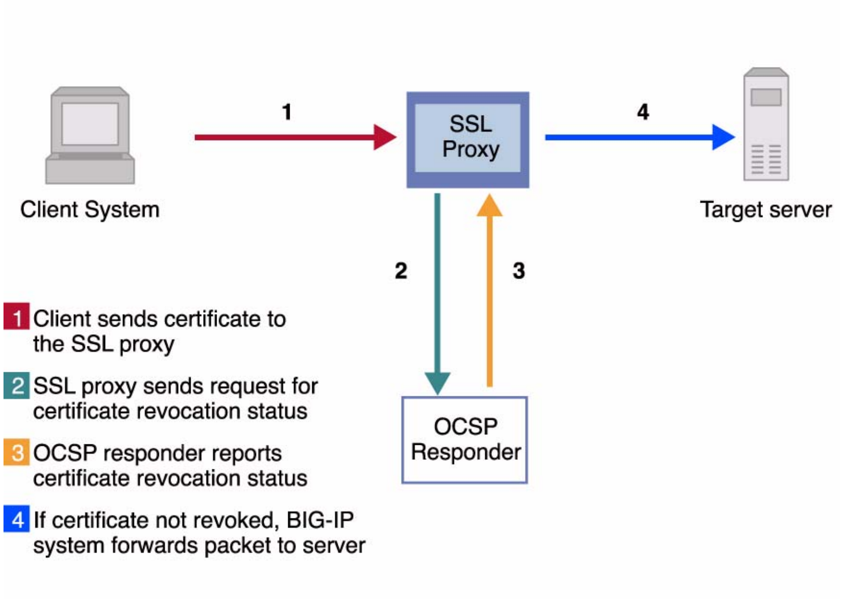
1
SSL Proxy
4
Client System
Target server
2
3
OCSP Responder
1 Client sends certificate to the SSL proxy
2 SSL proxy sends request for certificate revocation status
3 OCSP responder reports certificate revocation status
4 If certificate not revoked, BIG-IP system forwards packet to server

***Figure 2.1*** *A basic OCSP configuration*

# How does OCSP on the BIG-IP system work?

Before using OCSP, you must configure the OCSP client and the OCSP responder to work together. Configuring OCSP means creating a responder definition on the BIG-IP system, and then associating that responder definition with one or more SSL proxies. A ***responder definition*** is a set of data and instructions on the BIG-IP system that the corresponding responder needs when servicing an OCSP client request.

When a client sends a certificate to the BIG-IP system, the SSL proxy first checks that the signer of the certificate is listed in the trusted CAs file. If the certificate is listed, the BIG-IP system then checks to see if the certificate has been revoked. Without OCSP, the BIG-IP system can check revocation status by reading the certificate revocation list (CRL), if the CRL option is configured on the SSL proxy. With OCSP, however, the BIG-IP system bypasses the CRL and sends a revocation status request to the appropriate OCSP responder.

The BIG-IP system chooses the OCSP responder by checking the CA specified in the **Issuer** field of the original client certificate. The BIG-IP system then attempts to match that CA with a CA listed in the responder definitions associated with that SSL proxy. The responder definition that the BIG-IP system uses is a definition listed in the **OCSP responder list** option of the SSL proxy.

If a match exists, the BIG-IP system checks the target URL within the client certificate's **AuthorityInfoAccess** (**AIA**) field, if the field exists. The BIG-IP system then uses that URL, unless the **Ignore AIA** parameter is enabled within the responder definition. In this case, the BIG-IP system uses the URL specified in the **Responder URL** parameter of the matching responder definition to send the request for certificate revocation status.

If no match exists, the BIG-IP system checks the **Responder CAs** parameter of another responder defined in the SSL proxy. If all responder definitions are checked and no match is found, the certificate verification fails and the BIG-IP system denies the client request.

# Configuring OCSP

Using the BIG-IP system, you can create OCSP responder definitions that correspond to various responders. The BIG-IP system stores responder definitions in the **bigip.conf** file. Figure 2.2 shows an example of a responder definition.

```
responder my_responder {
    url "http://192.168.103.155:8080/"
    calist file /config/bigconfig/ssl.crt/cacerts.crt
    respcert file /config/bigconfig/ssl.crt/VAfile.crt
    signcert /config/bigconfig/ssl.crt/sign.crt
    signkey /config/bigconfig/ssl.key/sign.key
    req sign digest sha1
    req certid digest sha1
    ignore aia enable
    trust signer disable
    valperiod 120
}
```

***Figure 2.2*** *Sample responder definition in the* ***bigip.conf*** *file*

A single responder definition can target a specific responder, or multiple responder definitions can target the same responder. The only unique attribute is the responder name. Each responder itself is associated with a certificate authority (CA), and multiple responders can be associated with the same CA.

Figure 2.3 shows this scenario, where responders **1** and **2** are associated with CA **1**, but responder **3** is associated with CA **2**.

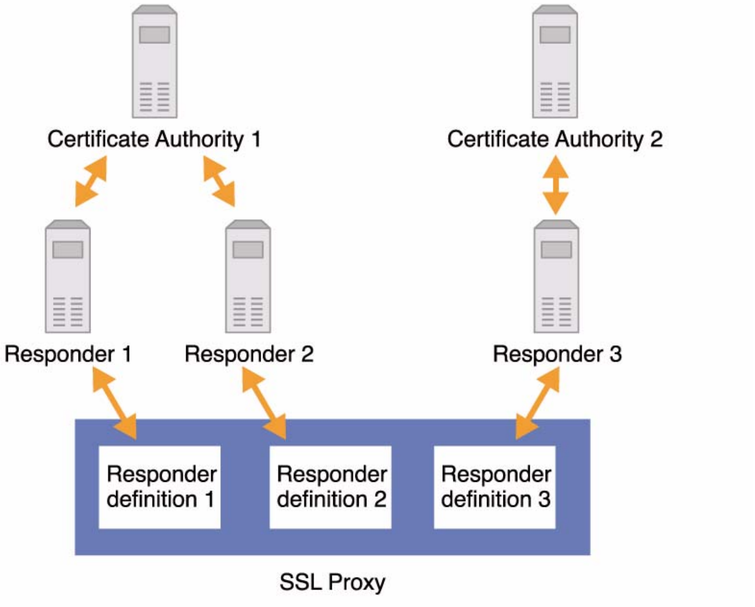

***Figure 2.3*** *Relationship of responders to CAs*

Once you have created all of your responder definitions, you can configure an SSL proxy to use one or more specific responder definitions when processing requests over SSL.

There are two tasks required for configuring OCSP. The first task is to create the OCSP responder definition. The second task is to configure the SSL proxy's **OCSP responder list** option. This option allows you to specify the OCSP responder definitions that the proxy will use to obtain revocation status.

## Creating an OCSP responder definition

To fully implement a responder definition on the BIG-IP system, you must define a set of parameter values. You configure these parameters through the bigpipe responder command. Table 2.1 lists and describes the configurable parameters for a responder definition.

| Parameter | Description | Keyword | Data type | Valid values /range | Default /Initial value |
|---|---|---|---|---|---|
| Responder name | A name that identifies a responder definition.<br>This parameter is required. | responder | String | N/A | Empty string |
| Responder CAs | An X509 store containing the certificates of the certificate authorities (CAs) that are to be serviced by this particular responder. The CAs in this store must match the issuer of the certificate currently being validated with OCSP. The match is determined by inspecting the **subject** field of the issuing certificate. You populate this store by specifying a bundle-type **.crt** file, which contains all necessary CA certificates.<br>This parameter is required. | calist file | String (file name) | N/A | Empty string |
| Responder URL | The URL used to contact the OCSP service on the responder.<br>This parameter is required. | url | String (valid URL) | Valid URL format | Empty string |
| Validity period | A number of seconds that is used to specify an acceptable error range. This parameter is used when the OCSP responder clock and a client clock are not synchronized, which could cause a certificate status check to fail. This value must be a positive number.<br>This parameter is required. | valperiod | Integer (seconds) | 0 to MAX | 300 |

***Table 2.1*** *Configurable OCSP responder parameters*

| Parameter | Description | Keyword | Data type | Valid values /range | Default /Initial value |
|---|---|---|---|---|---|
| Responder certificate | A certificate that verifies the signature of the response from the responder. This parameter is needed in the event that the responder is not covered by the certificates already loaded into the responder's CA store.<br>This parameter is optional. | respcert file | String (file name) | N/A | Empty string |
| Request signing certificate and request signing key | A certificate and key used to sign an OCSP request.<br>Special meanings:<br>If the certificate is specified but the key is not specified, then the private key is read from the same file as the certificate.<br>If neither the certificate nor the key is specified, then the request is not signed.<br>If the certificate is not specified and the key is specified, then the configuration is considered to be invalid.<br>This parameter is optional. | sign cert and sign key | Strings (file names) | N/A | Empty string |
| Request signing digest algorithm | The algorithm for signing the request, using the signing certificate and key.<br>Special meanings:<br>This parameter has no meaning if request signing is not in effect (that is, both the **request signing certificate** and **request signing key** parameters are empty).<br>This parameter is required only when request signing is in effect. | req sign digest | String | md5 \| sha1 | sha1 |
| Request CertID digest algorithm | The algorithm for hashing the certificate information used to create the certificate ID that is sent to the responder.<br>This parameter is optional. | req certid digest | String | md5 \| sha1 | sha1 |
| Ignore AIA | An instruction to ignore the URL contained in the certificate's **AIA** fields and to always use the URL specified by the responder instead,<br>Special meanings:<br>If not defined, this value is assumed to be zero (**0**).<br>This parameter is optional. | ignore aia | String | enable \| disable | disable |

***Table 2.1*** *Configurable OCSP responder parameters*

| Parameter | Description | Keyword | Data type | Valid values /range | Default /Initial value |
|---|---|---|---|---|---|
| Explicit trust of the response signer | An instruction to:<br>Search the SSL proxy's list of trusted CAs for the certificate used to sign the response.<br>Refrain from constructing a chain.<br>Special meanings:<br>If not defined, this value is assumed to be zero (**0**).<br>This parameter is optional. | trust signer | String | enable \| disable | disable |

***Table 2.1*** *Configurable OCSP responder parameters*

### To create a responder definition

Use the following **bigpipe responder** command syntax to create an OCSP responder definition.

```
b responder <name> [calist file <filename>]
     [url <url>]
     [valperiod <number>]
     [respcert file <filename>]
     [signcert file <filename>]
     [signkey file <filename>]
     [req sign digest (sha1 | md5)]
     [req certid digest (sha1 | md5)]
     [ignore aia (enable | disable)]
     [trust signer (enable | disable)]
```

### To view responder definition parameters

Use the following commands, specifying a responder name, to show responder parameter values.

```
b responder <name> calist file [show]
b responder <name> url [show]
b responder <name> valperiod [show]
b responder <name> respcert file [show]
b responder <name> signcert file [show]
b responder <name> signkey file [show]
b responder <name> req sign digest [show]
b responder <name> req certid digest [show]
b responder <name> ignore aia [show]
b responder <name> trust signer [show]
```

# Configuring the SSL proxy

Once you have created one or more OCSP responder definitions, you need to specify which responder definition the SSL proxy should use when it responds to the BIG-IP system's request for certificate revocation status. You do this by using the **bigpipe proxy** command to configure the **OCSP responder list** option.

◆ **Important**

*If the **OCSP responder list** option is not configured on the SSL proxy, then the certificate is automatically validated.*

Table 2.2 describes the **OCSP responder list** option.

| SSL proxy option | Description | Keyword | Data type | Valid values /range | Default /initial value |
|---|---|---|---|---|---|
| OCSP responder list | A list of OCSP responder definitions.<br>Special meanings:<br>If one or more responders are listed, then OCSP validation is enabled.<br>This parameter is optional. | ocsp responders | String (space-delimited list) | N/A | Empty string |

***Table 2.2*** *Configurable SSL proxy options for OCSP validation*

After you have configured the **OCSP responder list** option, and the BIG-IP system has received certificate revocation status from a responder, the SSL proxy inserts a certificate status header into the original client request. Note that the SSL proxy only inserts this header when previously configured to insert headers into client requests. (For more information on configuring this option, see the ***BIG-IP Reference Guide***).

The name of the certificate status header is **SSLClientCertificateStatus**. Like other certificate-related headers that the proxy inserts into a request, the **SSLClientCertificateStatus** header is most useful when the proxy is configured to request, but not require, certificates. For more information on configuring the SSL proxy, see the ***BIG-IP Reference Guide***.

### To configure the SSL proxy for OCSP validation

Use the following **bigpipe proxy** command syntax to configure the **OCSP responder** list option.

```
b proxy <ip addr>:<service> [ocsp responders <list of responders>]
```

### To display a list of existing OCSP responders

Use the following **bigpipe proxy** command syntax to display a list of existing OCSP responders.

```
proxy <ip addr>:<service> ocsp responders [show]
```

◆ **Note**

*The BIG-IP system allows you to enable both the CRL and the OCSP options on the SSL proxy. Most users need to enable either one or the other, but not both. However, in the rare case that you want to enable both options, be aware that both the search of the CRL file and the connection to the responder must be successful. Otherwise, the BIG-IP system fails to obtain status.*

3

# Certificate Header Format in Client Requests

- The SSLClientCertSerialNumber header format

# The SSLClientCertSerialNumber header format

One of the options available for configuring an SSL proxy is the ability to insert headers into HTTPS client requests. Some headers correspond to a field of a client certificate, such as certificate status, version, issuer, and signature algorithm. Once the SSL proxy has inserted these headers, you can create a rule that load balances traffic based on the value of these headers.

One of these headers is the **SSLClientCertSerialNumber** header. In previous releases of the BIG-IP system, if the value of the **SSLClientCertSerialNumber** header was less than or equal to four bytes, the BIG-IP system displayed that value in decimal format. Any value greater than four bytes was displayed in hexidecimal format.

This inconsistency in format has been removed so that users writing rules to balance traffic based on a client certificate's serial number can write to a consistent format, regardless of the length of the serial number.

The format of the **SSLClientCertSerialNumber** header, when inserted into a client request, now has the following syntax, where **hh** is a two-digit hexidecimal number:

```
SSLClientCertSerialNumber: [(Negative)] hh[:hh]*
```

Thus, the serial number in the header contains two lower-case hexidecimal digits (**0** to **f**), which represent each byte of the serial number. Each byte is separated by a colon (**:**). The following are examples of headers in this format:

- **SSLClientCertSerialNumber: 10**
  This hexidecimal value represents the decimal number 16.
- **SSLClientCertSerialNumber: 20:0b:3d**
  This hexidecimal value represents the decimal number 2,100,029.

If, for some reason, the incoming serial number is explicitly encoded as a negative value, the string **(Negative)** appears before the serial number. For example:

- **SSLClientCertSerialNumber: (Negative) 01**
  This hexidecimal value represents the decimal number -1.

# 4

# Preventing SYN Flood Attacks

# Introducing SYN Check

Before you read about the SYN Check$^{TM}$ feature, it is helpful to understand the type of denial-of-service attack known as a SYN flood.

## Understanding SYN flood attacks

A ***SYN flood*** is an attack against a system for the purpose of exhausting that system's resources. An attacker launching a SYN flood against a target system attempts to occupy all available resources used to establish TCP connections by sending multiple SYN segments containing incorrect IP addresses. Note that the term ***SYN*** refers to a type of connection state that occurs during establishment of a TCP/IP connection.

More specifically, a SYN flood is designed to fill up a SYN queue. A ***SYN queue*** is a set of connections stored in the connection table in the SYN-RECEIVED state, as part of the standard three-way TCP handshake. A SYN queue can hold a maximum number of connections in the SYN-RECEIVED state.

Connections in the SYN-RECEIVED state are considered to be half-open and waiting for an acknowledgement from the client. When a SYN flood causes the maximum number of allowed connections in the SYN-RECEIVED state to be reached, the SYN queue is said to be full, thus preventing the target system from establishing other legitimate connections. A full SYN queue therefore results in partially-open TCP connections to IP addresses that either do not exist or are unreachable. In these cases, the connections must reach their timeout before the server can continue fulfilling other requests.

## Alleviating SYN flooding

The BIG-IP system includes a feature designed to alleviate SYN flooding. Known as ***SYN Check***, this feature sends information about the flow, in the form of cookies, to the requesting client, so that the SYN-RECEIVED state normally stored in the connection table for the initiated session does not need to be kept. Because the SYN-RECEIVED state is not kept for a connection, the SYN queue cannot be exhausted, and normal TCP communication can continue.

The SYN Check feature complements the existing adaptive reaper feature in the BIG-IP system. While the adaptive reaper handles established connection flooding, SYN Check prevents connection flooding altogether. That is, while the adaptive reaper must work overtime to flush connections, the SYN Check feature prevents the SYN queue from becoming full, thus allowing the target sysem to continue to establish TCP connections.

# Configuring SYN Check activation

To activate the SYN Check feature, you can configure two basic settings: a setting at the global level, and a setting for each virtual server. These values are the global variable **syncookie_threshold** and the virtual server attribute **syncookie_threshold**. The threshold numbers that you assign to the global variable and the virtual server attribute specify the number of concurrent connections that can be made to one or more virtual servers prior to the SYN Check feature being activated.

You can configure either a global threshold or a virtual server threshold, or both. When both the global and virtual server thresholds are defined, the lower threshold always takes precedence. Thus, if a virtual server's SYN Check threshold is set to **10,000**, and the global threshold is set to **2,000**, the BIG-IP system generates SYN cookies when 2,000 concurrent connections have been established.

## Activating SYN Check based on a connection threshold

You can configure the BIG-IP system to activate the SYN Check feature when some threshold of connections has been reached on one or all virtual servers. To set the threshold on an individual virtual server, you use the **bigpipe virtual** command. To set the threshold on all virtual servers, you use the **bigpipe global** command.

The default global value for the SYN Check threshold is **150,000**. For individual virtual servers, SYN Check thresholds are disabled by default.

For example, to set the SYN Check threshold on a global basis to 300,000 concurrent connections, you type the following command:

```
bigpipe global syncookie_threshold 300000
```

Running this command causes the BIG-IP system to activate SYN Check when the number of concurrent connections on all virtual servers exceeds 300,000.

To set the SYN Check threshold on an individual virtual server to 10,000 concurrent connections, you type the following command:

```
bigpipe virtual 10.1.2.3:80 syncookie_threshold 10000
```

Typing this command causes the BIG-IP system to activate SYN Check when the number of concurrent connections for that virtual server exceeds 10,000.

## Activating SYN Check for all connections

Rather than activating SYN Check by defining a specific threshold of connections, you can configure the BIG-IP system to activate SYN Check for all connections, for an individual virtual server only.

You do this by setting the virtual server SYN Check threshold to **0**. For example:

```
bigpipe virtual 10.1.2.3:80 syncookie_threshold 0
```

By setting the SYN Check threshold on virtual server **10.1.2.3:80** to **0**, you effectively cause the BIG-IP system to always send SYN cookies after the very first connection is established.

◆ **Tip**

*While activating SYN Check for all concurrent connections seems to have its advantages, system performance is improved if you define a threshold other than **0**. This is because when SYN Check is activated, the client, in response to the target system, is required to return the SYN cookie, incremented by 1, as the acknowledgement number in the ACK flag. This causes some latency that can be avoided if you set a threshold of connections that is considerably higher than **0**.*

5

# Logging Hardware Failures

# The system_check script

The BIG-IP system includes a script, called **system_check**, which monitors certain hardware components and notifies the user of their status. The hardware components that the **system_check** script monitors are:

- Chassis fan speed
- CPU fan speed (one or two)
- Power supply (for platforms that include redundant power supplies)

It is the UNIX **cron** daemon that runs the **system_check** script. The **cron** daemon runs the script on an ongoing basis at a regular interval, specified in the **/config/crontab** file.

## Failure notification

When run, the **system_check** script reports the hardware status to the console. The script also logs all fan and power-supply failures to the file **/var/log/bigip**.

In addition to displaying status to the console and logging failures in the file **/var/log/bigip**, the BIG-IP system also displays an alarm condition on the front panel LEDs of the Application Switch platform. Table 5.1 shows the LED usage for alarm conditions related to fans and power supplies.

| Hardware component | Status LED | Activity LED | Alarm LED |
|---|---|---|---|
| Chassis and CPU fans | X | X | Blinking red |
| Power supply 1 (upper) | Blinking yellow | Solid green | Blinking red |
| Power supply 2 (lower) | Solid green | Blinking yellow | Blinking red |

***Table 5.1*** *LED usage for hardware failures*

## Configuring system_check monitoring

You can customize the behavior of the **system_check** script in the following ways:

- By changing the interval at which the **cron** daemon runs the script
- By changing the way that the script displays output on the console
- By disabling the monitoring of redundant power supplies.

### To change the system_check monitoring interval

You can change the interval at which the **cron** daemon runs the **system_check** script by editing the interval value specified in the **/config/crontab** file.

### To configure the system_check output to the console

By default, the status of chassis fans, CPU fans, and power supplies is suppressed from appearing on the console. (This is known as ***quiet mode***.) Only error conditions are displayed. This automatic suppression of status is specified by default in the **/config/crontab** file, with the string **system_check -q**.

You can remove the suppression of status and thus cause the BIG-IP system to display status on the console.  You can do this by editing the **/config/crontab** file to remove the **-q** option from the string **system_check -q**.

To display full SNMP platform table information (that is, run in debug mode), type the following:

```
system_check -d
```

### To enable or disable power-supply monitoring

You can enable or disable the monitoring of redundant power supplies by configuring the database key **Local.Platform.PowerSupplyMonitor**. The default setting enables the monitoring of the power supplies.

If the database key is not present or is set to **1**, the power supplies are monitored. If the database key is set to **0**, the  power supplies are not monitored.

# Index