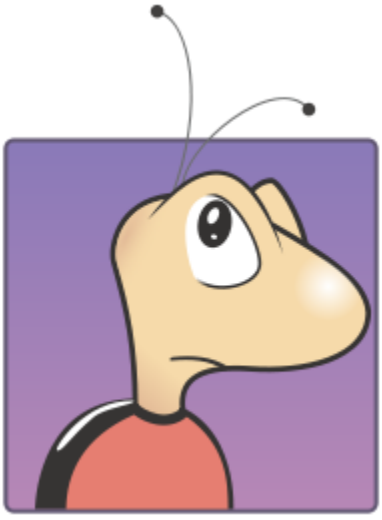
Vulnerability Disclosure



Authentication Mechanism Bypass

In

Bugzilla

Netanel Rubin

06.09.2015

Vulnerable Versions

? – Fully Patched (5.0.0)

Summary

Bugzilla is a very popular open source bug tracking software written in Perl. It allows an organization to easily organize the bugs in its products and developers to conveniently document any bug they've found or fixed.

This vulnerability allows an attacker to register under any email address, causing the system to grant permissions the attacker should not have obtained otherwise.

This vulnerability has been tested and found working on `Bugzilla.mozilla.org` - the Bugzilla for the Mozilla corporation and on `Bugzilla.wikimedia.org` - the Bugzilla for the Wikimedia foundation. On both domains we've been granted permissions that allowed us to view confidential data among other things.

Technical Description

Bugzilla authentication mechanism relies on email validation. Prior to the actual registration the user enters his email and an activation link is sent there by the system. This link contains a token which allows the user to set his real name and his account password. After that every change to the email address requires both the old and new addresses to be validated.

When the user registers with his email address it is saved in the `tokens` DB table along with the token to activate the address. The email address is stored in a special column named *'evendata'*, which supposed to contain the data associated with the token.

When the user validates his email address using the token mailed to him, his email address is fetched from that column and is used to create his account.
After his account is created, the auto-assign group regex is triggered and it validates the user email address against group matching regexes.

This makes the email address of the user an extremely important part in the permissions mechanism in the system, but in order to use a certain address an attacker must validate he owns it using the token sent to that address.

After the system validates the email address entered for dangerous characters, a new registration token is created for it. This can be seen in the following code:

```
sub check_and_send_account_creation_confirmation {
    my ($self, $login) = @_;

       # Check the email address of the user
    $login = $self->check_login_name($login);
    if ($login !~ /$creation_regexp/i) {
        ThrowUserError('account_creation_restricted');
    }

    # Create and send a token for this new account.
    require Bugzilla::Token;
    Bugzilla::Token::issue_new_user_account_token($login);
}
```

Bugzilla/User.pm::check_and_send_account_creation_confirmation()

Following the code to *'issue_new_user_account_token()'*:

```
sub issue_new_user_account_token {
    my $login_name = shift; # The email address we entered
    my $template = Bugzilla->template;
    my $vars = {};

      # Create the new token in the DB
    my ($token, $token_ts) = _create_token(undef, 'account', $login_name);

      # Set the email address, expiration date, and token as the email message
variables
      # (Bugzilla->params->{'emailsuffix'} is empty by default [and in almost all
installations])
    $vars->{'email'} = $login_name . Bugzilla->params->{'emailsuffix'};
    $vars->{'expiration_ts'} = ctime($token_ts + MAX_TOKEN_AGE * 86400);
    $vars->{'token'} = $token;

      # Render the email message using the complied variables
    my $message;
    $template->process('account/email/request-new.txt.tmpl', $vars, \$message)
      || ThrowTemplateError($template->error());

      # Send the confirmation email
    MessageToMTA($message);
}
```

Bugzilla/Token.pm::issue_new_user_account_token()

Looking at this code we realize the token for our user registration is created in the DB and then inserted into the email message sent to our address.
It is important to note that the system doesn't validate our token has been created correctly in the database. Instead, it just assumes it did and sends us the confirmation email.

We need to find a way to corrupt the email address as it is being inserted into the DB. The column the address is stored in, *'eventdata'*, is of type *'tinytext'*. *This* data type represents a text string and is limited to 255 bytes in MySQL. For other DBs not supporting this data type Bugzilla specifies 255 bytes as the exact size of the column using the regular '*text'* data type.

But what happens when we insert more than 255 bytes? Does the DB raise an exception? Does it crash? No – It automatically truncates the data so it fits the column size.
We can use this behavior to create an email address larger than 255 bytes in a way that will cause the system to miss identify our domain (the only part we cannot entirely control).

We will use an address ending with the domain we want so it is exactly 255 bytes in length. Then, we will append our owned domain to that address so the email will be sent to us.
When the email will be inserted into the DB, it will be truncated, so the application will assume we identified an email under the domain we actually wanted, causing a privilege escalation.

POC

```
POST /createaccount.cgi HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: 76

login=[MALICIOUS_EMAIL_ADDRESS]&token=[CSRF_TOKEN]
```

Suggested Fix

Make sure the length of the email address doesn't exceed the size of the DB column.