Abstract

The HCL Compass release supports enhanced security for Compass authenticated users. Compasscan now limit the number of login attempts for a user before the user becomes locked out. Compass can also limit the number login attempts from a host connecting through CompassWeb.

Content

These enhancements provide protection for Compassdatabases against brute-force password attacks. If an attacker is trying to determine the password of a particular user, they might send many login attempts with different username and password combinations until they find one that works. To protect against this Compasscan lock out a user or a host if a certain number of failed attempts have occurred. When a user or host locks out, the login error message is the same as if they entered an incorrect username and password. This prevents the attacker from seeing that any subsequent attempts will fail, causing them to waste resources on a continued fruitless attack. This severely limits the ability of an attacker to succeed in guessing the password.

When a user account is locked out, any subsequent login attempts for that user will be denied until the account lockout is removed by a user administrator. Any additional failed login attempts for this or any other user could result in the host eventually locking out. If the host is locked out then any login from that host will fail, but it will not cause any additional account lockouts. This limits the ability of an attacker from maliciously locking out many users.

The host lockout also limits the ability of an attacker to burden the system with a flood of login attempts. Once a host locks out after a certain number of failed attempts, Compasswill not allow any logins from that host. Fewer resources are consumed for any login attempts from locked out hosts. This prevents the attack from consuming too many resources, allowing legitimate use of the system even while an attack is underway.

This new feature is useful for deployments that have accounts that are not LDAP authenticated. Only ClearQuest-authenticated users are protected with this feature. For Compassusers that are LDAP authenticated, the LDAP server can be configured to enable account lockouts. The new lockout features introduced in this version are not meant to be a complete replacement for LDAP authentication. LDAP still provides many security capabilities that Compassdoes not have, such as minimum password complexity and maximum password age. LDAP authentication is also useful for managing identity in a common location. If you need those capabilities, you should continue to use LDAP authentication. If you have accounts that are not protected by LDAP, this feature will protect those accounts.

This feature is available for use in databases using at least Feature Level 9 (FL9). Note that once a database is upgraded to Feature Level 9, versions of Compassprior to 8.0.1.6 will not be able to login to the Compassdatabase. Feature Level prevention of down-rev client login remains as before, and is independent of this feature.

Configuring Brute Force Protection

The default configuration of Compassis that lockouts are disabled. You must configure it using the installutil loginsecurity command described below. To set loginsecurity configuration, a user needs Security Administrator privileges.

Login security is configured using the loginsecurity command. The syntax is as follows:

```
Usage: installutil loginsecurity
dbset_name (use empty quotes as default)
cq_login
cq_password
[-secret secret]
{-set {[-file configfileIn] [-setsecret new_secret]} | [-remove]} |
{-get [-file configfileOut]}
```

When setting the configuration you use the -set argument. You set lockout configuration using a file with the configuration options in them. Specify this file using the -file argument. There is also a secret word you can set that can be used if you are a Security Administrator and you are locked out. You would need this secret word and valid login credentials to remove the lockout on yourself. See the section below for removing lockout on yourself. An example configuration file is as follows:

```
# Comments are preceded by '#' and are ignored.
# This enables lockouts
lockout_enable 1

# Host lockout configuration
# This sets the HOST lockout threshold to 10 tries
lockout_threshold HOST 10
# This sets the HOST reset period to 60 seconds.
lockout_reset HOST 60

# User lockout configuration
# This sets the USER lockout threshold to 10 tries
lockout_threshold USER 10
# This sets the USER reset period to 60 seconds.
lockout_reset USER 60

# Whitelist / blacklists
lockout_whitelist HOST whitelisthost
lockout_whitelist USER whitelistuser1, whitelistuser
lockout_blacklist HOST blacklisthost3
lockout_blacklist USER bl_user1
lockout_blacklist USER bl_user2,bl_user3

# Login attempt table cleanup
# This says to cleanup any failed login attempts older
# than one day (60s/m * 60m/hr * 24hr/day), and only
```

```
# in about 1 in 20 login attempts (5%).
login_cleanup_age 8640000
login_cleanup_probability 5
```

Lockouts are enabled by the first line. Specifying a 1 enables lockouts and 0 disables them. The rest of the options have no effect if lockouts are disabled, but you can still include them in the configuration. If you decide to re-enable the lockouts you can just save the configuration to a file, change the value to 1 in the saved file and then set the configuration again with this edited file. The configuration file can include comments. Comments must be on their own line and the line must begin with "#". Blank lines are ignored. More details about the configuration options are given in the Maintaining Brute Force Protection section.

You can specify both configuration file and secret in the same command, or you can specify them separately. This may be useful if you want to change the configuration but not the secret word.

You can remove the configuration and secret word by using the -set and -remove arguments. The configuration and secret word are completely removed from the database. If you want to keep a copy of the configuration you should first get it using the -get option, as described next. Removing the configuration does not remove any existing lockouts, but login security is disabled once the configuration is removed. The lockouts will not have any effect until login security is enabled again.

You use the same installutil command to get the configuration. Instead of using the -set argument you use the -get argument. If you specify the -file argument then the configuration is written to the specified file. If you don't specify a -file argument then the configuration is written to the console.

The output of this command merely indicates whether a secret word has been set or not, it does not say what the word actually is. Once you set the secret word, it can not be obtained from the database. The secret word is not actually stored in the database as-is. Instead, a one-way hash of it is stored in the database, and thus it cannot be retrieved. The admin must not forget this secret word or they will not be able to self-unlock their accounts.

The brute-force protection is configured only at the working-master but the protection works on a site-by-site basis. That is, you set the configuration on the master site, then replicate the configuration to the other sites using normal MultiSite synchronization. To ensure that accounts lock out quickly on sites being attacked, each site maintains its own list of lockouts. While this may allow an attacker additional login attempts to different sites, it is still difficult for the attacker to determine that an account or host is locked out, and typically the number of additional sites is small.

Note: Do not run the loginsecurity -set command on any site other than the working master. If you accidentally do this, then remove the configuration on the same site, using -set -remove and replicate the changes to all databases. Then set the configuration on the master.

Maintaining Brute Force Protection

Once the login security is enabled and configured, there are commands the administrator will use to look for lockouts and remove them. To find and remove lockouts, a user needs User Administrator privileges. Knowing which accounts are locked out is the first step. The following examples assume that the user running the command is a User Administrator.

Getting lockouts

To get a list of lockouts in the database, use the `installutil getlockouts` command.

```
Usage: installutil getlockouts
dbset_name (use empty quotes as default)
cq_login
cq_password
[-secret secret]
[-type {ANY|LOGIN|HOST}]
[-match param]
[-max maxItems]
```

The first three arguments are self-explanatory. The -type is used to specify which type of lockout to list. You can specify that you want to see lockouts only for users (USER) or lockouts only for hosts (HOST). By default, the ANY option is chosen, which lists both users and hosts. You can limit the number of lockouts listed to make it easier to read using the -max option. By default, there is no limit to the number of lockouts listed.

If you only want the user lockouts then specify “-type USER”, for example:

```
installutil getlockouts mydbset admin admin_pwd -type USER
```

If you want to see if userA is locked out, you would specify both the “-type USER” and “-match userA”, for example:

```
installutil getlockouts mydbset admin admin_pwd -type USER -match userA
```

The command lists lockouts that match exactly and completely. For example, you cannot match by partial username. The following command would not show if userA was locked out:

```
installutil getlockouts mydbset admin admin_pwd -type USER -match user
```

The output is sorted by the USER name or HOST name. When both USER and HOST lockouts are listed, first USER lockouts are listed, and then HOST lockouts are listed. If the "-max" argument is used, then it is possible that not all lockouts will be shown. It will only list as many lockouts as you specify in that option.

Getting lockouts if you are locked out

If you are already locked out, then when you run the `installutil getlockouts` command, it will fail with invalid credentials. You must supply the secret word specified during configuration using the "-secret" argument along with valid Compass credentials. The lockout is temporarily bypassed just for the purpose of this command. This allows a locked out user administrator to query the lockouts and remove them.

```
installutil getlockouts mydbset admin admin_pwd -secret my_secret -type USER
-match admin
```

Removing lockouts

Once you know what is locked out, you can start removing lockouts. Remove lockouts using the `installutil removelockouts` command. The syntax of this command is as follows:

```
Usage: installutil removelockouts
dbset_name (use empty quotes as default)
cq_login
cq_password
[-secret secret]
[-type {ANY|USER|HOST}]
[-match param]
```

The syntax is very similar to the `getlockouts` command. Any lockouts returned by the getlockouts command would be removed if you used the same arguments for the removelockouts command. Once the lockout is removed, the user or host has the full number of attempts before the user or host identified by the parameter locks out again. Here are some examples:

If you want to remove all USER lockouts then specify "-type USER", for example:

```
installutil removelockouts mydbset admin admin_pwd -type USER
```

If you want to remove only the lockout for userA, you would specify both the "-type USER" and "-match userA", for example:

```
installutil removelockouts mydbset admin admin_pwd -type USER -match userA
```

The command removes lockouts that match exactly and completely. For example, you cannot remove lockouts by partial username. The following command would not remove the lockout on userA:

```
installutil removelockouts mydbset admin admin_pwd -type USER -match user
```

Removing lockout on self

If you are locked out, due to an attack or user error, the lockout on your account can only be removed if you supply a valid -secret argument, just as you did with getlockouts. Without this

argument, the command will fail with an 'invalid credentials' error. For example:

```
installutil removelockouts dbset_name admin admin_pwd -secret my_secret -type
USER -match admin
```

Listing login attempts

To get a list of failed login attempts, use the `installutil getloginattempts` command.

```
Usage: installutil getloginattempts
dbset_name (use empty quotes as default)
cq_login
cq_password
[-secret secret]
[-type {ANY|LOGIN|HOST}]
[-match param]
[-max maxItems]
```

This command will list any failed login attempts. The arguments are similar to the `installutil getlockouts` command. The attempts are listed from oldest to newest. For each failed login attempt, a USER value is logged. For Web, and OSLC APIs, a HOST is also logged, so each failed login may provide two lines of output, one for USER, and one for HOST. Included in the output of this command is date when the login attempt occurred, the type of the parmeter that was used (e.g. "USER" or "HOST") and the actual parameter used (e.g. "bob").

Administrators may use this to monitor failed login attempts for the Compass database. The output may be used to identify hosts that are being used to mount attacks, which should be added to the host blacklist. It may also reveal that attacks are under way on particular accounts that might be sensitive. Steps could then be taken to ensure that the accounts are properly secured.

The login attempts are listed from oldest to newest. If the "-max" argument is used to limit the number of login attempts printed, then the newest login attempts might not be shown. Login attempts are periodically removed from the system. See **Cleaning up the login attempts.**

Note: The `installutil getlockouts` was added in 8.0.1.6.

Brute-force protection scenarios

The system uses the configuration options to govern how it protects from brute force attacks. Brute force attacks are characterized by a malicious agent attempting to quickly login many times with different username and password combinations in an attempt to find one that works. Lockouts occur when the system detects that the threshold for failed login attempts has been reached for a particular parameter. Two kinds of parameters can lockout independently, the username and the host. More details are given in the following scenarios.

Malicious attacks on one user account – non-web clients

Malicious agents may target a single account on the system, rapidly trying many possible passwords. The system locks out this account once a certain number of failed attempts have occurred. The threshold is specified by the lockout_threshold option in the configuration file. This is followed by either the word “USER” or “HOST” to specify which parameter type to lock out. To cause a user lockout after 10 failed attempts you would use this configuration option:

```
lockout_threshold USER 10
```

Once the threshold has been reached, the account is locked out. Even if the correct credentials are supplied, it will still fail with a generic “invalid credentials” error. There is no specific indication given that the account is actually locked out. This is important, to avoid revealing any information about the validity of the credentials or whether the account is locked out.

These attacks may occur either by a person manually entering the user password in the Compass client (either Eclipse or Windows client), or using a CQPerl or VisualBasic script and the APIs to try many different passwords for a user.

Malicious attacks via Compass Web, OSLC, or ClearCase Integrations through the web

Another form of attack is where a malicious agent attempts to use Compass Web, OSLC, or ClearCase integrations through the web to login to many different users from a single host. They may do this because they know accounts will lock out and seek to try as many username/password combinations as possible, or to simply cause a denial-of-service type attack, trying to lock out as many users as possible. The system can protect against this by locking out the host after a certain number of failed attempts. This is configured using the following line in the configuration options:

```
lockout_threshold HOST 50
```

This tells the system to lockout a host if it has failed 50 login attempts. Any subsequent attempts will always fail with a generic “invalid credentials”. Furthermore, no user accounts will get locked out from login attempts from this host.
Note that the host lockout only occurs for login attempts via Compass Web or the OSLC interface or for ClearCase integrations that connect through Compass Web.

Denial-of-service mitigation and user-error mitigation

Any system of brute-force protection via lockouts is susceptible to denial-of-service attacks as the attacker may purposely attempt to lock out important users and hosts. It is also possible for users to accidentally lock themselves out by repeatedly using the wrong password. This adds administrative overhead as an administrator must take the time to periodically find and remove lockouts on valid users. There are several built-in mitigations to reduce the potential disruption for these kinds of attacks and user errors. These include an automatic lockout reset capability and whitelist/blacklist for hosts and users.

To reduce the potential for denial-of-service attacks caused by a malicious agent intentionally causing lockouts, the system allows lockouts to automatically reset. It is important to note that there is a difference between automatically resetting the lockout and removing the lockout via the installutil removelockouts command. The former (reset) simply allows one more attempt before locking out again. The second (removal) completely removes the lockout and allows the full number of retries before locking out again.
The automatic lockout resets are mainly effective against denial-of-service attacks when the HOST lockout is configured in addition to the USER lockouts. The idea is that once the HOST locks out, then any subsequent login attempts from that host will no longer cause USER lockouts. For this to work, the HOST must remain locked out so that the username will automatically reset once the configured reset period has elapsed. An example configuration for this is given in the next section.

In addition to denial-of-service mitigation, the automatic resets also reduce the amount of administration needed to respond to user errors. If a user is accidentally locked out due to too many failed attempts, they only would need to wait for the reset period to elapse before trying again. They would not have to contact support unless they truly forgot their credentials and need a password reset.

Lockout Reset Period

The login security can be configured to automatically reset a lockout after a certain amount of time has passed where no log in attempts have been made for that login parameter. This can be configured using the following line:

```
lockout_reset USER 60
```

This line configures the lockout reset period for USER parameters to 60 seconds. This can be thought of as a cooling-off period that allows the lockouts to automatically reset without administrative intervention. Once 60 seconds have passed and there have been no additional login attempts for the locked out parameter, the lockout expires and the user can make one more attempt. If that attempt fails, then the parameter locks out again. The user can try again in 60 seconds, ad-infinitum. If there is no lockout_reset line for the parameter type then there is no automatic reset. You can also specify a value of 0 which would explicitly disable the reset.

For an effective defense against denial-of-service attack causing lockouts on users, set the HOST reset period to a value higher than the USER reset period (or disabling automatic reset for HOST). This allows the USER lockout to be reset while the attacking HOST remains locked out. Remember, when a host is locked out, then any subsequent login attempts will not cause or prolong any user lockouts. Then the user can log in with valid credentials to remove the lockout. An attacker could circumvent this by mounting an attack from a different HOST (or proxy), but that may lock out eventually as well.

Here is an example configuration that would do this. The HOST locks out after 10 tries and remains locked out for 60 minutes. The USER locks out after 3 tries, but it resets within 1

minute. Once the USER lockout expires, the HOST lockout persists, since the reset period is longer. This prevents that host from causing future lockouts, giving the rightful user a chance to login and clear the lockout.

```
lockout_threshold HOST 10
lockout_reset HOST 3600
lockout_threshold USER 10
lockout_reset USER 60
```

Increasing Lockout Reset Period

If a constant reset period does not provide enough disincentive for the brute-force attack, the administrator can configure an increasing reset period. This is done by passing a negative number in the configuration, as follows:

```
lockout_reset HOST -60
```

When the first lockout on the HOST parameter occurs, it waits 60 seconds before resetting the lockout. If, after the reset period another failed attempt occurs, the new reset period is 120 seconds. Each subsequent reset period is a multiple of the absolute value provided in the configuration (e.g., 180 seconds, 240 sections, 300 seconds, ...).

Once a user or host is locked out, the user or anyone using the host must wait for the lockout to reset automatically or for the user administrator to remove it. If the lockout automatically resets, the person has one more try before the parameter locks out again, as described above. If, after the lockout resets, the user successfully logs in, then the lockout is completely removed (as if the administrator ran installutil removelockout).

Blacklists

If an attacker consistently mounts denial-of-service or brute-force attacks via a group of hosts or proxy servers, these hosts can be added to a blacklist. Any incoming request from a blacklisted host will fail as if the host is locked out. It will simply fail with invalid credentials, and it will not cause any additional user lockouts. A blacklisted host consumes less database, CPU, and memory resources during the login attempt, reducing the ability of an attacker to mount an effective resource-based denial-of-service campaign.

A blacklist is a list of users or IP addresses that will be prevented from logging in. If a USER or HOST is in the blacklist then Compass does not even attempt to authenticate the user credentials. A blacklist is specified in the configuration file that is set with the installutil loginsecurity command. The relevant configuration lines look like this:

```
lockout_blacklist HOST blacklisthost3
lockout_blacklist USER bl_user1
lockout_blacklist USER bl_user2,bl_user3
```

The lines can specify either HOST or USER blacklist values and you can specify multiple

comma-separated values. Note, if a USER or HOST value is in the blacklist then the login attempt is not even recorded. This is to prevent database resources from being consumed by the blacklisted login attempts. The host must be specified as an IP address. Currently, only Compass Web, OSLC, and VersionVault integrations through Compass Web will log host addresses during failed login attempts. These are the only connections that will lockout host addresses.

An administrator can use the `installutil getloginattempts` command to get a list of failed login attempts. From that list the administrator can determine whether an attack is underway from a particular host, and then add that host to the blacklist.

Whitelists

There might be a HOST or a USER that should never be locked out. It may be so important that these never lockout that it is worth accepting the risk of an attacker breaking into this account using brute-force methods, or mounting a denial-of-service attack via a whitelisted host or user. Suppose you have an important service account. The account does not have any admin privileges but it can perform certain specific, regular, and important tasks defined for that service user. To prevent the account from being locked out, which would prevent the important regular tasks from being completed, the username could be added to a USER whitelist. Any username on the USER whitelist would never be locked out no matter how many attempts were made. It is therefore important that a suitably strong password be used for this account, and that it be changed regularly.

An example of a HOST that should never be locked out might be an important proxy server used by Compass Web users to login to ClearQuest. When connecting to Compass Web servers using a proxy server, Compass sees the requests as coming from the proxy-server rather than the actual location of the browser client. If an attacker mounts an attack through the proxy he can effectively cause a denial-of-attack by locking out the proxy server and preventing all users who use that proxy server from connecting. To prevent this proxy server from being locked out the customer can put the proxy server on a HOST whitelist.

While the whitelisted values never lock out, the other parameter in the failed login attempt is still logged if it is not also in a whitelist. For example, if the user is in a whitelist but the host is not, then the failed login attempt is still logged for the host. This allows some protection in spite of the whitelisted login parameters.

Whitelist values are specified in the configuration file, just like blacklists. Here are some examples:

```
lockout_whitelist HOST whitelisthost
lockout_whitelist USER whitelistuser1, whitelistuser
```

The hosts must be specified as an IP address. Currently, the HOST is only used by Compass Web, OSLC, and ClearCase integrations through Compass Web connections.

Reference of login security configuration options

A reference for configuration options follows.

Enabling lockouts

```
lockout_enable {0|1}
```

Enable or disable lockouts. Use 0 to disable or 1 to enable lockouts.
To disable lockouts:
```
lockout_enable 0
```

To enable lockouts:
```
lockout_enable 1
```

Setting lockout thresholds

```
lockout_threshold {USER|HOST} {threshold}
```

Specifies how many failed login attempts should occur before a parameter locks out.

To lockout users after 5 attempts:
```
lockout_threshold USER 5
```

To lockout hosts after 10 attempts:
```
lockout_threshold HOST 10
```

Automatic Lockout Resets

```
lockout_reset {USER|HOST} {reset_period_in_seconds}
```

Specifies how long until a lockout automatically resets, allowing one more login attempt before locking out. The reset period is specified in seconds. The reset happens the next time the user or host is used during the login after the reset period has elapsed.

To reset USER lockouts after 60 seconds:
```
lockout_reset USER 60
```

To reset HOST lockouts after 10 minutes:
```
lockout_reset HOST 600
```

To reset HOST lockouts after 60 seconds after first lockout, then 120 seconds after second lockout, then 180 seconds after third lockout, etc. The reset period starts again at 60 seconds after a successful login or after the lockout is removed.
```
lockout_reset HOST -60
```

Note: The installutil getlockouts command will continue to show that the parameter is locked out

past the reset period. This is expected since the reset actually occurs on the next successful login with that user or host after the reset period has elapsed.

Adding to a whitelist or blacklist

You can add users or hosts to a whitelist or blacklist. A whitelist value never locks out, but during a failed login attempt that uses a whitelist value, a non-whitelist value may get locked out. For example, suppose BOB is on the USER whitelist. If a failed login attempt occurs from the host with IP address W.X.Y.Z, and this host is not on the HOST whitelist, then a failed login attempt will be logged for HOST W.X.Y.Z, but not for the username BOB.
If a value is on a blacklists, the failed login is not logged (in order to prevent an attack from consuming database space) but it still behaves as it it was locked out. This prevents any lockout on values not in the blacklist. For example, suppose host W.X.Y.Z is on the blacklist but JOE is not. If a failed login attempt comes from W.X.Y.Z, it will not contribute to a lockout of JOE.
The configuration for whitelist or blacklist is one or more lines like this:

```
lockout_whitelist {USER|HOST} value1[,value2[,value3[,...]]]
lockout_blacklist {USER|HOST} value1[,value2[,value3[,...]]]
```

Multiple lockout_whitelist or lockout_blacklist lines may exist in the configuration, making it easier to organize this file. Values must be comma separated and white space is ignored.

Cleaning up the login attempts

Failed login attempts are stored in the Compass master database. Over time this list may grow very large. You should configure Compass to automatically clean up this list, removing the record of attempts that are no longer needed. You configure this by specifying two values. The first value is how old (in seconds) the record of a login attempt must be before it can be removed. The second value is the probability that any given login should do the cleanup. The cleanup will happen during a successful login, and the probability that the cleanup will happen is specified by login_cleanup_probability. For example if you specify 600 for login_cleanup_age and 5 for login_cleanup_probability, then cleanup will happen roughly once for every 20 logins and it will remove the records for any attempts that are older than 10 minutes. If you specify 100 for login_cleanup_probability then cleanup happens on every login. If you specify 0 then cleanup never happens. The default value for login_cleanup_probability is 1 (1%). The default value for login_cleanup_age is one day.

```
login_cleanup_age {age_to_cleanup_in_seconds}
login_cleanup_probability {percent_chance_that_cleanup_will_occur}
```

Note – there is no way to view the failed login attempts except by connecting to your database directly and running an SQL query. This capability may be added to installutil in a future release.

Multisite and Login Security

Configuration of login security is made on the working master for a multisite family of databases. This is automatically replicated to all the databases with normal MultiSite synchronization. Each site separately keeps track of login attempts and lockouts and these are not replicated between databases.