Problem description

Envoy is configured to log access in `istio-proxy` proxy, whether it's HTTP or TCP. For HTTP Listeners, it also logs details about the HTTP protocol. In our recent update of Istio, we found out that we're missing a white space between the URL and the Protocol:

```
clientapi-67745bbfb-w8gdk istio-proxy [2019-03-13T14:57:14.768Z] "GET /.well-
known/apple-app-site-associationHTTP/1.1" 200 - 0 175 48 46
"86.18.85.81,100.118.92.64" "swcd (unknown version) CFNetwork/976
Darwin/18.2.0" "700659de-12e5-9053-9146-80bc14a0e45f" "api.[redacted]com"
"127.0.0.1:80" inbound|80||clientapi-external-service.default.svc.cluster.local
- 100.97.36.219:80 100.118.92.64:0
```

This makes log processing more difficult, so we tried to find out why is this happening, and how can we resolve it.

Investigation

This log format comes from the Envoy listener, which is configured by Pilot's LDS (Listener Discovery) service. Unfortunately, the format is hardcoded:

```
    if env.Mesh.AccessLogFile != "" {
        fl := &fileaccesslog.FileAccessLog{
            Path:   env.Mesh.AccessLogFile,
            Format: EnvoyHTTPLogFormat,
        }

        connectionManager.AccessLog = []*accesslog.AccessLog{
            {
                Config: util.MessageToStruct(fl),
                Name:   xdsutil.FileAccessLog,
            },
        }
    }
```

```
EnvoyHTTPLogFormat = "[%START_TIME%] \"%REQ(:METHOD)% %REQ(X-ENVOY-ORIGINAL-
PATH?:PATH)%" +
    "%PROTOCOL%\" %RESPONSE_CODE% %RESPONSE_FLAGS% %BYTES_RECEIVED%
%BYTES_SENT% " +
    "%DURATION% %RESP(X-ENVOY-UPSTREAM-SERVICE-TIME)% \"%REQ(X-FORWARDED-
FOR)%\" " +
    "\"%REQ(USER-AGENT)%\" \"%REQ(X-REQUEST-ID)%\" \"%REQ(:AUTHORITY)%\"
\"%UPSTREAM_HOST%\" " +
    "%UPSTREAM_CLUSTER% %UPSTREAM_LOCAL_ADDRESS% %DOWNSTREAM_LOCAL_ADDRESS% " +
    "%DOWNSTREAM_REMOTE_ADDRESS% %REQUESTED_SERVER_NAME%\n"
```

As you can see, they're simply missing the space between `%REQ(X-ENVOY-ORIGINAL-PATH?:PATH)%` in the first line and `"%PROTOCOL%` in the second. Unfortunately that's also true for their latest version of Istio, `1.1.0.snapshot.1` as well.

As you can see, in `1.0.2`, they didn't configure the format, they let Envoy have its default:

```
    if env.Mesh.AccessLogFile != "" {
        fl := &fileaccesslog.FileAccessLog{
            Path: env.Mesh.AccessLogFile,
        }

        connectionManager.AccessLog = []*accesslog.AccessLog{
            {
                Config: util.MessageToStruct(fl),
                Name:   fileAccessLog,
            },
        }
    }
```

The first version they started to configure logging in `1.0.3`.

Update

They fixed this in `1.1.0-snapshot.3`:

```
EnvoyTextLogFormat = "[%START_TIME%] \"%REQ(:METHOD)% %REQ(X-ENVOY-ORIGINAL-
PATH?:PATH)% " +
    "%PROTOCOL%\" %RESPONSE_CODE% %RESPONSE_FLAGS% %BYTES_RECEIVED%
%BYTES_SENT% " +
```

Also, in `1.1.0-snapshot-5` they added increased configurability around the `istio-proxy` logs, including changeable log format and switch from text to JSON:

```
    # If accessLogEncoding is TEXT, value will be used directly as the log
format
    # example: "[%START_TIME%] %REQ(:METHOD)% %REQ(X-ENVOY-ORIGINAL-
PATH?:PATH)% %PROTOCOL%\n"
    # If AccessLogEncoding is JSON, value will be parsed as map[string]string
    # example: '{"start_time": "%START_TIME%", "req_method": "%REQ(:METHOD)%"}'
    # Leave empty to use default log format
    accessLogFormat: '{{ .Values.global.proxy.accessLogFormat }}'
    # Set accessLogEncoding to JSON or TEXT to configure sidecar access log
    accessLogEncoding: '{{ .Values.global.proxy.accessLogEncoding }}'
```

Conclusion

As of now there is no easy way to fix this. Even if we would hijack the sidecar proxy bootstrap configuration and manually overwrite the envoy startup args, where we could set up a default logging format, Pilot would override it with the hardcoded incorrect format, since Envoy configures the logging for each individual listener via LDS, which is more specific than the default log format.

It appears this will get resolved only by upgrading at one point.

+1, Fun ticket

There is a fun ticket about this. In order, the following happens:

- someone raises a ticket since `istio-proxy` is not logging JSON anymore, and they want to process logs in Kibana (reasonable request)
- Istio team wrote and applied a fix for it in the past
- Istio team removed the fix as a result of a merge
- Istio team misunderstands the question and explains how protobuf is better than JSON and the preferred way of communication (reasonable, but that wasn't the question. The original question was about log format, not component communication.)
- as a solution the Istio team recommends the ticket owner to fetch the config via JSON through the admin port, which is again, valid but irrelevant from the question point of view. How they think Kibana and the proxy config comes together?

Confusion multiplied by misunderstanding, equals fun. Anyway, it looks like eventually they sort it out! 🤞

SRE @ [redacted]

13, March 2019