Weak Password Recovery Mechanism in InvoicePlane CRM

Mar 16, 2021

Summary

InvoicePlane is one of the popular open-source CRM. During the search for a PHP based open-source CRM in Github, this comes mostly within first ten.

The latest version of InvoicePlane (v1.5.11) has a weak password recovery mechanism which could lead to authenticated remote code execution. Without further wasting your time let's dive into the details.

- CWE-640: Weak Password Recovery Mechanism for Forgotten Password
- CWE-770: Allocation of Resources Without Limits or Throttling

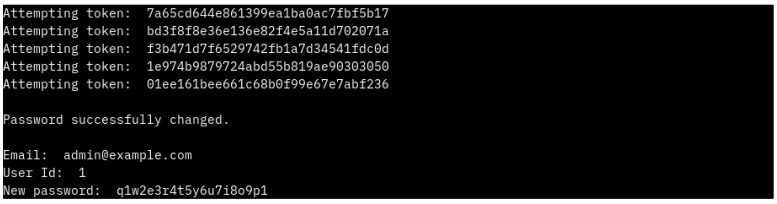
The application allows users to reset their passwords by sending a reset link via email. As usual, the application allows the user to reset a new password by visiting the link. Here the things seem to be normal on the surface, but when we dive into source code things are different.

Proof of Concept

The password reset feature doesn't have any rate-limiting mechanism implemented to prevent brute force. When a user requests a password reset, the application generates a reset token stores it in the database. The problem is, during this token generation application concatenates the current time and email to generate an MD5 hash then uses this generated hash as a token. Since there is no rate limiting, if the attacker has the users' email address predicting this token becomes so easy. During the password generation, through the response date header attacker can figure out the time value as well.

```
168             // Test if a user with this email exists
169             if ($this->db->where('user_email', $email)) {
170                 // Create a passwordreset token
171                 $email = $this->input->post('email');
172                 $token = md5(time() . $email);
173
174                 // Save the token to the database and set the user to inactive
175                 $db_array = array(
176                     'user_passwordreset_token' => $token,
177                 );
178
179                 $this->db->where('user_email', $email);
180                 $this->db->update('ip_users', $db_array);
```

As seen on the above image, at line 172 application concatenates the current server time and user email to generate a token.



```
Attempting token:  7a65cd644e861399ea1ba0ac7fbf5b17
Attempting token:  bd3f8f8e36e136e82f4e5a11d702071a
Attempting token:  f3b471d7f6529742fb1a7d34541fdc0d
Attempting token:  1e974b9879724abd55b819ae90303050
Attempting token:  01ee161bee661c68b0f99e67e7abf236

Password successfully changed.

Email:  admin@example.com
User Id:  1
New password:  q1w2e3r4t5y6u7i8o9p1
```

The above image shows the successful password change using the exploit.

notnnor
notnnor@gmail.com