Lasso

Version 4

Built at 2024-05-02T13:25:14

1 Lasso

EDB developed a small multi-platform application called Lasso. Lasso can safely gather relevant diagnostics data on a system where Postgres and other relevant supported software, such as Barman, is running.

You can also run Lasso on systems where Postgres isn't installed to gather all relevant information about the underlying operating system.

Consult the specific information for your operating system.

> **Important**
>
> Lasso can't affect your data. It gathers statistics and diagnostics information from your Postgres server, with imperceptible effects on the workload. *No actual data* from the rows of your Postgres user tables is gathered. [^1]

[^1]: Gathered diagnostics data files are available for your inspection.

Lasso is crucial for support operations because it allows EDB's engineers to have a centralized and standardized source of information about your system. This ability greatly improves our resolution times and quality of services.

2 Release notes

Lasso - Version 4.15.0

Released: 23 Apr 2024

Lasso Version 4.15.0 includes the following enhancements and bug fixes:

Type	Description	Addresses
Feature	Lasso now gathers information about the package origins: list of repositories, repository configuration and HTTP(S) proxies in use for package download, if any.	DC-31
Feature	Lasso now gathers information about the EPAS code packages, including functions and procedures inside the packages.	DC-320
Feature	Packages for Debian 12 ("Bookworm")	DC-888
Improvement	Lasso now shows a hint message if connecting to the database with an user that doesn't have access to the custom schema the edb_wait_states extension was installed on	DC-977
Bug fix	Fix issue where Lasso was trying to set lock_timeout on PostgreSQL older than 9.3	DC-219
Doc improvement	Lasso bundle is no longer mentioned in the Lasso documentation and Knowledge Base Articles	DC-885

Lasso - Version 4.14.0

Released: 05 Mar 2024

Lasso Version 4.14.0 includes the following enhancements and bug fixes:

Type	Description	Addresses
Feature	Lasso now collects device mapper (`dm-X`) mappings for `iostat` on non-container instances.	DC-970
Improvement	Changed message about the lock file detected.	DC-974
Bug fix	Fixed issue on lock file removal for some exception cases.	DC-947
Bug fix	Fixed detection issue on non-Redwood instances for `SYS.ALL_PART_*` relations.	DC-973
Bug fix	Removed the `tools/pem` directory when there's no PEM agent service.	DC-859
Doc improvement	Added requirements for running Lasso on Windows Server.	DC-250
Doc improvement	Added clarification related to the usage of `libpq` and `.pgpass` .	DC-292

Lasso - Version 4.13.0

Released: 08 Feb 2024

Lasso Version 4.13.0 includes the following enhancements and bug fixes:

Type	Description	Addresses
Feature	Added Lasso packages for SLES 15.	DC-288
Feature	Added new `--info` option to display all collector information with revision/version.	DC-943
Feature	Added support for `edb_queue` .	DC-965
Feature	Added command-line arguments to specify the range for `edb_wait_states` : `--sampling-start` and `--sampling-end` .	DC-951
Feature	Now gather server information from `edb_wait_states` .	DC-953
Feature	Now gather operating system information from `edb_wait_states` .	DC-954
Feature	Now gather load profile from `edb_wait_states` .	DC-955
Feature	Now gather top wait events from `edb_wait_states` .	DC-956
Feature	Now gather top SQL statements from `edb_wait_states` .	DC-957
Feature	Now gather transaction stats from `edb_wait_states` .	DC-958
Feature	Now gather WAL stats from `edb_wait_states` .	DC-959
Feature	Now gather shared buffers stats from `edb_wait_states` .	DC-960
Feature	Now gather tuple stats from `edb_wait_states` .	DC-961
Feature	Now gather temp files stats from `edb_wait_states` .	DC-962
Feature	Now gather information about user sessions from `edb_wait_states` .	DC-963
Feature	Now gather information about database settings from `edb_wait_states` .	DC-964
Feature	Added documentation about `edb_wait_states` to the Lasso docs	DC-972
Bug fix	`lasso --version` can now be executed without a config file and the output was simplified.	RT100700, DC-943
Bug fix	Fixed an issue such that Lasso no longer fetches `/proc` information if running from inside a container.	RT101043, DC-968

Type	Description	Addresses
Bug fix	Added `SYS` schema qualifier for the `ALL_PART_*` relations.	RT101043, DC-969

Lasso - Version 4.12.0

Released: 09 Jan 2024

Lasso Version 4.12.0 includes the following enhancements and bug fixes:

Type	Description	Addresses
Feature	Now produce the file `postgresql/conns_per_second.out`, which shows the current rate of new connections established to the database during an observation period of 3 seconds.	DC-892
Feature	Now produce the file `postgresql/running_activity_maxage.out`, which shows the age of the oldest running backend, transaction, and query in the database.	DC-893
Bug fix	Fixed an issue where extremely large data files were allowed to be included in the Lasso report. In some situations, this condition caused elevated memory usage. Now, by default any data file larger than 500 MB is discarded (configurable with a new command line argument `--local-size-limit`) with a corresponding entry in the new `contents.data` file.	RT100724, DC-944
Bug fix	Fixed an issue where Lasso RPM packages were marked as obsolete even if they were the latest version.	RT100304, DC-899
Bug fix	Fixed an issue where the Lasso lock file wasn't being deleted at the end of the execution.	DC-894

> **Deprecation notice**
>
> - PostgreSQL 11 is now EOL and considered deprecated (DC-945).

Lasso - Version 4.11.0

Released: 14 Dec 2023

Lasso Version 4.11.0 includes the following enhancements and bug fixes:

Type	Description	Addresses
Feature	Implemented EDB Postgres Advanced Server and 2ndQPG flavor detection.	DC-875
Feature	Now gathering EDB Postgres Advanced Server Catalog - Views related to partitioned tables in Oracle syntax.	(DC-266)
Feature	Added support for PostgreSQL 16.	DC-887
Feature	Added support for RHEL9.	DC-802, DC-803

Type	Description	Addresses
Bug fix	Fixed situation when `ls pg_data` command takes much time to complete with extremely large number of files. Added a default `timeout` of `120s` for this case.	DC-845

Lasso - Version 4.10.0

Released: 07 Nov 2023

Lasso Version 4.10.0 includes the following enhancements and bug fixes:

!!! Deprecation Deprecated Ubuntu Bionic (18.04) as it is EOL since June 2023. (DC-877)

Type	Description	Addresses
Feature	Implemented a feature (filelock) to not rely on an EPEL dependency.	DC-863

Lasso - Version 4.9.0

Released: 17 Oct 2023

Lasso Version 4.9.0 includes the following enhancements and bug fixes:

Type	Description	Addresses
Feature	Added limits for the postmaster process.	DC-162
Feature	Added progress reporting views.	(DC-170)
Feature	Added waits profiling by way of `pg_stat_activity`.	DC-841
Bug fix	Fixed issue related to the report filename when the hostname is a socket path.	DC-769
Bug fix	Fixed issue for the SET ROLE syntax.	DC-842
Bug fix	Fixed issue on hostname from config file.	DC-864

Lasso - Version 4.8.0

Released: 06 Jul 2023

Lasso Version 4.8.0 includes the following enhancements and bug fixes:

Type	Description	Addresses
Feature	Added support for PGD Proxy.	
Feature	PGD Proxy support: Added new option `--pgd-proxy-configuration` for custom installations of pgd-proxy (Default: `/etc/edb/pgd-proxy/pgd-proxy-config.yml`).	DC-696

Type	Description	Addresses
Feature	PGD Proxy support: Now collect the contents of the PGD Proxy configuration file.	DC-697
Feature	PGD Proxy support: Now collect the output of `systemctl status pgd-proxy`.	DC-698
Feature	PGD Proxy support: Now collect the output of `systemctl cat pgd-proxy`.	DC-699

Lasso - Version 4.7.0

Released: 01 Jun 2023

Lasso Version 4.7.0 includes the following enhancements and bug fixes:

Type	Description	Addresses
Feature	Now collect output of `systemd-detect-virt`.	DC-76
Feature	Now collect the output of `lsmod`.	DC-197
Feature	Now collect the content of `harp.cluster.init.yml`	DC-628
Security fix	Redacted password for Lasso with `--password` seen in `ps` and `top` outputs.	DC-627
Bug fix	Fixed a bug whereby it wasn't possible to run `lasso --help` without a configuration file.	DC-490

Lasso - Version 4.6.0

Released: 06 Dec 2022

Lasso Version 4.6.0 includes the following enhancements and bug fixes:

Type	Description	Addresses
Feature	Lasso packages are now open to all customers through EDB repository (https://www.enterprisedb.com/repos).	DC-443
Enhancement	Updated the main Lasso Knowledge Base article to reflect the installation from packages and the configuration file.	KB-54
Bug fix	Improved Debian/Ubuntu packages to comply with the Lintian.	DC-435

Lasso - Version 4.5.0

Released: 15 Nov 2022

Lasso Version 4.5.0 includes the following enhancements and bug fixes:

Type	Description	Addresses
Enhancement	Added support for PostgreSQL 15.	DC-428
Enhancement	Improved execution on situations where there is a short timeout or when Lasso is executed on the standby.	DC-169

Lasso - Version 4.4.0

Released: 22 Sep 2022

Lasso Version 4.4.0 includes the following enhancements and bug fixes:

Type	Description	Addresses
Enhancement	Now collect output of `pg_shmem_allocations` in Postgres version 13 and later.	DC-195
Enhancement	Documented options for `edb-lasso.conf`.	DC-308

Lasso - Version 4.3.0

Released: 25 Aug 2022

Lasso Version 4.3.0 includes the following enhancements and bug fixes:

Type	Description	Addresses
Enhancement	Added support for `etcd`.	DC-367, DC-371, DC-370, DC-369, DC-368
Bug fix	Fixed a bug and now gather databases and tablespaces sizes regardless of `CONNECT` privilege.	DC-325

Lasso - Version 4.2.0

Released: 18 May 2022

Lasso Version 4.2.0 includes the following enhancements and bug fixes:

Type	Description	Addresses
Enhancement	Now collect systemd information for HARP.	DC-331
Enhancement	Now collect configuration files included in `pgbouncer.ini`.	DC-322
Enhancement	Added support to allow the user to specify custom paths to pgbouncer configuration files.	DC-321
Enhancement	Now gather etcdctl endpoint commands, if this is the consensus database.	DC-309
Enhancement	Added lasso option to specify a different path for harp's `config.yml`.	DC-306
Enhancement	Added helper function to get harp configuration file.	DC-305

Type	Description	Addresses
Enhancement	Added ability to get harpctl outputs.	DC-304
Enhancement	Added ability to get HARP `config.yml`.	DC-303
Enhancement	Added ability to identify all the pgbouncer instances on the node.	DC-157
Enhancement	Now collect systemctl output for pgbouncer.	DC-156
Enhancement	Now collect `pgbouncer.ini` configuration file.	DC-151
Other	Fixed SLES smoke tests that are failing.	DC-349
Other	Added support for Ubuntu 22.04.	DC-284
Bug fix	Now redact sslpassword.	DC-348

Lasso - Version 4.1.1

Released: 10 May 2022

Lasso Version 4.1.1 includes the following enhancements and bug fixes:

Type	Description	Addresses
Bug fix	Removed the psycopg2-binary warning from the bundle execution.	

Lasso - Version 4.1.0

Released: 03 May 2022

Lasso Version 4.1.0 includes the following enhancements and bug fixes:

Type	Description	Addresses
Enhancement	Built PPC64le Lasso bundle.	DC-276
Enhancement	Added `.exe` extension to Windows artifacts.	DC-285
Other	Added smoke tests for Almalinux 8 (x86_64).	DC-294
Other	Added smoke tests for RHEL (x86_64 and PPC64le).	DC-295
Other	Added smoke tests for SLES 12 (x86_64 and PPC64le).	DC-296
Other	Added smoke tests for SLES 15 (x86_64 and PPC64le).	DC-297
Bug fix	Lasso now uses MD5 hashing of tarball in FIPS enable systems.	DC-310

Lasso - Version 4.0.0

Released: 30 Mar 2022

Lasso Version 4.0.0 includes the following enhancements and bug fixes:

Type	Description	Addresses
Feature	Rebranded Data Collector to Lasso, the diagnostic tool from EDB.	
Feature	Rebranding: Updated copyright notes.	DC-239
Feature	Rebranding: Updated license on all files.	DC-228
Feature	Rebranding: Changed the name of the bundle to reflect the new brand of the tool.	DC-208
Feature	Rebranding: Generated output tarball now has the new brand and is identified with EDB.	DC-49
Feature	Rebranding: The connection Lasso creates to the database now sets the `application_name` to `edb-lasso` to identify the process Lasso is running.	DC-222
Feature	Rebranding: Updated the documentation to reflect the new brand name.	DC-184
Feature	Added a report that gathers output from `pg_replication_origin_status` to help with logical replication diagnostics. This includes native logical replication and others like BDR or xDB.	DC-232

3 Installing Lasso

EDB distributes the application through the EDB website and grants usage to customers using a *company token* as a means of authentication.

Linux

You can install Lasso on any major supported Linux distribution by following the corresponding Linux installation option for your system from the EDB Downloads page. Or, you can install it directly from the EDB repositories for Linux page. Choose Lasso and following the installation instructions.

After installing the EDB repository for your subscription on your system, you can install Lasso on Linux using the package manager tool for your Linux distribution. Examples include (but aren't limited to):

- Debian / Ubuntu:

```
apt install edb-lasso
```

- RHEL / CentOS / Oracle Linux 7:

```
yum install edb-lasso
```

- RHEL / Rocky / AlmaLinux / Oracle Linux 8+:

```
dnf install edb-lasso
```

If your system has internet access, then installing the `edb-lasso` package using your package manager installs these Lasso dependencies:

- `python3`
- `python3-psycopg2`
- `python3-setuptools`

- `python3-psutil`

If your system doesn't have internet access, then you need to download these dependencies and install them manually. Consult EDB Support for more details.

Windows

Lasso for Windows is a single-file binary executable called `lasso-windows-X.Y.Z.exe`, where `X.Y.Z` is the current Lasso version. You can download it from the EDB Downloads page.

On Windows, Lasso doesn't require installing any dependencies, but Lasso can run only on Windows Server 2019 or later.

Configuration file

If you try to run Lasso without a configuration file, this error occurs:

```
    ERROR: no configuration file for Lasso could be found. Please create a configuration file and try again.
```

Lasso requires a configuration file, which can be one of the following options. (It uses the first match.)

- `./edb-lasso.cfg` in the same directory where Lasso is running
- `./edb-lasso.conf` in the same directory where Lasso is running
- `$HOME/.edb-lasso.conf`
- `/etc/edb-lasso.conf`

The minimum configuration file looks like this:

```
[customer]
id=XXXXX
token=YYYYYYYYY
```

Replace the `id` and `token` value with the information found in your company page in the Support Portal. (In the left menu bar, select **Company info > Company**.) Enter the **Company code** value from this page in the `id` field in the configuration file. Enter the **Token** value in the `token` field in the configuration file.

> **Important**
>
> A configuration file for Lasso is mandatory and must contain at least the customer id and token.

For more details about the Lasso configuration, see Configuration.

Executing Lasso

After installing Lasso and creating an appropriate configuration file, a standard Lasso execution on Linux is:

```
    lasso
```

On Windows, to run Lasso, you need to do the following:

1. In the **Start** menu search field, enter `cmd`.
2. On the **Start** menu, right-click **Command Prompt** and select **Run as administrator**.
3. If Windows prompts you to allow Command Prompt to make changes in the system, select**Yes**.
4. At the Command Prompt, `cd` to the directory where the Lasso `.exe` file is located.
5. Execute the command `lasso-windows-X.Y.Z.exe`, where `X.Y.Z` is the Lasso version.

```
    lasso-windows-X.Y.Z.exe -H IP_ADDRESS -p PORT --password PASSWORD DATABASE_NAME USER_NAME
```

Alternatively, if you fill in the settings under the `postgresql` section in the configuration file telling Lasso how to connect to Postgres, then you can execute Lasso from the Windows Explorer. Right-click the `.exe` file, and then select **Run as administrator**.

For more details on the many Lasso options, see Usage.

4 Usage

Lasso uses the `libpq` environment variables to get the Postgres connection parameters. You can find the list of the environment variables in the PostgreSQL documentation.

You can also pass the connection parameters, among other options, as command line arguments:

```
usage: lasso [-h] [-H HOST_NAME] [-p PORT] [--password PASSWORD]
             [--lock-timeout LOCK_TIMEOUT]
             [--statement-timeout STATEMENT_TIMEOUT]
             [--sampling-start SAMPLING_START] [--sampling-end SAMPLING_END]
             [--local-size-limit LOCAL_SIZE_LIMIT] [--bindir BINDIR]
             [--depth [{surface,shallow,deep}]]
             [--describe [{short,json,full}]] [--version] [--info]
             [--latest-version] [--system-only | --barman]
             [--barman-configuration BARMAN_CONFIGURATION]
             [--repmgr-configuration REPMGR_CONFIGURATION]
             [--efm-configuration EFM_CONFIGURATION]
             [--xdb-pubserver-configuration XDB_PUBSERVER_CONFIGURATION]
             [--xdb-subserver-configuration XDB_SUBSERVER_CONFIGURATION]
             [--pgbouncer-configuration PGBOUNCER_CONFIGURATION]
             [--harp-configuration HARP_CONFIGURATION]
             [--etcd-configuration ETCD_CONFIGURATION]
             [--pgd-proxy-configuration PGD_PROXY_CONFIGURATION] [--upload]
             [--keep-report] [--is-latest-version]
             [dbname] [user]

EDB Lasso (for 2ndquadrant)

positional arguments:
  dbname                Database name to connect to (default user)
  user                  Database user name (default user)

optional arguments:
  -h, --help            show this help message and exit
  -H HOST_NAME, --host-name HOST_NAME
                        Database host name or socket directory (default local
```

```
                      socket)
-p PORT, --port PORT  Database server port (default 5432)
--password PASSWORD   Database server password
--lock-timeout LOCK_TIMEOUT
                      Database connection lock timeout (default 3s)
--statement-timeout STATEMENT_TIMEOUT
                      Database connection statement timeout (default 5min)
--sampling-start SAMPLING_START
                      Start of the range to take sampling data from
                      edb_wait_states (default: timestamp 1 hour from now)
--sampling-end SAMPLING_END
                      End of the range to take sampling data from
                      edb_wait_states (default: timestamp now)
--local-size-limit LOCAL_SIZE_LIMIT
                      Size limit, in bytes, of each file inside the tarball
                      (minimum 1024, default 524288000)
--bindir BINDIR       PostgreSQL binaries directory (autodetect by default)
--depth [{surface,shallow,deep}]
                      Depth of the report (default deep)
--describe [{short,json,full}]
                      Describes every single module, in terms of action and
                      output
--version             Shows Lasso version
--info                Shows Lasso modules revision
--latest-version      Shows the latest available version of EDB Lasso, taken
                      from the EDB Web Services
--system-only         Gather only system-related information - without
                      requiring a PostgreSQL connection
--barman              Gather Barman status, enabled by default when Lasso is
                      run as 'barman' user and the executable exists.
                      Defaults to False. When Barman reporting is enabled,
                      we do not gather PostgreSQL related information
--barman-configuration BARMAN_CONFIGURATION
                      Barman configuration file. By default use the native
                      algorithm in Barman to find the configuration file.
                      Valid only if Barman reporting is enabled
--repmgr-configuration REPMGR_CONFIGURATION
                      Path to the repmgr.conf file, if using a non-default
                      path
--efm-configuration EFM_CONFIGURATION
                      Path to the EFM properties file, if using a non-
                      default path
--xdb-pubserver-configuration XDB_PUBSERVER_CONFIGURATION
                      Path to the xDB publication server configuration file,
                      if using a non-default path
--xdb-subserver-configuration XDB_SUBSERVER_CONFIGURATION
                      Path to the xDB subscription server configuration
                      file, if using a non-default path
--pgbouncer-configuration PGBOUNCER_CONFIGURATION
                      Path to the pgbouncer.ini file. You can specify
                      multiple files separated by comma
--harp-configuration HARP_CONFIGURATION
                      Path to the config.yml file, if using a non-default
                      path
--etcd-configuration ETCD_CONFIGURATION
                      Path to the etcd.conf file, if using a non-default
                      path
--pgd-proxy-configuration PGD_PROXY_CONFIGURATION
                      Path to the pgd-proxy-config.yml file, if using a non-
```

```
                      default path
--upload              Report tarball file is sent to EDB at the end of the
                      execution. The file will be removed if successfully
                      uploaded unless --keep-report is specified
--keep-report         Keep a local copy of the report even after a
                      successful upload to EDB
--is-latest-version   Only check if this is the latest available version of
                      Lasso and returns exit code 0 if this is the latest
                      version and 1 otherwise
```

For details about how to use each of these arguments, see Report types.

5 Configuration

Besides being mandatory for the customer `id` and `token`, the Lasso configuration file also allows you to omit most of the command line options that your environment might require. (You can see a list of all the command-line options in Usage.)

Lasso looks for configuration files in the following paths, in this order, and uses the first match:

1. `./edb-lasso.cfg` in the same directory where Lasso is running
2. `./edb-lasso.conf` in the same directory where Lasso is running
3. `$HOME/.edb-lasso.conf`
4. `/etc/edb-lasso.conf`

A template file for the configuration file looks like this:

```
 ; Lasso template configuration file
;
; Copyright (C) EnterpriseDB UK Limited 2015-2024 - All Rights Reserved.
; Licensed only for use with an EnterpriseDB subscription

[customer]
; the "Company code" from the customer page in the EDB Portal
id=
; the "Token" from the customer page in the EDB Portal
token=
; the depth of the Lasso report. Must be one between: surface, shallow and deep
depth=deep

[postgresql]
; Lasso uses the following connection string parameters to connect to your
; PostgreSQL cluster and take diagnostics. By default it will attempt a peer
; connection to socket under directory /var/run/postgresql
; dbname=
; user=
; port=
; hostname can be a host or a socket directory
; hostname=/var/run/postgresql
; password=

; session variable to control timeout on lock waits
; lock_timeout=3s
; session variable to control statement execution time
; statement_timeout=5min
```

```
[environment]
; PostgreSQL binaries directory. Lasso will try to detect this automatically if
; not set
; bindir=
; if the host is able to access the internet and reach the EDB servers to upload
; Lasso reports
; external_access=yes

[barman]
; path to barman.conf, if using a non-default one
; configuration=

[repmgr]
; path to repmgr.conf, if using a non-default one
; configuration=

[efm]
; path to efm.properties, if using a non-default one
; configuration=

[xdb]
; path to xdb_pubserver.conf, if using a non-default one
; pubserver_configuration=
; path to xdb_subserver.conf, if using a non-default one
; subserver_configuration=

[pgbouncer]
; path to pgbouncer.ini, if using a non-default one
; configuration=

[harp]
; path to config.yml, if using a non-default one
; configuration=

[pgd-proxy]
; path to pgd-proxy-config.yml, if using a non-default one
; configuration=
```

You can use this template to set up your configuration file. Uncomment the desired parameters and set their values according to your environment.

On Linux, you can find that same template configuration file at `/etc/edb-lasso.conf.templ`.

You can see more details about how each of these arguments is used in Report types.

6 Security considerations

When running queries in the database, Lasso tries to use a role that has enough privileges to gather the required information from the tool from which metrics are being gathered.

The following are the tools and the roles that Lasso tries to use for each of them. Lasso tries to use the first available role in each tool role list.*Initial connection role* means the role that was provided through Lasso CLI when running the tool—usually postgres or enterprisedb.

- PostgreSQL:
  - pg_monitor
  - Initial connection role

- PgLogical:
  - pglogical_superuser
  - Initial connection role
- PGD:
  - bdr_monitor
  - Initial connection role
- PEM:
  - pem_user
  - Initial connection role
- Repmgr:
  - Initial connection role
- xDB:
  - Initial connection role

Most of the PGD gatherings try using the bdr_monitor role. However, the one in charge of gathering conflicts tries to use the role bdr_read_all_conflicts for that purpose. That's the only exception.

In any of the cases, it uses a read-only transaction while querying metrics and configurations from the database.

7 Report types

Local/full PostgreSQL report

This type of report is useful when the server has a PostgreSQL cluster that's up, running, and accessible. In addition to all information that's gathered in the system-only report, it also gathers configurations and metrics from the PostgreSQL instance.

For this report, you need to run Lasso as the postgres or enterprisedb user.

> **Important**
>
> The same user that runs the Postgres server process must run the command. The user must have permission to write in the current working directory.

A standard Lasso run consists of:

```
lasso
```

By default, Lasso tries to connect using Unix sockets, the default Postgres port (5432), the default database, and the user name (postgres). You can specify different values at the command line:

```
lasso -p PORT DATABASE_NAME USER_NAME
```

By default, Lasso looks for a Unix socket under `/var/run/postgresql/`, but you can specify any socket listed by `SHOW unix_socket_directories` using the `-H` argument. For example, usually a Lasso report for the EDB Postgres Advanced Server is gathered with:

```
lasso -H /tmp -p 5444 edb
```

If a password is required, it's a common practice to create a `~/.pgpass` file for the postgres or enterprisedb operating system user. You can find more details about the `.pgpass` file in the PostgreSQL documentation. As Lasso connects to the database using `libpq`, then Lasso uses the `.pgpass` file as appropriate.

If the `.pgpass` file isn't found or a corresponding entry for the database connection doesn't exist in the `.pgpass` file, then Lasso prompts for a password. Alternatively, you can specify a password using the `PGPASSWORD` environment variable:

```
    PGPASSWORD=xxxx lasso -p PORT DATABASE_NAME USER_NAME
```

On Windows, typically you run Lasso like this:

```
    lasso-windows-X.Y.Z.exe -H IP_ADDRESS -p PORT --password PASSWORD DATABASE_NAME USER_NAME
```

For a list of all command line arguments detailed in Usage, run:

```
    lasso --help
```

You can specify Postgres settings in the Lasso configuration file, under the `postgresql` setting. For more details, see Configuration.

> **Important**
>
> When run on a local instance, Lasso requires a user that belongs to the pg_monitor role, which is available in Postgres 10 and later. For earlier versions of Postgres, you must run as a superuser. For more information, see Default Roles in the PostgreSQL documentation.

Lasso transparently gathers information related to all supported versions of EDB Postgres Distributed (PGD) and pglogical.

Remote/database-only PostgreSQL report

If you need to gather data from a remote PostgreSQL instance, you can run a remote PostgreSQL report.

As explained in Local/full PostgreSQL report, you can change the connection-string parameters of Lasso. For example, to gather a PostgreSQL report from a remote server located in the IP 192.168.0.10 running on port 5433, you can invoke Lasso from another server as follows:

```
    lasso -H 192.168.0.10 -p 5433
```

This type of report brings only PostgreSQL-related info from the PostgreSQL instance that Lasso was connected to. It means it can't gather files like `postgresql.conf` and `pg_hba.conf`. It also can't gather configurations and metrics from the underlying operational system or information from PostgreSQL-related tools, like EFM configuration files. With that in mind, we recommend running a local PostgreSQL report, if possible.

> **Important**
>
> This is the only option for gathering PostgreSQL information from an instance hosted on a DBaaS provider such as EDB Big Animal or Amazon RDS.

> **Important**
>
> This is the only option for gathering PostgreSQL information from an instance running under Kubernetes.

Barman report

Lasso can run on systems where Barman is installed. In that case, it gathers Barman-related information, as well as system information, that helps during analysis.

> **Important**
>
> You must run the command as the same user that runs the Barman process, and you must have enough permission to write in the current working directory. If you installed Barman using RPM/DEB packages, Barman is configured to run as the barman user.

The Barman report is enabled by default if Lasso runs as the barman user and looks for configuration files in the expected locations, typically `/etc/barman.conf`. Execute `man 5 barman` for details.

In general, if you installed Barman using EDB-certified RPM and DEB packages, all you need to do is execute Lasso as the barman user.

Managing custom installations of Barman

If you have custom installations of Barman, you can enable the barman report by passing the `--barman` option to Lasso:

```
lasso --barman
```

You can also point to a specific global configuration file by using the `--barman-configuration` option:

```
lasso --barman-configuration /opt/barman/barman.conf
```

Replication Manager (repmgr) report

Lasso can run on systems where repmgr is installed. In that case, it gathers repmgr-related information.

In general, if you installed repmgr using EDB-certified RPM and DEB packages, all you need to do is execute Lasso.

How Lasso finds the repmgr configuration file

Lasso uses the following approach while trying to identify the repmgr configuration file. It uses the first one it finds.

1. Use the one provided to the `--repmgr-configuration` option, if given.
2. Check the paths provided by packages. For example, in CentOS/RHEL packages, the configuration file is usually put under `/etc/repmgr/<PG_VERSION>/repmgr.conf`. Lasso inspects those folders, if they exist, in descending order, and uses the first repmgr configuration file that it finds.
3. Check if a `repmgr.conf` file exists in the current directory.
4. Check if a `/etc/repmgr.conf` file exists.
5. Check the path given by `pg_config --sysconfdir` for the `repmgr.conf` file.
6. Check the output of `systemctl cat <service_name>`, where `<service_name>` is any service whose name contains `repmgr`, and try to get the `repmgr.conf` file from the service unit file. It checks the units in descending order, just as it does in approach 2, and uses the first match.

Managing custom installations of repmgr

You can point to a specific repmgr configuration file by using the `--repmgr-configuration` option:

```
    lasso --repmgr-configuration /opt/repmgr/repmgr.conf
```

We recommend this approach, as it guarantees Lasso will use the correct repmgr configuration file instead of trying to find it.

Postgres Enterprise Manager (PEM) report

Lasso can run on systems where PEM is installed. In that case, it also gathers PEM-related information, like some configuration files and information about services.

Lasso inspects the well-known paths for configuration files as well as well-known service names. If you have custom PEM installations, Lasso doesn't gather the related information.

Failover Manager (EFM) report

Lasso can run on systems where EFM is installed. In that case, it gathers EFM-related information.

In general, if you installed EFM using EDB-certified RPM and DEB packages, all you need to do is execute Lasso.

How Lasso finds the EFM configuration file

Lasso uses the following approach while trying to identify the EFM configuration file. It uses the first one it finds.

1. Use the one provided to the `--efm-configuration` option, if given.
2. Check the paths provided by packages. The configuration file is usually put under `/etc/edb/efm-<EFM_VERSION>/efm.properties`. Lasso inspects those folders, if they exist, in descending order, and uses the first EFM configuration file that it finds.
3. Check the output of `systemctl cat <service_name>`, where `<service_name>` is replaced with any service whose name starts with `edb-efm-`. Then try to get the `efm.properties` file from the service unit file. It checks the units in descending order, just as it does in approach 2, and uses the first match.

How Lasso finds the EFM binary

Lasso inspects the paths provided by packages. The binary file is usually put under `/usr/edb/efm-<EFM_VERSION>/bin/efm`. Lasso inspects these folders, if they exist, in descending order, and uses the first EFM binary file that it finds.

Managing custom installations of EFM

You can point to a specific EFM configuration file by using the `--efm-configuration` option:

```
    lasso --efm-configuration /opt/EFM/efm.properties
```

We recommend this approach, as it guarantees Lasso will use the correct EFM configuration file instead of trying to find it.

Replication Server (xDB) report

Lasso can run on systems where xDB is installed. In that case, it gathers xDB-related information.

In general, if you installed xDB using EDB-certified RPM packages or the bitrock installer, all you need to do is execute Lasso.

How Lasso finds the xDB configuration file

Lasso uses the following approach while trying to identify the xDB configuration files. It uses the first one it finds.

On Linux:

1. Use the ones provided to the `--xdb-pubserver-configuration` and `--xdb-subserver-configuration` options, if given.
2. Check the paths provided by xDB 7 RPM package and bitrock installer. The configuration files are usually put under `/etc/edb/xdb/etc`. Lasso inspects this folder, if it exists, and uses the `xdb_pubserver.conf` and `xdb_subserver.conf` files found under that folder.
3. Check the paths provided by xDB 6 RPM package and bitrock installer, which are usually `/usr/ppas-xdb-<XDB_VERSION>/etc` and `/opt/PostgreSQL/EnterpriseDB-xDBReplicationServer/etc`, respectively. Lasso inspects these folders and, if they exist, uses the `xdb_pubserver.conf` and `xdb_subserver.conf` files found under the folder. In the case of RPM packages, as there can be a lot of folders, they're inspected from newest XDB version to oldest.

On Windows:

1. Use the ones provided to the `--xdb-pubserver-configuration` and `--xdb-subserver-configuration` options, if given.
2. Check the paths provided by xDB 7 packages. The configuration files are usually put under `<drive>:\Program Files\edb\EnterpriseDB-xDBReplicationServer\etc`. Lasso inspects this folder, if it exists, and uses the `xdb_pubserver.conf` and `xdb_subserver.conf` files found under that folder.
3. Check the paths provided by xDB 7 packages. The configuration files are usually put under `<drive>:\Program Files\PostgreSQL\EnterpriseDB-xDBReplicationServer\etc`. Lasso inspects this folder, if it exists, and uses the `xdb_pubserver.conf` and `xdb_subserver.conf` files found under that folder.

How Lasso finds the xDB binary

Lasso uses the following approach while trying to identify the xDB binary file. It uses the first one it finds.

On Linux:

1. Check the paths provided by the xDB 7 RPM package and bitrock installer. The binary files are usually put under `/etc/edb/xdb/bin`. Lasso inspects this folder, if it exists, and uses the `edb-repcli.jar` file found under that folder.
2. Check the paths provided by the xDB 6 RPM package and bitrock installer, which are usually `/usr/ppas-xdb-<XDB_VERSION>/bin` and `/opt/PostgreSQL/EnterpriseDB-xDBReplicationServer/bin`, respectively. Lasso inspects these folders and, if they exist, uses the `edb-repcli.jar` file found under the folder. For RPM packages, as there can be a lot of folders, they're inspected from newest XDB version to oldest.

On Windows:

1. Check the paths provided by the xDB 7 packages. The binary files are usually put under `<drive>:\Program Files\edb\EnterpriseDB-xDBReplicationServer\bin`. Lasso inspects this folder, if it exists, and uses the `edb-repcli.jar` file found under that folder.
2. Check the paths provided by the xDB 7 packages. The binary files are usually put under `<drive>:\Program Files\PostgreSQL\EnterpriseDB-xDBReplicationServer\bin`. Lasso inspects this folder, if it exists, and uses the `edb-repcli.jar` file found under that folder.

Managing custom installations of xDB

You can point to a specific xDB publication or subscription server configuration file by using the `--xdb-pubserver-configuration` and `--xdb-subserver-configuration` options.

On Linux:

```
    lasso --xdb-pubserver-configuration /opt/xDB/pubserver.conf --xdb-subserver-configuration
/opt/xDB/subserver.conf
```

On Windows:

```
    lasso-windows-X.Y.Z.exe --xdb-pubserver-configuration C:\\xDB\pubserver.conf --xdb-subserver-
configuration C:\\xDB\subserver.conf
```

We recommend this approach, as it guarantees Lasso will use the correct xDB configuration files instead of trying to find it.

PgBouncer report

Lasso can run on systems where PgBouncer is installed. In that case, it gathers PgBouncer-related information.

In general, if you installed PgBouncer using EDB-certified RPM and DEB packages, and PgBouncer instances are up and running, all you need to do is execute Lasso.

How Lasso finds the PgBouncer configuration file

Lasso uses the list of running processes to identify the PgBouncer processes and uses the configuration file retrieved from there.

You can also specify one or more PgBouncer configuration files with the Lasso argument `--pgbouncer-configuration`. It's mainly useful in scenarios where a PgBouncer instance isn't running or if Lasso can't detect the configuration file from the process list.

Lasso collects information from all identified PgBouncer instances.

HARP report

Lasso can run on systems where HARP is installed. In that case, it gathers HARP-related information.

In general, if you installed HARP using EDB-certified RPM and DEB packages, all you need to do is execute Lasso.

How Lasso finds the HARP configuration file

Lasso uses the following approach while trying to identify the HARP configuration file. It uses the first one it finds.

1. Use the one provided to the `--harp-configuration` option, if given.
2. Check if the `/etc/harp/config.yml` file exists.

Managing custom installations of HARP

You can point to a specific HARP configuration file by using the `--harp-configuration` option:

```
    lasso --harp-configuration /opt/harp/config.yml
```

We recommend this approach, as it guarantees Lasso will use the correct HARP configuration file instead of trying to find it.

Running Lasso on a HARP proxy node

The HARP proxy nodes don't have a Postgres or EDB Postgres Advanced Server service running on them, as they're intended only as a proxy for connections coming in from the applications.

When using Lasso to gather information from a HARP proxy node, you must use the `--system-only` option so it doesn't try to connect to the database server.

PGD Proxy report

Lasso can be run on systems where PGD Proxy is installed. In that case, it gathers PGD Proxy-related information.

In general, if you installed PGD Proxy using EDB-certified RPM and DEB packages, all you need to do is execute Lasso.

How Lasso finds the PGD Proxy configuration file

Lasso uses the following approach while trying to identify the PGD Proxy configuration file. It uses the first one it finds.

1. Use the one provided through `--pgd-proxy-configuration` option, if given
2. Check if `/etc/edb/pgd-proxy/pgd-proxy-config.yml` file exists

Managing custom installations of PGD Proxy

You can point to a specific PGD Proxy configuration file by using the `--pgd-proxy-configuration` option:

```
    lasso --pgd-proxy-configuration /opt/pgd-proxy/config.yml
```

We recommend this approach, as it guarantees Lasso will use the correct PGD Proxy configuration file instead of trying to find it automatically.

Running Lasso on a PGD Proxy node

There are cases where the PGD Proxy nodes don't have a Postgres/EDB Postgres Advanced Server service running on them. They're intended only as a proxy for connections coming in from the applications.

When using Lasso to gather information from a standalone PGD Proxy node, you must use the `--system-only` option so it doesn't try to connect to the database server.

etcd report

Lasso can run on systems where etcd is installed. In that case, it gathers etcd-related information.

How Lasso finds the etcd configuration file

Lasso uses the following approach while trying to identify the etcd configuration file. It uses the first one it finds.

1. Use the one provided to the `--etcd-configuration` option, if given.
2. Check if the `/etc/etcd/etcd.conf` file exists.

System-only report

Lasso can also run on systems where Postgres or Barman aren't installed to gather all relevant information regarding the underlying operating system.

You can run a system-only report:

```
lasso --system-only
```

> **Important**
>
> Despite the argument being called `--system-only`, in this mode, Lasso gathers information about all the aforementioned tools except PostgreSQL/EDB Postgres Advanced Server and Barman.

edb_wait_states report

For each database where the `edb_wait_states` extension is installed, Lasso gathers important performance information from this extension and includes it in the `postgresql/dbs/DBNAME/edb_wait_states/` folder.

The `edb_wait_states` extension requires defining a sampling range for retrieving data, so Lasso exposes 2 arguments for defining the sampling range: `--sampling-start` and `--sampling-end`. For example:

```
lasso --sampling-start "2024-02-06 13:30:00+0000" --sampling-end "2024-02-06 14:00:00+0000"
```

In this case, the timestamps are in UTC. It's possible to specify a different timezone. For example:

```
lasso --sampling-start "2024-02-06 10:30:00-0300" --sampling-end "2024-02-06 11:00:00-0300"
```

These arguments are ignored if the `edb_wait_states` extension isn't installed.

If the `edb_wait_states` extension is installed and these arguments are omitted, then Lasso considers `--sampling-start` as 1 hour from now and `--sampling-end` as now.

For more details about the `edb_wait_states` extension, see the documentation.

8 Returning the Lasso report

The script produces a TAR file containing the gathered data in the directory where you executed Lasso. This file is the Lasso report.

You can also use Lasso on a server that has no Postgres installation. In that case, use the `--system-only` option to produce a TAR file that contains only system-related information.

Unless you're running Lasso on an isolated network system, without external access to EDB's infrastructure, you can upload the produced tarball directly through Lasso using the `--upload` option. For more information, see Servers accepting upload of reports.

Alternatively, you can attach the report to a specific support ticket through the Portal. Or, use the **Support operations > Upload report** menu from your company's page.

9 What do we gather in our Lasso report?

A Lasso report is designed to allow any EDB engineer to rapidly feel the pulse of your database by getting relevant information from your operating system and your Postgres server.

You can get a list with a detailed description of the information that's gathered from your system through `lasso` by entering:

```
    lasso --describe
```

If you need more information, contact your account manager at EDB.

Lasso informs its level of depth when scraping your system. There are three levels:

- **Surface**: Noninvasive gathering process, mainly used to perform a supportability check of the instance and focused on the current moment. In terms of Postgres, the focus is on the instance, its configuration, and its general health. This level is the minimal level of scraping.
- **Shallow**: Gathering of performance-related information, including basic trending data. In terms of Postgres data, its focus is moved down to the database. This level is suitable for support-incident resolution.
- **Deep**: Most detailed level of scraping, suitable for advanced and security health checks but also for severe support incidents. In terms of Postgres data, its focus is on tables and indexes.

10 Detailed breakdown of gathered data

GNU/Linux operating system

barman crontab/cron (`barman_crontab_cron`)

Output from `crontab -l` , if running as barman. Content of `/etc/cron.d/barman` , if it exists.

Report output:

- File `/linux/barman_cron.data` : Content of `/etc/cron.d/barman` , if it exists
- File `/linux/barman_crontab.data` : Output from `barman crontab -l` , if barman user

Depth: Surface

Security impact: Low — Might have entries in `crontab/cron` with sensitive data.

debug_sources (`debug_sources`)

Count files under `/usr/src/debug` to detect the applications whose source code is present in the system and facilitate live debugging.

Report output:

- File `/linux/debug_sources.data` : Sources for GNU debugger

Depth: Surface

Security impact: Low — No known security impact.

EFM CLI (`efm_cli`)

Get output of `efm cluster-status` command.

Report output:

- File `/tools/efm/cli/cluster_status.out` : Output of `efm cluster-status cluster_name` command

Depth: Surface

Security impact: Low — No known security impact.

EFM configuration (`efm_configuration`)

EFM properties and nodes configuration files.

Report output:

- File `/tools/efm/config/efm.nodes` : EFM nodes file
- File `/tools/efm/config/efm.properties` : EFM properties file

Depth: Surface

Security impact: Low — No known security impact.

EFM systemctl (`efm_systemctl`)

When EFM services are detected, collects status and cat of the corresponding services. Checks for any service whose name starts with `edb-efm-`.

Report output:

- File `/tools/efm/systemd/service_name_cat.data`: Output of `systemctl cat service_name`
- File `/tools/efm/systemd/service_name_status.data`: Output of `systemctl status service_name`

Depth: Surface

Security impact: Low — No known security impact.

etcd CLI (`etcd_cli`)

Gathers the output of some `etcdctl` commands, if `etcdctl` is available in the server. The commands are `endpoint status` and `endpoint health`.

Report output:

- File `/tools/etcd/cli/endpoint_status.out`: Output of `etcdctl endpoint status` command
- File `/tools/etcd/cli/endpoint_health.out`: Output of `etcdctl endpoint health` command

Depth: Surface

Security impact: Low — No known security impact.

etcd configuration (`etcd_configuration`)

Collects `etcd` configuration file that's found in the server.

Report output:

- File `/tools/etcd/config/basename`: `etcd` configuration file

Depth: Surface

Security impact: Low — No known security impact.

etcd systemctl (`etcd_systemctl`)

When etcd services are detected, collects status and cat of the corresponding services. Checks for any service whose name starts with `etcd`.

Report output:

- File `/tools/etcd/systemd/service_name_cat.data`: Output of 'systemctl cat service_name
- File `/tools/etcd/systemd/service_name_status.data`: Output of 'systemctl status service_name

Depth: Surface

Security impact: low — No known security impact.

HARP CLI (`harp_cli`)

Gathers output of a few `harpctl` command outputs using the `config.yml` file, which is found in the server. The commands are: `cluster` , `proxies` , `locations` , `nodes` , and `version` .

Report output:

- File `/tools/harp/cli/version.out` : Output of `harpctl -f conf_file_path version` command
- File `/tools/harp/cli/proxies.out` : Output of `harpctl -f conf_file_path get proxies -o yaml` command
- File `/tools/harp/cli/nodes.out` : Output of `harpctl -f conf_file_path get nodes -o yaml` command
- File `/tools/harp/cli/locations.out` : Output of `harpctl -f conf_file_path get locations -o yaml` command
- File `/tools/harp/cli/cluster.out` : Output of `harpctl -f conf_file_path get cluster -o yaml` command

Depth: Surface

Security impact: Low — No known security impact.

HARP configuration (`harp_configuration`)

Collects HARP configuration file that's found in the server.

Report output:

- File `/tools/harp/config/harp.cluster.init.yml` : HARP bootstrap configuration file
- File `/tools/harp/config/basename` : HARP configuration file

Depth: Surface

Security impact: Low — No known security impact.

HARP systemctl (`harp_systemctl`)

When HARP services are detected, collects status and cat of the corresponding services. Checks for any service whose name starts with `harp` .

Report output:

- File `/tools/harp/systemd/service_name_cat.data` : Output of `systemctl cat service_name`
- File `/tools/harp/systemd/service_name_status.data` : Output of `systemctl status service_name`

Depth: Surface

Security impact: Low — No known security impact.

Block devices layout (`linux_block_devices_layout`)

Information on block devices layout from the `lsblk` command.

Report output:

- File `/linux/lsbk.data`: `lsbk` command output

Depth: Surface

Security impact: Low — No known security impact.

Processor governor (`linux_cpu_governor`)

Processor scaling governor from the files in `/sys/devices/system/cpu`.

Report output:

- File `/linux/sys/energy_perf_bias.data`: Intel Performance and Energy Bias attributes
- File `/linux/sys/intel_pstate.data`: Intel pstate configuration
- File `/linux/sys/cpu_scaling_driver.data`: Available CPU scaling driver
- File `/linux/sys/cpu_scaling_available_governors.data`: Available CPU scaling governors
- File `/linux/sys/cpu_scaling_governor.data`: Active CPU scaling governor

Depth: Surface

Security impact: Low — No known security impact.

Mounted file systems and available space (`linux_devices_info`)

List-mounted file systems through the `mount` command and free space using `df`.

Report output:

- File `/linux/diskspace.data`: Amount of available disk space
- File `/linux/mount.data`: Output of the `mount` command

Depth: Surface

Security impact: Low — No known security impact.

File systems configuration (`linux_disk_configuration`)

Disk configuration obtained through the `/etc/fstab` file.

Report output:

- File `/linux/fstab.data`: Contents of /etc/fstab

Depth: Surface

Security impact: Low — No known security impact.

OS distribution, kernel, and device data (`linux_distro_collector`)

Information about the Linux distribution currently in use returned by the `lsb_release` command.

Report output:

- File `/linux/release.data` : Linux distribution currently in use
- File `/linux/release_source.data` : Name of the collected file or the executed command

Depth: Surface

Security impact: Low — No known security impact.

Hardware (`linux_hardware_info`)

Hardware info through `lspci` .

Report output:

- File `/linux/lspci.data` : Hardware info from `lspci`

Depth: Surface

Security impact: Low — No known security impact.

HTTP(s) proxies in use for package downloads (`linux_http_proxy_configuration`)

Gathers information about HTTP(s) proxies in use for package downloads. Passwords are redacted.

Report output:

- File `/linux/packages-yum-config-manager.data` : YUM configuration
- File `/linux/packages-dnf-config-manager.data` : DNF configuration
- File `/linux/etc_environment.data` : Contents of /etc/environment

Depth: Surface

Security Impact: *Low* — No known security impact.

Hypervisor (`linux_hypervisor_collector`)

Information about the type of virtualization used, as returned by the `systemd-detect-virt` command.

Report output:

- File `/linux/hypervisor.data` : Name of the collected file or the executed command

Depth: Surface

Security impact: Low — No known security impact.

Kernel (`linux_kernel_info`)

Kernel info, transparent huge pages status, and disk scheduler configuration. Obtained by combining the output of the commands `uname` and `ipcs` with the contents of the `/proc` and `/sys` file systems.

Report output:

- File `/linux/read_ahead.data` : Info on the read ahead
- File `/linux/schedulers.data` : Scheduler info from `/sys` dir
- File `/linux/sys/kernel_mm_transparent_hugepage.data` : Transparent huge pages info
- File `/linux/ipcs.data` : `ipcs` command output
- File `/linux/uname.data` : `uname` command output

Depth: Surface

Security impact: Low — No known security impact.

Kernel limits (`linux_kernel_limits`)

Configuration file for the `pam_limits` module.

Report output:

- File `/linux/limits.data` : Content of the `limits.conf` file

Depth: Surface

Security impact: Low — No known security impact.

Processor usage statistics (`linux_mpstat`)

Processor statistics from the `mpstat` command.

Report output:

- File `/linux/mpstat.data` : Output from `mpstat -P ALL 1 10`

Depth: Surface

Security impact: Low — No known security impact.

Network interfaces (`linux_network_interfaces`)

Network interface information from the `ip` and `ifconfig` commands.

Report output:

- File `/linux/ifconfig.data` : Output from `ifconfig`
- File `/linux/ip_address_list.data` : Output from `ip address list`

Depth: Surface

Security impact: Low — No known security impact.

Installed packages via rpm or dpkg (`linux_packages_info`)

Information about the system packages installed using `rpm` or `dpkg` .

Report output:

- File `/linux/packages-dpkg.data` : List of packages installed using `dpkg`
- File `/linux/packages-rpm.data` : List of packages installed using `rpm`

Depth: Surface

Security impact: Low — No known security impact.

Installed packages origins (`linux_packages_origin_info`)

Information about the packages origins.

Report output:

- File `/linux/packages-apt_conf.data` : `apt` configuration
- File `/linux/packages-apt-cache-policy.data` : `apt` configuration
- File `/linux/packages-apt-list-installed.data` : Repositories that were used to install packages
- File `/linux/packages-yum-repolist.data` : Repositories that are enabled in `yum`
- File `/linux/packages-dnf-module-list.data` : Repositories that are enabled in `dnf`
- File `/linux/packages-dnf-repolist.data` : Repositories that are enabled in `dnf`
- File `/linux/packages-yum-list-installed.data` : Repositories that were used to install packages
- File `/linux/packages-dnf-list-installed.data` : Repositories that were used to install packages

Depth: Surface

Security Impact: *Low* — No known security impact.

PostgreSQL disk layout (`linux_postgresql_disk_layout`)

List all files in the PostgreSQL data directory using `find` for links and `ls` for files.

Report output:

- File `/linux/pg_ls.data` : List of files inside the data directory
- File `/linux/pg_links.data` : List of links inside the data directory

Depth: Surface

Security impact: Low — No known security impact.

SELinux (`linux_sestatus`)

SELinux status from `sestatus` .

Report output:

- File `/linux/sestatus.data` : Output from `sestatus`

Depth: Surface

Security impact: Low — No known security impact.

System identification (`linux_system_identity`)

Collect hostname, network interfaces, system info (uname), system identifier, and release info.

Report output:

- File `/linux/id/system_release.data` : OS information from `/etc/system-release`
- File `/linux/id/os_release.data` : OS information from `/etc/os-release`
- File `/linux/id/machine_id.data` : Machine ID contained in `/etc/machine-id`
- File `/linux/id/uname.data` : Information about the running kernel
- File `/linux/id/hostname.data` : Fully qualified domain name
- File `/linux/id/interfaces.data` : Network addresses of the host

Depth: Surface

Security impact: Low — No known security impact.

dmesg and /proc information (`linux_system_info`)

System info from the contents of the `/proc` filesystem and through the output of `dmesg` command.

Report output:

- File `/linux/lsmod.data` : `lsmod` output
- File `/linux/dmesg_with_timestamp.data` : `Dmesg` output (human-readable timestamps)
- File `/linux/dmesg.data` : `dmesg` output
- File `/linux/proc/sys_net_ipv4.data` : Network info from `/proc`
- File `/linux/proc/sys_vm.data` : VM info from `/proc`

- File `/linux/proc/sys_kernel.data` : Kernel info from `/proc`
- File `/linux/vmstat.data` : VM statistics from `/proc`
- File `/linux/proc/mounts.data` : Mount points from `/proc`
- File `/linux/proc/uptime.data` : Uptime info from `/proc`
- File `/linux/proc/loadavg.data` : Load avg from `/proc`
- File `/linux/proc/meminfo.data` : Memory info from `/proc`

Depth: Surface

Security impact: Low — No known security impact.

System status — device mapper devices (`linux_system_status_dmdevices`)

Get information about device mapper devices.

Report output:

- File `/linux/lsdevmapper.data` : Information about `/dev/mapper` device mapper symlinks

Depth: Surface

Security impact: Low — No known security impact.

System status — iostat (`linux_system_status_iostat`)

System status from the `iostat` command.

Report output:

- File `/linux/iostat.data` : Info on I/O statistics

Depth: Surface

Security impact: Low — No known security impact.

System status — nfsiostat (`linux_system_status_nfsiostat`)

System status from the `nfsiostat` command.

Report output:

- File `/linux/nfsiostat.data` : nfs I/O statistics

Depth: Surface

Security impact: Low — No known security impact.

System status — ps (`linux_system_status_ps`)

System status from the `ps` command.

Report output:

- File `/linux/ps.data`: Active processes info

Depth: Surface

Security impact: Low — Some processes might contain sensitive data in their names.

System status — sar (`linux_system_status_sar`)

System status from the `sar` command.

Report output:

- File `/linux/sar.data`: Actual `sar` info
- File `/linux/sar-yesterday.data`: `sar` info from yesterday

Depth: Surface

Security impact: Low — No known security impact.

System status — top (`linux_system_status_top`)

System status from the `top` command.

Report output:

- File `/linux/top.data`: Process information

Depth: Surface

Security impact: Low — Some processes might contain sensitive data in their names.

System status — vmstat (`linux_system_status_vmstat`)

System status from the `vmstat` command.

Report output:

- File `/linux/vmstat.data`: Info on processes, memory, paging, block IO, traps, disks, and CPU activity

Depth: Surface

Security impact: Low — No known security impact.

systemctl units (`linux_systemctl_units`)

Systemctl list-units on a `systemd` server.

Report output:

- File `/linux/systemd/list-units.data` : Output of `systemctl list-units`

Depth: Surface

Security impact: Low — No known security impact.

tuned (`linux_tuned`)

Tuned status and profiles.

Report output:

- Directory `/linux/tuned/tune-profiles` : Files from `/etc/tune-profiles`
- Directory `/linux/tuned/tuned` : Files from `/etc/tuned`
- File `/linux/tuned/tuned.conf` : File `/etc/tuned.conf`
- File `/linux/tuned/tuned-list.data` : Output from `tuned_adm list`
- File `/linux/tuned/tuned-active.data` : Output from `tuned_adm active`

Depth: Surface

Security impact: Low — No known security impact.

PEM configuration (`pem_configuration`)

PEM configuration files from PEM agent, PEM server, and PEM web server.

Report output:

- File `/tools/pem/config/edb-ssl-pem.conf` : PEM web server SSL configuration file
- File `/tools/pem/config/edb-pem.conf` : PEM web server configuration file
- File `/tools/pem/config/install-config` : PEM server configuration file (installation config file)
- File `/tools/pem/config/config_setup.py` : PEM server setup configuration file
- File `/tools/pem/config/pem.wsgi` : PEM server WSGI definition file
- File `/tools/pem/config/agent.cfg` : PEM agent configuration file

Depth: Surface

Security impact: Low — No known security impact.

PEM systemctl (`pem_systemctl`)

When PEM is detected, collects PEM agent and PEM web server status and content.

Report output:

- File `/tools/pem/systemd/service_name_cat.data` : Output of `systemctl cat service_name`
- File `/tools/pem/systemd/service_name_status.data` : Output of `systemctl status service_name`

Depth: Surface

Security impact: Low — No known security impact.

PgBouncer configuration (`pgbouncer_configuration`)

PgBouncer configuration files.

Report output:

- File `/tools/pgbouncer/num/config/basename` : PgBouncer configuration file from instance num

Depth: Surface

Security impact: Low — No known security impact.

PgBouncer systemctl (`pgbouncer_systemctl`)

When PgBouncer services are detected, collects status and cat of the corresponding services. Checks for any service that contains any of the PgBouncer configuration files.

Report output:

- File `/tools/pgbouncer/num/systemd/service_name_cat.data` : Output of `systemctl cat service_name` from instance num
- File `/tools/pgbouncer/num/systemd/service_name_status.data` : Output of `systemctl status service_name` from instance num

Depth: Surface

Security impact: Low — No known security impact.

PGD Proxy configuration (`pgd_proxy_configuration`)

Collects PGD Proxy configuration file that's found in the server.

Report output:

- File `/tools/pgd-proxy/config/basename` : PGD Proxy configuration file

Depth: Surface

Security impact: Low — No known security impact.

PGD Proxy systemctl (`pgd_proxy_systemctl`)

When PGD Proxy services are detected, collects status and cat of the corresponding services. Checks for any service whose name starts with `pgd-proxy` .

Report output:

- File `/tools/pgd-proxy/systemd/service_name_cat.data` : Output of `systemctl cat service_name`
- File `/tools/pgd-proxy/systemd/service_name_status.data` : Output of `systemctl status service_name`

Depth: Surface

Security impact: Low — No known security impact.

postgres/enterprisedb crontab (`postgres_enterprisedb_crontab`)

Output from `crontab -l` , if running as postgres or enterprisedb.

Report output:

- File `/linux/enterprisedb_crontab.data` : Output from `enterprisedb crontab -l` , if enterprisedb user
- File `/linux/postgres_crontab.data` : Output from `postgres crontab -l` , if postgres user

Depth: Surface

Security impact: Low — Might have entries in crontab/cron with sensitive data.

PostgreSQL systemctl (`postgresql_systemctl`)

Collects PostgreSQL service status and content.

Report output:

- File `/linux/systemd/service_name_cat.data` : Output of `systemctl cat service_name`
- File `/linux/systemd/service_name_status.data` : Output of `systemctl status service_name`

Depth: Surface

Security impact: Low — No known security impact.

repmgr CLI (`repmgr_cli`)

Collects output of `repmgr cluster crosscheck` and `repmgr daemon status` using the `repmgr.conf` file, which is found in the server.

Report output:

- File `/tools/repmgr/cli/daemon_status.out` : Output of `repmgr daemon status -f conf_file_path` command

- File `/tools/repmgr/cli/cluster_crosscheck.out` : Output of `repmgr cluster crosscheck -f conf_file_path` command

Depth: Surface

Security impact: Low — No known security impact.

repmgr configuration (`repmgr_configuration`)

Collects repmgr configuration file that's found in the server.

Report output:

- File `/tools/repmgr/config/repmgr.conf` : repmgr configuration file

Depth: Surface

Security impact: Low — No known security impact.

repmgr systemctl (`repmgr_systemctl`)

When repmgr services are detected, collects status and cat of the corresponding services. Checks for any service whose name starts with `repmgr` .

Report output:

- File `/tools/repmgr/systemd/service_name_cat.data` : Output of `systemctl cat service_name`
- File `/tools/repmgr/systemd/service_name_status.data` : Output of `systemctl status service_name`

Depth: Surface

Security impact: Low — No known security impact.

xDB CLI (`xdb_cli`)

xDB output from several CLI commands, from the xDB publication and subscription server that are running.

Report output:

- Directory `/tools/xdb/cli` : xDB CLI print commands

Depth: Surface

Security impact: Low — No known security impact.

xDB configuration (`xdb_configuration`)

xDB configuration files.

Report output:

- File `/tools/xdb/config/xdbReplicationServer.config` : xDB startup configuration
- File `/tools/xdb/config/edb-repl.conf` : xDB replication configuration
- File `/tools/xdb/config/xdb_subserver.conf` : xDB subscription server configuration
- File `/tools/xdb/config/xdb_pubserver.conf` : xDB publication server configuration

Depth: Surface

Security impact: Low — No known security impact.

xDB systemctl (`xdb_systemctl`)

When xDB services are detected, collects status and cat of `edb- xdbpubserver` and `edb-xdbsubserver` .

Report output:

- File `/tools/xdb/systemd/service_name_cat.data` : Output of `systemctl cat service_name`
- File `/tools/xdb/systemd/service_name_status.data` : Output of `systemctl status service_name`

Depth: Surface

Security impact: Low — No known security impact.

Microsoft Windows operating system

PEM configuration — Windows (`pem_configuration_windows`)

PEM configuration files from PEM agent, PEM server, and PEM web server in a Windows environment.

Report output:

- File `/tools/pem/config/edb-ssl-pem.conf` : PEM web server SSL configuration file
- File `/tools/pem/config/edb-pem.conf` : PEM web server configuration file
- File `/tools/pem/config/pem.wsgi` : PEM server WSGI definition file
- File `/tools/pem/config/agent.cfg` : PEM agent configuration file

Depth: Surface

Security impact: Low — No known security impact.

PEM sc (`pem_sc`)

When PEM is detected, collects PEM agent and PEM web server status and content

Report output:

- File `/tools/pem/sc/service_name_query.data` : Output of `sc query service_name`

Depth: Surface

Security impact: Low — No known security impact.

Disk information (`win_disk_information`)

Disk and controller information from the system registry.

Report output:

- File `/windows/enum_ide.reg` : Local machine ide device settings
- File `/windows/enum_scsi.reg` : Local machine scsi device settings

Depth: Surface

Security impact: Low — No known security impact.

Hosts file (`win_hosts`)

Host files and network-related information.

Report output:

- File `/windows/services.data` : Windows `services` file
- File `/windows/protocol.data` : Windows `protocol` file
- File `/windows/networks.data` : Windows `networks` file
- File `/windows/hosts.sam` : Windows `hosts.sam` file
- File `/windows/hosts.data` : Windows `hosts` file

Depth: Surface

Security impact: Low — No known security impact.

MsInfo (`win_msinfo`)

`MsInfo32` report in `NFO` and `TXT` format.

Report output:

- File `/windows/msinfo_report.txt` : Information from the `MsInfo32` in textual format
- File `/windows/msinfo_report.nfo` : Information from the `MsInfo32` in `NFO`

Depth: Surface

Security impact: Low — No known security impact.

ODBC/64 (`win_odbc32_info`)

ODBC configuration from the 64-bit registry section.

Report output:

- File `/windows/user_odbc_wow64.reg` : User DSN list
- File `/windows/localmachine_odbcinst_wow64.reg` : List of installed ODBC drivers
- File `/windows/localmachine_odbc_wow64.reg` : System DSN list

Depth: Surface

Security impact: Medium — ODBC connection information could expose the presence of other databases or connection information to PostgreSQL that can be used to attack the system.

ODBC/32 (`win_odbc64_info`)

ODBC configuration from the 32-bit registry section.

Report output:

- File `/windows/user_odbc.reg` : User DSN list
- File `/windows/localmachine_odbcinst.reg` : list of installed ODBC drivers
- File `/windows/localmachine_odbc.reg` : System DSN list

Depth: Surface

Security impact: Medium — ODBC connection information could expose the presence of other databases or connection information to PostgreSQL that can be used to attack the system.

systeminfo (`win_systeminfo`)

Output of the `systeminfo` command.

Report output:

- File `/windows/systeminfo_report.txt` : Information from the `systeminfo` command

Depth: Surface

Security impact: Low — No known security impact.

Disk volumes (`win_volumes`)

Volume list from `WMI`.

Report output:

- File `/windows/association_structure` : Association between drive letters and physical drives
- File `/windows/volume_disk` : Volume list from the WMI subsystem
- File `/windows/logical_disk_list` : Logical disk list from the WMI subsystem
- File `/windows/disk_partition_list` : Disk partition list from the WMI subsystem
- File `/windows/disk_drive_list` : Disk list from the WMI subsystem

Depth: Surface

Security impact: Low — No known security impact.

xDB CLI — Windows (`xdb_cli_windows`)

xDB output from several CLI commands, from the running xDB publication and subscription servers.

Report output:

- Directory `/tools/xdb/cli` : xDB CLI print commands

Depth: Surface

Security impact: Low — No known security impact.

xDB configuration — Windows (`xdb_configuration_windows`)

xDB configuration files.

Report output:

- File `/tools/xdb/config/xdbReplicationServer.config` : xDB startup configuration`
- File `/tools/xdb/config/edb-repl.conf` : xDB replication configuration`
- File `/tools/xdb/config/xdb_subserver.conf` : xDB subscription server configuration
- File `/tools/xdb/config/xdb_pubserver.conf` : xDB publication server configuration

Depth: Surface

Security impact: Low — No known security impact.

xDB sc (`xdb_sc`)

When xDB is detected, collects xDB publication and subscription server status.

Report output:

- File `/tools/xdb/sc/service_name_query.data` : Output of `sc query service_name`

Depth: Surface

Security impact: Low — No known security impact.

PostgreSQL/BDR3 instance

Current archiver stats (`postgresql_archiver`)

Statistics about the archiver process activity (from `pg_stat_archiver`).

Report output:

- File `postgresql/archiver.out`

Depth: Surface

Security impact: Low — No known security impact.

Available extensions (`postgresql_available_extensions`)

List of extensions available on the server.

Report output:

- File `postgresql/available_extensions.out`

Depth: Surface

Security impact: Low — No known security impact.

Current bg_writer stats (`postgresql_bgwriter`)

Statistics about the background writer process activity (from `pg_stat_bgwriter`).

Report output:

- File `postgresql/bgwriter.out`

Depth: Surface

Security impact: Low — No known security impact.

Directory with binaries (`postgresql_bin_dir`)

PostgreSQL binary directory.

Report output:

- File `/postgresql/postgresql_bin_path.data` : Path to the PostgreSQL bin directory

Depth: Surface

Security impact: Low — No known security impact.

Current configuration (`postgresql_configuration`)

PostgreSQL current configuration.

Report output:

- File `postgresql/configuration.out`

Depth: Surface

Security impact: Medium — `postgresql.conf` might contain bad security policies

Configuration files (`postgresql_configuration_files`)

PostgreSQL configuration files and the data directory path. Passwords contained in well-known connection strings are redacted for information-security reasons.

Report output:

- File `/postgresql/pg_ident.conf` : PostgreSQL ident configuration file
- File `/postgresql/pg_hba.conf` : PostgreSQL host-based authentication file
- File `/postgresql/postgresql.auto.conf` : PostgreSQL auto configuration file
- File `/postgresql/recovery.done` : PostgreSQL `recovery.done` file
- File `/postgresql/recovery.conf` : PostgreSQL `recovery.conf` file
- File `/postgresql/postgresql.conf` : PostgreSQL configuration file

Depth: Surface

Security impact: Medium — `pg_hba.conf` and `pg_ident.conf` might expose potential security holes, such as trusted connections.

Current rate of new connections established to the DB (`postgresql_conns_per_second`)

Current rate of new connections established during 3s observation period.

Report output:

- File `postgresql/conns_per_second.out`

Depth: Surface

Security impact: Low — No known security impact.

Databases (`postgresql_databases`)

List of databases in the PostgreSQL node.

Report output:

- File `postgresql/databases.out`

Depth: Surface

Security impact: Low — No known security impact.

postgresql_db_bdr_tables_and_views (`postgresql_db_bdr_tables_and_views`)

Collect all the tables and views of the BDR extension, except for:

- `bdr.apply_log`
- `bdr.conflict_history`
- `bdr.consensus_kv_data`
- `bdr.internal_node_pre_commit`
- `bdr.replication_status`
- `bdr.state_journal`
- `bdr.stat_activity`

Report output:

- File `/postgresql/dbs/dbname/bdr/*` : Content of all tables under the BDR schema

Depth: Shallow

Security impact: Low — No known security impact.

postgresql_db_pglogical_tables_and_views (`postgresql_db_pglogical_tables_and_views`)

Collect all the tables and views of the pglogical extension.

Report output:

- File `/postgresql/dbs/dbname/pglogical/*` : Content of all tables under the pglogical schema

Depth: Shallow

Security impact: Low — No known security impact.

Database/role setting (`postgresql_db_role_setting`)

List of database/role settings in the PostgreSQL node.

Report output:

- File `postgresql/db_role_setting.out`

Depth: Shallow

Security impact: Low — No known security impact.

Node and snapshot data (`postgresql_node`)

Information about the running PostgreSQL node.

Report output:

- File `postgresql/node.out`

Depth: Surface

Security impact: Low — No known security impact.

pg_config (`postgresql_pg_config`)

PostgreSQL `pg_config` command output.

Report output:

- File `/postgresql/pg_config.data` : `pg_config` command output

Depth: Surface

Security impact: Low — No known security impact.

pg_controldata (`postgresql_pg_controldata`)

PostgreSQL `pg_controldata` information.

Report output:

- File `/postgresql/pg_controldata.data` : `pg_controldata` command output

Depth: Surface

Security impact: Low — No known security impact.

Version (`postgresql_pg_version`)

PostgreSQL client and server version.

Report output:

- File `/postgresql/postgresql_server_version.data` : PostgreSQL server version
- File `/postgresql/postgresql_client_version.data` : PostgreSQL client version

Depth: Surface

Security impact: Low — No known security impact.

Current pg_prepared_xacts contents (`postgresql_prepared_xacts`)

Status of prepared xacts (from `pg_prepared_xacts`)

Report output:

- File `postgresql/prepared_xacts.out`

Depth: Surface

Security impact: Low — No known security impact.

Current pg_replication_origin_status contents (`postgresql_replication_origin`)

Status of replication origins (from `pg_replication_origin_status`)

Report output:

- File `postgresql/replication_origins.out`

Depth: Surface

Security impact: Low — No known security impact.

Current pg_replication_slots contents (`postgresql_replication_slots`)

Replication slots (from `pg_replication_slots`).

Report output:

- File `postgresql/replication_slots.out`

Depth: Surface

Security impact: Low — No known security impact.

Roles (`postgresql_roles`)

Database roles from `pg_roles` .

Report output:

- File `postgresql/roles.out`

Depth: Shallow

Security impact: Medium — `pg_roles` might contain bad security policies.

Current activity stats (`postgresql_running_activity`)

Information related to the current activity on running processes (from `pg_stat_activity`).

Report output:

- File `postgresql/running_activity.out`

Depth: Shallow

Security impact: Low — Queries in `pg_stat_activity` could contain user names and application names.

Age of current oldest running backend/transaction/query in the cluster (`postgresql_running_activity_oldestage`)

Age of current oldest running backend/transaction/query in the cluster.

Report output:

- File `postgresql/running_activity_maxage.out`

Depth: Surface

Security impact: Low — No known security impact.

Active locks (`postgresql_running_locks`)

List of active locks.

Report output:

- File `postgresql/running_locks.out`

Depth: Surface

Security impact: Low — No known security impact.

pg_server_limits (`postgresql_server_limits`)

Real effective kernel OS limits for the postmaster PID.

Report output:

- File `/postgresql/pg_server_limits_PORT.data` : `prlimit` for postmaster PID

Depth: Surface

Security impact: Low — No known security impact.

Current pg_shmem_allocations contents (`postgresql_shmem_allocations`)

Status of shared memory allocations (from `pg_shmem_allocations`).

Report output:

- File `postgresql/shmem_allocations.out`

Depth: Surface

Security impact: Low — No known security impact.

Current pg_stat_progress_analyze contents (`postgresql_stat_progress_analyze`)

`ANALYZE` progress.

Report output:

- File `postgresql/pg_stat_progress_analyze.out`

Depth: Surface

Security impact: Low — No known security impact.

Current pg_stat_progress_basebackup contents (`postgresql_stat_progress_basebackup`)

`BASEBACKUP` progress.

Report output:

- File `postgresql/pg_stat_progress_basebackup.out`

Depth: Surface

Security impact: Low — No known security impact.

Current pg_stat_progress_copy contents (`postgresql_stat_progress_copy`)

`COPY` progress.

Report output:

- File `postgresql/pg_stat_progress_copy.out`

Depth: Surface

Security impact: Low — No known security impact.

Current pg_stat_progress_vacuum contents (`postgresql_stat_progress_vacuum`)

`VACUUM` progress.

Report output:

- File `postgresql/pg_stat_progress_vacuum.out`

Depth: Surface

Security impact: Low — No known security impact.

Current pg_stat_replication contents (`postgresql_stat_replication`)

Replication connections (from `pg_stat_replication`).

Report output:

- File `postgresql/replication.out`

Depth: Surface

Security impact: Low — No known security impact.

Server subscription statistics (`postgresql_subscription_statistics`)

Statistics of subscriptions.

Report output:

- File `postgresql/subscription_statistics.out`

Depth: Shallow

Security impact: Low — No known security impact.

Server subscriptions (`postgresql_subscriptions`)

List of subscriptions.

Report output:

- File `postgresql/subscriptions.out`

Depth: Shallow

Security impact: Low — No known security impact.

Tablespaces (`postgresql_tablespaces`)

Tablespaces information and location.

Report output:

- File `postgresql/tablespaces.out`

Depth: Surface

Security impact: Low — No known security impact.

Workload characteristics using waits (`postgresql_waits_stats`)

PostgreSQL workload characterization using built-in wait events.

Report output:

- File `postgresql/running_waits_sample.out` : Workload characterization using built-in wait events

Depth: Surface

Security impact: Low — No known security impact.

Details for every PostgreSQL/BDR3 database

BDR1 replication slots (`postgresql_db_bdr1_replication_slots`)

List of replication slots with 9.6 format for BDR1.

Report output:

- File `bdr1_replication_slots.out`

Depth: Surface

Security impact: Low — No known security impact.

BDR2 replication slots (`postgresql_db_bdr2_replication_slots`)

List of replication slots with 9.6 format for BDR2.

Report output:

- File `bdr2_replication_slots.out`

Depth: Surface

Security impact: Low — No known security impact.

BDR conflict_history_summary aggregation (`postgresql_db_bdr3_conflict_history_summary_agg`)

Collect aggregate count for all types of conflicts.

Report output:

- File `bdr_conflict_history_summary_agg.out`

Depth: Surface

Security impact: Low — No known security impact.

BDR current activity stats (`postgresql_db_bdr3_stat_activity`)

Information related to the current activity on running processes (from `bdr.stat_activity`).

Report output:

- File `bdr_stat_activity.out`

Depth: Shallow

Security impact: Low — Queries in `bdr.stat_activity` could contain user names and application names.

BDR sequences (`postgresql_db_bdr_sequences`)

List of the BDR sequences.

Report output:

- File `bdr_sequences.out`

Depth: Surface

Security impact: Low — No known security impact.

BDR version (`postgresql_db_bdr_version`)

Currently used version of BDR.

Report output:

- File `bdr_version.out`

Depth: Surface

Security impact: Low — No known security impact.

edb_wait_states database settings (`postgresql_db_edb_wait_states_database_settings`)

Shows information on database settings.

Report output:

- File `edb_wait_states/database_settings.out`

Depth: Shallow

Security impact: Low — No known security impact.

edb_wait_states load profile (`postgresql_db_edb_wait_states_load_profile`)

Average load profile of transactions.

* *Report output:*

- File `edb_wait_states/load_profile.out`

Depth: Shallow

Security impact: Low — No known security impact.

edb_wait_states operating system information (`postgresql_db_edb_wait_states_operating_system_information`)

Information about the operating system.

Report output:

- File `edb_wait_states/operating_system_information.out`

Depth: Shallow

Security impact: Low — No known security impact.

edb_wait_states server information — part 1 (`postgresql_db_edb_wait_states_server_information_1`)

Information about the Postgres server — part 1.

Report output:

- File `edb_wait_states/server_information_1.out`

Depth: Shallow

Security impact: Low — No known security impact.

edb_wait_states server information — part 2 (`postgresql_db_edb_wait_states_server_information_2`)

Information about the Postgres server — part 2.

Report output:

- File `edb_wait_states/server_information_2.out`

Depth: Shallow

Security impact: Low — No known security impact.

edb_wait_states shared buffers statistics (`postgresql_db_edb_wait_states_shared_buffers_stats`)

Shows statistics in terms of buffer hits and misses.

Report output:

- File `edb_wait_states/shared_buffers_stats.out`

Depth: Shallow

Security impact: Low — No known security impact.

edb_wait_states temp file statistics (`postgresql_db_edb_wait_states_temp_file_stats`)

Shows statistics about temp files usage.

Report output:

- File `edb_wait_states/temp_file_stats.out`

Depth: Shallow

Security impact: Low — No known security impact.

edb_wait_states top 10 SQL statements by cputime (`postgresql_db_edb_wait_states_top_sql_statements_cputime` **)**

Shows which SQL statements are taking more CPU time.

Report output:

- File `edb_wait_states/top_sql_statements_cputime.out`

Depth: Shallow

Security impact: Low — No known security impact.

edb_wait_states top 10 SQL statements by dbtime (`postgresql_db_edb_wait_states_top_sql_statements_dbtime` **)**

Shows which SQL statements are taking more database time.

Report output:

- File `edb_wait_states/top_sql_statements_dbtime.out`

Depth: Shallow

Security impact: Low — No known security impact.

edb_wait_states top 10 SQL statements by waittime (`postgresql_db_edb_wait_states_top_sql_statements_waittime` **)**

Shows which SQL statements are waiting for more time.

Report output:

- File `edb_wait_states/top_sql_statements_waittime.out`

Depth: Shallow

Security impact: Low — No known security impact.

edb_wait_states top 10 wait events (`postgresql_db_edb_wait_states_top_wait_events` **)**

Shows which events are taking more time on the cluster.

Report output:

- File `edb_wait_states/top_wait_events.out`

Depth: Shallow

Security impact: Low — No known security impact.

edb_wait_states transactions statistics (`postgresql_db_edb_wait_states_transaction_stats`)

Shows statistics in terms of commits and rollbacks.

Report output:

- File `edb_wait_states/transaction_stats.out`

Depth: Shallow

Security impact: Low — No known security impact.

edb_wait_states tuple statistics (`postgresql_db_edb_wait_states_tuple_stats`)

Shows statistics in terms of tuple reads and writes.

Report output:

- File `edb_wait_states/tuple_stats.out`

Depth: Shallow

Security impact: Low — No known security impact.

edb_wait_states user sessions (`postgresql_db_edb_wait_states_user_sessions`)

Shows information about user sessions.

Report output:

- File `edb_wait_states/user_sessions.out`

Depth: Shallow

Security impact: Low — No known security impact.

edb_wait_states WAL statistics (`postgresql_db_edb_wait_states_wal_stats`)

Shows statistics in terms of WAL writes.

Report output:

- File `edb_wait_states/wal_stats.out`

Depth: Shallow

Database extensions (`postgresql_db_extensions`)

List of extensions in the database.

Report output:

- File `extensions.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database indexes (`postgresql_db_indexes`)

List of indexes in the database.

Report output:

- File `indexes.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database procedural languages (`postgresql_db_languages`)

Procedural languages in the database.

Report output:

- File `language.out`

Depth: Shallow

Security impact: Low — No known security impact.

BDR monitor_group_raft (`postgresql_db_monitor_group_raft`)

Check the raft status in the BDR cluster.

Report output:

- File `bdr_monitor_group_raft.out`

Depth: Surface

Security impact: Low — No known security impact.

BDR monitor_group_versions (`postgresql_db_monitor_group_versions`)

Check the version of all BDR nodes.

Report output:

- File `bdr_monitor_group_versions.out`

Depth: Surface

Security impact: Low — No known security impact.

BDR monitor_local_replslots (`postgresql_db_monitor_local_replslots`)

Check all the replication slot status.

Report output:

- File `bdr_monitor_local_replslots.out`

Depth: Surface

Security impact: Low — No known security impact.

Database operators (`postgresql_db_operators`)

Operators in the database.

Report output:

- File `operator.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database partitioned tables (`postgresql_db_partitioned_tables`)

Information about partitioned tables (using declarative partitioning).

Report output:

- File `partitioned_table.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database PEM pem.agent table tuples (`postgresql_db_pem_agent`)

Data from metatable `pem.agent`.

Report output:

- File `pem_agent.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database PEM pem.agent_config table tuples (`postgresql_db_pem_agent_config`)

Data from metatable `pem.agent_config`.

Report output:

- File `pem_agent_config.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database PEM pem.agent_heartbeat table tuples (`postgresql_db_pem_agent_heartbeat`)

Data from metatable `pem.agent_heartbeat`.

Report output:

- File `pem_agent_heartbeat.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database PEM pem.agent_server_binding table tuples (`postgresql_db_pem_agent_server_binding`)

Data from metatable `pem.agent_server_binding`.

Report output:

- File `pem_agent_server_binding.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database PEM pem.config table tuples (`postgresql_db_pem_config`)

Data from metatable `pem.config` .

Report output:

- File `pem_config.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database PEM pem.email_group table tuples (`postgresql_db_pem_email_group`)

Data from metatable `pem.email_group` .

Report output:

- File `pem_email_group.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database PEM pem.email_group_option table tuples (`postgresql_db_pem_email_group_option`)

Data from metatable `pem.email_group_option` .

Report output:

- File `pem_email_group_option.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database PEM pem.probe table tuples (`postgresql_db_pem_probe`)

Data from metatable `pem.probe` .

Report output:

- File `pem_probe.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database PEM pem.probe_schedule table tuples (`postgresql_db_pem_probe_schedule`)

Data from metatable `pem.probe_schedule` .

Report output:

- File `pem_probe_schedule.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database PEM pem.schema_version() output (`postgresql_db_pem_schema_version`)

Output from function `pem.schema_version()` .

Report output:

- File `pem_schema_version.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database PEM pem.server table tuples (`postgresql_db_pem_server`)

Data from metatable `pem.server` .

Report output:

- File `pem_server.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database PEM pem.server_heartbeat table tuples (`postgresql_db_pem_server_heartbeat`)

Data from metatable `pem.server_heartbeat` .

Report output:

- File `pem_server_heartbeat.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database PEM pem.smtp_spool table tuples (`postgresql_db_pem_smtp_spool`)

Data from metatable `pem.smtp_spool` .

Report output:

- File `pem_smtp_spool.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database PEM pem.snmp_spool table tuples (`postgresql_db_pem_snmp_spool`)

Data from metatable `pem.snmp_spool` .

Report output:

- File `pem_snmp_spool.out`

Depth: Shallow

Security impact: Low — No known security impact.

Pglogical subscription status (`postgresql_db_pglogical_subscription_status`)

List of tables replicated by pglogical.

Report output:

- File `pglogical_subscription_status.out`

Depth: Surface

Security impact: Low — No known security impact.

Database functions (`postgresql_db_pkgs`)

Database packages/functions/procedures with arguments.

Report output:

- File `pkgs.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database functions (`postgresql_db_procs` **)**

Functions in the database.

Report output:

- File `proc.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database publication tables (`postgresql_db_publication_tables` **)**

List of tables of publications of the database.

Report output:

- File `publication_tables.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database publications (`postgresql_db_publications` **)**

List of publications of the database.

Report output:

- File `publications.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database repmgr repmgr.events table tuples (`postgresql_db_repmgr_events` **)**

Data from metatable `repmgr.events`.

Report output:

- File `repmgr/events.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database repmgr repmgr.monitoring_history table tuples (`postgresql_db_repmgr_monitoring_history`)

Data from metatable `repmgr.monitoring_history` .

Report output:

- File `repmgr/monitoring_history.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database repmgr repmgr.nodes table tuples (`postgresql_db_repmgr_nodes`)

Data from metatable `repmgr.nodes` .

Report output:

- File `repmgr/nodes.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database repmgr repmgr.replication_status table tuples (`postgresql_db_repmgr_replication_status`)

Data from metatable `repmgr.replication_status` .

Report output:

- File `repmgr/replication_status.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database repmgr repmgr.show_nodes table tuples (`postgresql_db_repmgr_show_nodes`)

Data from metatable `repmgr.show_nodes` .

Report output:

- File `repmgr/show_nodes.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database schema (`postgresql_db_schemas` **)**

List of schemas in the database.

Report output:

- File `schemas.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database statistics (`postgresql_db_statistics` **)**

Statistics of the database.

Report output:

- File `statistics.out`

Depth: Deep

Security impact: Low — No known security impact.

Database subscription tables (`postgresql_db_subscription_tables` **)**

List of tables of subscriptions of the database.

Report output:

- File `subscription_tables.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database tables (`postgresql_db_tables` **)**

List of tables in the database.

Report output:

- File `tables.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database types (`postgresql_db_types` **)**

Types in the database.

Report output:

- File `type.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database xDB _edb_replicator_pub.xdb_mmr_pub_group table tuples (`postgresql_db_xdb_mmr_pub_group` **)**

Data from metatable `_edb_replicator_pub.xdb_mmr_pub_group`.

Report output:

- File `xdb_mmr_pub_group.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database xDB _edb_replicator_pub.xdb_pub_database table tuples (`postgresql_db_xdb_pub_database` **)**

Data from metatable `_edb_replicator_pub.xdb_pub_database`.

Report output:

- File `xdb_pub_database.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database xDB _edb_replicator_pub.xdb_pub_replog table tuples (`postgresql_db_xdb_pub_replog` **)**

Last 50 rows from metatable `_edb_replicator_pub.xdb_pub_replog`.

Report output:

- File `xdb_pub_replog.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database xDB _edb_replicator_pub.xdb_publication_subscriptionstable tuples (`postgresql_db_xdb_publication_subscriptions`)

Data from metatable `_edb_replicator_pub.xdb_publication_subscriptions`.

Report output:

- File `xdb_publication_subscriptions.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database xDB _edb_replicator_pub.xdb_publications table tuples (`postgresql_db_xdb_publications`)

Data from metatable `_edb_replicator_pub.xdb_publications`.

Report output:

- File `xdb_publications.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database xDB _edb_replicator_pub.rrep_mmr_pub_group table tuples (`postgresql_db_xdb_rrep_mmr_pub_group`)

Data from metatable `_edb_replicator_pub.rrep_mmr_pub_group`.

Report output:

- File `xdb_rrep_mmr_pub_group.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database xDB _edb_replicator_pub.rrep_mmr_txset table tuples (`postgresql_db_xdb_rrep_mmr_txset`)

Last 10 rows from metatable `_edb_replicator_pub.rrep_mmr_txset`.

Report output:

- File `xdb_rrep_mmr_txset.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database xDB _edb_replicator_pub.rrep_properties table tuples (`postgresql_db_xdb_rrep_properties`)

Data from from metatable `_edb_replicator_pub.rrep_properties` .

Report output:

- File `xdb_rrep_properties.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database xDB _edb_replicator_pub.rrep_publication_tablestable tuples (`postgresql_db_xdb_rrep_publication_tables`)

Data from from metatable `_edb_replicator_pub.rrep_publication_tables` .

Report output:

- File `xdb_rrep_publication_tables.out`

Depth: Shallow

Security impact: Low — No known security impact.

Database xDB _edb_replicator_pub.rrep_txset table tuples (`postgresql_db_xdb_rrep_txset`)

Data from metatable `_edb_replicator_pub.rrep_txset` .

Report output:

- File `xdb_pub_rrep_txset.out`

Depth: Shallow

Security impact: Low — No known security impact.

Oracle-compatibile partitioning key view (`postgresql_epas_all_part_key_columns`)

Provides partitioning key details (`all_part_key_columns` , EDB Postgres Advanced Server specific).

Report output:

- File `epas_all_part_key_columns.out`

Depth: Surface

Security impact: Low — No known security impact.

Oracle-compatibile all partitioned table view (`postgresql_epas_all_part_tables`)

All partitioned tables view (from `all_part_tables` , EDB Postgres Advanced Server specific).

Report output:

- File `epas_all_part_tables.out`

Depth: Surface

Security impact: Low — No known security impact.

Oracle-compatibile subpartitioning key view (`postgresql_epas_all_subpart_key_columns`)

Provides subpartitioning key details (`all_subpart_key_columns` , EDB Postgres Advanced Server specific).

Report output:

- File `epas_all_subpart_key_columns.out`

Depth: Surface

Security impact: Low — No known security impact.

Oracle-compatibile all table partitions view (`postgresql_epas_all_tab_partitions`)

All partitions of all partitioned tables view (`all_tab_partitions` , EDB Postgres Advanced Server specific).

Report output:

- File `epas_all_tab_partitions.out`

Depth: Surface

Security impact: Low — No known security impact.

Oracle-compatibile all table subpartitioning view (`postgresql_epas_all_tab_subpartitions`)

All subpartitions of all partitioned tables view (`all_tab_subpartitions` , EDB Postgres Advanced Server specific).

Report output:

- File `epas_all_tab_subpartitions.out`

Depth: Surface

Security impact: Low — No known security impact.

EDB Postgres Advanced Server-specific dblink information (`postgresql_epas_dblink`)

Current EDB Postgres Advanced Server dblink information from `edb_dblink` .

Report output:

- File `epas_dblink.out`

Depth: Surface

Security impact: Low — No known security impact.

EDB Postgres Advanced Server-specific queue information (`postgresql_epas_queue`)

Current EDB Postgres Advanced Server queue information from `edb_queue` .

Report output:

- File `epas_edb_queue.out`

Depth: Surface

Security impact: Low — No known security impact.

EDB Postgres Advanced Server-specific queue callback information (`postgresql_epas_queue_callback`)

Current EDB Postgres Advanced Server queue callback information from `edb_queue_callback` .

Report output:

- File `epas_edb_queue_callback.out`

Depth: Surface

Security impact: Low — No known security impact.

EDB Postgres Advanced Server-specific queue table information (`postgresql_epas_queue_table`)

Current EDB Postgres Advanced Server queue table information from `edb_queue_table` .

Report output:

- File `epas_edb_queue_table.out`

Depth: Surface

Security impact: Low — No known security impact.

Barman (Backup and Recovery Manager)

Barman check (`barman_check`)

Collect the status of the Barman `check` framework for all configured servers.

Report output:

- File `/barman/barman_check.data` : Output of `barman check`

Depth: Surface

Security impact: Low — No known security impact.

Barman diagnose (`barman_diagnose`)

Collect the Barman diagnosis information.

Report output:

- File `/barman/diagnose.data` : Output from `barman diagnose`

Depth: Surface

Security impact: Low — No known security impact.

Barman executable location (`barman_executable`)

Collect the `barman` executable location.

Report output:

- File `/barman/barman_location.data` : The path of the main barman executable

Depth: Surface

Security impact: Low — No known security impact.

Barman module location (`barman_modules_path`)

Collect the location of the Barman Python modules.

Report output:

- File `/barman/barman_python_verbose.data` : The list of Barman Python modules

Depth: Surface

Security impact: Low — No known security impact.

11 Servers accepting upload of reports

> **Note**
>
> This information applies only to Lasso executables with external network access. You can verify that your Lasso can access the Internet using the `--version` option. Look for the line `External network access enabled`.

The Lasso `--upload` command sends the tarball to one of the front-line application servers that are configured in high availability and load balancing to receive Lasso reports for EDB as part of the Customer Portal.

These servers are part of an ISO 27K1 infrastructure and fully comply with the European Union's General Data Protection Regulation (GDPR).

Provide this list of servers to your infrastructure team so that they can open outbound connections to port 443 of the following IPv4 addresses:

1. 78.46.204.124
2. 88.99.39.37
3. 88.99.39.42
4. 88.99.80.86
5. 116.203.69.173
6. 116.203.69.177
7. 116.203.69.179
8. 116.203.69.185
9. 116.203.69.248
10. 116.203.69.253
11. 159.69.178.25
12. 159.69.178.96
13. 159.69.181.84
14. 159.69.181.86
15. 195.201.138.157
16. 195.201.99.62

If you also use IPv6, add this list:

1. 2a01:4f8:1c17:482e::1
2. 2a01:4f8:1c17:4f1a::1
3. 2a01:4f8:1c17:5197::1
4. 2a01:4f8:1c17:529d::1
5. 2a01:4f8:c010:1210::1
6. 2a01:4f8:c010:154c::1
7. 2a01:4f8:c010:1550::1
8. 2a01:4f8:c010:1551::1
9. 2a01:4f8:1c1c:c82e::1
10. 2a01:4f8:1c1c:c836::1
11. 2a01:4f8:1c1c:c837::1
12. 2a01:4f8:1c1c:c95f::1
13. 2a01:4f8:1c1c:c97e::1
14. 2a01:4f8:1c1c:ca43::1
15. 2a01:4f8:1c1c:ca4e::1
16. 2a01:4f8:1c1c:dab3::1