Secret scanning

About secret scanning

While your team collaborates on code to build software, sensitive information such as passwords, tokens, private keys, environment variables, `.pem` files or other secrets may accidentally get added to your repositories. As soon as code containing secrets gets pushed into your repositories, secrets are added to the commit history and persist until they are revoked. Unidentified breaches may result in the potential use of secrets by any users with access to your repositories.

To improve the security of your repositories and help you make sure that secrets are not accidentally exposed in your code, Bitbucket scans your repositories for secrets and triggers notifications when leaked secrets are detected within new commits.

Email notifications are sent out to everyone involved in the commit of the secret: the authors, committers, and the developer who pushed or merged the code containing secrets into the repositories. Note that Bitbucket doesn't send email notifications externally. This means that Bitbucket will send an email only to:

- user that exists in your instance
- user that has a configured email in your instance
- user that has at least read access to a repository where a secret is detected.

Even if a mail server isn't configured in your instance, Bitbucket records an alert about the detected secret in the audit log, which you can access from **Administration** > **Audit log**, and in the `audit.log` file on the file system (`$BITBUCKET_HOME/log/audit`). Learn more about these records in the section Track alerts about leaked secrets.

Secret scanning is enabled by default in your Bitbucket instance, and both global and system admins can disable or enable secret scanning by modifying the configuration properties in the `bitbucket.properties` file.

Customize the scanner

The scanner makes use of default patterns to scan your repositories and can detect a majority of the generic secrets. You can customize the scanner to optimize the performance and reduce the number of false positives—modify or delete the default patterns or add your own regex patterns and file paths.

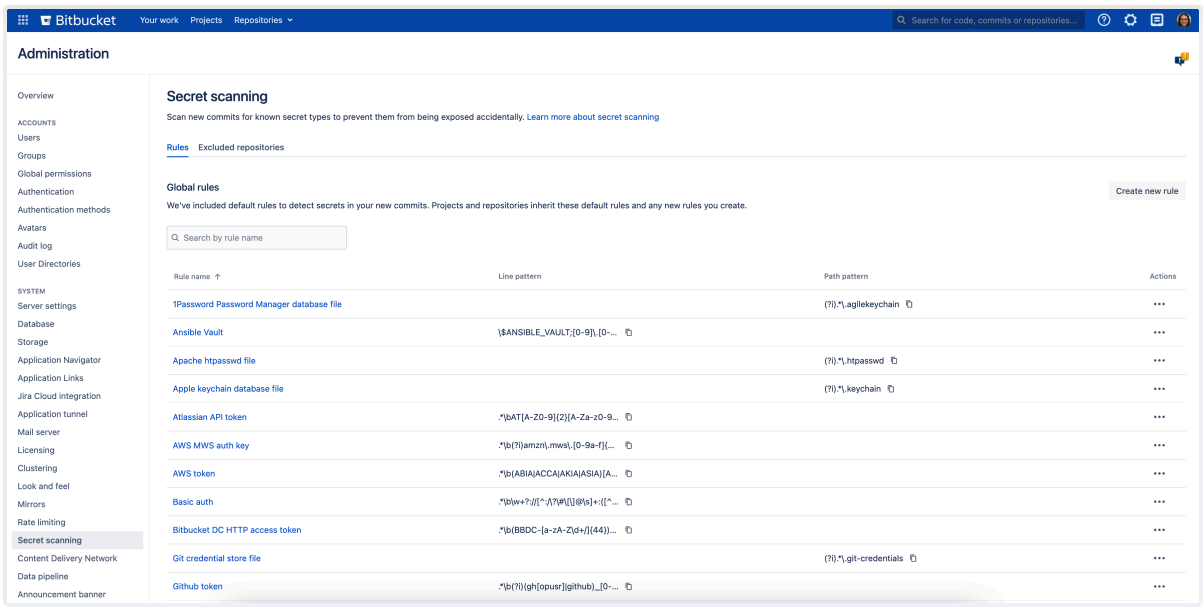
Project or repository admins, can further configure the scanner at each level. You can't remove the rules inherited from the higher levels unless you have admin permission to the original location (repository, project, global). Forked repositories will use their own rules and will not sync rules from the parent.

To customize:

1. From either the **System**, **Project**, or **Repository** settings, select **Secret scanning**.
2. Select **Create new rule** to add your own rule or select **More actions ...** > **Edit** to modify a default rule.
3. Enter the rule details such as name, including regex for **Line pattern** or **Path pattern**.

   When you specify both path and line pattern regex, scanning looks for the pattern only in the specified file paths.

4. Select **Save**.



Learn more about regex patterns and Bitbucket secret scanning rules

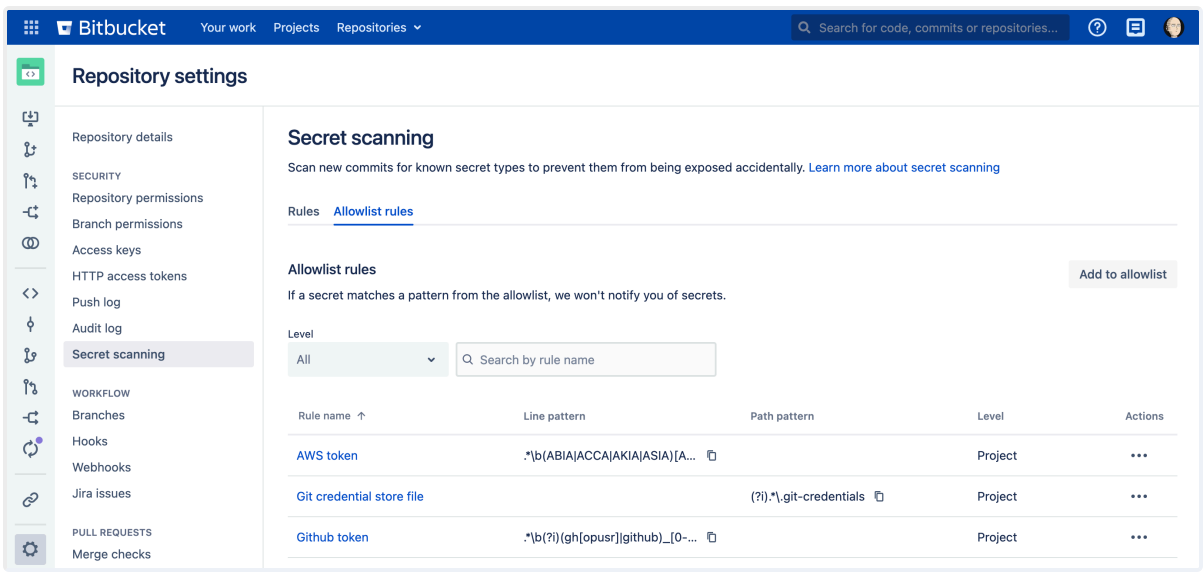
Customize allowlist rules

You can add an allowlist at the project or repository level to negate matches from scanning rules. Allowlist rule patterns can be defined the same as scanner rules. Any matches to your allowlist will not trigger a notification. Repositories inherit allowlists created at the project level.

> If scanning and allowlist rules match, the allowlist rule takes precedence.

To customize allowlist rules:

1. From either the **Project** or **Repository settings**, select **Secret scanning**.
2. Select **Allowlist rules** tab.
3. Select **Add to allowlist** to add a rule pattern.



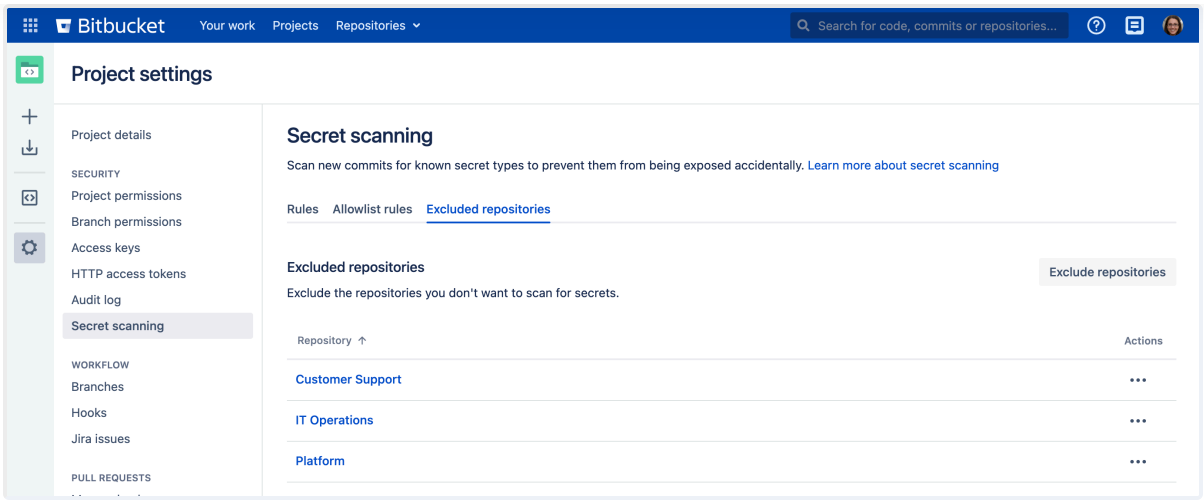
Exclude repositories from scanning

You can exclude specific repositories from scanning at the global or project level. Admins can also exclude all personal repositories at the global level. When excluded, Bitbucket won't scan any new commits added to the repository.

To exclude repositories from scanning:

1. From either the **System** or **Project** settings, select **Secret scanning**.
2. Select **Excluded repositories** tab.
3. Select **Exclude repositories** to exclude specific repositories from scanning.
4. Select **Exclude**.

To remove a repository, select **More actions ...** > **Remove**.



Resolve issues detected by secret scanning

Once a secret has been detected in a repository by the scanner, it is good practice to treat the secret as being compromised. Any secrets that are detected by the scanner should be invalidated in the tool they were created in (either through revoking, deleting, or rotating said secret) and new secrets should be generated in their place. Invalidating secrets may involve working with other teams in your organization such as DevOps or Security to make sure that existing systems and workflows are not impacted.

Removing the secret from Git history by force pushing does not guarantee that the commit is completely cleaned up. There are lots of edge cases (such as other branches, pull requests, forks or local copies) where the commit may still be referenced and so will never be completely removed from Git, even if it is not part of the main branch. Additionally, anyone with access to the repository may have already seen the secret and made a copy of it elsewhere. Therefore, **we don't recommend** that you revoke the secret from your commit history as the only preventative measure. The secret must always be revoked in the tool they were created in too.

If the detected secret is a false positive, contact your admin and ask them to make modifications to the secret scanner to reduce the chance of a false positive. Here are some ways to narrow down false positives:

- Modify the regex in a secret scanning rule (sometimes the regex used may be too permissive).
- Provide a path pattern to limit the files and directories scanned.
- Use the allowlists at the project and repository levels to specify patterns that aren't secrets.
- Exclude the repositories you don't want to scan for secrets.

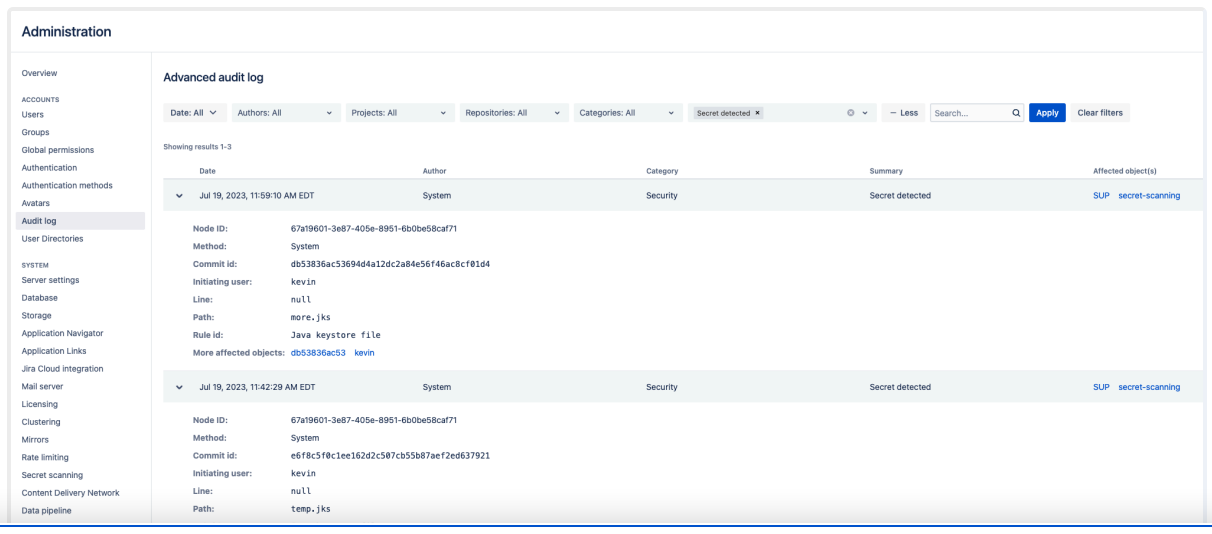
Track alerts about leaked secrets

Along with sending email notifications about detected secrets when a mail server is configured, Bitbucket also records these alerts in two locations:

- the audit log that you can access from **Administration** > **Audit log**
- the `audit.log` file on the file system (`$BITBUCKET_HOME/log/audit`).

Audit log in the user interface

To find alerts about detected secrets in the audit log, select **Administration** > **Audit log**. You can filter the list of alerts by selecting the **+ More** button and **Secret detected** for **Summary**.



```
Line: null
Path: more.jks
Rule id: Java keystore file
More affected objects: db53836ac53 kevin
----
## When a secret is found within a file:
...
Line: 1
Path: env
Rule id: Basic auth
More affected objects: 0a1fea57765 kevin
```

The audit.log file

In `$BITBUCKET_HOME/log/audit`, Bitbucket includes a JSON record for each audit event. If secret scanning is triggered, a new record with the relevant information will be appended. If pragmatically parsing these JSONs, we may want to look inside the `auditType` key for entries similar to the following:

```
...
  "auditType": {
    "action": "Secret detected",
    "actionI18nKey": "bitbucket.secretscanning.audit.action.secretdetected",
```

Secret scanning performance

While only new commits are scanned as soon as they're pushed, the performance of secret scanning can be affected by a few other factors along with the number of pushed commits:

- the size of the commit diff
- the number of scanning rules
- the complexity of scanning rules
- the complexity of allowlist rules
- the high load caused by third-party plugins or REST API activity in your instance.

A system administrator can track the performance of secret scanning with the properties from the `application-default.properties` file that are described in the following table.

Property	Default value	Description
`secretscanning.max.threads`	`${scaling.concurrency}` – the number of detected CPU cores	Controls the maximum number of threads allowed per node to perform secret scanning
`secretscanning.scan.timeout`	1000	Controls the timeout in milliseconds for streaming the diff for a commit and detecting any secrets
`secretscanning.scan.batch.size`	200	Controls the number of commits sent to the executor to be scanned for secrets. If the number of commits pushed exceeds the batch size, multiple batches will be sent.
`secretscanning.scan.commit.limit`	10000	Controls the maximum number of commits to be scanned in a single push
`secretscanning.email.maxsecrets`	100	Controls the maximum number of detected secrets that a single

Bitbucket Data Center 9.3 (Latest)
Documentation

Get started with Bitbucket Data Center
Use Bitbucket Data Center
Administer Bitbucket Data Center
Users and groups
Advanced repository management
External user directories
Global permissions
Setting up your mail server
Integrate with Atlassian applications
Connect Bitbucket to an external database
Migrating Bitbucket Data Center to another server
Migrate Bitbucket Server from Windows to Linux
Run Bitbucket in AWS
Specify the Bitbucket base URL
Configuring the application navigator
Managing apps
View and configure the audit log
Monitor security threats
Update your license key
Configuration properties
Change Bitbucket's context path
Data recovery and backups
Disable HTTP(S) access to Git repositories
Mirrors
Bitbucket Mesh

Related content

Configuration properties

Bitbucket OAuth 2.0 provider API

Commit checker for Jira issues

Get started with Git

Export and import projects and repositories

How to update your add-on

Controlling access to code