Knowledge

K6917: Overview of BIG-IP persistence cookie encoding

Published Date: **Nov 4, 2015** Updated Date: **Feb 21, 2023**

Applies to:

Topic

When you configure a cookie persistence profile to use the **HTTP Cookie Insert** or **HTTP Cookie Rewrite** method, the BIG-IP system inserts a cookie into the HTTP response, which well-behaved clients include in subsequent HTTP requests for the host name until the cookie expires. The cookie, by default, is named **BIGipServer<pool_name>**. The cookie is set to expire based on the expiration setting configured in the persistence profile. The cookie value contains the encoded IP address and port of the destination server.

Description

IPv4 pool members

Address encoding

The BIG-IP system uses the following address-encoding algorithm:

1. Convert each octet value to the equivalent 1-byte hexadecimal value.
2. Reverse the order of the hexadecimal bytes and concatenate to make one 4-byte hexadecimal value.
3. Convert the resulting 4-byte hexadecimal value to its decimal equivalent.

For example, if the IP address of the destination server is **10.1.1.100**, the BIG-IP LTM system encodes the address as follows:

```
10.1.1.100 = 0x0A . 0x01 . 0x01 . 0x64
```

```
Reverse byte order, concatenated = 0x6401010A
```

```
0x6401010A = 1677787402
```

The address encoding algorithm is performed algebraically, as follows, for address **(a.b.c.d)**:

```
a + b*256 + c*(256^2) + d*(256^3)
```

For example, if the IP address of the destination server is **10.1.1.100**, the encoded address is derived as follows:

```
a=10; b=1; c=1; d=100
```

```
10 + 1*256 + 1*(256^2) + 100*(256^3) = 1677787402
```

Port encoding

The BIG-IP system uses the following port encoding algorithm:

1. Convert the decimal port value to the equivalent 2-byte hexadecimal value.
2. Reverse the order of the 2 hexadecimal bytes.
3. Convert the resulting 2-byte hexadecimal value to its decimal equivalent.

For example, if the port of the destination server is **8080**, the BIG-IP LTM system encodes the port as follows:

```
8080 = 0x1F90
```

```
Reverse byte order = 0x901F
```

```
0x901F = 36895
```

Note*: If the port value is less than 256, the first byte in step 1 is* ***0x00****. For example, if the port value is* ***80****, the BIG-IP LTM system encodes the port as follows:* ***80*** *=* ***0x0050***

```
Reverse byte order = 0x5000
```

```
0x5000 = 20480
```

Persistence cookie value

The BIG-IP system combines the two encoded values and inserts them into the persistence cookie. For example, using the IP address and port **10.1.1.100:8080** as encoded previously, the persistence value that the BIG-IP LTM system encodes in the cookie is as follows:

```
1677787402.36895.0000
```

Note*: The field following the port encoding is reserved for future use and always contains four zeros as placeholders.*

Decoding persistence cookie values

You can decode the cookie value by reversing the encoding algorithms previously detailed.

For example, using the IP address and port **10.1.1.100:8080** as previously encoded, the persistence value that the BIG-IP LTM system encodes in the cookie is as follows:

```
1677787402.36895.0000
```

The first field in the cookie references the IP address of the destination server.

1. Convert the decimal value **1677787402** to its 4-byte hexadecimal equivalent:
   For example using& the Linux command line **print** command:

   ```
   printf "%#x\n" 1677787402
   ```

   The returned hex is: **0x6401010A**
2. Split the hexadecimal output into four separate hexadecimal bytes and reverse the byte order:
   `0x0A 0x01 0x01 0x64`
3. Convert each 1-byte hexadecimal value to its equivalent decimal value, one per octet:
   For example using the Linux command line **echo** command:

   ```
   echo  $((0x0A)).$((0x01)).$((0x01)).$((0x64))
   ```

   The command output is: **10.1.1.100**

The second field in the cookie references the port of the destination server.

1. Convert the decimal value **36895** to the equivalent 2-byte hexadecimal value:
   For example using the Linux command line **printf** command:

   ```
   printf "%#x\n" 36895
   ```

The command output is: **0x901f**

2. Reverse the order of the two hexadecimal bytes by swapping the value pairs after 0x:
   `0x1f90`

   Note: *In the previous example, the values of **90** and **1f** are swapped.*
3. Convert the resulting 2-byte hexadecimal value to its decimal equivalent:
   For example using the Linux command line **echo** command:

   ```
   echo $((0x1f90))
   ```

   The command output is:

   <"example2">8080

Note: *You can use an iRule to intercept and decode persistence cookies using the previous algorithms. For example, refer to **Persistence Cookie Logger** on DevCentral.*

IPv4 pool members in non-default route domains

The Route Domains feature is introduced in BIG-IP 10.0.0. For more information about route domains, refer to the ***BIG-IP Local Traffic Manager: Implementations*** manual.

Note: *For information about how to locate F5 product manuals, refer to **K98133564: Tips for searching AskF5 and finding product documentation**.*

If a pool member resides in a non-default route domain (for example, route domain ID **5**), a different encoding is used to calculate the value of the persistence cookie.

The persistence cookie for a host in a non-default route domain is the concatenation of the following:

- `rd`
- `<The route domain ID>`
- `o`
- `00000000000000000000ffff`
- `<The hexadecimal representation of the IP address of the pool member>`
- `o`
- `<The port number of the pool member>`

For example, if a connection was load balanced to the **192.0.2.1%5:80** pool member, the BIG-IP system would insert the following cookie:

```
BIGipServer<pool_name>=rd5o00000000000000000000ffffc0000201o80
```

IPv6 pool members

If the pool member is an IPv6 host, the persistence cookie is the concatenation of:

- `vi`
- `<The full hexadecimal IPv6 address>`
- `.`
- `<The port number calculated in the same way as for IPv4 pool members>`

For example, if a connection was load balanced to the **[2001:0112::0030]:80** pool member, the BIG-IP system would insert the following cookie:

```
BIGipServer<pool_name>=vi20010112000000000000000000000030.20480
```

Note: *For information about an issue where the port value for an IPv6 pool member is incorrectly translated to a random number, refer to **K13816: The BIG-IP system may generate persistence cookies with an incorrectly-formatted value**.*

IPv6 pool members in non-default route domains

Note: *IPv6 Route Domains feature support is introduced in BIG-IP 11.1.0. For more information, refer to* ***K13388: Route domains support for IPv6***.

If a pool member resides in a non-default route domain (for example, route domain ID **3**), a different encoding is used to calculate the persistence cookie value.

The persistence cookie for a host in a non-default route domain is the concatenation of the following:

- `rd`
- `<The route domain ID>`
- `o`
- `<The full hexadecimal IPv6 address>`
- `o`
- `<The port number of the pool member>`

For example, if a connection was load balanced to the **2001:0112::0030%3:80** pool member, the BIG-IP system would insert the following cookie:

```
BIGipServer<pool_name>=rd3o20010112000000000000000000000030o80
```

Note: *For information about an issue where the port value for an IPv6 pool member is incorrectly translated to a random number, refer to* ***K13816: The BIG-IP system may generate persistence cookies with an incorrectly-formatted value***.

Encrypting cookies

In some environments, it may be unacceptable to disclose the IP and Port numbers of origin web servers (OWS) behind the BIG-IP system in an HTTP cookie. If your security policy requires this information to be further obfuscated, refer to the processes described in **K14784: Configuring cookie encryption within the HTTP profile (10.x - 15.x)** or **K23254150: Configuring cookie encryption for BIG-IP persistence cookies from the cookie persistence profile**.

Related Content

- **K5714: Troubleshooting cookie persistence**
- F5 DevCentral video: **BIG-IP Cookie Persistence**