GLPI < 9.5.5 Stored Cross-Site Scripting (XSS)

Stealing Privileged Accounts

What's GLPI?

GLPI is an incredible ITSM software tool that helps you plan and manage IT changes in an easy way, solve problems efficiently when they emerge and allow you to gain legitimate control over your company's IT budget, and expenses.

Many companies use GLPI to manage their clients and tickets. GLPI has different kind of users. To manage plugins the user needs to has a tech role. So if you steal this kind of account you can go to administration page and dump the whole database to clone in your local lab.

The Vector

During the plugin installation process, GLPI reads the setup.php file from plugins to show informations as author, plugin version, license and etc...

The problem here is simple, they do not validate and sanitize these informations before render to the user. Trust the user input is never a good idea.

GLPI loads the plugin using the setup.php file, as we can see in this piece of code.

glpi > inc > plugin.class.php

```
243      * @return void
244      **/
245     static function load($plugin_key, $withhook = false) {
246        global $LOADED_PLUGINS;
247
248        $loaded = false;
249        foreach (PLUGINS_DIRECTORIES as $base_dir) {
250           if (!is_dir($base_dir)) {
251              continue;
252           }
253
254           if (file_exists("$base_dir/$plugin_key/setup.php")) {
255              $loaded = true;
256              $plugin_directory = "$base_dir/$plugin_key";
257              include_once("$plugin_directory/setup.php");
258              if (!in_array($plugin_key, self::$loaded_plugins)) {
259                 self::$loaded_plugins[] = $plugin_key;
260                 $init_function = "plugin_init_$plugin_key";
261                 if (function_exists($init_function)) {
262                    $init_function();
263                    $LOADED_PLUGINS[$plugin_key] = $plugin_directory;
264                    self::loadLang($plugin_key);
265                 }
266              }
267           }
268           if ($withhook) {
269              self::includeHook($plugin_key);
270           }
271
272           if ($loaded) {
273              break;
274           }
275        }
276     }
```

Fig.1 - Load Plugins

```
20  function plugin_version_dashboard(){
21          global $DB, $LANG;
22
23          return array('name'                 => __('Dashboard','dashboard'),
24                                  'version'                   => '1.0.3',
25                                  'author'                      => '<a href="https://plugins.glpi-project.org/#/plugin/dashboard"> Stevenes Donato </b> </a>',
26                                  'license'                   => 'GPLv2+',
27                                  'homepage'                  => 'https://plugins.glpi-project.org/#/plugin/dashboard',
28                                  'minGlpiVersion'        => '9.4'
29                                  );
30  }
31
```

Fig.2 - The setup.php file from plugins.

Plugins need to have a function called plugin_version_NAMEOFPLUGIN(). This function will return the array that contains all informations as version, author, license, homepage and the min GLPI version. We saw this function at Fig.2.

```
1488      */
1489      public function getInformationsFromDirectory($directory) {
1490
1491         $informations = [];
1492         foreach (PLUGINS_DIRECTORIES as $base_dir) {
1493            if (!is_dir($base_dir)) {
1494               continue;
1495            }
1496            $setup_file  = "$base_dir/$directory/setup.php";
1497
1498            if (file_exists($setup_file)) {
1499               // Includes are made inside a function to prevent included files to override
1500               // variables used in this function.
1501               // For example, if the included files contains a $plugin variable, it will
1502               // replace the $plugin variable used here.
1503               $include_fct = function () use ($directory, $setup_file) {
1504                  self::loadLang($directory);
1505                  include_once($setup_file);
1506               };
1507               $include_fct();
1508               $informations = Toolbox::addslashes_deep(self::getInfo($directory));
1509
1510               // plugin found, don't parse others directories
1511               break;
1512            }
1513         }
1514
1515         return $informations;
1516      }
1517
```

Fig.3 - Calling GetInfo()

```
glpi > inc > plugin.class.php
1462      * @return string|array The specific information value requested or an array of all information if $info is null.
1463      **/
1464      static function getInfo($plugin, $info = null) {
1465
1466         $fct = 'plugin_version_'.strtolower($plugin);
1467         if (function_exists($fct)) {
1468            $res = $fct();
1469            if (!isset($res['requirements']) && isset($res['minGlpiVersion'])) {
1470               $res['requirements'] = ['glpi' => ['min' => $res['minGlpiVersion']]];
1471            }
1472         } else {
1473            Toolbox::logError("$fct method must be defined!");
1474            $res = [];
1475         }
1476         if (isset($info)) {
1477            return (isset($res[$info]) ? $res[$info] : '');
1478         }
1479         return $res;
1480      }
1481
```

Fig.4 - Getting Informations

To display these informations to the user, Glpi uses /front/plugin.php that calls view.class.php. As the following image shows, there is no checking to escape html/javascript code. All information is concatened directly to HTML.

```
glpi > inc > marketplace > view.class.php
447     */
448    static function getPluginCard(array $plugin = [], string $tab = "discover"):string {
449       $plugin_key    = $plugin['key'];
450       $plugin_inst   = new Plugin;
451       $plugin_inst->getFromDBbyDir($plugin_key);
452       $plugin_state = Plugin::getStateKey($plugin_inst->fields['state'] ?? -1);
453       $buttons      = self::getButtons($plugin_key);
454
455       $authors = implode(', ', array_column($plugin['authors'] ?? [], 'name', 'id'));
456       $authors_title = Html::clean($authors);
457       $authors = strlen($authors)
458          ? "<i class='fas fa-fw fa-user-friends'></i>$authors"
459          : "";
460       $licence = is_string($plugin['license']) && isset($plugin['license']) && strlen($plugin['license'])
461          ? "<i class='fas fa-fw fa-balance-scale'></i>{$plugin['license']}"
462          : "";
463       $version = strlen($plugin['version'] ?? "")
464          ? "<i class='fas fa-fw fa-code-branch'></i>".$plugin['version']
465          : "";
466       $stars = ($plugin['note'] ?? -1) > 0
467          ? self::getStarsHtml($plugin['note'])
468          : "";
469
470       $home_url = strlen($plugin['homepage_url'] ?? "")
471          ? "<a href='{$plugin['homepage_url']}' target='_blank' >
472             <i class='fas fa-home add_tooltip' title='".__s("Homepage")."'></i>
473             </a>"
474          : "";
475       $issues_url = strlen($plugin['issues_url'] ?? "")
476          ? "<a href='{$plugin['issues_url']}' target='_blank' >
477             <i class='fas fa-bug add_tooltip' title='".__s("Get help")."'></i>
478             </a>"
479          : "";
480       $readme_url = strlen($plugin['readme_url'] ?? "")
481          ? "<a href='{$plugin['readme_url']}' target='_blank' >
482             <i class='fas fa-book add_tooltip' title='".__s("Readme")."'></i>
483             </a>"
484          : "";
```

Fig.5 - Display

Here we can see the alert being executed.

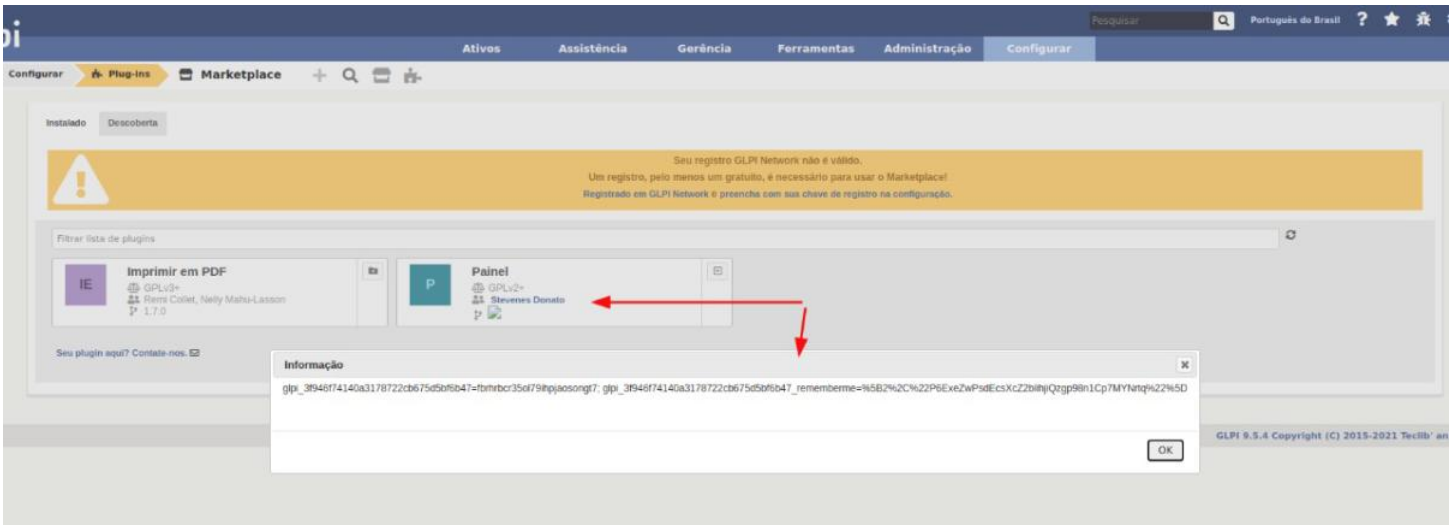

Fig.6 - Alert executed

References

1. Dashboard Plugin
2. GLPI 9.5.4