



BGuardian

Bacula Systems Documentation

Contents

1	Scope	3
2	Features	3
3	Architecture	5
4	Installation	6
5	Configuration	7
6	Operations	23
7	Best Practices	30
8	Limitations	31
9	Troubleshooting	32

Contents

The following article aims at presenting the reader with information about the **Bacula Enterprise BGuardian Plugin** (Bacula Guardian). The document briefly describes the target technologies of the plugin, defines the scope of its operations, and presents its main features.

Bacula Enterprise BGuardian Plugin is intended to become the ultimate tool designed to facilitate and automate some of the most important tasks around security analysis for a backup environment using Bacula Enterprise, providing a comprehensive overview of your system's security posture, highlighting potential issues, suspicious activities, and weak points.

With its advanced capabilities and comprehensive approach, BGuardian safeguards your environment by meticulously scrutinizing the whole Bacula Configuration, the evolution and behavior of the executed jobs, as well as the status of the different components of the target system.

Using statistical analysis and best-practices knowledge, it provides backup poisoning detection features, as well as secure configuration assessment. It does it by extracting valuable insights from the gathered information and presenting it in the form of easy to understand reports. On the other hand, it also generates persistent alerts that serve as a framework to control and act upon any found issue.

Even if BGuardian represents a very important help in terms of security for a given environment, it is crucial to remark that security must be considered as a whole in any organization. It starts on the very internal roots of any corporation when a security plan and recovery strategies are well-defined and followed. It continues with the application of best practices at all levels: Using secure communications, using less privileges principles (Zero-trust), using secure network architectures, strong passwords policies, monitoring and auditing processes, multifactor authentication, data encryption, data immutability and many other technical features. However, probably one of the most important parts, comes with the application of common sense and a good education in secure practices for all the people inside the organization, which means things like avoid phishing attacks, be careful with any untrusted or not updated software, not share private or internal information in sensible places, not reuse passwords and many more.

Together with BGuardian, **Bacula Enterprise** offers all the other needed features to make the backup environment an extremely secure place. In order to have more information about them, refer to the appropriate section associated to the different security features listed in: Security Features.

Through subchapters, more in-depth information can be found about the following topics:

1 Scope

Bacula Enterprise BGuardian Plugin currently supports any platform where Bacula Director can be deployed.

This plugin is available since **Bacula Enterprise 16.0.12**.

See also:

- *BGuardian Features*
- *BGuardian Installation*
- *BGuardian Configuration*
- *BGuardian Operations*
- *BGuardian Best Practices*
- *BGuardian Limitations*
- *BGuardian Troubleshooting*

Go back to the *BGuardian plugin main page*.

2 Features

2.1 General Features

The main feature this plugin offers is to act as an assistant in order to help the system administrator to have a more solid and secure environment. This can be divided into the following generic features:

- Backup poisoning detection: Mark jobs with unexpected values in the amount of data processed, which could be a result of ransomware activities
- Secure configuration assessment: Make suggestions of configuration modifications to help to comply with secure recommendations and best-practices
- Failure patterns detection: Detection of potential issues related to running services
- Friendly reports generation: Detailed logging of analysis activities while running
- Persistent alerts generation: Summarized information that generates Bacula Events when each alert is created or recovered, to be updated with each analysis execution

2.2 Services

Below is the list of services/checks that the plugin provides:

- Strong password checking
- Duplicated password checking
- Strong permissions in Bacula configuration
- Correct users for running processes
- Recent successful catalog backup

- Recent successful backup of Bacula configuration
- Recent usage of Restore jobs
- Recent usage of Verify jobs
- Recent usage of Copy/Migration jobs to a different storage tier
- Usage of malware protection in jobs
- Usage of restricted consoles
- Usage of antivirus jobs for every client
- Usage of Events in Message resources
- Usage of encryption in the environment
- Detect cloud devices without encryption
- Usage of volume protection in the environment
- Director status (errors, FIPS usage, debug flags)
- Usage of DirAddress setting to limit Director service to be listening on specific interfaces
- Reachability and status of Storage Daemons (errors, FIPS usage, debug flags)
- Control of having enough free space on Storage Daemon devices
- Control of having enough free space for Deduplication in Dedup enabled Storage Daemons
- Detection of any kind of errors in Global Endpoint Deduplication engine (general errors, container errors, vacuum errors...)
- Detection of orphan, suspect or missed references in Global Endpoint Deduplication engine
- Recent execution of the Global Endpoint Deduplication Vacuum process for Dedup enabled Storage Daemons
- Reachability and status of File Daemons (errors, FIPS usage, debug flags)
- Usage of security plugin in each Client
- Check running Bacula versions among the different daemons and report any unsupported differences
- Check recent executions of PostgreSQL vacuum procedures over key tables for Bacula
- Check recent executions of PostgreSQL analyze procedures over key tables for Bacula
- Check if PostgreSQL configuration values are under or over the recommended thresholds
- Backup poisoning detection through deviation analysis
- Detection of jobs under a threshold success ratio
- Detection of jobs without a recent successful execution
- Detection of jobs failed consecutively a specified number of times
- Detection of successful Full backup jobs that did not backup any data
- Detection of jobs that were not copied to any 2-Tier storage layer
- Detection of jobs that were never verified
- Detection of jobs where a restore was never attempted
- Detection of BWeb users which do not have 2Factor authentication enabled
- Detection of Incremental or Differential jobs whose predecessor is no longer in the catalog

- Detection of jobs where “will not descend” is reported due to ‘onefs = yes’. This is a possible indication that data which might be expected to be backed up is not being backed up
- Report of jobs where viruses or malware is found
- Report of recent recorded Bacula events about security (for instance, failed bconsole connections)
- Report of jobs with Global Endpoint Deduplication enabled which show a low deduplication ratio

This list of features will be growing with future versions of this plugin.

See also:

- *BGuardian Scope*
- *BGuardian Architecture*
- *BGuardian Installation*
- *BGuardian Configuration*
- *BGuardian Operations*
- *BGuardian Best Practices*
- *BGuardian Limitations*
- *BGuardian Troubleshooting*

Go back to the *BGuardian plugin main page*.

3 Architecture

Bacula Enterprise BGuardian Plugin is a Bacula **Director plugin** which may be run manually, or automatically on a periodic schedule via a Bacula Admin Job.

This tool is packed as a configurable daemon built on top of the Java language. The daemon can automatically communicate with Bacula through the following channels:

- bconsole
- bdirjson tool
- Direct connection to the SQL catalog

Once the different services are performed, BGuardian writes user-friendly HTML reports into the local filesystem, JSON reports that can be processed by any other tool, as well as other internal files which provide a persistent alert framework with information that is kept through subsequent daemon executions and that is available also for other layers of Bacula.

Below is a simplified vision of the BGuardian architecture within a generic **Bacula Enterprise** deployment:

See also:

- *BGuardian Scope*
- *BGuardian Features*
- *BGuardian Installation*
- *BGuardian Configuration*
- *BGuardian Operations*
- *BGuardian Best Practices*

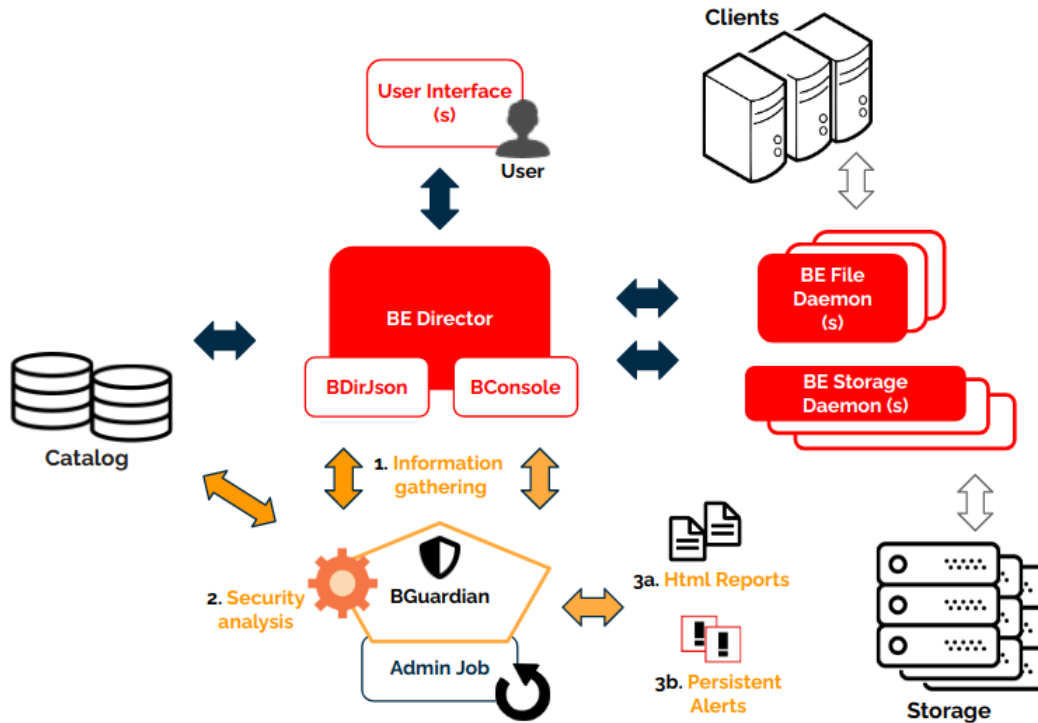


Fig. 1: BGuardian Plugin Architecture

- *BGuardian Limitations*
- *BGuardian Troubleshooting*

Go back to the *BGuardian plugin main page*.

4 Installation

This article describes how to install Bacula Enterprise BGuardian Plugin.

4.1 Prerequisites

- The BGuardian Plugin need to be installed on the host where the Bacula Director is installed.
- The plugin works through a Java daemon, therefore Java needs to be installed onto the host through a JRE or JDK package (openjdk-11-jre for example).
- The Java environment needs to be in version 11 or above and the Java binary must be available in the system PATH.

4.2 Installation Methods

- `BGuardianInstallationWithBIM` (recommended)
- `BGuardianInstallationPackageManagers`

4.3 Result

The package installs the following elements:

- Jar libraries in `/opt/bacula/lib` (such as `bacula-bguardian-dir-plugin-x.x.x.jar`). Note that the version of the jar archive is not aligned with the version of the package. However, that version can be shown in the json reports.
- The `bguardian` shell script file in `/opt/bacula/bin` which invokes the jar files. The `bguardian` script searches for the most recent `bacula-bguardian-dir-plugin-x.x.x.jar` file in order to launch it, even though usually there should only be one file.

See also:

- *[BGuardian Scope](#)*
- *[BGuardian Features](#)*
- *[BGuardian Architecture](#)*
- *[BGuardian Configuration](#)*
- *[BGuardian Operations](#)*
- *[BGuardian Best Practices](#)*
- *[BGuardian Limitations](#)*

Go back to the *[BGuardian plugin main page](#)*.

5 Configuration

The following chapter presents information on how to configure BGuardian.

5.1 Base Function

Basics

BGuardian needs to be able to connect to `bdirjson` tools, `bconsole` and the catalog.

`BConsole` and `bdirjson` commands are expected to be inside the `/opt/bacula/bin` directory. If the installation was done using custom paths, it will be needed to create symlinks to this location.

BGuardian uses the same kind of connection to the catalog that Bacula instance is using in the same host. Therefore, it will use the same credentials and database name that is configured in the 'Catalog' resource of the Director configuration.

BGuardian may be executed manually from the command line by running the script. Normally, if the connection to the database is using 'peer' mode it should be run by the bacula user:

Listing 1: Admin Job

```
sudo -u bacula /bin/bash /opt/bacula/bin/bguardian
```

Note that /bin/bash may not be required if the bacula user can find the bash binary regularly.

It is recommended to run it from a Bacula Admin Job and schedule it once a day:

Admin Job example:

Listing 2: Admin Job

```
Job {
  Name = "BGuardian"
  Type = Admin
  Schedule = Daily0100
  JobDefs = JobDefault

  Runscript {
    RunWhen = Before
    RunOnClient = no
    Command = "sudo -u bacula /bin/bash /opt/bacula/bin/bguardian"
  }
}
```

Daily Schedule example:

Listing 3: Schedule

```
Schedule {
  Name = "Daily0100"
  Run = Level=Full daily at 01:00
}
```

BGuardian will run an analysis of all the configured services and produce some reports by default, as well as the corresponding alerts, that are stored as individual files. However, it also has a mode where it is possible to interact with the existing alerts in the system in order to show them, remove them or mark them to be ignored.

In general there is no need to change any parameter of BGuardian as it will work out of the box. However, it is possible to tune it in order to adjust the tests which are run and the results based on the specific environment it is being run in.

The daemon can receive parameters in 3 different forms:

1. Daemon parameters

The format is:

```
--parameter_name parameter_value
```

Note the '-' before parameter name and the space in between to put the value.

Example:

```
sudo -u bacula /bin/bash /opt/bacula/bin/bguardian --dev_min_executions 10 --max_days_
↪copy 30
```


2. Parameters in a file

It's needed to use the special parameter:

```
--config_file "/opt/bacula/etc/bguardian.conf"
```

Then, the file may contain the parameters using:

```
par1=val1  
par2=val2  
par3=val3  
...
```

Example:

Listing 4: Events Configuration

```
reports_keep_number=10  
disable_events=true  
configuration_checks_exclude=restore,verify,copy  
services_exclude=successratio,restorefrequency  
success_factor=0.7
```

3. Special alert commands

There are some commands regarding the alerts that have some shortcuts which may be invoked directly as:

```
# List active alerts in json format  
sudo -u bacula /bin/bash /opt/bacula/bin/bguardian list  
  
# List ignore alerts in json format  
sudo -u bacula /bin/bash /opt/bacula/bin/bguardian list_ignore  
  
# List active alerts in text format  
sudo -u bacula /bin/bash /opt/bacula/bin/bguardian list_text  
  
# List ignore alerts in text format  
sudo -u bacula /bin/bash /opt/bacula/bin/bguardian list_ignore_text  
  
# Add ignore  
sudo -u bacula /bin/bash /opt/bacula/bin/bguardian ignore "code"  
  
# Remove ignore  
sudo -u bacula /bin/bash /opt/bacula/bin/bguardian remove_ignore "code"  
  
# Remove alert  
sudo -u bacula /bin/bash /opt/bacula/bin/bguardian remove_alert "code"
```

BGuardian can send events regarding the alerts that it generates, which is recommended. To benefit from this feature, it is necessary to enable events in the proper message resource, adding the special 'events' keyword to the configuration:

Events configuration example:

Listing 5: Events Configuration

```
Messages {
  Name = Standard
  mailcommand = "/tmp/regress/bin/bsmtp -h localhost -f \"\"(Bacula regression) %r\" -
↪s \"Regression: %t %e of %c %l\" %r"
  operatorcommand = "/tmp/regress/bin/bsmtp -h localhost -f \"\"(Bacula regression)
↪%r\" -s \"Regression: Intervention needed for %j\" %r"
  console = all, !skipped, !terminate, !restored, events
  append = "/tmp/regress/working/log" = all, !skipped, events
  catalog = all, !skipped, events
}
```

Go back to the [main configuration page](#).

5.2 Parameters

One of the most important reasons to modify the default parameters is to select the services to include or exclude during the execution of BGuardian. By default, all services are included.

Additionally, some services allow sub-checks and those may also be excluded. The parameters to control these two features are:

- `service`: For the main services
- `configuration_checks_exclude`: To exclude checks from the configuration security service

After selecting the services to apply, there are also parameters which can control the results of those services.

Described below are all services and all of their specific parameters, as well as what actions are recommended to take if results are detected for each of them.

Fileset Common Parameters

The following parameters are applicable to the general behavior of the plugin:

Option	Required	Default	Values	Example	Description
mode	No	check	check, alert	alert	Run normal check mode or alert mode to query current alerts or add/remove alerts or ignores
service	No	configurationsecurity, infected, securityevents, deviation, successratio, failedinarow, empty, lastgood, nocopy, noverify, lowdedup, orphanchain, restorefrequency, differentfilesystem, nototp	List (separated by ‘;’) of elements from: configurationsecurity, infected securityevents, deviation, successratio, failedinarow, empty, lastgood, nocopy, noverify, differentfilesystem, lowdedup, orphanchain, restorefrequency, differentfilesystem, nototp	configurationsecurity, nocopy	Select the services that BGuardian will execute
service_exclude	No		List (separated by ‘;’) of elements from: configurationsecurity, infected, securityevents, deviation, successratio, failedinarow, empty, lastgood, nocopy, noverify, lowdedup, orphanchain, restorefrequency, differentfilesystem, nototp	configurationsecurity, nocopy	Select the services that BGuardian will exclude. Use this variable to exclude a list or the ‘service’ one to include a list, but do not use both
config_file	No		The path pointing to a file containing any combination of plugin parameters	/opt/bacula/etc/defaults/bguardian.conf	Allows to define settings file where configure any parameter of the plugin. Therefore you don’t need to put them directly in the Plugin line of the file-set
log	No		An existing path with enough permissions for File Daemon to create a file with the provided name	/tmp/bguardian.log	Generates additional log in addition to what is shown in job log. This parameter is included in the backend file, so, in general, by default the log is going to be stored in the working directory
de-	No		0, 1, 2, 3, 4, 5,	8	Generates the

Configuration security service

This service is activated if the service parameter contains the keyword: **configurationsecurity**.

Alert code is: **GC_[SUBSERVICE]**. Subservice codes are detailed below.

The purpose of this service is to report the result of different checks regarding security, status and best practices related to how Bacula is configured and running in the environment.

Option	Re-quired	Default	Values	Example	Description
max_days_backup	No	15	Integer	5	Maximum number of days without a backup of the configuration or catalog before generating an alert
max_days_copy	No	15	Integer	3	Maximum number of days without any copy job before generating an alert
max_days_verify	No	30	Integer	30	Maximum number of days without any verify job before generating an alert
max_days_restore	No	30	Integer	60	Maximum number of days without any restore job before generating an alert
max_days_vacuum	No	7	Integer	2	Maximum number of days without running vacuum in dedup enabled storage daemons
dedup_free_space_min	No	107374182400	Integer	536870912000	Limit on bytes of free space for dedup engines before raising the alert
permissions_base_paths	No	/opt/bacula	List of paths	/opt/bacula/etc	List of paths of bacula installation files where checking permissions
config_backup_job_name	No	BaculaDirectorCon-figs	Job Name	BaculaConfig	Name of the job of the Backup of the configuration of Bacula, to analyze if it's regularly run
catalog_backup_job_name	No	(auto-detected)	Job Name	BaculaCatalog	Name of the job of the Backup of the catalog of Bacula, to analyze if it's regularly run
min_free_space_percent	No	5	Integer	5	Minimum percentage of free space for a given Storage Daemon before generating an alert
configuration_checks_exclude	No		*List of checks separated by ',' (see next point)	restore, verify, copy, malware, antivirus	List of subchecks to exclude from the execution of this Configuration security service

Configuration security service subchecks

Configuration security is a special service of BGuardian that checks many things related with the configuration of the environment.

By default, it will check everything, but it is possible to exclude some checks using the **configuration_checks_exclude** parameter and adding a list of keywords there (separated by ',').

Below we briefly describe the keyword and the function of each of the services and what is the recommended action if a related issue is reported. The code serves for the alerts functionality and to quickly identify each issue:

- **passwords**: Checks duplicated passwords and the strength of them inside Bacula Director configuration.
 - **Code**: GC_PASSWOR

- **Action:** Make your passwords unique and use a strong keyword (+8 characters, include upper and lower-case, digits and symbols)
- **catalog_backup:** Checks configuration and recent execution (max_days_bacula_backup parameter) of the backup of the catalog of Bacula (configurable by config_backup_job_name).
 - **Code:** GC__CATALOG
 - **Action:** Immediately run a backup of the catalog and make sure your schedule is frequent enough (once a week at least)
- **config_backup:** Checks configuration and recent execution (max_days_bacula_backup parameter) of the backup of the configuration of Bacula (configurable by catalog_backup_job_name).
 - **Code:** GC__CONFIG_
 - **Action:** Immediately run a backup of the catalog and make sure your schedule is frequent enough (once a week at least)
- **restore:** Checks execution of some recent restore (max_days_restore parameter). It is a best practice to run some restore of the different kinds of data from time to time.
 - **Code:** GC__RESTOR
 - **Action:** Run some restore for each kind of data and each kind of storage from time to time.
- **verify:** Checks execution of some recent verify job (max_days_verify parameter). It is a best practice to verify the data of your jobs.
 - **Code:** GC__VERIFY
 - **Action:** Configure verify jobs for your data
- **copy:** Checks execution of some recent restore (max_days_copy parameter). It is a best practice to use a multi-tier strategy with your backups.
 - **Code:** GC__COPY
 - **Action:** Define a second storage tier and configure copy jobs to send your data there.
- **malware:** Checks jobs that could activate the Malware protection function, but have not enabled it.
 - **Code:** GC__MALWAR
 - **Action:** Enable Malware detection for any reported system that could be suitable to be infected because its location, usage pattern and data kind
- **antivirus:** Checks the existence of one antivirus job for each client.
 - **Code:** GC__ANTIVI
 - **Action:** Configure an antivirus job for your clients, specially for file servers.
- **consoles:** Checks the usage of restricted consoles.
 - **Code:** GC__CONSOL
 - **Action:** If you have different users accessing your Bacula environment, configure a restricted console for each of them, using only the minimal needed permissions
- **events:** Checks the activation of Events in Message resources for auditing purposes.
 - **Code:** GC__EVENTS
 - **Action:** Enable events messages in your configuration and store them at your convenience (it is recommended to store them in a file and also in the catalog)
- **dir_status:** Checks the status of the Director daemon to see if there is any reported error with the service.

- **Code:** GC__DIR_ST
 - **Action:** Review the status of your Director service and start or restart it
- **dir_address:** Checks the usage of DirAddress setting to limit Director service to be listening on specific interfaces
 - **Code:** GC__DIR_AD
 - **Action:** Use the DirAddress setting, so you limit the service to be listening only on the required interface
- **sd_status:** Checks the status of the Storage Daemon(s) to see if there is any reported error with the service or if there is no connectivity with some of them.
 - **Code:** GC__SD_STA
 - **Action:** Review the status of the affected Storage Daemon and the connectivity to it from the Director host. Start or restart the service if needed
- **sd_free:** Checks free space in each Storage Daemon is above the threshold (defined by min_free_space_percent parameter).
 - **Code:** GC__SD_FREE
 - **Action:** Review the status of the affected Storage Daemon and start or restart it
- **dedup:** Enable getting dedup status for Storage Daemons in order to make dedup checks around Global Endpoint Deduplication. (Requires to not exclude sd_status)
- **ded_errors:** Check if GED is reporting some general error. (Requires to not exclude dedup)
 - **Code:** GC__DED_ER
 - **Action:** Review the message and the status of the affected Storage Daemon. Restart it, run vacuum procedures and re-check
- **ded_orphan:** Check if GED is reporting some orphan reference. (Requires to not exclude dedup)
 - **Code:** GC__DED_OR
 - **Action:** Run vacuum as soon as possible. If it is not solved, run scrub process.
- **ded_vacuum:** Check if GED vacuum procedure was executed recently enough. The number of days can be controlled with max_days_vacuum parameter. (Requires to not exclude dedup)
 - **Code:** GC__DED_VA
 - **Action:** Run vacuum as soon as possible.
- **ded_idx:** Check if GED is marking some error with the indexes. (Requires to not exclude dedup)
 - **Code:** GC__DED_ID
 - **Action:** Review the status of the filesystem holding the indexes.
- **ded_miss:** Check if GED is marking some missed reference. (Requires to not exclude dedup)
 - **Code:** GC__DED_MI
 - **Action:** Run vacuum as soon as possible. If it is not solved, run scrub process.
- **ded_free:** Check the free space available in GED engine (it relies on parameter dedup_free_space_min_bytes). (Requires to not exclude dedup)
 - **Code:** GC__DED_FR
 - **Action:** Provide more space to your GED containers filesystem. You can also purge data from your dedup storages and then run vacuum process to try to recover some space.

- **ded_suspect:** Check if GED is reporting some suspect reference. (Requires to not exclude dedup)
 - **Code:** GC__DED_SU
 - **Action:** Run vacuum as soon as possible. If it is not solved, run scrub process.
- **ded_derr:** Check if GED is reporting errors in the dedup engine. (Requires to not exclude dedup)
 - **Code:** GC__DED_DE
 - **Action:** Run vacuum as soon as possible. If it is not solved, run scrub process.
- **ded_cerr:** Check if GED is reporting errors in the containers. (Requires to not exclude dedup)
 - **Code:** GC__DED_CE
 - **Action:** Run vacuum as soon as possible. If it is not solved, run scrub process.
- **fd_status:** Checks the status of the client File Daemon(s) to see if there is any reported error with the service or if there is no connectivity with some of them.
 - **Code:** GC__FD_STA
 - **Action:** Review the status of the affected File Daemon and the connectivity to it from the Director host. Start or restart the service if needed
- **fips:** Checks if FIPS is enabled on daemons supporting it (for DIR requires to not exclude 'dir_status' ; for FDs requires to not exclude 'fd_status' ; for SDs requires to not exclude 'sd_status').
 - **Code:** GC__FIPS
 - **Action:** Consider enabling FIPS to the affected Daemon if your security posture needs to be very high
- **trace:** Checks if trace is enabled in any daemon with the risk of fulling a disk (for DIR requires to not exclude 'dir_status' ; for FDs requires to not exclude 'fd_status' ; for SDs requires to not exclude 'sd_status').
 - **Code:** GC__TRACE
 - **Action:** Disable debug and trace from the affected Daemon as soon as possible if you are not doing debug activities anymore
- **versions:** Checks if the FDs and/or SDs Bacula versions are aligned with the version of the Director (for FDs requires to not exclude 'fd_status' ; for SDs requires to not exclude 'sd_status').
 - **Code:** GC__VERSIO
 - **Action:** Install a supported Bacula version in the affected Daemon. Storage Daemon and Director must be on the same version, while File Daemons can run an older version than the Director.
- **security_plugin:** Checks if the security plugin is deployed in each FD (requires to not exclude 'fd_status').
 - **Code:** GC__SECURI
 - **Action:** Install the security plugin in the affected File Daemons
- **permissions:** Checks if permissions are strong enough for the given path and subpaths (permissions_base_paths parameter).
 - **Code:** GC__PERMIS
 - **Action:** Correct the permisisions on the affected paths. Usually you need to exclude the 'others' group from any bacula directory and to protect the bacula configuraton from undesirable writes.
- **running_processes:** Checks if running processes are running with root user, which is generally not recommended for secure environments.
 - **Code:** GC__RUNNIG

- **Action:** Configure your daemons with the correct user. Director and Storage Daemon do not need to be run with root.
- **encryption:** Check if encryption is used at all in the environment
 - **Code:** GC__ENCRYP
 - **Action:** Consider using encryption for any sensitive data or any untrusted storage.
- **volprotection:** Check if volume protection is used at all in the environment
 - **Code:** GC__VOLPRO
 - **Action:** Consider enabling volume protection for you disk backup over linux, as well as any backup sent to NAS from NetApp, DataDomain or HPE StoreOnce.
- **pg_vacuum:** Check if PostgreSQL vacuum process was executed recently enough on the key tables
 - **Code:** GC__PG_VAC
 - **Action:** Run Vacuum on the affected tables as soon as possible, during a low load window in your environment.
- **pg_analyze:** Check if PostgreSQL analyze process was executed recently enough on the key tables
 - **Code:** GC__PG_ANA
 - **Action:** Run Analyze on the affected tables as soon as possible, during a low load window in your environment.
- **pg_config:** Check if PostgreSQL configuration parameters are inside the recommended margins
 - **Code:** GC__PG_CON
 - **Action:** Correct the mentioned values to comply with the recommended configuration

Infected service

This service is activated if the service parameter contains the keyword: **infected**.

Alert code is: **GIN**

The purpose of this service is to reports jobs where some virus, ransomware or malware were detected, so a summary of them is easily available while a new alert for any new entry will also be generated.

Action: If you find any job containing some kind of malware or virus you should quickly isolate that system from your network and run healing activities on it. After, run a new backup and check that no more virus or malware are detected.

Security events service

This service is activated if the service parameter contains the keyword: **securityevents**.

Alert code is: **GSE**

This service will report any recent event registered in Bacula core with the security category.

Option	Re-quired	De-fault	Val-ues	Ex-am-ple	Description
securi-tyevents_days_since	No	15	Inte-ger	30	Defines the number of days to consider, from today, for the report of security events

Action: Review the nature of the event and act in consequence. If you find, for instance, many failed attempts to connect to the Director from BConsole, consider to change the location of your consoles or improve the security posture at networking level for them.

Deviation service

This service is activated if the service parameter contains the keyword: **deviation**.

Alert code is: **GDV**

The purpose of this service is to analyze job executions statistically and find deviation from the expected values. Calculations are done over the size, number of files and duration of the jobs.

Depending on what kind of jobs and the nature of data of your environment, you may need to adjust the following parameters to maximize the utility of the deviation results. It is possible to adjust the different thresholds, as well as to decide if results should only be listed following regression deviation and deviation from average (default behavior) or exclude deviation from average if it is generating too much information and it is not pointing to issues in the environment (`dev_include_by_avg` parameter).

Parameters for deviation service are explained below:

Option	Required	Default	Values	Example	Description
<code>dev_factor</code>		0.4	Float	0.25	Defines what jobs that will trigger the alert of deviation. It means what relation with the calculated standard deviation is considered significant enough. 1 means 200% of the standard deviation, 0.5 means 150% of the standard deviation. Example: If the standard deviation for size is 100Mb, with a value of 0.5: A job with a deviation from the average or regression of 160Mb will be selected, a job with a deviation from the average of 50Mb won't be selected, a job with a deviation from the average of 120Mb won't be selected
<code>dev_severity_low_limit</code>		0.8	Float	0.8	Defines the limit to consider a selected deviated job as severity Low
<code>dev_severity_medium_limit</code>		0.9	Float	0.9	Defines the limit to consider a selected deviated job as severity Medium
<code>dev_min_regression_accuracy</code>		0.4	Float	0.4	Minimum value of regression accuracy in order to use the regression analysis
<code>dev_min_dev_from_avg</code>		2	Integer	2	Minimum deviation from the average to consider a job as deviated for selection based on average
<code>dev_min_duration</code>		3600	Integer	3600	Minimum duration of a job in order to consider deviation by duration as something significant to select the job
<code>dev_min_executions</code>		20	Integer	20	Minimum number of executions of a given job in order to consider it for deviation analysis
<code>dev_include_by_avg</code>		1	1, true, yes, Yes; 0, false, no, No	false	Enable/disable selection of results based on the average deviation
<code>dev_min_files</code>		50	Integer	100	Minimum number of files in a job in order to consider deviation by number of files as something significant to select the job
<code>dev_min_size</code>		10485760	Integer	52428800	Minimum size of a job in order to include it in deviation analysis

Action: When this service reports results, you should review every result and analyze what caused the given deviation. A deviation can be caused by many reasons, like sudden new information to backup, a sudden slowness problem,

some controlled massive deletion... However, the same effect can be caused from ransomware activities or even a not controlled or desired user activity. If you find such event, you will need to solve it and consider adjusting or re-running some backups. If there is an explanation, every issue can be marked to be ignored in further alerts through the ignoring mechanism.

Failed in a row service

This service is activated if the service parameter contains the keyword: **failedinarow**.

Alert code is: **GFR**

The purpose of this service is to report jobs that have failed N or more times in a row, according to the parameter `failedrow_times`.

Option	Required	Default	Values	Example	Description
failedrow_times	No	3	Integer	5	Defines threshold of times a given job failed in order to select it to be included in the report

Action: Review as soon as possible the affected jobs and analyze the causes of the failure. You can run them manually to check the result and then adjust the configuration or the schedule according to the results of your analysis.

Restore frequency service

This service is activated if the service parameter contains the keyword: **restorefrequency**.

Alert code is: **GRF**

The purpose of this service is to report jobs whose restore frequency is below the factor established by the parameter: `restore_factor`.

Option	Required	Default	Values	Example	Description
restore_factor	No	0.15	Float	0.5	Defines threshold of restoring frequency for a given job in terms of %

Action: Review your backup policies and include some periodic restores on it, in order to ensure your backups and restore strategies are correct and agile enough to be correctly prepared for the time when an urgent restore comes.

Success ratio service

This service is activated if the service parameter contains the keyword: **successratio**.

Alert code is: **GSR**

The purpose of this service is to report jobs whose successratio is below the factor established by the parameter: `success_factor`.

Option	Re-quired	De-fault	Val-ues	Ex-ample	Description
success_factor	No	0.8	Float	0.75	Defines threshold of success ratio for the executions of a given job in terms of %
success_severity_medium_limit	No	0.4	Float	0.5	Defines the limit to consider a selected deviated job as severity Medium
success_severity_low_limit	No	0.6	Float	0.7	Defines the limit to consider a selected deviated job as severity Low

Action: Review the affected jobs and analyze the causes of the failures. If they are apparently random, consider to run a load analysis over your system in order to spread better the load over your network and hosts.

No Copy service

This service is activated if the service parameter contains the keyword: **nocopy**.

Alert code is: **GNC**

The purpose of this service is to reports jobs not included in any 2-tier policy, which means jobs that have never been copied or migrated.

Option	Re-quired	De-fault	Val-ues	Ex-ample	Description
no-copy_grace_period_days	No	10	Integer	10	Jobs that are more recent than the days defined by this parameter won't be included in the report

Action: Review your backup policies and include a 2-Tier storage where sending the reported jobs through the configuration and execution of Copy jobs.

No Verify service

This service is activated if the service parameter contains the keyword: **noverify**.

Alert code is: **GNV**

The purpose of this service is to reports jobs not included in any verification policy, which means jobs that have never been verified.

Option	Re-quired	De-fault	Val-ues	Ex-ample	Description
noverify_grace_period_days	No	20	Integer	50	Jobs that are more recent than the days defined by this parameter won't be included in the report

Action: Review your backup policies and include a verification phase where you run periodic verify jobs. Include the listed jobs in the report.

Empty service

This service is activated if the service parameter contains the keyword: **empty**.

Alert code is: **GE**

The purpose of this service is to reports Full jobs that were successful but have no contents (no files and no bytes stored).

Action: Review the affected jobs, including the joblog and the configuration. You may need to adjust the configuration or to run again the affected jobs.

Last good service

This service is activated if the service parameter contains the keyword: **lastgood**. Alert code is: **GLG**

The purpose of this service is to reports jobs where its last successful execution is older than the number of days specified by the parameter: `lastgood_max_since_days`.

Option	Re-quired	De-fault	Val-ues	Ex-ample	Description
last-good_max_since_days	No	5	In-te-ger	10	Jobs that have no successful execution more recent than the days defined by this parameter will be included into the report

Action: Review as soon as possible the affected jobs and their latest executions. Run them manually if they have been missed, cancelled or failed.

Low Dedup

This service is activated if the service parameter contains the keyword: **lowdedup**.

Alert code is: **GLD**

The purpose of this service is to provide a report of jobs that are not having good enough deduplication and that are potentially miss-using resources for information that are not a good candidate to be deduplicated. This information is also useful if running out of space and need to delete or migrate the information of some jobs that are using a good amount of storage.

Option	Re-quired	De-fault	Val-ues	Ex-ample	Description
dedup_ratio_min	No	40	Float	60	Defines threshold of dedup ratio. Jobs with a lower ratio will be reported. The threshold represents a percentage and jobs store it as 'compressratio' in the catalog
dedup_size_min	No	52428800	Long	262144000	Defines the limit in size to consider a selected low dedup job. Jobs with smaller size won't be considered

Action: Consider disabling deduplication for the affected jobs if the ratio is very poor for your needs. If running out of space, consider deleting (or moving to a different storage with a Migration) large jobs with a poor ratio that are old enough for your needs.

Orphan Chain

This service is activated if the service parameter contains the keyword: **orphanchain**.

Alert code is: **GOC**

Incremental and Differential jobs are part of a chain, and they are dependent on the information of a previous job. Sometimes, due to human mistakes or due to a bad retention policy, chains can be broken, and a dependent job is recycled before an after one. This service will detect this situation and report jobs that are ‘orphan’ in these terms.

It is important to note that depending on the backup nature, Incremental or Differential jobs alone can be still useful. For instance, for any job that contains files, the information is still fully recoverable, and they can also be based on a previous older Full or Incremental having the restore job still working. However, for some Virtual Machine or Database plugins, it is possible that one Incremental or Differential job without their predecessor will not contain recoverable information. In general, it is important to try to avoid having any orphan job and this service is intended to help in that direction.

Action: If you ever detect an orphan job, review your backup policies regarding retention times and adjust them, if necessary, to not be automatically producing any orphan job. Check also for the affected jobs that you have other valid copies of the information, if you don’t, run new jobs as soon as possible.

Different filesystem

This service is activated if the service parameter contains the keyword: **differentfilesystem**.

Alert code is: **GDFS**

This service will detect and report jobs where the log message ‘xxx is a different filesystem. Will not descend from yyyy’ is produced. Paths reported there will be compared with the list of excluded ‘known’ paths configured by ‘different_fs_exclude’ parameter. If they do not match, jobs will be reported.

This situation happens when jobs are using filesets with OneFS option enabled. Depending on the environment, this behavior is absolutely desirable, but can hide some non desired exclusion. This service helps to avoid those kind of situations.

Option	Re- quired	De- fault	Values	Example	Description
different_fs_exclude	No	/proc, /sys, /tmp, /boot	List of paths separated by ‘,’	/proc, /sys, /tmp, /boot, /myNonDesiredFS2, /mnt/myNonDesiredFS1,	List of known paths that are on different filesystems and there is no problem if they are reported as paths that won’t be backed up

Action: Review the path that was excluded and consider modifying your backup configuration if it is necessary to include it, or add it to the list to be excluded on this service otherwise.

NoTOTP service

This service is activated if the service parameter contains the keyword: **nototp**.

Alert code is: **GNT**

The purpose of this service is to report users that have not enabled TOTP 2-Tier authentication mechanism.

Action: Consider to enable the TOTP 2-Tier authentication mechanism for the reported users in order to improve your security posture regarding BWeb access.

Go back to the [main configuration page](#).

5.3 BWeb

BGuardian is connected to BWeb in two forms.

The first of them is about the links that generates automatically to be able to open the joblogs of the reported job entries in the html reports. This feature can be disabled with the parameter: `bweb_jobid_link`

The second is the ability to open directly the html report from BWeb. To accomplish this goal, BGuardian can generate a symlink to the reports directory inside the proper BWeb directory, as it is shown below:

Listing 6: **Reports symlink**

```
# ls -l /opt/bweb/html/.reports
lrwxrwxrwx 1 root root 39 jul 28 11:12 /opt/bweb/html/.reports -> /tmp/regress/working/
↳ bguardian/.reports
```

That symlink can be created manually, but if we run once BGuardian with the root user, the symlink will also be created. Note that if you run BGuardian from an AdminJob as we recommend, BGuardian will be executed by the ‘bacula’ user.

Once BGuardian detects the presence of BWeb and the symlink, the output of the script, available in the AdminJob joblog will show this kind of lines:

Listing 7: **HTML report in BWeb**

```
Open html report from: https://your.bweb.host.name:9180/.reports/Report__2023-07-28_
↳ 051730.html
```

As a result, you can copy that URL and directly see the report in your web browser.

Future versions of BGuardian and BWeb will expand this integration with more features, like listing the available reports and allowing to directly click on the links to see them.

Go back to the *main configuration page*.

See also:

- [BGuardian Scope](#)
- [BGuardian Features](#)
- [BGuardian Architecture](#)
- [BGuardian Installation](#)
- [BGuardian Operations](#)
- [BGuardian Best Practices](#)
- [BGuardian Limitations](#)
- [BGuardian Troubleshooting](#)

Go back to the *BGuardian plugin main page*.

6 Operations

The following article describes details regarding the different operations of **Bacula Enterprise BGuardian Plugin**, which essentially are to generate detailed reports and alerts with grouped information coming from the details of those reports.

6.1 Reports

Once BGuardian completes the different analysis that has been configured for it will produce a report of what has been found in different formats:

- It will directly output the detailed information to STDOUT in human-friendly text format. If it was invoked from a job, these output will be visible in the joblog.
- It will produce a computer-friendly report inside the configured reports_path, this is in json format.
- It will produce a friendly HTML report with the essential information inside the configured reports_path.

Text and json reports will contain the following data:

- A summary of the services that were run
- Produced errors, if any
- Version of BGuardian
- Date of the report
- Summary of the configuration used
- Summary of alerts generated
- List of issues found organized by service
- Totals for ignored results, passed checks and alerts
- Paths of the generated Reports

In general, the format of the issues is structured like this:

Listing 8: **Events Configuration**

```
Severity | Code | Entity+Details | Description.
```

Where:

- **Severity**: Reflects the relative relevance of the issue. It can be High, Medium or Low
- **Code**: Identifies the issue in a unique form. This code can be used to call the alerts function and ignore the issue in future executions
- **Entity** and **details** will represent the element affected by the issue (usually a job, a daemon name or a user).
- **Description**: Shows a message describing the situation

Below there is an output text example:

Listing 9: **BGuardian Example**

```
$ sudo -u bacula /bin/bash /opt/bacula/bin/bguardian  
Cleaning old reports in /tmp/regress/working/bguardian/.reports...  
Running service: configurationsecurity
```

(continues on next page)

```

Running service: deviation
Running service: successratio
Running service: failedinarow
Running service: empty
Running service: nocopy
Alert: GNC__guardianjob partially recovered:GuardianJob
Alert: GNC__guardianjob recovered
Running service: noverify
Alert: GNV__guardianjob partially recovered:GuardianJob
Alert: GNV__guardianjob recovered
Running service: restorefrequency
Running service: nototp
bweb_user not found
===== BGUARDIAN Report =====
Version: 1.0.0
Report Date: 2023-06-16 12:23:21
===== Config =====
ALERT_OPERATION : LIST
MODE : CHECK
REPORTS_KEEP_NUMBER : 100
SUCCESS_SEVERITY_LOW_LIMIT : 0.6
DEV_MIN_EXECUTIONS : 5
DEV_INCLUDE_BY_AVG : true
...
=====
===== Active alerts =====
GC__CATA | LOW | Service: configurationsecurity | Entity: catalog_backup
GC__CONF | LOW | Service: configurationsecurity | Entity: config_backup
GC__CONS | LOW | Service: configurationsecurity | Entity: consoles
GC__COPY | LOW | Service: configurationsecurity | Entity: copy
GC__EVEN | LOW | Service: configurationsecurity | Entity: events
GC__MALW | LOW | Service: configurationsecurity | Entity: malware
GC__PASS | LOW | Service: configurationsecurity | Entity: passwords
GC__PERM | LOW | Service: configurationsecurity | Entity: permissions
GC__REST | LOW | Service: configurationsecurity | Entity: restore
GC__SECU | LOW | Service: configurationsecurity | Entity: security_plugin
GC__VERI | LOW | Service: configurationsecurity | Entity: verify
GRF__guardianjob | LOW | Service: restorefrequency | Entity: guardianjob
GSR__guardianjob | LOW | Service: successratio | Entity: guardianjob
=====
##### Service: Configuration security #####
HIGH | GC__CONF | Catalog Backup Job executions : Catalog Backup Job was not run last 10
↳days
HIGH | GC__CATA | Config Backup Job executions : Config Backup Job was not run last 10
↳days
MEDIUM | GC__COPY | 2-Tier Jobs executions : 2-Tier Jobs (Copy or Migration) were not
↳run last 15 days
MEDIUM | GC__EVEN | Audit events : Events are not enabled in any Director Message
↳resource. They are important to keep track of important events related with security
MEDIUM | GC__PASS__Fileset_MySQLDumpUser | Not protected plugin password : Fileset_
↳MySQLDumpUser contains a password or key directly inside the plugin line. It's
↳recommended to store it in an external protected file

```


(continued from previous page)

```

MEDIUM | GC__PERM__/_opt/bacula/lib | Too open permissions : Too open permissions found.
↳for path: /opt/bacula/lib
MEDIUM | GC__VERI | Verify Jobs executions : Verify Jobs were not run last 30 days
LOW | GC__MALW__GuardianJob | Malware protection : GuardianJob has not enabled Malware.
↳protection. It could be enabled, as fileset signature is compatible
LOW | GC__SECU__127.0.0.1-fd | Plugin security usage : 127.0.0.1-fd has no installed.
↳plugin security. This is recommended for security reasons
LOW | GC__REST | Restore Jobs executions : Restore Jobs were not run last 30 days
LOW | GC__CONS | Restricted consoles : No restricted console was found. If external.
↳connections are allowed, it is recommended to use restricted consoles for them
##### Service: Success Ratio #####
LOW | GSR__GuardianJob | Job: GuardianJob | Executions: 11 | Ratio: 63,6%
##### Service: Restore frequency #####
MEDIUM | GRF__GuardianJob | Job: GuardianJob | Executions: 7 | Restores: 0 | Ratio: 0
=====
Ignored results: 0
Passed checks: 11
Alerts: 13
=====
Json report built in: /tmp/regress/working/bguardian/.reports/Report__2023-06-16_122324.
↳json
Html report built in: /tmp/regress/working/bguardian/.reports/Report__2023-06-16_122324.
↳html

```

The HTML report will reflect the issue information in a summarized way, with collapsible blocks with a more friendly format. It is also possible to see a summary of the issues grouped by severity.

Here we show an example HTML report:

BGuardian: Issues Report Date: 2023/06/16 12:31:40

Configuration security: Reports the result of different checks around security related with Bacula Configuration


Check	Details	Code	Severity
Catalog Backup Job executions	Catalog Backup Job was not run last 10 days	GC__CONF	HIGH
Config Backup Job executions	Config Backup Job was not run last 10 days	GC__CATA	HIGH
2-Tier Jobs executions	2-Tier Jobs (Copy or Migration) were not run last 15 days	GC__COPY	MEDIUM
Audit events	Events are not enabled in any Director Message resource. They are important to keep track of important events related with security	GC__EVEN	MEDIUM
Not protected plugin password	Fileset_MySQLDumpUser contains a password or key directly inside the plugin line. It's recommended to store it in an external protected file	GC__PASS_Files...	MEDIUM
Too open permissions	Too open permissions found for path: /opt/bacula/lib	GC__PERM__/_opt/...	MEDIUM
Verify Jobs executions	Verify Jobs were not run last 30 days	GC__VERI	MEDIUM
Malware protection	GuardianJob has not enabled Malware protection. It could be enabled, as fileset signature is compatible	GC__MALW__Guard...	LOW
Plugin security usage	127.0.0.1-fd has no installed plugin security. This is recommended for security reasons	GC__SECU__127.0...	LOW
Restore Jobs executions	Restore Jobs were not run last 30 days	GC__REST	LOW
Restricted consoles	No restricted console was found. If external connections are allowed, it is recommended to use restricted consoles for them	GC__CONS	LOW

Deviation: Reports jobs presenting a significant deviation (in terms of size, duration or number of files) from the calculated estimated value

Jobid	Name	Level	Starttime	Size (read)	Size (write)	Files	Duration	Executions	Average	Message	Code	Severity
9	GuardianJob	F	2023-06-16 00:00:00	142,39 MIB	142,43 MIB	4,02 K	3s	5	3,41 MIB / 3,41 MIB w - 919 files - 0s	Significant deviations found: Job size (bytes read) increased 4080,2% from the expected estimated value of 3,41 MIB.	GDV__GuardianJo...	HIGH

Success Ratio: Reports jobs whose successratio is below the factor of 80%

Restore frequency: Reports jobs whose restore frequency is below the factor of 0,15%



Summary

High severity issues: **3**

Medium severity issues: **7**

Low severity issues: **4**

Fig. 2: BGuardian HTML Report

For the deviation service, not that it will visually mark what value (from files, size or time) has been increased or

decreased significantly enough to select and include the job into the report.

In future versions of Bacula, the Web User Interface will interpret this information and also make it directly accessible through the Web layer.

BGuardian will generate one json report and one html report for every execution by default. By default, will keep 100 reports of each kind before removing the oldest ones. This report rotation capability can be adjusted with the parameter `reports_keep_number`.

Go back to the [main operations page](#).

6.2 Alerts

On top of the reporting features described in the Reports section BGuardian also implements an Alert framework that will keep track of those issues through executions of the tool through the time. The purpose of this framework is to ease the management of the different issues and to provide the proper functions to see the status of an environment in a given point in time when BGuardian is run regularly.

Alerts will group the information of the issues by bigger entities. For example, the same job can present many records inside the deviation service, with different executions of it. In the issue report we will see individually each entry, while in the alerts framework we will only have a single alert for that jobname, grouping all the affected executions.

BGuardian allows to precisely select the services that are desired to be run. However, the backup administrator will find situations where the service is still interesting to be run, while there are some records that he/she acknowledged already and does not desire to see anymore in the reports or in the alert lists. Here is where the ignore feature comes in place. Using the code of a given issue (or part of it in some situations) it is possible to mark an issue or a group of them to be ignored in further executions.

In this section, the alerts structure, the different commands and soem examples of them are shown

Alerts structure

Alerts are store in the path defined by 'alerts_path' parameter. Each alert is represented by .json file, whose name is the code of the alert and the kind of it.

Listing 10: Alerts files

```
$ ls -l
total 56
-rw-rw-r-- 1 bac bac 343 jun 16 12:31 GC__CATA.on.json
-rw-rw-r-- 1 bac bac 342 jun 16 12:31 GC__CONF.on.json
-rw-rw-r-- 1 bac bac 396 jun 16 12:31 GC__CONS.on.json
-rw-rw-r-- 1 bac bac 324 jun 16 12:31 GC__COPY.on.json
-rw-rw-r-- 1 bac bac 394 jun 16 12:31 GC__EVEN.on.json
-rw-rw-r-- 1 bac bac 401 jun 16 12:31 GC__MALW.on.json
-rw-rw-r-- 1 bac bac 482 jun 16 12:31 GC__PASS.on.json
-rw-rw-r-- 1 bac bac 371 jun 16 12:31 GC__PERM.on.json
-rw-rw-r-- 1 bac bac 312 jun 16 12:31 GC__REST.on.json
-rw-rw-r-- 1 bac bac 406 jun 16 12:31 GC__SECU.on.json
-rw-rw-r-- 1 bac bac 310 jun 16 12:31 GC__VERI.on.json
-rw-rw-r-- 1 bac bac 1298 jun 16 12:31 GDV__guardianjob.on.json
-rw-rw-r-- 1 bac bac 219 jun 16 12:31 GRF__guardianjob.on.json
-rw-rw-r-- 1 bac bac 239 jun 16 12:31 GSR__guardianjob.on.json
```

The 'on.json' extension represents an active alert, while the 'off.json' extension represents an alert that will be ignored in future executions.

List alerts

The following command lists active alerts:

Listing 11: Active alerts

```
# Text mode
sudo -u bacula /bin/bash /opt/bacula/bin/bguardian list_text

# Json mode
sudo -u bacula /bin/bash /opt/bacula/bin/bguardian list
```

Example output in text format:

Listing 12: Active alerts

```
GC__CONF | LOW | Service: configurationsecurity | Entity: config_backup | Details: {
↪ "CONFIG_BACKUP":{"passed":false,"description":"Catalog Backup Job executions","details
↪ ":"Catalog Backup Job was not run last 10 days","subCheck":"CONFIG_BACKUP","severity":
↪ "HIGH"}}
GC__CONS | LOW | Service: configurationsecurity | Entity: consoles | Details: {"CONSOLES
↪ ":"passed":false,"description":"Restricted consoles","details":"No restricted console
↪ was found. If external connections are allowed, it is recommended to use restricted
↪ consoles for them","subCheck":"CONSOLES","severity":"LOW"}}
```

The structure of an alert is similar to the structure of an issue. It is composed by:

Listing 13: Alert structure

```
Code | Severity | BGuardian service | Affected entity | Issue details
```

The following command lists ignore alerts:

Listing 14: List Ignore alerts

```
# Text mode
sudo -u bacula /bin/bash /opt/bacula/bin/bguardian list_ignore_text

# Json mode
sudo -u bacula /bin/bash /opt/bacula/bin/bguardian list_ignore
```

Ignore alerts

The following command adds 'ignores', which means that issues matching the ignore code will be ignored from active alerts and from further BGuardian executions.

Listing 15: Ignore an alert

```
# One single code
sudo -u bacula /bin/bash /opt/bacula/bin/bguardian ignore code1

# Several codes at once
sudo -u bacula /bin/bash /opt/bacula/bin/bguardian ignore code1, code2, code3
```

Using the codes shown in list alerts, we could ignore them with the following commands

Listing 16: Ignore some alerts

```
sudo -u bacula /bin/bash /opt/bacula/bin/bguardian ignore GC__CONF, GC__CONS
```

There are services that have entities, but also more information. For example the deviation service works with jobnames and with job ids. Example:

Listing 17: Deviation issue

```
##### Service: Deviation #####  
HIGH | GDV__GuardianJob__9 | Job: 9 GuardianJob F 2023-06-16 00:00:00  
  | Size (read): 142,39 MiB +4080,1% | Size (write): 142,43 MiB  
  | Files: 4,02 K | Duration: 3s  
  | Executions: 5 | Average: 3,41 MiB r | 3,41 MiB w - 819 files - 0s  
  | Details: Significant deviations found: Job size (bytes read) increased 4080,1%  
↪from the expected estimated value of: 3,41 MiB.
```

It is possible to ignore here any execution of the GuardianJob. To do so:

Listing 18: Ignore alert by entity

```
sudo -u bacula /bin/bash /opt/bacula/bin/bguardian ignore GDV__GuardianJob
```

However, it is also possible to ignore only particular jobids:

Listing 19: Ignore alert by id

```
sudo -u bacula /bin/bash /opt/bacula/bin/bguardian ignore GDV__GuardianJob__9
```

Doing so, new deviated results of the same job will be still considered in next executions.

Manually remove alerts

It is possible to remove active alerts using a very similar format:

Listing 20: Remove active alert

```
sudo -u bacula /bin/bash /opt/bacula/bin/bguardian remove_alert GDV__GuardianJob__9
```

To remove something from the ignore list:

Listing 21: Remove ignore

```
sudo -u bacula /bin/bash /opt/bacula/bin/bguardian remove_ignore GC__CATA
```

Alerts recovery

When a situation marked by a given issue is solved, BGuardian will automatically remove the associated alert.

Events

In order to notify the backup administrator when an alert is created or recovered, BGuardian uses the Events feature. This means it will send an Message of type Event with the information of what happened.

Example of BGuardian event about permissions recovery (source 'bguardian'):

Listing 22: Events

```
*list events
Automatically selected Catalog: MyCatalog
Using Catalog "MyCatalog"
+-----+-----+-----+-----+-----+
| time           | daemon          | source          | type            | events |
+-----+-----+-----+-----+-----+
| 2023-06-16 13:20:57 | 127.0.0.1-dir | *Director*     | daemon         |      |
| Director configuration reloaded |
| 2023-06-16 13:21:34 | 127.0.0.1-dir | *Console*     | connection     |      |
| Connection from 127.0.0.1:8101 |
| 2023-06-16 13:21:34 | 127.0.0.1-dir | **bguardian** | configurationsecurity |      |
| BGuardian alert [GC__PERM] was recovered |
| 2023-06-16 13:21:34 | 127.0.0.1-dir | *Console*     | connection     |      |
| Disconnection from 127.0.0.1:8101 |
| 2023-06-16 13:21:37 | 127.0.0.1-dir | *Console*     | connection     |      |
| Connection from 127.0.0.1:8101 |
+-----+-----+-----+-----+-----+
```

Note that in order to have events feature working, it is needed to enable them in the Message resources as discussed in the Configuration section.

Go back to the [main operations page](#).

See also:

- [BGuardian Scope](#)
- [BGuardian Features](#)
- [BGuardian Architecture](#)
- [BGuardian Installation](#)
- [BGuardian Configuration](#)

- [BGuardian Best Practices](#)
- [BGuardian Limitations](#)
- [BGuardian Troubleshooting](#)

Go back to the [BGuardian plugin main page](#).

7 Best Practices

The following article presents best practices regarding usage and performance.

7.1 Usage

BGuardian performs a good number of different checks. Ideally, a backup environment would comply with all the rules of those checks and no issue or alert would be detected. However, in many cases, there are some external constraints around resources or about the nature of the data that can make it difficult to comply with some of them.

The main goal of this tool is to help the administrator to keep the system safe and to detect any non-desired behavior. To reach this goal, the tool should generate alerts or issues only when they are actually something that the administrator is going to change or review. This will be only possible if services producing results that will not be actually solved are deactivated, as well as marking those specific alerts that cannot either be solved as something to ignore.

The recommended usage cycle with this tool is:

1. Run it manually one time with default parameters.
2. From the reported information, select the services that are relevant for the given environment.
3. Configure BGuardian with the selection of relevant services.
4. Tune selected services, if needed, to reduce the number of results (adjust dates, factors...).
5. Configure an Admin Job to invoke BGuardian with all the resulting command line parameters needed as discovered in the previous steps.

Once the periodical Admin Job is in place, the user should be notified with the events of every alert creation. Then he should review the situation of a given alert and solve the problem if necessary, or mark the alert to be ignored otherwise.

Go back to the [Best Practices](#) article.

7.2 Performance

The performance of this plugin is mainly dependent on:

- The performance/response time of the catalog, which is highly dependent on the size of it, the underlying filesystem and the configuration at the database level
- The number of daemons (Storage Daemons, File Daemons) available in the environment
- The configured services to be executed
- The load of the host at the moment of BGuardian execution

In summary, it is not possible to establish an exact reference about how much time the daemon execution will take to complete. It usually should take some minutes to complete. However, there are some services that will take significant time to complete if the catalog is very large as the implied queries are heavy. These services are 'deviation' and, specially, 'orphanchain'. If there are some services taking too long, it is possible to define different parameters for different executions of the service. For example, we can run a set of services daily or even hourly, while running

different services once a week. This is easily set up by using different AdminJobs with command line parameters, testing different services, and then associating them with different schedules.

The general recommendation is to run the daemon once a day over a time window where the load of the backup environment is low. Following this principle, it should be possible to run smoothly this plugin using all the checks without inconvenience.

Go back to the *Best Practices* article.

See also:

- *BGuardian Scope*
- *BGuardian Features*
- *BGuardian Architecture*
- *BGuardian Installation*
- *BGuardian Configuration*
- *BGuardian Operations*
- *BGuardian Limitations*
- *BGuardian Troubleshooting*

Go back to the *BGuardian plugin main page*.

8 Limitations

The following article presents limitations of BGuardian Plugin.

While this tool is designed to help to monitor and detect important issues related with security and all kind of best-practices to keep an environment safe, it should not be used standalone as the only measure of protection.

As stated in the introductory article, many other mechanisms should be used to keep an environment safe, where doing things securely should be treated as a general culture put in place in each and every element of an organization, from software elements to human practices.

- BGuardian is only available for Bacula environments deployed over **PostgreSQL databases**.
- Catalog running on MySQL is not supported.
- The `restart` command has limitations with plugins, as it initiates the Job from scratch rather than continuing it. Bacula determines whether a Job is restarted or continued, but using the `restart` command will result in a new Job.

See also:

- *BGuardian Scope*
- *BGuardian Features*
- *BGuardian Architecture*
- *BGuardian Installation*
- *BGuardian Configuration*
- *BGuardian Operations*
- *BGuardian Best Practices*
- *BGuardian Troubleshooting*

Go back to the [BGuardian plugin main page](#).

9 Troubleshooting

In this article, there are suggested solutions to common situations that can cause trouble during the usage of the BGuardian plugin.

9.1 Log files and debug

The plugin supports a debug parameter and produces internal log files with details of what is happening. Increasing the debug parameter to a higher number will produce a more detailed output inside the plugin debug logs.

The location of the logs are controlled by the log parameter, by default they are placed in the working directory, inside 'bguardian' directory.

The internal plugin logging framework rotates log files automatically. Currently, each file can be 50Mb at maximum and the plugin will keep 25 files.

The ".err" file that can be found close to the log files, can show contents even if no real error happened in the jobs. It can show contents too even if debug is disabled. This file is not rotated, but it is expected to be a small file in general. If you still need to rotate it, you can include it in a general rotating tool like 'logrotate'.

9.2 Out of Memory

If you ever face *OutOfMemory* errors from the Java daemon (you will find them in the bguardian-debug.err file), you have probably a large environment with a large catalog and/or many different daemons to connect to.

To overcome this situation you can increase JVM memory, you will need to create the following file:

Listing 23: **Memory parameters file**

```
/opt/bacula/etc/bguardian_backend.conf'
```

Then, add the following parameters to the file:

Listing 24: **Memory parameters**

```
BGUARDIAN_JVM_MIN=2G  
BGUARDIAN_JVM_MAX=8G
```

Those values will define the MIN (BGUARDIAN_JVM_MIN) and MAX (BGUARDIAN_JVM_MAX) memory values assigned to the JVM Heap size. In this example we are setting 2Gb for the minimum, and 8Gb for the maximum. In general, those values should be more than enough to handle every situation.

The '/opt/bacula/etc/bguardian_backend.conf' won't be modified through package upgrades, so your memory settings will be persistent.

See also:

- [BGuardian Scope](#)
- [BGuardian Features](#)
- [BGuardian Architecture](#)
- [BGuardian Installation](#)

- *BGuardian Configuration*
- *BGuardian Operations*
- *BGuardian Best Practices*
- *BGuardian Limitations*

Go back to the *BGuardian plugin main page*.