Jakub Brzozowski (/)

Pentester, bug bounty hunter and security researcher. Also huge fan of Star Wars and coffee connaisseur ☕

(https://twitter.com/redfr0g_) (https://www.linkedin.com/in/heregoesusername/) (https://github.com/redfr0g)

The (in)secure story of OctoPrint

11 MAY 2021 » WEB-APPLICATIONS (/CATEGORY/WEB-APPLICATIONS)

TL;DR

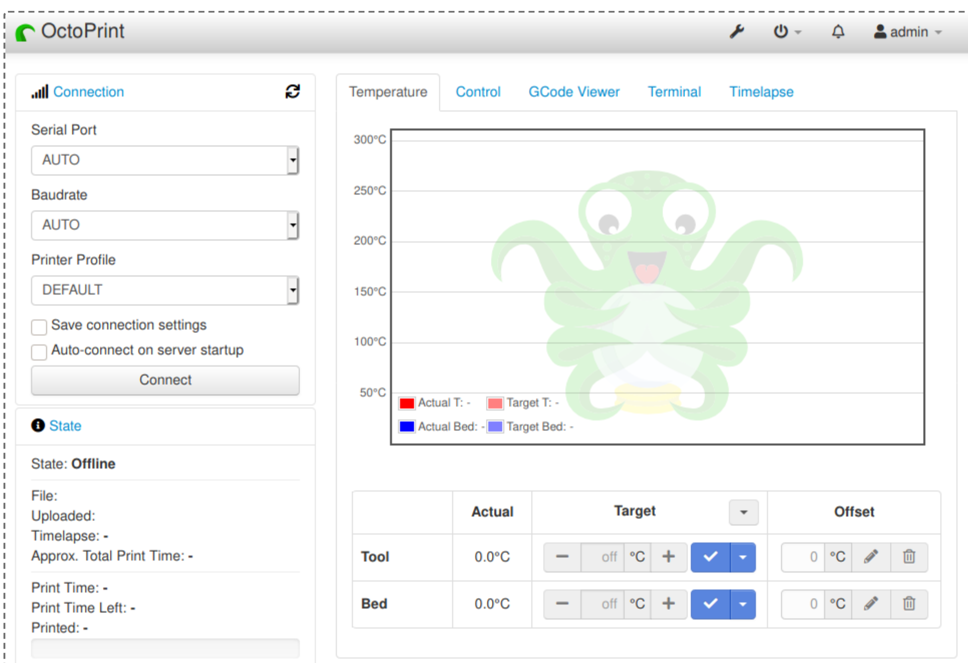
When an attacker gains access to publicly exposed and insecured OctoPrint panel he can execute commands on the host due to the way the application is designed. In order to secure your OctoPrint instances **NEVER** expose the host to the public access and if you have to, use VPN or other access control to secure it.

The application

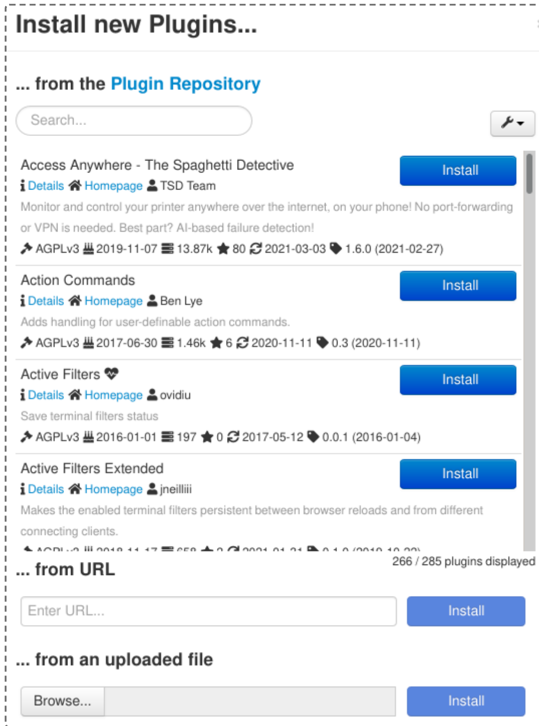
One time when I was browsing github in a quest for finding an interesting software to fiddle a little bit with, I came across an application named OctoPrint. If you're into 3D printing this name may ring a bell with you. That is because OctoPrint is a software that brings a nice and responsive web interface to your 3D printer, and from what I have deducted, it is pretty popular among the 3D printing community. In fact there is a ton of features that come along with this software such as project files upload, live print monitoring and even custom plugin installation. As you probably imagine this sounded as a juicy app to put on the workbench.

First blood

Nice thing about OctoPrint is that it can be installed as a Python package, so the installation boils down to simple `pip install OctoPrint` . After the installation the web interface can be accessed on the default port 5000. Initial configuration allows us to set administrator password and set variables for 3D printer (which I obviously didn't own at the moment of testing the app).



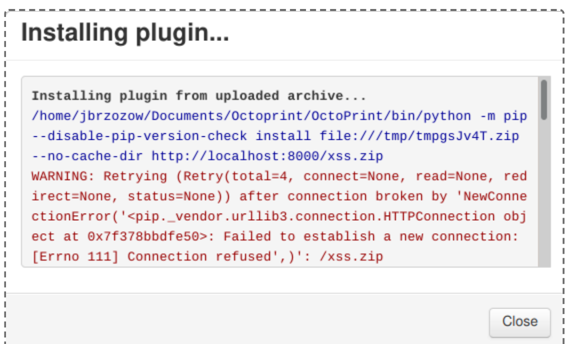
After logging in to a privileged account (Administrator or Operator) there is quite a lot of web interface functionalities to fiddle with. What caught my attention was that the application allows you to install custom plugins as Python packages right from the web interface.



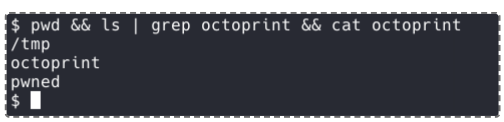
How dangerous installation of untrusted packages can be, was very nicely describen by **Aryx** in his blog (https://www.ayrx.me/look-before-you-pip) (not mentioning the package confusion vulnerabilities). By leveraging this we can easily modify an existing plugin and turn it into a malicious one. Then we can use it to execute system commands using Python `os` library. By adding the following code to the `setup.py` file we can run commands on the host.

```
import os
os.system('echo "pwned" > /tmp/octoprint')
```

Then we have to upload the plugin.



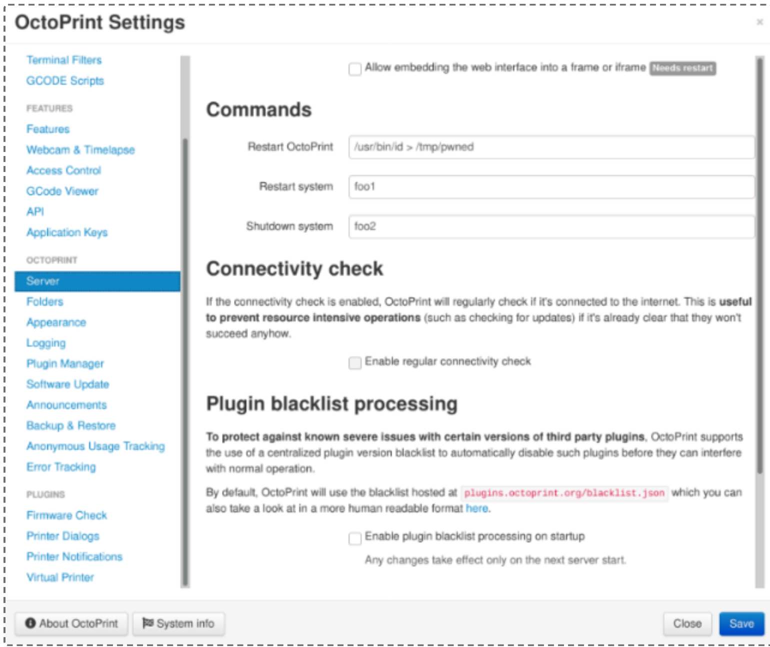
And we have confirmed RCE.



So by altering plugin code we can achieve RCE on the host but this behaviour is quite obvious and is unavoidable when installing packages without any encryption on integrity check.

Kill two birds with one stone

Another interesting functionality is that administrators can specify what commands can be run on the host, in order to reboot, or shut down the operating system, or to restart an OctoPrint instance.



These commands are executed using Python `subprocess` with shell argument set to `True` as can be seen on the following code snippet.

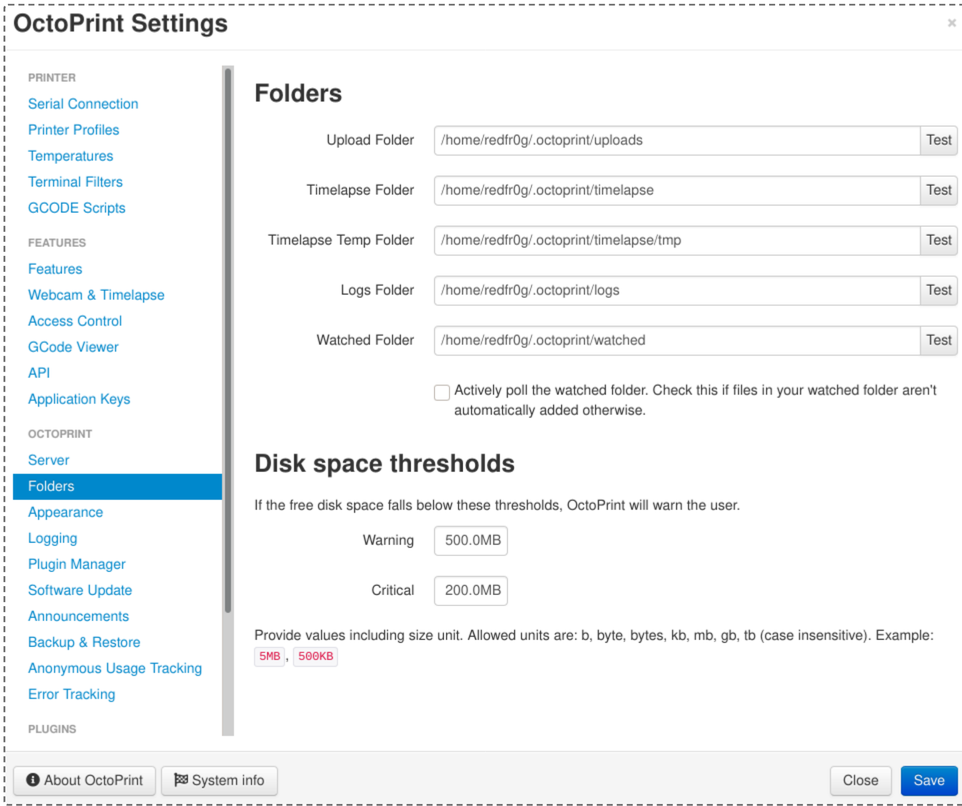
```
try:

      def execute():
          # we run this with shell=True since we have to trust whatever
          # our admin configured as command and since we want to allow
          # shell-alike handling here...
          p = sarge.run(
              command_spec["command"],
              close_fds=CLOSE_FDS,
              stdout=sarge.Capture(),
              stderr=sarge.Capture(),
              shell=True,
          )
```

This means that all commands passed to this function will be executed just as they would be typed in the terminal on host (more or less). Adding that such execution is triggered by a simple POST request to an API endpoint, with just a few lines of code we can write an exploit that executes reverse shell payload on the host.

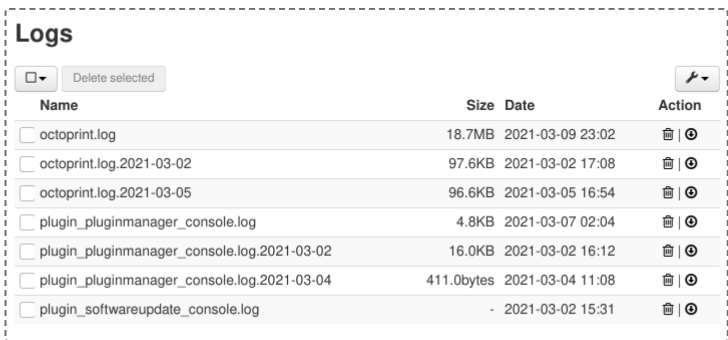
This obviously poses security risk but to exploit this functionalities one must have `Administrator` or `Operator` privileges in the application. From attacker perspective this can be obtained by bypassing authentication or taking over an already created account.

Show me the logz

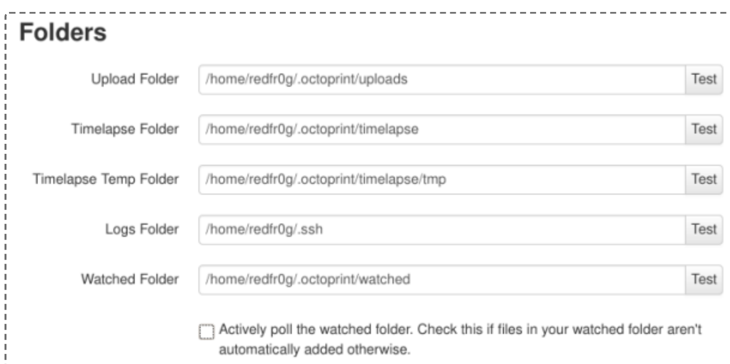
It is nice to be able to debug stuff right from the web interface and OctoPrint gives us this opportunity to not only download and manage log files from the browser, but also to specify the exact path where the files are saved. By using OctoPrint interface we can configure all the logging paths.

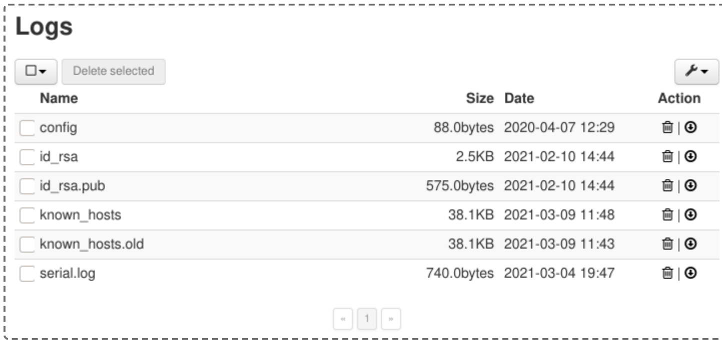


And list or download all the log files in the specified directory.



Interestingly, if the path is changed to any writable path, the application will gladly include all files in the directory and list them in the web interface. This wouldn't be too bad at all however user can not only view the files but also download them. Considering this, it is possible to change the log folder to i.e. `~/.ssh/` directory and get access to SSH private keys.



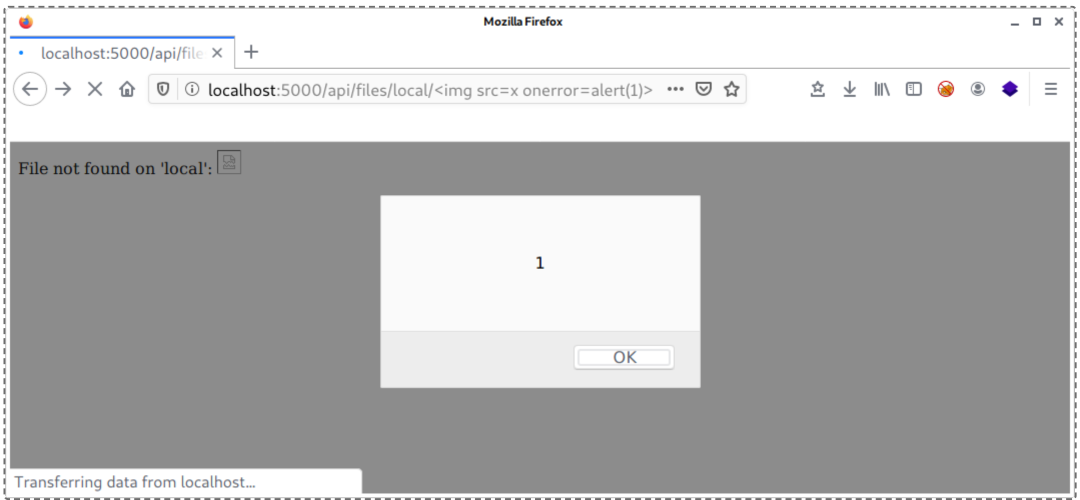


If the host has SSH service running we could gain a stealthy and persistent access this way.

Not existing path is an existing problem

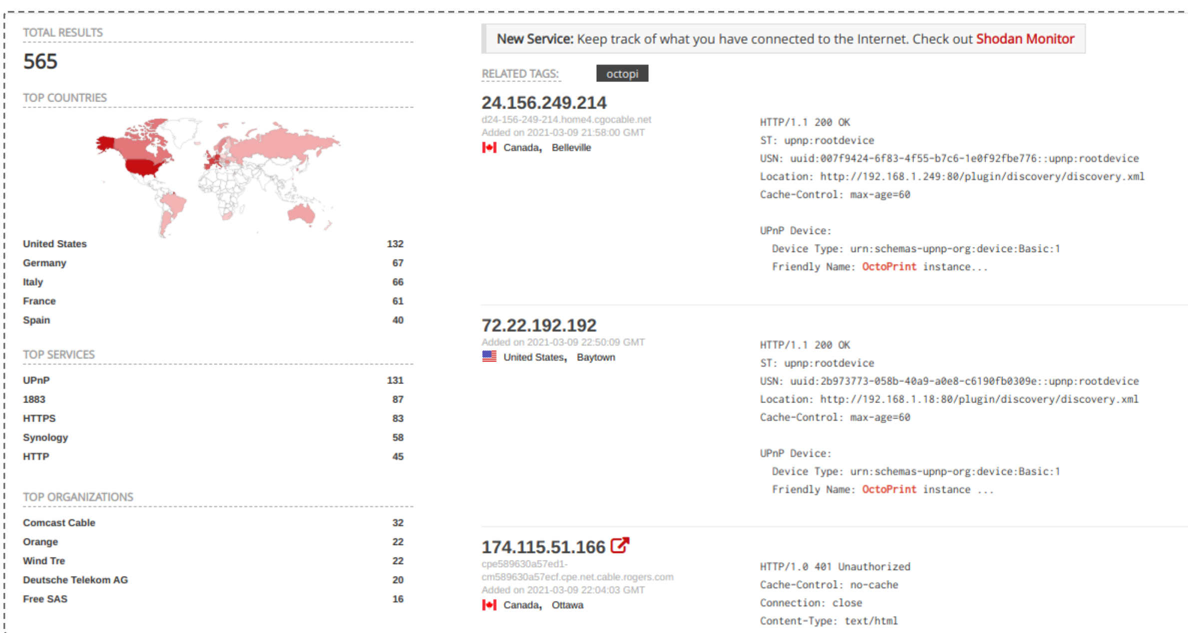
After some fuzzing I eventually found a reflected XSS in the `/api/files` endpoint. The web server was behaving weird when a non existing file was requested or upon making a HTTP DELETE request to it. The path was echoed back together with an error message. The coffin to the nail was that the path was URL decoded which meant that injecting any type of bracket to the document body is possible. Simple `img` payload did the trick.

```
<img src=x onerror=alert(1)>
```



Summary and security reminder

When considering the severity of the above vulnerabilities we must keep in mind that OctoPrint software was never meant to be exposed to the web as it integrates with physical device that is 3D printer. The dangers of such integration were thoroughly described by **@xme** in the following article (https://isc.sans.edu/forums/diary/3D+Printers+in+The+Wild+What+Can+Go+Wrong/24044/). At the time of writing this article over 500 hosts that can be identified as OctoPrint interfaces were listed on Shodan (though this number is slowly decreasing it is still a large number of potential targets).



For maximum security never expose OctoPrint interface to the internet and deploy it locally on a host physically separated from your network i.e. Raspberry Pi. If the remote access is needed always make sure that the application is only accessible via VPN with strong security controls. If you want live preview or control over the printing consider using native OctoPrint plugins such as Telegram bot (https://github.com/fabianonline/OctoPrint-Telegram) for monitoring the printing process.

CVEs

- CVE-2021-32561 - Reflected Cross-Site Scripting in the /api/files
- CVE-2021-32560 - Local File Read

Timeline

- 2.03.2021 - Vulnerabilities were reported to OctoPrint team
- 2.03.2021 - Received OctoPrint team response
- 3.03.2021 - CVEs were reserved for the identified vulnerabilities
- 27.04.2021 - Patch 1.6.0 was released addressing identified vulnerabilities
- 28.04.2021 - CVEs were published
- 10.05.2021 - The article describing identified vulnerabilites was released

References:

1. https://isc.sans.edu/forums/diary/3D+Printers+in+The+Wild+What+Can+Go+Wrong/24044/
2. https://www.ayrx.me/look-before-you-pip
3. https://docs.octoprint.org/en/master/
4. https://docs.octoprint.org/en/master/

Related Posts

- Yeswehack's Dojo #17 XSS challenge writeup (https://www.brzozowski.io/web-applications/2022/06/06/yeswehack-dojo-17-challenge-writeup.html) (Categories: web-applications (/category/web-applications))
- Intigriti's April XSS challenge writeup (https://www.brzozowski.io/web-applications/2022/04/24/intigritis-april-xss-challenge-writeup.html) (Categories: web-applications (/category/web-applications))
- Hacking into NAS (https://www.brzozowski.io/web-applications/2021/06/21/hacking-into-nas.html) (Categories: web-applications (/category/web-applications))
- Actually useful XSS trick (https://www.brzozowski.io/web-applications/2021/04/20/actually-useful-xss-trick.html) (Categories: web-applications (/category/web-applications))