User Guide

1.7.2
Monday, April 27, 2015

Table of Contents

User Guide

Getting Started

These instructions are for the Code Dx evaluation distribution. Please refer to the Install Guide for detailed information on getting Code Dx up and running.

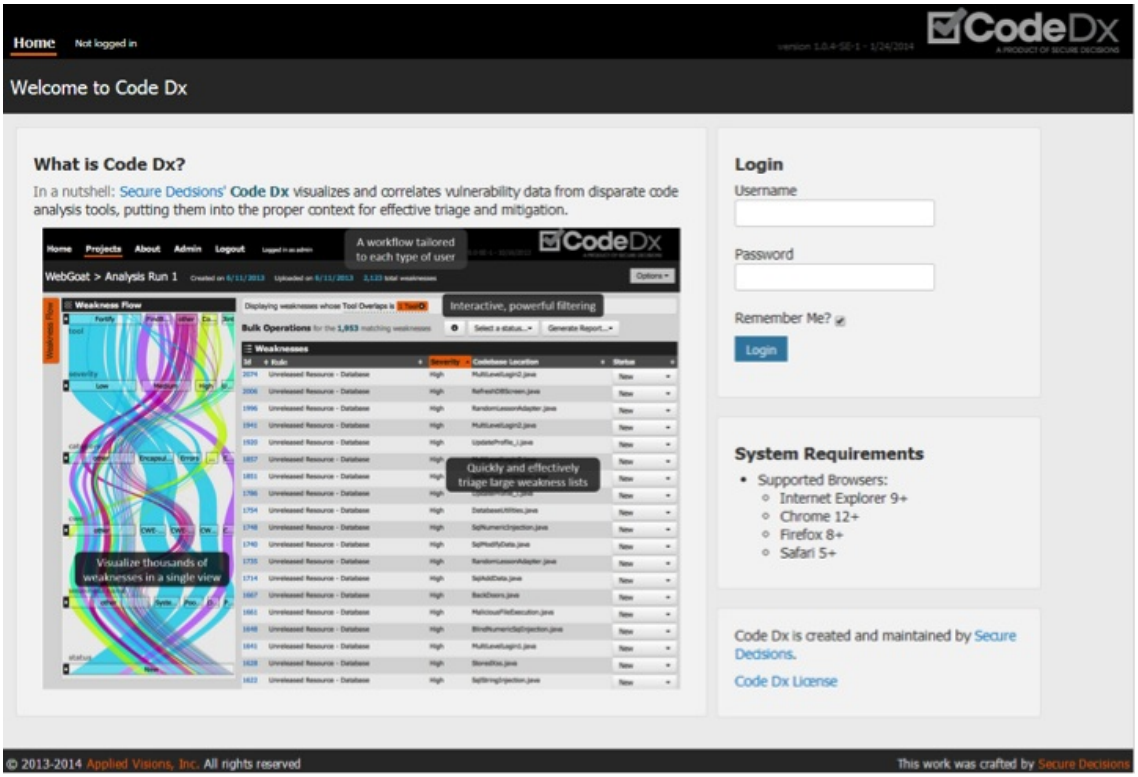
Please note that for .NET analysis, Code Dx requires the installation of the .NET runtime, FxCop (Code Analysis) 10, 11, or 12, and CAT.NET v1. See the Installing .NET Tools section at the end of this section on instructions on how to install these tools.

Starting Code Dx

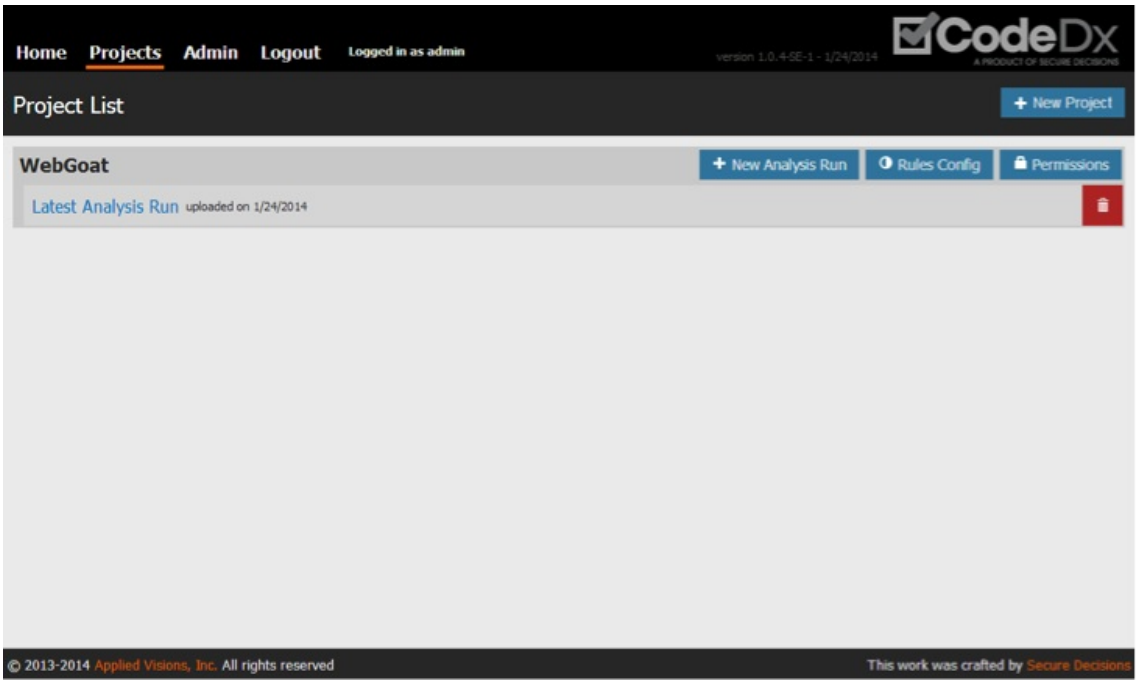
1. Unzip the Code Dx zip file to a location on your hard disk
2. Open the folder containing the extracted contents
3. Navigate to the codedx folder
4. Start Code Dx by:
    - On Windows: double-click start-win.bat
    - On Linux or Mac: in a shell, execute start-linux.sh or start-mac.sh, respectively
5. Wait until the following message appears in the console:The Server is now ready!
6. Your default browser should open automatically and open: http://localhost:8080/

Code Dx Quick Start

1. Once Code Dx is open in your browser, you should see this:



2. In the login area, sign-in using the default admin user:
    - Username: admin
    - Password: secret
3. Open the Projects page. From here you have the choice of opening the pre-loaded WebGoat project – by clicking on the Latest Analysis Run link – or creating a new project and uploading your own Java binaries/source, C/C++ source, Ruby on Rails source, JavaScript source, Python Source, or .NET binaries/source.



Installing the .NET Tools

It is recommended that the latest version of .NET be installed.

Code Dx is capable of running multiple .NET analysis tools on your codebase. FxCop and CAT.NET are two of the supported tools and are developed and distributed by Microsoft. The end-user license agreements for these products forbid their redistribution, therefore, Secure Decisions is unable to legally bundle these tools. So in order for Code Dx to run these tools on your behalf, you must install them separately. Code Dx will then automatically discover their location and run them.

Depending on the version of FxCop you plan to use, it will either be bundled with Visual Studio (as Code Analysis) or in the Windows SDK. For the best results, install Visual Studio 2012 or 2013 Premium. This will give you the latest rules available. Code Dx will automatically discover the location of the latest version of FxCop installed on your machine. If you would like to provide a specific location, set the “fxcop.path” property in the Code Dx configuration file.

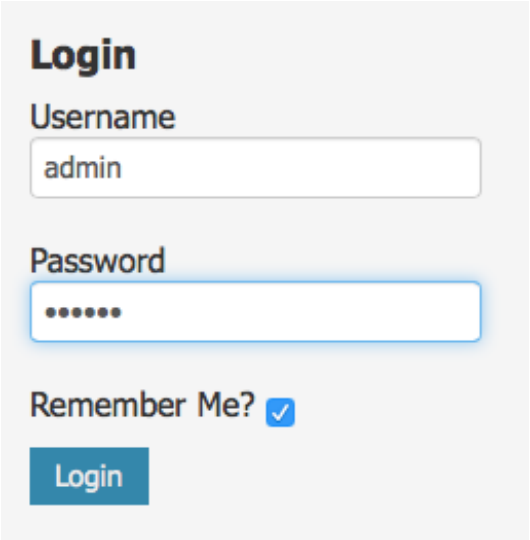
CodeDx will work with either CAT.NET 32-bit or CAT.NET 64-bit. These can be downloaded from the Microsoft website. CAT.NET 32-bit has an installer and Code Dx will automatically look in the default installation directory for this application. The 64-bit version is in a zip file. The best approach to using the 64-bit version is to

overwrite the 32-bit files with the 64-bit files. Alternatively, the path can be manually set using the “cat.net.path” property in the Code Dx configuration file.

Session Management

Logging In

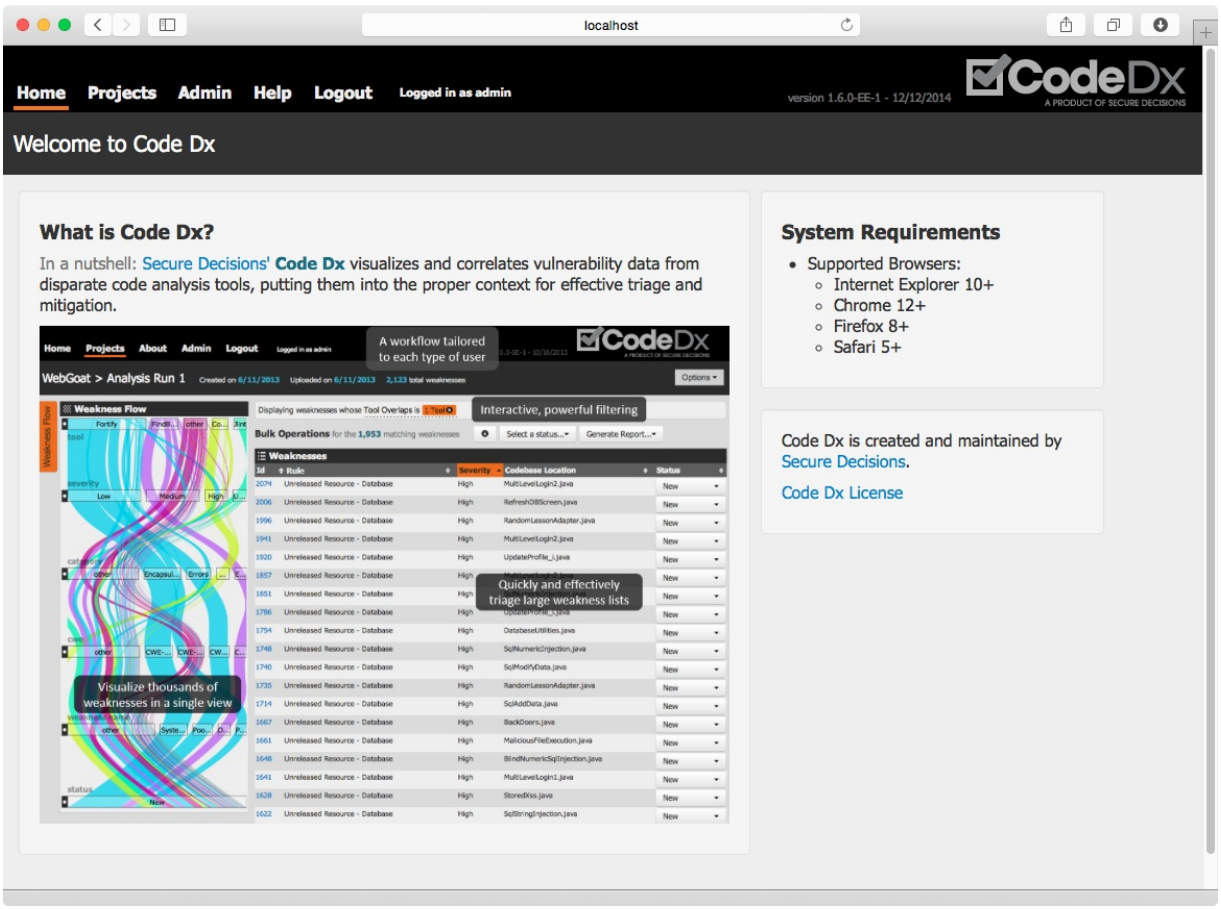
The first thing you should do is log into Code Dx. If this is the first time visiting the site after installation, the only useable login credentials will be the Super User's credentials, as configured during installation.



If *Remember Me* is checked, the server will remember your session until you explicitly log out. This means that even if you leave the site and come back, or if the server restarts, you will not need to log in again. The *Remember Me* option can be disabled entirely or configured to just remember the username, so please keep in mind that the behavior of this option might vary dependent on how Code Dx is configured by your administrator.

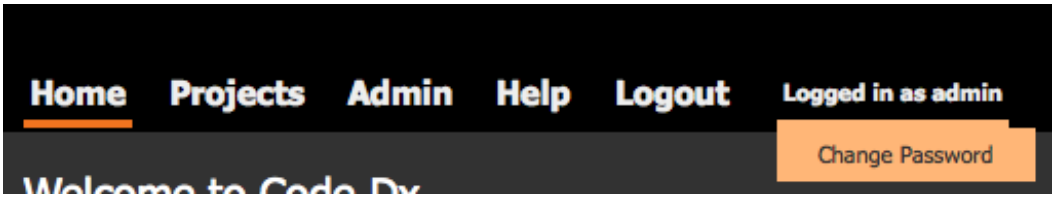
Once more users are added to the Code Dx system, they will be able to log in using this same form.

Log in as the Super User (for this guide, the Super User's username is admin). Once logged in, the Home page will display *Logged in as admin* at the top, and the log in form will be gone. Note that there are now additional page links to visit.

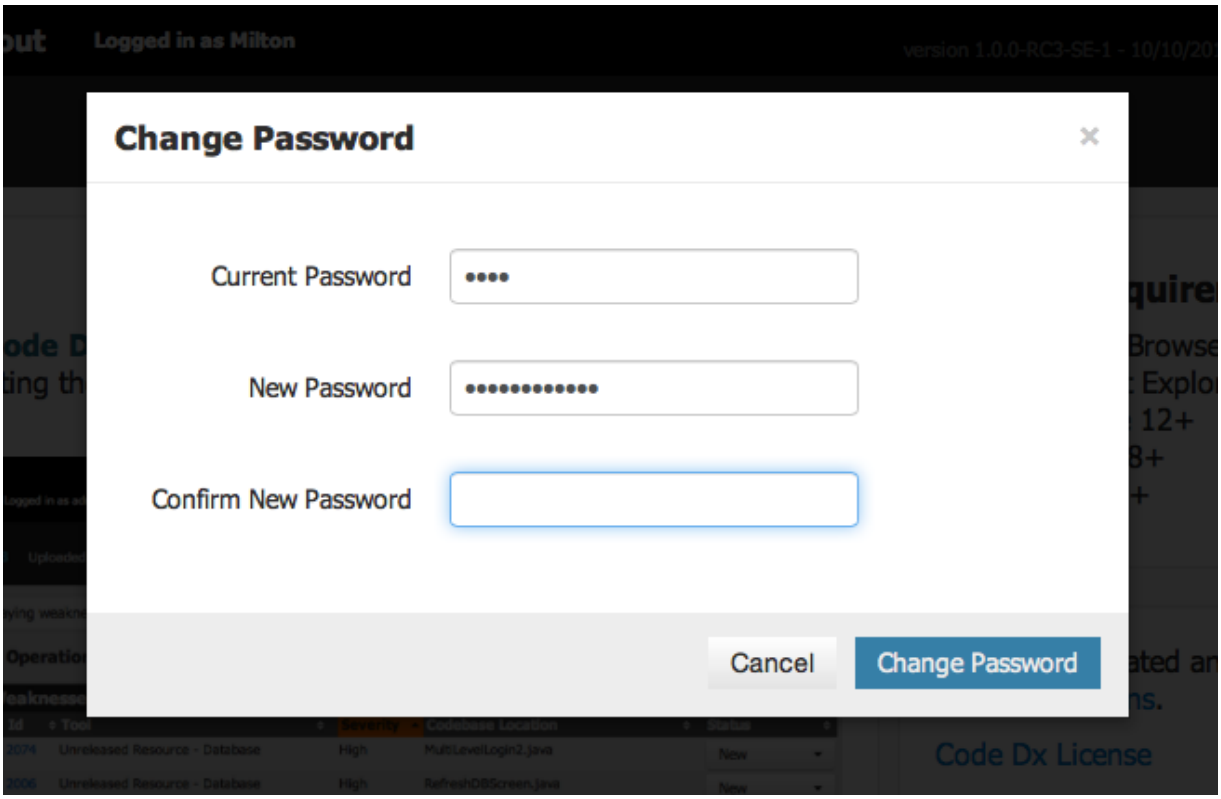


Changing your Password

To change your password, hover over the *Logged in as …* text in the top navigation area and select the *Change Password* option:



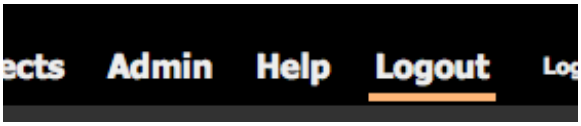
From there a, new dialogue will show up allowing you to change your password after confirming your existing one.



In the event you've forgotten your password, please contact your Code Dx administrator so that they may reset your account's password.

Logging Out

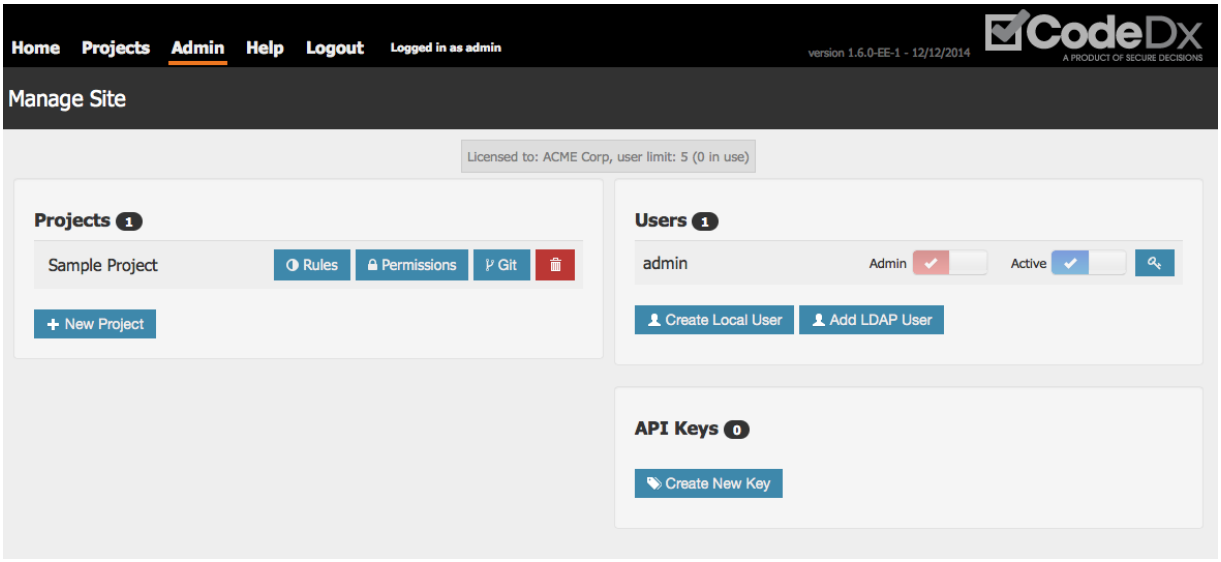
Logging out can happen via one of two approches. The first, is by selecting the *Logout* option from the navigation menu:



The second is an automated logout once your session expires. If you leave the Code Dx site for a certain period of time (this is configuration dependent but is usually 30 minutes) you will be automatically logged out. If you select the *Remember Me* option when logging in, Code Dx will remember you on that computer for your next site visit, but only if this option is enabled by your administrator.

Code Dx Administration

Admin users have access to the Admin page, where they can easily manage the Code Dx site.

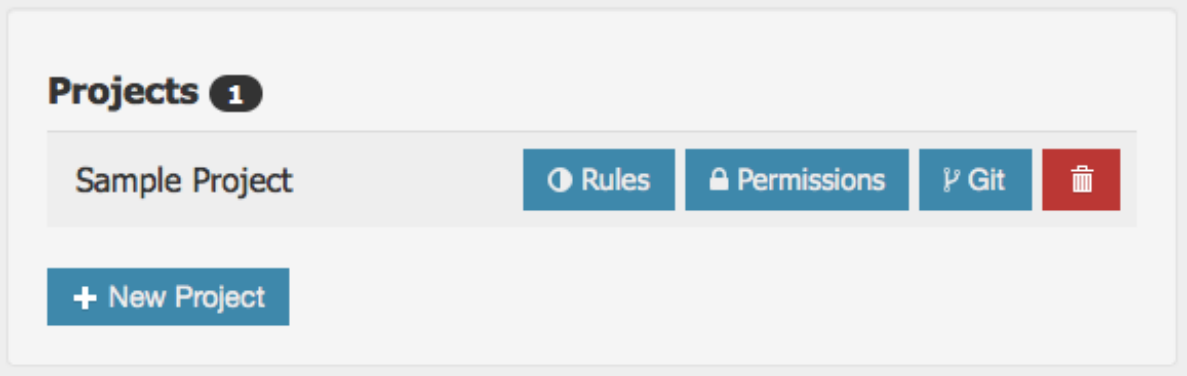


The Admin page is divided into three sections: project management; user management; and API key management.

Projects Administration

The Projects section looks similar to the Projects page. You can create new projects and you will see the *Rules*, *Permissions*, and *Git* buttons. The main differences are:

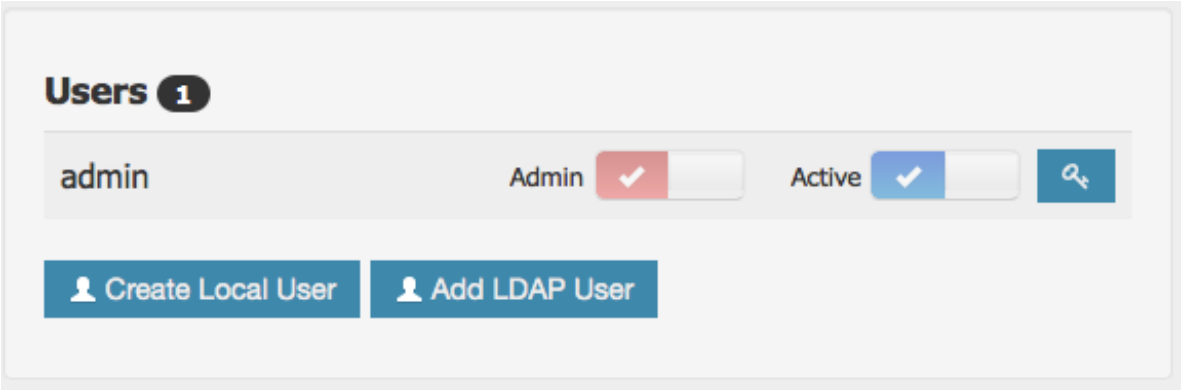
- The Admin page's project list allows you to delete projects; the Projects page does not.
- The Admin page's project list will not display each project's list of analyses; the Projects page does.



For details on the various project configurations please see the Project Management section.

Users Administration

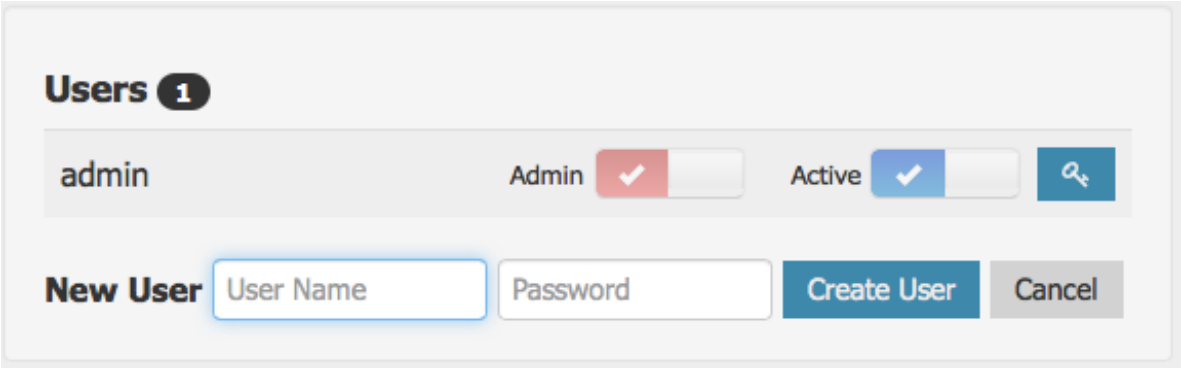
The Users section lets admins add new users, control whether they are admins, and reset passwords. Users cannot be deleted, but they can be marked as *inactive*. The Super User's *admin* and *active* states may not be modified.



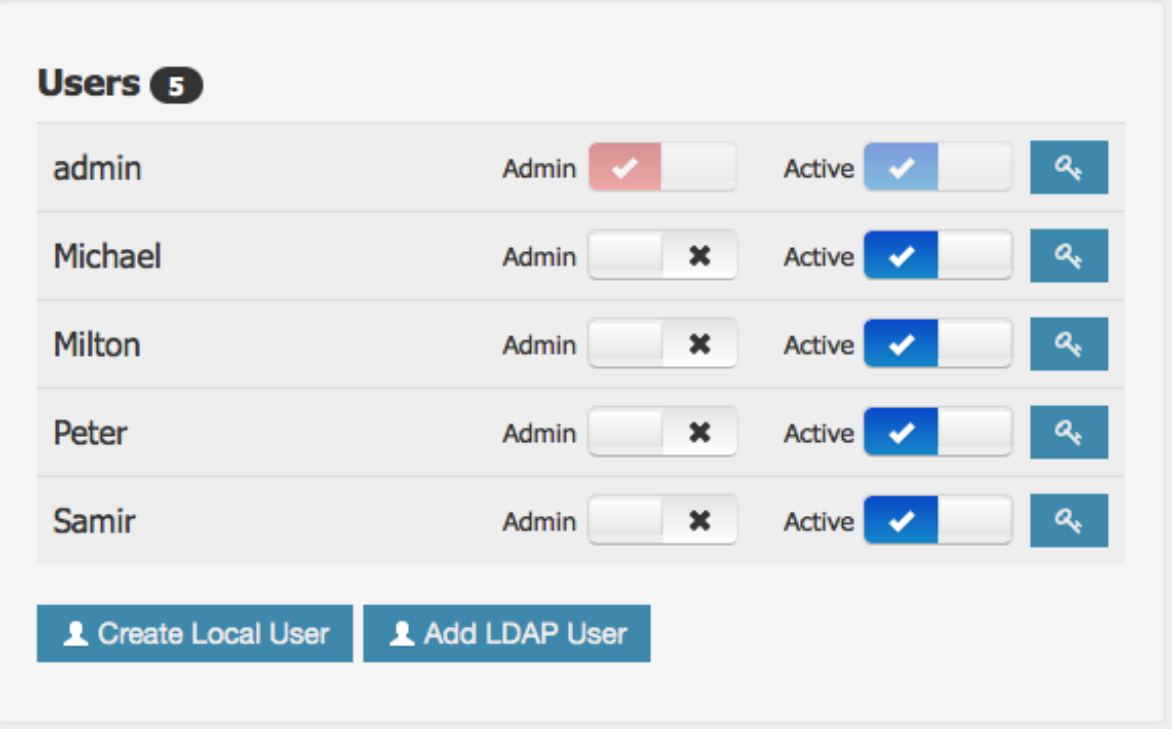
There are two ways to add users to Code Dx:

- **Local Users** exist only within Code Dx. You pick a username and password for them. Code Dx keeps their credentials in its database.
- **LDAP Users** can be added to Code Dx by their username, but their password is managed by an external LDAP server. When an LDAP user logs in, Code Dx will send their credentials to that server in order to authenticate the user.

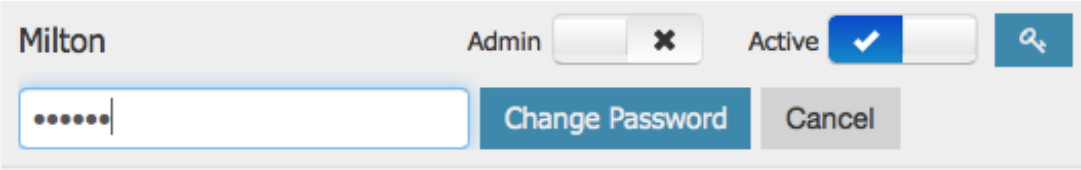
Adding a local user is simple. Just click the *Create Local User* button to open the *New User* form. Enter the name and password for the user you want to create, then click *Create User*.



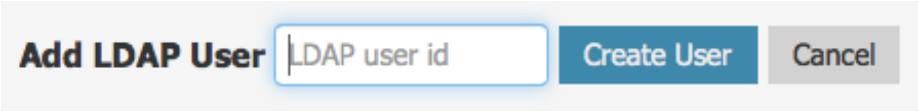
After adding a few more local users, the User List will look like this.



To reset an existing user's password, click on the key icon to the far right and enter in the new user password.

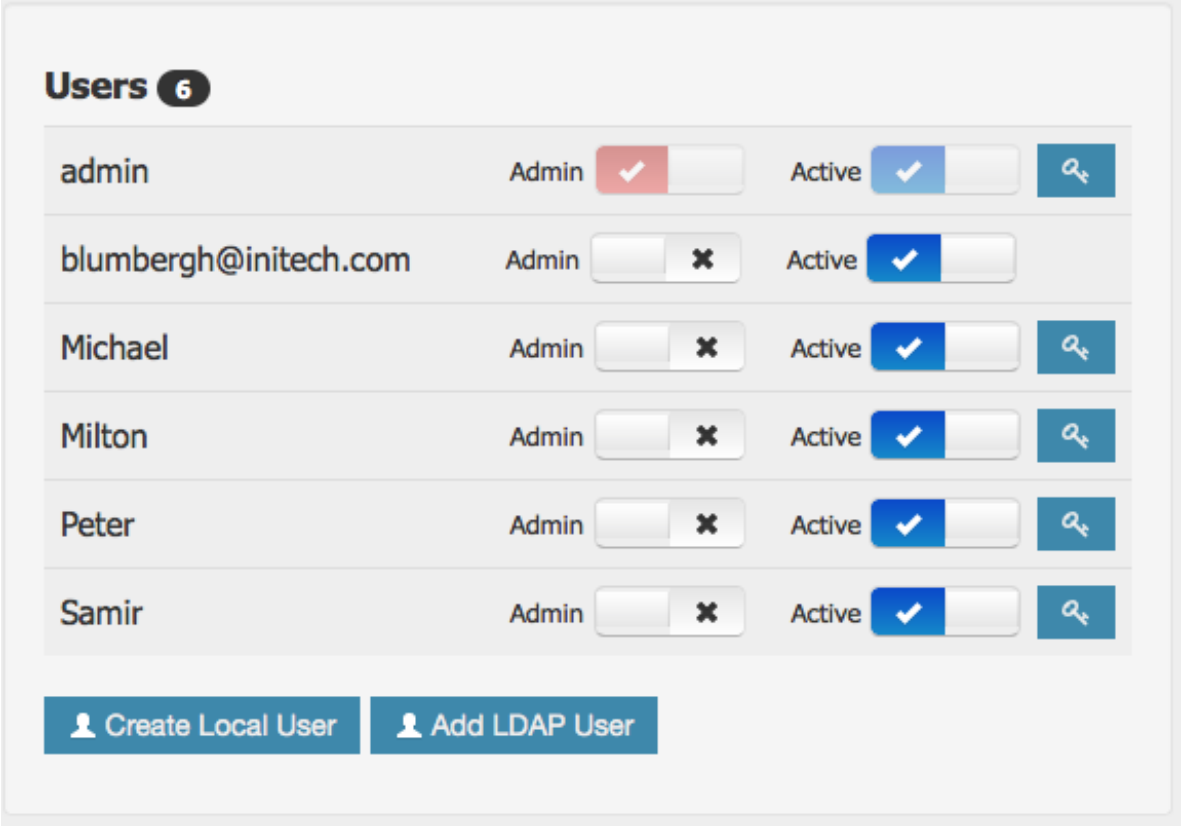


Adding an LDAP user is easy as well (note that you need to have LDAP configured in order to add LDAP users – see the Installation Guide for instructions on how to configure Code Dx for LDAP integration). Just click the *Add LDAP User* button to open its corresponding form.

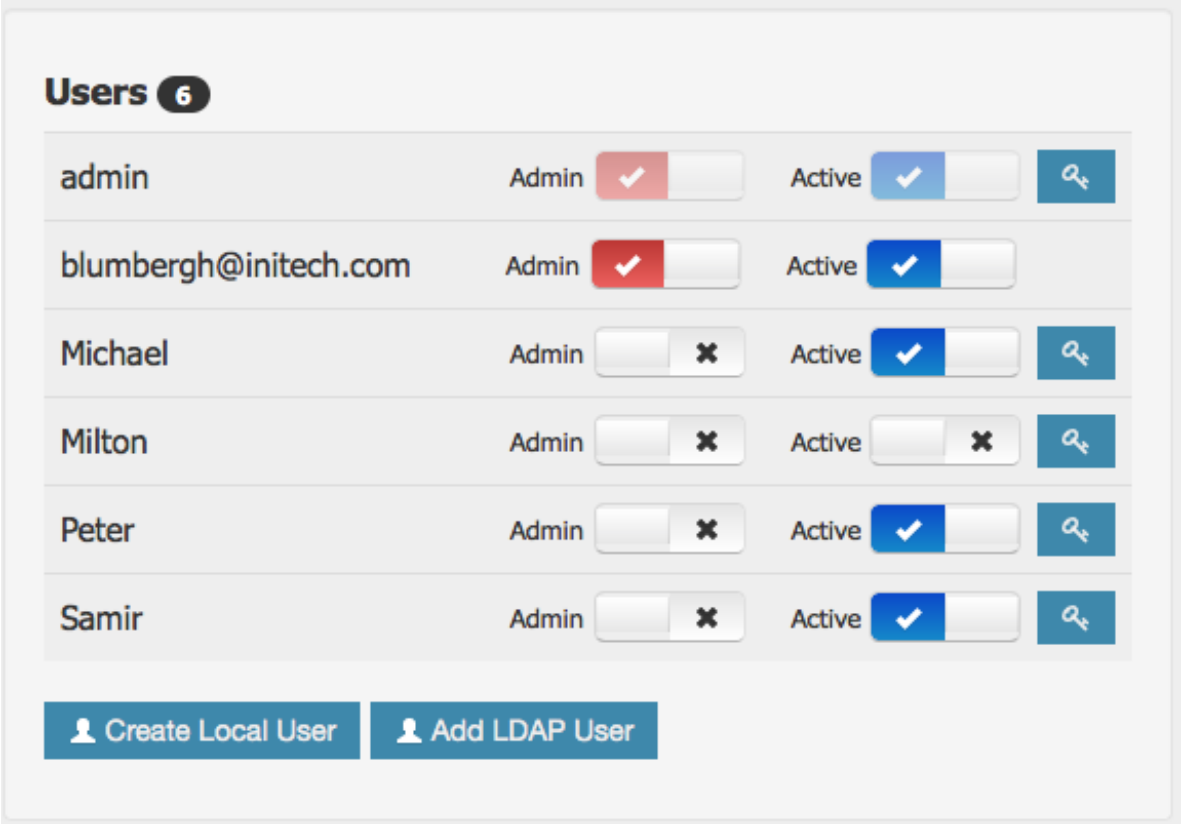


Since the user already exists in your LDAP system, your only job is to let Code Dx know that they exist by adding their "principal" to the Code Dx system. Depending on the LDAP configuration, the principal may be different; for example Bill Lumbergh's username at Initech is blumbergh, so his principal might be blumbergh or blumbergh@initech.com. Once you've added Bill to Code Dx as an LDAP user, he can

log into Code Dx with his Initech password instead of having to remember a new password just for Code Dx.



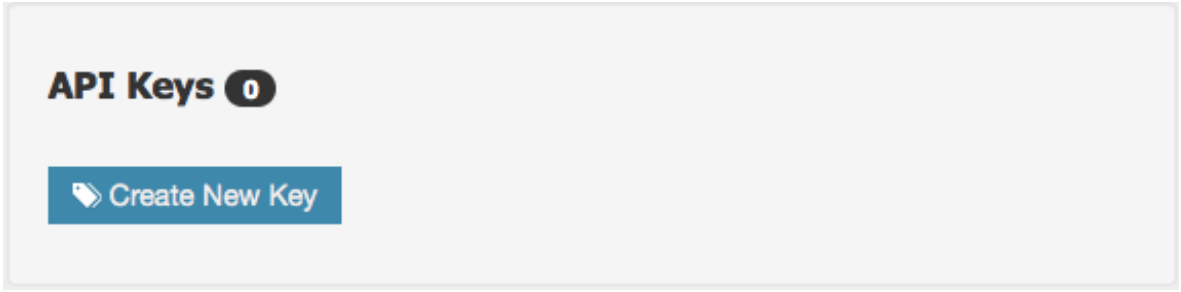
You can easily make any user an *admin* or change whether or not they are *active* with a simple switch.



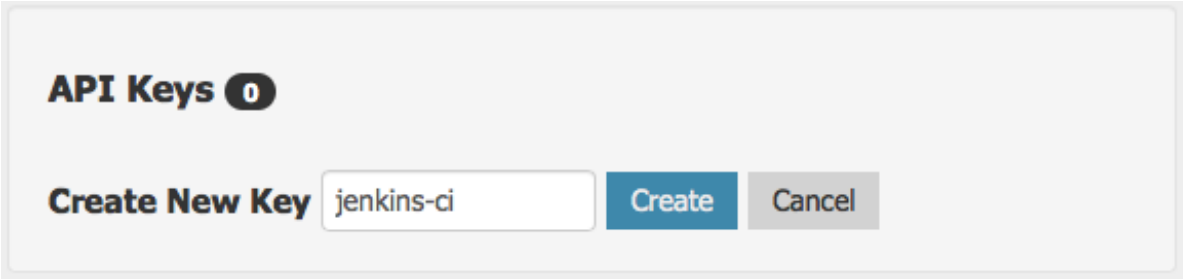
In the screenshot above, Milton has been marked as *inactive*, and blumbergh has been made an admin. Note the column of *Admin* switch. When an *Admin* switch is showing a checkmark with a red background, the corresponding user has administrative privileges within Code Dx. Also note the *Active* buttons. When an *Active* button is showing an *X* with a grey background, the corresponding user is *inactive*. Being *inactive* is like being deleted in that the user cannot log into Code Dx. It is different from being deleted in that any activity performed by that user is still recorded by Code Dx.

API Keys Administration

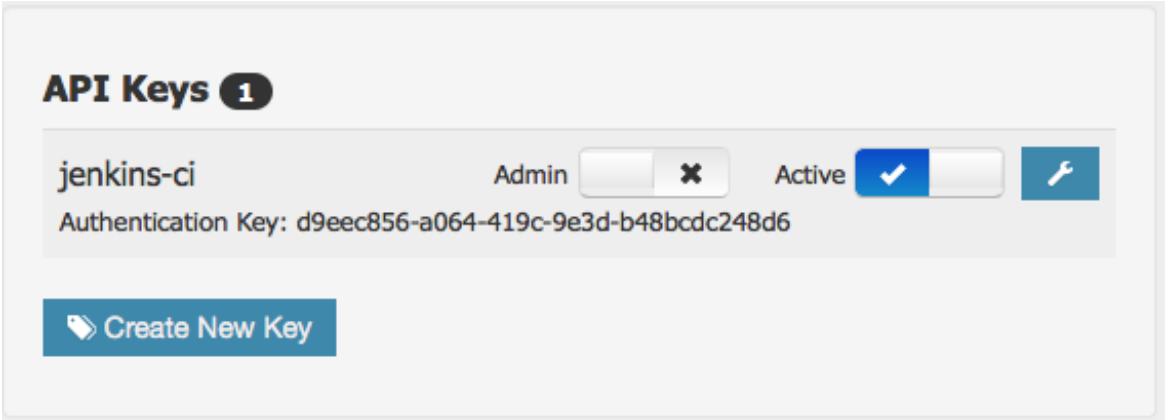
API keys can be generated for use with Code Dx's API authentication. Typically one key would be generated for a specific purpose, such as integrating with a specific tool. This would allow for fine-grained control over each API key's active/inactive state, as well as project permissions to dictate which projects and what permissions each key has access to.



Clicking on the *Create new Key* button will offer up a form to enter in a name for the new API key:



Entering in the new name, and pressing enter or the*Create* button will create the new API key displaying it in the Key listing:



The key can be regenerated at any point in time by clicking on the wrench icon.

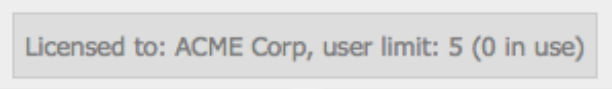
Managing permissions for each API key is done from the[project permissions management](#) just as with regular users.

For more information on Code Dx API capabilities, please read the[Code Dx API Guide](#).

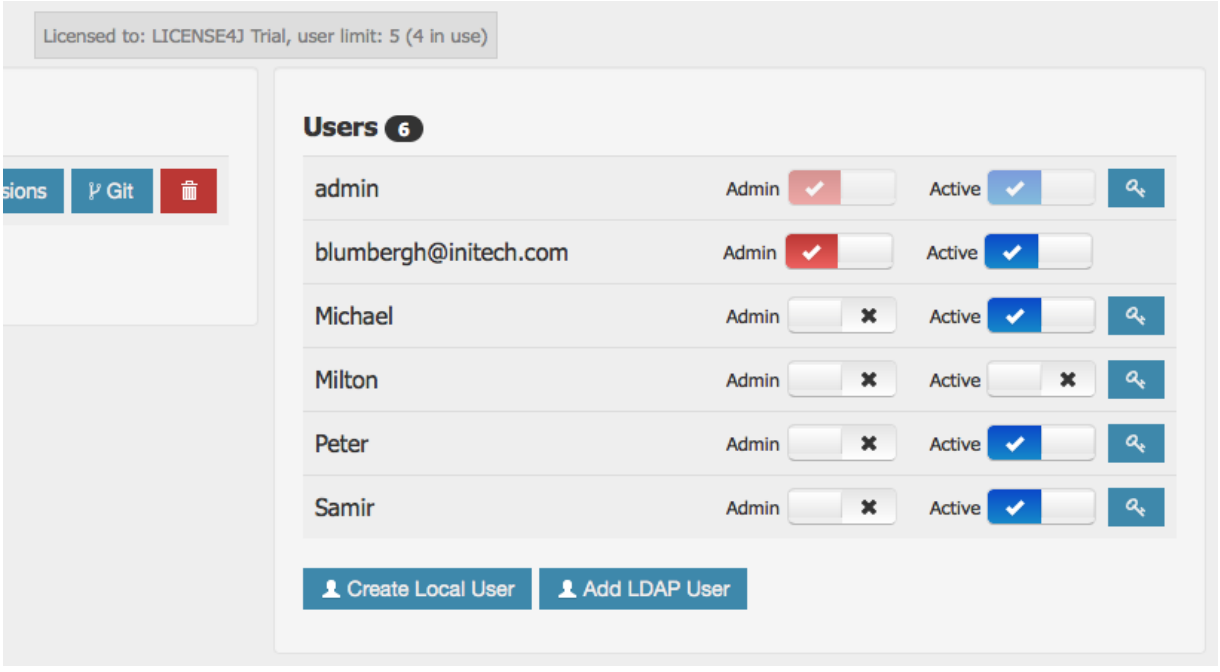
License Management

Code Dx requires a valid license to run. This license will be provided to you when you get the download instructions for Code Dx. This will be in the form of a file with a .lic extension that needs to be placed in the Code Dx configuration directory. The Install Guide has additional information on where to place the license file so Code Dx recognizes it.

The summary information for the currently active license is always displayed at the top of the Admin page:



Depending on the type of license you received, it may have a user-count restriction. This restriction is on the number of active user accounts managed by Code Dx, regardless of whether they're Code Dx local users or LDAP users. A license is not tied to a named user and the system admin user does not count against the number of licensed users. So in the example we've created so far we can see that Peter, Michael, Samir, and blumbergh@initech.com all count against the license count. Milton is marked as an inactive user and therefore does not end up using an active user license:



If the system reaches the limit of the licensed user count, an error message will be displayed when creating new users. This can be remedied by inactivating users that no longer have a need to sign in and use Code Dx. Alternatively, arrangements can

be made with Code Dx to upgrade and replace the current active license with one that has a larger user-limit.

Project Management

Terminology

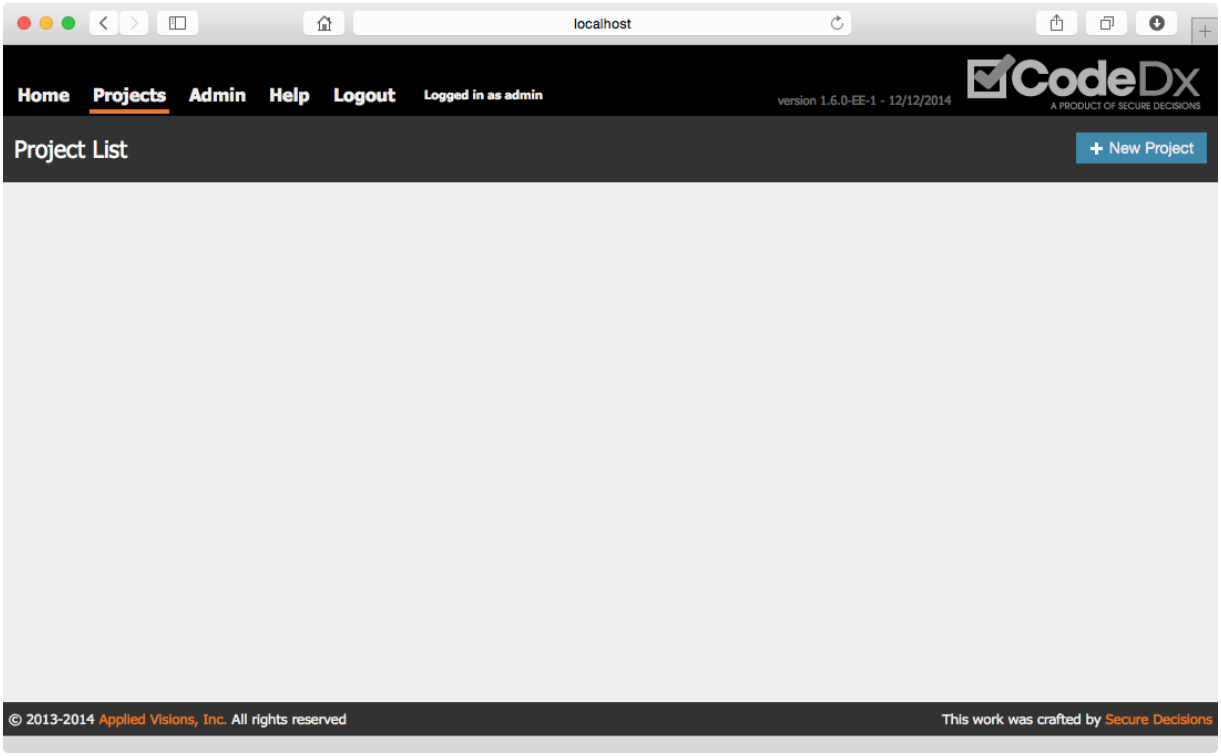
The following are the key terms used throughout Code Dx and this guide.

- **Project**: a collection of scans over time for a target software.
- **Analysis Run**: a correlated set of scans conducted by one or more tools to identify potential vulnerabilities within a specific snapshot of the target software.
- **Weakness**: a finding reported by a tool. Until a manual review process has occurred, these findings are identified as *potential vulnerabilities* and therefore referred to as *weaknesses* within Code Dx.

Projects are composed of any number of *Analysis Runs*, which are in turn composed of any number of *Weaknesses*.

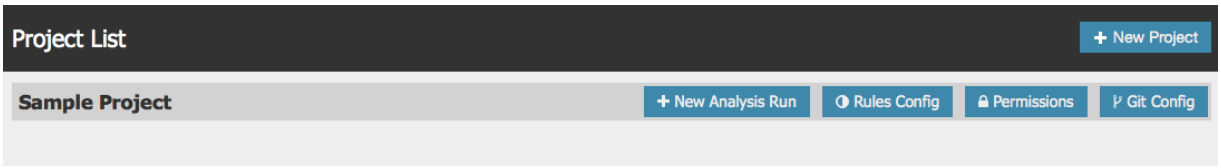
Project Lifecycle

The Projects page presents a list of projects, each with a list of their respective analysis runs. To access the Projects page, just click the *Projects* link in the page header after logging in. If this is the first time using Code Dx, the *Project List* may be empty. Users with admin privileges can create new projects by clicking on the button labeled *New Project*.



Click the button to open the *New Project* form.

Create a new project by entering a name for it and clicking the*Create Project* button. The new project should appear in the project list.

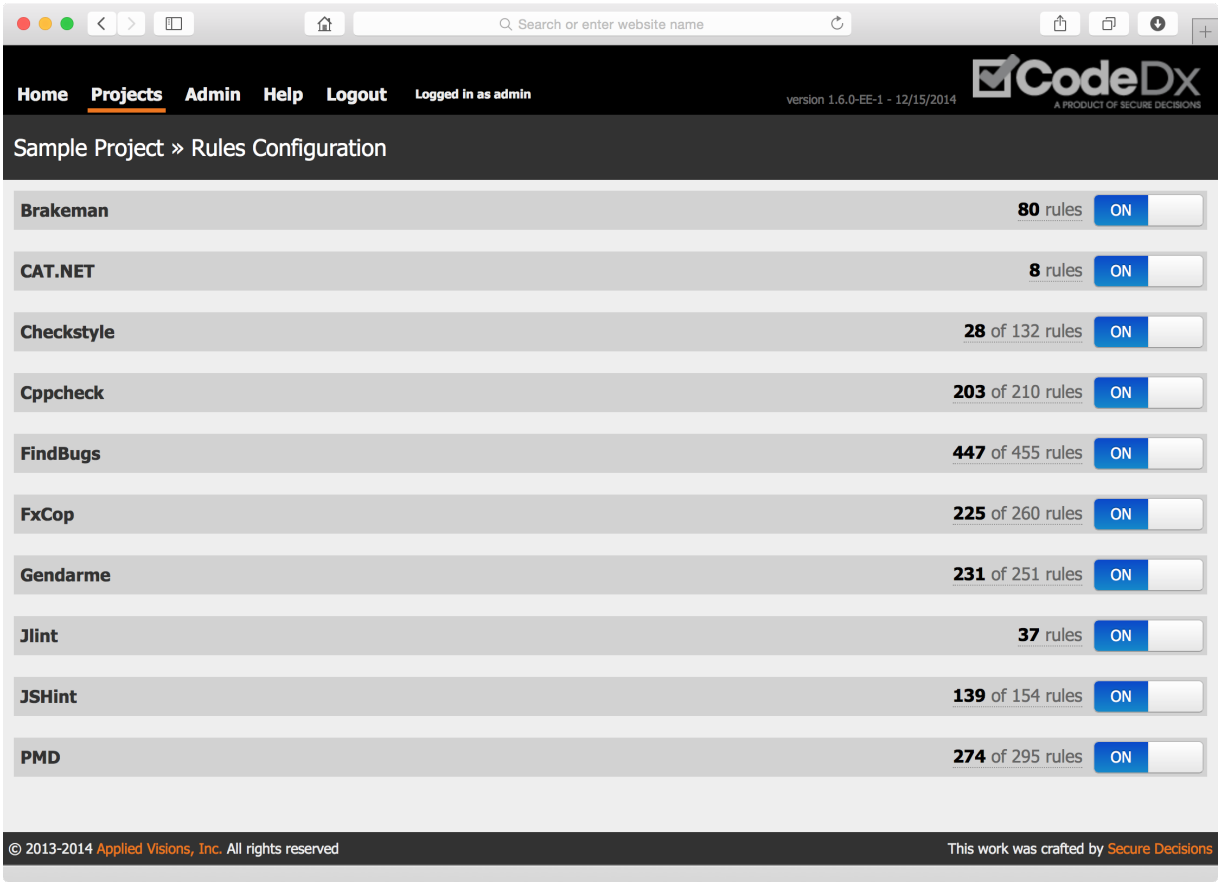


Once a project is created it is recommended to assign one or more users to it and give them the manage permissions. This enables them to create and delete analyses for any project, manage the rules configuration for their projects, and manage permissions for users assigned to their projects.

Rules Configuration

Each project has the ability to define which Rules will be enabled or disabled for its analyses. Users with manage permissions on the project will be allowed to modify

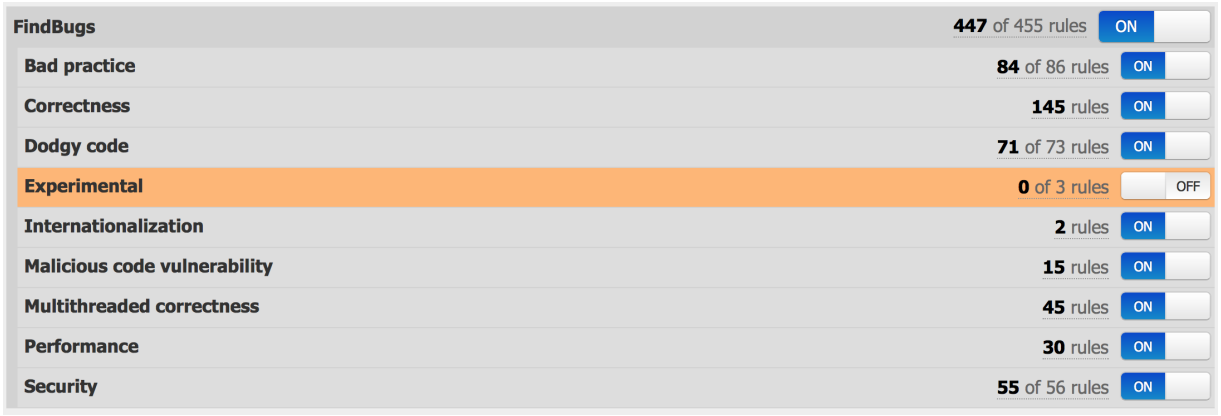
the rule configuration. Clicking on the *Rules Config* button from the Projects page, or *Rules* button from the Admin page will lead you to the project specific Rules Configuration page.



As you upload tool results to create analysis runs, Code Dx will show the corresponding rules for those tools in the Rules Configuration page. Rules are organized in a hierarchy that is grouped by the tools running the checks. Each tool can be disabled entirely, or expanded by clicking on it, to reveal the groups, sub-groups, and individual rules. Each level in the rule hierarchy can be enabled or disabled as a whole. Certain groups or rules will be disabled by default. The default enabled state is carefully selected by Code Dx to provide the best results for the Code Dx users. However, this can be overridden at any time from this page by just re-enabling the desired rules.

Note that any changes made on this page are project-wide, impacting all users of the project.

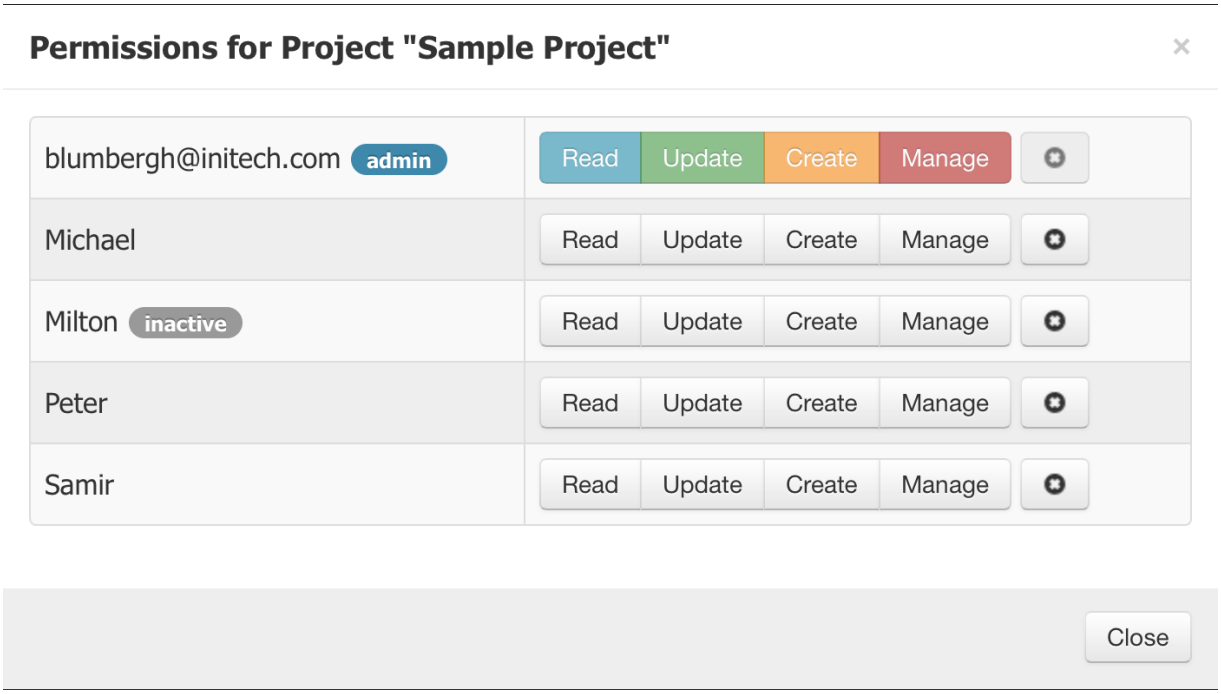
For instance, the following screenshot shows the *Experimental* group within *Findbugs* disabled by default.



Code Dx uses the enabled state of rules when accepting new data files. During new analyses, if the rule for a given weakness is disabled, it will be rejected and won't be added to the list of weaknesses for the analysis run in question. In addition, while disabling rules within the Rules Configuration page, a purge option will be displayed, when applicable, to remove existing weaknesses for this project that match the newly disabled rules. Unless this is a temporary change for experimentation, it is highly recommended to purge these weaknesses to improve the performance and responsiveness of Code Dx.

Permissions Configuration

To manage the permissions for a project click the *Permissions* button on the Projects page.



The Permission Management popup will appear. In this view, there is a row for each user. Each button represents a tier of permissions which that user has in that project. All permissions are per-user, per-project, meaning that a user's permissions for one project are not necessarily the same as his or her permissions for another project. For each user, if they are marked as *admin* or *inactive*, the view will display a marker next to their name to show that fact.

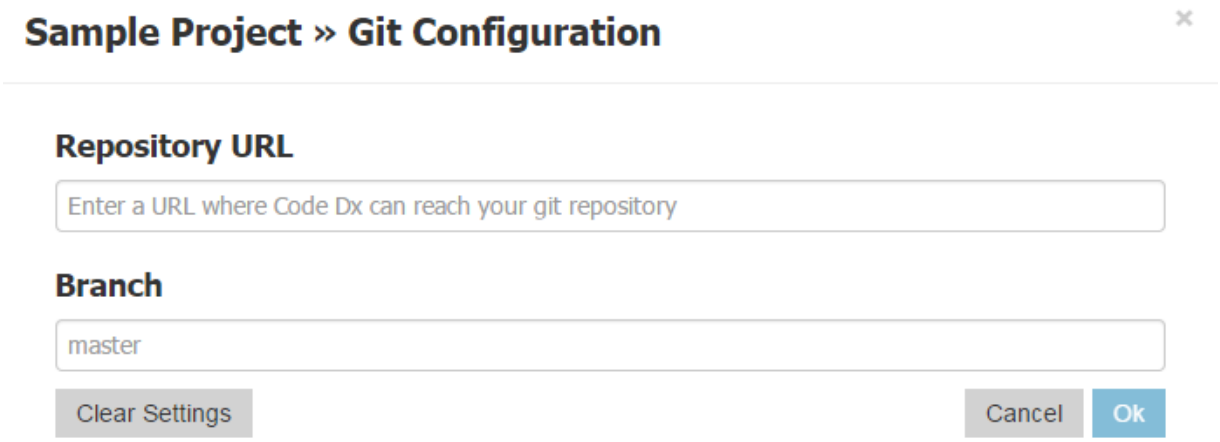
The different tiers of permissions are as follows:

- *Read* means that a user can see a project and all of its contents. If a user doesn't have *Read* permissions for a project, that project won't even appear in the Projects page for that user.
- *Update* means that a user can change the status of weaknesses in a project
- *Create* means that a user can create new analyses in a project
- *Manage* means that a user can manage the project configuration (rules, permissions, and git) within a project, as well as delete analysis runs in that project.

Clicking one of the permissions buttons in the Permission Management popup will give the corresponding user all permissions up to that tier. For example, giving a user *Create* permissions will also give him or her*Read* and *Update* permissions. Clicking the *X* button will clear all permissions for that user. Admin users automatically have all permissions; you cannot give or take away permissions for admin users.

Git Configuration

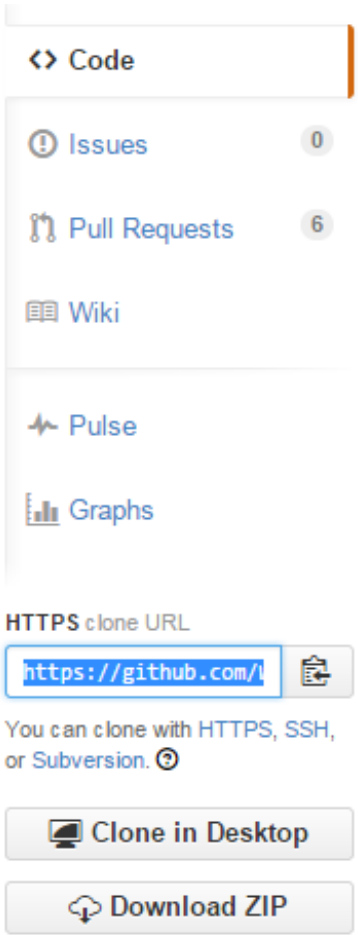
To manage the git configuration for a project, click the *Git Config* button on the Projects page, or the *Git* button in the Projects list on the Admin page.

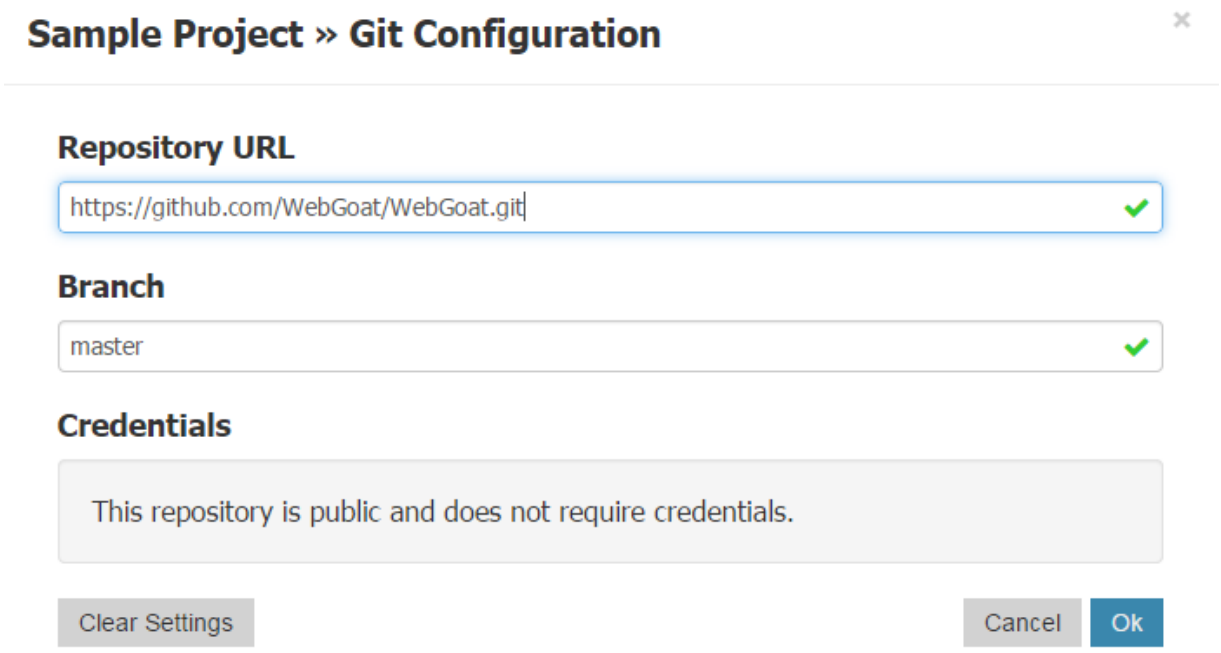


The *Git Configuration* popup will appear. The form inside is used to tell Code Dx to use a Git repository as the subject of analysis for this project. Once configured, Code Dx will automatically include the contents of the configured repository as an input for each analysis with this project.

The form (shown above) has two fields: *Repository URL* and *Branch*. The *Repository URL* should be filled out with the URL that you would use to clone the repository. The *Branch* field should be filled with the name of the branch in that repository that you want Code Dx to analyze. If left blank, Code Dx will assume you mean the "master" branch, which is the main branch for most Git repositories.

For many projects, setting up a Git configuration is as easy as copying the repository's URL into the form. For example, if you wanted to analyze the contents of the open-source WebGoat repository, you would find the clone URL on the side of the GitHub repository page, and copy it into the *Repository URL* field of the *Git Configuration* form.


Code
Issues 0
Pull Requests 6
Wiki
Pulse
Graphs
HTTPS clone URL
https://github.com/
You can clone with HTTPS, SSH, or Subversion.
Clone in Desktop
Download ZIP


Sample Project » Git Configuration
Repository URL
https://github.com/WebGoat/WebGoat.git
Branch
master
Credentials
This repository is public and does not require credentials.
Clear Settings
Cancel
Ok

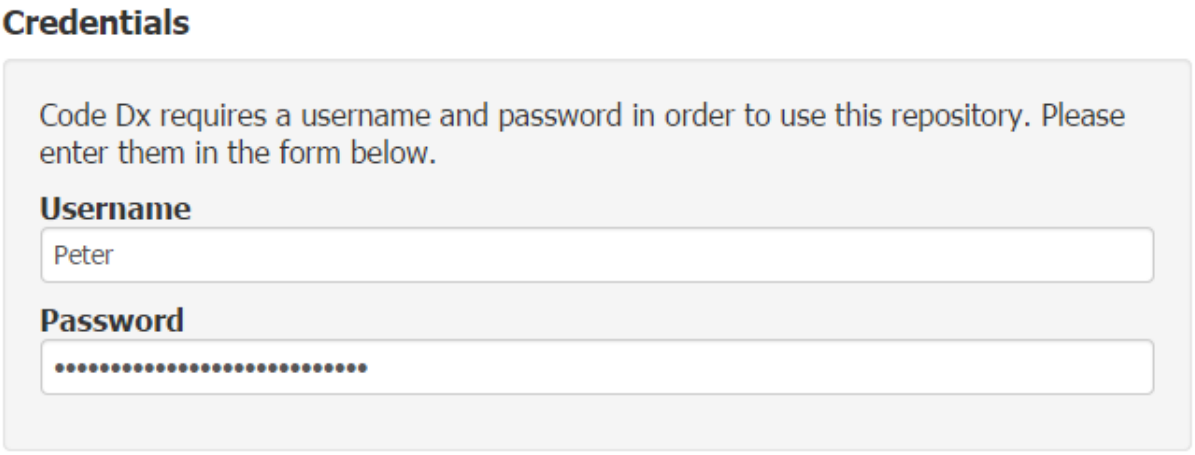
Code Dx will verify the repository's existence and determine whether it needs credentials to connect. For public (open-source) repositories, no credentials are required, and you can press the *Ok* button to save and close the form. If this is the case, you may skip to Saving the Git Configuration; otherwise, read on.

Git Credentials

Some Git repositories are private, and require credentials for access. Code Dx supports two forms of authentication; HTTP and SSH. Depending on the URL in the *Repository URL* field, Code Dx will automatically determine which type of credentials are required.

HTTP Credentials

HTTP credientals are a *username* and *password*. For GitHub repositories, these will generally be your GitHub account name and password. GitHub also supports creating "Personal access tokens" (see https://github.com/settings/applications), which can be used in place of a password.



SSH Credentials

SSH uses a pair of files known together as a "keypair", or separately as a "private key" and "public key". For Code Dx to connect to a repository via SSH, it needs your "private key". The system in charge of the repository's security will also need your "public key".

If you are trying to access a private GitHub repository, visit your SSH Keys page at https://github.com/settings/ssh to register your SSH key with GitHub. GitHub also

provides help with SSH-related issues at https://help.github.com/categories/ssh/

Some users will already have an SSH keypair on their computer. The two files are generally located in <userhome>/.ssh/ and will be named id_rsa for the private key, and id_rsa.pub for the public key. It is possible to use this pair, but you may want to generate a separate pair for use with Code Dx.

Once you have located or generated a keypair, copy the contents of the private key file into the *Private Key* field of the form.

When generating a keypair, you have the option to provide a "passphrase" for the private key. If you do this, Code Dx will need that passphrase in order to use your private key. Enter it in the *Key Passphrase* field of the form.

Saving the Git Configuration

Once you have entered a *URL*, an optional *Branch*, and entered whatever *Credentials* are necessary, you can click the *Ok* button to save the configuration. Doing so will close the form and tell Code Dx to obtain a local clone of the configured repository. Depending on the size of the repository, the length of its history, and your network connection, the clone operation may take anywhere from seconds to hours. Once started, a progress bar will be displayed underneath the project's title in the Projects page.

The "cloning" job has several subtasks, so you will see the progress bar fill up several times. When the job is complete, the progress bar will turn blue, wait for a couple of seconds, then slide out of view.

Once the clone is ready, the New Analysis page will automatically include the latest contents of the configured branch of the configured repository as an input. See the Analyses section for more detail.

Analyses

This section explains the analysis capabilities of Code Dx. All editions of Code Dx come with built-in tools to scan the applications of interest to you. The languages we support and expected inputs for the built-in scanners are described in the *Built-in Code Scanners* and the *Built-in Dependency Scanners* sections. In addition to the built-in tools, the Enterprise Edition of Code Dx can import the results of several commercial and open source tools. The supported tools and generic input formats are described in the *Importing Scan Results* section. There are a number of different options to configure and run analyses for Code Dx: manually using the web interface; from the IDE or Jenkins plugins; automatically (such as from your continous integration server) using the API. These are all detailed in the *Starting Analyses* section.

Built-in Code Scanners

Code Dx analyzes C/C++, Java, .NET, Ruby on Rails, Python, and Javascript applications. For all supported languages, Code Dx will analyze the source using bundled tools built specifically for a target language. For applications built with any combination of the supported languages, Code Dx will run the appropriate checkers on the provided source.

For Java applications, Code Dx supports scanning compiled bytecode. In fact, the preferred approach for Java projects is to upload **both** source and bytecode to Code Dx. This yields the best coverage for issue detection.

For .NET applications, Code Dx supports scanning compiled DLLs. It is also recommended that the source be uploaded. This will provide better source location information and will allow for viewing the source while looking at weakness details. **Note:** If you choose to upload an entire Visual Studio solution folder, there may be duplicates of the build DLLs and 3rd party DLLs. This will cause a longer analysis time and possibly incorrect results if some DLLs are stale. To achieve the best results, upload a zip that contains only the DLLs and PDB files for the binaries you wish to analyze. Upload the source as a separate zip.

Code Dx accepts application inputs in the following formats:

- **C/C++ source zip archives** – zip files containing C/C++ source files that will be analyzed by Code Dx's bundled tools. Code Dx will scan the contents of the zip file for any .h, .c, .hpp, and .cpp files.
- **Java source zip archives** – zip archives containing Java source files – with a .java extension – to be analyzed by Code Dx's bundled tools.
- **Java bytecode zip archives** – zip archives containing .class or .jar bytecode files intended for the JVM.
- **.NET source zip archives** – zip archives containing C# or VB.NET source files – with a .cs or .vb extension.
- **.NET DLLs** – zip archives containing compiled DLLs. You must also include the PDB files for DLLs you wish to scan. Code Dx will only scan DLLs with corresponding PDB files – unless there are no PDB files, in which case Code Dx will scan all DLLs but source location information may be sub-optimal.
- **Ruby on Rails archives** – zip archives containing Ruby source files that are inside an app/ directory.
- **Python zip archives** - zip archives containing Python source files.
- **Javascript zip archives** – zip archives containing .js files.

Note that **Code Dx enforces a single source zip archive per analysis run** So even though Code Dx supports multiple languages, the expectation is that they will all be packaged in a single zip archive to enable consistent path correlation across all the checkers. Although source and bytecode inputs can be uploaded in separate files, they do not have be split up. A single zip file containing C/C++ source, Java source, Java bytecode, .NET DLLs, .NET source, Ruby on Rails source, Python Source, and Javascript source is perfectly acceptable.

Built-in Dependency Scanners

Code Dx also scans input to check for dependencies with known vulnerabilities.

The following are checked:

- **Java** - in Java projects, .jar and .war files
- **.NET** - in .NET projects, .exe and .dll files
- **JavaScript** - JavaScript files are checked by name or a hash of the file (minified JavaScript incorporated into a different source file will not be checked)

Importing Scan Results

The Enterprise Edition of Code Dx supports importing the results of 20+ commercial and open source static analysis tools, in addition to a couple of generic weakness listing formats. The list of supported tools for scan imports includes the built-in ones mentioned in the previous section. If one of the tools you want to import is not supported, please let us know. However, in the meantime you can convert your data to the generic *Code Dx Input XML* format. The schema definition for this format and a sample file are included in the download you received for Code Dx.

The following is the list of supported tools and import formats supported by the Enterprise Edition of Code Dx:

- **AppScan XML output** – Code Dx supports AppScan outputs in .xml.
- **Brakeman JSON output** - Brakeman is one of the built-in scanners, but if run externally, its .json outputs are accepted by Code Dx.
- **CAT.NET XML output** – CAT.NET .xml outputs are accepted by Code Dx.
- **Checkmarx XML output** - Checkmarx reports in xml format are accepted by Code Dx.
- **Checkstyle XML output** - .xml output from Checkstyle is accepted by Code Dx.
- **Clang HTML output** – Code Dx supports HTML output from Clang but expects it in a .zip archive since Clang outputs one HTML file per checked source file.
- **CodeSecure XML outputs** – Armorize's Code Secure .xml outputs are processed by Code Dx.
- **CodeSonar XML outputs** – there are different options and certain caveats to the CodeSonar outputs, so please read the *CodeSonar Support* section for the details.
- **Code Dx XML format** – for cases where you have data from a custom tool or from a tool that isn't supported by Code Dx, you can convert the output to the

Code Dx .xml format and input that directly for analysis. The schema and sample output file for the Code Dx format is supplied to you with the installation files.

- **CppCheck XML v2 output** – Code Dx supports the v2.xml output from CppCheck.
- **Coverity JSON output** – Code Dx supports .json formatted output from Coverity using their 'cov-commit-defect' command line tool. For example: cov-commit-defect --preview-report /<outputpath>/results.json
- **Dependency Check** - Code Dx supports Dependency Check outputs in.xml
- **error-prone ouput** – raw plain-text error-prone output is accepted by Code Dx, such as in .txt files
- **FindBugs XML output** – although Code Dx includes FindBugs as a built-in scanner, it will accept raw .xml FindBugs outputs.
- **Fortify FPR files** – Code Dx will process the analysis results detected by Fortify and stored in .fpr files.
- **FxCop XML output** – just like with other built-in tools, raw.xml FxCop outputs are accepted by Code Dx.
- **Gendarme XML output** – same as above, raw Gendarme .xml outputs are accepted by Code Dx.
- **JLint output** – Code Dx processes the raw output from JLint and expects it in a plain text format, such as in .txt files
- **JSHint output** – raw JSHint output is accepted by Code Dx and is expected in plain text format, such as .txt files
- **OCLint output** – Code Dx accepts .xml output files generated by OCLint
- **Parasoft JTest/C++Test/dotTest XML output** – Code Dx accepts .xml outputs for these three Parasoft tools
- **Pylint** - Code Dx supports Pylint .json output
- **PMD XML output** – same as with other built-in tools, raw.xml PMD results are accepted by Code Dx.
- **Retire.js JSON output** - Retire.js is a built-in scanner, but if run externally, its output in JSON format is accepted by Code Dx.
- **SATE XML format** – Code Dx supports the .xml format for NIST's Static Analysis Tool Exposition V (SATE V).
- **Other source zip archives** – in addition to source file types supported by the Standard Edition, Code Dx will accept any zip archive as source input. While the source itself isn't scanned, its contents are searched for matching files to the weakness reported ones.

CodeSonar Support

There are two ways in which CodeSonar results can be exported for use in Code Dx. One way is to use CodeSonar-Scrape, a tool created and maintained by the Code Dx team. This tool will automatically scrape all of the content from a CodeSonar analysis and save the data to a zip file. The zip file can be uploaded directly to Code Dx. It will include descriptions (tracing) information and may also include links back to Code Sonar findings in the hub, and documentation. Documentation for this tool can be found in the CodeSonar-Scrape User Guide. If you need CodeSonar-Scrape or have questions on the topic please contact us.

The other way to export the data is to click the “XML” link on the main analysis page in CodeSonar. The following table columns must be enabled before doing so:

1. ID
2. Class
3. Rank
4. File Path
5. Link Number
6. Categories

This XML file can be directly imported into Code Dx. It should be noted that using this method will not result in having CodeSonar descriptions, hub links, or documentation links in Code Dx.

Starting Analyses

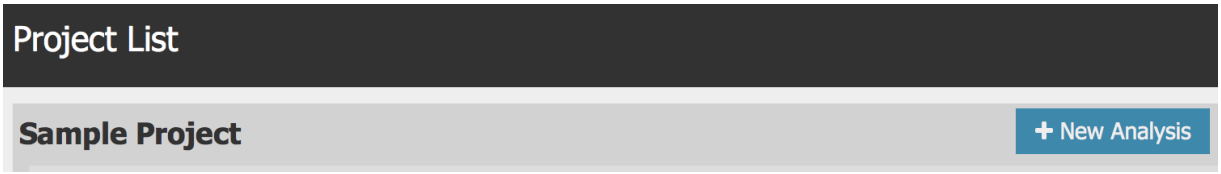
There are a number of different ways to prepare and initiate an analysis within Code Dx:

- Manually from the web interface
- Manually from the IDE plugins
- Automatically using the API

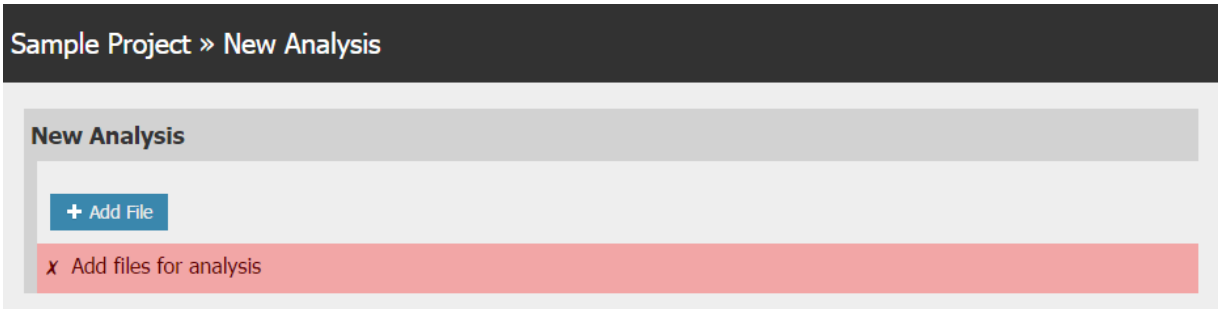
Note that only users with the create permission for projects can initiate new analyses.

Starting Analyses Manually from the Web Interface

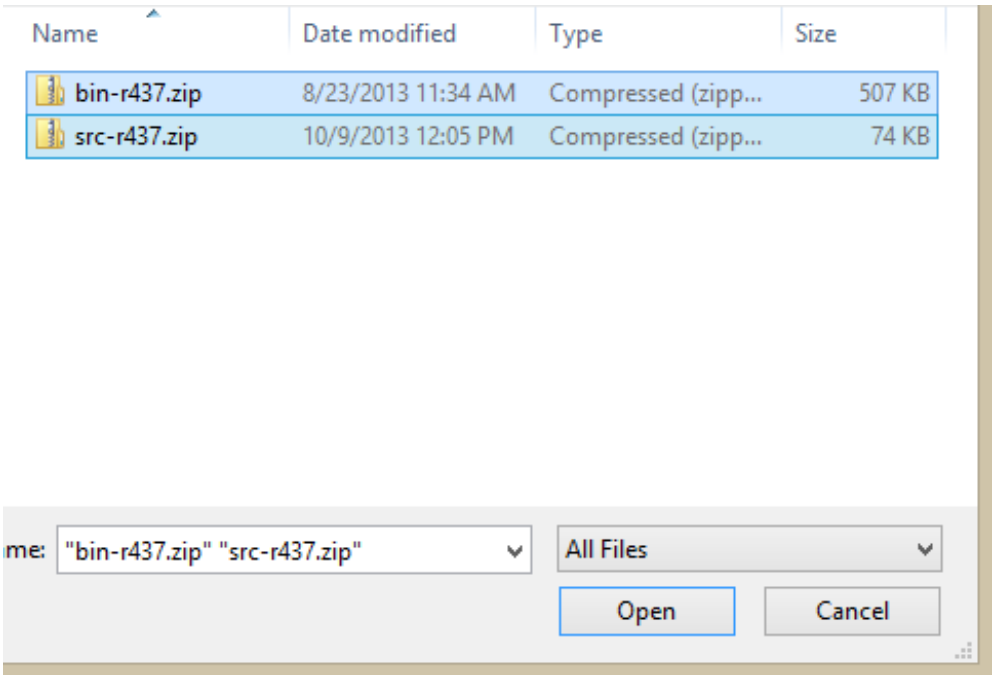
Analyses can be prepared and initiated manually from the Code Dx web interface. To do so, the first step is to go to the Projects page, find the project that you want to run the analysis for, and click the *New Analysis* button.



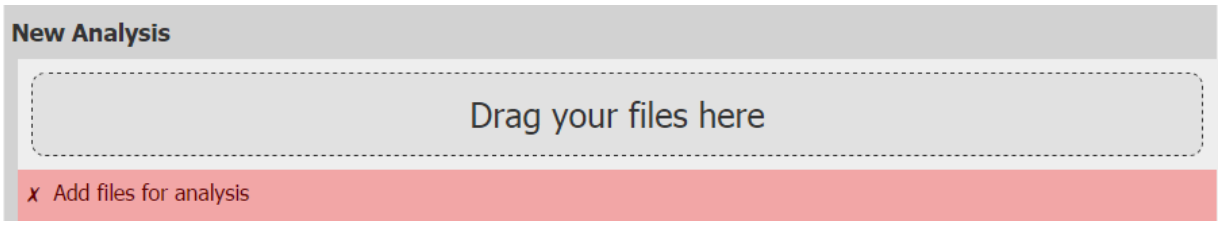
This will take you to the New Analysis page.



To add a file to the page, you can use the*Add File* button. A file picker dialog will open and you may select one or more files, as is shown in the next image.

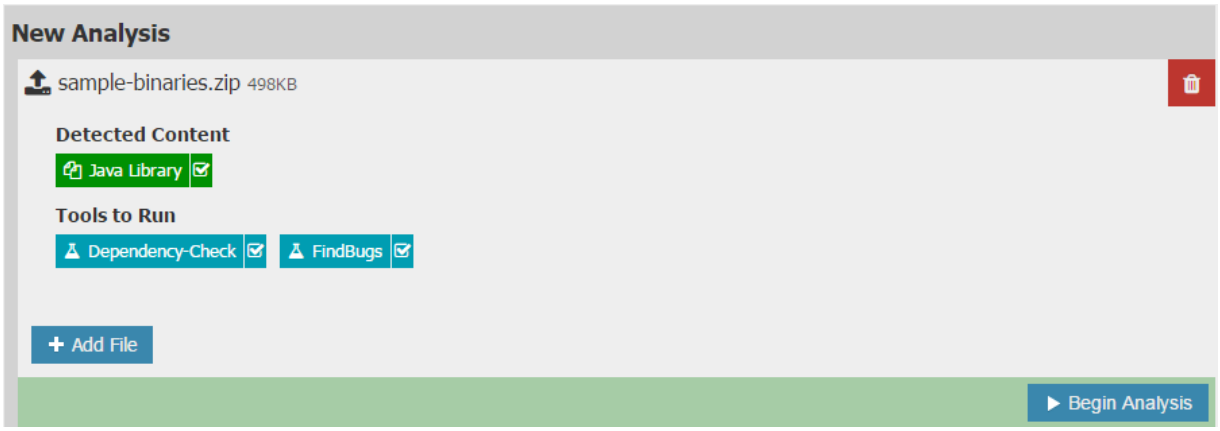


Alternatively you can drag the files over the same button area. When dragging and dropping, the page will change to display the drop region:

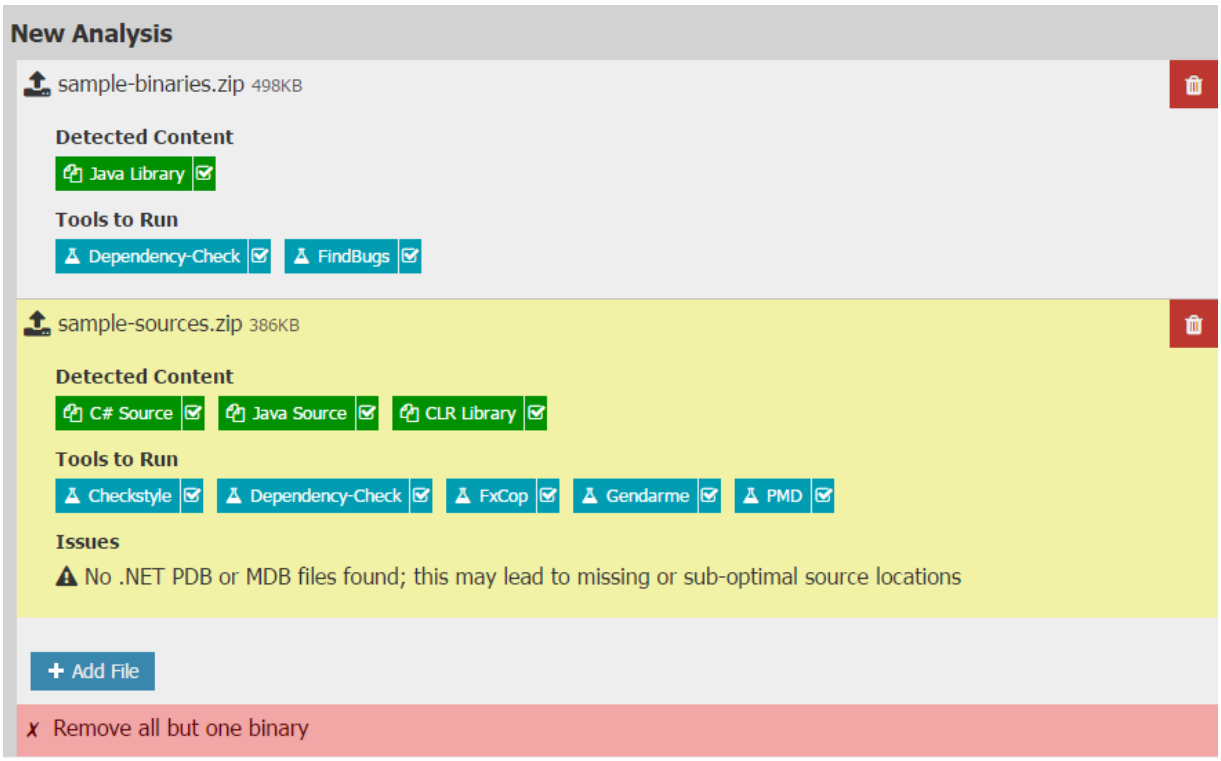


Please note that the drag-and-drop functionality is not supported by all browsers.

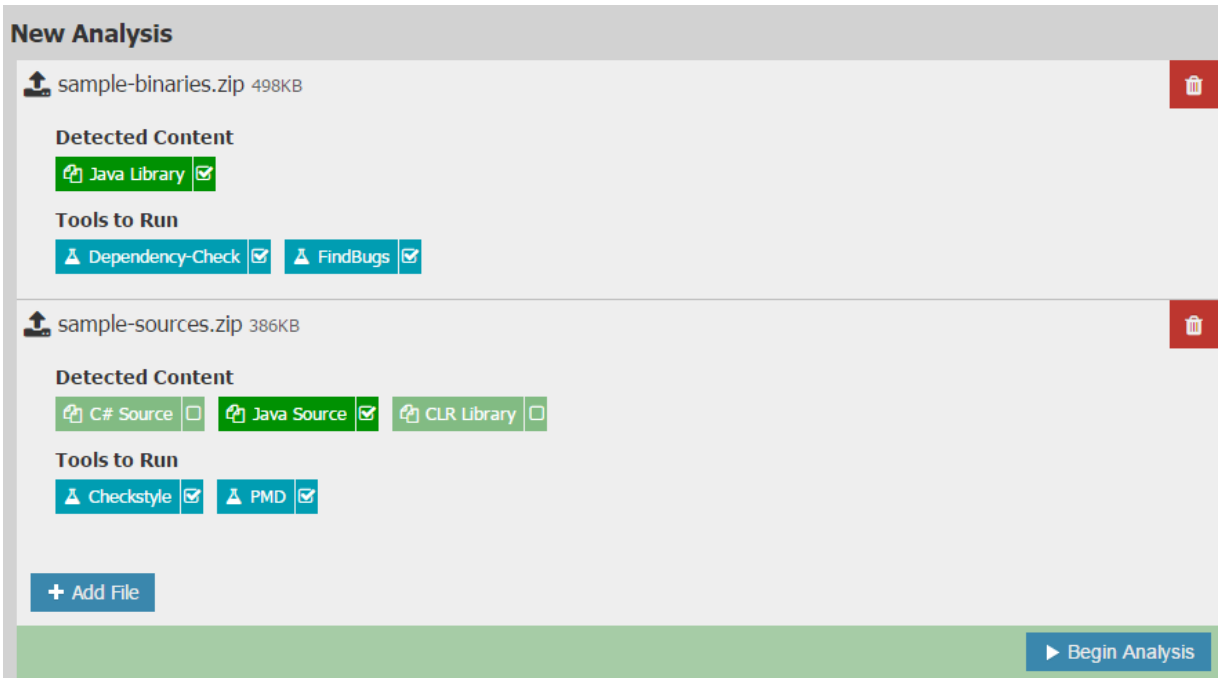
As you add files to the page, they will be uploaded to the Code Dx server for identification. Once the server has identified the file contents, the page will update to display the detected content along with any errors or warnings about the contents.



In the image above, a zip file containing Java.class files was added, and tagged as a *Java Library*. Based on this content, Code Dx identified *Dependency-Check* and *FindBugs* as the tools to run on this file. Each tag in the *Detected Content* and *Tools to Run* sections can be disabled. If desired, click the checkbox on the tag to disable (or re-enable) that tag. Sometimes, disabling a content tag will make Code Dx decide that a certain tool is no longer applicable to that file. Disabling a tag in the *Tools to Run* section will tell Code Dx not to run that tool, even though it is applicable to that file.



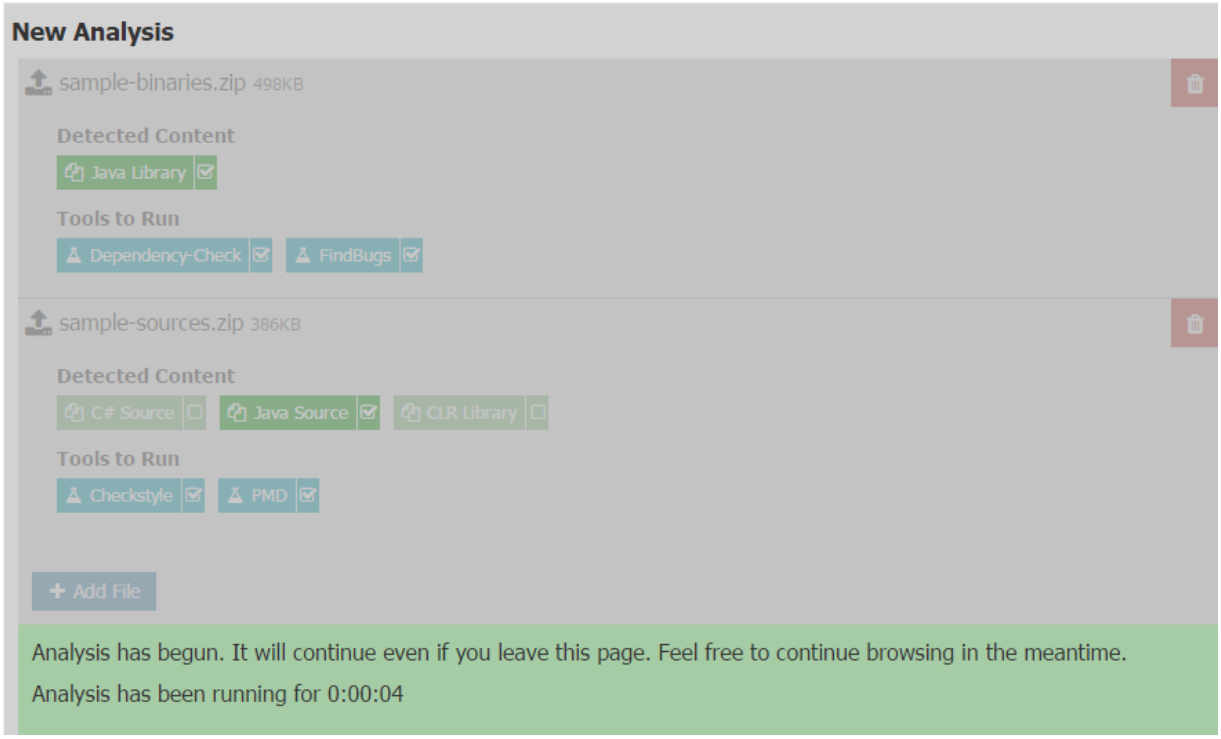
In the image above, a second zip file was added, containing.java files as well as some C# source files and .NET (CLR) compilation artifacts. The file was tagged as *C# Source*, *Java Source*, and *CLR Binary*. Code Dx identified five different tools to run on that file. Additionally, since both files have been tagged as a "Library", Code Dx won't allow an analysis. This can be solved by disabling the *CLR Library* tag on the new file. In this example, since we are only interested in the Java-related contents of the project, we disable the *C# Source* tag as well.



With the two tags unchecked, the warnings and tools that were only applicable to those tags have disappeared, and Code Dx will once again allow analysis to start.

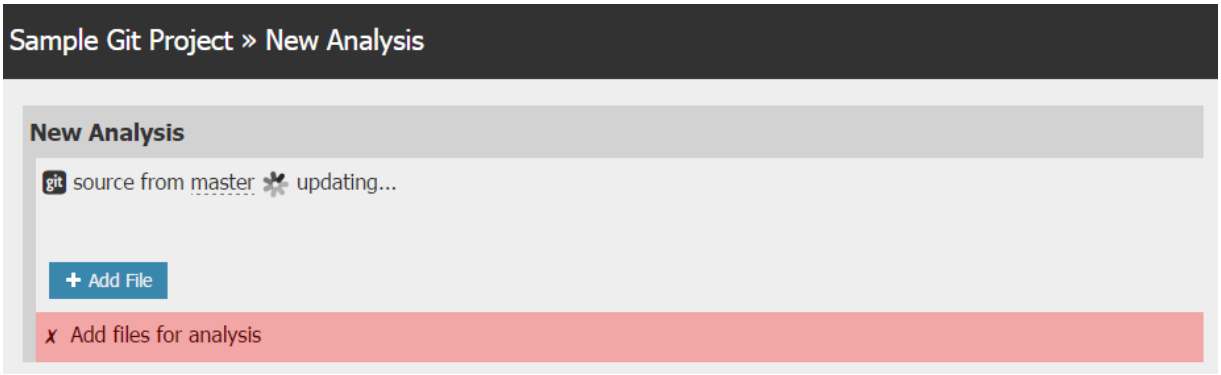
Once ready, click the *Begin Analysis* button at the bottom of the files area to start the analysis of those files. If for some reason there is a problem with the files, the *Begin Analysis* button will be replaced by one or more messages indicating what is wrong. You'll have to address whatever problems are mentioned there before starting an analysis.

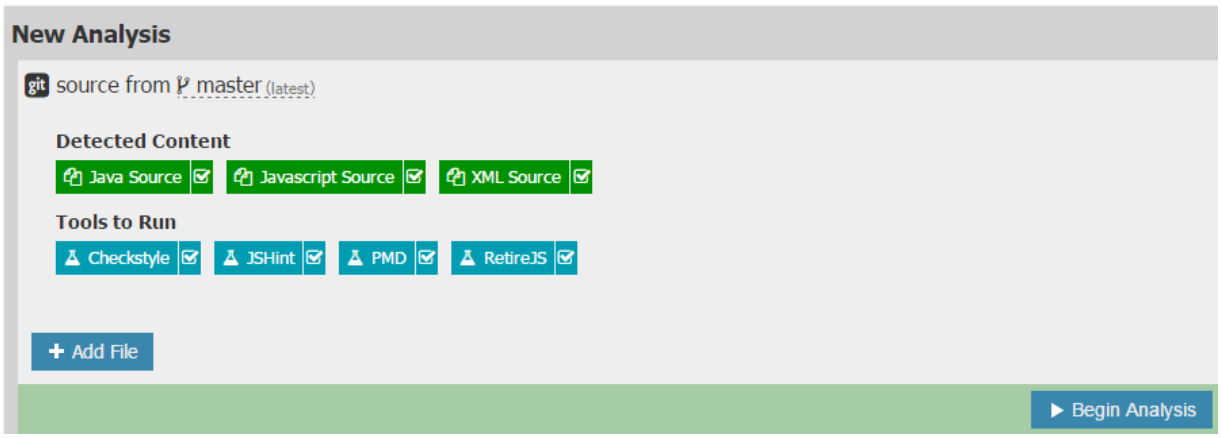
Once started, the page will display a timer to indicate how long the analysis has taken. Once complete, the timer will be replaced by a link to the analysis results page.



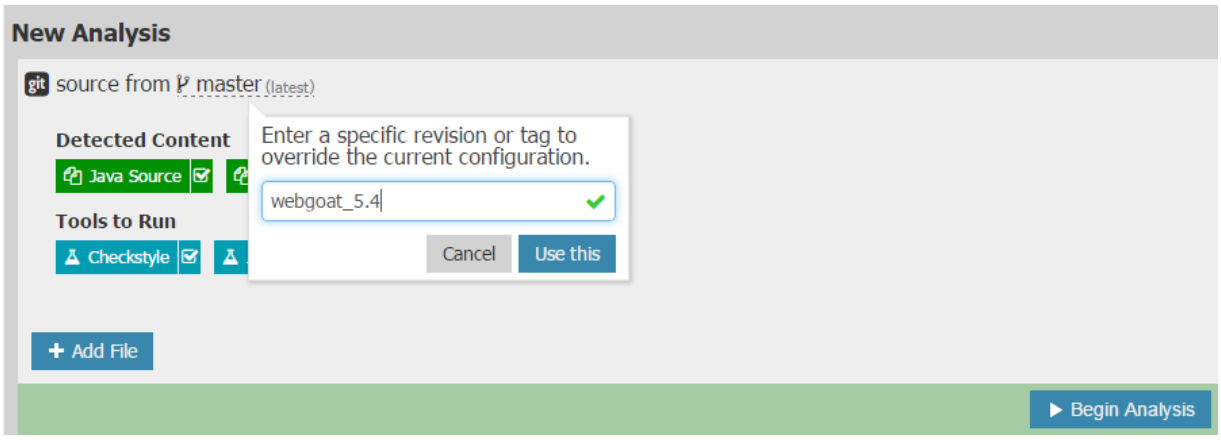
Inputs from Git Repositories

If you set up a Git configuration for a project (an Enterprise-Edition-only feature), the New Analysis page will automatically include the latest contents of the configured branch of the configured repository as an input.





Normally, Code Dx will update the local clone and check out the appropriate branch before sending the files to the analysis. If you set up your configuration to use the master branch, it will fetch the latest changes from master. As development is done on that branch, analysis of that branch will change along with the contents. But if you want to analyze a specific point in the repository, you can tell Code Dx to use a specific tag or commit by clicking on the underlined section of the input.



Fill in the field with a tag name or a commit hash, and click the *Use this* button.

Starting Analyses Manually from the IDE Plugins

Code Dx offers plugins for Visual Studio and Eclipse. These plugins offer many features to view and interact with the results of Code Dx analyses within the comfort of developers' familiar development environment. Among the features offered by the IDE plugins is the ability to initiate a scan directly from the development environment. This simplifies the process of packaging the relevant source and compiled artifacts (when applicable) since it is largely an automated process beyond some basic configuration options. For more details on how to initiate analyses from the IDE plugins, please see the Plugins Guide's relevant sections for the Visual Studio and Eclipse plugins.

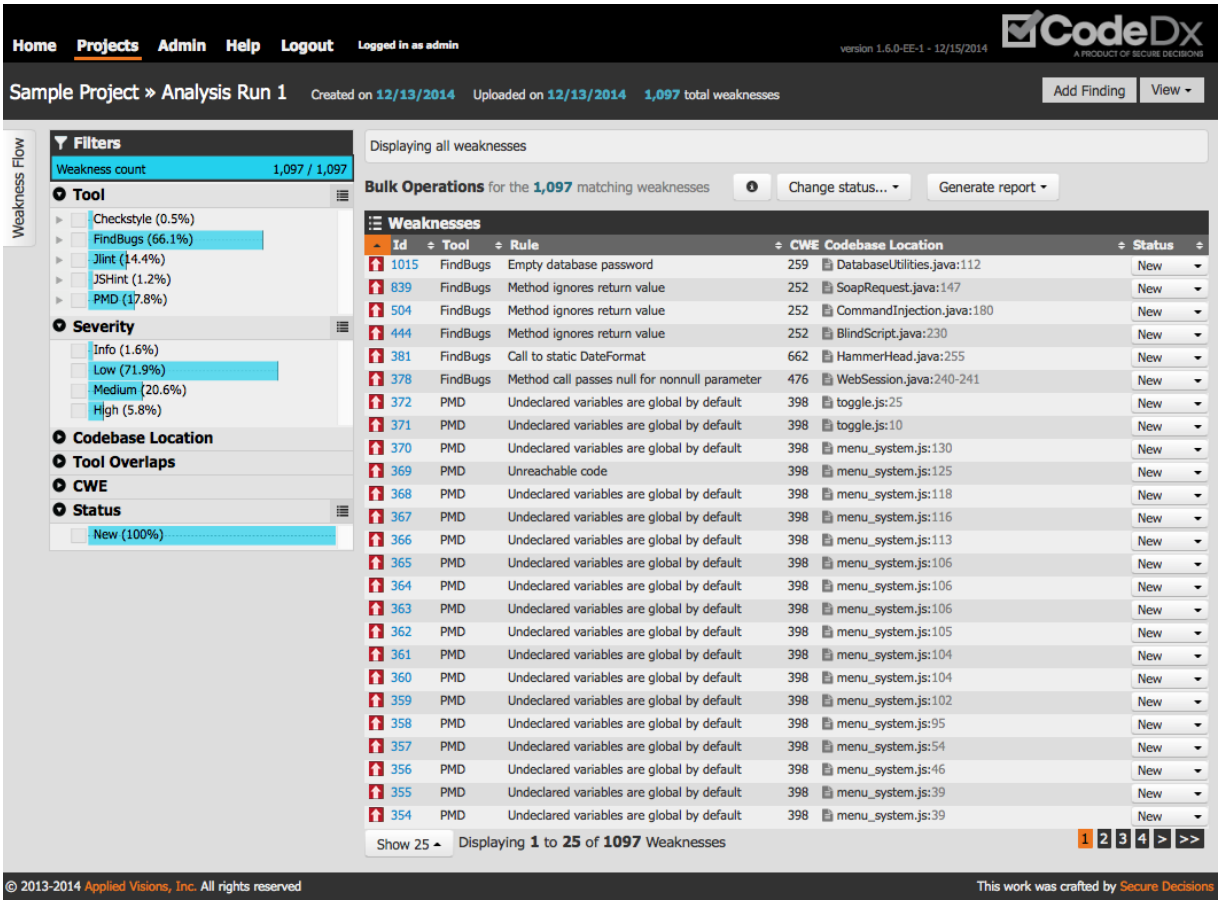
Starting Analyses Automatically Using the API

Code Dx offers an expanding API to interface with the system's functionality programmatically. The ability to push files for an analysis by Code Dx is exposed by the API. This enables automated integration scenarios such as continous integration. In a continous integration scenario a post-build step can be added to the build jobs to automatically push the source and compiled artifacts to Code Dx for analysis. This type of setup is strongly recommended for development teams to catch potential issues within their codebases early for quick remediation. "Test early and often" is advice that most certainly applies to static analysis. Code Dx does offer a Jenkins plugin, to facilitate use in a continuous integration context.

In order to use an API key for automated analyses, the key must be assigned the create permission for the project. The API call to push the files and initiate the analysis is documented in the API Guide.

Analysis Results

To view the results of the most recent analysis for a given project, click on the *Latest Analysis Run* link from the Projects page. This will take you to the Analysis Run page.



The Analysis Run page serves as the primary area for weakness triage within Code Dx and is structured around a powerful set of filtering options to enable quick weaknesses grouping and drill-down. In addition, manual findings can be added from this page as well to augment the ones uncovered by the static analysis tools.

This section is structured around the various user interface elements on this page that contribute towards the triage process.

Filtering

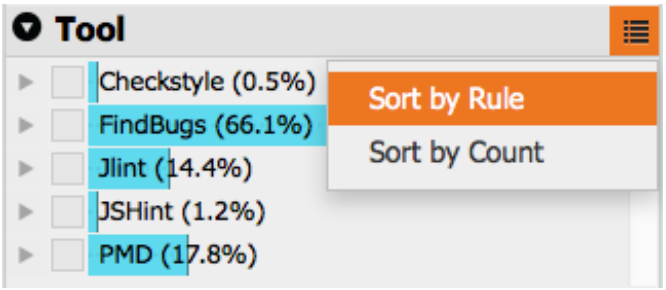
The filters are interactive bar charts that show the distribution of various properties of all weaknesses in the displayed analysis run. Each bar has a check box next to it that lets you filter on that value. Some filters have a tree structure, where certain elements can be expanded to reveal more elements. These elements will have a triangle next to them which you can click to expand or collapse them.

As you check and uncheck boxes, the entire page will update to match the current filter state. When the page first loads, all filters are in an off state, and the page

displays data for every weakness in the analysis.

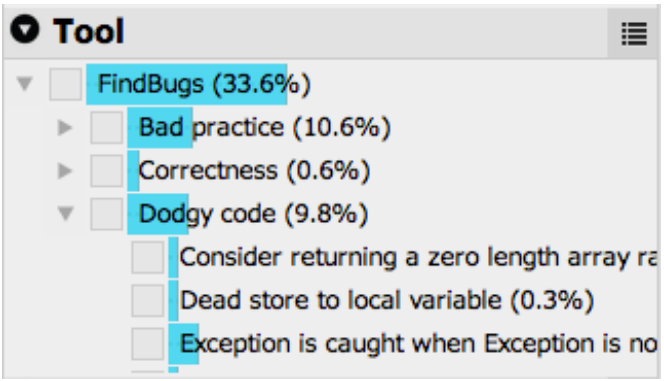
When the page is first loaded, certain filters will be expanded by default (Tool, Severity, and Status) while others will be in a collapsed state. Clicking the arrow to the left of each filter will toggle the collapse or expand state.

Expanded filters have sorting options as well. Clicking the sort button in the filter header will open a menu containing the possible sort choices.



Tool Filter

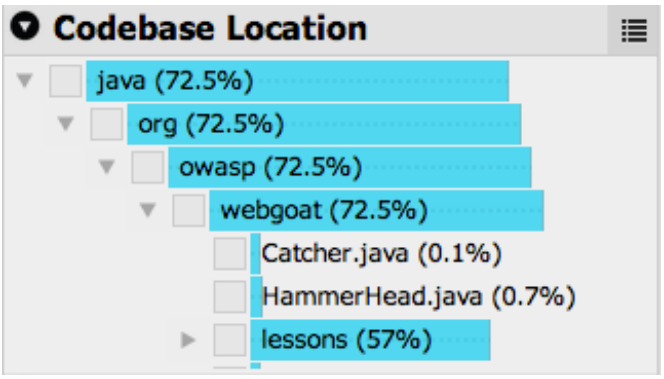
The Tool filter shows the hierarchical breakdown of "Tool" > "Rule Group" -> "Rule". "Tool" is the name of the tool that reported a weakness; "Rule Group" is a tool-specific category that a weakness can fall under; "Rule" is the identity of the weakness as reported by the tool.



Codebase Location Filter

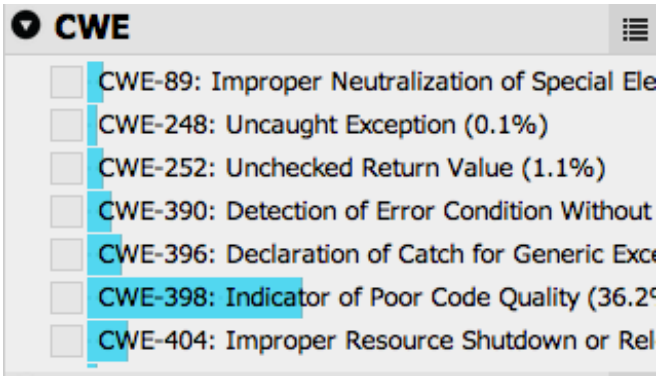
The Codebase Location filter shows where each weakness is located, reflecting the directory and file hierarchy of the codebase.

For .NET results, in some cases (especially if PDB files are not uploaded), source locations may not be available. Instead, a *Logical Locations* item will be shown. Under it will be locations organized by namespace, class, and method.



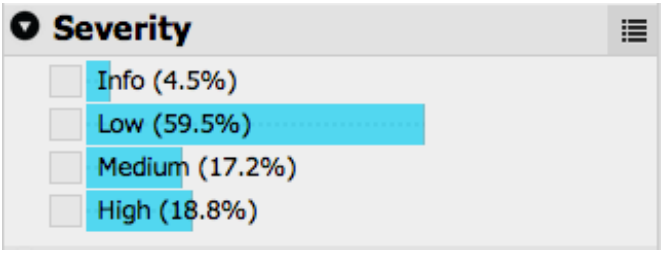
CWE Filter

The CWE (Common Weakness Enumeration) filter shows the distribution of weaknesses by what CWE they correspond to. For more information about the CWE system, see the official CWE site or CWE-Vis.



Severity Filter

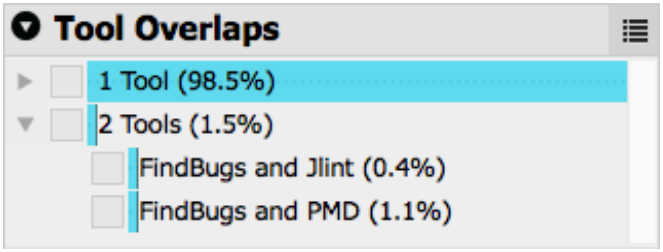
The Severity filter shows the distribution of weaknesses by how severe they are reported to be. Code Dx maps all reported severities to a scale of Info, Low, Medium, and High. Some tools don't report a severity, so they are represented as Unspecified.



Tool Overlaps Filter

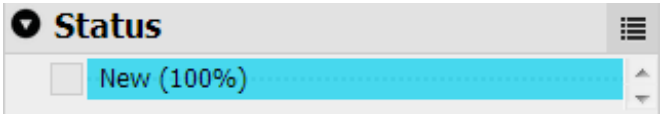
Since an analysis is a collection of many tool outputs, there is a chance that

multiple tools reported on the same weakness. This filter helps find weaknesses reported by different tools by correlating the reported weakness locations and corresponding CWE entries.



Status Filter

The status filter shows the distribution of each weakness's triage status. At first, all weaknesses in an analysis are set to New, but over time, weaknesses' statuses will be changed by users.



Filter Breadcrumbs

As you activate filters in the Analysis Run page, the page will update and filter breadcrumbs will appear. The breadcrumbs show an overview of what your current filter state is; they also let you turn off bits of the filter by clicking the *X* in each orange box.

Bulk Operations

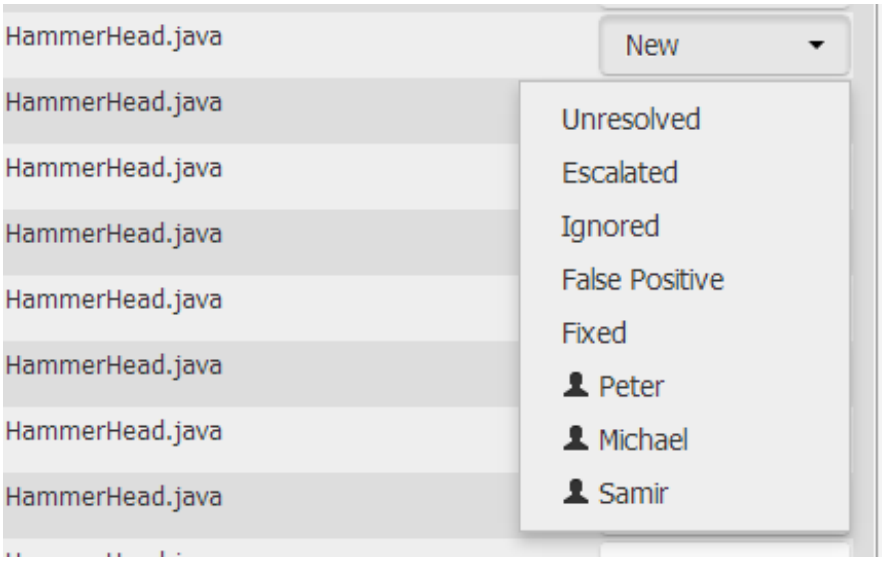
Certain operations can be performed in bulk on weaknesses that match the current filter state. From the *Bulk Operations* area you can:

- *Change status* to change the triage status for all of the filtered weaknesses at once instead of doing so one weakness at a time.
- *Generate report* lets you generate a report that contains all of the currently filtered weaknesses. If no filters are set, a report will be generated for all weaknesses in the analysis. Currently, the reports can be generated in PDF, CSV, and XML formats.

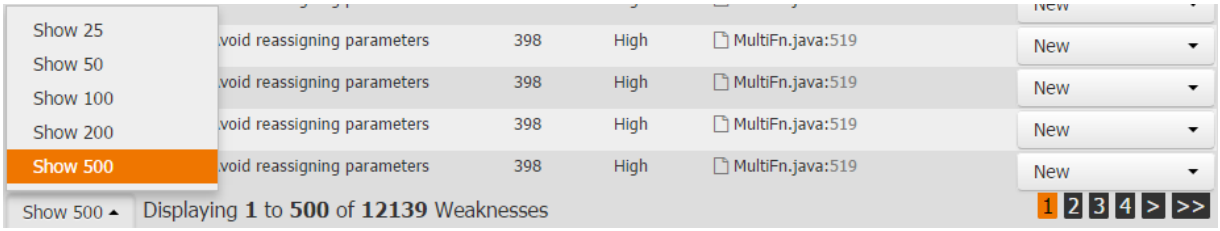
Weakness Table

The weakness table shows a simple largely text-based representation of each weakness individually. The number in the *Id* column is the unique identifier assigned to each weakness and the text for the Id doubles as a link to the weakness's details.

For users that have update permissions for a project, the *Status* column will have a widget that lets you change the current status of a weakness.



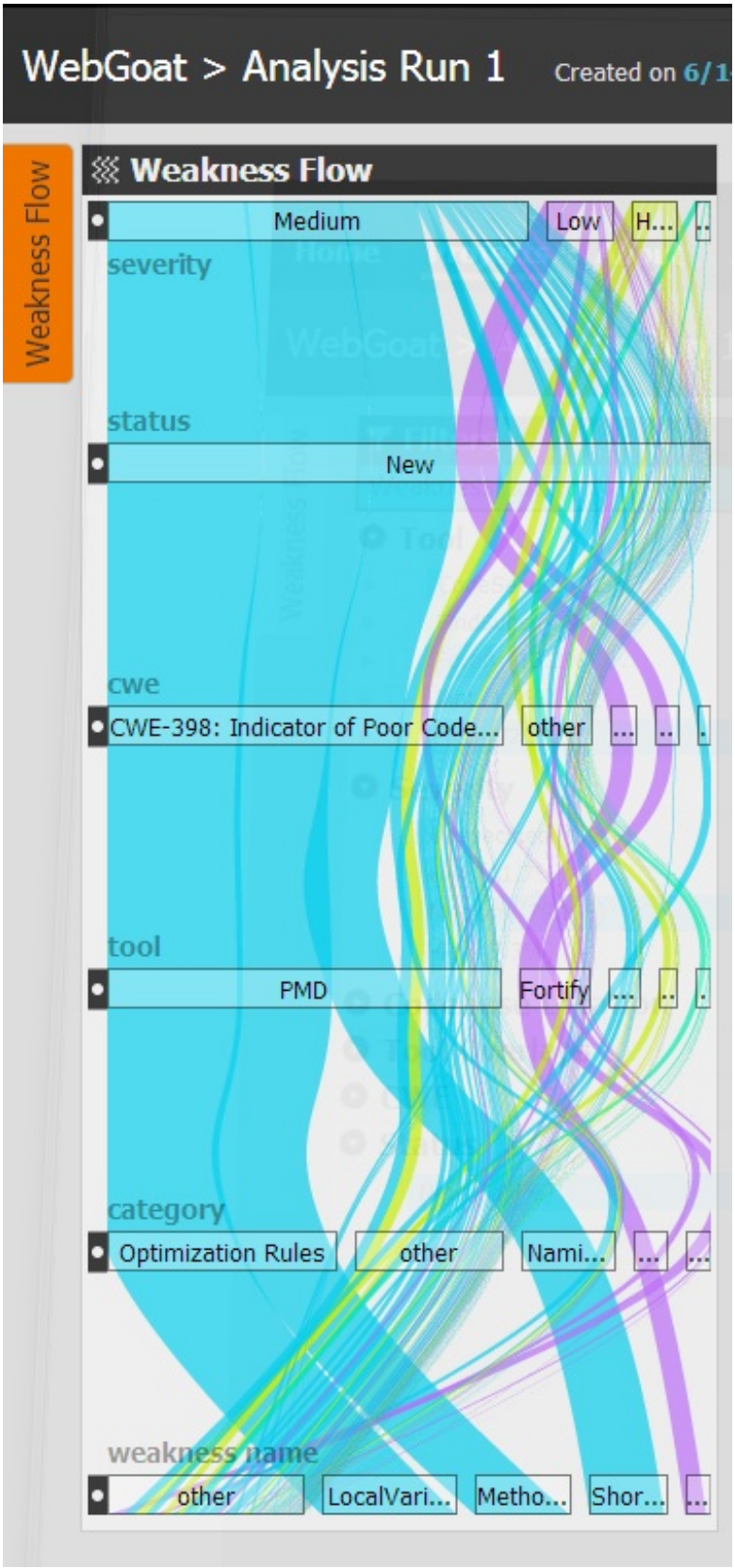
Analyses often have more weaknesses than can be displayed in the *Weakness Table* all at once. Because of this, the table is split into pages. By default, each page shows 25 weaknesses. Users can change the number of weaknesses per page using the *Show* button, seen below.



Weakness Flow

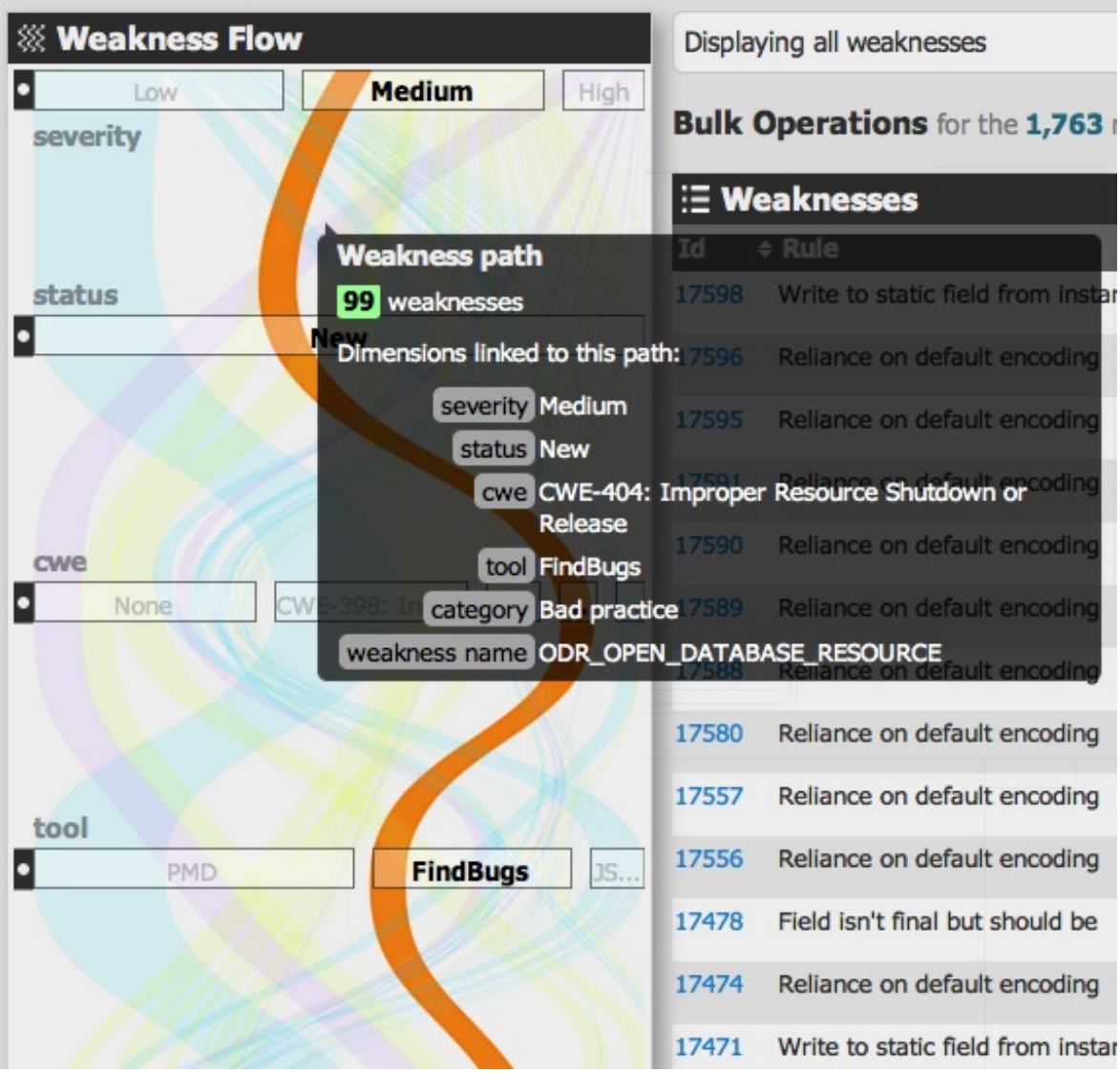
The *Weakness Flow* is a categorical breakdown of the weaknesses in an analysis. By default the weakness flow is collapsed to the left side of the Analysis Run page. Clicking on its drawer icon will expand it out. Clicking back on the same icon will

hide it back to the side.


WebGoat > Analysis Run 1
Created on 6/1
Weakness Flow
Weakness Flow
Medium
Low
H...
severity
status
New
cwe
CWE-398: Indicator of Poor Code...
other
tool
PMD
Fortify
category
Optimization Rules
other
Nami...
weakness name
other
LocalVari...
Metho...
Shor...

Each row represents different values in a category. For example, the *severity* category has values for High, Low, Medium, Info, and Unspecified.

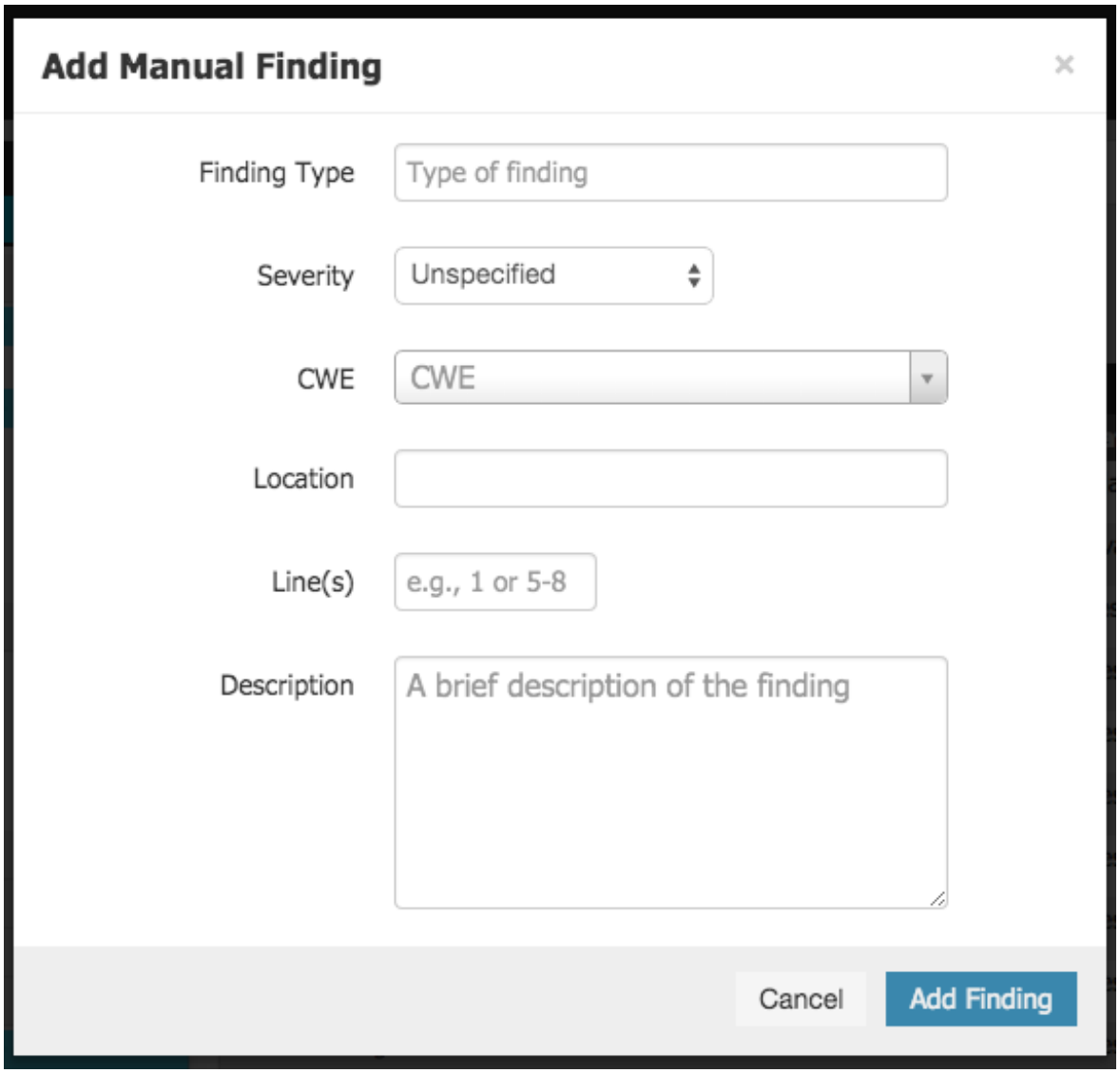
Each path (colorful, curvy lines) represents a set of weaknesses that have values matching each category value that the path passes through. Hovering the mouse over one of the paths will reveal more information about that path.



The black boxes with white circles at the side of each row are draggable. You can use them to re-order the rows in the weakness flow, updating the visualization in real time.

Adding Manual Findings

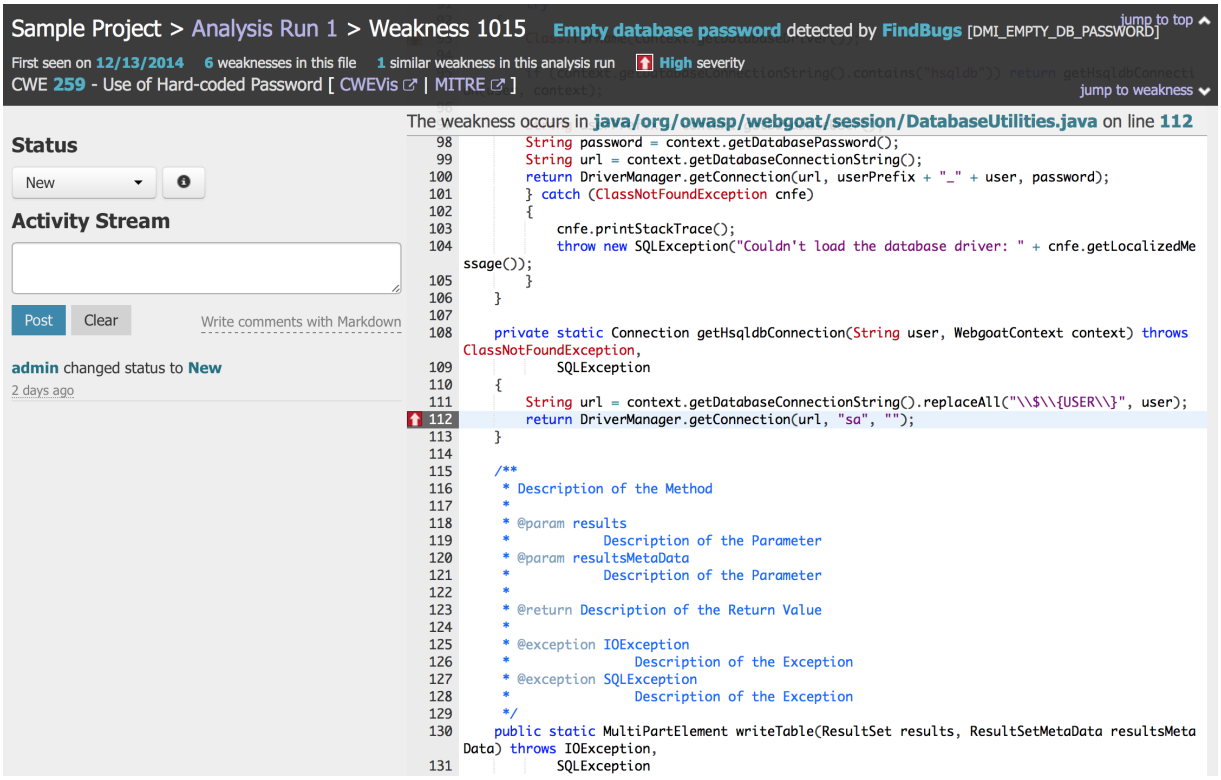
In the Enterprise Edition of Code Dx, users with the create permission for a project, will have access to the *Add Finding* button located in the page header. This allows you to manually add weaknesses to Code Dx during a manual code review for instance, as opposed to the ones automatically discovered by the static analysis tool. Clicking on the *Add Finding* button will trigger the following form to appear.



Once you've filled out the form and are ready, clicking on the *Add Finding* button will dismiss the form and update the Analysis Run page with the new finding. To delete or edit a manually added weakness, click on the weakness Id from the Analysis Run page to see its details view and from there you will see the options to edit and delete it (at present the edit and delete options are only visible for manually entered weaknesses).

Weakness Details

To access the details for a single weakness, navigate to the Analysis Run page, locate the weakness that you are interested in within the *Weakness Table*, and click the link in the *Id* column.



Details Summary

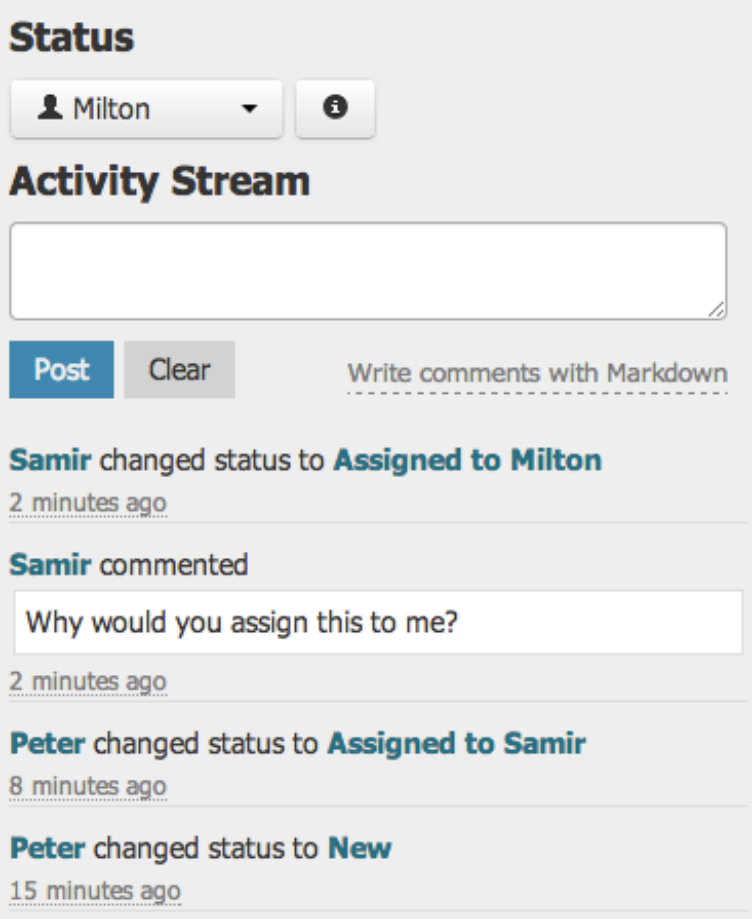
The header summary gives a quick overview of the weakness and the file where it is located. If the weakness is associated with a CWE, the CWE is noted, with links to CWEVis and the official CWE Mitre site.

The summary area also has "jump links". One link will scroll the source viewer to the location of the weakness in the file. The other link (which appears once you scroll down the page) will bring you back to the top of the page.

Activity Stream

The Activity Stream area has widgets that let you change the status of the weakness as well as comment on it. As users change the status and comment on a

weakness, messages appear in the activity stream, with newer messages at the top.



Source Display

The Source Display area shows the contents of the file where the weakness is located. Clicking the *jump to weakness* link in the header area will scroll the source display so that it shows the exact lines of the weakness, which are highlighted in dark grey in the line number gutter. The presence of severity markers in the gutter denote other weaknesses in the same file. When multiple weaknesses are present in a single line, the severity marker will show the highest level severity at that line. If you hover your mouse over any highlighted lines, a popup containing links to the details pages for the other weaknesses will appear.

948: Comparison of String objects using == or !=

Found by **FindBugs** on line **234** with CWE **597**

37: String Cmp

Found by **Jlint** on line **234** with CWE **597**

```
227
228         Input b = new Input();
229         b.setType(Input.SUBMIT);
230         b.setValue("Submit");
231         b.setName("SUBMIT");
232         ec.addElement(b);
233
234         if (s.getParser().getRawParameter("SUBMIT", "") != "")
235         {
236             if (s.getParser().getRawParameter("check1004", "") != "")
237             {
238                 makeSuccess(s);
239             }
240             else
241             {
242                 StringBuffer shipment = new StringBuffer();
243                 for (int i = 1001; i < 1001 + rewardsMap.size(); i++)
244                 {
245
246                     if (s.getParser().getRawParameter("check" + i, "") != "")
247                     {
248                         shipment.append(((Reward) rewardsMap.get(i)).getName() + "<br>");
249                     }
```