

Sonatype CLM Server - Policy Management

Contents

1	Introduction	1
2	What is Sonatype CLM?	3
2.1	The Four Criteria of Governance	3
2.2	Enforcement Points and Communication	4
2.3	Summary	5
3	What is a Policy?	6
3.1	Basic Policy Anatomy	6
3.2	Organizations, Applications and Inheritance	7
3.3	Summary	8
4	Organization and Application Management	10
4.1	Organizational Structure	10

4.2	Creating an Organization	11
4.3	Creating an Application	12
4.4	Organization, Application, and Inheritance	14
4.5	The Power of Inheritance	14
4.6	Avoiding Policy Micromanagement	14
4.7	Permissions and Roles	15
4.8	Summary	16
5	Policy Development	17
5.1	Advanced Anatomy of a Policy	17
5.2	Risk and Organizational Intent	20
5.3	Summary	20
6	Policy Creation	22
6.1	Getting Started	23
6.2	Step 1: Understand the Policy Intent	24
6.3	Step 2: Decide on a Descriptive Policy Name	24
6.4	Step 3: Choose an Appropriate Threat Level	26
6.5	Step 4: Choose the Application Matching Parameters	27
6.6	Step 5: Create Constraints with Conditions	28

6.7	Step 6: Set Policy Actions	33
6.8	Summary	35
7	Policy Elements	37
7.1	What is a Label?	37
7.2	Creating, Editing and Deleting a Label	38
7.3	Creating a Condition Based on a Label	40
7.4	What is a License Threat Group?	41
7.5	Creating, Editing, and Deleting a License Threat Group	42
7.6	Creating a Condition Based on a License Threat Group	45
7.7	Creating a Condition Based on an Unassigned License Threat Group	45
7.8	What is a Tag?	46
7.9	Creating, Editing, and Deleting Tags	48
7.10	Applying a Tag	50
7.11	Matching Policies to Specific Applications	51
7.12	Viewing Tag-based Policies	52
7.13	Summary	53
8	Manual Application Evaluation	54
8.1	Evaluating via the CLM Server	54

8.2	Successful Evaluations and Report Generation	56
8.3	Summary	57
9	Reviewing Evaluation Results	58
9.1	Accessing the Application Composition Report	59
9.2	Reviewing the Report	61
9.3	Summary	66
10	Importing Policies	67
10.1	Sonatype Sample Policy Set	67
10.2	Importing a Policy to an Organization	68
10.3	Importing a Policy to an Application	70
10.4	Summary	70
11	Policy Monitoring	72
11.1	Setup Policy Monitoring for an Application	73
11.2	Configuring Notification Times	77
11.3	Summary	78
12	Conclusion	79

List of Figures

4.1	Using New Organization button	11
4.2	Using Global Create Button	11
4.3	Using New Application button	12
4.4	Using Global Create Button	12
5.1	Editing a Policy and its Attributes	18
6.1	Using New Policy Button	23
6.2	Using Global Create Button	23
6.3	Naming the Policy	25
6.4	Editing the Policy Threat Level	26
6.5	Example Constraint	28
6.6	Adding Constraints	29

6.7	Policy Actions Example	33
6.8	Setting Policy Actions	34
7.1	Using New Label Button	38
7.2	Using Global Create Button	39
7.3	Label Example	39
7.4	Creating a Label Condition	40
7.5	Using New License Threat Group Button	42
7.6	Using Global Create Button	43
7.7	Creating a License Threat Group	44
7.8	Creating a Condition Evaluating a License Threat Group	45
7.9	Creating a Condition Evaluating an unassigned License Threat Group	46
7.10	Example of Applied Tags	47
7.11	Using New Tag Button	48
7.12	Using Global Create Button	48
7.13	Creating a Tag	49
7.14	Example of Tags with Description	50
8.1	Evaluate an Application	56
8.2	Violations Report after Scan	57

9.1	Reporting Area	60
9.2	Application Area	61
9.3	Summary Tab of an Application Composition Report	62
9.4	Policy Tab of an Application Composition Report	63
9.5	Security Issues Tab of an Application Composition Report	63
9.6	License Analysis Tab of an Application Composition Report	64
9.7	Component Information Panel CIP for a Specific Component	64
9.8	Policy Section for a Specific Component Displayed on the Component Information Panel	65
10.1	Organization View with Import Button	69
10.2	Import Policy Dialog	69
11.1	Example of a Policy Monitoring Email	73
11.2	Access Application Management Area	74
11.3	Selecting a Sonatype CLM Stage to Monitor	75
11.4	Adding Email Recipient	76
11.5	Sample Email Notification	77

List of Tables

6.1 Threat Levels	26
-----------------------------	----

Chapter 1

Introduction

Sonatype CLM is a powerful system for improving how your teams consume open source components. For the first time, you have the ability to take pen and paper rules, and turn them into enforceable policies for component usage within your enterprise. Better yet, all of this can be done simply and easily, with results that provide a detailed analysis of the health of your applications. And this happens not in days or months, but in most cases, just a few seconds.

While there are options for getting Sonatype CLM going right *out of the box* with our sample policies and policy elements, the road to designing and refining your own custom policies is where you want to be.

Before we get there though, we should first take a closer look, defining every part of the policies, dissecting it really, and explaining it in detail. Of course, we'll also walk you through the stages of importing policies as well.

To get started, let's see what you'll find in this guide:

- Introduction to Policy, Governance, and CLM
 - Organization and Application Management
 - Policy Development and Management
 - Policy Element Overview and Usage
 - Manual Application Scanning and Evaluation
-

- Basic Reporting
- Importing Policy

Remember, before you get started you need to make sure you have, at a minimum:

- Installed Sonatype CLM Server (or Nexus Pro: Sonatype CLM Edition)
- Installed your Sonatype CLM License

Chapter 2

What is Sonatype CLM?

CLM is simply the acronym for **Component Lifecycle Management**. In much the same way that software follows a process that results in an application, components have their own process as well.

Where various development environments, build systems, and other support tools are used in software development, Sonatype CLM supports the management of component usage throughout your development lifecycle. Rather, it is the set of processes that include development, production, and everything else in between. We tend to call these processes, the ones that manage a component's lifecycle, governance.

Governance can be a big undertaking though, and can often mean different things within an organization, so let's distill it down a bit.

2.1 The Four Criteria of Governance

Governance is really about putting the power of information back into the hands of the business and the development teams under its direction. More over, it's about four key criteria:

- Creating an Inventory of Applications
 - Establishing Rules
 - Choosing Appropriate Actions
-

- Making Adjustments

First, we have to create an inventory of our applications and the components within those applications. Next, we work towards determining which components are undesirable to our business, as well as those that are preferred. This is because the overarching goal of governance is founded in creating rules that encourage good component usage, while simultaneously discourage bad component usage. These rules form the foundation of your policy, because they define what a bad component looks like.

It doesn't stop with rule creation though. In fact, once we have established our rules, we need to decide what should happen when an unwanted component is encountered. For example, maybe an email should go to the development manager, or perhaps we should prevent a build from completing successfully.

Ultimately, choosing the appropriate action is just as important as the rule itself. If you are too strict, development can stop, or be held up. However, if you are too lenient, potentially dangerous components could end up being part of your application.

The answer to this is found in two concepts, **enforcement points** and **communication**. We'll talk about those next.

2.2 Enforcement Points and Communication

When you hear enforcement point, you may get images of a militant checkpoint guarded on both sides, ready to stop anything from getting in or out. This tends to be a really negative, and sometimes warranted fear for developers. They are used to development being interrupted by a *Scan and Scold* mentality, where issues are found in a review process, and then the developers are scolded.

The reality of the modern and more effective software development team should be quite a bit softer. In addition it should be grounded on the principle of communication as your greatest tool, where Sonatype CLM provides a way to communicate the results of validation the rules that comprise your policy.

In Sonatype CLM, we consider an enforcement point simply a stage of development. This could be actual development done in an IDE, such as Eclipse, or perhaps a CI Server, such as Jenkins. In many cases, it is your repository manager Nexus Pro.

While the actions you take at each enforcement point can differ, remember that not every enforcement point needs to have a hard stop. For example, if you are using Sonatype CLM for IDE (Eclipse), this enforcement point can assist you in working towards improving component selection without halting

development.

This is done simply by giving a developer access to the rules you want to hold them to, identifying when components they are using don't meet those rules, and providing information for components that can be used instead. In this stage, it is communication heavy, and action light.

Now, given this it is possible for a development team to pass unwanted components off into the next stage of development. Luckily, we can save our harsher actions for further stages - enforcement points - in the development process. For example, we could fail a promotion of a release build that contains components not meeting our policies.

Ultimately, with Sonatype CLM, how you manage each enforcement point is up to you. Keep in mind though, your development team has the same goal as you. That is, producing great products that make your customer happy, and deliver highest quality and value. Don't let the software take the place of including them in the discussion and conveying your goals, and ultimately your policy, but rather define and create your policies with the development team. This integrating approach should be used for all other groups involved in your software development lifecycle, e.g. the operations team or the quality assurance team, as well.

2.3 Summary

For the most part, this section covers a general overview of Sonatype CLM. We touch briefly on high-level concepts like policy and enforcement points. It should leave you wondering what's next, which is expected and perfectly okay. The next section will begin to bring all these concepts together.

Chapter 3

What is a Policy?

When we talk about policy within the paradigm of Sonatype CLM, we refer to it as a way to identify and reduce risk through a concise set of rules for component usage. These rules can be used to assist at every step of the component and development lifecycle, and can be customized for specific applications and organizations. In general, policy, within the context of Sonatype CLM, is a broad term used to encapsulate:

- Conditions
- Actions

In some ways rules as a description is a bit generic, so let's dig a bit deeper, and look at another concept you are likely familiar with, an `If/Then` statement.

3.1 Basic Policy Anatomy

One of the easiest way to break down the various elements of a policy, at least the most basic parts, is to think of a policy as an `If/Then` statement. That is, a policy simply says that if something happens, then perform a certain action. If a component meets a set of criteria, then take a certain action, or in some cases no action at all.

If it's still a bit fuzzy, an example will probably help. Let's say we have a known rule in our development organization that says if a component used in an application has a security vulnerability, the application can not be released. To do this, we tell our development team to review components before release and if a component has a security issue, we don't promote the release. Congratulations, you have formed, at least in the aether, your first policy.

Now, let's take a slightly closer look, and define the basic policy anatomy. There are actually three key parts to a policy:

Conditions

conditions are the `if` part of the `if-then` statements.

Constraints

a constraint is really just a way to organize multiple conditions (if-then statements). Our example only had one so far. Let's say we decided we wanted to add that if a security issue is found and it has a CVSS of 2 or lower, only a warning should occur, but the release should not be prohibited.

Actions

actions are simply the `then` part of the if-then statement. Basically, what you want to have happen.

The above does a good job of telling us what makes up a policy in Sonatype CLM, but you are likely thinking, not all policies should be the same, I need a way to demonstrate which policies are the most important. We thought that too, and that is why in Sonatype CLM, all policies are assigned a threat level ranging from zero to ten (0-10). This score is completely subjective and will be unique in your organization.

OK, so now that we've opened up our concept of policy a bit, exposing the inner workings so to speak, the next question you should have is, "Where do we create policies?"

Well, as you likely recall, we can manage policy for both organizations and applications. We've learned a bit about these already, but let's go ahead and have a quick review.

3.2 Organizations, Applications and Inheritance

As a quick overview, the differences between an organization and application are as follows.

Organizations

- Require a name
- Provide an option to attach an icon
- Serve as a way to group applications.

Applications

- Require a name, application ID, and an organization.
- Provide an option to attach an icon.
- Represent a one-to-one relationship (App ID) between an actual application (or project), and Sonatype CLM.

Inheritance

Now, there is one final difference between organizations and applications, and that is inheritance, which is simply the ability for elements from an organization to transfer down to an application.

Ultimately this allows you to avoid micromanagement. For example, let's say there is a policy that says no component can have a security violation score greater than 3. This rule should apply to a number of applications that are all associated to a particular organization. Thanks to inheritance I can create a policy for the organization and all applications will inherit this policy. This also means I won't need to make changes to the policy for each application, rather I only make the change once, at the organization level, and it will affect any application attached to that organization. This policy will also apply to any additional applications I create under this organization in the future.

The important thing here, is to start thinking about policy more holistically. This is even before you begin to create policy in Sonatype CLM, you should think about where your policies will be created, and what applications will share similar policies. If nothing else, start writing out policies you would like to experiment with. Communicate those to your teams and start incorporating a proper feedback loop.

3.3 Summary

This section was all about policy. It's primarily theory, but theory that is quite important for your practical implementation. Sonatype CLM will build on everything discussed here. As a recap, here's what you should walk away with:

- Policy is an aggregation of rules that are basically If/Then statements - your policies
 - Each policy consists of one or many conditions
 - Multiple conditions together form a constraint
-

- When all conditions are met a policy results in the execution of an action
- Applications inherit policies from the organization they are associated to

Chapter 4

Organization and Application Management

You've likely heard that Sonatype CLM provides you with information about the components inside your applications. In addition to that information, you will see whether or not that component meets the rules for component usage that your organization has established - your policy. In order to provide that information however, there needs to be a link between your application and Sonatype CLM. But, how do we create that link, and where do we start?

Well, let's do a little introspection, and take a look at the idea of organizational structure.

4.1 Organizational Structure

When you launch Sonatype CLM for the first time, even after setting up and configuring your security parameters, there will be little to no information, a blank slate if you will.

Now, you could go off and simply start creating organizations and applications, as it's a fairly simple process. However, it would be wiser to think about how your particular business organizes applications. For many teams this follows a "command and control" structure, or rather one where various business units are responsible for specific applications. For others, applications create more logical categories, such as internal, or perhaps, commercial units, each having sets of applications below them.

In both cases these are units which simply contain applications, and there is some correlation between each application, even if it is only surface level. This idea of containers and correlation is the exact principle behind organizations.

Organizations, when looking simply at their most basic function, serve as a container for applications. While we cover how organizations manage policy and the other policy elements in just a moment, it's important to think about how you will set up your organizations before you begin creating them in Sonatype CLM.

Once you've done that though, you are ready to create an organization.

4.2 Creating an Organization

Organizations are created via the Sonatype CLM Server. Make sure you have proper access, which includes at least admin-level permissions (a member of the admin security group).

There are two main ways to create an organization:

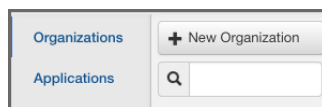


Figure 4.1: Using New Organization button



Figure 4.2: Using Global Create Button

The essential difference between the two options lies in Global Create button, which simply provides access to create an organization from anywhere within the Sonatype CLM Server.

Regardless of the option you choose, to create the organization, you only need to enter a name, and then click the *Save* button.

OPTIONAL ROBOT: As an option, you can add an icon for your organization, but this is not required. The image should be sized to 160 x 160 pixels and use the PNG format. Images with different sizes will be scaled. Alternatively you can press *Want a robot* to use a robot image. Each time you click on the link, a new robot image will appear.

4.3 Creating an Application

Earlier, we talked about a link between applications being developed, and the policy (or policies) that will be governing the components used in those applications. That link is provided by creating an application record within Sonatype CLM.

There are two main ways to create an application:

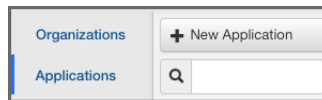


Figure 4.3: Using New Application button



Figure 4.4: Using Global Create Button

The essential difference between the two options lies in Global Create button, which simply provides access to create an application from anywhere within the Sonatype CLM Server.

Regardless of the option you choose, the information necessary to create an application is the same. Each application has three essential parts, which have been described below. Descriptions for optional items have been included as well.

Application Name (*required*)

This can be anything you want it to be, but it should be something people recognize. For example, Employee Intranet Application for Android, or International Bank Transfer Application. It is, quite simply, just a name, and it should be one that your users can identify with easily, as this is the name they will see in the various tools that connect to Sonatype CLM.

Application ID (*required*)

An Application ID, or App ID, is a unique identifier that you define for the application. In many ways, it's like a national identifier for the application. Most users will never see the application ID. However, it is used in a number of manual locations, including the various APIs that Sonatype CLM provides.

Organization (*required*)

Applications can share the same organization, and depending on the organization that you choose this determines a number of things such as which policies the application will be evaluated against. We'll discuss this more in the next section, for now, just treat this as a logical container helping to group your applications.

Note

Once an organization has been selected for an application, it can not be changed.

Contact (*optional*)

The contact is the person that is responsible for the application, or at the very least, should be contacted if there is an issue. It will be displayed in the reporting area of Sonatype CLM, as well as the PDF version of the report.

Icon (*optional*)

You can add an icon for your application, to help make it more easily identifiable. The image should be sized to 160 x 160 pixels and use the PNG format. Images with different sizes will be scaled. Alternatively you can press *Want a robot* to use a robot image. Each time you click on the link, a new robot image will appear.

4.4 Organization, Application, and Inheritance

So we understand the difference between organizations and applications, but how should this affect how we manage policy?

In fact the concept of policy may still be confusing. For now though, let's just think about it as a set of rules for components you can or can't use in your applications. Given this, each application is different, why not create policy (rules) for each individual application?

This is actually a common question when it comes to creating your policies for the first time. Our inclination tends to be to match a policy to an application. That is because we can sometimes think of applications as being very unique, and for that reason they will each have their own policies, different from other applications. This might be true. However, inheritance from an organization to an application plays a big role in making policy management much easier. Let's look at how the concept of inheritance works.

4.5 The Power of Inheritance

Because an organization can have multiple applications attached, and those applications inherit policies as well as labels, and license threat groups, creating a policy at the organization level allows us to manage a single policy across hundreds of applications. With this inheritance, you can make one modification and have that change affect all, or at least large numbers of, applications.

Now, let's put this in contrast with creating a policy at the application level, which seems similar, but the lack of inheritance from one application to another changes things up.

4.6 Avoiding Policy Micromanagement

In the case of organization level policy (rules), which appears across many different applications, the application level policy is meant for precise scenarios where the policy is only intended for a single application.

Doing this takes into account something specific we want to identify or keep out of a single application, but not others. The more of these application level policies that are created though, means the more

micromanagement, and in turn, opportunity for error, will occur. So, keeping them at a minimum, and only for those unique scenarios is ideal.

This is likely better conveyed in an example. Imagine two applications with 4 policies each. Two policies for these applications are identical. If you have this setup with two separate applications, any change to the identical policies has to be done once for each application. However if we move these policies to the organization that both applications belong to, we only have to change one. Now imagine a similar scenario with a larger number of shared policies as well as applications. Without organizations to inherit from this would become unmanageable quickly.

Alright, we understand organization level policies are a good idea, and application level policies should be used as minimally as possible. This tends to create a question of, "How do I know when to create an organization vs. application policy?" In reality the first limitation will actually be related to what you have access to create and/or view.

4.7 Permissions and Roles

Security administration can't really begin to take place until your first organization and application are created. This will need to be done by an administrator, as demonstrated in the examples above.

Once created, each organization and application has two available roles displayed in the *Security* section of the application and organization overview.

Owner

has full access to manage policy for the assigned organization or application.

Developer

has view only access for the assigned organization or application.

Note

Assigning a person to an organization role grants the same permissions to the applications within that organization.

Once you understand permissions, you can develop policies. Communicate with others to avoid micromanagement. If you don't have access to create an organization-level policy, it doesn't mean that it wouldn't benefit to create policy for applications within that organization.

4.8 Summary

OK, so in total, this first concept of organization vs. application is pretty simple. However, it's important to remember what you do here is setting up how you can manage policy later. If you followed our steps you should have done at least two things in this section:

- Created an organization
- Created an application

You should also have a clear understanding about inheritance and how organizations and applications differ. Finally, when you are first starting out, it is a good idea to experiment by creating a few one-off policies at the application level. Be diligent though, when you find yourself copying constraints and conditions into policies used by more than one application, it's likely time to consider simply adding the policy at the organization level.

Chapter 5

Policy Development

Sonatype CLM uses the term **policy** to broadly refer to the set of policies and policy elements (e.g. labels and license threat groups) used to ensure components in an application meet a specific set of standard. In the past, we colloquially compared these to *rules*.

The process of creating this set of rules based on specific factors is considered policy development. Combining this with the ongoing refinement and adjustment is the broader category of policy management. No matter what it is called though, the end result should always be actionable results that are representative of your organizations risk tolerance. Put a bit more simply, Sonatype CLM policy provides a means to organize risk data.

Before we expand on risk, let's dig a little deeper, and really take a look at what we mean when we talk about policy, expanding everything that goes into its development.

5.1 Advanced Anatomy of a Policy

If you have taken a look at any of Sonatype CLM documentation or poked around the imported policies in the Sonatype CLM server user interface, you may have already seen what we refer to as basic anatomy of a policy - all the pieces that go together and define your policy.

Security-High

9

Application Matching

Which applications should this policy apply to?

☒ All Applications in My Organization
☐ Applications with one or more of these tags None selected

Constraints

CVSS >=7 and <10

Security Vulnerability Severity

>=

7

-

+

Security Vulnerability Severity

<

10

-

+

Security Vulnerability Status

is not

Not Applicable

-

+

All

of the above conditions will trigger this constraint.

Actions

	Enforcement Points		
Stage	Warn	Fail	Notifications
Develop			✉
Build			✉
Stage Release			✉
Release		!	✉
Operate			✉

Monitoring Notifications

If you have enabled monitoring, who should be notified?

☒ MySecurityTeam@mycompany.com

Cancel

Save

Figure 5.1: Editing a Policy and its Attributes

So, branching beyond the simple concept of *If/Then* statements, let's break policy down into each part you can interact with keeping an eye on the editing screen for a policy displayed in Figure 5.1.

Policy Name

This name will be displayed on the *Policy* tab in the application composition report. Others will see

this regularly, so it should be unique, clear, and concise.

Threat Level

A number, 10 - 0, that is color coded (red, orange, yellow, dark blue, and light blue), and represents the perceived severity if this policy is violated. The number will also be used to create the order in which policy violations are displayed.

Constraints

Each policy must have at least one constraint. When a constraint is fulfilled, a policy violation occurs. A constraint itself consists of the **Constraint Name** and at least one condition. Make sure it clearly identifies the conditions that you have added for the Constraint.

Conditions

A condition is considered the *if* part of an if-then statement. There are a wide range of conditions possible, that have their own set of values you can choose from.

Any/All

Any/all is required when there are multiple conditions. This tells the policy whether all of the listed conditions, or simply any of them, must be met in order to have a policy violation.

Actions

The *then* part of an if-then statement. The action chosen here will be taken when the policy constraint and its associated conditions have been met.

Stage

These stages represent the enforcement point. They are *Develop*, *Build*, *Stage Release*, *Release* and *Operate*.

Warn and Fail

Each enforcement point for each stage can be configured to cause a 'Warn'ing or a 'Fail'ure.

Custom

This action allows you to select an email, or emails, to send notification to when new policy violations have occurred.

Now that we understand the different attributes of an individual policy, you're likely eager to create your own policy. We'll tackle that next in Chapter 6, but first let's take a look at some important items to consider when developing our policy.

Note

When viewing policy violations counts, please keep in mind that despite the number of constraints fulfilled, only one policy violation is counted.

5.2 Risk and Organizational Intent

To establish an understanding of risk within the context of CLM, we need to identify the various avenues of risk a component can have. The most common are security, licensing and architectural considerations.

Of course, we shouldn't limit our thinking to these three alone, and in the long term you will define those specific to your organization. However, they will serve us well to get started.

When creating CLM policies, we need to consider what is the risk we want reported and how we want it reported. Take a look at these very simple example policies, one each for licensing, security, and architecture.

Licensing

- Don't allow distributed code to have GPL
- Only allow GPL that has a Commercial license

Security

- Don't allow components with a CVSS score > 7

Architecture

- Don't allow components that are older than 5 years
- Don't allow Struts version 2.3.15.1

The above policies represent an organizational intent to not allow GPL highly insecure components. It further qualifies that old components and a specific version of struts are not to be used. In this way, our three policies are actually working together to form an overall policy approach.

As you move on to create your own policies, it becomes very important to think about how policies can build upon each other. In many ways this is a holistic approach, something that rules simply can't do.

5.3 Summary

Policy development should be treated to be as important as any other aspect of Sonatype CLM. While it is more of the planning and thinking stage of policy, it can make the difference between well thought out and

applicable policies and policies that require constant change, or seem to disrupt rather than compliment the development life cycle. Soon you'll be creating policies on your own. For now though, be sure you take these key items away in this section:

- Importance of Policy Development
- Risk and Organizational Intent

Chapter 6

Policy Creation

So you are ready to create your first policy. Great! If you have reached this point, it is important to be sure you have everything in place before you begin creating your own custom policies:

- Organizations and applications are set up as desired in the Sonatype CLM server.
- Users have been assigned to these organizations and applications with the proper roles to fulfill your security requirements.
- You have determined the risks that apply to your organization and/or applications.

If you haven't done so, it's also a good idea to write out the policies you want to create. While you will find the actual creation process is pretty easy, the more thought you put into your policies and their structure the better they will be.

Also, don't forget, that once created, a policy needs to be adjusted and modified over time. We refer to that more largely as Policy Management.

6.1 Getting Started

We've decided to break policy creation into six total steps. Each step deals with a specific area of a policy and will be described in detail. Before you go to the first step, you should know there are two key ways to create a policy.

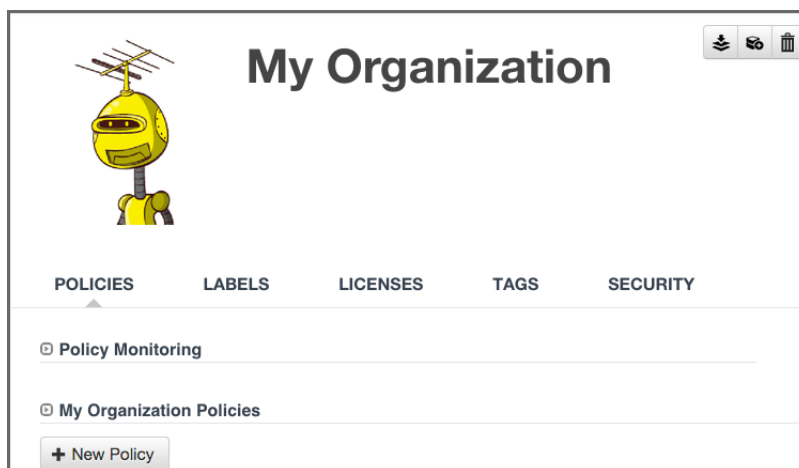


Figure 6.1: Using New Policy Button



Figure 6.2: Using Global Create Button

Note

Our instructions follow the process for creating a policy for an organization. This is where most of your policies will be created. However, if you need to create an application-specific policy, just substitute application for organization.

There is really no difference here, as both require that you have the organization or application open at the time of creation. The one advantage with using the Global Create button is that you can create no matter which tab of the currently selected organization or application you are in.

6.2 Step 1: Understand the Policy Intent

So, let's build on the third item we mentioned in the introduction, and first think about the intent of the policy we are going to create. Here are some questions to consider:


- Is this policy for all applications, or should it only be matched against certain ones?
- What rules do we want to create, or in other words, what things do we not want in our applications, or this specific set of applications?
- How do we want to react when violations occur?

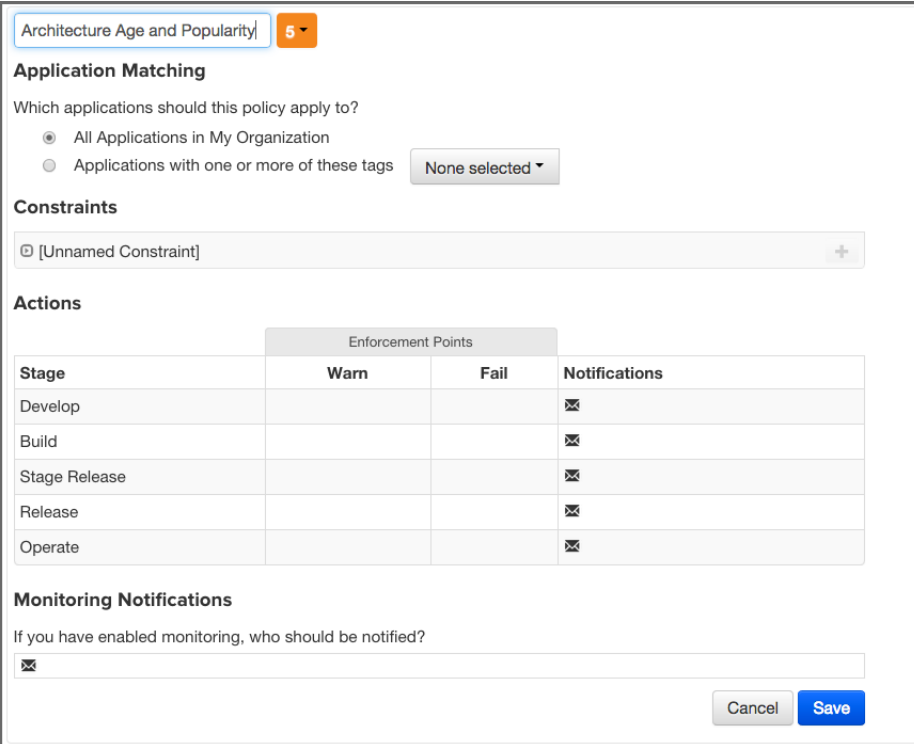
These questions are really just the start, but in short you should ask what applications, what causes a violation, and what action should be taken. For our example policy, we want to focus on architecture aspects, with key areas being *Age* and *Popularity* of our components.

6.3 Step 2: Decide on a Descriptive Policy Name

Since architecture is our focus we are going to use it as part of the name for our policy. In addition we are going to add *Age* and *Popularity* to the name to be more specific. This will help us later when we start creating additional architecture related policies. Lets go ahead and create this policy:

1. Log into the Sonatype CLM Server (*by default this is available at <http://localhost:8070>*) using a user account with Owner-level permissions for the organization (a member of the Organization's Owner Group).
-

2. Next, click the *Organizational Design* icon  to access the **Organizational Design** area.
3. In the menu on the left, click on *Organizations*, and then click the organization you want to add a policy to.
4. To create a new policy, click the *New Policy* button. You should now see the policy editing screen (as displayed in Figure 6.3).
5. The policy name input field should have focus and contain *Enter Policy Name*.
6. Enter the new policy name by typing *Architecture Age and Popularity*



Architecture Age and Popularity 5

Application Matching

Which applications should this policy apply to?

☒ All Applications in My Organization
☐ Applications with one or more of these tags None selected

Constraints

+ [Unnamed Constraint]

Actions

Stage	Enforcement Points		Notifications
	Warn	Fail	
Develop			<input checked="" type="checkbox"/>
Build			<input checked="" type="checkbox"/>
Stage Release			<input checked="" type="checkbox"/>
Release			<input checked="" type="checkbox"/>
Operate			<input checked="" type="checkbox"/>

Monitoring Notifications

If you have enabled monitoring, who should be notified?

☒

Cancel Save

Figure 6.3: Naming the Policy

6.4 Step 3: Choose an Appropriate Threat Level

Threat level is one of the easiest concepts we'll cover in this guide, yet it has the greatest chance to cause huge problems. At its core, this is simply a value for severity your business associates with a given policy. These values range from 1 to 10, each corresponding to range of severity, and a specific color:

Table 6.1: Threat Levels

High	Red	8-10
Medium	Orange	4-7
Low	Yellow	2-3
Informational	Dark Blue	1
None	Light Blue	0

To assign threat level to a policy, you will need to be in the policy edit mode (click on the editing icon). Next, use the drop down, which is located next to policy screen. This is displayed in Figure 6.4.

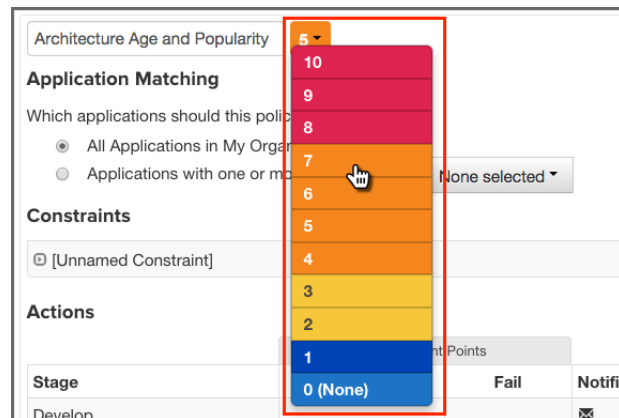


Figure 6.4: Editing the Policy Threat Level

When a threat level is selected by any component that has violated a policy will be displayed in the order of the severity of the violation. This is straight forward, but let's take a look at the details and see where some issues can arise.

Just like taste is subjective, you have to realize that severity is subjective. What may be a level 10 for one part of the business may be quite different for another.

A bigger concern is what message you want to communicate to the recipients of the analysis. For example members of your team will naturally be inclined, to treat high severity as very important, and critical. If they see too many of these, the prospect of ever making any headway could be impacted in a negative way and you might dismay e.g. your developers to even get started on trying to fix the violations since there are just too many. As with any control mechanism it is probably advisable to be a bit lenient at first and start to get stricter at a later stage, when the worst problems are dealt with. Otherwise you could cause undue stress to development and management teams. The important thing is to keep in mind is how you will assign severity, and the impact it has on those reviewing violations.

Especially in the beginning, our recommendation is to reduce, if not exclude the high severity threat levels altogether. Remember, at the core, this number is purely for ordering how a violation will be displayed to a user of Sonatype CLM. Because this needs to be interpreted, the actual value is very subjective. It's easy to overwhelm members of your team by seeing large numbers of highly severe violations. It might even be reasonable to set severity to the lowest setting for a violation (Low - 1) in the beginning. This way, members of your teams can get used to seeing the results that are produced.

In our example policy, architecture quality is pretty important, but it's not the most important issue, so we'll leave the default threat level of 5. This will give us a violation higher up in the display, but it's certainly not the worst.

Tip

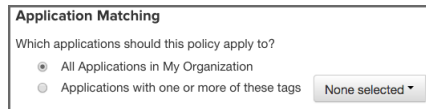
Remember, threat is subjective, and is most useful for ordering violations.

6.5 Step 4: Choose the Application Matching Parameters

In many cases, policies will be evaluated against all applications. However, as your applications become more diverse and there are various groups of applications with similar characteristics, you may find the need to develop policies for these specific cases. This will go hand-in-hand with the [creation of tags](#), which application owners will apply to their applications.

Given the ability to create tags, there is an endless amount of refinement that can be done in order to focus a particular policy on a set of applications with similar characteristics. While we won't get into the creation of tags here, thinking about this as a way to create more finely tuned policy is a good first step. For now, we'll assume we have some pretty generalized architecture requirements for all applications,

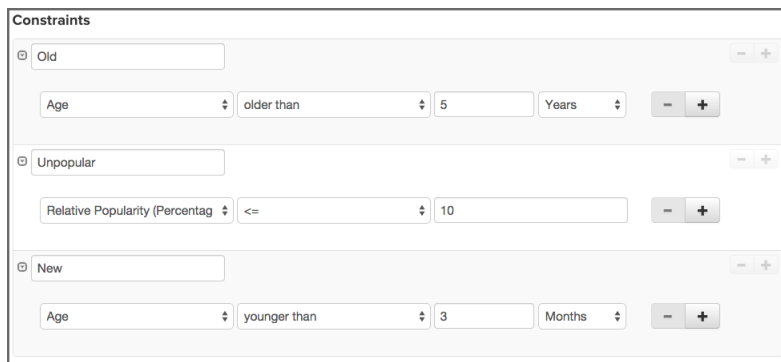
and want to make a policy that will match all applications in this organization.



A dialog box titled "Application Matching" with the question "Which applications should this policy apply to?". It contains two radio button options: "All Applications in My Organization" (which is selected) and "Applications with one or more of these tags". To the right of the second option is a dropdown menu currently showing "None selected".

6.6 Step 5: Create Constraints with Conditions

By now, we should know that an easy way to look at conditions is to consider them the *if* part of an *if/then* statement. In contrast, a constraint is not part of the statement, but rather a container for conditions. This can come in real handy when you want to consolidate a policy, by grouping similar conditions.



A panel titled "Constraints" containing three expandable sections, each with a minus/plus icon on the right. The first section, "Old", contains a condition: "Age" (dropdown), "older than" (operator), "5" (value), and "Years" (unit), with minus/plus icons. The second section, "Unpopular", contains a condition: "Relative Popularity (Percentag" (dropdown), "<=" (operator), "10" (value), and empty unit, with minus/plus icons. The third section, "New", contains a condition: "Age" (dropdown), "younger than" (operator), "3" (value), and "Months" (unit), with minus/plus icons.

Figure 6.5: Example Constraint

But why do this?

Well, for one thing, it reduces time. Next, because constraints are displayed on the application composition report, along with violations, it becomes easier for a member of your team to process information that is similar. Of course, it also has the added benefit of not having to create hundreds of small, one-off policies.

For example, in our *Architecture Age and Quality* policy, we may start with two conditions, one for age and one for popularity. This will work, but it's likely better, especially if we want to expand that later on, we isolate these as separate constraints.

As a good practice, grouping similar conditions is the best way to build your constraints. Again, this will help de-clutter a multi-policy environment, as well as help someone reading any associated violations on the application composition report. For our example, we'll be placing both conditions in the same constraint.

1. Click on the *Expand/Collapse* icon (shaped like a right-facing triangle). It's next to the *Constraint Name* and should display *Unnamed Constraint*.
2. Once the constraint is expanded, click the *Constraint Name* field and enter *Age and Popularity* as the constraint name where the field initially displays *Enter Constraint Name*.
3. Using the drop downs and the add (+) button, add two conditions.
 - a. *Age* should be checked to be *older than* the value of 3 Years.
 - b. *Relative Popularity (Percentage)* should be set to be greater than or equal to (\geq) 50 percent (50%).
4. Select *All* in the drop down below the conditions as the aggregating rule so that all of the above conditions have to be fulfilled to trigger a policy violation for the constraint.

The screenshot shows a 'Constraints' dialog box. At the top, there's a text field with 'ex. Unpopular' and a right-facing triangle icon. Below this, there are two rows of conditions. The first row has a dropdown for 'Age', a dropdown for 'older than', a text field for '3', a dropdown for 'Years', and a button with '-' and '+'. The second row has a dropdown for 'Relative Popularity (Percentage)', a dropdown for '>=', a text field for '50%', and a button with '-' and '+'. At the bottom, there's a dropdown for 'Any or All' and a text field that says 'of the above conditions will trigger this constraint.'

Figure 6.6: Adding Constraints

Now, before we move on, let's take a look at some of the specific aspects of conditions that can tend to be a bit confusing.

Policy Type Sonatype CLM has an algorithm for determining the type of policy you created. This type is based on the types of conditions you include and is mainly used in reporting purposes. For example the trending report aggregate the different policy types into separate lists. The rules to determine the type of a policy are:

- If there are any security conditions, it is considered a security type policy.
- If there are any license conditions, it is considered a license type policy.

- If there are any age or popularity conditions, it is considered a quality type policy.
- If there are any conditions not mentioned above, it is considered an other type policy.

Any vs. All If a component meets any constraint, the policy is considered violated. However, inside each constraint, when you have more than one condition, you have the ability to require that a component must either meet all conditions or just any of the conditions to trigger a violation.

This is a straightforward concept, but it can produce vastly different results. For example, selecting *Any* as an option will tend to produce a lot more violations, it's the equivalent of placing an *or* between each condition. In addition even if you resolve one violation for a component, if the component meets other conditions later on, it might cause it to still violate the policy. On the other hand, using *All* allows you to establish that all conditions must be met. It's like placing an *and* between each condition. In this case, if you address any of the conditions, the violation will be resolved.

Let's look at an example that would further explain this, and then we will move on to creating conditions in our policy. Below, we have a couple of policies and a component. If we wanted to exclude components that were younger than four years and had less than fifty percent popularity, which would we want to use? Also, would the component listed violate this policy?

Policy 1 with Conditions

- Age is < 4 years
- Relative Popularity is \leq than 50%
- **Any** of the above conditions trigger the constraint.

Policy 2 with Conditions

- Age is < 4 years
- Relative Popularity is \leq than 50%
- **All** of the above conditions trigger the constraint.

Component Data

- Age is 3 years
- Relative Popularity is 85%

It may seem like these two policies are exactly the same, but there is a key difference, *Any* vs. *All*. So, our first policy states that if *Any* of the conditions are met that a violation will occur. Given that, our component violates this policy. This is good, but we only want a component to violate a policy when Age is less than three years, and popularity is below 50%. We understand that there may be circumstances

where a component might need to fall outside one of those boundaries, but if it's both, we know we have a problem. Our component is pretty popular, so it might be a case where this component, even though it is newer than we prefer, actually resolves an issue another component might have. For this reason, the best policy for us to choose is Policy 2, and our component would not actually cause a violation.

Available Conditions

The following list enumerates the available conditions and describes the operator and the value used for the evaluation.

Age

Verify if the components age based on the date it was placed in the Central Repository in months, days or years is older or younger than a specified value.

Coordinates

Verify if the coordinates for a component *match* or *do not match* (e.g. *groupId:artifactId:version*).

The coordinates may be fixed, or a wildcard can be used. For example, *org.sonatype.com:nexus-indexer:1.**. Here, components with a groupId of *org.sonatype.com*, an artifactId of *nexus-indexer*, and any version starting with *1.* would be matched.

Note

This condition is only applicable for Maven components.

Additional Examples

- A fixed coordinate: *org.sonatype.nexus:nexus-indexer:1.0*
- A wildcard coordinate to an inclusive group and version: *org.sonatype*:nexus-indexer:1.**
 - this condition would match, for example *org.sonatype.test* or *org.sonatype.example*, an artifact named *nexus-indexer* in these groups, and any version that starts with *1.* for those groups and that specific artifact.

Note

The use of the wildcard is limited to the end of each parameter in the coordinate.

Identification Source

Verify if the component identification *is* or *is not* based on data from *Sonatype* or your *Manual* identification

Label

Verify if a label with a specific label name *is* or *is not* attached to a component

License

Verify if the component license *is* or *is not* a specified license. If the component license has been overridden, anything listed as declared or observed will be ignored. If license status is not overridden, then any occurrence of the selected license (observed or declared) will be considered a violation.

License Threat Group

Verify if the components license *is* or *is not* in one of the groups *Banned*, *Copyleft*, *Liberal*, *Non Standard*, *Not Observed* or *Weak Copyleft*. If the component license has been overridden anything listed as declared or observed will be ignored. If license status is not overridden, then any occurrence of the selected license (observed or declared) in the selected License Threat Group, will be considered a violation.

License Threat Group Level

Verify if the components license threat in its license threat group is either lesser <, lesser or equal <=, greater > or greater or equal >= a specified value. Value range from zero for no threat to the highest threat level value of ten.

License Status

Verify if the user defined License Status *is* or *is not* one of the possible values of *Open*, *Acknowledged*, *Not Applicable* or *Confirmed*.

Match State

Verify if the match of the comparison of the component to known components from the Central Repository with the same coordinates *is* or *is not exact*, *similar* or *unknown*.

Proprietary

Verify if a component *is* or *is not* considered proprietary.

Relative Popularity (Percentage)

Verify if a the relative popularity of the component version as compared to other versions of the same component group and artifact parameter is either lesser <, lesser or equal <=, greater > or greater or equal >= a specified percent value.

Security Vulnerability

Verify if a security vulnerability is *present* or *absent*. Keep in mind that just because there is no known, or *present*, security issue, doesn't necessarily mean none exists.

Security Vulnerability Severity

Verify if a security vulnerability with a numeric severity is either lesser <, lesser or equal <=, greater > or greater or equal >= a specified value.

Security Vulnerability Status

Verify if a the components security vulnerability status *is* or *is not* one of the possible values of *Open*, *Acknowledged*, *Not Applicable* or *Confirmed*.

Now that you know about all the available conditions and how they can be combined to match all or any condition to create a constraint we are ready to determine what should happen if a constraint is met - the policy actions.

6.7 Step 6: Set Policy Actions

In order to understand Policy Actions, we first need to discuss enforcement points, something we mention in our other guides, as well as much of the material you have likely used to make your decision to move forward with Sonatype CLM.

Actions			
Stage	Enforcement Points		Notifications
	Warn	Fail	
Develop			✉
Build	⚠		✉
Stage Release	⚠		✉
Release		❌	✉
Operate			✉ ArchitectureLead@MyCompany.com

Figure 6.7: Policy Actions Example

An enforcement point is simply a stage in Component Lifecycle Management as well as a parallel point in the Software Development Lifecycle. Given a particular stage, we can evaluate components within an application, making sure that they meet our policies (rules) that have been designed to manage component consumption. For example, we can show developers when a component they are using is too old to meet our policy on component age.

The actions we take at these enforcement points (stages) can simply be passive; in other words, only displaying violation of our policy to us and/or notifying specific people when a violation occurs.

In contrast, they can also be active, such as preventing a staged build from moving into production. Of course the caveat here, is you will need the proper CLM product (or products) to match the supported enforcement point. Additionally, we need to be careful, not to disrupt the process of development, by too quickly exposing large numbers of high-threat violations.

For our example, we will add a notification action for our build stage.

1. On the row for the **Build** stage, click on the *Envelope* icon in the **Notifications** column
2. The **Notifications** dialog will display and allow you to provide email addresses that should be notified in the event of a policy violation

Actions			
	Enforcement Points		
Stage	Warn	Fail	Notifications
Develop			<input checked="" type="checkbox"/>
Build			<input checked="" type="checkbox"/> ArchitectureLead@MyCompany.com
Stage Release			<input checked="" type="checkbox"/>
Release			<input checked="" type="checkbox"/>
Operate			<input checked="" type="checkbox"/>

Figure 6.8: Setting Policy Actions

Tip

Notifications will only display new violations found in the latest scan. If you find yourself not receiving notification, verify there are new violations, as well as confirm you have configured your Sonatype CLM server SMTP settings.

With that out of the way, let's also take a look at the supported enforcement points, or stages of the development cycle, as well as how actions are best utilized in each one.

Develop - Integrated Development Environments (Eclipse)

In the first stage, **Develop**, the approach should be limited towards providing information, while at the same time ensuring development can continue. However, a developer can certainly see when components that don't meet the policies of the business are being used, and make adjustments accordingly.

Note

Actions are currently selectable in this stage. However, they have no effect regardless of the tool used.

Build - Continuous Integration Servers /Ad Hoc Scanning (Hudson/Jenkins)

Next, the **Build** stage has options for both preventing a build, as well as simply warning. As you manage policy, making necessary adjustments over time, it's best to take an approach that allows for your development teams to be eased into dealing with violations. For this reason, it's better to start by simply warning when the build for an application contains components that violate your policies.

Stage Release

Stage Release is perhaps the most important stage to consider. The concept of a staged release involves placing a near final build into a staging repository prior to having it officially released. Because of this, it gives the opportunity to prevent an application from being released with components that have violated policy. Thus, setting the action for a Stage Release to Fail, is our recommendation.

Release

Release is the final push for a project into production. While there should be the closest scrutiny of policy violations at this point, there will be a similar recommendation to fail a release based on severe violations. In most cases, you should ideally be finding new violations only.

Tip

If you have setup policy monitoring, it is a good idea to monitor your release stage, as this is likely the best representation of your production application.

Operate

The operate stage can be set via a variety of tools, but in all of these cases, it is set manually. For this reason, providing any warning or fail actions will not produce any different result.

Now, given the above, the natural thing to do seems to fail the process from moving forward any time problems are found. However, that's actually not the best approach for everyone, and it's not one we recommend. Instead, only consider failing the Stage Release as we've recommended, and even then, only consider doing so for the most severe violations.

While it is possible to fail the *Build* stage, this might put an unwanted roadblock in the way of your development process. Even without failing a build, your team can always be aware of the violations and trouble spots, but at the same time, still progress forward. A good alternative to build failure, is instead adding a notification to a stage so that when a policy violation is encountered. This way, responsible parties are aware without needing to routinely review a report, or have their own process stopped by an interruption event, like failing a build.

6.8 Summary

Congratulations! You should now have created a custom policy that checks for a specific age and popularity of all the components. If you've been using the stand alone scanner, or if you are already using other

enforcement points, take a look at how your applications react to this new policy. Here are the highlights you should know about:

- Create a Custom Policy
- Understand Advantages of Multiple Constraints
- Best Practices for Actions and Stages

Chapter 7

Policy Elements

One of the more interesting pieces of Sonatype CLM Policy, and something we've only alluded to in a number of our guides, is policy elements. At this time there two specific policy elements we can work with:

- Labels
- License Threat Groups
- Tags

These elements are unique in that they have both links to policy as well as general Sonatype CLM functionality. Let's take a closer look at each one of these.

7.1 What is a Label?

We just learned all about creating policies. Constraints, conditions and even actions, all make a lot of sense, but what in the world could labels have to do with policy?

Well, labels are actually one of the more powerful features of Sonatype CLM. They should have a familiar look, since you've likely used other systems that employ a sort of tagging or labeling.

Labels are metadata. More specifically a label is metadata that is assigned to a component within the context of a particular application or organization. Labels can assist with identifying components you want to review, approve, or even avoid altogether. We call this label assignment.

When labels are assigned, this is an action that takes place in the application composition report. Before it can be assigned though, a label needs to exist for a particular organization or application.

As we learned in our section regarding Organizations vs. Applications, inheritance plays a big role in policy. The same thing is true for labels, in that if a label is created in an organization, any application attached to that organization will also have the label available for use when assigned. In fact, the system will prompt you to choose the scope (organization or application) a label should exist in when it is assigned.

7.2 Creating, Editing and Deleting a Label

We've determined that assigning a label is an important action, but how do we build policy around this? That's actually simple, we just add a condition, based on a specific label that you have created, being present. The one caveat, is that the label needs to exist within the application or the organization in which you are creating the condition.

There are two key ways to create a label:

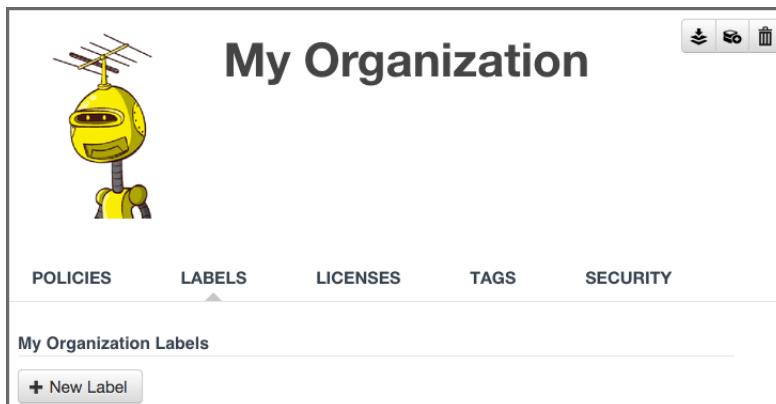


Figure 7.1: Using New Label Button



Figure 7.2: Using Global Create Button

There is really no difference here, as both require that you have the organization or application open at the time of creation. The one advantage with using the Global Create button is that you can create no matter which tab of the currently selected organization or application you are in, whereas you will need to be on the Label tab otherwise.

When creating your label, remember to use something that is easily identifiable. If you're following along with our example in the next section, *Architecture-Approved* is a good example.

A screenshot of the 'Label Name' dialog box. The dialog has a title bar 'Label Name'. Below the title bar is a text input field containing 'Architecture-Approved'. Below the input field is a section titled 'Select Color' with a grid of nine color swatches, each labeled 'Architecture-Approved'. Below the color swatches is a section titled 'Description' with a text input field containing 'Component is approved by the Software Architecture Lead.' At the bottom of the dialog are two buttons: 'Cancel' and 'Save'.

Figure 7.3: Label Example

Once you have entered the name, you can enter an optional description, and then click the *Save* button.

Editing Labels

To make changes to a label, click on the label and the label information will be displayed below.

Deleting Labels

To delete a label, just click on the X next to the label name.

A few things to remember:

- An organization's labels can be seen by any of its applications, the reverse is not true.
- Labels can only be edited (or deleted) at the level they were created.

7.3 Creating a Condition Based on a Label

In the example below a new condition for the label *Architecture-Approved* will be added to an existing policy with an existing constraint and condition.

In our instructions, we've made an assumption that you understand how to [create a policy](#).



The screenshot shows a 'Constraints' section in a web interface. It contains a list of constraints, each with a dropdown menu for the constraint type, a dropdown for the operator, a text input for the value, and a dropdown for the unit. To the right of each constraint are minus and plus buttons. A new constraint is being added, indicated by a plus button in the top right corner. The new constraint has a dropdown menu for the constraint type, a dropdown for the operator, a dropdown for the value, and a dropdown for the unit. The new constraint is 'All' of the above conditions will trigger this constraint.

Figure 7.4: Creating a Label Condition

1. Open an existing policy.
2. In the *Constraints* area of the policy, click on the + icon, located next to the right of an existing condition.

Tip

Make sure you use the correct + icon, as it can be easy to add a new constraint by mistake.

3. Now, in the *Conditions* area, change *Label* in the first drop down menu from *Age* to 'Label'.
4. Next, in the second drop down menu select *is not* for the operator.
5. Finally, in the third drop down menu, select the *Architecture-Approved* label you just created.
6. Click the *Save* button to finish.

Tip

Because our example uses a constraint with an existing condition, we have also chosen to force a violation only when all conditions have been met. In this scenario it may be appropriate to **consider a waiver as an alternative**.

7.4 What is a License Threat Group?

License threat groups, are simply groups of licenses, broken into categories of severity for the various types of licenses. They can help you to achieve your goals related to enforcing the usage of components with licensing that matches the scope of your application.

Their primary purpose is to serve as the data points for the License section of the application composition report. Moreover, they are a way to group risk, associated with licensing. By default, there are four license threat groups included with Sonatype CLM:

Copyleft

Strong copyleft licenses go a step further from weak copyleft licenses and mandate that any distributed software that links or otherwise incorporates such code be licensed under compatible licenses, which are a subset of the available open-source licenses. As a result, these licenses have been called viral.

Non Standard

Something out of the ordinary (e.g. If we ever meet, give me a beer license).

Weak Copyleft

Free software licenses that mandate that source code that descended from software licensed under them, will remain under the same, weak copyleft, license. However, one can link to weak copyleft code from code under a different license (including non-open-source code), or otherwise incorporate it in a larger software. Otherwise, weak copyleft licenses allow free distribution, use, selling copies of the code or the binaries (as long as the binaries are accompanied by the (unobfuscated) source code), etc.

Liberal

These licenses allow you to do almost anything conceivable with the program and its source code, including distributing them, selling them, using the resultant software for any purpose, incorporating into other software, or even converting copies to different licenses, including that of non-free (so-called “proprietary”) software.

Note

Consult with your legal department for EXACT definitions. Information provided above is from the following [reference](#).

7.5 Creating, Editing, and Deleting a License Threat Group

An important aspect of license threat groups is that each one also has a threat level, just like policy (from zero signifying no threat all the way up to 10). Unless you have specific legal recommendation / council, the default license threat groups will suffice, especially in the beginning.

If you desire, you can edit these default groups, or create entirely new ones. When creating license threat groups, keep in mind that they will be inherited from the organization to all associated applications.

There are two key ways to create a license threat group:

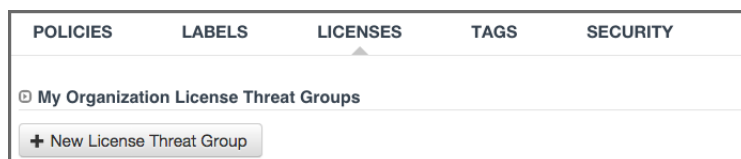


Figure 7.5: Using New License Threat Group Button



Figure 7.6: Using Global Create Button

There is really no difference here, as both require that you have the organization or application open at the time of creation. The one advantage with using the Global Create button is that you can create no matter which tab of the currently selected organization or application you are in, whereas you will need to be on the License tab otherwise.

The following information needs to be completed before a license threat group can be saved.

Name

This is the name for your license threat group. When creating or editing the name of a license threat group, remember to use something that is easily identifiable. If you're following along with our example in the next section, use *Banned Licenses*.

Threat Level

This is the level of threat this group of licenses should represent.

Applied and Available Licenses

Adding licenses to the license threat group is not an actual requirement, but there really isn't much use for simply creating a group as a placeholder. So this is treated as a required field.

On the left are licenses that are included in the license threat group. Click on a license to remove it.

+ On the right are the licenses that can be added the group. Click on a license to add it.

When everything is done your screen should look like Figure 7.4 and you can click the *Save* button to finish.

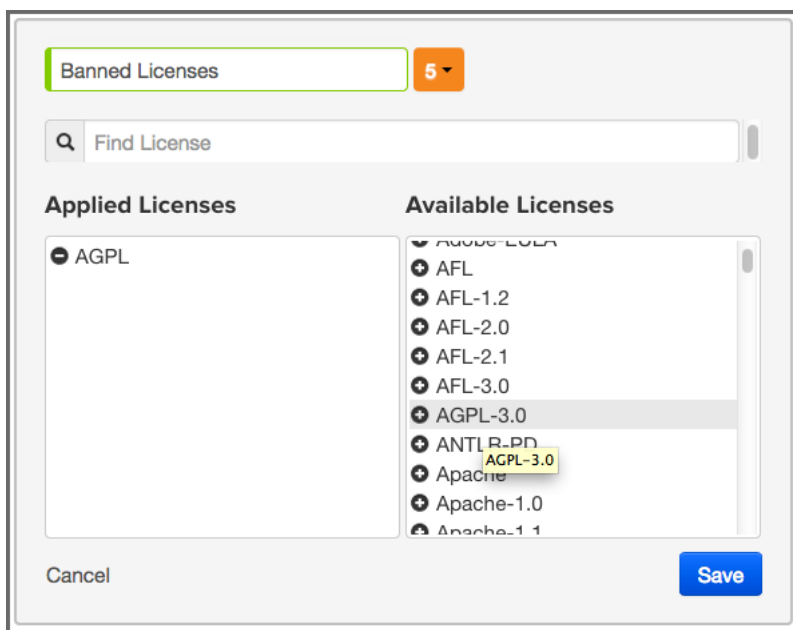


Figure 7.7: Creating a License Threat Group

Editing

To make changes to a license threat group, click on the *Edit* icon (shaped like a pencil).

Deleting

To delete a license threat group, just click on the *Delete* icon (shaped like a trash can) next to the label name.

A few things to remember:

- A set of four default license threat groups are provided.
- Applications inherit license threat groups from their organization.
- An organization's license threat groups can be seen by any of its applications, the reverse is not true.
- License threat groups can only be edited (or deleted) at the level they were created.

7.6 Creating a Condition Based on a License Threat Group

In the example below a new condition for the license threat group, *Banned Licenses*, will be added to a new policy.

In our instructions, we've made an assumption that you understand how to [create a policy](#).

1. Create a new policy.
2. In the *Constraints* area click on the *Expand/Collapse* icon (shaped like a right-facing triangle). It's next to the *Constraint Name* and should display *Unnamed Constraint*.
3. Once the constraint is expanded, click the *Constraint Name* field and enter *Banned License*.
4. Now, in the *Conditions* area, change *Age* in the first drop down menu to *License Threat Group*.
5. Next, in the second drop down menu choose *is* for the operator.
6. Finally, in the third drop down menu, find and select the *Banned License* label you just created.
7. Click the *Save* button to finish.

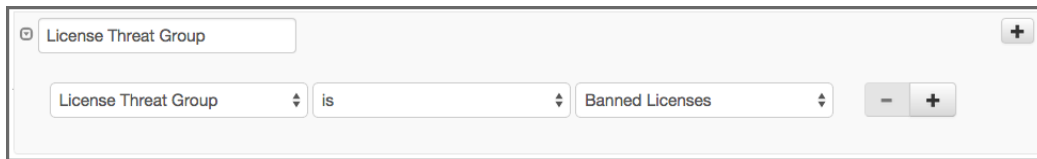


Figure 7.8: Creating a Condition Evaluating a License Threat Group

7.7 Creating a Condition Based on an Unassigned License Threat Group

In most cases, a license is associated with one or more [License Threat Groups](#). However, it is possible for a license to have no association with any [License Threat Group](#). You can create a Policy to detect when a component has a license that is not assigned to any License Threat Group.

In the example below a new condition for detecting components with licenses not assigned to any License Threat Group will be added to a new policy.

In our instructions, we've made an assumption that you understand how to [create a policy](#).

1. Create a new policy.
2. In the *Constraints* area click on the *Expand/Collapse* icon (shaped like a right-facing triangle). It's next to the *Constraint Name* and should display *Unnamed Constraint*.
3. Once the constraint is expanded, click the *Constraint Name* field and enter *Unassigned LTG*.
4. Now, in the *Conditions* area, change *Age* in the first drop down menu to *License Threat Group*.
5. Next, in the second drop down menu choose *is* for the operator.
6. Finally, in the third drop down menu, find and select *[unassigned]*.
7. Click the *Save* button to finish.

The screenshot shows a web-based interface for creating a constraint. At the top, there is a text input field containing 'Unassigned LTG' and a small right-facing triangle icon to its right. Below this, there is a row of three dropdown menus. The first dropdown menu is labeled 'License Threat Group', the second is labeled 'is', and the third is labeled '[unassigned]'. To the right of these dropdowns are two buttons: a minus sign '-' and a plus sign '+'. The entire interface is enclosed in a light gray border.

Figure 7.9: Creating a Condition Evaluating an unassigned License Threat Group

A violation of the policy above can be remediated simply by assigning the license involved to a [License Threat Group](#).

To remediate a specific component, use the Component Information Panel (CIP) *License* tab to set the license *Status* to *Selected* or *Overridden*, and then select a license that is associated with at least one [License Threat Group](#). Managing component licenses is discussed further in the [Editing License Status and Information](#) section.

7.8 What is a Tag?

In any given business, you could have hundreds, maybe even thousands of applications. Even if you are just getting started, it's likely you have a handful of applications. However, as unique as applications can be, they tend to share some similarities.

For example, you might have applications that process or store sensitive information, maybe even personally identifiable information for your users. Since attacks are often aimed at these types of applications, you will definitely want to make sure your policies that identify high and critical threat security vulnerabilities are included during the evaluation of these types of applications.

Unfortunately, especially as the number of applications in your business increases, identifying an application by name may not be helpful. To address this, tags provide a way to quickly identify characteristics of an application.

Using specific text and color, tags can help group particular applications with similar attributes. While the tag can ultimately be anything you want, and attached to any application, you will want to take a much more thought-out approach, similar to what is recommended for labels.

As we will see later, in order to maximize the benefits tags can offer, you will want to take advantage of tag matching between policies and applications. For now though, let's see how to create, apply, and delete tags.

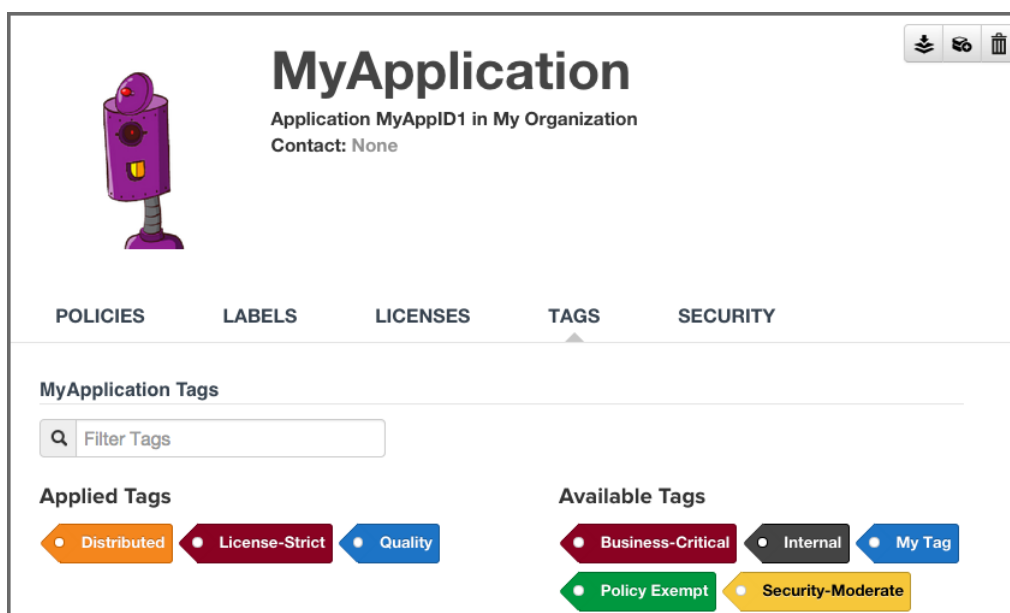


Figure 7.10: Example of Applied Tags

7.9 Creating, Editing, and Deleting Tags

Tags are created, edited, and deleted at the organization level and then applied individually for each application. There are two key ways to create a tag, again, only done at the organization level.

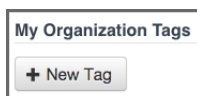
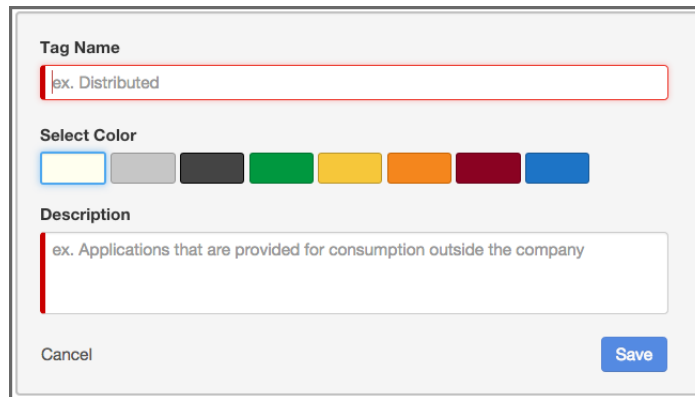


Figure 7.11: Using New Tag Button



Figure 7.12: Using Global Create Button

There is really no difference here, as both require that you have the organization open at the time of creation. The one advantage with using the Global Create button is that you can create a tag no matter which tab of the currently selected organization you are in, whereas you will need to be on the Tags tab otherwise.



The image shows a 'Create Tag' dialog box with the following elements:

- Tag Name:** A text input field with the placeholder text 'ex. Distributed'.
- Select Color:** A row of eight color swatches: white, light gray, dark gray, green, yellow, orange, red, and blue.
- Description:** A text input field with the placeholder text 'ex. Applications that are provided for consumption outside the company'.
- Buttons:** 'Cancel' and 'Save' buttons at the bottom.

Figure 7.13: Creating a Tag

There are three elements of tag:

Tag Name

When creating your tag, keep in mind that the tag describes characteristics of an application, and will be used to match an application to corresponding policies. The name should be easily identified by the user.

Tag Description

The tag description is displayed when a user hovers the mouse over the tag. This can offer additional information, such as the types of policies that will be matched to applications that have applied the tag.

Tag

Color

The color selection is left to however your organization chooses to implement. The default is white.

If you made a mistake and want to edit the tag, simply click on the tag body (anything but the *x*), and you can edit the tag information. However, if you want to permanently delete the tag, click on the *x*.

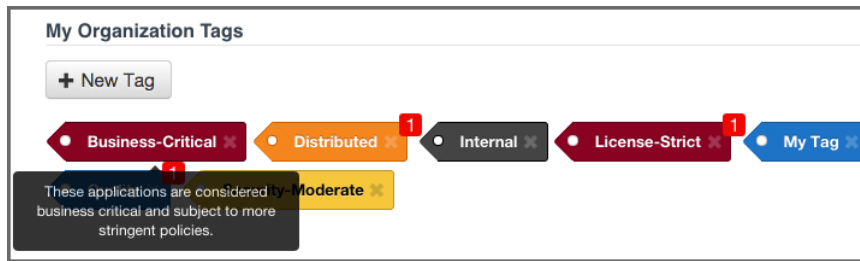


Figure 7.14: Example of Tags with Description

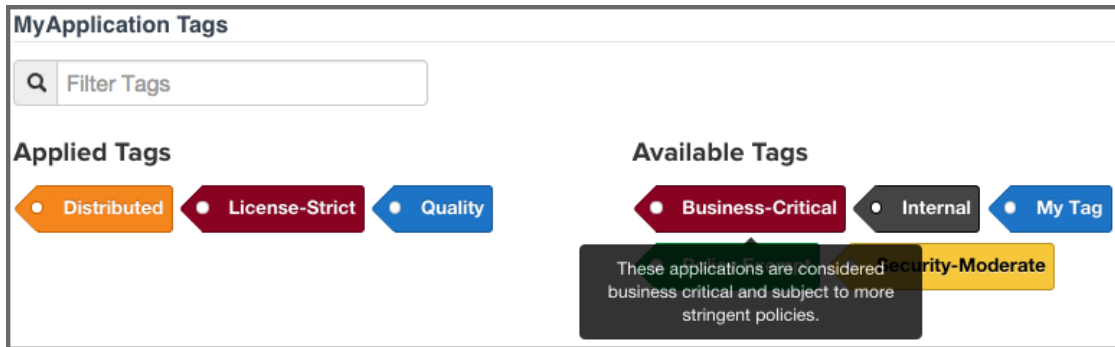
Note

Deleting any tag will ask for you to confirm, since that action can not be undone. If the tag is currently applied to an application you will be shown the names of all applications that would be affected before you confirm the deletion. You will not be able to delete a tag that has already been related to a policy, and will be shown the names of any related policies if you try. Should you still wish to delete the tag, you will have to disassociate it from any related policies first.

7.10 Applying a Tag

Depending on how your business uses tags, and establishes control within CLM, the people applying tags may be different from those creating them. It is important though to understand that while tags are provided to identify characteristics of an application, a more important usage is to provide a way for policy managers to create specific policies that consider those application characteristics. For this reason, when applying a tag, your application may be evaluated by a specific set of policies. This is a good thing, but it also makes the application of tags an act that requires careful consideration. To apply a tag to an application, follow the instructions below.

1. First, log in to the Sonatype CLM Server using a user account with at least Owner-level permissions for the application (a member of the Owner Group).
 2. Next, click on the *Application* link, and then click on the *application* you want to apply the tag to. The **Application Management** area will be displayed.
 3. Now, click on *Tags* tab. There are two columns, one for available tags, as well as those that have already been applied. Simply click on the tag to move it from one column to the other. If there are a lot of tags, and you are having trouble locating a specific one, simply type in the filter the name of tag you would like to use.
-



Tip

Mouse over a tag to see the full description.

7.11 Matching Policies to Specific Applications

By now, you have likely created tags, and perhaps even applied some to your applications. Those are great features, but the real power of tags comes when we match a policy to a specific set of applications.

Up to this point (before tags), an organization-level policy would apply to all applications. To address this, you could create a new organization, or develop specific policies for each application, but in both cases, that results in a lot of micromanagement. In contrast, tags provide an opportunity to create a policy and then pick unique groups of applications (based on their applied tags) the policy should be evaluated against.

Given this, it is important to think about the applications your business develops, as well as the types of policies you will use to evaluate your applications. Elements like the type of data, the exposure (public or private), as well as whether or not the application interfaces with the Internet, are a great place to start.

When you create your tags, make sure that it's clear to users that will be using the tags. In other words, it shouldn't be ambiguous as to the type of applications the tags represent. For example instead of creating the tag, *External*, a more descriptive tag would be *Distributed*. Some additional tag suggestions might be:

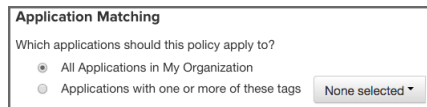
- Sensitive Information
 - Personal Information
-

These are just suggestions of course, but you should get the key point. When adding a tag to an application, you can expect policies that have identified the same tag to be evaluated against your application.

Now, that's quite a bit of discussion on the theory and proper way to utilize tags, let's take a look at how to make the match happen.

To select the tag a policy will be evaluated against:

1. Create a new policy, or edit an existing one.
2. In the Application Tags area of the policy editor, choose the tags that represent the applications you want to evaluate the policy against. By default, no tags will be selected. This means the policy will apply to all application regardless of their tags.
3. Finish creating or editing your policy and click the *save* button. From this point forward, the policy will be evaluated against applications based on the tags you selected. In addition, applications will only see the policies they are evaluated against.



7.12 Viewing Tag-based Policies

Policies that have been set to match applications with specific tags are visible in the same area as all other policies. However, there is a slight difference between what is displayed at the organization level and the application level.

At the organization level

All policies for the organization will be displayed. Policies that have selected specific applications, based on a matching tags applied to those applications, will be indicated by a special icon.


At the application level

Only the policies that an application is evaluated against will be displayed in the Policy tab. This includes:

- Policies created at the organization level, and set to match all applications.

- Policies created at the organization level, and set to match specific tags currently applied to the application.
- Policies created at the application level.

Tip

When viewing policies at the application level, be sure to look for the special tag icon , which indicates the application is evaluated against the policy given a tag (or tags) applied to the application.

7.13 Summary

It can be easy to forget about policy elements, and in most cases these should be reserved for more advanced users. For example, deciding what labels to use, and binding them to a specific process is very important in helping to ensure they aren't overused. Equally important is creating tags that will provide an automatic evaluation of applications against policies with matching tags. In the case of license threat groups, you likely want to consult your legal team to make sure you remain compliant to parameters they have established. In fact, if you haven't already they should be included in your policy development discussions.

As far as this section goes, here's what you should have taken away.

- Understanding policy elements (labels, license threat groups, and tags).
 - Create a label and a condition based on it.
 - Create a license threat group and a condition based on it.
 - Create tags at the organization level, and apply to applications.
 - Understand the impact of matching policies to applications using tags.
-

Chapter 8

Manual Application Evaluation

In order to evaluate an application, you need to have created at least one organization and one application, as well as created or imported at least one policy at either the organization or application level. You will also need to make sure you have the proper permissions to view report information for the application you wish to evaluate.

While evaluations can be initiated from various tools featuring CLM integration (e.g. Sonatype CLM for CI, IDE, and Nexus Pro), the quickest way to get started is to perform an evaluation via the CLM server.



This will generate a report for your application quite easily, and is a great way to create a quick baseline of your application's health.

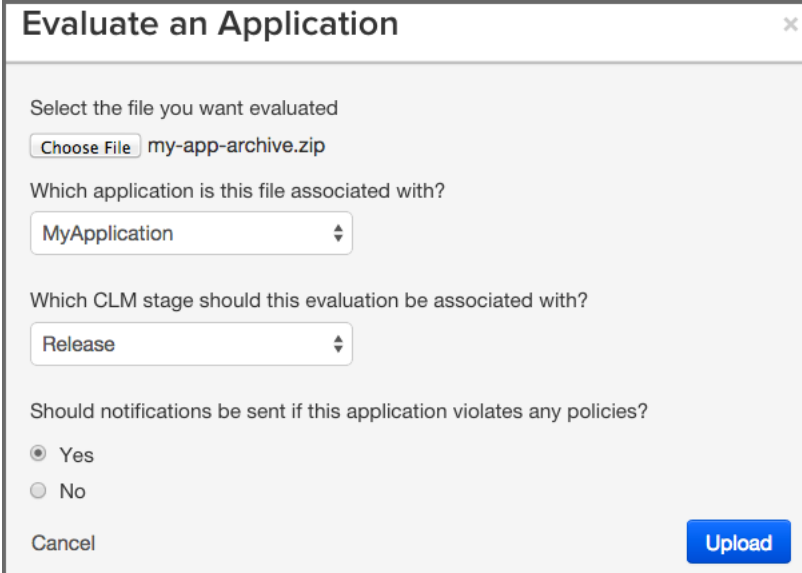
8.1 Evaluating via the CLM Server

As mentioned previously, before you can evaluate an application, you will need to make sure you have:

- Created an organization
 - Created an application
 - Imported or created a policy
-

With the above complete, you are ready to evaluate an application via the CLM Server.

1. First, log in to the CLM Server. At a minimum you will need to be a member of the Application Owner role.
2. Next, click the *Organizational Design* icon  to access the **Organizational Design** area. Once there, click *Applications* (located in the menu on the left side of the screen), and then choose an application.
3. In the top right of the **Application Management** area, click the *Evaluate an Application* icon .
4. A modal dialog will display providing a number of required fields.
 - a. First, choose the bundle (application) you want evaluated. Clicking *Choose File* will allow you to browse your directories for the application you wish to evaluate.
 - b. Next, choose the application in Sonatype CLM you want to associate with the evaluation. By default, this will be pre-populated with the name of the application you first selected.
 - c. After choosing the application to evaluate, you will need to specify the stage, this will affect where the report is displayed, and will overwrite the most recent report for the application and stage selected.
 - d. Finally, if you have configured notifications for your policy, or policies, you can choose whether or not you want those notifications sent.
5. Click the *Upload* button to begin evaluating the chosen application.
6. The Evaluation Status will display, showing you the progress of your evaluation. When complete, you can click the *View Report* button to view the results of your evaluation.

A dialog box titled "Evaluate an Application" with a close button (X) in the top right corner. The dialog contains the following fields and controls:

- Text: "Select the file you want evaluated"
- File selection: A button labeled "Choose File" followed by the text "my-app-archive.zip".
- Text: "Which application is this file associated with?"
- Application dropdown: A dropdown menu showing "MyApplication".
- Text: "Which CLM stage should this evaluation be associated with?"
- CLM stage dropdown: A dropdown menu showing "Release".
- Text: "Should notifications be sent if this application violates any policies?"
- Radio buttons: Two radio buttons labeled "Yes" (selected) and "No".
- Buttons: A "Cancel" button on the bottom left and an "Upload" button on the bottom right.

Figure 8.1: Evaluate an Application

Note

You can also evaluate an application via the *Organizations* area, simply click on *Organizations* instead of *Applications* and follow the instructions from there. You will still need to have created an application, and the application won't be pre-filled for you in the form.

8.2 Successful Evaluations and Report Generation

If your evaluation completed successfully, a report will be generated for the application.

The log output of the command execution will provide a summary as well as a link to the produced results similar to

```
[INFO] Policy Action: Warning
[INFO] Summary of policy violations: 4 critical, 85 severe, 46 moderate
[INFO] The detailed report can be viewed online
at http://localhost:8070/ui/links/application/my-app/report/95c4c14e
```

This report is available on the Sonatype CLM Server in the *Reports* section. If you kept our defaults, the report will be listed under the Build Stage. So, what are you waiting for? You should see something similar to the results displayed in Figure 8.2

Q

Filter Applications





Application Name ▼	Build Violations	Stage Release Violations	Release Violations	Contact	Organization
<div> MyApplication</div>	<div>281593</div> <div>1 minute ago</div>	<div>281593</div> <div>1 minute ago</div>			My Organization
<div> My Application 4</div>			<div>65</div> <div>1 month ago</div>		My Organization 3
<div> My Application 3</div>	<div>6111</div> <div>5 months ago</div>				My Organization 7
<div> My Application 2</div>	<div>6111</div> <div>5 months ago</div>	<div>6111</div> <div>1 month ago</div>	<div>6111</div> <div>6 months ago</div>	John Smith	My Organization 4

Figure 8.2: Violations Report after Scan

Note

As mentioned previously, if you specify a stage not represented by in the Sonatype CLM Server, there will not be a visible link to the report.

8.3 Summary

Did you get a chance to evaluate some application? Pretty simple right? Your main goal should be to scan an application, which will vet components against the policy assigned either at the organization or application level. If you haven't already, be sure to go to the Sonatype CLM Server, and check out the results of your report.

Chapter 9

Reviewing Evaluation Results

The Application Composition Report provides the results of an evaluation of your application. The results are broken into three key categories:

- Policy Violations
- Security Vulnerabilities
- License Issues.

As mentioned previously, this will be the same report, whether you are using the stand-alone scanner, the CLM Maven plugin, the manual evaluation, or any of the integrated enforcement points (e.g. Sonatype CLM for CI, IDE, Nexus Pro).

Let's take a look at how to access the report first.


Note

Depending on the enforcement point, or the stage options you manually selected, your report may be listed under different stages in the *Reporting* area of the Sonatype CLM Server. For example, the default location for the stand alone scanner, is the build stage.

9.1 Accessing the Application Composition Report

No matter how the scan was performed, all reports reside on the Sonatype CLM Server and are automatically associated with the corresponding application (via the application identifier). However, there are two distinct ways to access the Application Composition reports.

Via the Reports Area

When you log into the Sonatype CLM Server the Dashboard is displayed by default. Click the **Reports** icon . If multiple applications have been scanned, you will see all of them here.

Note

You will need to be a member of at least the developer group for the application you wish to see a report for.



Important

Users of Nexus CLM Edition do not have access to the Sonatype CLM Dashboard. Because of this, these users will not be taken to the dashboard after logging in, nor will they see the dashboard icon. Rather, the reports area will display by default.

Each application has a separate row with columns for:

- Application Name
- Violations (by stage)
- Contact
- Organization (for the corresponding application)

Each Violation column contains a Violation Summary (total counts for *Critical*, *Severe* and *Moderate* policy violations). In addition, the time the last report was generated (e.g. *2 minutes ago*) is provided.

To access the Application Composition report, click the Violation Summary for the corresponding application and stage.





Filter Applications						
Application Name ▼	Build Violations			Stage Release Violations		
	Release Violations			Contact		
	Organization					
 MyApplication	28	159	3	28	159	3
	1 minute ago			1 minute ago		
 My Application 4				6	5	
				1 month ago		
 My Application 3	6	11	1			
	5 months ago					
 My Application 2	6	11	1	6	11	1
	5 months ago			6	11	1
				1 month ago		
				6 months ago		
				John Smith		
				My Organization 4		

Figure 9.1: Reporting Area

Tip

By default this view will be sorted alphabetically by the application name. In addition to the filter, you can also click on the application or organization columns to sort alphabetically ascending/descending.

Via the Application Area

The Application area is the same place where you can manage policy for your application, reviewing policies unique to the application, as well as those inherited from the organization. Located just below the application identifier and organization, you will see three columns:

- Build
- Stage Release
- Release

These represent the Sonatype CLM stage where the report was generated for/from. For example, if you use the Sonatype CLM stand-alone scanner and don't specify the CLM Stage, it will default to build. When your scan completes and the report is uploaded, it would appear below *Build*. This is highlighted in Figure 9.2.

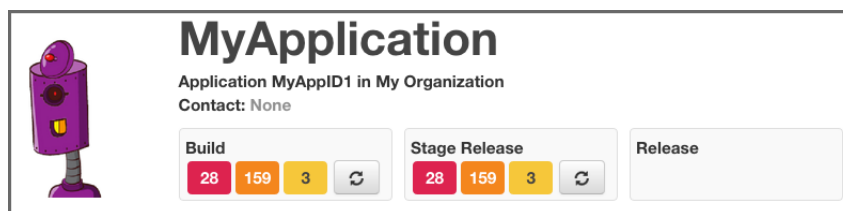


Figure 9.2: Application Area

9.2 Reviewing the Report

At first glance, you may be surprised at what you see. If you expected an application to have no issues, and now see it has a great deal, don't get upset. . . yet.

In many cases, a policy can be too stringent or may indicate issues that are not exactly applicable to your application. For example, you may have a security issue that would only affect applications exposed to the public, while your application is for internal use only. Another great example is a license that constrains your code in the event you intend to sell the application.

With that worry out of the way, let's take a look at what's actually in each report.

The *Summary* tab of the report shows a breakdown of what was found. This includes counts for policy violations, security vulnerabilities and license-related issues.

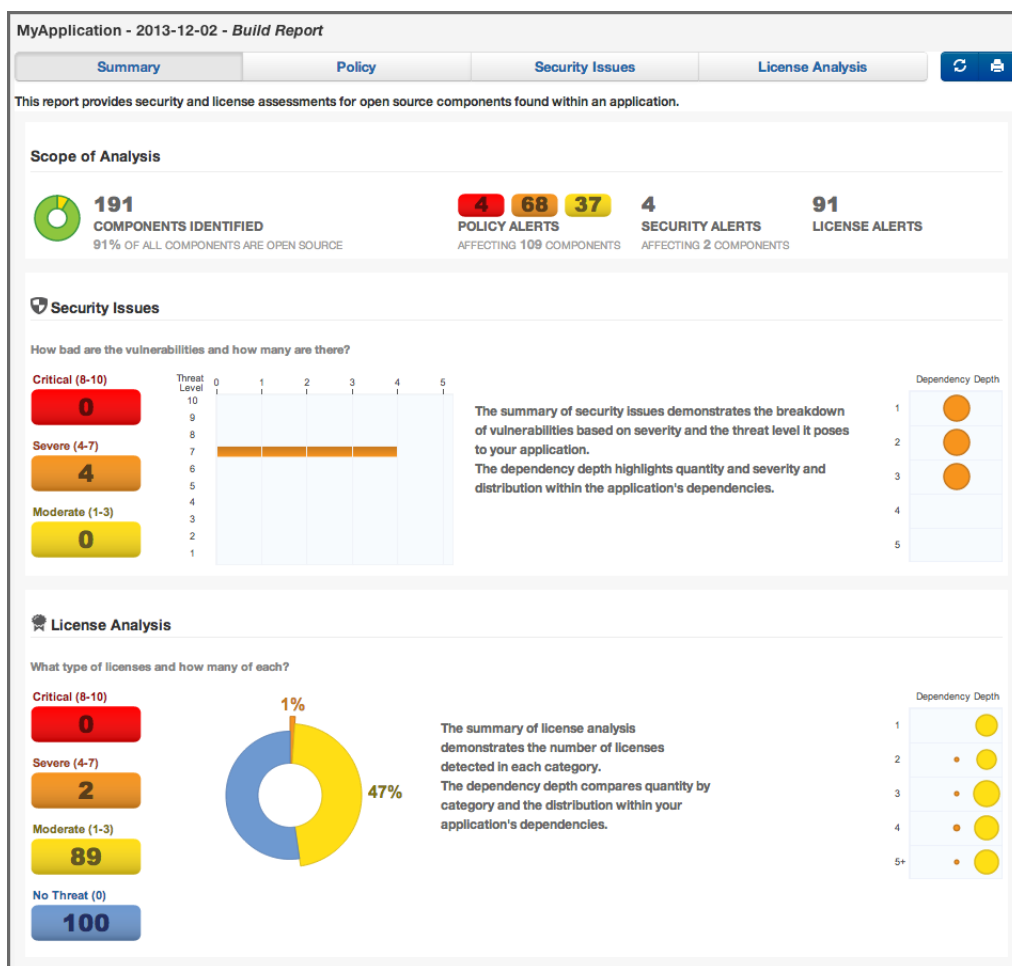


Figure 9.3: Summary Tab of an Application Composition Report

The *Policy* tab provides a list of all components that were found in your application. An example is displayed in Figure 9.4. The list of components is ordered by the level of the threat violation that has been assigned to the policy. In instances where a component has violated multiple policies, only the violation with the highest threat is displayed.

To view the other violations you can use the component information panel (described below), or change what is displayed using the Violations filter on the right. This will allow you to see all violations for your component, though that may result in the appearance of duplicated components.

Tip

We have an **entire guide** dedicated to the various Sonatype CLM Reports.

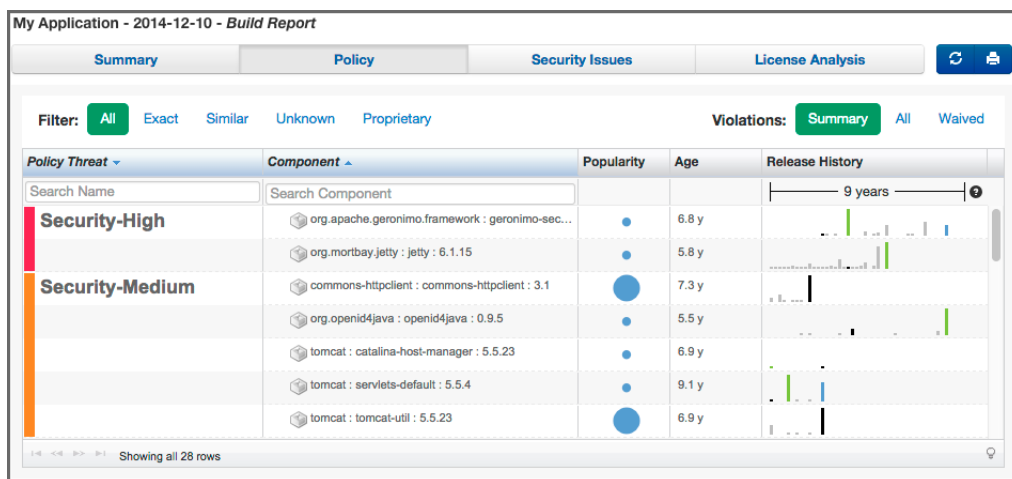


Figure 9.4: Policy Tab of an Application Composition Report

The *Security Issues* tab displayed in Figure 9.5 displays all components containing security issues.

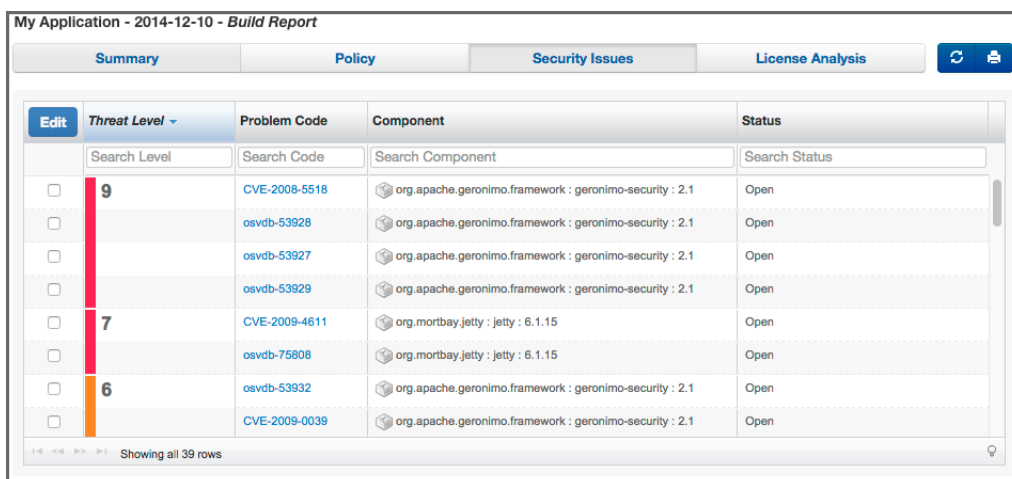


Figure 9.5: Security Issues Tab of an Application Composition Report

The *License Analysis* tab displayed in Figure 9.6 displays all components and the determined details about their license(s).

Edit	License Threat	Component	Status
<input type="checkbox"/>	Not Declared, GPL, GPL-2.0+	cobertura : cobertura : 1.6	Open
<input type="checkbox"/>	MIT, Apache-2.0, CC-BY, GPL-2.0+, LGPL-2.1+, ...	edu.ucar : unidataCommon : 4.2.20	Open
<input type="checkbox"/>	GPL-3.0, No Sources	javancss : javancss : 29.50	Open
<input type="checkbox"/>	Apache-2.0, Non-Standard	org.mortbay.jetty : jetty : 6.1.15	Open
<input type="checkbox"/>	LGPL-2.1, BSD-3-Clause, LGPL-2.0+, Non-Standard	edu.stanford.ejalbert : BrowserLauncher2 : 1.3	Open
<input type="checkbox"/>	LGPL-2.1, No Sources	org.opencms.modules : com.alkacon.opencms.v8.twitter : 8.0.2	Open
<input type="checkbox"/>	LGPL-3.0, LGPL	ch.qos.logback : logback-access : 0.6	Open
<input type="checkbox"/>	EPL-1.0, No Sources	org.eclipse.foundation : org.apache.lucene.spellchecker : 2.9.1.v20100...	Open

Showing all 27 rows

Figure 9.6: License Analysis Tab of an Application Composition Report

In the *Policy*, the *Security Issues* as well as the *License Analysis* tabs, you can get access to more information about a particular component by clicking on a row in the table representing the component you are interested. The Component Information Panel CIP, with an example displayed in Figure 9.7 shows more specific information about the component.

Component Info | Policy | Similar | Occurrences | Licenses | Edit Vulnerabilities | Labels | Audit Log

Group: **org.apache.struts.xwork**
 Artifact: **xwork-core**
 Version: **2.2.1.1**

Declared License: **Apache-2.0**
 Observed License: **BSD, Apache-2.0**
 Effective License: **Apache-2.0, BSD**

Highest Policy Threat: **9** within 5 policies
 Highest Security Threat: **10** within 29 security issues

Cataloged: **3 years ago**
 Match State: **exact**
 Identification Source: **Sonatype**

Popularity: [Chart showing popularity over time]
 License Risk: [Chart showing license risk over time]
 Security Alerts: [Chart showing security alerts over time]

Figure 9.7: Component Information Panel CIP for a Specific Component

Clicking on the *Policy* header in the component information panel displays all policy violations for the

selected component. As you can see from the example displayed in Figure 9.8 the policies as well as the constraints and the condition values that triggered the policy violation are displayed.

Policy/Action	Constraint	Condition Value	Waivers
Security-High	CVSS >=7 and <10	Found 15 Security Vulnerabilities with Severity >= 7 Found 27 Security Vulnerabilities with Severity < 10 Found 29 Security Vulnerabilities with Status OPEN	Waive
Security-Medium	CVSS >=4 and <7	Found 25 Security Vulnerabilities with Severity >= 4 Found 14 Security Vulnerabilities with Severity < 7 Found 29 Security Vulnerabilities with Status OPEN	Waive
Security-Unscored	CVSS = 0	Found 2 Security Vulnerabilities with Severity = 0 Found 29 Security Vulnerabilities with Status OPEN	Waive
Security-Low	CVSS < 4	Found 4 Security Vulnerabilities with Severity < 4 Found 29 Security Vulnerabilities with Status OPEN	Waive

Figure 9.8: Policy Section for a Specific Component Displayed on the Component Information Panel

A number of specifics used in the tabs and the panel are detailed in the following:

Threat Level

We briefly mentioned above, that policy violations are organized by threat level. The threat level breakdown is as follows.

- Red / High (10 - 8) - Indicates a component with a severe threat, and should be treated seriously.
- Orange / Medium (7 - 5) - Indicates a component with a moderate threat, and should be treated seriously.
- Yellow / Low (4 - 2) - Indicates a component with a low threat, and may not pose any serious threat to your application.
- Dark Blue / Informational (1) - Indicates that there is a very low threat, and you should just be aware of a possible issue.
- Light Blue / None (0) - Indicates that no policy has been violated by the component.

Matching

It's likely that you started seeing an area that indicates *matching*. As a quick definition, matching employs a series of in-depth algorithms to determine if a component found in your application matches anything known to the Central Repository, or known to the Sonatype CLM Server. That's right, through a claiming process and a proprietary component configuration, you can teach Sonatype CLM to recognize components it may not have otherwise.

PDF Printing

The application composition report can be printed to PDF simply by clicking the print icon located in the upper right corner of the report.

Re-evaluation

Eventually, when you begin to manage and modify policies, you may simply want to compare the results from the most recent report with your policy modifications. The re-evaluate button, located to the left of the pdf/print icon will allow you to refresh the results without having to generate a whole new report.

9.3 Summary

With the ability to access and review results of your scanned applications, you are well on your way towards true governance of your components. In many ways, you are now nearing the end of the component lifecycle. Remember, even with these results your most important tool will still be communication. Don't be dismayed or persuaded by the results. Go over everything with your teams and work towards your business goals together. For now though, this is what you should take away:

- There are four tabs on a application composition report - *Summary*, *Policy*, *Security Issues* and *License Analysis*.
 - You can inspect details about a specific component in the component information panel CIP.
 - A report can be created in PDF format to allow printing.
 - Refreshing a report can be triggered from the user interface.
-

Chapter 10

Importing Policies

Setting up policies can be quite complex and labor intensive. To make the process easier and give you a head start we have created some sample policies and provide an import feature.

We actually recommend you don't begin using Sonatype CLM by creating a bunch of policies right out of the gate. Instead, we've created a set of policies, which include other policy elements such as labels and license threat groups, that you can import into your Sonatype CLM installation.

Eventually, and there is a very short time between now and eventually, you will need to create, or at least modify, policies. For now, we'll want to focus on populating your organizations and applications with policies provided by Sonatype.

10.1 Sonatype Sample Policy Set

The easiest way to establish policies for your applications is to use the sample policy set provided by Sonatype. While the included policies are not meant to be a perfect match for every business, they have been created with our extensive experience working with customers and developing policy for our own internal practices.

Our sample policy set can be downloaded here:

Sonatype-Sample-Policy-Set.json

This policy set is an example of managing components for security, licensing, and architectural issues. It also introduces the detection of unknown and patched components used in building your applications. The sample policy set can be used to gather information about the components used to build your applications without warnings and failures occurring in the developer, build, or Nexus environments.


This is the perfect set of policies to use in order to gather information and understand how policy management will work for your environment, without potentially distracting the people who are building and delivering your applications.

Note

The sample policy set includes several, preset tags. The tags have been used in the Application Matching area for a number of the included policies. Policies using the tags will be indicated by a special tag icon. In order to utilize the policies, you must have applied the corresponding tag to your application(s). For more information on tags, please see the [Policy Elements section of our Policy Management Guide](#).

10.2 Importing a Policy to an Organization

Once you have acquired a policy file to import, you can follow these steps:

1. Log into your Sonatype CLM server with a user account that has proper permissions to import policy for a specific organization (at least a member of the owner group for the organization would be required).
 2. Next, click the *Organizational Design* icon  to access the **Organizational Design** area.
 3. Click on *Organizations* in the left menu, and then click the organization you wish to import the policy to.
 4. Click the *Import* button in the top right corner of the organization view displayed in Figure 10.1.
 5. Click the *Choose File* button in the **Import Policy** dialog displayed in Figure 10.2 and select the policy JSON file in the file browser.
 6. Click the *Import* button in the **Import Policy** dialog.
 7. Confirm that the list of policies contains the imported policies.
-

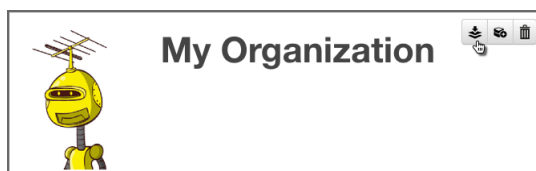


Figure 10.1: Organization View with Import Button

If you are importing to an organization, that already has some policies, labels, license threat groups, and/or tags set up, consider the following rules:


- Existing policies will be deleted during the import procedure.
- Importing policies also includes an import of associated policy elements (labels, license threat groups, and tags). The following logic will be used for Policy Elements:
 - Labels - the CLM server attempts to match labels against existing ones in a case-insensitive manner. This allows for updating the description or color of existing labels, while preserving any triage effort already done to apply these labels to components. If your import contains labels that aren't already present in the system, they will be created.
 - License Threat Groups - the CLM server will delete all existing license threat groups, and then import the new ones.
 - Tags - the CLM Server attempts to match tags against existing ones in a case-insensitive manner. This allows for updating the description or color of existing tags, while preserving any current matching of tags between policies and applications.



Figure 10.2: Import Policy Dialog

10.3 Importing a Policy to an Application

An application inherits policies from the organization. However it can be useful to have additional policies for fine grained control.

1. Log into your Sonatype CLM server with a user account with an **Administrator** role or as an **Owner** of the application you wish to import policy to.
2. Next, click the *Organizational Design* icon  to access the **Organizational Design** area.
3. Two columns will be displayed on the left. Click on *Applications*, and then click the application you chose to import the policy to.
4. Click the *Import* button in the top right corner of the application view, which is identical to the organization view displayed in Figure 10.1.
5. Click the *Choose File* button in the **Import Policy** dialog displayed in Figure 10.2 and select the policy JSON file in the file browser.
6. Click the *Import* button in the **Import Policy** dialog.
7. Confirm that the list of policies contains the imported policies.

The policy information will be imported, and the following rules will be applied:

- Duplication of organization policies is invalid, so you will not be able to import the same policy file into an organization and then into an application associated to it.
- When a policy is imported, any existing application policies will be deleted and replaced with the imported configuration.
- For label imports, the same logic as during imports at the organization level described in Section 10.2 applies.
- Attempting to import policies that contain tags will cause the entire import to fail.

10.4 Summary

If you are having trouble coming up with your own, custom policies, importing any of our sample policies can be a great way to get started. While this may not be an exact fit, in most situations it provides a good

baseline for improving the health of your applications. Better yet, if you want to modify the policies after import, you can do that as well.

Chapter 11

Policy Monitoring

At some point, your applications will be out of development, have completed their final build, moved beyond staging, and have been officially released. However, while there shouldn't be changes to your application that is now considered to be in production, new security vulnerabilities and license issues could arise. For this reason, as well as any other, Sonatype CLM allows you to monitor individual policies for each application.

When a policy is monitored, you pick an application, as well as a Sonatype CLM stage to monitor to use as a base for evaluating policy against. After that we'll show you how to configure which policies you would like to receive a notification for, given a component is found to be in violation.

If some of this sounds familiar, that's good, because it is nearly identical to standard policy evaluation, component violations, and the notification option for policies. There really is no difference other than being able to choose which Sonatype CLM stage you will use for monitoring. Though that is a powerful option.

In this section, we'll cover everything you need to setup policy monitoring at the organization and application level. In general, we make a few assumptions, including:

- You have your Sonatype CLM Server up and running, and accessible.
 - You have created an organization and an application, as well as setup or imported some basic policies.
 - You are somewhat familiar with the Sonatype CLM Server.
-

If any of these sounds like strange concepts, you'll want take a few steps back and go over those topics first. With that said, let's go monitor some policies.

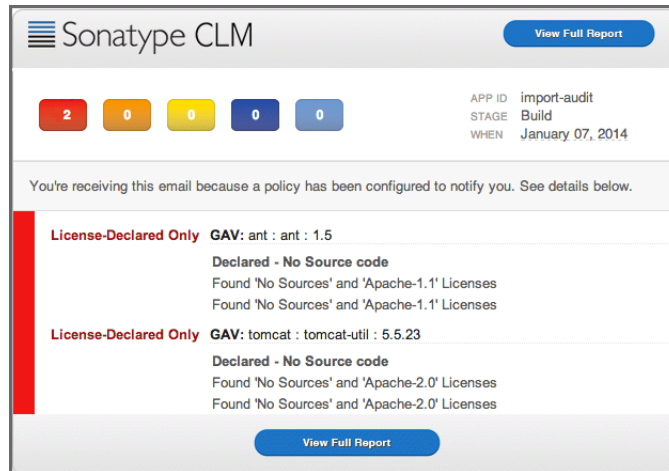


Figure 11.1: Example of a Policy Monitoring Email

Note

Policy Monitoring is not available to customers with a Nexus CLM License. Contact your CLM Admin for additional information.


11.1 Setup Policy Monitoring for an Application

The setup for policy monitoring is straightforward. In general you will likely want to avoid monitoring every single policy. Not only is that a lot of policies to monitor, your signal to noise ratio will be off. That is, you might possibly get a lot of notifications for things like old component, or components that are now unpopular.

That's not to discourage you from monitoring these policies, they are important to. However, monitoring, and in turn, the notifications that are associated with monitoring are best reserved for policies that deal with elements like security vulnerabilities and license issues - that is, those representing a high level of risk.

First, choose an application to monitor

While you can choose any application to monitor, most people start by monitoring an application in production. In many cases production applications have likely been around longer than your implementation of Sonatype CLM. However, you are not prevented from choosing any application to monitor.

1. Log into your Sonatype CLM server with a user account that has proper permissions to make changes to an application policy (at least a member of the owner group for the application would be required).
2. Next, click the *Organizational Design* icon  to access the Organizational Design area.
3. Click *Applications* in the menu on the left, and then click on the application you want to monitor.
4. The **Application Management** area will be displayed.

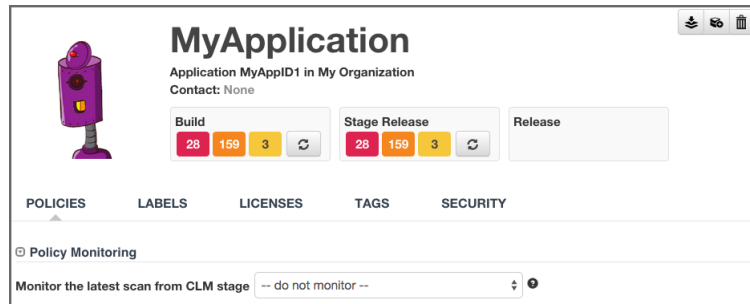


Figure 11.2: Access Application Management Area

Great, now you can move on to picking the stage you want to monitor.

Next, choose a Sonatype CLM Stage to monitor

Each of the Sonatype CLM Stages available for monitoring, are identical to the Sonatype CLM stages displayed when setting policy actions. In general, if you are going to be monitoring a production application, you will want to choose the stage that represents the most recent data. For our example, we'll choose Build, but again, you can choose any stage.

1. First, click on the *Policy Monitoring* section to reveal the options for selecting a stage to monitor.
2. Next, using the drop down menu, select the stage you wish to monitor. You will notice this drop down selection completes the sentence, " *Monitor the latest scan from CLM stage...* "



Figure 11.3: Selecting a Sonatype CLM Stage to Monitor

You are almost there, now all you need to do is add notifications.

Each policy you want to monitor will need to have someone added to the notifications. These are set below the **Actions** section of each policy.

1. From the *Application Management* area, click on the edit button of the policy you want to monitor.
2. Next, just below the **Actions** section, click the *Email* icon to open the Monitoring Notification dialog. Add the email addresses for those individuals you wish to have notified when a new violation for this application is encountered.
3. Click *Done*, and then *Save* to save your edits.

Security-High

Security-High 10

Constraints

CVSS >=7 and <10

Actions

Stage	Enforcement Points		Notifications
	Warn	Fail	
Procure			✉
Develop			✉
Build			✉ jwayman@sonatype.com
Stage Release			✉
Release			✉
Operate			✉

Monitoring Notifications

If new information becomes available and causes this policy to be violated, who should be notified?

✉

Cancel Save

Figure 11.4: Adding Email Recipient

Tip

Remember, you can only edit a policy based on your permissions and where it was created, if you don't see the edit button for a policy, you either need to adjust your permissions, or switch to the organization the policy was inherited from.

Congratulations! Your application now has a policy that will be monitored. To monitor more applications and/or policies, simply repeat the steps above.

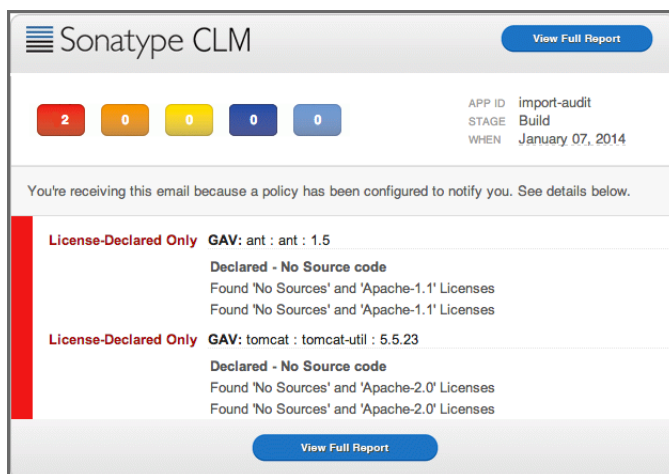


Figure 11.5: Sample Email Notification

Tip

While it is possible to follow these same steps and set policy monitoring at the organization level, you may want to think through that a bit more before blanketing all applications within a particular organization with policy monitoring notifications. In many cases, we find that monitoring is best done on a few, high risk, production applications.

11.2 Configuring Notification Times

By default any new notifications for policies that are being monitored will be sent out at 12 AM (per the CLM Server time). If you would like to update this, simply edit your `config.yml` file for the Sonatype CLM Server. The lines you will need to look for are as follows

```
# Hour of the day(0-23) to schedule Policy Monitoring execution. The
default is midnight.

#policyMonitoringHour: 0
```

11.3 Summary

At end of this section you should find yourself ready to go out and monitor policy for any application. Remember though, policy monitoring is not the only way to share results, and you should think about which applications, and which policies should be monitored.

Sonatype CLM already provides a number of ways to be notified and informed when a component in one of your applications violates a policy. While you could go and mimic this using the policy monitoring feature, the better practice is to use the policy monitoring notification for only those policies and applications that pose the greatest potential risk. In this, we stress, there's likely nothing worse than getting an email at 3 AM in the morning telling you that a component used by your company's intranet is now violating your architecture age policy. While that's important information, it's not 3 AM - important information.

Chapter 12

Conclusion

Congratulations! Understanding policies is no small feat. While it's easy to use terms like governance, risk reduction, and policy enforcement, you now have the tools to work towards those goals. Just to highlight our goals, here are some things you can certainly take away:

- You understand the difference between applications and organizations.
- You can identify the various parts of a policy.
- You know how to get started with your own governance using the Sonatype sample policies.