

# Building Advanced Threat Intel Queries Utilising Regex and TLS Certificates - (BianLian)

embee-research.ghost.io/building-advanced-censys-queries-utilising-regex-bianlian/

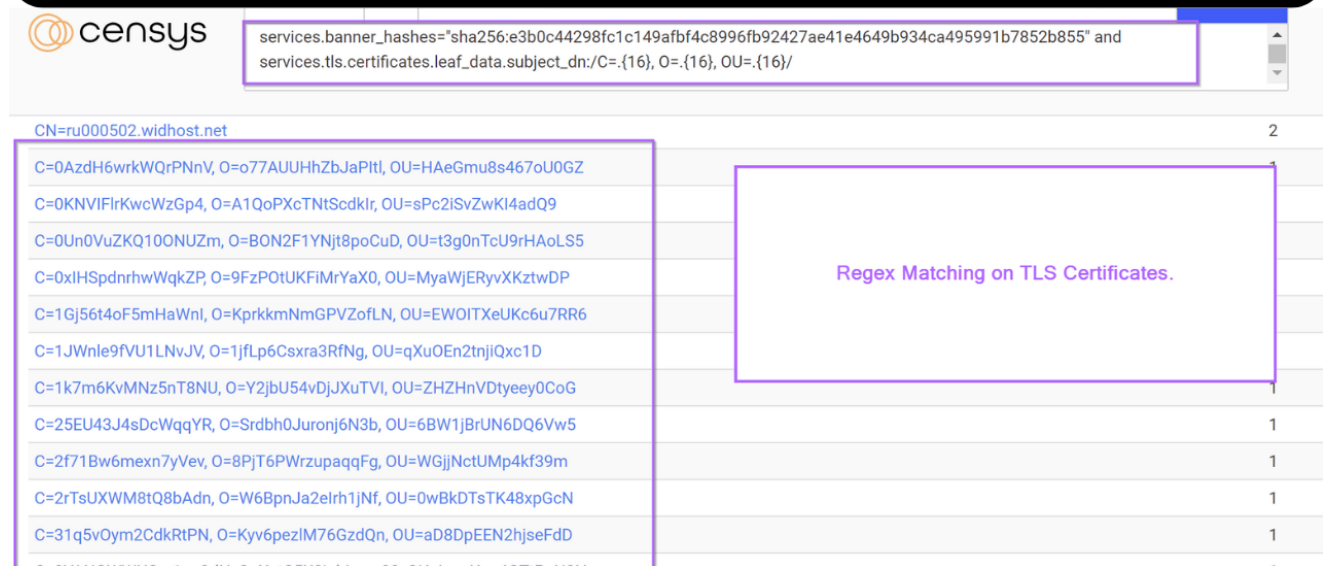
Matthew

November 27, 2023

## Advanced

Creating Regex Signatures on TLS Certificates with Censys.

## Regex Matching on TLS Certificates



The screenshot shows the Censys search interface. The search query is: `services.banner_hashes="sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855" and services.tls.certificates.leaf_data.subject_dn:/C={16}, O={16}, OU={16}/`. The results table shows a list of certificates with their subject DN fields. A purple box highlights the search query, and another purple box highlights the text "Regex Matching on TLS Certificates." overlaid on the results table.

Result	Count
CN=ru000502.widhost.net	2
C=0AzdH6wrkWqrPNnV, O=o77AUUHhZbJaPtl, OU=HAeGmu8s467oU0GZ	
C=0KNVIFlrKwcWzGp4, O=A1QoPXcTntScdKlr, OU=sPc2iSvZwKl4adQ9	
C=0Un0VuZKQ100NUZm, O=BON2F1YNjt8poCuD, OU=t3g0nTcU9rHAoLS5	
C=0xiHSpdnrhwWqkZP, O=9FzP0tUKFiMrYaX0, OU=MyaWJERyvXKztwDP	
C=1Gj56t4oF5mHaWnl, O=KprkkmNmGPVZofLN, OU=EW0ITXeUKc6u7RR6	
C=1JWnle9fVU1LNvJV, O=1jflp6Csxra3RfNg, OU=qXu0En2tnjiQxc1D	
C=1k7m6KvMNz5nT8NU, O=Y2jbU54vDjJXuTVI, OU=ZHZHnVDtyeey0CoG	
C=25EU43J4sDcWqqYR, O=Srdbh0Juronj6N3b, OU=6BW1JBrUN6DQ6Vw5	1
C=2f71Bw6mexn7yVev, O=8PjT6PWrzupaqqFg, OU=WGjjNctUMp4kf39m	1
C=2rTsUXWM8tQ8bAdn, O=W6BpnJa2eirh1jNf, OU=0wBkDTsTK48xpGcN	1
C=31q5vOym2CdkRTPN, O=Kyv6pezIM76GzdQn, OU=aD8DpEEN2hjseFdD	1

In this post we will investigate a Bianlian C2 address and use TLS certificates to obtain another ~50 servers.

Our primary focus will be building a regex query that matches the unique TLS structure used by recent BianLian.

Regex and TLS Certs present a great opportunity to build queries without introducing geographic limitations like ASN numbers and hosting providers. By using TLS Certificates, we can also avoid limiting searches to specific port numbers or port counts. Effectively catching actors that avoid patterns demonstrated in previous posts. (Eg [here](#) and [here](#))

Our final query will look something like this.

```
services.banner_hashes="sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855" and services.tls.certificates.leaf_data.subject_dn:/C={16}, O={16}, OU={16}/
```

Note that regular expression searches are a paid feature of Censys. But for the purposes of demonstrating interesting concepts I thought this blog would be useful

At the end of this blog, there is a query that will catch the same certificates without utilising Regex. This can be used if you don't have a paid Censys account.

## Performing an Initial Search

We can begin the search by looking up the initial IP of `185.248.100[.]118` which was obtained on ThreatFox.

Note that we can prepend `ip:` if we want an overview before jumping directly to the IP page.

The screenshot shows the Censys search interface. At the top, a search bar contains the query `ip:185.248.100.118`. Below the search bar, the results page for the host `185.248.100.118` is displayed. The host information includes the operating system (Linux), the network (IPSERVER-RU-NET Fiord (44812)), and the location (Moscow, Russia). A list of services is shown below the host information, including `_22/SSH`, `137/NETBIOS`, `443/UNKNOWN`, and `631/IPP`. A purple box highlights the text "Initial IP with 4 running services." and another purple box highlights the service list.

Clicking on the IP Address takes us to the full page, where we can see an interesting TLS certificate on port 443.

# UNKNOWN 443/TCP

11/25/2023 06:20 UTC

## Software

linux

[VIEW ALL DATA](#)

## Details

### TLS

#### Handshake

Version Selected TLSv1\_3

Cipher Selected TLS\_CHACHA20\_POLY1305\_SHA256

#### Certificate

Fingerprint c4b42e9c59f8f79da83385d08d3876dcd19987be433fd6148abe32d14254e8af

Subject C=ID5hgJb31CGtxS3R, O=NgOiQK7LZP5nKyTE, OU=fcr8shEwbsebOGQc

Issuer C=QaXvvhYVZUBot5hh, O=YUs8vBiShD2OI59X, OU=JptfXUKIKxJYLGdX

#### Fingerprint

JARM 3fd21b20d3fd3fd21c43d21b21b43d1ec49a4b64df0a9e9f328abd60285841

JA3S 475c9302dc42b2751db9edcac3b74891

If we click on the "view all data" tab in the top right of the above screenshot, we can view the full information for port 443.

This enables us to see all available information for a given port, which is typically significantly more than what is available within the default summary view.

The first thing that stands out is the empty `services.banner` field. A lack of service banner (although boring at initial glance) could be a great pivot point.

443/UNKNOWN TCP

[View Definition](#)

Attribute	Value	
services.banner		<input type="button" value="Q"/>
services.banner_hashes	sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855	<input type="button" value="Q"/>
services.certificate	c4b42e9c59f8f79da83385d08d3876dcd19987be433fd6148abe32d14254e8af	<input type="button" value="Q"/>
services.discovery_method	PREDICTIVE_METHOD_20	<input type="button" value="Q"/>
services.extended_service_name	UNKNOWN	<input type="button" value="Q"/>
services.jarm.fingerprint	3fd21b20d3fd3fd21c43d21b21b43d1ec49a4b64df0a9e9f328abd60285841	<input type="button" value="Q"/>
services.jarm.cipher_and_version_fingerprint	3fd21b20d3fd3fd21c43d21b21b43d	<input type="button" value="Q"/>
services.jarm.tls_extensions_sha256	1ec49a4b64df0a9e9f328abd60285841	<input type="button" value="Q"/>
services.jarm.observed_at	2023-11-25T11:39:07.471152539Z	<input type="button" value="Q"/>
services.observed_at	2023-11-25T06:20:35.669842333Z	<input type="button" value="Q"/>
services.perspective_id	PERSPECTIVE_NTT	<input type="button" value="Q"/>
services.port	443	<input type="button" value="Q"/>

We can click on the blue search box next to the `services.banner` field to perform a pivot. This will pre-build the query and save us time finding the exact field to search.

It's often better to use the banner hash when we want an exact match, this will avoid any issues where a value might be incorrectly typed when searching. This also ensures an exact match is queried.

Performing a basic pivot from the banner hash, we have an initial result of 14,605,188 servers. An admittedly huge number, But we'll improve this later.

The screenshot shows a search interface with a search bar containing the query: `services.banner_hashes="sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e464"`. Below the search bar, a list of results is displayed. The first result is for IP **45.60.8.222**, located in Florida, United States, with 14790 UNKNOWN services. The second result is for IP **45.60.64.238**, located in California, United States, with 14083 UNKNOWN services. Both results show 3 other services: 53/DNS, 21/HTTP, and 631/IPP.

If we go back to the initial summary view, we can see that the service was identified as "Unknown".

# UNKNOWN 443/TCP

11/25/2023 06:20 UTC

## Software

linux

[VIEW ALL DATA](#)

## Details

### TLS

#### Handshake

Version Selected TLSv1\_3

Cipher Selected TLS\_CHACHA20\_POLY1305\_SHA256

#### Certificate

Fingerprint c4b42e9c59f8f79da83385d08d3876dcd19987be433fd6148abe32d14254e8af

Subject C=ID5hgJb31CGtxS3R, O=NgOiQK7LZP5nKyTE, OU=fcr8shEwbsebOGQc

Issuer C=QaXvvhYVZUBot5hh, O=YUs8vBiShD2OI59X, OU=JptfXUKIKxJYLGdX

#### Fingerprint

JARM 3fd21b20d3fd3fd21c43d21b21b43d1ec49a4b64df0a9e9f328abd60285841

JA3S 475c9302dc42b2751db9edcac3b74891

We can again use the "view all data" to find the "Unknown" field and perform a pivot with the search box.

services.port	443	<a href="#">Q</a>
services.service_name	UNKNOWN	<a href="#">Q</a>
services.software.uniform_resource_identifier	cpe:2.3:o:*:linux:*:*:*:*:*	<a href="#">Q</a>
services.software.port	o	<a href="#">Q</a>

This returns a lot of results, so we can take the resulting query and add it to our initial empty banner search.

[Hosts](#)  [Search](#)

[Report](#)

### Hosts

Results: 26,408,012 Time: 0.51s

[49.232.125.116](#)

TENCENT-NET-AP Shenzhen Tencent Computer Systems Company Limited (45090) Beijing, China

#### 8 Matched Services

3454/UNKNOWN

3656/UNKNOWN

3695/UNKNOWN

4242/UNKNOWN

4369/UNKNOWN

3004/UNKNOWN

3138/UNKNOWN

3337/UNKNOWN

#### 2 Other Services

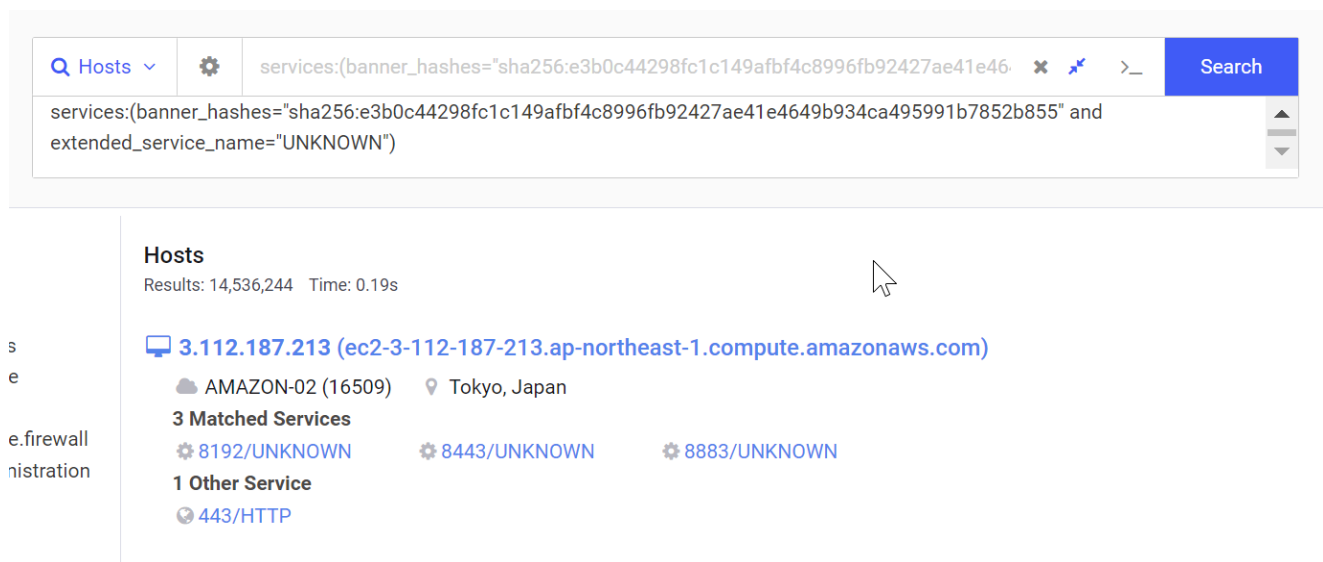
5601/HTTP

9200/HTTP

Sadly this doesn't actually reduce the hits to any useful degree (in this case), but it's a useful technique to know so I wanted to include it in the post.

Since the addition of `UNKNOWN1` only reduced the hits by a very very small degree, it doesn't matter whether we include it or not for the remainder of this search.

For the remaining queries we can leave it out.



The screenshot shows a search interface with a search bar containing the query: `services:(banner_hashes="sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855" and extended_service_name="UNKNOWN")`. Below the search bar, the results for the host `3.112.187.213 (ec2-3-112-187-213.ap-northeast-1.compute.amazonaws.com)` are displayed. The host is located in Tokyo, Japan, and has 3 matched services: `8192/UNKNOWN`, `8443/UNKNOWN`, and `8883/UNKNOWN`, and 1 other service: `443/HTTP`.

Moving on, we can go back to the field details page and observe the certificate information in more detail.

Of particular interest here is that only the `C=`, `O=` and `OU=` fields are present in the subject and issuer fields.

Typically there are additional values like Location `L`, State `ST` and others. The absence of these values is an additional indicator that can be used.

For example, a "normal" certificate should look more like this.

#### Certificate

**Fingerprint** 61b39fafe53fbc0db8497849456c48ecf22c0168d60b44c52b93c6fd2aea30c6

**Subject** C=CN, ST=Jiangsu, L=Nanjing, O=Huawei Software Technologies Co., Ltd., CN=n.cdnhwc3.com

**Issuer** C=BE, O=GlobalSign nv-sa, CN=GlobalSign RSA OV SSL CA 2018

We can also note that all of the `C=`, `O=`, `OU=` values are exactly 16 characters in length.

The lack of "regular" fields and presence of exactly 16 characters is itself a pattern worth signaturing.

services.source_ip	167.248.133.49	<input type="text" value="🔍"/>
services.tls.version_selected	TLSv1_3	<input type="text" value="🔍"/>
services.tls.cipher_selected	TLS_CHACHA20_POLY1305_SHA256	<input type="text" value="🔍"/>
services.tls.certificates.leaf_fp_sha_256	c4b42e9c59f8f79da83385d08d3876dcd19987be433fd6148abe32d14254e8af	<input type="text" value="🔍"/>
services.tls.certificates.leaf_data.subject_dn	C=ID5hgJb31CGtxS3R, O=NgOiQK7LZP5nKyTE, OU=fcr8shEwbsebOGQc	<input type="text" value="🔍"/>
services.tls.certificates.leaf_data.issuer_dn	C=QaXvvhYVZUBot5hh, O=YUs8vBiShD2OI59X, OU=JptfXUKIKxJYLgDX	<input type="text" value="🔍"/>
services.tls.certificates.leaf_data.pubkey_bit_size	2048	<input type="text" value="🔍"/>
services.tls.certificates.leaf_data.pubkey_algorithm	RSA	<input type="text" value="🔍"/>
services.tls.certificates.leaf_data.tbs_fingerprint	f704519c6a204975d691fb4a7ae8f669ce457dd8872857e56e050a2f9a62ad52	<input type="text" value="🔍"/>

We can use the blue search box again (right side of the detailed fields view) to automatically build an exact query.

The screenshot shows a search interface with a query bar containing the following text: `services.tls.certificates.leaf_data.subject_dn="C=ID5hgJb31CGtxS3R, O=NgOiQK7LZP5nKyTE, OU=fcr8shEwbsebOGQc"`. Below the query bar, the search results are displayed for the host **185.248.100.118**. The results include the operating system (Linux), IP address (IPSERVER-RU-NET Fiord (44812)), location (Moscow, Russia), and a list of services: 1 Matched Service (443/UNKNOWN), 3 Other Services (22/SSH, 137/NETBIOS, 631/IPP).

## Converting a Field to Regex

The initial pre-built query is an exact string search. (Indicated by the presence of = and the entire thing encased in quotes)

```
services.tls.certificates.leaf_data.subject_dn="C=ID5hgJb31CGtxS3R,
O=NgOiQK7LZP5nKyTE, OU=fcr8shEwbsebOGQc"
```

We can turn this into a regex by adjusting the equals = to a colon : and changing the double quotes " to forward slashes /

We can then validate the syntax by modifying the query and ensuring the same result is returned.

The screenshot shows a search interface with a search bar containing the query: `services.tls.certificates.leaf_data.subject_dn:/C=ID5hgJb31CGtxS3R, O=NgOiQK7LZLZP5nKyTE, OU=fcr8shEwbsebOGQc/`. Below the search bar, the results section is titled "Hosts" and shows "Results: 1 Time: 0.21s". The single result is for IP `185.248.100.118`, which is identified as a Linux system with the name `IPSERVER-RU-NET Fiord (44812)` located in `Moscow, Russia`. It has a `remote-access` tag and lists services: `1 Matched Service` (`443/UNKNOWN`) and `3 Other Services` (`22/SSH`, `137/NETBIOS`, `631/IPP`).

After validating the syntax, we can remove the main values and replace them with `.{16}`.

This ensures that exactly 16 characters are present following any of the `C=`, `O=` or `OU=` values.

In a real example you would probably want to avoid the `.` wildcard and use a more specific query like `[a-zA-Z0-9]{16}`, `[^,]{16}` or `\w{16}`. But for this example we will keep it simple with a `.`

Adjusting the query returns `56` results for matching certificates.

The screenshot shows a search interface with a search bar containing the query: `services.tls.certificates.leaf_data.subject_dn:/C=.{16}, O=.{16}, OU=.{16}/`. Below the search bar, the results section is titled "Hosts" and shows "Results: 56 Time: 0.17s". The first result is for IP `91.102.162.229`, identified as `SISTEMDC (211804)` in `Istanbul, Turkey`. It has tags for `email`, `file-sharing`, and `remote-access`. It lists `2 Matched Services` (`80/UNKNOWN`, `443/HTTP`) and `11 Other Services` (`21/FTP`, `22/SSH`, `25/SMTP`, `53/DNS`, `110/POP3`, `143/IMAP`, `465/SMTP`, `587/SMTP`, `993/IMAP`, `995/POP3`, `8083/HTTP`).

## Validating a Regex Query with the "Build Report" Feature



By using the "report" button in the top-right corner, we can view a specific field for all of our returned results.

This can be extremely useful for validating that results are matching as intended.

Note in the screenshot below, that the "build report" function will show results for other services running on returned servers.

For example, a server with a matching hit on port 443, may also have a non-matching hit on port 8080. If both services contain a relevant field, they both will be displayed. Hence the additional results that don't match our query.

Either way this function is still useful for validating results.

The screenshot shows a search interface with a 'Breakdown Field' input containing 'services.tls.certificates.leaf\_data.subject\_dn' and a 'Number of Buckets' input set to 100. A blue 'BUILD REPORT' button is visible. Below the search is a table titled 'Report for Hosts' with columns for the field name, the subject DN value, and a percentage. A callout box highlights a row with the subject DN 'C=0AzdH6wrkWqrPNnV, O=o77AUUHHZbJaPtlI, OU=HAeGmu8s467oU0GZ'.

services.tls.certificates.leaf_data.subject_dn	Percentage
C=CN, ST=BJ, L=BJ, O=AODUN, OU=AODUN	3.1%
CN=srv82054434.ultrasrv.net	2.79%
CN=localhost	2.48%
C=XX, ST=XX, L=XX, O=XX, OU=XX, CN=u22.isp, emailAddress=root@u22.isp	2.17%
C=Ychxl0xoT8nxSLaO, O=DNmluS6s9MUK5LR4, OU=qs08cf2XLedOalub	1.55%
C=ru000502.widhost.net	1.24%
C=0AzdH6wrkWqrPNnV, O=o77AUUHHZbJaPtlI, OU=HAeGmu8s467oU0GZ	0.62%
C=0KNVIFirKwcWzGp4, O=A1QoPXcTntScdklr, OU=sPc2iSvZwKI4adQ9	0.31%
C=0Un0VuZKQ100NUZm, O=BON2F1YNjt8poCuD, OU=t3g0nTcU9rHAoLS5	0.31%
C=0xlHSpdnhwWqkZP, O=9FzPOTUKFiMrYaX0, OU=MyaWjERyvXKztwDP	0.31%
C=1Gj56t4oF5mHaWnl, O=KprkkmNmGPVZofLN, OU=EW0ITXeUKc6u7RR6	0.31%
C=1JWnle9fVU1LNvJV, O=1jflp6Csxra3RfNg, OU=qXuOEn2tnjQxc1D	0.31%
C=1k7m6KvMNz5nT8NU, O=Y2jbu54vDjXuTVI, OU=ZHZHnVDtyeey0CoG	0.31%
C=25EU43J4sDcWqqYR, O=Srdhb0Juronj6N3b, OU=6BW1jBrUN6DQ6Vw5	0.31%
C=2f71Bw6mexn7yVev, O=8PJT6PWrzupaqqFg, OU=WGjjNctUMp4kf39m	0.31%
C=2rTsUXWM8tQ8bAdn, O=W6BpnJa2elrh1jNf, OU=DwBkDTsTK48xpGcN	0.31%
C=31q5vOym2CdkRtPN, O=Kyy6pezlM76GzdQn, OU=aD8DpEEN2hjseFdD	0.31%
C=3YANGWwNSm1pg9dN, O=XetC5X8jubLzgz08, OU=lacqXuo4GTtPaN8N	0.31%
C=3smmnyE72wwcX0D1, O=ogSDhLWQbCFMhhSW, OU=ZvSdWMEhNFHZddsl	0.31%

With the query now validated further, we can add it to our initial empty banner search.

```
services.banner_hashes="sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855" and services.tls.certificates.leaf_data.subject_dn:/C={16}, O={16}, OU={16}/
```

This results in 53 returned servers.

Q Hosts services.banner\_hashes="sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e464 x >\_ Search

services.banner\_hashes="sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855" and services.tls.certificates.leaf\_data.subject\_dn:/C=.{16}, O=.{16}, OU=.{16}/

---

**Hosts**  
Results: 53 Time: 0.27s

**213.139.205.146**  
 Linux SHOCK-1 (395092) Hesse, Germany  
 remote-access

**6 Matched Services**  
 443/UNKNOWN 5000/UNKNOWN 6388/UNKNOWN 6806/UNKNOWN 8000/UNKNOWN  
 8080/UNKNOWN

**1 Other Service**  
 >\_ 22/SSH

**2.58.14.41 (mta0.zampa.space)**  
 Linux CROWNCLOUD (199959) North Holland, Netherlands  
 remote-access


**1 Matched Service**  
 443/UNKNOWN

**1 Other Service**  
 >\_ 22/SSH

## Investigating Hits With VirusTotal

Investigating the first 2 hits (that aren't our initial IP) with VirusTotal, we can see that both have been marked as BianLian trojan.

Did you intend to search across the file corpus instead? [Click here](#)



Community Score ⌵

⚠ 8 security vendors flagged this IP address as malicious

2.58.14.41 (2.58.14.0/23)


AS 199959 ( Gwy It Pty Ltd)

---

DETECTION
DETAILS
RELATIONS
COMMUNITY 1

Join the [VT Community](#) and enjoy additional community insights and crowdsourced detections, plus an API key to [automate checks](#).

**Comments (1)** ⓘ




**drb\_ra**

📅 3 days ago

Bianlian Go Trojan Found  
 C2: 2[.]58[.]14[.]41:443  
 Certificate: c4be8524b7026424d88445b0243fde79e006c775444123f5aa5fe96a06048464  
 Country: Netherlands (AS199959)  
 ASN: CROWNCLOUD

#c2 #Bianlian\_Go\_Trojan

Did you intend to search across the file corpus instead? [Click here](#)



Community Score ⌵

⚠ 11 security vendors flagged this IP address as malicious

213.139.205.146 (213.139.204.0/22)


AS 395092 ( SHOCK-1)

---

DETECTION
DETAILS
RELATIONS
COMMUNITY 4

Join the [VT Community](#) and enjoy additional community insights and crowdsourced detections, plus an API key to [automate checks](#).

**Comments (4)** ⓘ



**drb\_ra**

📅 2 days ago

Bianlian Go Trojan Found  
 C2: 213[.]139[.]205[.]146:8080  
 Certificate: 66768ca9454d70e25ccc297fb7232085754deb302f9b84f25a69026c0fd81d95  
 Country: Germany (AS395092)  
 ASN: SHOCK-1

#c2 #Bianlian\_Go\_Trojan

To save time scrolling through the page, we can use the **Build Report** feature again to obtain an easy list of all returned IP's.

services, you could query for `services.service_name: http` and then generate a report on the breakdown of the field `services.port`.

Breakdown Field ip	Number of Buckets 50	<a href="#">BUILD REPORT</a>
-----------------------	-------------------------	------------------------------

## Report for Hosts

ip	hosts
<a href="#">2.58.14.41</a>	1 1.92%
<a href="#">3.76.100.131</a>	1 1.92%
<a href="#">13.59.168.154</a>	1 1.92%
<a href="#">13.215.227.78</a>	1 1.92%
<a href="#">13.215.228.73</a>	1 1.92%
<a href="#">23.152.0.64</a>	1 1.92%
<a href="#">34.207.174.202</a>	1 1.92%
<a href="#">34.219.121.232</a>	1 1.92%
<a href="#">43.139.241.58</a>	1 1.92%
<a href="#">45.45.219.141</a>	1 1.92%
<a href="#">45.56.165.27</a>	1 1.92%
<a href="#">45.56.165.30</a>	1 1.92%
<a href="#">45.80.151.49</a>	1 1.92%
<a href="#">45.86.163.188</a>	1 1.92%
<a href="#">45.86.163.224</a>	1 1.92%

We can also scroll down to the JSON output and combine it with CyberChef to obtain a text-based list.

# JSON Report

```
{
  "query": "services.banner_hashes=\"sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855\" and services.tls.certificates.leaf_data.subject_dn:/C={16}, O={16}, OU={16}/",
  "field": "ip",
  "total": 53,
  "duration": 1006,
  "total_omitted": 3,
  "potential_deviation": 0,
  "buckets": [
    {
      "key": "2.58.14.41",
      "count": 1
    },
    {
      "key": "3.76.100.131",
      "count": 1
    },
    {
      "key": "13.59.168.154",
      "count": 1
    },
    {
      "key": "13.215.227.78",
      "count": 1
    },
    {
      "key": "13.215.228.73",
      "count": 1
    },
    {
      "key": "23.152.0.64",
      "count": 1
    },
    {
      "key": "34.207.174.202",
      "count": 1
    },
    {
      "key": "34.219.121.232",
      "count": 1
    },
    {
      "key": "43.139.241.58",
      "count": 1
    },
    {
      "key": "45.45.219.141",
      "count": 1
    },
    {
      "key": "45.56.165.27",
      "count": 1
    },
    {
      "key": "45.56.165.30",
      "count": 1
    },
    {
      "key": "45.56.165.39",
      "count": 1
    },
    {
      "key": "45.56.165.40",
      "count": 1
    },
    {
      "key": "45.56.165.41",
      "count": 1
    },
    {
      "key": "45.56.165.42",
      "count": 1
    },
    {
      "key": "45.56.165.43",
      "count": 1
    },
    {
      "key": "45.56.165.44",
      "count": 1
    },
    {
      "key": "45.56.165.45",
      "count": 1
    },
    {
      "key": "45.56.165.46",
      "count": 1
    },
    {
      "key": "45.56.165.47",
      "count": 1
    },
    {
      "key": "45.56.165.48",
      "count": 1
    },
    {
      "key": "45.56.165.49",
      "count": 1
    },
    {
      "key": "45.56.165.50",
      "count": 1
    },
    {
      "key": "45.56.165.51",
      "count": 1
    },
    {
      "key": "45.56.165.52",
      "count": 1
    },
    {
      "key": "45.56.165.53",
      "count": 1
    },
    {
      "key": "45.56.165.54",
      "count": 1
    },
    {
      "key": "45.56.165.55",
      "count": 1
    },
    {
      "key": "45.56.165.56",
      "count": 1
    },
    {
      "key": "45.56.165.57",
      "count": 1
    },
    {
      "key": "45.56.165.58",
      "count": 1
    },
    {
      "key": "45.56.165.59",
      "count": 1
    },
    {
      "key": "45.56.165.60",
      "count": 1
    },
    {
      "key": "45.56.165.61",
      "count": 1
    },
    {
      "key": "45.56.165.62",
      "count": 1
    },
    {
      "key": "45.56.165.63",
      "count": 1
    },
    {
      "key": "45.56.165.64",
      "count": 1
    },
    {
      "key": "45.56.165.65",
      "count": 1
    },
    {
      "key": "45.56.165.66",
      "count": 1
    },
    {
      "key": "45.56.165.67",
      "count": 1
    },
    {
      "key": "45.56.165.68",
      "count": 1
    },
    {
      "key": "45.56.165.69",
      "count": 1
    },
    {
      "key": "45.56.165.70",
      "count": 1
    },
    {
      "key": "45.56.165.71",
      "count": 1
    },
    {
      "key": "45.56.165.72",
      "count": 1
    },
    {
      "key": "45.56.165.73",
      "count": 1
    },
    {
      "key": "45.56.165.74",
      "count": 1
    },
    {
      "key": "45.56.165.75",
      "count": 1
    },
    {
      "key": "45.56.165.76",
      "count": 1
    },
    {
      "key": "45.56.165.77",
      "count": 1
    },
    {
      "key": "45.56.165.78",
      "count": 1
    },
    {
      "key": "45.56.165.79",
      "count": 1
    },
    {
      "key": "45.56.165.80",
      "count": 1
    },
    {
      "key": "45.56.165.81",
      "count": 1
    },
    {
      "key": "45.56.165.82",
      "count": 1
    },
    {
      "key": "45.56.165.83",
      "count": 1
    },
    {
      "key": "45.56.165.84",
      "count": 1
    },
    {
      "key": "45.56.165.85",
      "count": 1
    },
    {
      "key": "45.56.165.86",
      "count": 1
    },
    {
      "key": "45.56.165.87",
      "count": 1
    },
    {
      "key": "45.56.165.88",
      "count": 1
    },
    {
      "key": "45.56.165.89",
      "count": 1
    },
    {
      "key": "45.56.165.90",
      "count": 1
    },
    {
      "key": "45.56.165.91",
      "count": 1
    },
    {
      "key": "45.56.165.92",
      "count": 1
    },
    {
      "key": "45.56.165.93",
      "count": 1
    },
    {
      "key": "45.56.165.94",
      "count": 1
    },
    {
      "key": "45.56.165.95",
      "count": 1
    },
    {
      "key": "45.56.165.96",
      "count": 1
    },
    {
      "key": "45.56.165.97",
      "count": 1
    },
    {
      "key": "45.56.165.98",
      "count": 1
    },
    {
      "key": "45.56.165.99",
      "count": 1
    },
    {
      "key": "45.56.165.100",
      "count": 1
    }
  ]
}
```

The screenshot shows a web interface with a 'Recipe' panel on the left and an 'Input/Output' panel on the right. The 'Recipe' panel is titled 'Extract IP addresses' and has several options: 'IPv4' (checked), 'IPv6' (unchecked), 'Remove local IPv4 addresses' (unchecked), 'Display total' (unchecked), 'Sort' (unchecked), and 'Unique' (unchecked). The 'Input' panel shows a JSON query: `"query": "services.banner_hashes=\"sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855\" and services.tls.certificates.leaf_data.subject_dn:/C={16}, O={16}, OU={16}/", "field": "ip", "total": 53, "duration": 1006, "total_omitted": 3, "potential_deviation": 0, "buckets": [{"key": "2.58.14.41", "count": 1}, {"key": "3.76.100.131", "count": 1}, {"key": "13.59.168.154", "count": 1}, {"key": "13.215.227.78", "count": 1}, {"key": "13.215.228.73", "count": 1}, {"key": "23.152.0.64", "count": 1}, {"key": "34.207.174.202", "count": 1}, {"key": "34.219.121.232", "count": 1}, {"key": "43.139.241.58", "count": 1}, {"key": "45.45.219.141", "count": 1}, {"key": "45.56.165.27", "count": 1}, {"key": "45.56.165.30", "count": 1}, {"key": "45.56.165.39", "count": 1}, {"key": "45.56.165.40", "count": 1}, {"key": "45.56.165.41", "count": 1}, {"key": "45.56.165.42", "count": 1}, {"key": "45.56.165.43", "count": 1}, {"key": "45.56.165.44", "count": 1}, {"key": "45.56.165.45", "count": 1}, {"key": "45.56.165.46", "count": 1}, {"key": "45.56.165.47", "count": 1}, {"key": "45.56.165.48", "count": 1}, {"key": "45.56.165.49", "count": 1}, {"key": "45.56.165.50", "count": 1}, {"key": "45.56.165.51", "count": 1}, {"key": "45.56.165.52", "count": 1}, {"key": "45.56.165.53", "count": 1}, {"key": "45.56.165.54", "count": 1}, {"key": "45.56.165.55", "count": 1}, {"key": "45.56.165.56", "count": 1}, {"key": "45.56.165.57", "count": 1}, {"key": "45.56.165.58", "count": 1}, {"key": "45.56.165.59", "count": 1}, {"key": "45.56.165.60", "count": 1}, {"key": "45.56.165.61", "count": 1}, {"key": "45.56.165.62", "count": 1}, {"key": "45.56.165.63", "count": 1}, {"key": "45.56.165.64", "count": 1}, {"key": "45.56.165.65", "count": 1}, {"key": "45.56.165.66", "count": 1}, {"key": "45.56.165.67", "count": 1}, {"key": "45.56.165.68", "count": 1}, {"key": "45.56.165.69", "count": 1}, {"key": "45.56.165.70", "count": 1}, {"key": "45.56.165.71", "count": 1}, {"key": "45.56.165.72", "count": 1}, {"key": "45.56.165.73", "count": 1}, {"key": "45.56.165.74", "count": 1}, {"key": "45.56.165.75", "count": 1}, {"key": "45.56.165.76", "count": 1}, {"key": "45.56.165.77", "count": 1}, {"key": "45.56.165.78", "count": 1}, {"key": "45.56.165.79", "count": 1}, {"key": "45.56.165.80", "count": 1}, {"key": "45.56.165.81", "count": 1}, {"key": "45.56.165.82", "count": 1}, {"key": "45.56.165.83", "count": 1}, {"key": "45.56.165.84", "count": 1}, {"key": "45.56.165.85", "count": 1}, {"key": "45.56.165.86", "count": 1}, {"key": "45.56.165.87", "count": 1}, {"key": "45.56.165.88", "count": 1}, {"key": "45.56.165.89", "count": 1}, {"key": "45.56.165.90", "count": 1}, {"key": "45.56.165.91", "count": 1}, {"key": "45.56.165.92", "count": 1}, {"key": "45.56.165.93", "count": 1}, {"key": "45.56.165.94", "count": 1}, {"key": "45.56.165.95", "count": 1}, {"key": "45.56.165.96", "count": 1}, {"key": "45.56.165.97", "count": 1}, {"key": "45.56.165.98", "count": 1}, {"key": "45.56.165.99", "count": 1}, {"key": "45.56.165.100", "count": 1}]"`. The 'Output' panel shows a list of IP addresses: 2.58.14.41, 3.76.100.131, 13.59.168.154, 13.215.227.78, 13.215.228.73, 23.152.0.64, 34.207.174.202, 34.219.121.232, 43.139.241.58, 45.45.219.141, 45.56.165.27, 45.56.165.30, 45.56.165.39, 45.56.165.40, 45.56.165.41, 45.56.165.42, 45.56.165.43, 45.56.165.44, 45.56.165.45, 45.56.165.46, 45.56.165.47, 45.56.165.48, 45.56.165.49, 45.56.165.50, 45.56.165.51, 45.56.165.52, 45.56.165.53, 45.56.165.54, 45.56.165.55, 45.56.165.56, 45.56.165.57, 45.56.165.58, 45.56.165.59, 45.56.165.60, 45.56.165.61, 45.56.165.62, 45.56.165.63, 45.56.165.64, 45.56.165.65, 45.56.165.66, 45.56.165.67, 45.56.165.68, 45.56.165.69, 45.56.165.70, 45.56.165.71, 45.56.165.72, 45.56.165.73, 45.56.165.74, 45.56.165.75, 45.56.165.76, 45.56.165.77, 45.56.165.78, 45.56.165.79, 45.56.165.80, 45.56.165.81, 45.56.165.82, 45.56.165.83, 45.56.165.84, 45.56.165.85, 45.56.165.86, 45.56.165.87, 45.56.165.88, 45.56.165.89, 45.56.165.90, 45.56.165.91, 45.56.165.92, 45.56.165.93, 45.56.165.94, 45.56.165.95, 45.56.165.96, 45.56.165.97, 45.56.165.98, 45.56.165.99, 45.56.165.100.

Taking that list and validating servers with VirusTotal, we can see that every result was marked malicious, and all had at least one comment referencing BianLian.

## Final Results Checked Against Virustotal

2[.]58[.]14[.]41 - BianLian 8/88 VT  
3[.]76[.]100[.]131 - BianLian 11/88 VT  
13[.]59[.]168[.]154 - BianLian 11/88 VT  
13[.]215[.]227[.]78 - BianLian 11/88 VT  
13[.]215[.]228[.]73 - BianLian 11/88 VT  
23[.]152[.]0[.]64 - BianLian 11/88 VT  
34[.]207[.]174[.]202 - BianLian 11/88 VT  
34[.]219[.]121[.]232 - BianLian 9/88 VT  
43[.]139[.]241[.]58 - BianLian 13/88 VT  
45[.]45[.]219[.]141 - BianLian 6/88 VT  
45[.]56[.]165[.]27 - BianLian 12/88 VT  
45[.]56[.]165[.]30 - BianLian 11/88 VT  
45[.]80[.]151[.]49 - 4/88 BianLian  
45[.]86[.]163[.]188 - 9/88 BianLian  
45[.]86[.]163[.]224 - 13/88 BianLian  
54[.]193[.]91[.]232 - 11/88 BianLian  
65[.]109[.]3[.]80 - 6/88 BianLian  
66[.]29[.]155[.]44 - 6/88 BianLian  
85[.]13[.]118[.]11 - 6/88 BianLian  
87[.]247[.]185[.]109 - 6/88 BianLian  
91[.]102[.]162[.]229 - 9/88 BianLian  
94[.]131[.]98[.]34 - 14/88 BianLian  
94[.]198[.]50[.]195 - 11/88 BianLian  
103[.]57[.]250[.]152 - 14/88 BianLian  
104[.]36[.]229[.]15 - 13/88 BianLian  
104[.]194[.]11[.]252 - 6/88 BianLian  
104[.]238[.]34[.]130 - 6/88 BianLian  
104[.]238[.]35[.]163 - 11/88 BianLian  
104[.]238[.]60[.]64 - 9/88 BianLian  
104[.]238[.]61[.]150 - 9/88 BianLian  
104[.]243[.]32[.]53 - 5/88 BianLian  
104[.]243[.]33[.]83 - 6/88 BianLian  
104[.]243[.]33[.]84 - 7/88 BianLian  
108[.]174[.]60[.]151 - 8/88 BianLian  
120[.]48[.]110[.]233 - 5/88 BianLian  
139[.]59[.]40[.]48 - 8/88 BianLian  
143[.]198[.]46[.]29 - 9/88 BianLian  
149[.]154[.]158[.]34 - 10/88 BianLian  
149[.]154[.]158[.]199 - 11/88 BianLian  
149[.]248[.]14[.]201 - 6/88 BianLian  
168[.]119[.]88[.]236 - 6/88 BianLian  
173[.]254[.]235[.]30 - 10/88 BianLian  
185[.]240[.]103[.]195 - 9/88 VT BianLian  
185[.]248[.]100[.]118 - 6/88 BianLian  
188[.]34[.]130[.]46 - 4/88 BianLian  
192[.]52[.]166[.]233 - 6/88 VT BianLian  
192[.]236[.]192[.]207 - 6/88 VT BianLian  
194[.]213[.]18[.]45 - 9/88 VT BianLian  
195[.]128[.]235[.]20 - 7/88 VT BianLian  
198[.]199[.]76[.]216 - 6/88 VT BianLian

## Note on Searching Without Regex

---

The regular expression (Regex) feature is only available within the Paid version of Censys.

If you wish to obtain the same results on the free version. You can use the following query to obtain the same results. This works if you know the specific length of each field you want to search.

It won't work in other cases where you want to specify a length range (eg something like `.{14,16}`). But for hardcoded lengths, it can work and is functionally equivalent to `.{16}`

```
services.banner_hashes="sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855" and  
services.tls.certificates.leaf_data.subject_dn:"C=????????????????, O=????????????????, OU=????????????????"
```