Do not put the subscription-manager password onto the command line. (#...492)

Browse files

```
* Do not put the subscription-manager password onto the command line.

Fixes CVE-2022-0852

Passing values on the command line is insecure.  With this change,
the rhsm password is passed interactively to subscription-manager
instead of being passed on the commandline when we shell out to it.

The structure of this change deserves a bit of description.  Previously,
we called one function to assemble all of the information needed to
invoke subscription-manager and then returned that as a string that
could be run as a command line.  We called a second function with that
string to actually run the command.

To send the password interactively, we need to stop adding the password
to the string of command line arguments but it still makes sense to keep
the code that figures out the password together with the code which
finds the other command line args.  So it makes sense to keep a single
function to do that but return the password and other args separately.

We could use a dict, a class, or a tuple as the return value from the
function.  That doesn't feel too ugly.  But then we need to pass that
structure into the function which takes care of invoking
subscription-manager on the command line and that *does* feel ugly.
That function would have to care about the structure of the data we pass
in (If a tuple, what is the order?  If a dict, what are the field
names?, etc).  To take care of this, we can make the data structure that
we return from assembling the data a class and the function which calls
subscription-manager a method of that class because it's quite natural
for a method to have knowledge of what attributes the class contains.

Hmm... but now that we have a class with behaviours (methods), it starts
to feel like we could do some more things.  A function that fills in the
values of a class, validates that the data is proper, and then returns
an instance of that class is really a constructor, right?  So it makes
sense to move the function which assembles the data and returns the
class a constructor.  But that particular function isn't very generic:
it uses knowledge of our global toolopts.tool_opts to populate the
class.  So let's write a simple __init__() that takes all of the values
needed as separate parameters and then create an alternative constructor
(an @classmethod which returns an instance of the class) which gets the
```

```
data from a ToolOpt, and then calls __init__() with those values and
returns the resulting class.

Okay, things are much cleaner now, but there's one more thing that we
can do.  We now have a class that has a constructor to read in data and
a single method that we can call to return some results.  What do we
call an object that can be called to return a result?  A function or
more generically, in python, a Callable.  We can turn this class into a
callable by renaming the method which actually invokes
subscription-manager __call__().

What we have at the end of all this is a way to create a function which
knows about the settings in tool_opts which we can then call to perform
our subscription-manager needs::

    registration_command = RegistrationCommand.from_tool_opts()
    return_code = registration_command()

OAMG-6551 #done  convert2rhel now passes the rhsm password to subscription-manager securely.

* Modify the hiding of secret to hide both --password SECRET and
  --password=SECRET.  Currently we only use it with passwords that we
  are passing in the second form (to subscription-manager) but catching
  both will future proof us in case we use this function for more things
  in the future. (Eric Gustavsson)
  * Note: using generator expressions was tried here but found that they
    only bind the variable being iterated over at definition time, the
    rest of the variables are bound when the generator is used.  This
    means that constructing a set of generators in a loop doesn't really
    work as the loop variables that you use in the generator will have a
    different value by the time you're done.

    So a nested for loop and if-else is the way to implement this.

* Add global_tool_opts fixture to conftest.py which monkeypatches a
  fresh ToolOpts into convert2rhel.toolopts.tool_opts.  That way tests
  can modify that without worrying about polluting other tests.

  * How toolopts is imported in the code makes a difference whether this
    fixture is sufficient or if you need to do a little more work. If
    the import is::

      from convert2rhel import toolopts
      do_something(toolopts.tool_opts.username)

   then your test can just do::

      def test_thing_that_uses_toolopts(global_tool_opts):
          global_tool_opts.username = 'badger'

   Most of our code, though, imports like this::
```

```
      # In subscription.py, for instance
      from convert2rhel.toolopts import tool_opts
      do_something(tool_opts.username)

   so the tests should do something like this::

      def test_toolopts_differently(global_test_opts, monkeypatch):
          monkeypatch.setattr(subscription, 'tool_opts', global_tool_opts)

* Sometimes a process will close stdout before it is done processing.
  When that happens, we need to wait() for the process to end before
  closing the pty.  If we don't wait, the process will receive a HUP
  signal which may end it before it is done.
  * But on RHEL7, pexpect.wait() will throw an exception if you wait on
    an already finished process.  So we need to ignore that exception
    there.

lgtm is flagging some cases where it thinks we are logging sensitive
data.  Here's why we're ignoring lgtm:

* One case logs the username to the terminal as a hint for the user as
  to what account is being used.  We consider username to not be
  sensitive.
* One case logs the subscription-manager invocation.  We run the command
  line through hide_secrets() to remove private information when we do
  this.
* The last case logs which argument was passed to subscription-manager
  without a secret attached to it.  ie: "--password" is passed to
  subscription-manager without a password being added afterwards.  In
  this case, the string "--password" will be logged.

Testing farm doesn't have python-devel installed.
We need that to install psutil needed as a testing dependency.

Co-authored-by: Daniel Diblik <ddiblik@redhat.com>
```

main (#492)
v1.0 v0.26

abadger and danmyway committed on May 27

1 parent 8658831 commit 8d72fb030ed31116fdb256b327d299337b000af4

Showing **13 changed files** with **983 additions** and **287 deletions**.

Split Unified

4 .gitleaks.toml

```
@@ -0,0 +1,4 @@
+ [allowlist]
+ paths = [
```

```
+     '''tests/integration/tier0/check-cve/test_cve_fixes.py''',
+ ]
```

246 convert2rhel/subscription.py

```
109 109
110 110     # Loop the registration process until successful registration
111 111     attempt = 0
112     -   while True and attempt < MAX_NUM_OF_ATTEMPTS_TO_SUBSCRIBE:
113     -       registration_cmd = get_registration_cmd()
114     -
    112 +   while attempt < MAX_NUM_OF_ATTEMPTS_TO_SUBSCRIBE:
    113 +       registration_cmd = RegistrationCommand.from_tool_opts(tool_opts)
115 114         attempt_msg = ""
116 115         if attempt > 0:
117 116             attempt_msg = "Attempt %d of %d: " % (attempt + 1, MAX_NUM_OF_ATTEMPTS_TO_SUB
118 117         loggerinst.info("%sRegistering the system using subscription-manager ...", attemp
119 118
120     -       output, ret_code = call_registration_cmd(registration_cmd)
    119 +       output, ret_code = registration_cmd()
121 120         if ret_code == 0:
122 121             # Handling a signal interrupt that was previously handled by
123 122             # subscription-manager.
124 123             if "user interrupted process" in output.lower():
125 124                 raise KeyboardInterrupt
126 125             return
    126 +
127 127         loggerinst.info("System registration failed with return code = %s" % str(ret_code
128 128         if tool_opts.credentials_thru_cli:
129 129             loggerinst.warning(
138 138     loggerinst.critical("Unable to register the system through subscription-manager.")
139 139
140 140
141     - def get_registration_cmd():
142     -     """Build a command for subscription-manager for registering the system."""
143     -     loggerinst.info("Building subscription-manager command ... ")
144     -     registration_cmd = ["subscription-manager", "register", "--force"]
145     -
146     -     loggerinst.info("Checking for activation key ...")
147     -     if tool_opts.activation_key:
148     -         # Activation key has been passed
149     -         # -> username/password not required
150     -         # -> organization required
151     -         loggerinst.info("    ... activation key detected: %s" % tool_opts.activation_key)
152     -
153     -         # TODO: Parse the output of 'subscription-manager orgs' and let the
154     -         # user choose from the available organizations. If there's just one,
155     -         # pick it automatically.
```

```
156 -         # Organization is required when activation key is used
141 + class RegistrationCommand(object):
142 +     def __init__(self, activation_key=None, org=None, username=None, password=None, serve
143 +         """
144 +         A callable that can register a system with subscription-manager.
145 +
146 +         :kwarg server_url: Optional URL to the subscription-manager backend.
147 +             Useful when the customer has an on-prem subscription-manager instance.
148 +         :kwarg activation_key: subscription-manager activation_key that can be
149 +             used to register the system. Org must be specified if this was given.
150 +         :kwarg org: The organization that the activation_key is associated with.
151 +             It is required if activation_key is specified.
152 +         :kwarg username: Username to authenticate with subscription-manager.
153 +             Required if password is specified.
154 +         :kwarg password: Password to authenticate with subscription-manager.
155 +             It is required if username is specified.
156 +
157 +         .. note:: Either activation_key and org or username and password must
158 +             be specified.
159 +         """
160 +         self.cmd = "subscription-manager"
161 +         self.server_url = server_url
162 +
163 +         if activation_key and not org:
164 +             raise ValueError("org must be specified if activation_key is used")
165 +
166 +         self.activation_key = activation_key
167 +         self.org = org
168 +
169 +         self.password = password
170 +         self.username = username
171 +
172 +         if (password and not username) or (username and not password):
173 +             raise ValueError("username and password must be used together")
174 +
175 +         elif not password:
176 +             # No password set
177 +             if not self.activation_key:
178 +                 raise ValueError("activation_key and org or username and password must be
179 +
180 +     @classmethod
181 +     def from_tool_opts(cls, tool_opts):
182 +         """
183 +         Alternate constructor that gets subscription-manager args from ToolOpts.
184 +
185 +         convert2rhel's command-line contains the information needed to register
186 +         with subscription-manager. Get the information from the passed in
187 +         ToolOpts structure to create the RegistrationCommand.
188 +
```

```diff
+        :arg tool_opts: The :class:`convert2rhel.toolopts.ToolOpts` structure to
+            retrieve the subscription-manager information from.
+        """
+        loggerinst.info("Gathering subscription-manager registration info ... ")
+
+        registration_attributes = {}
         if tool_opts.org:
-            loggerinst.info("    ... org detected")
-
-        org = tool_opts.org
-        while not org:
-            org = utils.prompt_user("Organization: ")
-
-        registration_cmd.extend(("--activationkey=%s" % tool_opts.activation_key, "--org=
-    else:
-        loggerinst.info("    ... activation key not found, username and password required
+            loggerinst.info("    ... organization detected")
+            registration_attributes["org"] = tool_opts.org
+
+        if tool_opts.activation_key:
+            # Activation key has been passed
+            # -> username/password not required
+            # -> organization required
+            loggerinst.info("    ... activation key detected")
+            registration_attributes["activation_key"] = tool_opts.activation_key
+
+            while "org" not in registration_attributes:
+                loggerinst.info("    ... activation key requires organization")
+                # Organization is required when activation key is used
+                # TODO: Parse the output of 'subscription-manager orgs' and let the
+                # user choose from the available organizations. If there's just one,
+                # pick it automatically.
+                org = utils.prompt_user("Organization: ").strip()
+                # In case the user entered the empty string
+                if org:
+                    registration_attributes["org"] = org
+        else:
+            # No activation key -> username/password required
+            if tool_opts.username and tool_opts.password:
+                loggerinst.info("    ... activation key not found, using given username a
+            else:
+                loggerinst.info("    ... activation key not found, username and password
+
+            if tool_opts.username:
+                loggerinst.info("    ... username detected")
+                username = tool_opts.username
+            else:
+                username = ""
+                while not username:
```

```
+                    username = utils.prompt_user("Username: ")
+
+            registration_attributes["username"] = username
+
+            if tool_opts.password:
+                loggerinst.info("    ... password detected")
+                password = tool_opts.password
+            else:
+                if tool_opts.username:
+                    # Hint user for which username they need to enter pswd
+                    loggerinst.info("Username: %s", username)  # lgtm[py/clear-text-loggi
+                password = ""
+                while not password:
+                    password = utils.prompt_user("Password: ", password=True)
+
+            registration_attributes["password"] = password
+
+        if tool_opts.serverurl:
+            loggerinst.debug("    ... using custom RHSM URL")
+            registration_attributes["server_url"] = tool_opts.serverurl
+
+        return cls(**registration_attributes)
+
+    @property
+    def args(self):
+        """
+        This property is a list of the command-line arguments that will be passed to
+        subscription-manager to register the system. Set the individual attributes for
+        :attr:`server_url`, :attr:`activation_key`, etc to affect the values here.
+
+        .. note:: :attr:`password` is not passed on the command line. Instead,
+            it is sent to the running subscription-manager process via pexpect.
+        """
+        args = ["register", "--force"]
+
+        if self.server_url:
+            args.append("--serverurl=%s" % self.server_url)
+
+        if self.activation_key:
+            args.append("--activationkey=%s" % self.activation_key)
+
+        if self.org:
+            args.append("--org=%s" % self.org)
+

+        if self.username:
+            args.append("--username=%s" % self.username)
+

+        return args
+
```

```
+     def __call__(self):
+         """
+         Run the subscription-manager command.
+
+         Wrapper for running the subscription-manager command that keeps
+         secrets secure.
+         """
+         if self.password:
+             loggerinst.debug(
+                 "Calling command '%s %s'" % (self.cmd, " ".join(hide_secrets(self.args)))
+             )  # lgtm[py/clear-text-logging-sensitive-data]
+             output, ret_code = utils.run_cmd_in_pty(
+                 [self.cmd] + self.args, expect_script=(("[Pp]assword: ", self.password +
+             )
+         else:
+             # Warning: Currently activation_key can only be specified on the CLI. This is
+             # but there's nothing we can do about it for now. Once subscription-manager i
+             # https://issues.redhat.com/browse/ENT-4724 is implemented, we can change bot
+             # and activation_key to use a file-based approach to passing the secrets.
+             output, ret_code = utils.run_subprocess([self.cmd] + self.args, print_cmd=Fal
 167  298
 168  -         if tool_opts.username:
 169  -             loggerinst.info("    ... username detected")
      299 +         return output, ret_code
 170  300
 171  -         username = tool_opts.username
 172  -         while not username:
 173  -             username = utils.prompt_user("Username: ")
 174  301
 175  -         if tool_opts.password:
 176  -             loggerinst.info("    ... password detected")
      302 + def hide_secrets(args):
      303 +     """
      304 +     Replace secret values with asterisks.
 177  305
 178  -         password = tool_opts.password
 179  -         while not password:
 180  -             password = utils.prompt_user("Password: ", password=True)
      306 +     This function takes a list of arguments which will be passed to
      307 +     subscription-manager on the command line and returns a new list
      308 +     that has any secret values obscured with asterisks.
 181  309
 182  -         registration_cmd.extend(("--username=%s" % username, "--password=%s" % password))
      310 +     :arg args: An argument list for subscription-manager which may contain
      311 +         secret values.
      312 +     :returns: A new list of arguments with secret values hidden.
      313 +     """
      314 +     obfuscation_string = "*" * 5
      315 +     secret_args = frozenset(("--password", "--activationkey", "--token"))
```

```
183  316
184  -     if tool_opts.serverurl:
185  -         loggerinst.debug("    ... using custom RHSM URL")
186  -         registration_cmd.append("--serverurl=%s" % tool_opts.serverurl)
     317  +     sanitized_list = []
     318  +     hide_next = False
     319  +     for arg in args:
     320  +         if hide_next:
     321  +             # Second part of a two part secret argument (like --password *SECRET*)
     322  +             arg = obfuscation_string
     323  +             hide_next = False
187  324
188  -     return registration_cmd
     325  +         elif arg in secret_args:
     326  +             # First part of a two part secret argument (like *--password* SECRET)
     327  +             hide_next = True
189  328
     329  +         else:
     330  +             # A secret argument in one part (like --password=SECRET)
     331  +             for problem_arg in secret_args:
     332  +                 if arg.startswith(problem_arg + "="):
     333  +                     arg = "{0}={1}".format(problem_arg, obfuscation_string)
190  334
191  - def call_registration_cmd(registration_cmd):
192  -     """Wrapper for run_subprocess that avoids leaking password in the log."""
193  -     loggerinst.debug("Calling command '%s'" % hide_password(" ".join(registration_cmd)))
194  -     return utils.run_subprocess(registration_cmd, print_cmd=False)
     335  +         sanitized_list.append(arg)
195  336
     337  +     if hide_next:
     338  +         loggerinst.debug(
     339  +             "Passed arguments had unexpected secret argument,"
     340  +             " '{0}', without a secret".format(sanitized_list[-1])  # lgtm[py/clear-text-l
     341  +         )
196  342
197  - def hide_password(cmd):
198  -     """Replace plaintext password with asterisks."""
199  -     return re.sub('--password=".*?"', '--password="*****"', cmd)
     343  +     return sanitized_list
200  344
201  345
202  346   def replace_subscription_manager():
```

2 convert2rhel/unit_tests/README.md

```
49  49      monkeypatch.setattr(package_handler, "get_packages_to_update",
        value=packages_to_update_mock)
```

```
50  50
51  51      assert package_handler.get_packages_to_update() is not None
52      -   assert packages_to_update_mock.assert_called_once_with(["package-1", "package-2"])
    52  +   packages_to_update_mock.assert_called_once_with(["package-1", "package-2"])
53  53  ```
54  54
55  55  And this other example, of a test that don't need any mocks for external
```

9 convert2rhel/unit_tests/conftest.py

```
2   2
3   3   import pytest
4   4
5       - from convert2rhel import redhatrelease, utils
    5   + from convert2rhel import redhatrelease, toolopts, utils
6   6   from convert2rhel.logger import setup_logger_handler
7   7   from convert2rhel.systeminfo import system_info
8   8   from convert2rhel.toolopts import tool_opts
60  60      setup_logger_handler(log_name="convert2rhel", log_dir=str(tmpdir))
61  61
62  62
    63  + @pytest.fixture
    64  + def global_tool_opts(monkeypatch):
    65  +     local_tool_opts = toolopts.ToolOpts()
    66  +     monkeypatch.setattr(toolopts, "tool_opts", local_tool_opts)
    67  +     return local_tool_opts
    68  +
    69  +
63  70  @pytest.fixture()
64  71  def pretend_os(request, pkg_root, monkeypatch):
65  72      """Parametric fixture to pretend to be one of available OS for convertion.
```

820 convert2rhel/unit_tests/subscription_test.py

Load diff

Large diffs are not rendered by default.

2 convert2rhel/unit_tests/systeminfo_test.py

```
161 161     @unit_tests.mock(logger, "LOG_DIR", unit_tests.TMP_DIR)
162 162     @unit_tests.mock(utils, "run_subprocess", RunSubprocessMocked(("rpmva\n", 0)))
163 163     def test_generate_rpm_va(self):
    164 +       # TODO: move class from unittest to pytest and use global tool_opts fixture
```

```
+        tool_opts.no_rpm_va = False
         # Check that rpm -Va is executed (default) and stored into the specific file.
         system_info.generate_rpm_va()

```

62 convert2rhel/unit_tests/utils_test.py

```
 # You should have received a copy of the GNU General Public License
 # along with this program.  If not, see <https://www.gnu.org/licenses/>.
 import getpass
+import logging
 import os
 import sys
 import unittest
         assert is_rpm_based_os() in (True, False)


+@pytest.mark.parametrize(
+    "command, expected_output, expected_code",
+    (
+        (["echo", "foobar"], "foobar", 0),
+        (["sh", "-c", "exit 56"], "", 56),
+    ),
+)
+def test_run_cmd_in_pty_simple(command, expected_output, expected_code, monkeypatch):
+    output, code = utils.run_cmd_in_pty(command)
+    assert output.strip() == expected_output
+    assert code == expected_code
+
+
+def test_run_cmd_in_pty_expect_script():
+    if sys.version_info < (3,):
+        prompt_cmd = "raw_input"
+    else:
+        prompt_cmd = "input"
+    output, code = utils.run_cmd_in_pty(
+        [sys.executable, "-c", 'print(%s("Ask for password: "))' % prompt_cmd],
+        expect_script=(("password: *", "Foo bar\n"),),
+    )
+
+    assert output.strip().splitlines()[-1] == "Foo bar"
+    assert code == 0
+
+
+@pytest.mark.parametrize(
+    "print_cmd, print_output",
+    (
+        (True, True),
```

```
+         (True, False),
+         (False, True),
+         (False, False),
+     ),
+ )
+ def test_run_cmd_in_pty_quiet_options(print_cmd, print_output, global_tool_opts, caplog):
+     global_tool_opts.debug = True
+     caplog.set_level(logging.DEBUG)
+
+     output, code = utils.run_cmd_in_pty(["echo", "foo bar"], print_cmd=print_cmd, print_o
+
+     expected_count = 1  # There will always be one debug log stating the pty columns
+     if print_cmd:
+         assert caplog.records[0].levelname == "DEBUG"
+         assert caplog.records[0].message.strip() == "Calling command 'echo foo bar'"
+         expected_count += 1
+
+     if print_output:
+         assert caplog.records[-1].levelname == "INFO"
+         assert caplog.records[-1].message.strip() == "foo bar"
+         expected_count += 1
+
+     assert len(caplog.records) == expected_count
+
+
+ def test_run_cmd_in_pty_check_for_deprecated_string():
+     with pytest.raises(TypeError, match="cmd should be a list, not a str"):
+         utils.run_cmd_in_pty("echo foobar")
+
+
  def test_get_package_name_from_rpm(monkeypatch):
      monkeypatch.setattr(utils, "rpm", get_rpm_mocked())
      monkeypatch.setattr(utils, "get_rpm_header", lambda _: get_rpm_header_mocked())
```

68 convert2rhel/utils.py

```
      return output, return_code


- def run_cmd_in_pty(cmd, print_cmd=True, print_output=True, columns=120):
+ def run_cmd_in_pty(cmd, expect_script=(), print_cmd=True, print_output=True, columns=120)
      """Similar to run_subprocess(), but the command is executed in a pseudo-terminal.

      The pseudo-terminal can be useful when a command prints out a different output with o
      session. E.g. yumdownloader does not print the name of the downloaded rpm if not exec
      Switching off printing the command can be useful in case it contains a password in pl

```

```diff
-    :param cmd: The command to execute, including the options, e.g. "ls -al"
-    :type cmd: string
+    :param cmd: The command to execute, including the options as a list, e.g. ["ls", "-al
+    :type cmd: list
+    :param expect_script: An iterable of pairs of expected strings and response strings.
+    these pairs, interactive programs can be scripted.  Example:
+        run_cmd_in_pty(['sudo', 'whoami'], [('password: ', 'sudo_password\n')])
+        Note1: The caller is responsible for adding newlines to the response strings wher
+        needed. Note2: This function will await pexpect.EOF after all of the pairs in exp
+        have been exhausted.
+    :type expect_script: iterable of 2-tuples or strings:
     :param print_cmd: Log the command (to both logfile and stdout)
     :type print_cmd: bool
     :param print_output: Log the combined stdout and stderr of the executed command (to b
     if print_cmd:
         loggerinst.debug("Calling command '%s'" % " ".join(cmd))

-    class PexpectSizedWindowSpawn(pexpect.spawn):
-        # https://github.com/pexpect/pexpect/issues/134
-        def setwinsize(self, rows, cols):
-            super(PexpectSizedWindowSpawn, self).setwinsize(0, columns)
-
-    process = PexpectSizedWindowSpawn(cmd[0], cmd[1:], env={"LC_ALL": "C"}, timeout=None)
-
-    # The setting of window size is super unreliable
-    process.setwinsize(0, columns)
+    process = PexpectSizedWindowSpawn(cmd[0], cmd[1:], env={"LC_ALL": "C", "LANG": "C"},
+    # Needed on RHEL-8+ (see comments near PexpectSizedWindowSpawn definition)
+    process.setwinsize(1, columns)
     loggerinst.debug("Pseudo-PTY columns set to: %s" % (process.getwinsize(),))

+    for expect, send in expect_script:
+        process.expect(expect)
+        process.send(send)
+
     process.expect(pexpect.EOF)
+    try:
+        process.wait()
+    except pexpect.ExceptionPexpect:
+        # RHEL 7's pexpect throws an exception if the process has already exited
+        # We're just waiting to be sure that the process has finished so we can
+        # ignore the exception.
+        pass
+
+    # Per the pexpect API, this is necessary in order to get the return code
+    process.close()
+    return_code = process.exitstatus
+
     output = process.before.decode()
```

```
196  213        if print_output:
197  214            loggerinst.info(output.rstrip("\n"))
198  215
199        -     process.close()  # Per the pexpect API, this is necessary in order to get the return
200        -     return_code = process.exitstatus
201        -
202  216        return output, return_code
203  217
204  218
     219  + # For pexpect released prior to 2015 (RHEL7's pexpect-2.3),
     220  + # spawn.__init__() hardcodes a call to setwinsize(24, 80) to set the
     221  + # initial terminal size. There is no official way to set the terminal size
     222  + # to a custom value before the process starts. This can cause an issue with
     223  + # truncated lines for processes which read the terminal size when they
     224  + # start and never refresh that value (like yumdownloader)
     225  + #
     226  + # overriding setwinsize to set the columns to the size we want in this
     227  + # subclass is a kludge for the issue. On pexpect-2.3, it fixes the issue
     228  + # because of the setwinsize call in __init__() at the cost of never being
     229  + # able to change the column size later.  On later pexpect (RHEL-8 has
     230  + # pexpect-4.3), this doesn't fix the issue of the terminal size being small
     231  + # when the subprocess starts but dnf download checks the terminal's size
     232  + # just before it prints the statusline we care about. So setting the
     233  + # terminal size via setwinsize() after the process is created works (note:
     234  + # there is a race condition there but it's unlikely to ever trigger as it
     235  + # would require downloading a package to happen quicker than the time
     236  + # between calling spawn.__init__() and spawn.setwinsize())
     237  + class PexpectSizedWindowSpawn(pexpect.spawn):
     238  +     # https://github.com/pexpect/pexpect/issues/134
     239  +     def setwinsize(self, rows, cols):
     240  +         super(PexpectSizedWindowSpawn, self).setwinsize(rows, 120)
     241  +
     242  +
205  243    def let_user_choose_item(num_of_options, item_to_choose):
206  244        """Ask user to enter a number corresponding to the item they choose."""
207  245        while True:  # Loop until user enters a valid number
```

4 plans/tier0.fmf

```
5    5      discover+:
6    6        test: basic-sanity-checks
7    7
     8    + /check_cve:
     9    +   discover+:
     10   +     test: check-cve-2022-1662
     11   +
8    12   /check_user_response:
```

```
  discover+:
    test: check-user-response
```

16 tests/ansible_collections/roles/install-testing-deps/tasks/main.yml

```diff
@@ -1,7 +1,11 @@
---
- name: Ensure python3
  yum:
-   name: python3
+   # gcc and python3-devel are needed for psutil
+   name:
+     - "python3"
+     - "gcc"
+     - "python3-devel"
    state: present

- name: Install pip if not present

- name: Install pytest framework dependencies
  pip:
-   name: ["pytest", "pytest-cov", "envparse", "click", "pexpect", "dataclasses", "jsonsc
+   name:
+     - "pytest"
+     - "pytest-cov"
+     - "envparse"
+     - "click"
+     - "pexpect"
+     - "dataclasses"
+     - "jsonschema"
+     - "psutil"
    # Use pip3 in case pip was installed via rpm package on this system
    executable: pip3
```

6 tests/integration/tier0/check-cve-2022-1662/main.fmf

```diff
@@ -0,0 +1,6 @@
+ summary: check cve-2022-1662 is fixed
+
+ tier: 0
+
+ test: |
+   pytest -svv
```

29 tests/integration/tier0/check-cve-2022-1662/test_cve_fixes.py

```
@@ -0,0 +1,29 @@
+ from multiprocessing import Pool
+
+ import psutil
+
+
+ def watchdog():
+     while True:
+         for process in psutil.process_iter():
+             # For some reason the psutil catches subscription-manager in process.name()
+             # as 'subscription-ma', thus using 'subscription' to catch it
+             if "subscription" in process.name():
+                 return process.cmdline()
+
+
+ def test_passing_password_to_submrg(convert2rhel):
+     username = "testname"
+     password = "EXAMPLE&hTYGHKPvU7Ewd"
+     with convert2rhel(f"-y --no-rpm-va -u {username} -p {password}") as c2r:
+         # Just to be sure, try to run through all three tries
+         # of the registration process in case the race condition applies
+         for subscription_try in range(2):
+             c2r.expect("Registering the system using subscription-manager ...")
+             # Run watchdog function using multiprocessing pool
+             # as soon as Convert2RHEL tries to call subscription-manager
+             with Pool(processes=1) as pool:
+                 watcher = pool.apply_async(watchdog, ())
+                 # Check for the password not being passed to the subscription-manager
+                 print(watcher.get())
+                 assert not [cmdline for cmdline in watcher.get() if password in cmdline]
```

2 tests/integration/tier0/check-user-response/test_user_response.py

```
41  41              env.str("RHSM_KEY"),
42  42          )
43  43      ) as c2r:
44  -           c2r.expect_exact(" ... activation key detected: ")
    44  +       c2r.expect_exact("activation key detected")
45  45          c2r.expect_exact("Organization: ")
46  46          c2r.sendline()
47  47          assert c2r.expect_exact(["Organization", "Registering the system"]) == 0
```

0 comments on commit `8d72fb0`

Please sign in to comment.