Hacking the Oce Colorwave printer: when a quick security assessment determines the success of a Red Team exercise.

Back in September 2019, as Red Timmy Security group, we have been involved in a Red Team exercise. We had to simulate the scenario of a malicious insider plugging a Raspberry Pi device in to the network to potentially use as a C&C, and to check how much time the guys monitoring the environment would have spent to detect it. Furthermore, the place where to hide our device had to be tricky enough to spot, with the aim to pour a pinch of extra pepper on the challenge against the blue team.

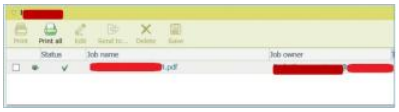
Consequently, we did not want simply to drop the device at the first location available, raising suspicions on other employees who were unaware of our real role and increasing the risk to be caught. Therefore the initial couple of days following the job engagement we behaved normally: we have been assigned a company laptop, a badge and completed all the on-boarding tasks, which are typical of when somebody is hired for real. Oh well, almost. The second day, after opening the browser of our new shining company laptop, we mistakenly typed in the number "1" in the URL bar, and discovered that some IP addresses were already visited in the chronology.

We selected the first URL in the list (a local IP address) and visited that. It was the web interface of the Océ Colorwave 500 printer, a big boy weighting more than 250 kilograms (see picture below).

Why should not we give a quick look at it? I guess you know how it went. We did a mini assessment of the printer that lasted a couple of hours. That was enough to discover five different flaws. However, for the matter of this story, only one was relevant. We provide the details of the rest at the end of this post.

Basically we discovered that the Oce Colorwave 500 printer was vulnerable to an authentication bypass bug on the page "*http://<printer-IP>/home.jsp*". When that page is browsed by an anonymous (unauthenticated) user clicking on the "*View All*" button with the intention to visualize the current job queue, the application returns a login panel asking for administrative credentials.

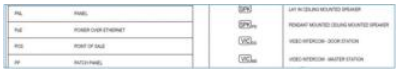
However it was sufficient to append the parameter "`openSI=[SI]`" to the "`home.jsp`" page (with "`[SI]`" being the name of a network user) to view the printer inbox of that user and all the relative print jobs, so completely skipping the authentication process.



Currently the name of the network user we wanted to spy the inbox of was nothing more than its Windows active directory username. The convention used internally by the target company was easy enough to understand (a combination of the user first name and surname, and of course AD was "publicly queryable"), that was really a joke to script an automated process collecting all the printed documents from the device's queue for subsequent manual analysis.

We did not need to poll the device frequently, just once per user, as the queue of the printed jobs was configured (a default setting of the printer we guess) so to be shred only every 2 weeks. This would have given us enough time to download everything submitted to the printer by anyone in the company that had used it in the last 14 days. After finishing the task, we started to manually analyze all the documents collected. A tedious but rewarding task at the end!

We were indeed lucky enough to find the blueprints/security plants of the customer building, including the space hosting us. At the beginning everything was quite tough to read as the blueprints were full of meaningless labels and symbols. But in the middle of the set of documents exfiltrated from the printer spool was also found a legend with the description of each symbol. Something similar to this:



After taking confidence with it, we knew the exact dislocation of desks, security cameras, conference rooms, security equipment including card readers, local alarms / sounders, electromagnetic locks, intercoms, patch panels, network cables, etc... in the building!

LATEST BLOG POSTS

IoT/ICS Armageddon: hacking devices like there's no tomorrow (part 1)

Challenges in the always moving cloud

The thin line between the cloud provider and the customer applications

When a Denial of Service matters: fighting with risk assessment guys

Bug bounty failure stories to learn from: how we ended up to hack a bank with no reward

Snooping on proprietary protocols with Frida

Fortinet SIEM vulnerability allows us to get RCE on internet exposed hosts

Critical Information Disclosure on WP Courses plugin exposes private course videos and materials [CVE-2020-26876]

Pulse Secure Windows Client <9.1.6 (CVE-2020-13162) - exploit

A Tale of Escaping a Hardened Docker container

BLOG ARCHIVE

See all posts

ABOUT THIS SITE

This may be a good place to introduce yourself and your site or include some credits.

It was the turning point of our exercise. The place we found for our Raspberry Pi device and what happened later is another story.

Most importantly, soon after the closure of the red team operation, the e-shredding functionality has been activated and applied for all print-jobs and the device immediately isolated from the network, put behind a firewall in a separate VLAN.

We have contacted the vendor of course and communicated our findings. After few days the vendor has replied informing us of the existence of a newer firmware version compared to the one running on the printer tested by us (that was 4.0.0.0). As no CVEs have been registered for these bugs in the past, we have decided to do that.

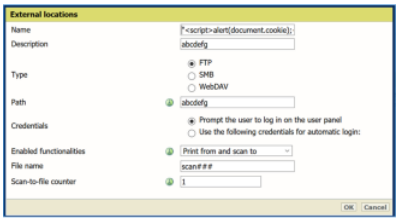
In total we have registered five CVE(s). All these bugs (two Reflected XSS, one Stored XSS and a systemic CSRF), including the authentication bypass already described (CVE-2020-10669), have been discovered by Giuseppe Cali with the contribute of Marco Ortisi. Below you can find the additional details for each of them.

CVE-2020-10667

The web application exposed by the Canon Oce Colorwave 500 printer (firmware version 4.0.0.0 and possibly below) is vulnerable to Stored XSS in "`/TemplateManager/indexExternalLocation.jsp`". The vulnerable parameter is "`map(template_name)`".

An external location is first created by browsing the URL "*http://<printer-IP>/TemplateManager/editTemplate.jsp?type=ExternalLocation*". Here the "`Name`" parameter (actually the "`map(template_name)`" parameter in the POST request generated after clicking the "*Ok*" button) is injected with the following sample JavaScript payload:

```
"<script>alert(document.cookie);</script>
```



After creating the external location template, any attempts to edit it will trigger the sample payload.

CVE-2020-10668

The web application exposed by the Canon Oce Colorwave 500 printer (firmware version 4.0.0.0 and possibly below) is vulnerable to Reflected XSS in "`/home.jsp`". The vulnerable parameter is "`openSI`". Browsing the URL:

http://<printer-IP>/home.jsp?openSI=</script><script>alert(document.cookie);</script>

an alert box will pop-up in the page, showing that the attacker's payload has been executed in the context of the user's browser session.

CVE-2020-10670

The web application exposed by the Oce Colorwave 500 printer (firmware version 4.0.0.0 and possibly below) is vulnerable to Reflected Cross-Site Scripting in "`/SettingsEditor/settingDialogContent.jsp`". The vulnerable parameter is "`settingId`". Browsing the URL:

http://<printer-IP>/SettingsEditor/settingDialogContent.jsp?settingId=

An alert box will pop-up in the page, showing that the attacker's payload has been executed in the context of the user's browser session.

CVE-2020-10671

The web application of the Canon Oce Colorwave 500 printer (firmware version 4.0.0.0 and possibly below) is missing any form of CSRF protections. This is a system-wide issue. An attacker could perform administrative actions by targeting a logged-in administrative user.

That'all for today. As usual we remind you that we will be at Blackhat Las Vegas with our Practical Web Application Hacking Advanced Course on 1-2 August and 3-4 August 2020.

authentication bypass | CSRF | Océ colorwave 500 | printer hacking | Reflected XSS | Stored XSS

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name *

Email *

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

Submit