
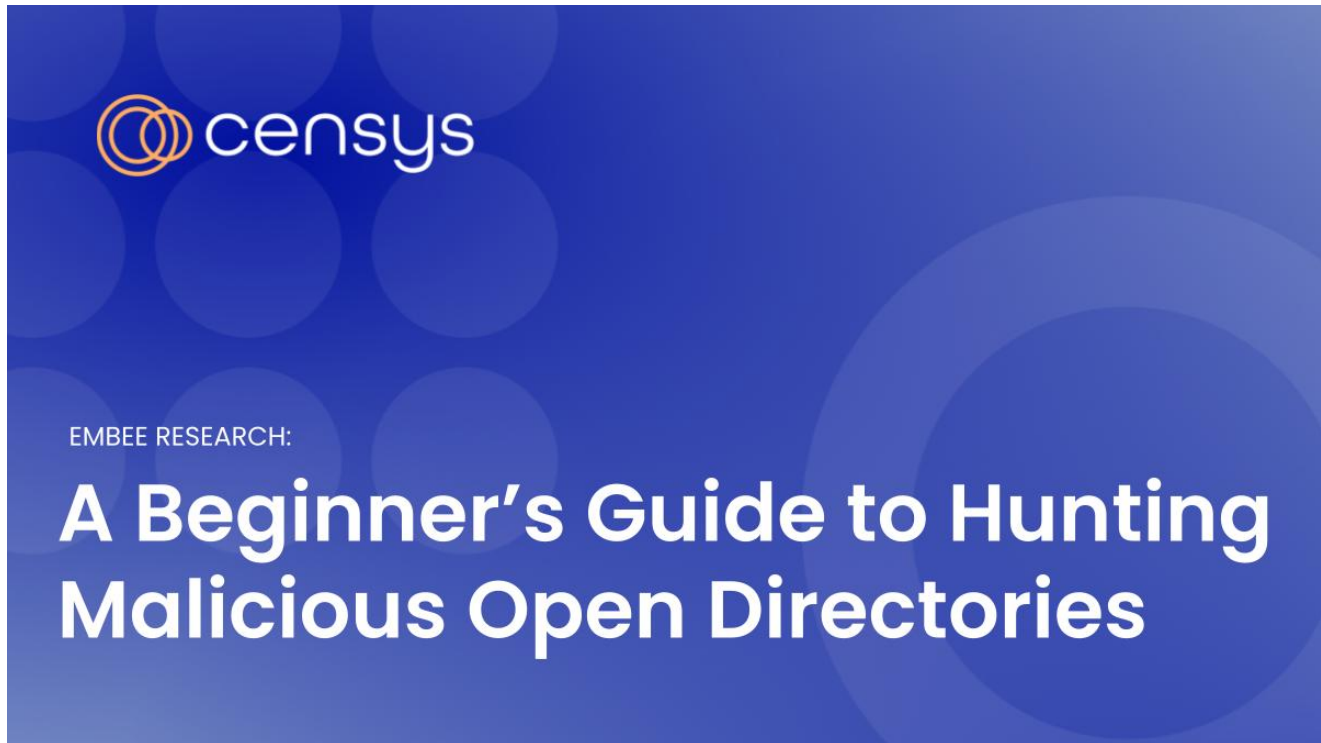


A Beginner's Guide to Hunting Malicious Open Directories

 censys.com/a-beginners-guide-to-hunting-open-directories/

July 22, 2024



Introduction

Threat analysts investigating malicious infrastructure are likely to encounter “open directories” during their investigations. These directories, commonly referred to as “opendirs” are openly accessible servers where threat actors host malicious files related to their operations.

An open directory is a simple concept that many will be familiar with. Despite this, there is little public documentation regarding their discovery and how to identify and track new open directory infrastructure.

This blog will cover the basics of an open directory, how you can discover them during hunting, and how to further your investigations into open directory infrastructure.

What Is An Open Directory?

An open directory is simply a server where a directory has been left “open” and is publicly accessible by browsing to the IP or domain of the site.

From the threat actors' perspective, this directory enables malicious files to be easily accessible and delivered when and where they are needed. This is often second stage files for malware, or tooling used in hands-on operations.

There are legitimate use cases as well, where a legitimate service needs to make a file publicly and easily accessible, but for today, we will focus on malicious use cases and how to separate them from genuine examples.

Below (shared by [RussianPanda9XX](#) on X/Twitter) is one malicious example where an open directory is hosting malicious files. This example shows an Apache-based open directory when viewed directly in a browser.



Another example is an open directory used by ValleyRat and reported by [Zscaler](#).

The open directory is utilising the [HFS](#) (HTTP File Server) software.

The appearance differs between software, but the functionality remains the same. Here is another example of slight differences between Apache and Python. These differences are covered in more detail in the "[Dorking The Internet](#)" Report by Censys.

← → ↻ Not secure 101.33.117.200

User

Login

Folder

Home

0 folders, 5 files, 3.4 MBytes

Search

 go

Select

All Invert Mask

0 items selected

Actions

Archive Get list

Server information

HttpFileServer 2.3m
 Server time: 2024/3/27 17:15:07
 Server uptime: 07:57:26

Name	.extension	Size	Timestamp	Hits
<input type="checkbox"/>	Client.exe	897.2 KB	2024/3/11 0:47:52	67
<input type="checkbox"/>	NTUSER.DXM	196.0 KB	2024/3/25 8:35:04	58
<input type="checkbox"/>	WINWORD2013.EXE	1.8 MB	2021/2/8 2:54:44	56
<input type="checkbox"/>	wwlib.dll	316.0 KB	2024/3/25 8:00:33	10
<input type="checkbox"/>	xig.ppt	215.0 KB	2024/3/25 8:31:47	55

Different Web Servers Render Slightly Different Outputs

Index of /

Name	Last modified	Size	Description
1GB.dat	2023-05-02 08:28	1.0G	
1MB.dat	2023-05-02 08:26	1.0M	
50kb.dat	2023-05-02 08:26	50K	
500kb.dat	2023-05-02 08:26	500K	
bak_index.html	2023-01-11 09:59	27	
cacti/	2021-10-27 14:08	-	
index.nginx-debian.html	2022-08-23 07:11	612	
odir-walker	2023-05-02 08:15	7.0M	
stuff/	2023-05-02 12:14	-	
test/	2023-05-03 09:13	-	
things_and_stuff	2023-05-02 13:26	0	

Apache-based directory listing
(file sizes are prettified)

Directory listing for /

- [1GB.dat](#)
- [1MB.dat](#)
- [500kb.dat](#)
- [50kb.dat](#)
- [bak_index.html](#)
- [cacti/](#)
- [index.nginx-debian.html](#)
- [odir-walker](#)
- [stuff/](#)
- [test/](#)
- [things_and_stuff](#)

Python-based directory listing
(note: no last-modification timestamp)

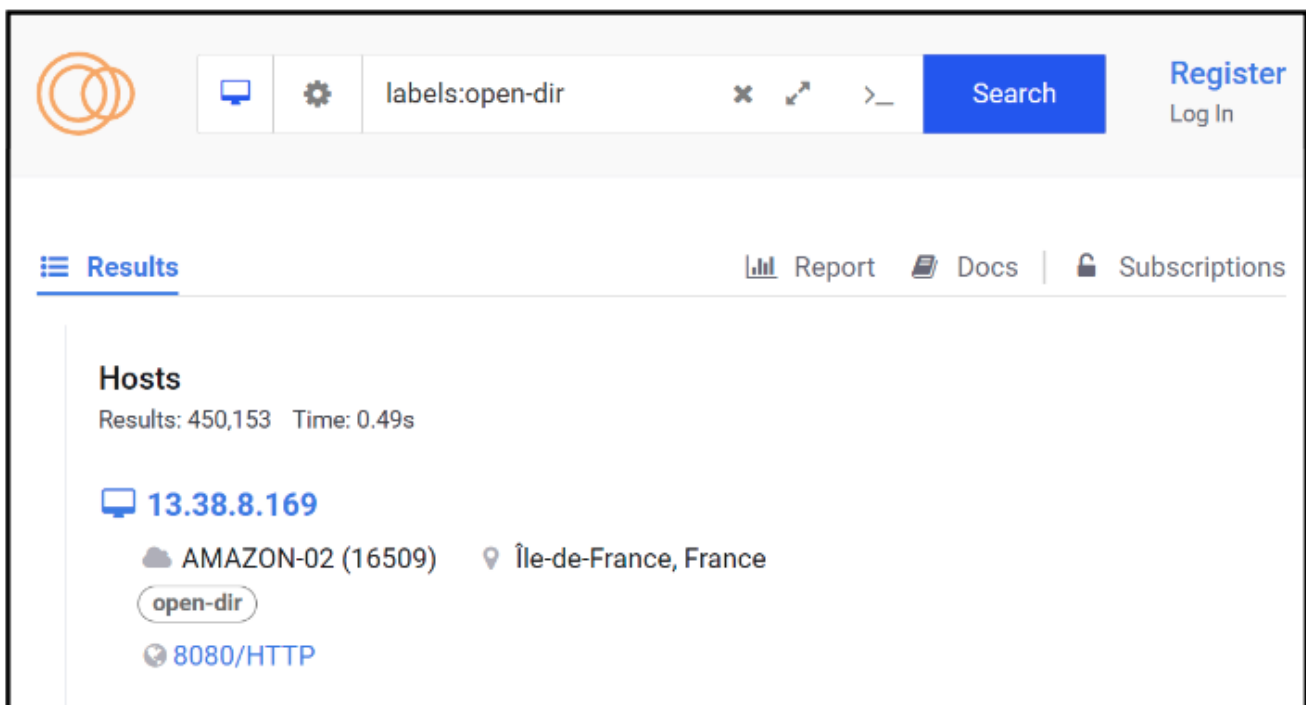
So How Can You Find an Open Directory?

Open directories can be discovered in the community edition of Censys by searching for the **open-dir** label.

Censys automatically scans the internet for open directories and applies the **open-dir** label, regardless of the software used. This means that Apache, Python, HFS and others will all be included and do not need to be searched individually.

This label will include all open directories, including both malicious and legitimate results. The label alone can return hundreds of thousands of results.

We see this below with a plain search for **labels:open-dir**, which returns 450,153 current open directories.



As shown, the search for **labels:open-dir** will return all results regardless of whether they are malicious or legitimate.

The remainder of this blog will demonstrate how to combine this query with additional parameters to identify only malicious results.

Section 1: Static File Names For Open Directory Hunting

The most simplistic method to identify malicious directories is to leverage file names from previous incidents.

Let's consider **81.71.147.[.]158**, which was shared by [@morimolymoly2](#) on Twitter/X. This IP contains an open directory with a large number of suspicious files.

If we search for the IP on Censys, we will end up at the host page where the following information is available on port 80.

http://81.71.147.158/

Status	200 OK
Body Hash	sha1:16fb49ac5ba1aaf72e276f05f43bb03db52feb2
HTML Title	Index of /
Response Body	EXPAND

```
# Index of /

![\[ICO\]](/icons/blank.gif) | [Name](?C=N;O=D) | [Last modified](?C=M;O=A) |
[Size](?C=S;O=A) | [Description](?C=D;O=A)
---|---|---|---|---

* * *

![\[ \]](/icons/binary.gif) | [02.exe](02.exe) | 2024-04-19 14:59 | 3.1M|
![\[ \]](/icons/unknown.gif) | [1.php](1.php) | 2023-12-25 17:47 | 16 |
![\[ \]](/icons/unknown.gif) | [1.rtf](1.rtf) | 2024-05-06 09:46 | 120 |
![\[TXT\]](/icons/text.gif) | [1.txt](1.txt) | 2024-04-07 11:55 | 56 |
![\[ \]](/icons/unknown.gif) | [ErrorBaseExec.jar](ErrorBaseExec.jar) |
2016-02-27 14:54 | 1.5K|
![\[ \]](/icons/unknown.gif) | [a.dll](a.dll) | 2016-07-31 21:46 | 464K|
![\[ \]](/icons/binary.gif) | [a.exe](a.exe) | 2024-05-07 09:47 | 28K|
![\[ \]](/icons/unknown.gif) | [a.hta](a.hta) | 2023-12-19 15:11 | 838 |
![\[IMG\]](/icons/image2.gif) | [a.jpg](a.jpg) | 2024-02-02 10:51 | 106K|
![\[ \]](/icons/text.gif) | [a.txt](a.txt) | 2023-12-20 15:35 | 1.2K|
![\[ \]](/icons/binary.gif) | [calc.exe](calc.exe) | 2019-12-07 17:09 | 27K|
![\[ \]](/icons/binary.gif) | [cmd.exe](cmd.exe) | 2024-04-26 11:03 | 3.1M|
![\[ \]](/icons/unknown.gif) |
![\[ \]](/icons/unknown.gif) | [getcatgetinfo.action](getcatgetinfo.action) | 2024-04-29 11:35 | 25 |
![\[ \]](/icons/unknown.gif) | [yaml-payload.jar](yaml-payload.jar) |
2024-03-22 10:26 | 2.2K|
![\[ \]](/icons/compressed.gif) | [yinyue.zip](yinyue.zip) | 2024-05-08 16:59
![\[ \]](/icons/compressed.gif) | [yinyue2.zip](yinyue2.zip) | 2024-05-08
M|
```

Short File Names

Payload Filename

The open directory on **81.71.147.158** contains multiple files that are unique enough to be used as pivot points. Which are values that are unique enough to be used in a query.

There are two primary patterns which stand out.

- **A.dll, a.exe, a.hta, a.jpg** – Suspicious files with short and single character file names.
- **Yaml-payload.jar** – Suspicious file containing “payload” in the file name. Likely related to a Java Deserialization exploit.

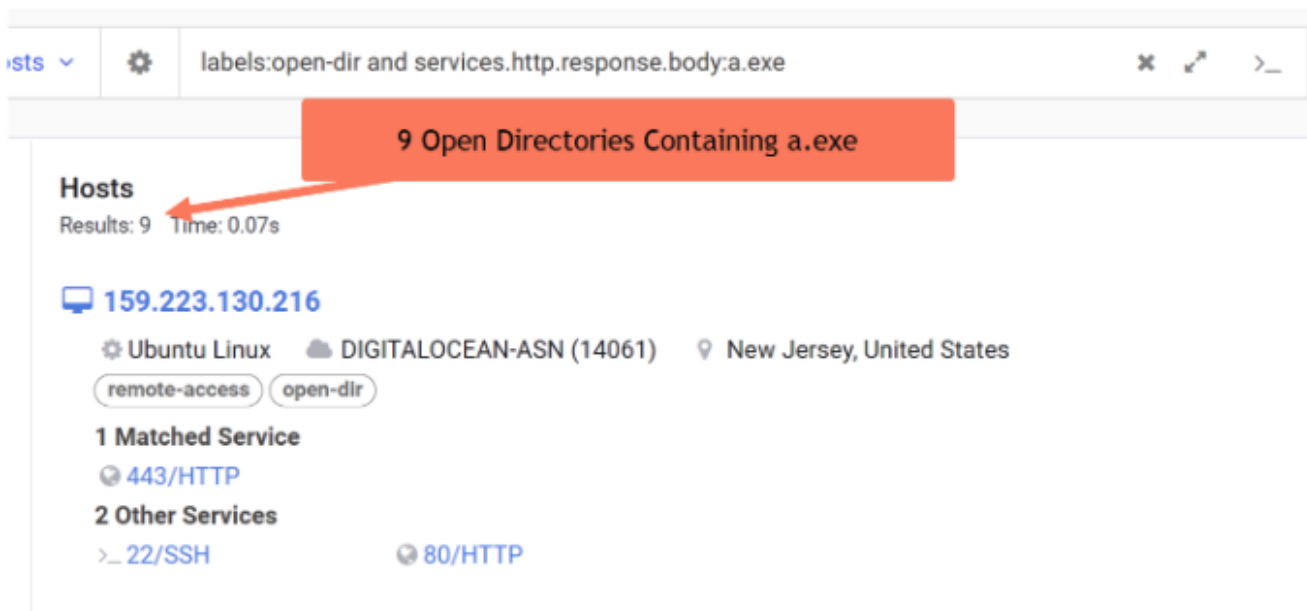
We can combine these file names with the **labels:open-dir** query to identify open directory infrastructure hosting files with the same names.

Pivoting on Static File Names

File names like **a.exe** can be combined with the **labels:open-dir** parameter to identify open directories hosting files with the same name (although not necessarily the same file contents).

Censys stores open directory content in the **services.http.response.body** field, so this is where we can place a file name in combination with **labels:open-dir**

Hence, we can search for open directories containing a.exe by searching for **labels:open-dir** and **services.http.response.body:a.exe**. This simple search returns 9 similar servers.



As shown above, the search reveals 9 open directories containing **a.exe**.

The first result from the search is **159.223.130[.]216**. Browsing to the associated host page shows an open directory hosting a.exe, as well as other files with similar single-character naming schemes. In addition to **a.exe**, we now have **b.exe**, **c.exe** and **curl.exe**.

Curl is a “legitimate” tool used to download files, so **curl.exe** is unlikely to be malware, but instead a supporting tool used to “install” curl during operations where the curl tool was not present.

To confirm the nature of these files, we can download them (using a sandbox or separate analysis machine) by browsing to the site directly and then performing manual analysis or submitting them to a sandbox. This is not always recommend for opsec reasons, but this is a topic for another discussion.

In cases where the infrastructure is not sensitive, the files can be scanned by inputting the URL directly into [VirusTotal](#). In this case, we can input **<IP>/b.exe** and see that it has 34

Details

http://159.223.130.216:443/

Status 200 OK

Body Hash sha1:f47ce2945f6f83114c2b08f7fe8f25138bf77c37

HTML Title Directory listing for /

Response Body EXPAND

```
# Directory listing for /
* * *
* [a.exe](a.exe)
* [b.exe](b.exe)
* [c.exe](c.exe)
* [curl.exe](curl.exe)
* * *
```

detections and contains a Sliver C2 Implant.

Note that this kind of scanning will often alert the actor that their servers are being investigated. You should take this into consideration when investigating infrastructure.

34 / 74

Community Score

34/74 security vendors and no sandboxes flagged this file as malicious

f1af3072dce3dde62fda28b7fcbacde155a0d06e3494b8cea199f53d2d20d0f0

b.exe

peexe 64bits

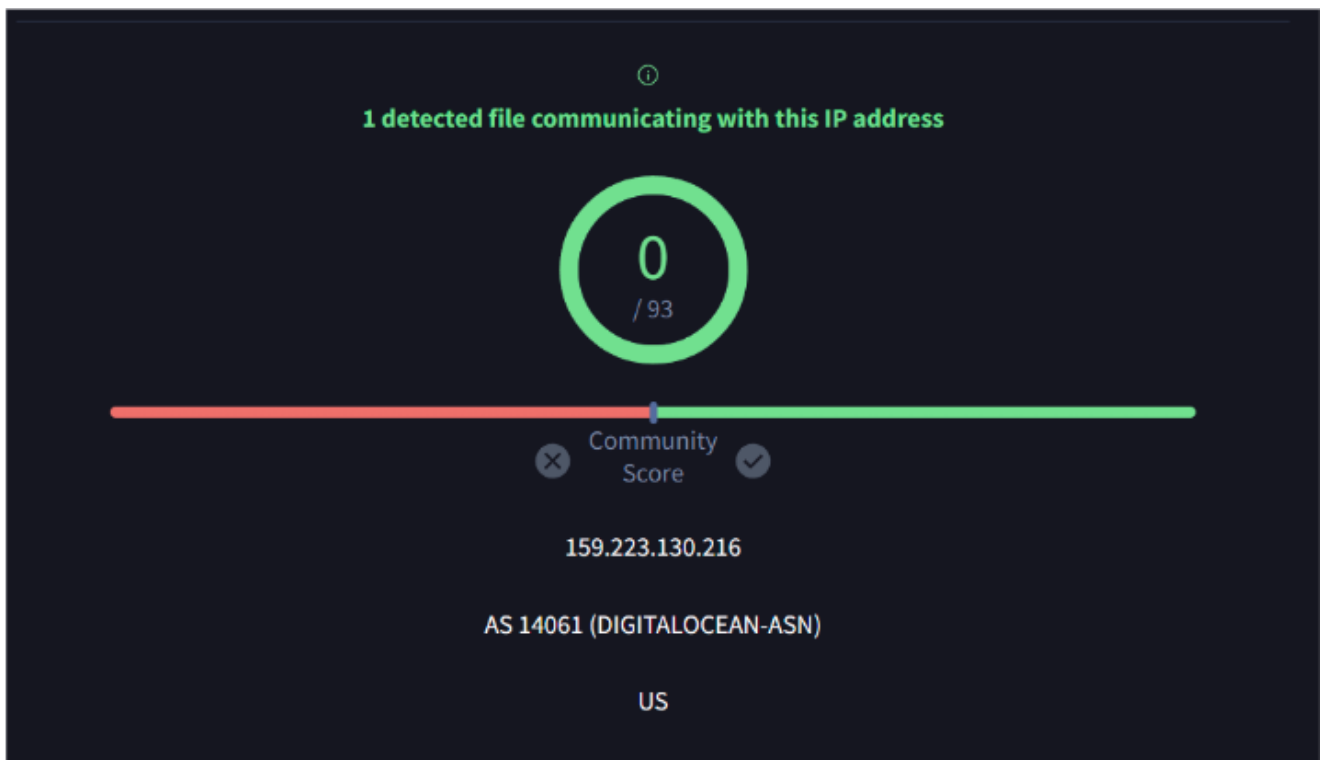
DETECTION DETAILS COMMUNITY

The **b.exe** file has 34 detections, but the open directory hosting it has 0.

This means that we've likely found some "new" infrastructure, using only a simple file name for our analysis.

We can continue investigating the search results for more instances of **a.exe**.

Another result from our prior search is **121.43.135[.]166**, which contains **a.exe** and



numerous other suspicious files.

```
# 121.43.135.166 - /
* * *

2021/10/6 1:33 3382 [1111.png](/1111.png)
2021/9/8 18:58 2097152 [a.exe](/a.exe)
2022/7/26 11:13 17537046 [apk.rar](/apk.rar)
2022/7/27 11:47 22925392 [app.apk](/app.apk)
2021/7/30 12:56 974848 [b.exe](/b.exe)
2021/9/15 19:01 4523520 [c.exe](/c.exe)
2022/4/1 15:04 2050048 [d - 副本 (2).exe](/d%20-%20%E5%89%AF%E6%9C%AC%20\
(2).exe)
2022/3/31 13:33 2049536 [d - 副本.exe](/d%20-%20%E5%89%AF%E6%9C%AC.exe)
2022/8/16 11:33 2051072 [d.exe](/d.exe)
2021/5/27 22:37 43548 [script.js](/script.js)
2022/12/14 10:30 316 [web.config](/web.config)

* * *
```

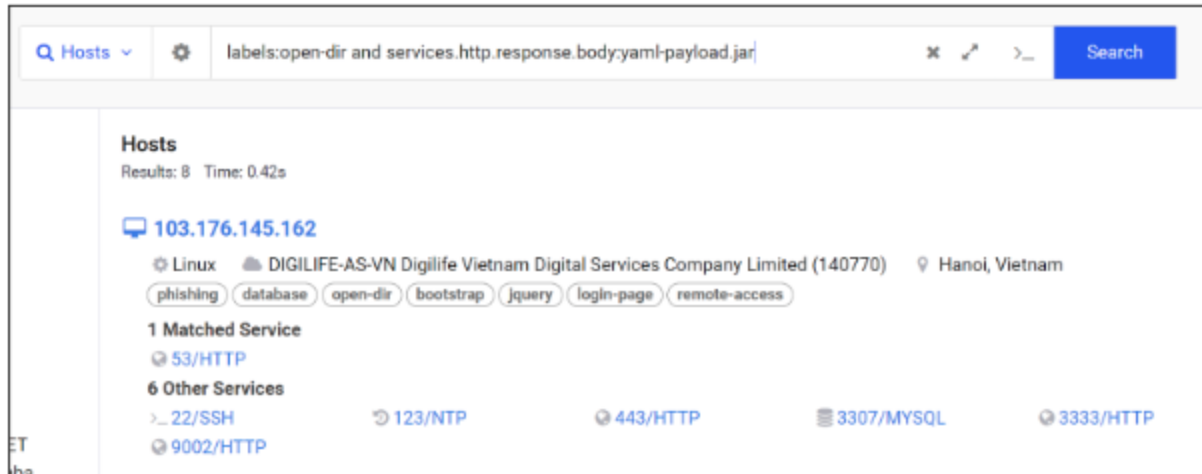
The same patterns of file naming can be seen across other servers returned by the search, which all contain **a.exe** in

combination with other suspicious files.

Pivoting on An Exploit File Name

The initial IP contained another suspicious file named **yaml-payload.jar**. Googling this file suggests that it is related to a Yaml Deserialization exploit.

Repeating the process from before, we can identify similar servers by searching for **labels:open-dir and services.http.response.body:"yaml-payload.jar"**



The simple search returns 8 open directories hosting files with the same name.

One of the results contains **yaml-payload.jar** and a reference to **artifact_x86.exe**, which is a common file name for Cobalt Strike.

This means that our simple pivot on **yaml-payload.jar** has likely led to the server of an actor leveraging Cobalt Strike.

The **artifact_x86.exe** name is unique enough to be utilised as another pivot point. An analyst could perform additional pivoting on the **artifact_x86.exe** name.

This would be as simple as repeating the previous searches with different filenames in the **services.http.response.body** field.

http://175.178.124.71:8888/

Status 200 OK

Body Hash sha1:ce0170cb333f4cdddbbf3c3fc7d9fffaceb77cd

HTML Title Directory listing for /

Response Body

```
# Directory listing for /
* * *
* [.bash_history](.bash_history)
* [.bash_logout](.bash_logout)
* [.bashrc](.bashrc)
* [.cache/](.cache/)
* [.pip/](.pip/)
* [.profile](.profile)
* [.pydistutils.cfg](.pydistutils.cfg)
* [.ssh/](.ssh/)
* [.sudo_as_admin_successful](.sudo_as_admin_successful)
* [ARL-plus-docker-2.7.1/](ARL-plus-docker-2.7.1/)
* [ARL-plus-docker-2.7.1.zip](ARL-plus-docker-2.7.1.zip)
* [artifact_x86.exe](artifact_x86.exe)
* [attack/](attack/)
* [yan1-payload.jar](yan1-payload.jar)
* [yan1-payload123.jar](yan1-payload123.jar)
* * *
```

Summary – Static File Names For Open Directory Hunting

Static file names serve as a simple and highly effective means to discover new malicious open directories.

By using public reporting (Social Media, Intel Repos, Internal Incidents), you can easily identify simple file names that can lead to new infrastructure.

To achieve all of this, simply search for **labels:open-dir** and then add your suspicious file name into the **services.http.response.body** field.

Section 2: Autonomous Systems and Hosting Providers

Open directory hunting can be heavily assisted by combining hosting providers with the **labels:open-dir** query.

This can be especially effective when an actor uses a unique or uncommon hosting provider.

Consider the IP **77.105.160.30** (initially shared by [@karol_paciorek](#)). This server has an open directory and is hosted on EVILEMPIRE with an Autonomous System Number of **216309**.

Since this is an

Hosts

Results: 1 Time: 0.05s

77.105.160.30

Ubuntu Linux EVILEMPIRE-AS (216309) Vienna, Austria

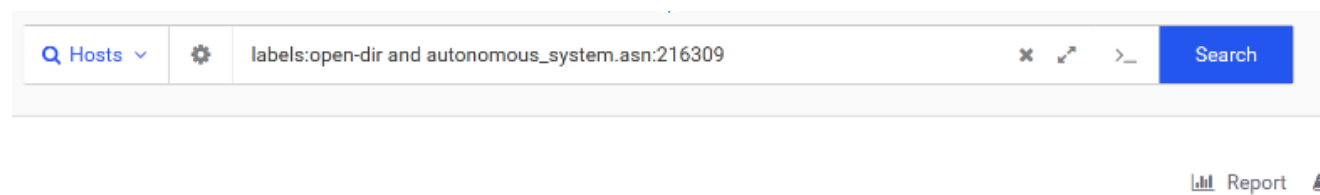
open-dir remote-access

>_ 22/SSH

80/HTTP

uncommon provider, we can discover more open directories by combining the associated ASN number with **labels:open-dir**.

The usage of EVILEMPIRE is unique enough that there are only 8 open directories hosted there.



Hosts

Results: 8 Time: 0.10s

77.105.132.5

Ubuntu Linux EVILEMPIRE-AS (216309) Uusimaa, Finland

open-dir remote-access

>_ 22/SSH

80/HTTP

77.105.135.22

Ubuntu Linux EVILEMPIRE-AS (216309) North Holland, Netherlands

remote-access open-dir

>_ 22/SSH

80/HTTP

One of those results is **77.105.132[.]27**, which matches our search criteria and has already been marked as a known C2.

77.105.132.27

EVILEMPIRE-AS (216309) Uusimaa, Finland

open-dir remote-access c2 network-administration react login-page

80/HTTP

1080/SOCKS

3389/RDP

8081/HTTP

50500/UNKNOWN

By browsing to the host page and looking at the directory contents, it appears to be hosting both Vidar and Lumma malware.

The screenshot shows the Censys Hosts interface for IP 77.105.132.27. The page displays the following information:

- Status:** 200 OK
- Body Hash:** sha1:3f854afbddd6674be4442cef5cf9a26de2a265f0
- HTML Title:** Index of /
- Response Body:** EXPAND

The expanded response body shows a directory listing for the path `/`. The listing includes the following entries:

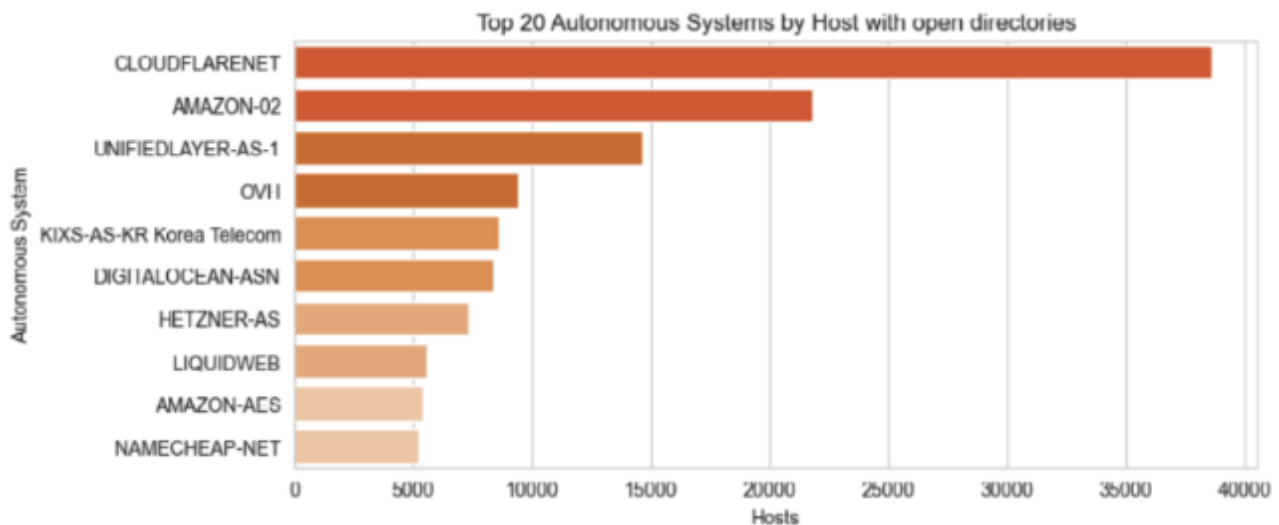
```
# Index of /  
  
![\[ICO\]](/icons/blank.gif) [Name](?C-N;O-D) [Last modified](?C-M;O-A)  
[Size](?C-S;O-A) [Description](?C-D;O-A)  
---|---|---|---|---  
  
* * *  
  
![\[ \]](/icons/binary.gif) [lumma0607.exe](lumma0607.exe) | 2024-07-06  
19:43 | 512K|  
![\[ \]](/icons/binary.gif) [vidar0607.exe](vidar0607.exe) | 2024-07-06  
19:39 | 447K|  
  
* * *  
  
Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.0.30 Server at 77.105.132.27 Port 80
```

Summary – Autonomous Systems and Hosting Providers

Combining open directory hunts with unusual hosting providers can quickly lead to new results.

This works best when the provider is uncommon or well known for hosting malicious actors. So be careful when applying this technique to a large provider such as Amazon or CloudFlare. Large providers such as these can be associated with tens of thousands of results, which can be difficult to parse and extremely prone to false positives without additional filtering.

Consider that CloudFlare was linked to 38,614 open directories in 2023 alone. With Amazon being linked to a considerable 21,805. More statistics are covered in detail in “[Dorking The Internet](#)”.



Section 3: File Name Patterns and Regular Expressions

In the first section, we used static file names to pivot to additional open directories. However there is a much better way to do this with the addition of regular expressions.

Consider the search for “*files named a.exe*” vs. the search for “ANY single character .exe.” The second option is more generic (in a good way) and allows for more effective searching.

We can abstract the **a.exe** file name using regular expressions to instead search for any single-character executable. For more effective searching, we can even expand this to any single-character filename with the extensions **exe**, **hta** or **rtf**.

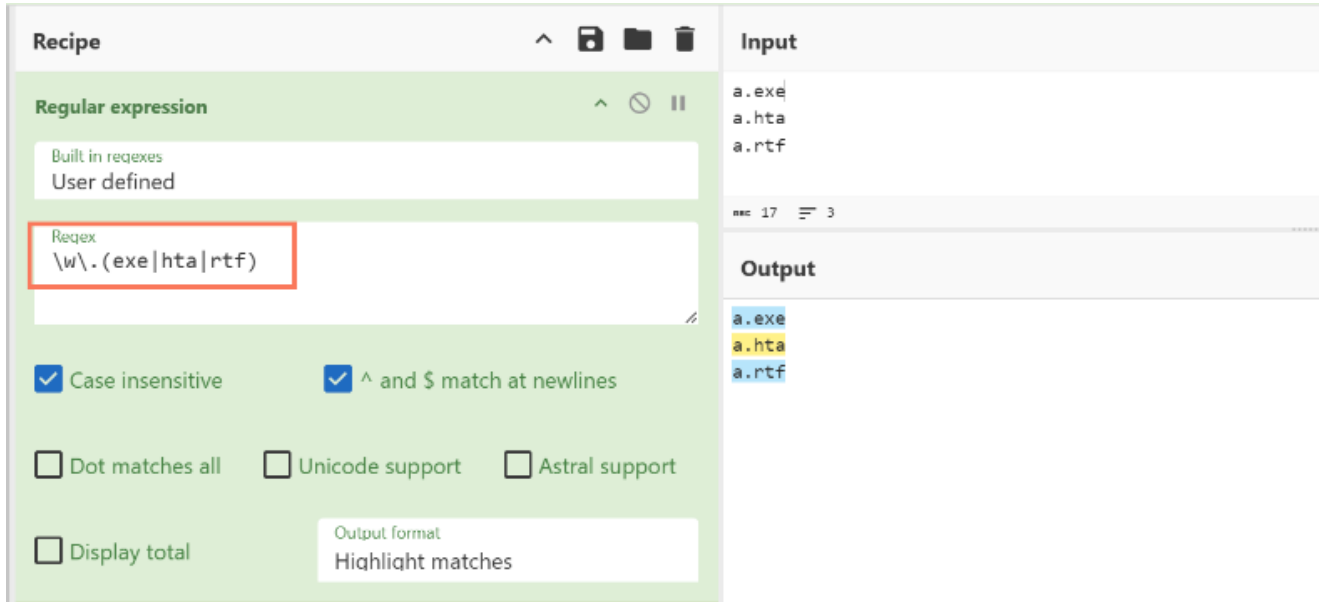
Consider the following file names from one of our previous searches.

```
![\[ \]](/icons/binary.gif) | [02.exe](02.exe) | 2024-04-19 14:59 | 3.1M|
![\[ \]](/icons/unknown.gif) | [1.php](1.php) | 2023-12-25 17:47 | 16 |
![\[ \]](/icons/unknown.gif) | [1.rtf](1.rtf) | 2024-05-06 09:46 | 120 |
![\[TXT\]](/icons/text.gif) | [1.txt](1.txt) | 2024-04-07 11:55 | 56 |
![\[ \]](/icons/unknown.gif) | [ErrorBaseExec.jar](ErrorBaseExec.jar) |
2016-02-27 14:54 | 1.5K|
![\[ \]](/icons/unknown.gif) | [a.dll](a.dll) | 2016-07-31 21:46 | 464K|
![\[ \]](/icons/binary.gif) | [a.exe](a.exe) | 2024-05-07 09:47 | 28K|
![\[ \]](/icons/unknown.gif) | [a.hta](a.hta) | 2023-12-19 15:11 | 838 |
![\[IMG\]](/icons/image2.gif) | [a.jpg](a.jpg) | 2024-02-02 10:51 | 106K|
![\[TXT\]](/icons/text.gif) | [a.txt](a.txt) | 2023-12-20 15:35 | 1.2K|
![\[ \]](/icons/binary.gif) | [calc.exe](calc.exe) | 2019-12-07 17:09 | 27K|
![\[ \]](/icons/binary.gif) | [cmd.exe](cmd.exe) | 2024-04-26 11:03 | 3.1M|
![\[ \]](/icons/unknown.gif) |
```

The open directory contains 9 files with only a single character before the extension (**a.dll**, **1.rtf**, **a.hta** etc).

Rather than searching for these names individually, let's create a regular expression that searches for single-character file names with any of the **exe**, **rtf** or **hta** extensions.

We can create a simple prototype using CyberChef, and then add it to the **services.http.response.body** field.



Note that we'll now need to utilise the raw HTML content and not the HTML rendering from previous screenshots, so we'll be adding `.\` to both sides of our regular expressions. This accounts for the HTML syntax (shown below) and specifies that we only want files named `a.exe`, not those containing `a.exe`.*

Below we can see the quotes “ before and after file names, which we should account for in our regular expression.

Taking the raw HTML into account, we can search for single-character filenames with the following query.

labels:open-dir and services.http.response.body:/.*\w\.(exe|hta|rtf)\".*/

For those unfamiliar with regular expressions, here is a visualization courtesy of regexper.com

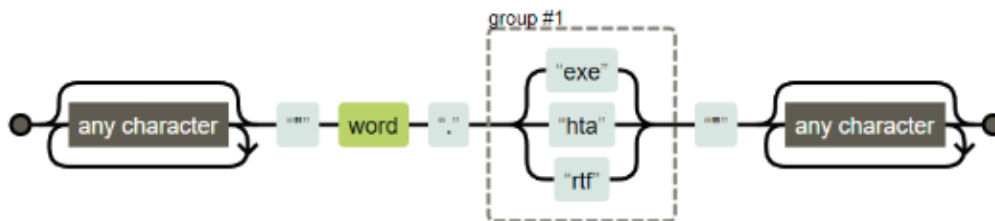
Executing this new search returns 57 results for open directories with single character file names.

One of these results is **20.98.129[.]89**, which contains a single character **e.hta** file (matching our regular expression) as well as a very suspicious **payload.exe**.

Another search result is **1.92.96[.]35**, where our regular expression has matched on single-

services.http.response.body

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">\n<html>\n<head>\n<meta http-equiv="Content-Type" content="text/html; charset=utf-8">\n<title>Directory listing for /</title>\n</head>\n<body>\n<h1>Directory listing for /</h1>\n<hr>\n<ul>\n<li><a href="a.exe">a.exe</a></li>\n<li><a href="b.exe">b.exe</a></li>\n<li><a href="c.exe">c.exe</a></li>\n<li><a href="curl.exe">curl.exe</a></li>\n</ul>\n<hr>\n</body>\n</html>\n
```



character file names like **f.exe**, **m.exe**, **m.hta**, **p.hta**.

In addition to the matching names, we now have an open directory referencing **cs4.9**, which is likely a reference to Cobalt Strike Version 4.9. The **cs4.9** string would be great to use for additional pivots.

Another search result is **38.206.173[.158]**, which appears to be hosting ransomware due to the presence of **unlocker.exe** and **READ_TO_DECRYPT.html**.

There are numerous opportunities here for files that could be used to pivot to additional servers.



labels:open-dir and services.h x ↗ >_

Search



ER

labels:open-dir and services.http.response.body:/*"\w\.(exe|hta|rtf)\".*/

Hosts

Results: 57 Time: 3.12s

 **121.36.98.41** (ecs-121-36-98-41.compute.hwclouds-dns.com)

 HWCSNET Huawei Cloud Service data center (55990)  Beijing, China

open-dir

1 Matched Service

 [8443/HTTP](#)

2 Other Services

 [80/HTTP](#)

 [443/HTTP](#)

HTTP 80/TCP


06/19/2024 06:55 UTC

OPEN DIR

Software

VIEW ALL DATA

GO

 Python Software Foundation SimpleHTTP 0.6 

Details

http://20.98.129.89/

Status

200 OK

Body Hash

sha1:4cd837728b2fc29a02563f6bb596b4382f5ab1d4

HTML Title

Directory listing for /

Response Body

EXPAND

```
# Directory listing for /  
  
* * *  
  
  * [e.hta](e.hta)  
  * [payload.exe](payload.exe)  
  
* * *
```



1.92.96.35



Search

ER

```
* [a.txt](a.txt)
* [b.exe](b.exe)
* [calc.msi](calc.msi)
* [cs/](cs/)
* [cs4.9/](cs4.9/)
* [dns.exe](dns.exe)
* [evil.png](evil.png)
* [evil1.hta](evil1.hta)
* [f.exe](f.exe)
* [fastjson_rec_exploit-master/](fastjson_rec_exploit-master/)
* [fr.exe](fr.exe)
* [frpc.ini](frpc.ini)
* [jjjj.exe](jjjj.exe)
* [m.exe](m.exe)
* [m.hta](m.hta)
* [main.exe](main.exe)
* [main2.exe](main2.exe)
* [mysps.exe](mysps.exe)
* [mysps2.exe](mysps2.exe)
* [ne.msi](ne.msi)
* [ne2.png](ne2.png)
* [net20ps2.exe](net20ps2.exe)
* [net3.exe](net3.exe)
* [netspy.exe](netspy.exe)
* [p.hta](p.hta)
* [pp.hta](pp.hta)
* [range.py](range.py)
* [SqlmapXPlus-main/](SqlmapXPlus-main/)
```

```
* [pocw8.xml](pocw8.xml)
* [pocw9.xml](pocw9.xml)
* [prepare.zip](prepare.zip)
* [r/](r/)
* [r.exe](r.exe)
* [READ_TO_DECRYPT.html](READ_TO_DECRYPT.html)
* [rivalz.exe](rivalz.exe)
* [simple.tar.gz](simple.tar.gz)
* [titan.exe](titan.exe)
* [titan_win_v0.0.7.exe](titan_win_v0.0.7.exe)
* [titan_windows.tar.gz](titan_windows.tar.gz)
* [unlocker.exe](unlocker.exe)
* [user.zip](user.zip)
* [ZE15.tar.gz](ZE15.tar.gz)
```

* * *

Summary – File Name Patterns and Regular Expressions

File Name Patterns in the form of regular expressions can be far more effective than static name searching. If you notice multiple file names with different but “similar” values, try using a regular expression to group them together.

Advanced queries utilising regular expressions will often lead to additional malicious servers.

Section 4: Combining File Extensions

File extensions can be another simple and effective way to identify suspicious open directories.

For example, the combination of an **.exe** and **.hta** file in the same open directory is rare and unlikely to occur in a legitimate directory. So we can use this idea to identify malicious servers. The same concept can be applied to a **.hta** and **.ps1** sharing the same directory.

Consider one of our previous results, which contains a mixture of **.hta**, **.png**, **.exe**, **.msi** and **.txt**. We can use this combination (or a subset) to identify additional infrastructure.



The screenshot shows a search interface with a toolbar at the top containing a logo, a monitor icon, a gear icon, the IP address 1.92.96.35, a search button, and an 'ER' button. Below the toolbar is a list of file name patterns, each preceded by an asterisk:

```
[a.txt](a.txt)
[b.exe](b.exe)
[calc.msi](calc.msi)
[cs/](cs/)
[cs4.9/](cs4.9/)
[dns.exe](dns.exe)
[evil.png](evil.png)
[evil1.hta](evil1.hta)
[f.exe](f.exe)
[fastjson_rec_exploit-master/](fastjson_rec_exploit-master/)
[fr.exe](fr.exe)
[frpc.ini](frpc.ini)
[jjjj.exe](jjjj.exe)
[m.exe](m.exe)
[m.hta](m.hta)
[main.exe](main.exe)
```

We can build an query that searches for all open directories containing both a **.hta** and **.exe** extension.

labels:open-dir and same_service(services.http.response.body:*.hta* and services.http.response.body:*.exe*) and not services.http.response.body:*htaccess*

A few quick notes on that query:

- Same_service – This tells the search only to include results where the files were observed on the same port. We don't want a server with .hta on port 443 and .exe separately on port 80.
- Not .htaccess – This is a legitimate file which matches on our wildcard search for .hta, we want to exclude this from our results without resorting to regular expressions.

Running that query returns 9 results, one of these is an open directory on **20.163.176[.]155**.

This directory matches our search for both a .hta and .exe file on the same service.

Details

<http://20.163.176.155:443/>

Status
200 OK

Body Hash
sha1:8f0ffa9ef9712ed5a603072bb2f90aefcd7fc78f

HTML Title
Directory listing for /

Response Body

EXPAND

```
# Directory listing for /

* * *

* [ps-updater.exe](ps-updater.exe)
* [update.hta](update.hta)
* [update.ps1](update.ps1)
* [update.ps1~](update.ps1~)
* [update2.hta](update2.hta)

* * *
```

This open directory on **20.163.176[.]155** contains multiple “update” files that are marked as downloaders by VirusTotal. So we've encountered another server containing malicious files.

The open directory also contains a new pattern of **ps1** and **exe** files, so we can adjust our query to search for this and identify further suspicious results.

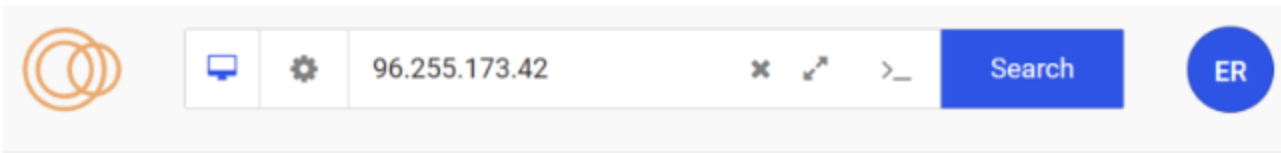
This query will search for open directories containing both a powershell script and executable file.

labels:open-dir and same_service(services.http.response.body:*.ps1* and services.http.response.body:*.exe*)

The screenshot shows a search interface with a search bar containing the query: `labels:open-dir and same_ser and services.http.response.body:*.ps1* and services.http.response.body:*.exe*`. The search results are displayed under the heading "Hosts" and show "Results: 84 Time: 0.21s". The first result is for the IP address **183.150.71.22**, which is associated with Microsoft and CHINANET-BACKBONE. The location is listed as "China" and "Zhejiang". A tag "open-dir" is present. Below the host information, there is a section for "1 Matched Service" which is **888/HTTP**, and a section for "2 Other Services" which are **5244/HTTP** and **9000/HTTP**.

The search returns 84 results for open directories containing powershell scripts and executables.

One of these results is **96.255.173[.142]**, which contains both **.exe** and **ps1**, as well as a collection of other suspicious files likely related to the PowerSploit Toolkit.



```
# Directory listing for /  
  
* * *  
  
* [51247.py](51247.py)  
* [agent.exe](agent.exe)  
* [AzureAdmins/](AzureAdmins/)  
* [BloodHound-linux-x64/](BloodHound-linux-x64/)  
* [BloodHound.py/](BloodHound.py/)  
* [bloodyAD/](bloodyAD/)  
* [chisel/](chisel/)  
* [cve-2019-9193.py](cve-2019-9193.py)  
* [CVE-2020-1472/](CVE-2020-1472/)  
* [gMSADumper/](gMSADumper/)  
* [krbrelayx/](krbrelayx/)  
* [lingolo/](lingolo/)  
* [linpeas.sh](linpeas.sh)  
* [MS17-010/](MS17-010/)  
* [powercat.ps1](powercat.ps1)  
* [powerview.ps1](powerview.ps1)  
* [PowerView.ps1](PowerView.ps1)  
* [powerview.py/](powerview.py/)  
* [Procmon.exe](Procmon.exe)  
* [Procmon64.exe](Procmon64.exe)  
* [RemotePotato0.exe](RemotePotato0.exe)  
* [SharpHound.ps1](SharpHound.ps1)  
* [test.txt](test.txt)  
* [Windows-Exploit-Suggester/](Windows-Exploit-Suggester/)  
* [winPEASany.exe](winPEASany.exe)  
  
* * *
```

Bonus Pivot On Filename Patterns

The previously shown directory contains multiple files with “power” in the file name, followed by a **ps1** or **py** extension.

Using regular expressions, we can turn this into a generic query for any open directory containing **.py** or **ps1** and a file name beginning with **power**.

We can first prototype a regular expression using CyberChef.

With the regular expression working, we can search for **.ps1** or **.py** files whose file name contains power.

The search returns 7 results, one of which is **95.111.214[.]111**.

The screenshot shows a 'Recipe' configuration window for a regular expression search. The 'Regular expression' field contains the pattern `power\w+\.(ps1|py)`. The 'Case insensitive' checkbox is checked. The 'Output format' is set to 'Highlight matches'. The 'Input' field contains the following text: `powercat.ps1`, `powerview.ps1`, and `powerview.py`. The 'Output' field shows the same text with the matches highlighted in blue and yellow.

The screenshot shows a search interface with a search bar containing the query `labels:open-dir and services.h`. Below the search bar, the search results are displayed as `labels:open-dir and services.http.response.body:/*power[a-z]+\.(ps1|py).*/`. A blue 'Search' button is visible on the right side of the search bar.

The screenshot shows a search result for a host. The host is identified as `96.255.173.42`, located in Maryland, United States. The search results show 7 results in 0.34s. The host is associated with the service `443/HTTP` and is labeled as `open-dir`. The search results also show `1 Matched Service`.

We can see that this directory contains even more files related to offensive Powershell toolkits.

Another result is `116.114.20[.]180`, which contains `powercat.ps1`, amongst other suspicious file names.

The Powercat file is likely a reference to the Powershell implementation of Netcat.

Details

http://95.111.214.111:8000/

Status

200 OK

Body Hash

sha1:a9b0a89ee78618f63d78319cc3ea3150a8565592

HTML Title

Directory listing for /

Response Body

EXPAND

```
# Directory listing for /
```

```
* * *
```

```
* [Invoke-ConPtyShell.ps1](Invoke-ConPtyShell.ps1)
* [Invoke-JSRatRegsvr.ps1](Invoke-JSRatRegsvr.ps1)
* [Invoke-JSRatRundll.ps1](Invoke-JSRatRundll.ps1)
* [Invoke-PoshRatHttp.ps1](Invoke-PoshRatHttp.ps1)
* [Invoke-PoshRatHttps.ps1](Invoke-PoshRatHttps.ps1)
* [Invoke-PowerShellIcmp.ps1](Invoke-PowerShellIcmp.ps1)
* [Invoke-PowerShellTcpOneLine.ps1](Invoke-PowerShellTcpOneLine.ps1)
* [Invoke-PowerShellTcpOneLineBind.ps1](Invoke-PowerShellTcpOneLineBind.ps1)
* [Invoke-PowerShellUdp.ps1](Invoke-PowerShellUdp.ps1)
* [Invoke-PowerShellUdpOneLine.ps1](Invoke-PowerShellUdpOneLine.ps1)
* [Invoke-PowerShellWmi.ps1](Invoke-PowerShellWmi.ps1)
* [Invoke-PsGcat.ps1](Invoke-PsGcat.ps1)
* [Invoke-PsGcatAgent.ps1](Invoke-PsGcatAgent.ps1)
* [powershell.ps1](powershell.ps1)
* [Remove-PoshRat.ps1](Remove-PoshRat.ps1)
```

```
* * *
```



```
# Index of /
* * *

[../](../)
[RingQ.exe](RingQ.exe)          14-Jun-2024 06:04   785K
[a.exe](a.exe)                   07-Jun-2024 04:06   465K
[abc](abc)                       14-Jun-2024 05:44   3555
[ant.aspx](ant.aspx)             26-May-2024 06:17    170
[app.exe](app.exe)              01-Jun-2024 12:14   465K
[bc.ps1](bc.ps1)                07-Jun-2024 03:52   348K
[cc.out](cc.out)                26-May-2024 07:30    1M
[ccapiserver.exe](ccapiserver.exe) 04-Jun-2024 09:30
163K
[hb.exe](hb.exe)                 07-Jun-2024 06:24    8M
[k5sprt64.dll](k5sprt64.dll)      04-Jun-2024 09:30    26
K
[main.txt](main.txt)            14-Jun-2024 06:04   286K
[pc.ps1](pc.ps1)                07-Jun-2024 08:42    15K
[powercat.ps1](powercat.ps1)     05-Jun-2024 15:17    37
K
[update](update)                26-May-2024 07:24    1M

* * *
```

Summary: Combining File Extensions

File extensions can be as useful and simple as file names when searching for open directories.

If an investigation shows an open directory with an unusual combination of file types, try utilising that in your query. You'd be surprised how simple this is, yet how often it can lead to new malicious results.

Conclusion

We've now shown 4 useful techniques for identifying and hunting malicious open directories. These techniques are extremely effective for finding and hunting open directory infrastructure used by Threat Actors.

Although this is not an exhaustive list, these techniques, both on their own and in combination are amazing methods to have in your investigation toolkit.

With the exception of regular expressions, all of these techniques are available to try out in the [Community edition](#) of Censys.

And for those who love statistics and deep dives, check out the [Dorking The Internet](#) report by Censys. This enormous 31 page report takes a look at all the nitty gritty of open directory exposure, including both how and where they are being found in both malicious and legitimate scenarios.

File Type	File Count	Host Count	Total Size
Executable files containing executable code (.exe, .dll)	3,654,925	18,441	60,476.24 GB
Cryptography crypto-related files (.sig, .pem, crt)	2,060,704	9,297	212.93GB
Database files containing database information (.sql, .mdb)	523,158	11,913	3,663.45 GB
Configuration files with configuration data (.conf, .cfg)	328,201	13,966	3.42 GB
Shell Files shell-related files (.bashrc, .profile)	252,109	15,536	195.41 GB
Fonts files containing font data (.ttf, .woff2)	226,461	10,954	33.01 GB
Books files used for e-books (.mobi, .epub)	140,812	372	548.69 GB
Presentation files used for presentations (.ppt, .pptx)	23,095	1,629	149.17 GB
Financial files used in financial applications (.mny, .qfx)	33	4	0.14 MB

Bonus Related Queries

For Community Users

ANY open directory ([Link](#))

labels:open-dir

Open directories containing a.exe files ([Link](#))

labels:open-dir and services.http.response.body:a.exe

Open directories containing "Payload.exe"

labels:open-dir and services.http.response.body:payload.exe

Open directories referencing CS4.9 ([Link](#))

labels:open-dir and services.http.response.body:cs4.9

For Users With Regular Expression Access

Open directories containing "Power" files with ps1 or py extension

labels:open-dir and services.http.response.body:/.*power[a-z]+\.(ps1|py).*/

```
* [lingolo/](lingolo/)
* [linpeas.sh](linpeas.sh)
* [MS17-010/](MS17-010/)
* [powercat.ps1](powercat.ps1)
* [powerview.ps1](powerview.ps1)
* [PowerView.ps1](PowerView.ps1)
* [powerview.py/](powerview.py/)
* [Procmon.exe](Procmon.exe)
* [Procmon64.exe](Procmon64.exe)
* [Powercat.ps1](Powercat.ps1)
```

Open directories containing single character RTF, HTA or PS1 files

labels:open-dir and services.http.response.body:/.*\Ww\.(hta|rtf|ps1)\W.*/

```
# Directory listing for /
* * *
* [e.hta](e.hta)
* [payload.exe](payload.exe)
* * *
```

Open directories referencing any version of Cobalt Strike in format csX.X ([Link](#))

labels:open-dir and services.http.response.body:/.*\W(cs|cobalt)[34]\.ld(\.exe)?.*/

Open directories with short numeric names for PNG files

labels:open-dir and services.http.response.body:/.*\"[0-9]{1,5}\.png\W.*/

[Visit Embee Research](#)

[Go to Censys Search](#)

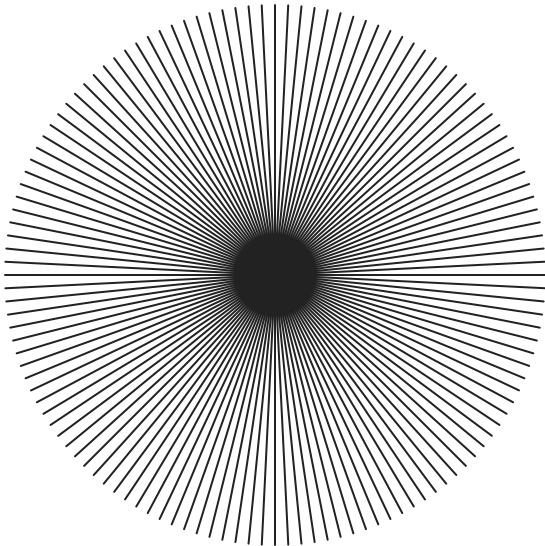
```
[.viminfo](.viminfo)
* [.Xauthority](.Xauthority)
* [1.apk](1.apk)
* [1.jpg](1.jpg)
* [bindtcp.elf](bindtcp.elf)
* [config.elf](config.elf)
* [config.jsp](config.jsp)
* [CS4.7/](CS4.7/)
* [frp/](frp/)
* [snap/](snap/)
* [xmrig.exe](xmrig.exe)

* * *
```

```
# Index of /

* [ 1.png](1.png)
* [ 2.jpg](2.jpg)
* [ a.htm](a.htm)
* [ index.htm](index.htm)
* [ web1.zip](web1.zip)
* [ web1/](web1/)
```

About the Author



Matthew

Embee Research

Matthew (aka @embee_research) is a security researcher based out of Melbourne, Australia. Matthew has a passion for all things malware, burritos and creating educational cyber content.