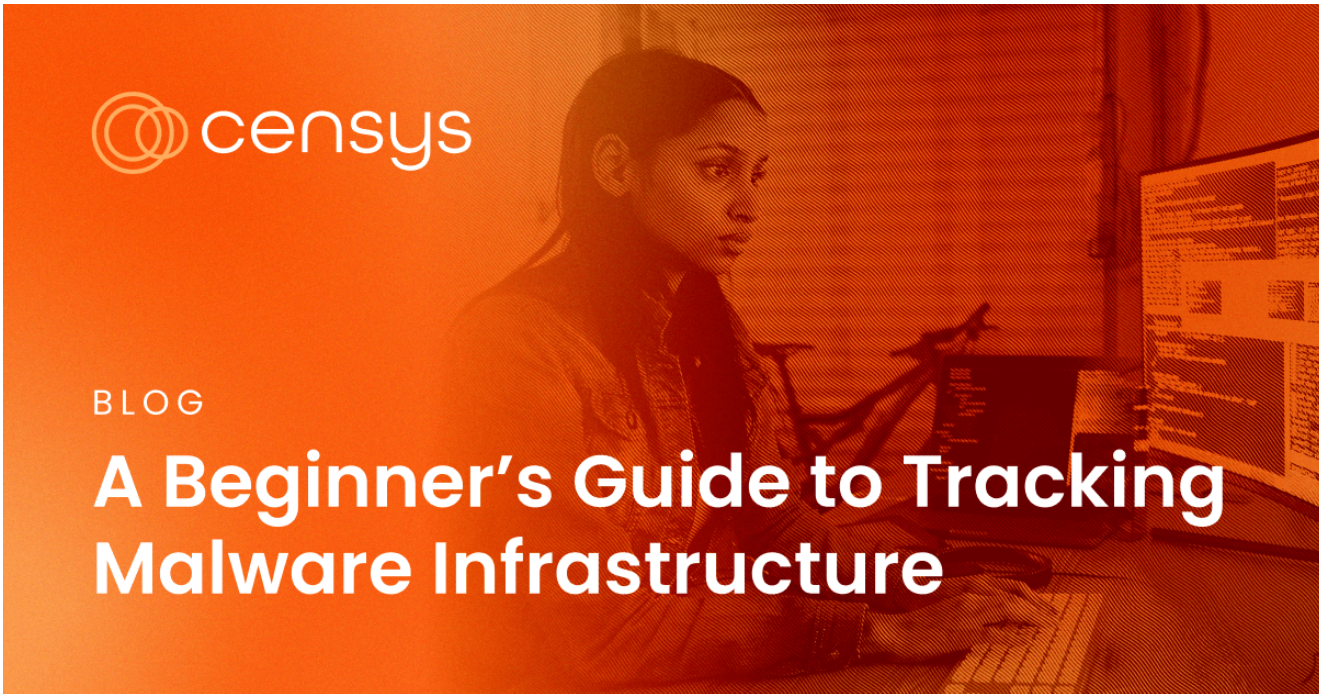
A Beginner’s Guide to Tracking Malware Infrastructure

censys.com/a-beginners-guide-to-tracking-malware-infrastructure/

February 9, 2024



Building queries for malware infrastructure can be a valuable step in the security lifecycle. Sadly, there are few resources for how to get started and which indicators can be used to build queries from. Today we aim to fill this gap by demonstrating approachable and high value methods that can be used to hunt for malware infrastructure.

What Is Query Building For Malware Infrastructure?

Query building is the process of observing suspicious or known malicious infrastructure and creating queries to identify the configuration pattern that the creator of the infrastructure has used. Since threat actors often re-use the same or similar configuration across multiple deployed servers, there is often a pattern that can be used to identify multiple servers from a single initial indicator.

A well built query allows an analyst to identify additional servers related to the actor’s infrastructure. The analyst can then proactively block, investigate or perform any additional actions needed to limit compromise and gather intelligence.

Why Build Queries On Malware Infrastructure?

Building queries on Malware Infrastructure can be a highly efficient means of obtaining IOC’s for blocking and hunting.

Traditional means of listing malware infrastructure involves obtaining a large set of unique malware samples and extracting individual IOC's from each file.

This can be a highly tedious and technical process requiring a dedicated reverse engineer to deconstruct a sample, develop and test a Yara hunting rule, acquire new samples, and then develop and apply a configuration extractor to obtain individual IOCs.

This reverse engineering capability involves a significant amount of technical know-how which most teams outsource to threat intelligence feeds. Outsourcing to threat intelligence feeds can be effective and there are good paid feeds available, but they are often expensive and can vary significantly in quality and timeliness.

Benefits of Building Infrastructure Queries

By developing your own infrastructure queries for the purposes of hunting, you can establish a far greater list of malware IOCs with a significantly smaller set of malware samples, technical expertise, and overall cost. You can also leverage queries to expand on alerting from your own environment, allowing you to establish a list of IOCs related to known malware impacting your organisation.

Using the techniques shown in this post, you can potentially identify dozens of current malware IOCs and infrastructure with only a single available sample or alert.

What Are The Indicators That We Can Use?

A single malicious IP address contains a great deal of information that can be used to identify additional servers. This is due to unique patterns related to the software and configuration deployed by an actor.

Since threat actors often re-use the same software and configurations across multiple instances of malicious infrastructure, a single pattern can be used to identify other servers.

Some of the most common indicators that threat actors will re-use are:

- **Certificate Information** – Fields inside of TLS and SSL certificates. Hardcoded values are often re-used.
- **Server Headers** – Actors deploying custom software may forget to change default headers that contain indicators.
- **Data in HTTP Responses** – Custom software containing unique values in HTTP responses
- **Location, ASN and Hosting Providers** – Actors re-using hosting providers for infrastructure. Similar servers may be hosted at the same ASN.
- **JA3 Hashes** – Actors deploying uncommon software configurations can be fingerprinted by JA3 signatures.

- **Port Configurations** – Actors will often leave the same ports open across infrastructure.
- **Regular Expressions** – Actors may deploy unique values with highly similar structure that can be captured with Regular Expressions.

Now that we've covered the key concepts, let's dive in with some examples.

Hunting Infrastructure with TLS Certificates

Threat actors and malware developers utilise TLS certificates to encrypt communications and establish connections between a target host and malicious infrastructure.

For many reasons, actors rarely deploy unique certificates for each deployed sample. This results in values within a single TLS certificate being present on numerous other servers, which introduces simple patterns that can be signatured and queried.

Example 1: Hunting AsyncRAT with TLS Certificates

The malware family of AsyncRAT contains a hardcoded TLS certificate left by the developer. This certificate contains the hardcoded subject common name value of **AsyncRAT Server**

Take for example an IP Address of **91.109.176[.]4.** Querying this IP in Censys Search confirms a subject common name (CN) value of **AsyncRAT Server** on port **8808**.

By expanding the host information and locating the exact field where the **AsyncRAT Server** value is stored **services.tls.certificates.leaf_data.subject_dn**, we can build a query to locate additional servers.

In this case, either the **subject_dn** or **issuer_dn** field can be used as they both contain the same hardcoded value.

By searching for **AsyncRAT Server** in either of these fields, we can locate an additional 110 servers with the same certificate value.

UNKNOWN 8808/TCP

01/28/2024 08:24 UTC

C2

VIEW ALL DATA

Software

AsyncRAT

Details

TLS

Handshake

Version Selected	TLSv1_0
Cipher Selected	TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA

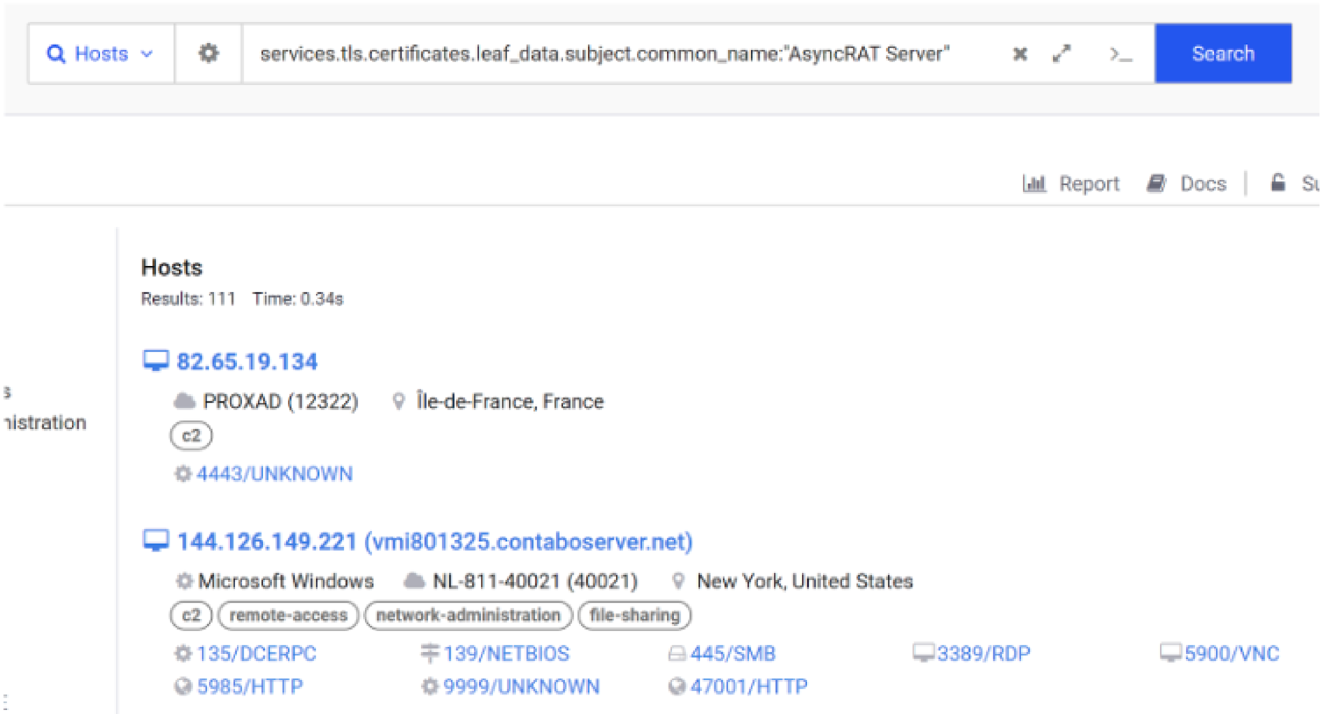
Certificate

Fingerprint	f364cbce06d0d5452f0a81bb55d8d7b02c3bf3ae4570d8779b64f80b4017791f
Subject	CN=AsyncRAT Server
Issuer	CN=AsyncRAT Server

Fingerprint

JARM	22b22b00022b22b22b22b22b22bd3b67dd3674d9af9dd91c1955a35d0e9
JA3S	bcf3a836c82d12ee988005fb0c011445

services.tls.certificates.leaf_fp_sha_256	f364cbce06d0d5452f0a81bb55d8d7b02c3bf3ae4570d8779b64f80b4017791f
services.tls.certificates.leaf_data.subject_dn	CN=AsyncRAT Server
services.tls.certificates.leaf_data.issuer_dn	CN=AsyncRAT Server

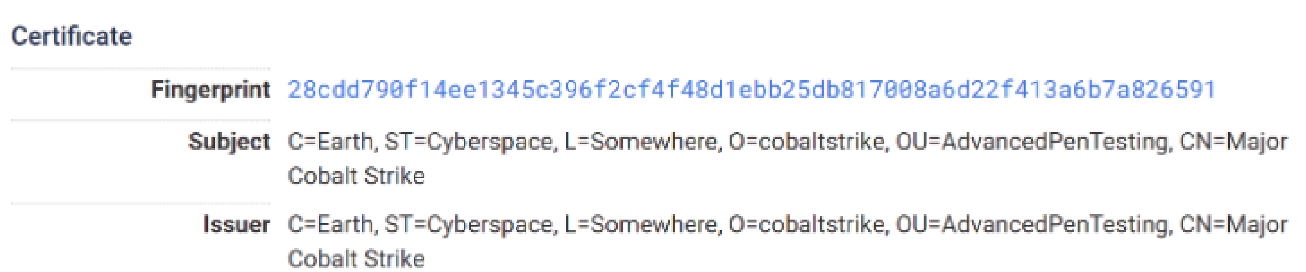


Example 2: Hunting Cobalt Strike with TLS Certificates

The infamous Cobalt Strike toolkit can also be tracked using TLS Certificate values.

This is primarily due to a default subject common name of **Major Cobalt Strike**

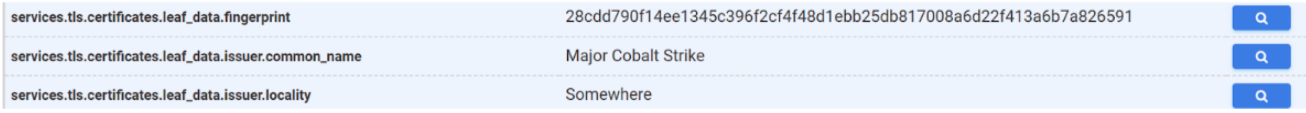
Take for example the IP address **23.98.137[.]196** with the following certificate on port **50050**.

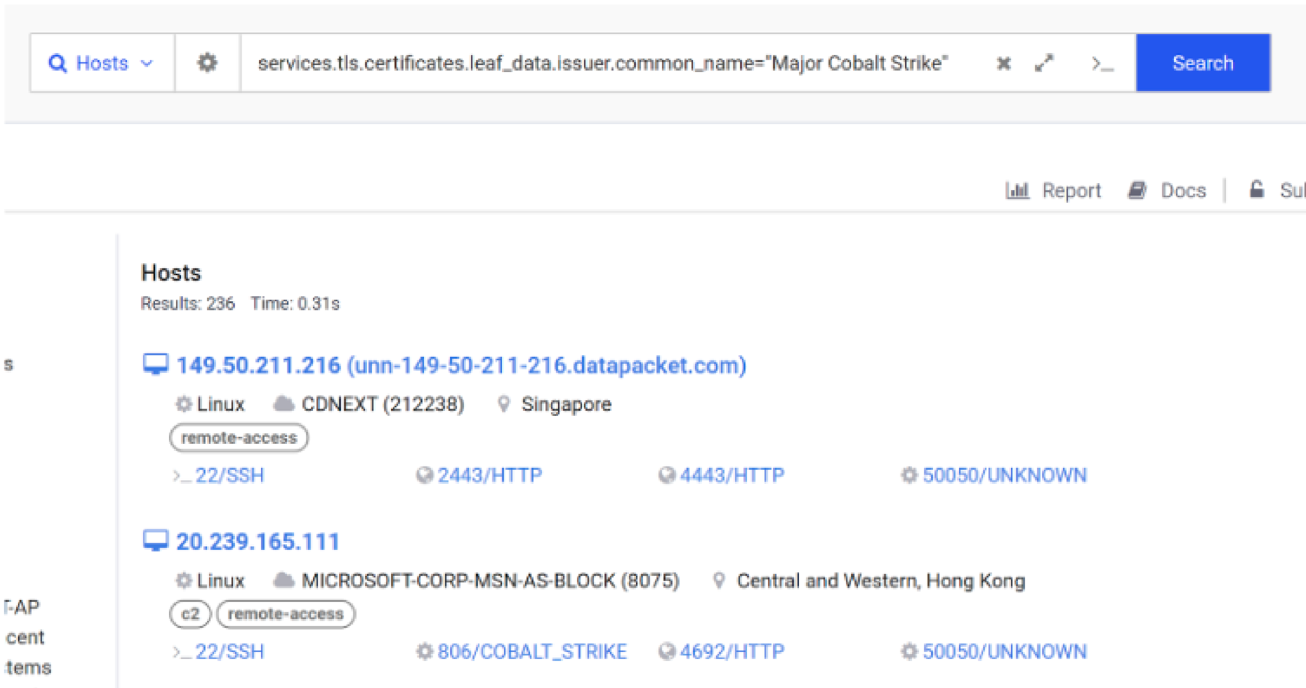


There are multiple hardcoded values here that can be utilised, but for the sake of simplicity we will leverage the issuer common name of **Major Cobalt Strike**.

We can expand the detailed host view again to determine the exact field name. **Services.tls.certificates.leaf_data.issuer.common_name**.

Querying this value returns **236** results. The chances of legitimate software containing "Major Cobalt Strike" is very low, so these are likely all active Cobalt Strike deployments.





Hunting Infrastructure with HTTP Response Titles

Developers of malware control servers often leave unique and identifying strings in web page data.

Most commonly these can be found in the HTML Titles and HTTP Bodies.

It's useful to note that these values can be changed, but many actors do not go to this effort and leave the identifying strings intact.

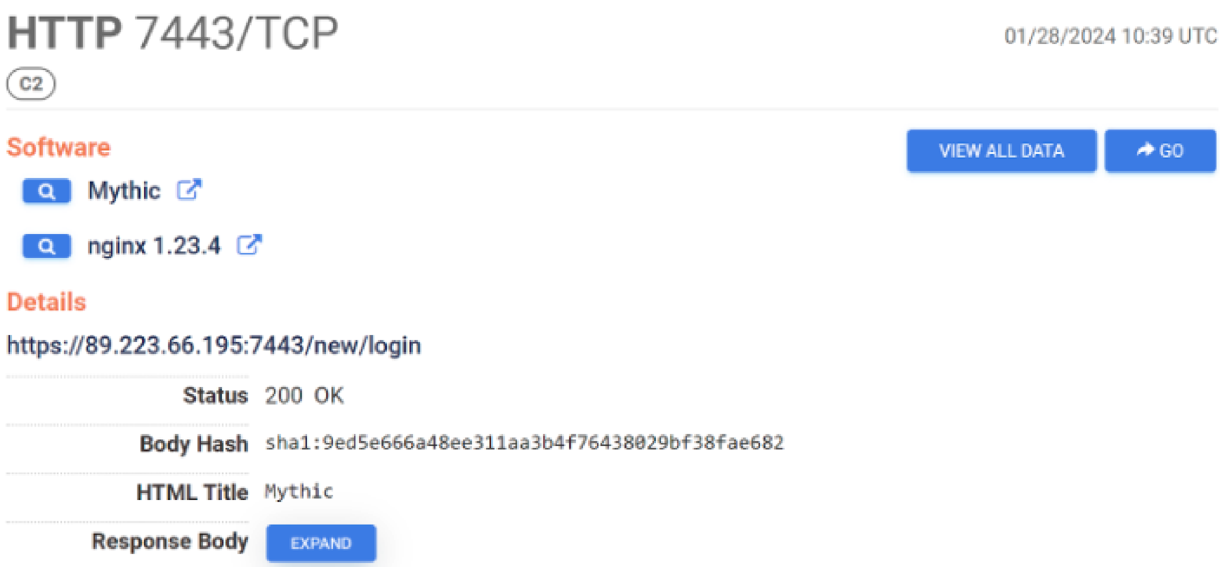
Example 1: Mythic C2 Framework

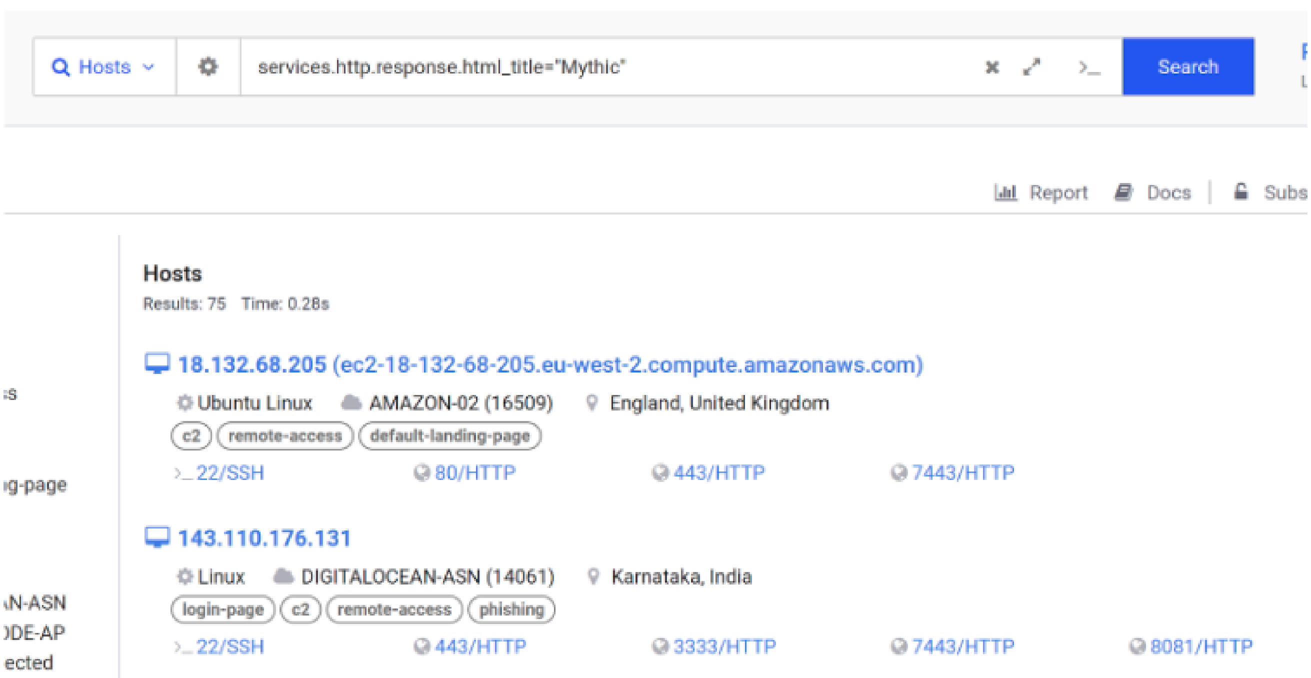
The Mythic C2 framework is often utilised by threat actors and contains a default HTML Title of **Mythic**.

Looking at the IP **89.223.66[.]195**, we can confirm the **Mythic** string present in the HTML title.

Querying for the **Mythic** string inside of **services.http.response.html_title**, we can locate a total of 75 servers.

Many of these servers have already been marked as C2 servers by the Censys platform. (You can locate other C2's with the query **labels:C2**)





Hunting Infrastructure with Service Banners

Threat actors and malware developers often leave identifying strings inside of service banners. These are often left intentionally by the author to limit misuse of the software.

These can be identified, queried, and tracked using similar methods to the HTML Titles.

Note that service banners may not be displayed by default, and you may need to open the detailed host view to see them.

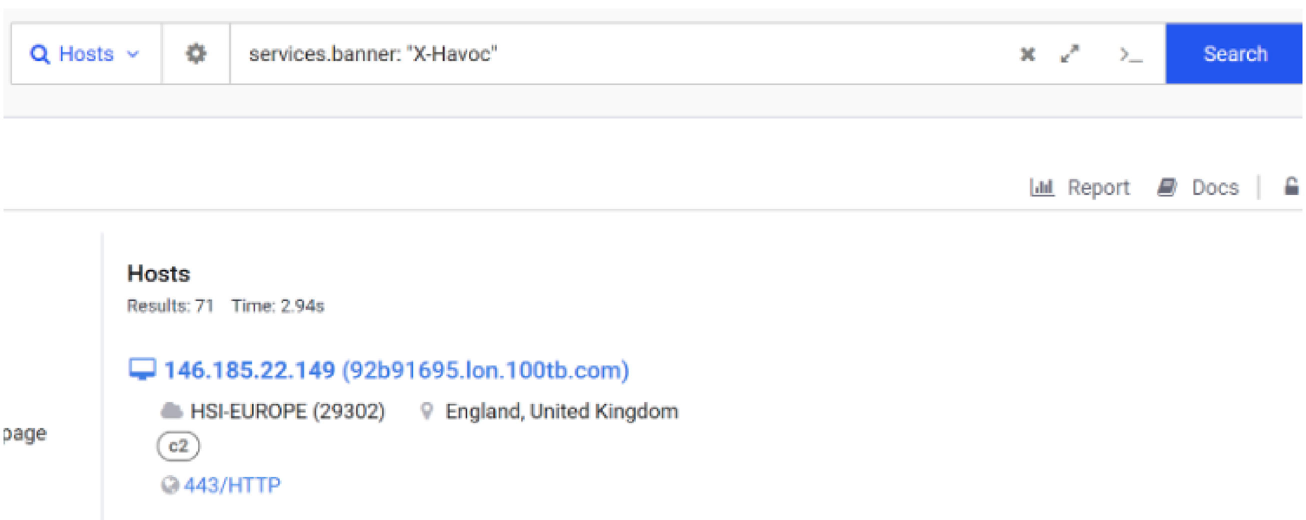
Example 1: Havoc C2 Framework

Take for example the Havoc C2 framework. Havoc is an open source C2 framework developed by C5pider that has been leveraged by threat actors due to its high quality implementation of modern offensive techniques.

With *default* settings, the Havoc Team Server contains the **X-Havoc** string inside of the service banner. This has been left intentionally by the author to limit mis-use of the software.

Searching for **services.banner** with the **X-Havoc** string returns a total of 71 results.

A string like this is specific and unlikely to have false positives. So there is a strong chance that these are all active deployments of Havoc.

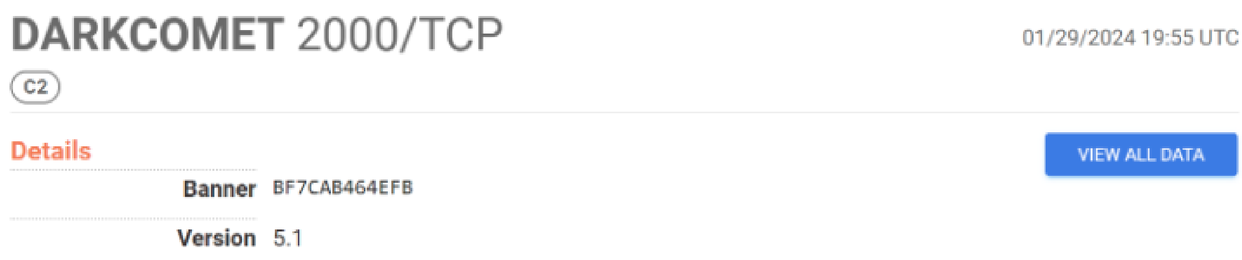


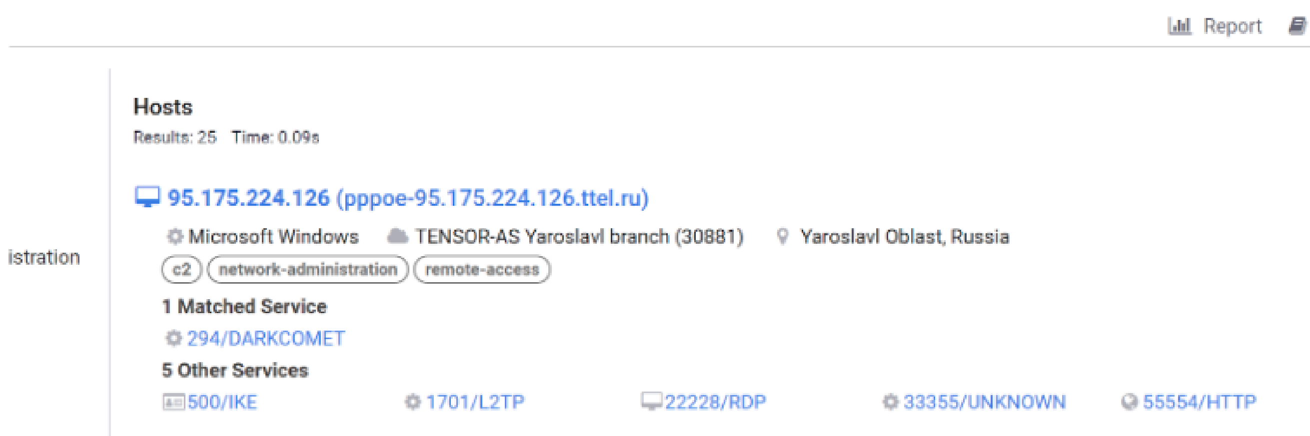
Example 2: Hunting DarkComet with Service Banners

A second example of hunting with service banners can be seen with DarkComet malware.

Looking a the IP address of **187.135.84[.]89**, we can observe a unique looking service banner of **BF7CAB464EFB** on port **2000**

We can search **services.banner** with the value **BF7CAB464EFB** to identify a total of 25 servers.





Summary – Hunting Infrastructure with Service Banners

Threat actors often work in a hurry and avoid changing default strings in custom or open source software. When investigating a host, be sure to check all service banners for unique or interesting strings.

These default strings are great indicators for building queries and you should absolutely use them to your advantage.

If you'd like to see for yourself, here is a prebuilt query for both Havoc and DarkComet.

Hunting with Locations and ASN Providers

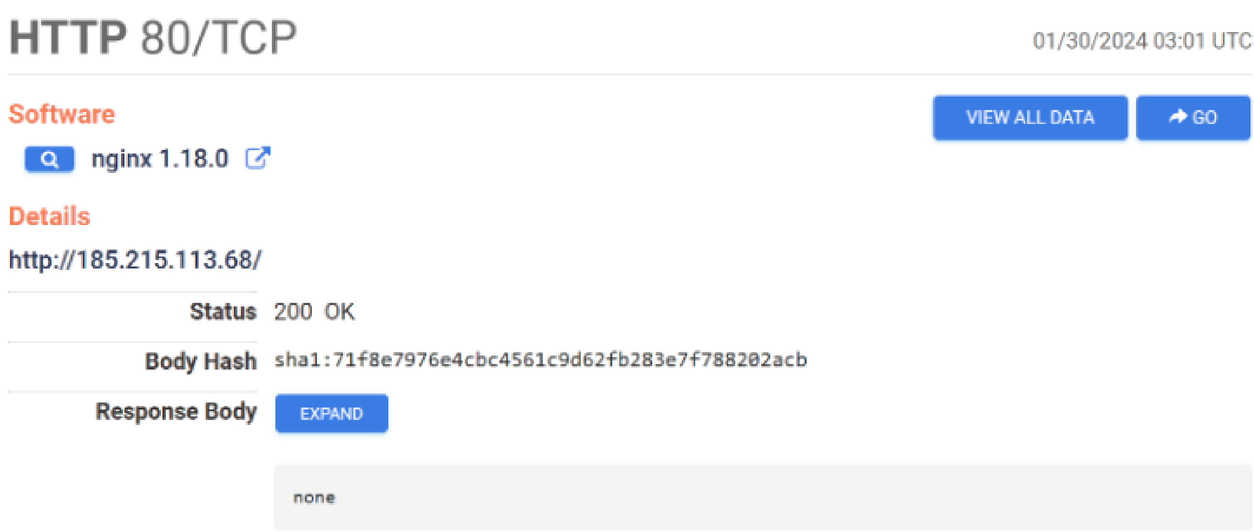
Threat actors often re-use the same hosting providers when deploying multiple servers for malicious purposes.

Often this is done to avoid takedowns and investigations. Other times it is done because hosting providers in an actor's home country are easier to obtain.

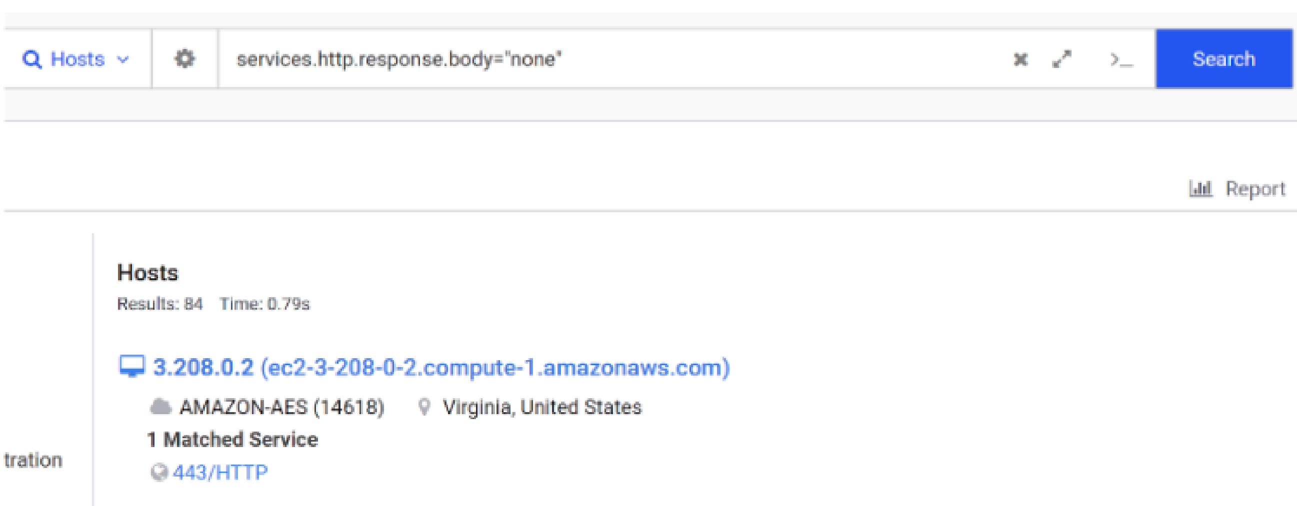
Regardless of the reasons, actors often re-use providers and we can use this to our advantage to locate malicious servers and to fine-tune existing queries on other indicators.

Example 1: Amadey Bot Servers

Let’s look at the IP address of **185.215.113[.]68** . This IP address has a relatively unique HTTP response body of **none**.



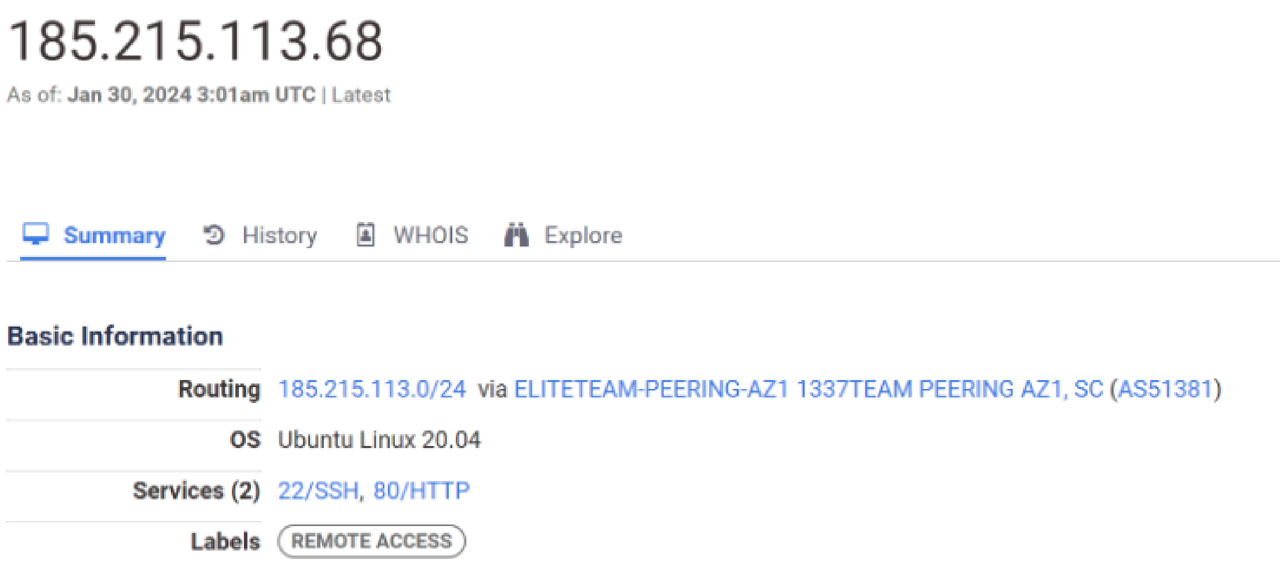
Honing in with the query of **services.http.response.body=”none”**, we have an initial result of 84 servers. Many of these servers are located in the United States and do not appear to be malicious in nature.



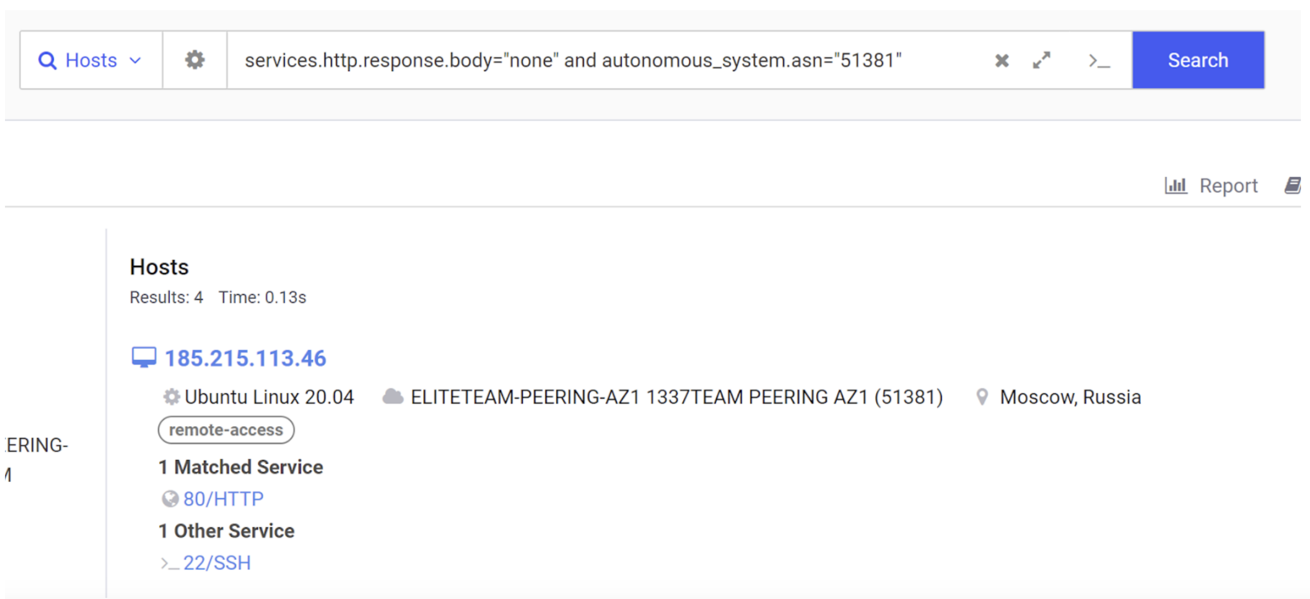
Returning to the initial results on the host page for **185.215.113[.]68**, we can see that the server is hosted in Moscow with an Autonomous System Number of **51381.**

We can add this number as an additional filter to hone in our query results.

By adding **autonomous_system.asn=51381** to our search, we have now limited our search



to only 4 results. Querying these results in Virustotal shows that they are all related to Amadey Bot malware.



Summary – Hunting with Locations and Hosting Providers

Threat actors often utilise the same hosting providers when deploying infrastructure. The hosting provider is not necessarily an indicator in itself, but it can be combined with other indicators to produce a highly effective query.

When investigating an IOC and finding that you have too many search results. Try adding the physical location of the server or the ASN Number to hone in your search.

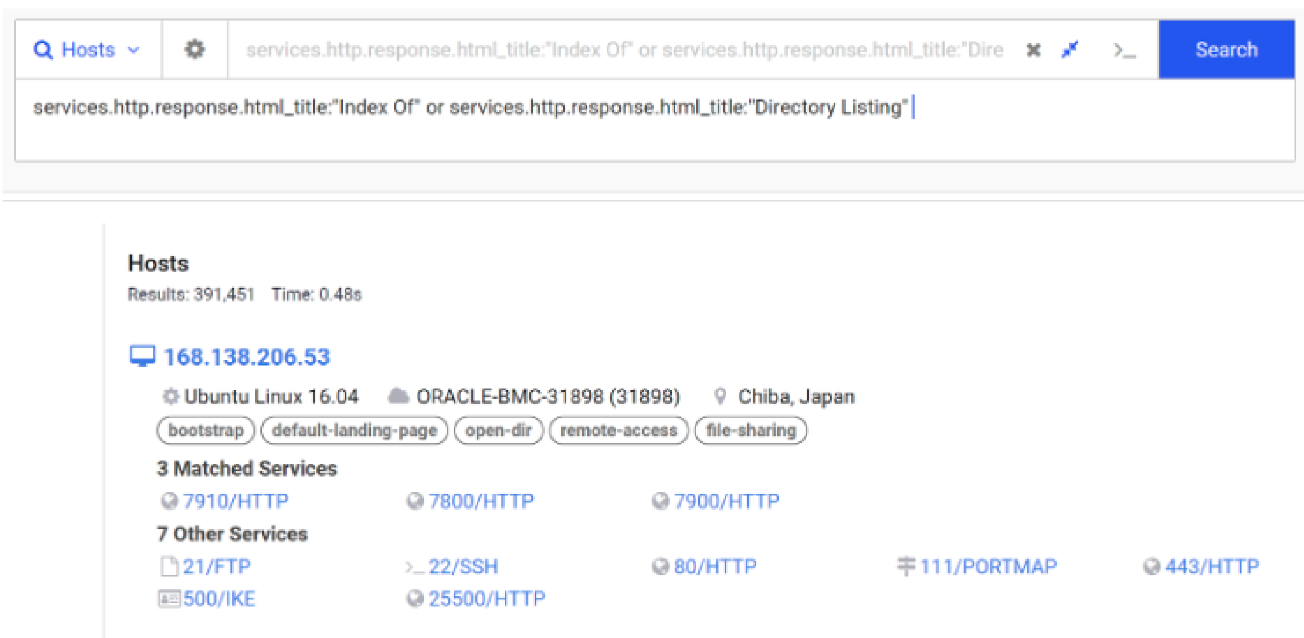
You can experiment with the Amadey example using this prebuilt query.

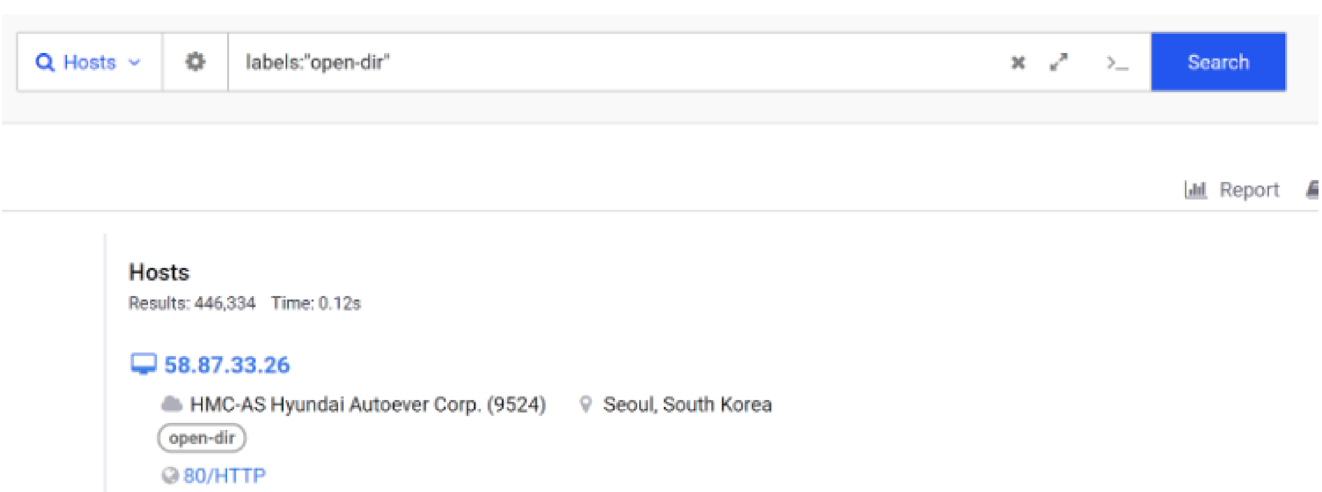
Hunting Infrastructure with Open Directories

Threat actors will often host malware and supporting software on open directories which are exposed to the public internet. These directories are generally deployed so that malware can easily retrieve additional files to facilitate exploitation.

To locate open directories, we can search for common directory titles like **Directory Listing** or **Index Of**.

Alternatively we can search for pre-labelled servers with the **open-dir** tag provided by Censys.



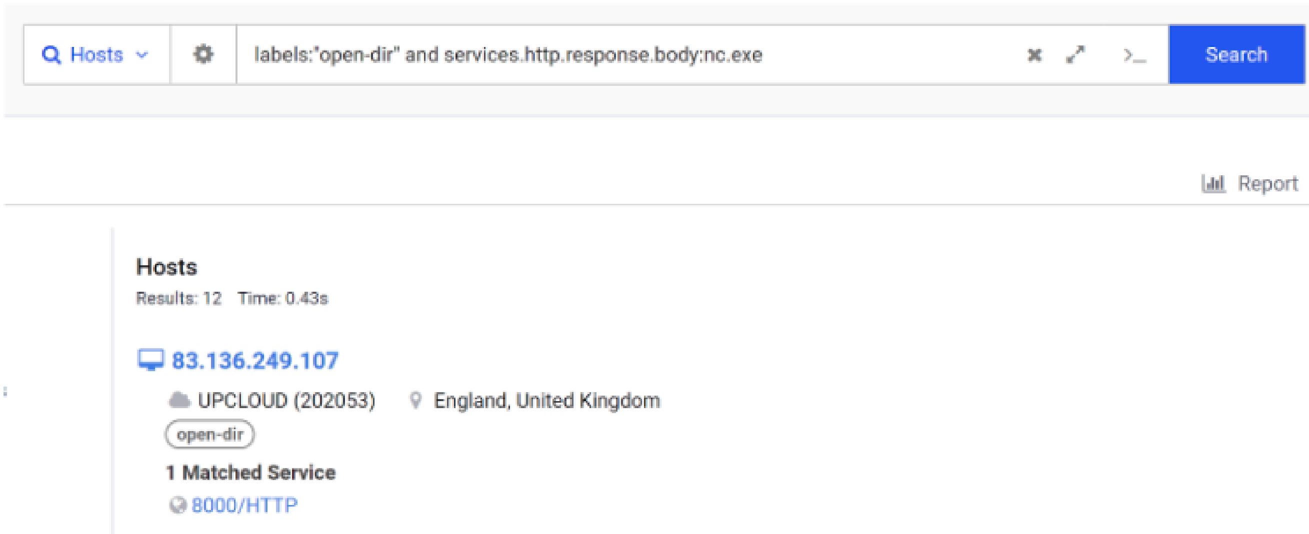


Example 1: Hunting Open Directories with Common File Names

Searching for open directories alone can return hundreds of thousands of results. Many of which are benign and non-malicious.

To identify malicious cases, we can combine the search with a specific file name related to suspicious software.

Take for example **nc.exe** which is a common file name for the netcat tool.



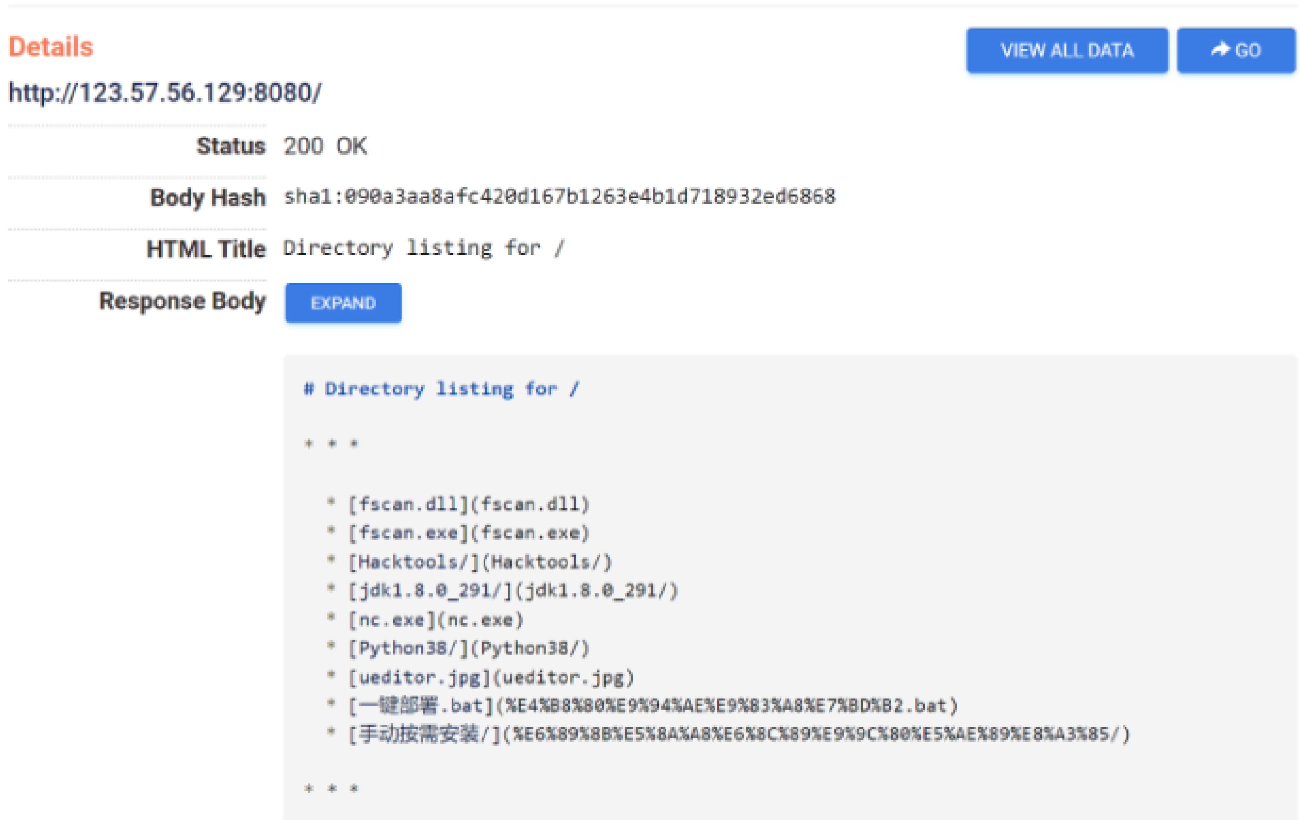
Investigating one of the first returned addresses of **123.57.56[.]129**, we can observe an open directory containing **nc.exe** as well as references to **Hacktools** and a **.bat** script with foreign characters. This information is enough to assume that the IP is highly suspicious.

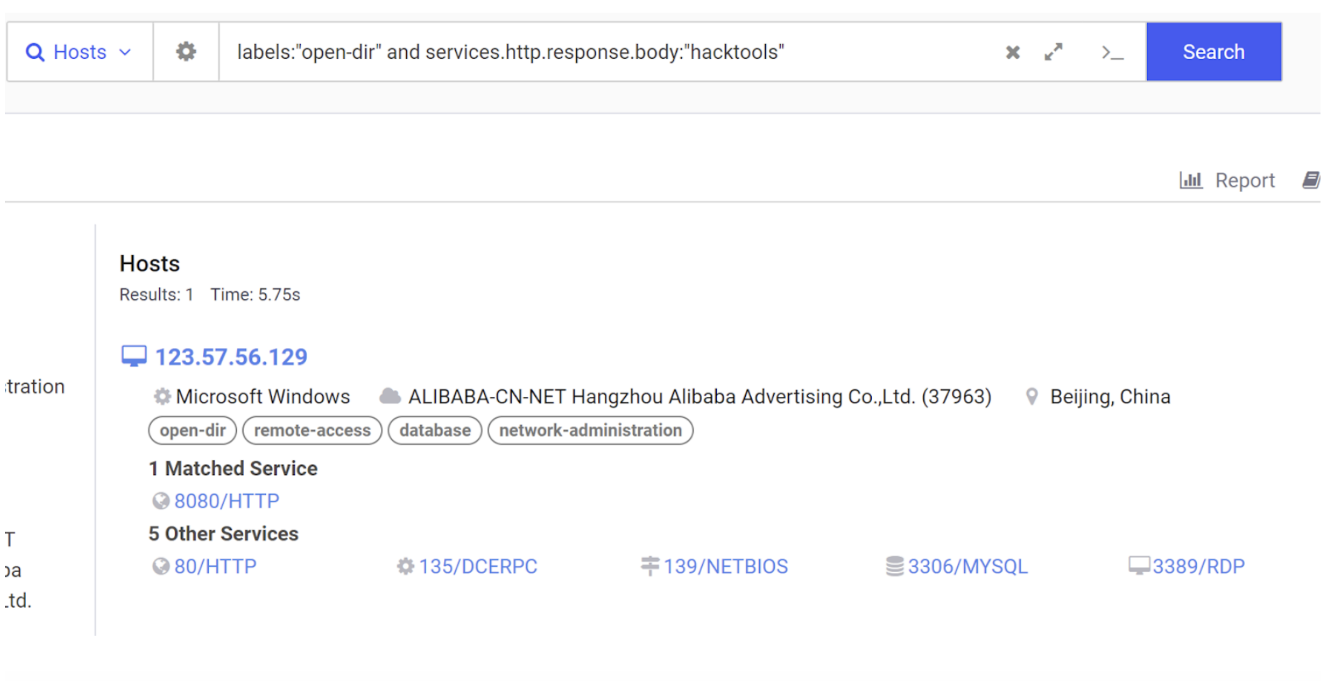
With these indicators identified we can attempt to retrieve the files to perform additional analysis and confirm malicious-ness, or we can use the new file names to refine the query and identify additional servers.

For example we can leverage the newly identified string of **hacktools** to create a new query.

We can do this with **labels:open-dir** and response bodies containing **hacktools**.

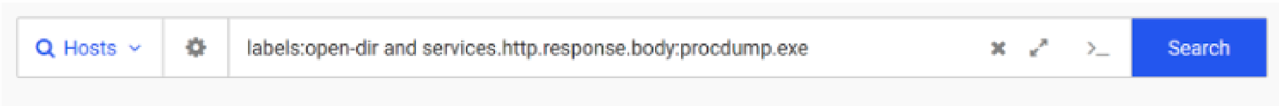
In this case only a single result is returned, but this demonstrates the concept of using initial results to establish new queries. This query could easily be re-run at a later date to identify new instances of the suspicious server.





Example 2: Open Directories Containing Procdump.exe

To demonstrate the concept further, we can search for open directories containing references to the process dumping tool **procdump.exe.**

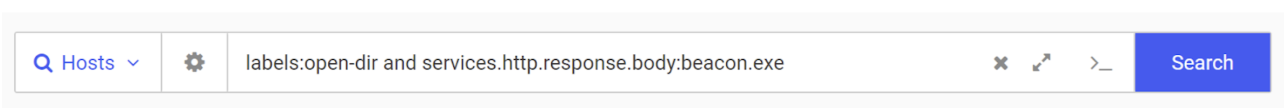


One of the results is a server hosting **procdump.exe**, **beacon.exe** and **shell.exe**.

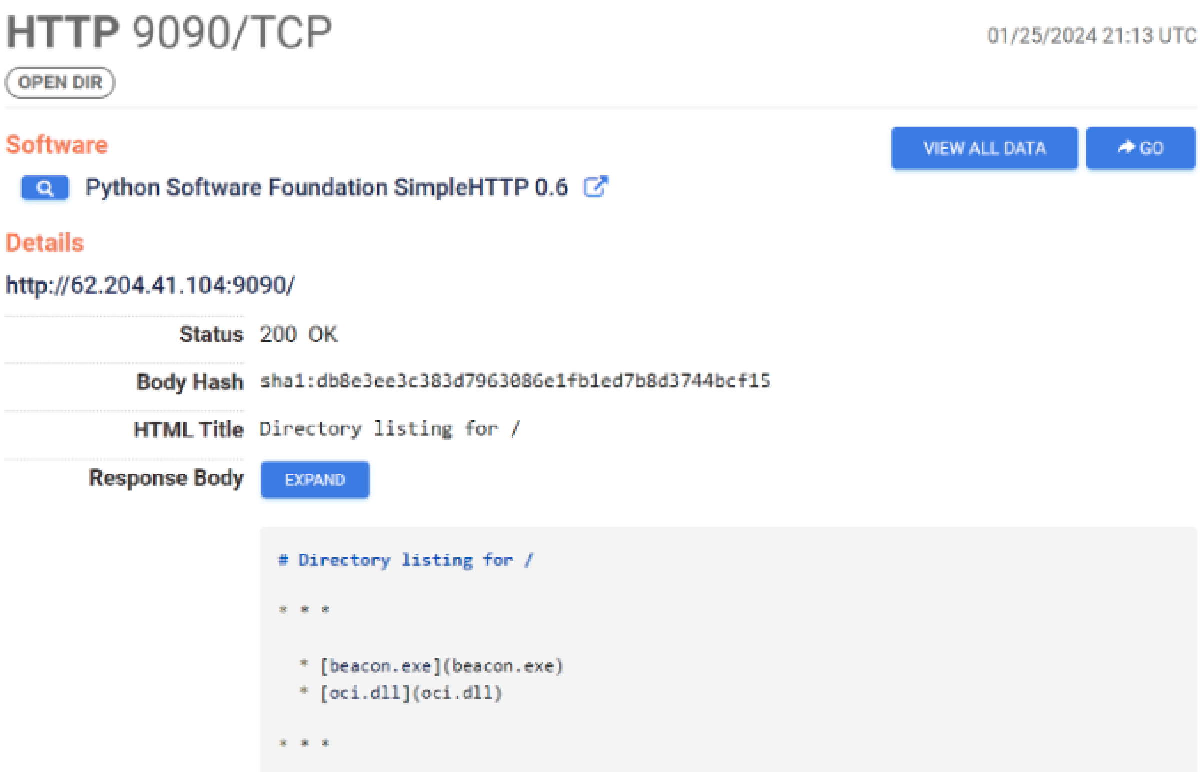
```
  * [npc](npc)
  * [npc.exe](npc.exe)
  * [passnb.jpg](passnb.jpg)
  * [passnb1.jpg](passnb1.jpg)
  * [passnb2.jpg](passnb2.jpg)
  * [passnb3.jpg](passnb3.jpg)
  * [passnb4.jpg](passnb4.jpg)
  * [passnb6.jpg](passnb6.jpg)
  * [passnb7.jpg](passnb7.jpg)
  * [pp.exe](pp.exe)
  * [procdump.exe](procdump.exe)
  * [qq.exe](qq.exe)
  * [rce.xml](rce.xml)
  * [redis/](redis/)
  * [shell.exe](shell.exe)
  * [shell1111.exe](shell1111.exe)
  * [sss.jpg](sss.jpg)
  * [windows_agent.exe](windows_agent.exe)
  * [ws123.exe](ws123.exe)
  * [yaml-payload-for-ruoyi-1.0-SNAPSHOT.jar](yaml-payload-for-ruoyi-1.0-SNAPSHOT.jar)
  * [yc.jpg](yc.jpg)
  * [yc1.jpg](yc1.jpg)
  * [yc2.jpg](yc2.jpg)
  * [yc3.jpg](yc3.jpg)

* * *
```

We can use these results to identify new strings and pivot to open directories containing **beacon.exe.**



This identifies a new server with IP **62.204.41[.]104** . This server contains references to `beacon.exe` but not the initial netcat or procdump. Highlighting the useful-ness of building new queries based on initial results.



Summary – Hunting Open Directories

Open directory hunting can be a useful means to hunt for suspicious servers. This is particularly useful when dealing with Downloader malware that has called out to a server with an open directory.

When building queries, use the pre-built **open-dir** tag and leverage known file names. Then add new file names and strings based on your results.

We've included some prebuilt queries here for beacon.exe, procdump.exe and nc.exe.

Incorporating Regular Expressions Into Hunting Queries

Advanced threat actors will often avoid using hardcoded values across multiple servers.

When unique values are deployed, this is often done via scripting and automated programs. This means that even though the values are unique, the "structure" of the values is often repeated and can be signatured using regular expressions.

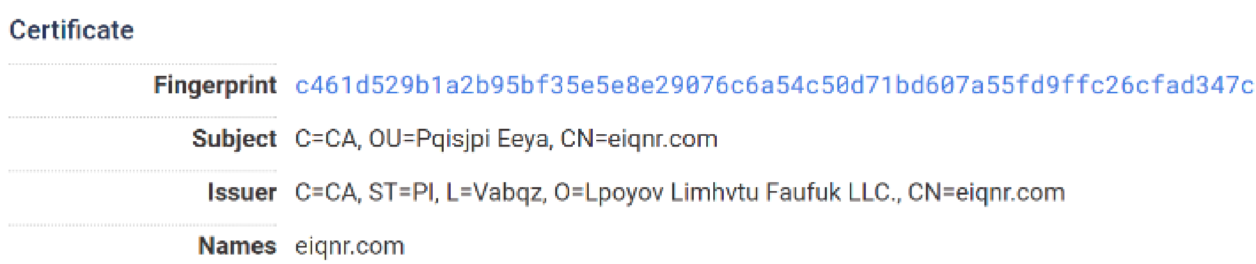
Let's take a look at some examples of unique values that can be signatured with regular expressions. **Note that Regular Expression searching is a paid feature of Censys Search.**

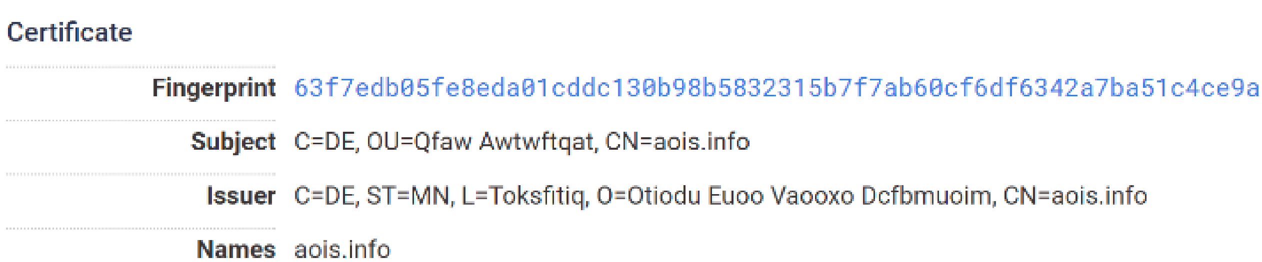
Example 1: Catching Qakbot Servers With Regular Expressions

A great example of unique values with the same structure is with Qakbot.

Qakbot uses an automated system to deploy and refresh unique TLS certificate values across servers. But these values have a similar structure which can be queried with regular expressions.

We can see two such certificates in the below screenshots.





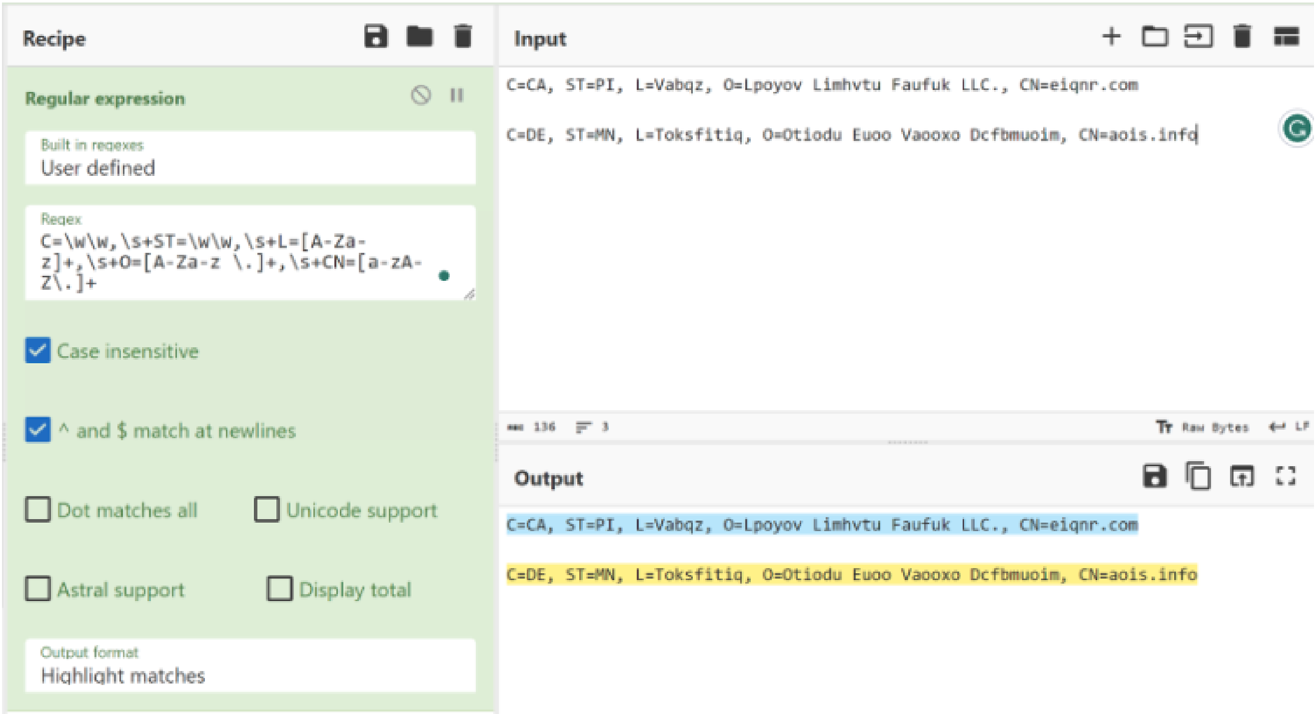
Observing the two certificate values, we can see that they are wildly different in their values. However, they follow a similar pattern which can be caught with regular expressions.

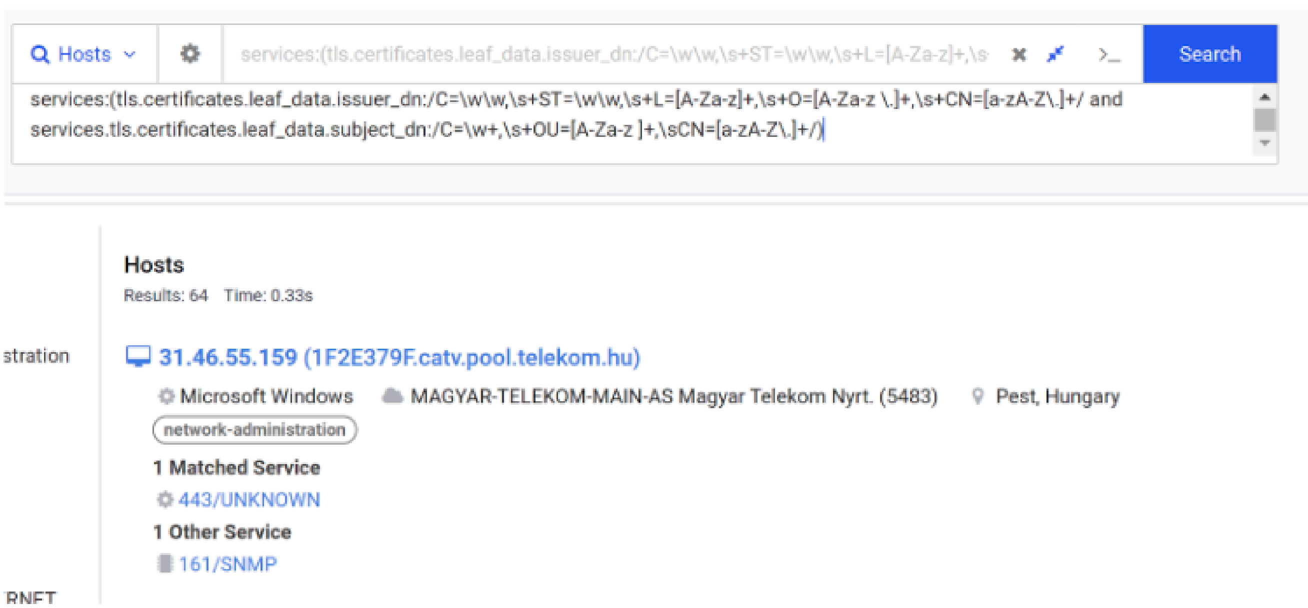
We can verify this by copying out the values and bringing in Cyberchef. This allows us to prototype a regular expression and confirm that we can match on both values.

Repeating this process for both the **issuer** and **subject fields** of the TLS certificate, we can develop a query that catches a total of 64 servers.

To verify that the results are matching as intended, we can generate a Censys report on the returned results. This allows us to list all returned certificate values in an easy-to-read list.

Below is a short snippet of the results, confirming that the query is working as intended.


Recipe
Regular expression
Built in regexes
User defined
Regex
C=\w\w,\s+ST=\w\w,\s+L=[A-Za-z]+,\s+O=[A-Za-z \.]+,\s+CN=[a-zA-Z\.]+
Case insensitive
^ and $ match at newlines
Dot matches all
Unicode support
Astral support
Display total
Output format
Highlight matches
Input
C=CA, ST=PI, L=Vabqz, O=Lpoyov Limhvtu Faufuk LLC., CN=eiqnr.com
C=DE, ST=MN, L=Toksfitiq, O=Otiodu Euoo Vaooxo Dcfbmuoim, CN=aois.info
136
3
Raw Bytes
LF
Output
C=CA, ST=PI, L=Vabqz, O=Lpoyov Limhvtu Faufuk LLC., CN=eiqnr.com
C=DE, ST=MN, L=Toksfitiq, O=Otiodu Euoo Vaooxo Dcfbmuoim, CN=aois.info


Hosts
Search
services:(tls.certificates.leaf_data.issuer_dn:/C=\w\w,\s+ST=\w\w,\s+L=[A-Za-z]+,\s+O=[A-Za-z \.]+,\s+CN=[a-zA-Z\.]+/ and services.tls.certificates.leaf_data.subject_dn:/C=\w+,\s+OU=[A-Za-z]+,\sCN=[a-zA-Z\.]+/)
Hosts
Results: 64 Time: 0.33s
stration
31.46.55.159 (1F2E379F.catv.pool.telekom.hu)
Microsoft Windows
MAGYAR-TELEKOM-MAIN-AS Magyar Telekom Nyrt. (5483)
Pest, Hungary
network-administration
1 Matched Service
443/UNKNOWN
1 Other Service
161/SNMP
RNET

Report for Hosts

services.tls.certificates.leaf_data.subject_dn	services	
CN=remote.kmsmachine.com	2	1.75%
C=AT, OU=Eelyzb Dqytutfees, CN=rdtpbohoacs.com	1	0.88%
C=AT, OU=Howi Khogtejiq, CN=nqxjewqexd.biz	1	0.88%
C=AT, OU=Iwkuitendp Ueootsdp Olzwoct, CN=uzauezol.org	1	0.88%
C=AT, OU=Wlpotaekss Ngjzoev Liebuxekzo, CN=zedaugebdki.net	1	0.88%
C=AU, OU=Eibosa, CN=qeoftu.biz	1	0.88%
C=AU, OU=Hjmxajofoud, CN=gpgeeetpko.info	1	0.88%
C=AU, OU=Htdelewnjei Opyozwbyi, CN=eobtfizyit.mobi	1	0.88%
C=AU, OU=Itaovuk Zeom Iltf, CN=tapiyeo.us	1	0.88%
C=AU, OU=Noteecakfax, CN=qkgadtolzoc.info	1	0.88%
C=AU, OU=Nuhr Ozdzoia Joomajswez, CN=malam.info	1	0.88%

Example 2: Catching BianLian Servers with Regular Expressions

The BianLian malware is another example of unique TLS certificate values containing identical structure and formatting.

Below we can observe two certificates. Both the **subject** and **issuer** contain only the “C”, “O” and “OU” fields. (This contrasts with a “typical” certificate, which would contain also contain the “ST” and “L” fields)

Each value contains exactly 16 characters with no spacing or use of special characters.

Certificate

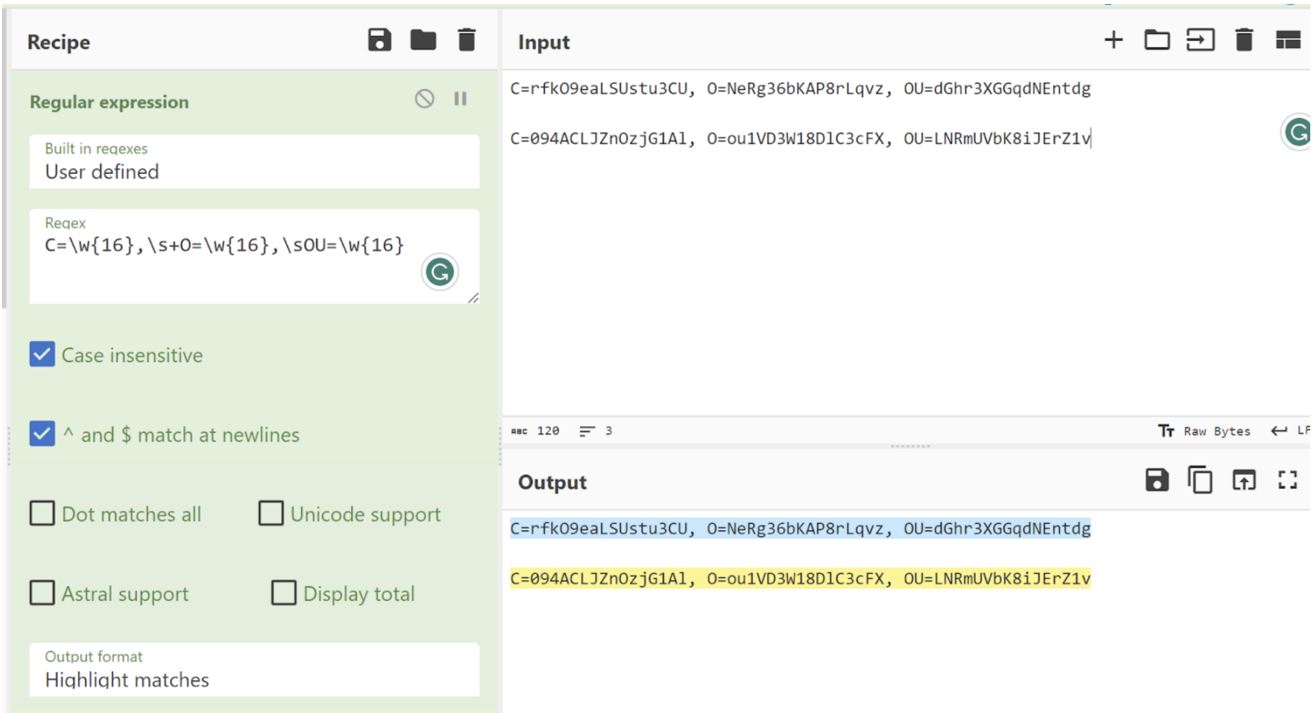
Fingerprint	a5d9dffa09a10e19b4e61a2abef4465e1b335429f6186d91b3a3216108695a1f
Subject	C=oYU8X79VU603mcrd, O=sgUG3eezeeFh1az0, OU=WWZUbScgyoHpoHYv
Issuer	C=094ACLJZnOzjG1Al, O=ou1VD3W18DlC3cFX, OU=LNRmUVbK8iJErZ1v

Certificate

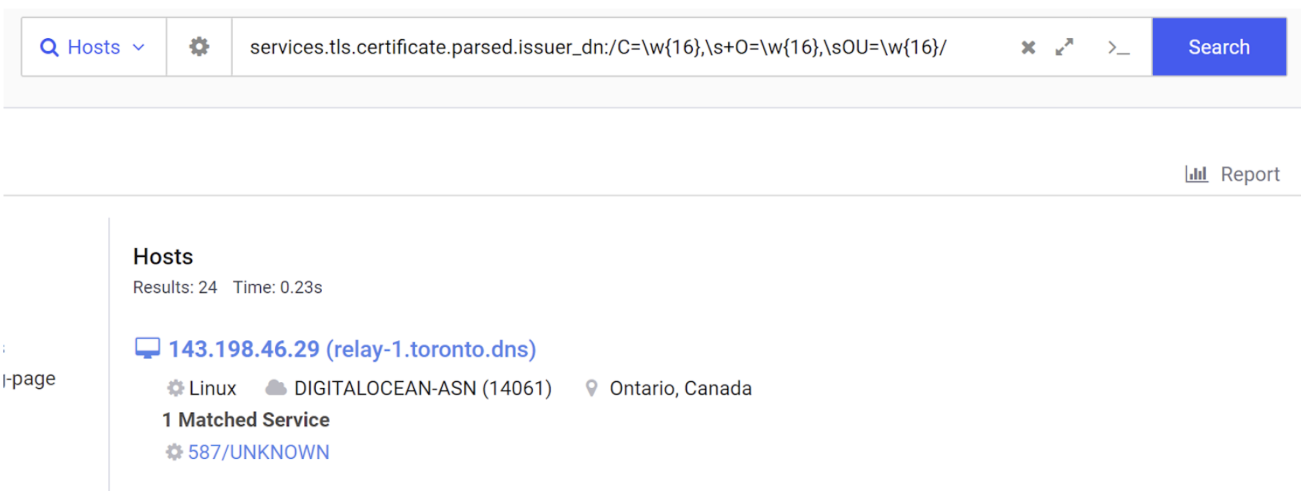
Fingerprint	cd3ed7f5f23859ca72dcba6df594e2535aa35941e8651d56cda5828d6811152f
Subject	C=cdYyG0sOqIzOmkla, O=8MzlX83olIoFQheF, OU=WNM2ElERlNG6OdJQ
Issuer	C=rfkO9eaLSUstu3CU, O=NeRg36bKAP8rLqvz, OU=dGhr3XGGqdNEntdg

Bringing the two issuer values to Cyberchef, we can prototype a regular expression that captures the values from both certificates.

We can now search on **services.tls.certificate.parsed.issuer_dn** with our regular



expression. This returns a total of 24 servers.



We can go ahead and generate another search report on the **services.tls.certificate.parsed.issuer_dn** to confirm that the results are matching as intended.

Report for Hosts

services.tls.certificate.parsed.issuer_dn	services	
C=US, O=Let's Encrypt, CN=R3	3	3.0%
C=US, O=Let's Encrypt, CN=Let's Encrypt Authority X3	2	2.0%
C=094ACLJZnOzjG1Al, O=ou1VD3W18DlC3cFX, OU=LNRmUVbK8iJErZ1v	1	1.0%
C=2KKrOzBHepYxrbkp, O=AntfFMeepCkP8nJt, OU=rEEQ1P3AMWRFWQML	1	1.0%
C=2QFc9Rb5lfJhPJEh, O=rqbp6JEyhwOBsa3B, OU=2jPhSdmtRzbwwcFN	1	1.0%
C=3ZKOxKCC7cOBl14g, O=BCZRdpf9U0WMgNqk, OU=drsboydsRdoKedmH	1	1.0%
C=3xUqggOLSNpJGPh3, O=akay9vQ4P6d2firL, OU=jA1OWJiuiXHG4zZn	1	1.0%
C=56Wbxj2fx8isgpNs, O=5UUPZwsL8YiwdGes, OU=mABs3sa958Q5Rwvj	1	1.0%
C=5kRvuhImaYNbwlKM, O=vCe7a9ukesWYkxfR, OU=c4pyKiCnqQBPwflj	1	1.0%
C=84BIxK3ZM0h0o6Zb, O=tuFIPgYoc7xhB2h5, OU=PpjWino9qdqMjzqT	1	1.0%
C=AHUZRRFlkuXGCthQ, O=43azCVfPSWJZiYlH, OU=nkt1ZcCx5v7TAYBP	1	1.0%

Example 3: Viper Servers with TLS Certificates and Regular Expressions

To demonstrate one more example, we can take a look at an IP of **139.155.90[.]81** which was marked as Viper Malware on ThreatFox.

We can view the TLS certificate information in Censys. Showing **subject** and **issuer** fields that are exactly 8 characters in length and contain only lowercase letters and numbers.

TLS

Handshake

Version Selected	TLSv1_2
Cipher Selected	TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256

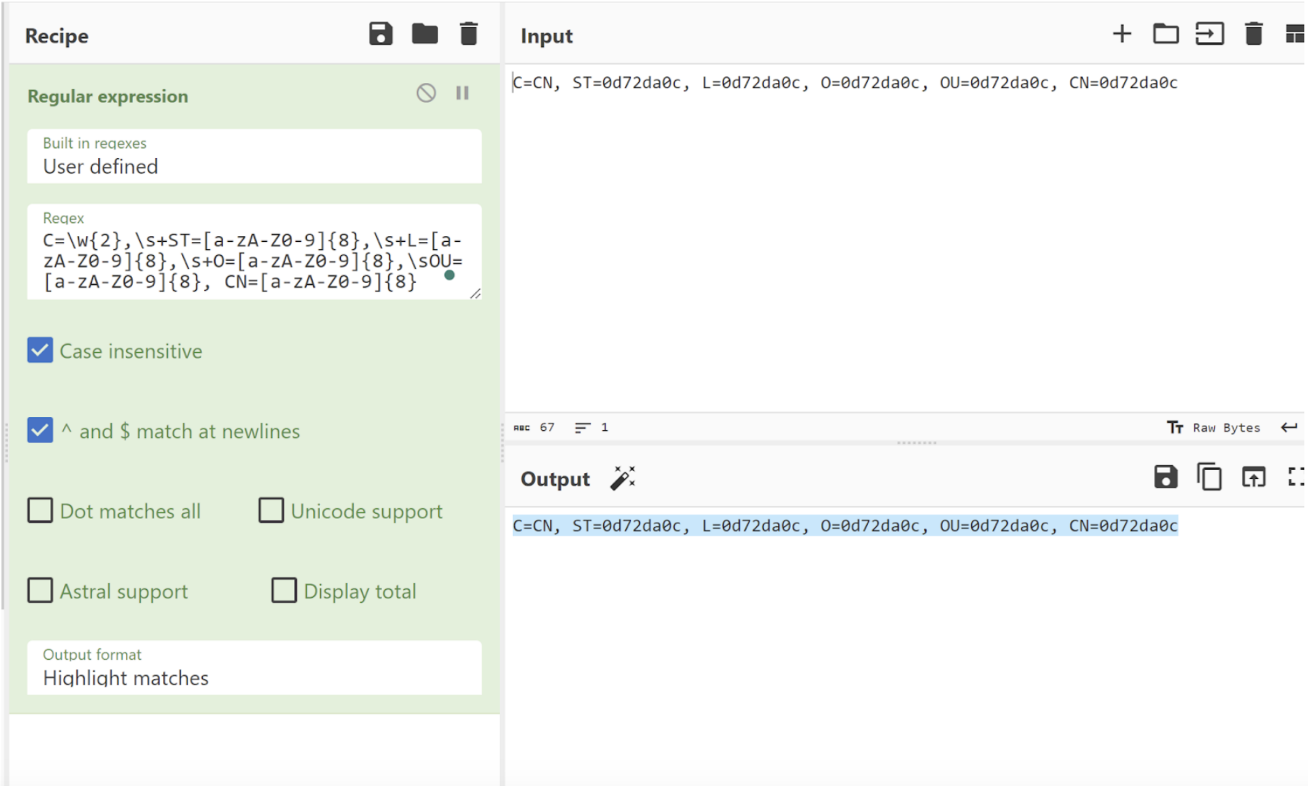
Certificate

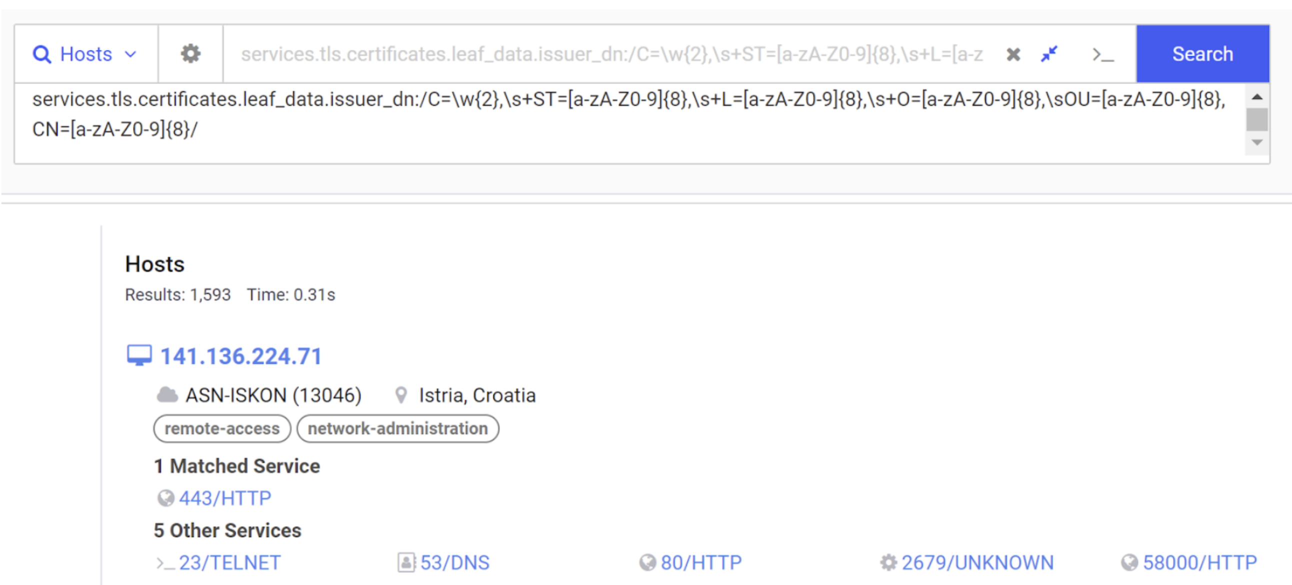
Fingerprint	4de3278507c89d2242a12c20b74878e3f84970c463a924771f156a3da7d7b5a1
Subject	C=CN, ST=d1d38ec9, L=d1d38ec9, O=d1d38ec9, OU=d1d38ec9, CN=d1d38ec9
Issuer	C=CN, ST=0d72da0c, L=0d72da0c, O=0d72da0c, OU=0d72da0c, CN=0d72da0c

Bringing one of the values into Cyberchef, we can again prototype a regular expression to match on the identified structure.

We can then search for this regular expression in the **services.tls.certificates.leaf_data.issuer_dn field.** This returns a total of 1593 results.

Generating another search report verifies that many of these results contain the same TLS structure as the initial server.


Recipe
Regular expression
Built in regexes
User defined
Regex
C=\w{2},\s+ST=[a-zA-Z0-9]{8},\s+L=[a-zA-Z0-9]{8},\s+O=[a-zA-Z0-9]{8},\sOU=[a-zA-Z0-9]{8}, CN=[a-zA-Z0-9]{8}
Case insensitive
^ and $ match at newlines
Dot matches all
Unicode support
Astral support
Display total
Output format
Highlight matches
Input
C=CN, ST=0d72da0c, L=0d72da0c, O=0d72da0c, OU=0d72da0c, CN=0d72da0c
67
1
Raw Bytes
Output
C=CN, ST=0d72da0c, L=0d72da0c, O=0d72da0c, OU=0d72da0c, CN=0d72da0c


Hosts
services.tls.certificates.leaf_data.issuer_dn:/C=\w{2},\s+ST=[a-zA-Z0-9]{8},\s+L=[a-z
Search
services.tls.certificates.leaf_data.issuer_dn:/C=\w{2},\s+ST=[a-zA-Z0-9]{8},\s+L=[a-zA-Z0-9]{8},\s+O=[a-zA-Z0-9]{8},\sOU=[a-zA-Z0-9]{8}, CN=[a-zA-Z0-9]{8}/
Hosts
Results: 1,593 Time: 0.31s
141.136.224.71
ASN-ISKON (13046)
Istria, Croatia
remote-access
network-administration
1 Matched Service
443/HTTP
5 Other Services
23/TELNET
53/DNS
80/HTTP
2679/UNKNOWN
58000/HTTP

C=CN, ST=00c8d2b8, L=00c8d2b8, O=00c8d2b8, OU=00c8d2b8, CN=00c8d2b8	1	0.02%
C=CN, ST=047e8a74, L=047e8a74, O=047e8a74, OU=047e8a74, CN=047e8a74	1	0.02%
C=CN, ST=0550a67c, L=0550a67c, O=0550a67c, OU=0550a67c, CN=0550a67c	1	0.02%
C=CN, ST=09bf642b, L=09bf642b, O=09bf642b, OU=09bf642b, CN=09bf642b	1	0.02%
C=CN, ST=14ed08de, L=14ed08de, O=14ed08de, OU=14ed08de, CN=14ed08de	1	0.02%
C=CN, ST=297762af, L=297762af, O=297762af, OU=297762af, CN=297762af	1	0.02%
C=CN, ST=363d6d9f, L=363d6d9f, O=363d6d9f, OU=363d6d9f, CN=363d6d9f	1	0.02%
C=CN, ST=3eea653a, L=3eea653a, O=3eea653a, OU=3eea653a, CN=3eea653a	1	0.02%
C=CN, ST=415d6028, L=415d6028, O=415d6028, OU=415d6028, CN=415d6028	1	0.02%
C=CN, ST=5bd251a1, L=5bd251a1, O=5bd251a1, OU=5bd251a1, CN=5bd251a1	1	0.02%
C=CN, ST=5e8651d1, L=5e8651d1, O=5e8651d1, OU=5e8651d1, CN=5e8651d1	1	0.02%
C=CN, ST=5f04e43e, L=5f04e43e, O=5f04e43e, OU=5f04e43e, CN=5f04e43e	1	0.02%
C=CN, ST=685ed9c2, L=685ed9c2, O=685ed9c2, OU=685ed9c2, CN=685ed9c2	1	0.02%
C=CN, ST=6b68e6f7, L=6b68e6f7, O=6b68e6f7, OU=6b68e6f7, CN=6b68e6f7	1	0.02%
C=CN, ST=6be9de02, L=6be9de02, O=6be9de02, OU=6be9de02, CN=6be9de02	1	0.02%
C=CN, ST=6e87138f, L=6e87138f, O=6e87138f, OU=6e87138f, CN=6e87138f	1	0.02%
C=CN, ST=87bb429a, L=87bb429a, O=87bb429a, OU=87bb429a, CN=87bb429a	1	0.02%
C=CN, ST=99625580, L=99625580, O=99625580, OU=99625580, CN=99625580	1	0.02%

Summary – Hunting Infrastructure with Regular Expressions

There will be cases where hardcoded values won't be enough to hunt infrastructure.

Many of these situations can be handled by identifying the structure of seemingly unique values and incorporating regular expressions into your queries.

If you're unfamiliar with regular expressions, there is a great free resource over at regexone that will help you get started.

How Can I Get Started?

All of the queries (excluding regular expressions) can be performed with a Censys Search Community account. You can sign up today and begin threat hunting, gathering intelligence, and building up lists of IOCs.

To obtain initial IOCs, we recommend using public IOC repositories like ThreatFox and URLHaus and starting your hunt from there. We can also recommend leveraging pre-built queries like those shared by drb_ra and Michael Koczwara.

Conclusion

We've now looked at several indicators that can be used to identify malicious infrastructure. You can and should use all of these to your advantage when investigating an IP address or performing a threat hunt.

Threat actors vary in quality and sophistication, some will be more difficult to track than others. But in many cases you can track actors using only the techniques shown here today.

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.