Vulnerability analysis

OIDCC

Example to illustrate the vulnerability.

```
{ok, Claims} =
  oidcc:retrieve_userinfo(
      Token,
      myapp_oidcc_config_provider,
      <<"client_id">>,
      <<"client_secret">>,
      #{}
  )
```

The vulnerability is present in *oidcc_provider_configuration_worker:get_ets_table_name/1*
The function `get_ets_table_name` is calling *erlang:list_to_atom/1*

```
Unset
get_ets_table_name(WorkerName) when is_atom(WorkerName) ->
    {ok, erlang:list_to_atom(erlang:atom_to_list(WorkerName) ++ "_table")};
get_ets_table_name(_Ref) ->
    error.
```

There might be a case (Very highly improbable) where the 2nd argument of *oidcc:retrieve_userinfo/5* is called with a different atom each time which eventually leads to the atom table filling up and the node crashing.

It is recommended to add a note in your documentation about this issue because even though it is highly improbable, it is still possible to use it in a vulnerable way.

```
{ok, Claims} =
  oidcc:retrieve_userinfo(
      Token,
      Myapp_oidcc_config_provider,      %Make sure to not create atoms dynamically here
      <<"client_id">>,
      <<"client_secret">>,
      #{}
  )
```

Our audit used Static Analysis with Data flow and control flow analysis.
The OIDCC Erlang code is secure ✅