ProOpen

Sichere dein pgBackRest gegen Ransomware!

Gunnar „Nick“ Bluth

PGConf.DE 2024

Hi, ich bin

- „Nick“ (eigentlich Gunnar) Bluth
- IT-Freelancer seit ~ 1997
- Inhaber und Geschäftsführer der Pro Open GmbH
- LinkedIn, Xing, WoT, ...
- Meist „bluthg“, „nickbluth“ oder „nickbloodh“
- nick@pro-open.de

Warum sollte man keine Daten verlieren?

- Reputation
- Geschäftsgrundlage
- Juristisches Risiko
  - Schadensersatz
- DSGVO & Co.
  - Strafen

Warum will jemand deine Daten?

- Schaden verursachen - „Zerstörer“
- Geschäftsgeheimnisse - „Diebe“
- Erpressung

„Zerstörer“

- `DROP TABLE customers CASCADE;`
- Früher war „DOS“ noch ein Betriebssystem ;-)

ProOpen

„Diebe“

- Deine Daten sind wertvoll
  - Für dich selbst
  - Für deine „Marktbegleiter“
  - Oder andere Interessierte
- Kunden, Umsätze, etc.
- Der Reputationsverlust kann noch teurer sein!

  Außer man ist eine Hotelkette ;-)

Und dann gibt es...

- ... diesen netten jungen Mann
- und hunderte von seinen Freunden...




WANTED BY THE FBI

MAKSIM VIKTOROVICH YAKUBETS

Conspiracy; Conspiracy to Commit Fraud; Wire Fraud; Bank Fraud; Intentional Damage to a Computer




Erpresser - Ransomware

- Sind Diebe
- Und Zerstörer
- Und bedrohen dich mit
  - Veröffentlichung / Verkauf
  - Verlust deiner Daten
  - Reputationsverlust

Was schützt uns?

- Sichere Systeme ;-)
- Backups!

- Aber das wissen auch die Freunde aus der Ransomware-Schiene ;-)

PgBackRest security <3

- Folgt man dem User guide von pgBackRest, ist schon sehr viel Gutes getan!
- Aber man kann das auf die Spitze treiben.

ProOpen

Schauen wir uns Szenarios an

- On-premise Datenbank
- Backup erfolgt mit pgBackRest
- Auf ein S3 Storage
  - Erstmal egal, wo!

Warum S3?

- Man kann durchaus geteilter Meinung sein, ob PostgreSQL in der Cloud eine gute Idee ist*
- Als Ablageort für Daten ist sie aber auf jeden Fall gut geeignet!

* Können wir uns gerne bei einem Getränk drüber austauschen!

Minimal feasible installation

- pgbackrest.conf:

```
repo1-s3-bucket=demo-bucket
repo1-s3-endpoint=s3.us-east-
1.amazonaws.com
repo1-s3-key=accessKey
repo1-s3-key-secret=verySecretKey
repo1-s3-region=us-east-1
repo1-type=s3
```

ProOpen

Angriff A) „Zerstörer“

- Wir haben Backups! Yay!
- Sicherheitslücke schließen
- Datenbank wiederherstellen
- fäddisch! ;-)

ProOpen

Oder deine Software ist richtig kaputt

- Und deine App arbeitet als superuser
- Dann ist dein Backup vielleicht auch weg!
- `ALTER SYSTEM SET archive_command TO ‘pgbackrest --stanza=my_stanza -stanza-delete‘; SELECT pg_reload_conf(); SELECT pg_switch_wal();`

Angriff B) „Dieb“

- Dein SIEM wird bestimmt merken, wenn jemand alle Daten rausträgt. Oder? ODER?
- Oder wenigstens fällt der gestiegene Traffic auf?
- Lösung:
  - Unterbinden
  - s.o.

ProOpen

Oder deine Software ist richtig kaputt

- Und deine App arbeitet als superuser oder so
- `SELECT pg_read_file( '/etc/pgbackrest.conf');`
- Jetzt hat der Angreifer deine S3 und kann dein Backup dort unbemerkt (?) raustragen
- Oder sogar eine Replika aufsetzen!

Cloud security... uhm...

- Aber wir würden ja keine Daten unverschlüsselt in die Cloud legen
- Oder?
- ODER?
- Erinnert sich jemand an den Vorfall bei Enterprise Rent A Car?

Also verschlüsseln!

- Ist nicht schwer:

```
repo1-cipher-pass=very_secret_and_long
repo1-cipher-type=aes-256-cbc
```

- Just do it!

Jetzt sind wir sicher! Oder?

- Ok, also Amazon kann dein Backup jetzt nicht mehr lesen…
- Aber der „Dieb“ hat deine `pgbackrest.conf` gesehen...

Angriff C) Ransomware

- Ist eigentlich A) plus B)
- Der Erpresser will
  - Deine Daten stehlen (mind. teilweise)
  - Deine Daten zerstören (durch Verschlüsselung)
  - Deine Backups zerstören oder verschlüsseln

Maßnahme #1

- Zieh eine zusätzliche Ebene ein!
- Setz einen „repo host“ auf
  - kann eine winzige VM sein!
- Repo host kennt die S3 Zugangsdaten
- DB Server kennt die encryption passphrase
- Repo host ist so „locked down“ wie irgend möglich
  - Kein LDAP/AD, „need to know“-Level Zugriff, ...

Der Zerstörer

- Bekommt evtl. die encryption passphrase
- Aber nicht die S3 Zugangsdaten

- Ergo: wir haben ein Backup!

Der Dieb - kommt in die DB

- Kann natürlich immer noch die Daten direkt aus der DB abgreifen (→ SIEM!)
- Aber nicht mehr vom S3

Der Dieb - hackt den Repo Host

- Hat die S3 Zugangsdaten
- Aber nicht die encryption passphrase!

Ein repo host ist also quasi „2FA“

- Um wirklich Schaden anzurichten, muss ein Angreifer sowohl den DB-Server als auch den repo host aufbrechen
- Zwischen den beiden Servern ist ausschließlich „`pgbackrest`“ erlaubt
- Also sind wir sicher! Yay!

Noch nicht ganz!

- Spannenderweise kann auch bei Einsatz eines repo host der DB Server dies tun:

```
pgbackrest -stanza=my_stanza stanza-delete
```

- https://github.com/pgbackrest/pgbackrest/issues/1682

Maßnahme #2

- Einschränken, was der DB Server auf dem repo host ausführen darf

  Z.B. mit „`rpgbackrest`“ (ist im o.g. bug report):

```
#!/bin/bash
if [[ "$*" =~ 'stanza-' ]]; then
  echo "Forbidden stanza command given" >&2
  exit 1
fi
/usr/bin/pgbackrest $*
```

Last resort: Maßnahme #3

- Für den Fall, dass ein Angreifer wirklich DB und repo host aufmacht…
- Haben wir „versioning“ auf dem S3 Bucket
- Gelöschte Files bleiben als „Versionen“ erhalten
- Sinnvollerweise länger als die Backup-Frequenz!

Last resort: Maßnahme #3

- Falls es nicht offensichtlich ist...
- Der S3-User, den pgBackRest benutzt, sollte das nicht sehen können!
- Die Zugangsdaten, die die Versionen sehen können, gehören an einen sicheren Ort!

Letzte Worte

- Andere Object-Stores dürften auch funktionieren
  - Versionierung?
- Hochladen der S3-LifeCyclePolicy mit `s3cmd`
- Sehen der S3-Versionen mit aktuellem `rclone`
- Kapseln des `archive_command`:

```
archive_command = my_script %f %p
and make pgbackrest happy
```

ProOpen

Fragen?

nick@pro-open.de

```
<?xml version="1.0" ?>
<LifecycleConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
        <Rule>
                <ID>pgBackRest-expiry</ID>
                <Filter>
                        <Prefix/>
                </Filter>
                <Status>Enabled</Status>
                <NoncurrentVersionExpiration>
                        <NoncurrentDays>8</NoncurrentDays>
                </NoncurrentVersionExpiration>
        </Rule>
</LifecycleConfiguration>
```