Expired Certificates

Let's Encrypt Root Certificate expired on September 30th and this change is causing some issues. We explain the possible problems below and also how to solve them.

This is not an issue with Meteor or Galaxy, but a natural process if you are using Let's Encrypt's generated certificates.

Can't run Meteor commands

Galaxy and all Meteor servers uses Let's Encrypt, which announced a change in May in this post about DST Root CA X3 expiring on September 30, 2021.

Older versions of Meteor, more specifically anything older than Meteor v1.9 shipped with a Node.JS version below v10, which used OpenSSL < 1.0.2.

If you are getting errors like Connection error (certificate has expired) when running Meteor commands it means that you are running a version of Meteor older than v1.9.

A workaround, for now, is to run all the meteor commands with the following environment variable ***NODE_TLS_REJECT_UNAUTHORIZED***, for example in the deploy command:

```
NODE_TLS_REJECT_UNAUTHORIZED=0 meteor deploy
```

Also note that if you are running old distributions, like Ubuntu 16 and before, locally, or in any of your CI pipelines you may also face this issue. In this case, we do recommend updating your distribution, or your local repository of root certificates (the how-to of this varies based on your distribution).

This is not a Meteor or Galaxy issue, but it's a change in the Let's Encrypt certificate in our resources that you are accessing.

Requests failing

If your server is accessing external resources where the target host is using Let's Encrypt certificates and your app is running an old Meteor version, you will also need to add `NODE_TLS_REJECT_UNAUTHORIZED` to your server environment variables.

If you are using Galaxy, it's as simple as adding this to your settings file:

```
{
  "galaxy.meteor.com": {
    "env": {
      "NODE_TLS_REJECT_UNAUTHORIZED": "0"
    }
  }
}
```

Please note: We don't recommend continued use of this workaround, as any SSL certificate is going to be authorized and you are exposing your application to serious security issues. The best option is to update Meteor to latest version, or at least Meteor 1.9 as it is the first using Node.js 12.

You can check our list of supported Meteor versions here. If your applications is not in one of them, you should migrate as soon as possible.

This is not a Meteor or Galaxy issue, but it's a change in the Let's Encrypt certificate in the external resource that you are accessing.

Client Compatibility

As stated before, Galaxy issues Let's Encrypt certificates automatically for all clients. This is source of confusion, as if you are depending on older clients being able to access your website, this won't work.

If Let's encrypt certificates are not good for your clients you would need to acquire other certificate from a different provider and upload your custom certificate into Galaxy.

You can also generate a Let's Encrypt certificate manually and upload to Galaxy, but specifying an alternative preferred chain on certbot:

```
sudo certbot certonly --manual --preferred-chain "ISRG Root X1" --preferred-challenges dns
```

More info can be obtained here.

If you are using Galaxy, you need to follow the requirements and steps here after generating the certificate. Galaxy only accepts custom certs in `.pem` format, the same as nginx uses.

This is not a Meteor or Galaxy issue, but it's a change in the Let's Encrypt certificate you are using.

Clients Known to be not working

Here is a succinct list of known to be not working clients:

- Mac OS X prior to 10.12.1. Any browser, except firefox that bundles root chains, won't work.
- Node.JS HTTP requests prior to v10. This includes any Meteor version prior to 1.9(except).
- Any client using OpenSSL 1.0.2 and before.

Please note that this is not an exhaustive list, but based on our reports and experience.

This is not a Meteor or Galaxy issue, but it's a change in the Let's Encrypt certificate you are using.