AWS:

Database Elevate Privilege & Remote Code Execution

Platform:

AWS Database for PostgreSQL 11.16 R1 & Aurora PostgreSQL (some version)

Class:

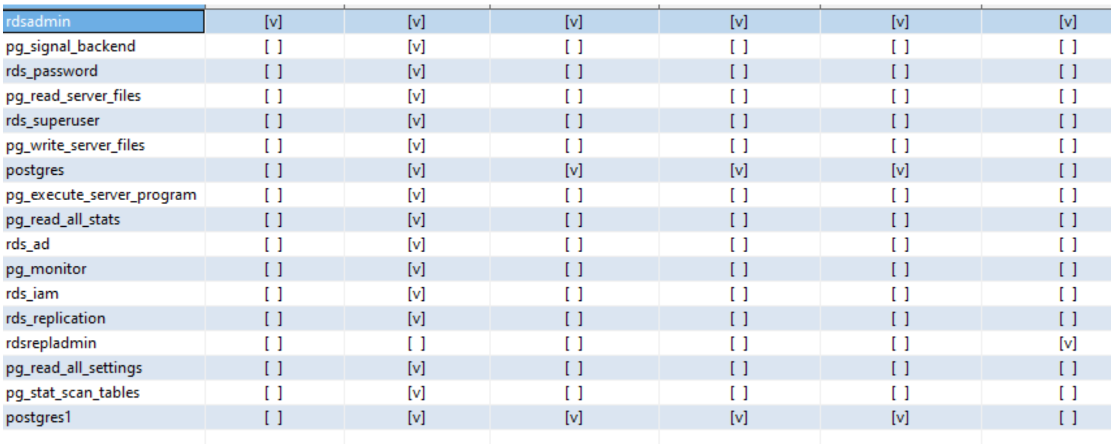
Remote Code Execution

Summary:

I found a Database Elevate Privilege in PostgreSQL and can be extended to Aurora with some specific versions. I think the reason is that the excessive permissions of the role rds_superuser. I suggest that you need some basic PostgreSQL Knowledge to read the follow parts, and forgive my poort english writing.
A briefly description (just follow me to do)：at first, we can create a non-rds_superuser which named postgres1, then create a extension pgtap with schema public and some functions which make them to be a member of extension pgtap. After that, grant rds_superuser to postgres1 which make postgres1 can alter extension pgtap to schema pg_catalog, then we get a function in pg_catalog. Next, install specific extension which can be hooked by the function I createtd when execute its sql. Normally to say, it prohibits direct privilege escalation, but we can alter a function owner to rdsadmin with lable security definer. At end, we got a function to execute with role rdsadmin, then it is still hard to do some high-risk operations but without load so library, we update table pg_language to set c is true and grant the right of c to postgres1, then create the file_fdw just by sql, we finally can enjoy code execution with file_fdw's option program.
The next is a actual operation with PostgreSQL 11.16 R1 with SQL I used and proof prensted in the form of pictures.

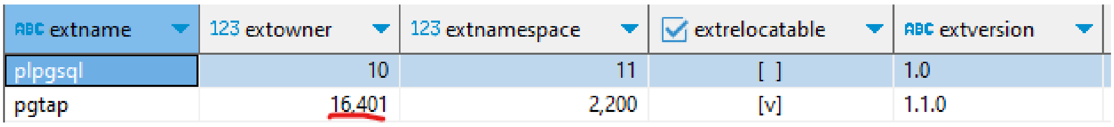
Attack Description (The SQL used is in Appendix):

1. Create a PostgreSQL server and connect it, then create the non-rds_superuse postgres1 and login.

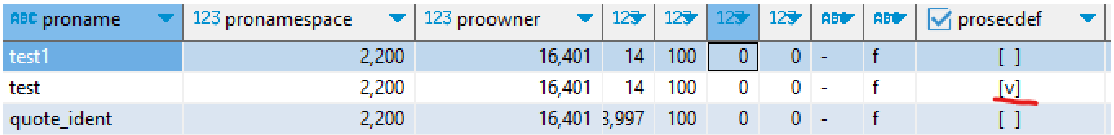


rdsadmin	[v]	[v]	[v]	[v]	[v]	[v]
pg_signal_backend	[ ]	[v]	[ ]	[ ]	[ ]	[ ]
rds_password	[ ]	[v]	[ ]	[ ]	[ ]	[ ]
pg_read_server_files	[ ]	[v]	[ ]	[ ]	[ ]	[ ]
rds_superuser	[ ]	[v]	[ ]	[ ]	[ ]	[ ]
pg_write_server_files	[ ]	[v]	[ ]	[ ]	[ ]	[ ]
postgres	[ ]	[v]	[v]	[v]	[v]	[ ]
pg_execute_server_program	[ ]	[v]	[ ]	[ ]	[ ]	[ ]
pg_read_all_stats	[ ]	[v]	[ ]	[ ]	[ ]	[ ]
rds_ad	[ ]	[v]	[ ]	[ ]	[ ]	[ ]
pg_monitor	[ ]	[v]	[ ]	[ ]	[ ]	[ ]
rds_iam	[ ]	[v]	[ ]	[ ]	[ ]	[ ]
rds_replication	[ ]	[v]	[ ]	[ ]	[ ]	[ ]
rdsrepladmin	[ ]	[ ]	[ ]	[ ]	[ ]	[v]
pg_read_all_settings	[ ]	[v]	[ ]	[ ]	[ ]	[ ]
pg_stat_scan_tables	[ ]	[v]	[ ]	[ ]	[ ]	[ ]
postgres1	[ ]	[v]	[v]	[v]	[v]	[ ]

2. Create extension pgtap and functions public.test(), public.test1(), public.quote_ident(name).

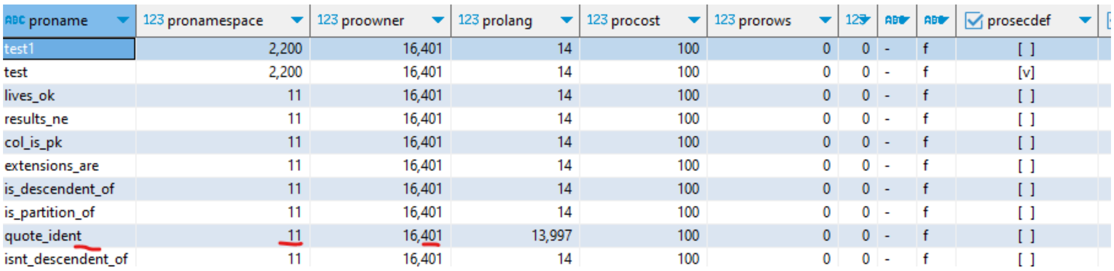


extname	extowner	extnamespace	extrelocatable	extversion
plpgsql	10	11	[ ]	1.0
pgtap	16,401	2,200	[v]	1.1.0



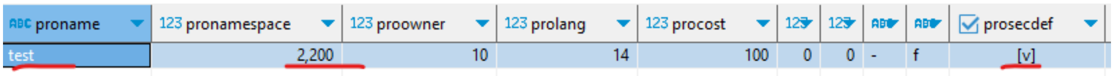
proname	pronamespace	proowner							prosecdef
test1	2,200	16,401	14	100	0	0	-	f	[ ]
test	2,200	16,401	14	100	0	0	-	f	[v]
quote_ident	2,200	16,401	3,997	100	0	0	-	f	[ ]

3. Alter extension pgtap add function public.quote_ident(name) and alter it to schema pg_catalog (Grant rds_superuser to postgres1).



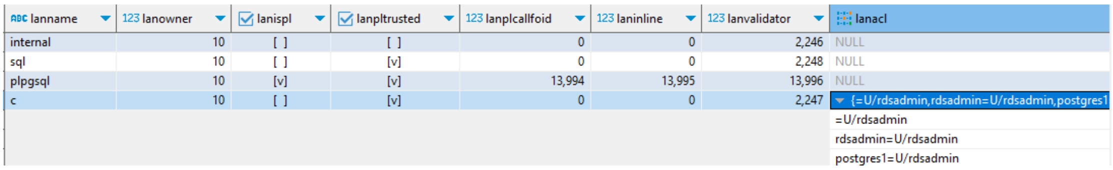
proname	pronamespace	proowner	prolang	procost	prorows				prosecdef
test1	2,200	16,401	14	100	0	0	-	f	[ ]
test	2,200	16,401	14	100	0	0	-	f	[v]
lives_ok	11	16,401	14	100	0	0	-	f	[ ]
results_ne	11	16,401	14	100	0	0	-	f	[ ]
col_is_pk	11	16,401	14	100	0	0	-	f	[ ]
extensions_are	11	16,401	14	100	0	0	-	f	[ ]
is_descendent_of	11	16,401	14	100	0	0	-	f	[ ]
is_partition_of	11	16,401	14	100	0	0	-	f	[ ]
quote_ident	11	16,401	13,997	100	0	0	-	f	[ ]
isnt_descendent_of	11	16,401	14	100	0	0	-	f	[ ]

4. Create extension cube to execute quote_ident in its SQL because of PostgreSQL unique type matching priority. Get a function public.test() which owner is rdsadmin and label is security definer.



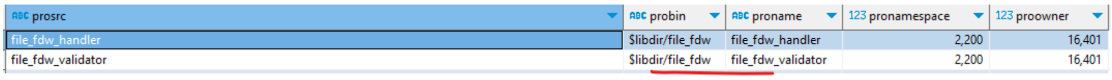
proname	pronamespace	proowner	prolang	procost					prosecdef
test	2,200	10	14	100	0	0	-	f	[v]

5. The public.test() will call function public.test1() which owner by postgres1, we can just update pg_language and grant language c to postgres. Put the logic to public.test1() and call public.test().



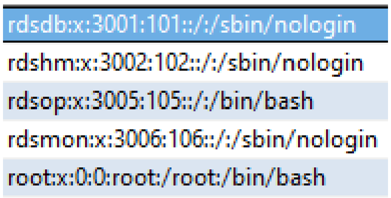
lanname	lanowner	lanispl	lanpltrusted	lanplcallfoid	laninline	lanvalidator	lanacl
internal	10	[ ]	[ ]	0	0	2,246	NULL
sql	10	[ ]	[v]	0	0	2,248	NULL
plpgsql	10	[v]	[v]	13,994	13,995	13,996	NULL
c	10	[ ]	[v]	0	0	2,247	{=U/rdsadmin,rdsadmin=U/rdsadmin,postgres1 =U/rdsadmin rdsadmin=U/rdsadmin postgres1=U/rdsadmin

6.	Create file_fdw with SQL, put the logic to public.test1() and call public.test() when some SQL can’t be execute. Just like file_fdw need pg_execute_server_program permission.



prosrc	probin	proname	pronamespace	proowner
file_fdw_handler	$libdir/file_fdw	file_fdw_handler	2,200	16,401
file_fdw_validator	$libdir/file_fdw	file_fdw_validator	2,200	16,401

7.	At last, we broke all security rules which I think is very hard, then enjoy your code execution with file_fdw’s options program.
(cat /etc/passwd).



```
rdsdb:x:3001:101::/:/sbin/nologin
rdshm:x:3002:102::/:/sbin/nologin
rdsop:x:3005:105::/:/bin/bash
rdsmon:x:3006:106::/:/sbin/nologin
root:x:0:0:root:/root:/bin/bash
```

Usage:

I will give all the sql I have used which named poc.sql.

Expected Result:

It shouldn’t be possible to be a superuser as aws document mentioned, even execute any code over host server.

Observed Result:

The Superuser Role is elevated and execute code on server.