Elastic Load Balancing RFC 2616 HTTP Header Update Tools

August 12th, 2019

Overview

In an effort to better conform to RFC2616, AWS is modifying the default request header processing of our Layer-7 Amazon Classic Load Balancers to match the behavior of the Application Load Balancers. This document provides details of the change as well as instructions about how to use the included tools to check a HTTP server's behavior and change a load balancer's behavior. This behavior can be enabled or disabled on any Classic Load Balancer using the AWS ELB API, and the script included with this document allows you to check or set desired behavior.

Recommended Actions

1. Familiarize yourself with the changes in HTTP header parsing.
2. Determine if it is safe to update to '*tolerant*' parsing behavior. You can use the included script '*test_clb_default_behavior_migration_checker.py*'.
3. If it is safe, update your load balancer to '*tolerant*' behavior using the included script '*manage-clb-header-policies.sh*' script.
4. If it is not safe, set your load balancer to '*legacy*' behavior using the included script '*manage-clb-header-policies.sh*' script.
5. If there are any issues after configuring a load balancer, you can change the behavior or remove the policy script '*manage-clb-header-policies.sh*' script.
6. To change the behavior of your load balancer in the future, you can run the '*manage-clb-header-policies.sh*' script.
7. You can also remove either policy from your load balancer with the '*manage-clb-header-policies.sh*' script, specifying the '*remove*' action.

Changes in HTTP Header Parsing

Overview of change

The update made to HTTP header parsing of Classic Load Balancer are to enable it to behave like a 'Tolerant Application', as defined in RFC 2616 section 19.3 [Appendix 2]. This causes the behavior to be less strict than the end-of-line marker defined in RFC 2616 section 2.2 [Appendix 1]. Most HTTP servers already perform this behavior and receiving client headers with LF end-of-line markers are handled automatically.

1. Update made – now tolerant by default.
2. If you were contacted, the ELBs listed are legacy.
3. We will remove legacy default on ELBs listed starting Jan 15, 2020.

Changes made vs. delayed.

We have changed the default behavior for Classic Load Balancers from legacy to tolerant. Additionally, we have updated all CLBs to use tolerant behavior with the exception of CLBs which we detected might be using LF as in-line characters. Those load balancers were not updated, and we have reached out to you with a list of CLBs that were not updated. Only those CLBs will have the legacy behavior, by default, as they were not updated. All other CLBs have been updated to use the tolerant behavior. This update is a software change and is independent of the policy that you can configure via the tools included in this archive [Appendix 4]. If you were not contacted, or a

This means all existing CLBs will exhibit

If you were contacted by AWS for this issue, we recommend you examine the load balancers listed in the notice and set the desired behavior.

Configuration options available

1. You can set either behavior.
2. Whatever you set will persist until you change it.

Paragraph 3: Choosing an option

1. All CLBs till now used legacy, safe to use legacy.
2. We think it's safe to set tolerant behavior.
3. We have not seen a use case for legacy behavior.
4. Investigate if your application requires '\n' in the same line of the HTTP header.

If your application uses HTTP headers in normal ways, it is likely you can enable the tolerant behavior of your load balancer with no impact. If your application requires LF ('\n') to be considered a valid in-line character, we recommend you enable the legacy behavior on your load balancer.

We recommend updating applications to support use of the tolerant behavior, however, we will continue to support workloads that require the legacy behavior by using the policy override.

If you are unsure which behavior, you can safely configure, and continue to use, the legacy behavior, as this is the behavior all Classic Load Balancers have exhibited prior to this change. If you require the legacy behavior, we recommend you configure the legacy behavior. If you were notified that AWS has not modified the behavior of specific Classic Load Balancers (via email or a Personal Health Dashboard notice), those load balancers will continue to use the legacy

r load balancers were configured for the legacy behavior, they will retain this configuration until at least January 15, 2020. This date does not affect your setting of legacy or tolerant. Note that newly created Classic Load Balancers will utilize the tolerant behavior until configured otherwise.

Changing Your CLB Behavior

If you need to change the behavior of our load balancer, you can use the included script '*manage-clb-header-policies.sh*' to do so. The script requires the AWS CLI to be installed and configured for access to your account, and requires the Bash environment to execute in. Alternatively, you can use the CLI or API directly as detailed below [Appendix 3]. Once configured, a load balancer will retain the configuration until you change it.

The script has 3 functions, '***check'***, '***tolerant'***, and '***legacy'***.

Check will look for policies of the given name on the given load balancer. Example usage:

```
./manage-clb-header-policies.sh ${MY_ELB_NAME} check ${MY_REGION}
```

If your load balancer is named '*my-load-balancer*' and is in the US-EAST-1 Region, to enable tolerant behavior, you would run:

```
./manage-clb-header-policies.sh my-load-balancer check us-east-1
```

Tolerant will configure the given load balancer for the new default, tolerant behavior. Example usage:

```
./manage-clb-header-policies.sh ${MY_ELB_NAME} tolerant ${MY_REGION}
```

If your load balancer is named '*my-load-balancer*' and is in the US-EAST-1 Region, to enable tolerant behavior, you would run:

```
./manage-clb-header-policies.sh my-load-balancer tolerant us-east-1
```

Legacy will configure the given load balancer for the previous default, legacy behavior.

```
./manage-clb-header-policies.sh ${MY_ELB_NAME} legacy ${MY_REGION}
```

If your load balancer is named '*my-load-balancer*' and is in the US-EAST-1 Region, to enable legacy behavior, you would run:

```
./manage-clb-header-policies.sh my-load-balancer legacy us-east-1
```

Remove will configure the given load balancer to use the default, tolerant, behavior (unless AWS has informed you differently for a specific load balancer).

```
./manage-clb-header-policies.sh ${MY_ELB_NAME} remove ${MY_REGION}
```

If your load balancer is named '*my-load-balancer*' and is in the US-EAST-1 Region, to enable legacy behavior, you would run:

```
./manage-clb-header-policies.sh my-load-balancer remove us-east-1
```

Checking a HTTP Endpoint for Behavior

The included Python script 'test_clb_default_behavior_migration_checker.py' will connect via HTTP to an endpoint and attempt to determine the handling of LF '\n' only characters in HTTP headers. If the script runs and reports all tests pass, you should enable the tolerant behavior on any Classic Load Balancers in front of the backend. If there are some cases that fail, you should investigate if the LF characters are required for your application or workload and set the policy you require. This script can be used to confirm changes made to your HTTP server will have no impact when migrating to the tolerant behavior. A check failure on this is not an indicator that your HTTP server would have a problem with the tolerant behavior, but that it may. Confirming that there will be no impact if any of the tests from this script fail requires understanding your application and clients, and the risk of changes cannot be determined from a script. Since we detected clients were sending you requests with the LF only character in HTTP headers, if you are unable to determine enabling tolerant behavior is safe for your application, we recommend you set the legacy behavior until it can be confirmed.

The script requires an IP (or hostname) for the -ip option and assumes TCP port 80. Help output details all of the options, and is included as [Appendix 3].

Example script usage with a tolerant HTTP endpoint

```
./test_clb_default_behavior_migration_checker.py -ip www.example.com -port 80
```

✅ `Initial CLB Health Check Passed`

✅ `LR and CRLF character in header name check passed`

✅ `LR and CRLF character in header value check passed`

✅ Final CLB Health Check Passed

✅ Your targets support tolerant behavior, you should set tolerant behavior on your CLB. If you need help, please reach out to AWS Support

Example script usage with a non-tolerant HTTP endpoint. (Note this endpoint does not require LF only characters in HTTP headers)

✅ Initial CLB Health Check Passed
❌ Your targets use legacy behavior. You may still be able to set tolerant behavior on your CLB, read the supporting document or reach out to AWS Support

Appendix 1 – RFC 2616 Section 2.2

RFC 2616 section 2.2 defines:

> HTTP/1.1 defines the sequence CR LF as the end-of-line marker for all protocol elements except the entity-body (see appendix 19.3 for tolerant applications). The end-of-line marker within an entity-body is defined by its associated media type, as described in section 3.7.
>
> CRLF = CR LF

Appendix 2– RFC 2616 Section 19.3

Further, in RFC 2616 section 19.3 it defines:

```
Although this document specifies the requirements for the generation
   of HTTP/1.1 messages, not all applications will be correct in their
   implementation. We therefore recommend that operational applications
   be tolerant of deviations whenever those deviations can be
   interpreted unambiguously.

   Clients SHOULD be tolerant in parsing the Status-Line and servers
   tolerant when parsing the Request-Line. In particular, they SHOULD
   accept any amount of SP or HT characters between fields, even though
   only a single SP is required.

   The line terminator for message-header fields is the sequence CRLF.
   However, we recommend that applications, when parsing such headers,
   recognize a single LF as a line terminator and ignore the leading CR.

   The character set of an entity-body SHOULD be labeled as the lowest
   common denominator of the character codes used within that body, with
   the exception that not labeling the entity is preferred over labeling
   the entity with the labels US-ASCII or ISO-8859-1. See section 3.7.1
   and 3.4.1.

   Additional rules for requirements on parsing and encoding of dates
   and other potential problems with date encodings include:

      - HTTP/1.1 clients and caches SHOULD assume that an RFC-850 date
        which appears to be more than 50 years in the future is in fact
        in the past (this helps solve the "year 2000" problem).

      - An HTTP/1.1 implementation MAY internally represent a parsed
        Expires date as earlier than the proper value, but MUST NOT
        internally represent a parsed Expires date as later than the
        proper value.

      - All expiration-related calculations MUST be done in GMT. The
        local time zone MUST NOT influence the calculation or comparison
        of an age or expiration time.

      - If an HTTP header incorrectly carries a date value with a time
        zone other than GMT, it MUST be converted into GMT using the
        most conservative possible conversion.
```

Appendix 3 – AWS CLI Commands

Setup environment variables (Bash), alternatively you can replace the variables as strings in commands

```
MY_ELB_NAME="elb-short-name"
MY_REGION="us-east-1" # Or other RegionName's as returned from 'aws ec2
describe-regions'
MY_PARSING_VERSION=1 # Set 1 for legacy behavior (ignore lone LF '\n')
MY_PARSING_VERSION=2 # Set 2 for tolerant behavior (treat lone LF '\n' as
CRLF '\r\n')
```

First command: Create the policy using the AWS CLI elb command 'create-load-balancer-policy', for example:

```
aws elb create-load-balancer-policy --region ${MY_REGION} --load-balancer-
name ${MY_ELB_NAME} --policy-name
RequestHeaderParsingPolicyVersion${MY_PARSING_VERSION} --policy-type-name
RequestHeaderParsingPolicyType --policy-attributes
AttributeName=RequestHeaderParsingPolicyVersion,AttributeValue=${MY_PARSING_V
ERSION})
```

Second Command: set the load balancer policy, using the AWS CLI elb command 'set-load-balancer-policies-of-listener', for example:

```
aws elb set-load-balancer-policies-of-listener --region ${MY_REGION} --load-
balancer-name ${MY_ELB_NAME} --load-balancer-port 0 --policy-names
RequestHeaderParsingPolicyVersion${MY_PARSING_VERSION})
```

Confirming the policy is created can be done with either 'describe-load-balancer' or 'describe-load-balancer-policies', for example:

```
aws elb --region us-east-2 describe-load-balancer-policies --load-balancer-
name ${MY_ELB_NAME}
```

The policy can only be set on Listener port 0, however it is not listed in the OtherPolicies for that listener. It cannot be set without existing on the load balancer. This is why the 'manage-clb-header-policies.sh' script deletes policies that it does not set.

```
aws elb --region us-east-2 describe-load-balancers --load-balancer-name
${MY_ELB_NAME}
```

Example output:

```
          "Listener": {
            "LoadBalancerPort": 0,
            "InstancePort": 0
          },
          "PolicyNames": []
        }
```

```
    ],
    "Policies": {
    …
      "OtherPolicies": [
        "RequestHeaderParsingPolicyVersion2"
    ]}
```

You can remove load balancer policy by setting an empty list '[]' as the policy for listener 0. To do this, use the AWS CLI elb command 'set-load-balancer-policies-of-listener', for example:

```
aws elb set-load-balancer-policies-of-listener --region ${MY_REGION} --load-
balancer-name ${MY_ELB_NAME} --load-balancer-port 0 --policy-names '[]'
```

Note that the Elastic Load Balancing API only indicates if a policy exists, and does not directly indicate if a policy is applied to a load balancer. Policies can exist without being applied, and the included script handles this by deleting previous policies whenever changing the policy. Customers who you utilize the AWS CLI or ELB API directly will have to track which setting you have set to be sure that an existing policy exists. The only indication that a policy is in use available is that an error will be returned if you attempt to delete it.

```
aws elb delete-load-balancer-policy --region ${MY_REGION} --load-balancer-
name ${MY_ELB_NAME} --policy-name RequestHeaderParsingPolicyVersion2
```

Example output when the policy is in place:

```
An error occurred (InvalidConfigurationRequest) when calling the
DeleteLoadBalancerPolicy operation: You cannot delete policy
[RequestHeaderParsingPolicyVersion2] since it is enabled or referred by other
enabled policy.
```

Appendix 4 – Migration Check Script Help Output

```
./test_clb_default_behavior_migration_checker.py -h

usage: test_clb_default_behavior_migration_checker.py [-h] -ip TARGET_IP
                                                      [-port TARGET_PORT]
                                                      [-path TARGET_PATH]
                                                      [-t TIMEOUT]
                                                      [-sip SOURCE_IP] [-s]
                                                      [-v] [--no-clb-hc] [-d]


Test if you can safely migrate your CLB to new default behavior


optional arguments:
  -h, --help            show this help message and exit
  -ip TARGET_IP, --target-ip TARGET_IP
                        IP address of the target. eg. -ip 192.168.10.2
  -port TARGET_PORT, --target-port TARGET_PORT
                        Port of the target. eg. -port 80
  -path TARGET_PATH, --target-path TARGET_PATH
                        A known good http path to test eg. /, /ping,
                        /?ping=healthcheck etc.
  -t TIMEOUT, --timeout TIMEOUT
                        Connection timeout to the target in seconds. eg. -t
60
  -sip SOURCE_IP, --source-ip SOURCE_IP
                        By default ip that resolves to hostname on the
machine
                        will be used to connect to the target. Use this
option
                        to override the behavior eg. -sip 192.168.10.10
```

```
  -s, --secure-listener
                        Set this flag if target uses a secure listener. This
                        option is a flag so just add the argument to set it
                        eg. -s
  -v, --verbose         Enabled debug level logging
  --no-clb-hc           By default the script runs a health check as CLB
                        before and after running other tests. Set this flag
to
                        disable that check. Other checks are still executed
  -d, --dry-run         Print the headers and paths that would be checked but
                        does not execute
```

Appendix 5 – Related Tools