THREATX

From ThreatX Labs:

Anatomy of a Targeted Credential Stuffing Attack

Javier Rivera, Senior Security Researcher

Contents

ABOUT THREATX

ThreatX's API protection platform and complete managed services make the world safer by protecting APIs from all threats, including DDoS attempts, complex botnets, zero-day and multi-mode attacks. ThreatX applies artificial intelligence and machine learning to detect and respond to even the slightest indicators of suspicious activity in real-time. Today, ThreatX protects APIs for companies in every industry across the globe. For more information, visit: www.threatx.com

Introduction

One of the main challenges that security operation centers (SOC) and threat hunting teams run into is trying to determine what is noise vs. a targeted attack when looking at millions (or even billions!) of requests in their logs. While you can use automation to detect anomalous or malformed traffic, accounting for all modifications to headers and body content is an ordeal.

Nowadays, there are bots and scanners that look for common endpoints (i.e., files, directories, API paths) on IPv4, and ongoing efforts from nation-state entities and cybersecurity companies are already moving into the IPv6 realm. These scanners can be categorized as follows:

- Active host scanners: Rely only on Layer 3 (ICMP) and Layer 4 (TCP/UDP/QUIC) information to determine whether an IP address is attached to a machine or device.
- Endpoint and directory enumerators: Focused on taking common endpoints, URI paths, and filenames and determining if they exist on targeted IP addresses.
- Vulnerability scanners: The most aggressive scanners targeting hosts looking for common (or not!) vulnerabilities that can be exploited. These can range from injection of malformed SQL (sqlmap) to arbitrary file reads and remote command execution.

Moreover, bots can vary from simple scripts that reuse common tools with default configurations to complex networks of compromised or borrowed IPs tailored to specific purposes (such as DDoS) and/or applications (e.g., targeting only WordPress sites or checking vulnerabilities only applicable to WordPress). This introduces a couple of pain points to those trying to analyze the data:

1. How do we classify and detect common scanners and tools?
2. How do we determine if an attack or payload is targeting the application's technology (e.g., nginx and Apache for web servers; WordPress and Rails for application frameworks)?
3. How do we find out if the attack is tailored to the application or just looking for generic entries?

These are some of the questions we try to provide guidance for by looking at some parts of the request, identifying what information they provide, and figuring out how to use them in a real-world scenario.

Hunting for the Outliers

We'll start by assuming there's at least access to the following HTTP traffic metadata:

1. User Agent: While attackers can easily rotate user agents, most "benign" and/or popular scanners and bots will customize their user agent to a specific string to provide some sort of identification to the hosts being scanned.
2. TLS Fingerprint: TLS fingerprints, as provided by tools such as JA3/JA3S/JARM, can offer insight on what low-level libraries are being used on both client and server sides, supplying a more complete picture of the real actor behind the requests being made. Although rotating these fingerprints is possible, this information can still be used to detect less sophisticated attacks.
3. Targeted URIs: Here is where knowledge of the application or host that the requests are trying to reach comes in handy. For example: a request is trying to access .htpasswd on a nginx server or trying to log in against WordPress's wp-login endpoint, but the targeted application does not use either. This is not normal user behavior, so you could be a bit more suspicious about the IP where that traffic is coming from.
4. Arguments and Body Content: Like URI, knowledge is key to detecting what should be allowed against the application you are trying to protect/investigate. This is one of the main pain points for automated traffic analysis, since the variability of endpoints and payloads that can be accepted by the application must be properly tracked and understood to provide better protection coverage.

   Moreover, the type of arguments or data that can be sent to the application can provide insight on what normal traffic should look like, and when there is an attack trying to bypass current protections by manipulating their values.
5. Source IP: Basic checks for source IP addresses for proxy or VPN capabilities (including Tor) can be used to determine the intent of a request by combining it with other metadata, including geolocation and if the IP has previously generated anomalous traffic. Nowadays, this is becoming more difficult to use as a clear separator due to the ubiquitous existence of hosting and cloud providers (AWS EC2 platform, VPS services like DigitalOcean, etc.). Also, the distribution of these can help us understand the speed and magnitude of the attack or scan.

While this is not an exhaustive list, it provides a baseline to start digging into the traffic and isolating the real attacks from the noise. Let's put these guidelines into use as a real-world attack example that we at ThreatX have seen recently.

Anatomy of a Custom Attack: Distributed Credential Stuffing

One of ThreatX's customers recently experienced a credential stuffing attack that we can use to illustrate a couple of ways you can start investigating anomalous traffic on your network. In this attack, the SOC first noticed suspicious traffic from various IP addresses using the following User-Agent header:

gfdfgsdfgsgfdDSFSD3223

Given that this is clearly not a common user agent, we decided to dig a bit deeper into what the requests were trying to accomplish.

Fingerprinting an attack's pattern and behavior

The next step was to see what other metadata we could extract from these events showing up in our platform. Fortunately, they had unique fingerprints. Traffic observed using this user agent:

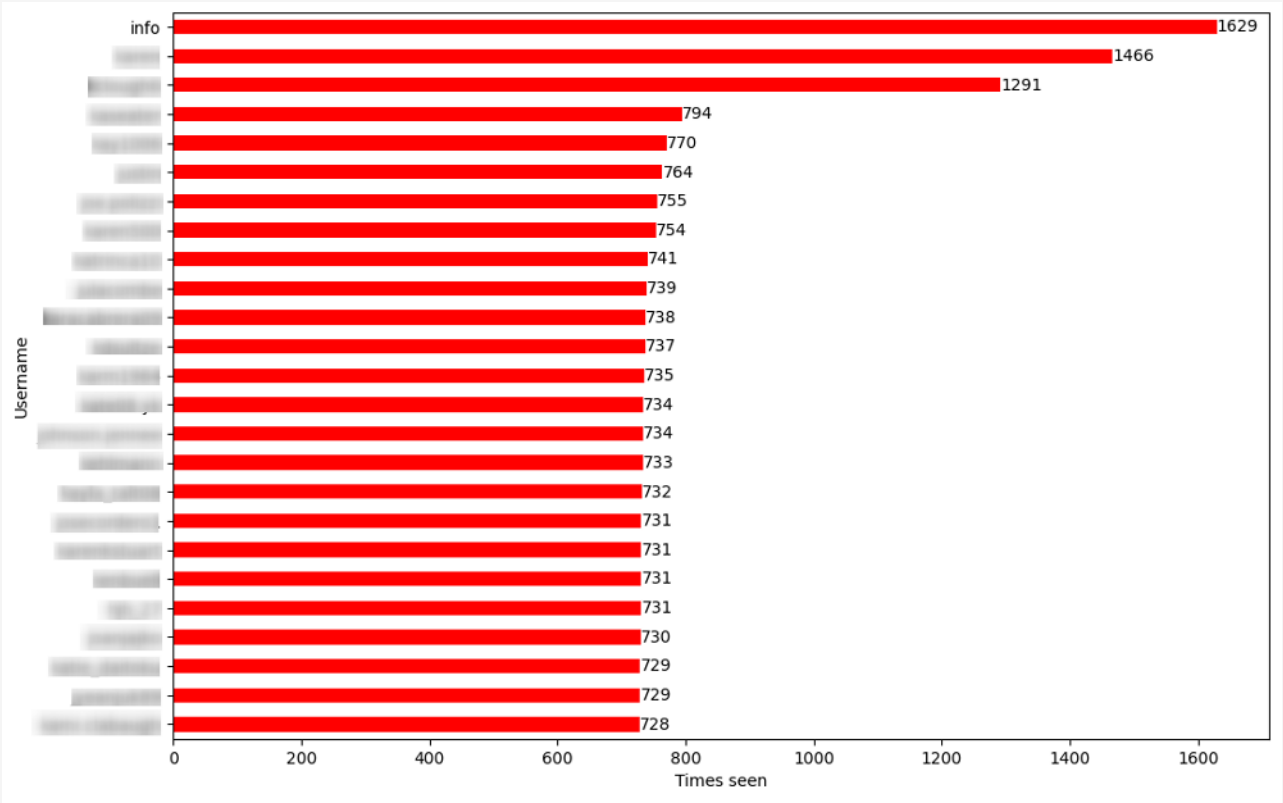
- Always used the POST HTTP method.
- Only targeted a single host/domain.
- Only made requests against the same login endpoint/URI for the application.
- Passed arguments as the POST parameters that were the same:
    - userName
    - password
    - ClientTimeZone

We can now use this information to further track this customized bot (we will get to why it is a bot later) while it is running, and even further across other customers that might show similar patterns (fortunately, that was not the case). Let's look at each argument being passed:

1. userName:

Over 1.6 million unique usernames were seen across all suspicious requests. This points to a credential stuffing attack. Figure 1 highlights the top 25 usernames.

Figure 1: Unique usernames observed in the suspicious botnet/credential stuffing attack



Even further, username reuse is more than 500 times in the top 1K! Note how the top username is info, which is a prevalent email user on many websites (i.e., info@website).

2. password:

Due to the nature of how our platform protects sensitive user data, we could not figure out how many unique passwords were used. But let's walk through the possibilities:

- Credential stuffing - Unique credentials would be observed. Note that duplicates might be observed (i.e., multiple users might show the same password).
- Credential bruteforcing – Every user/password pair would (most likely) have shared passwords, or each user will have multiple passwords tried against them. Some examples:
  - user1:password1, user2:password1, user3:password1
  - user1:password1, user1:password2, user1:password3

Keep in mind that this is just a general way to look at it, but attacks might have a more complex approach.

3. ClientTimeZone:

This was the most interesting argument, and the one that prompted us to start investigating other patterns outside the HTTP request data. Our customer had some rules in place already to block timezones that are not allowed to communicate with the host/application. This was done by checking if the provided value is a valid timezone. However, the bot was inserting IP addresses as a timezone.

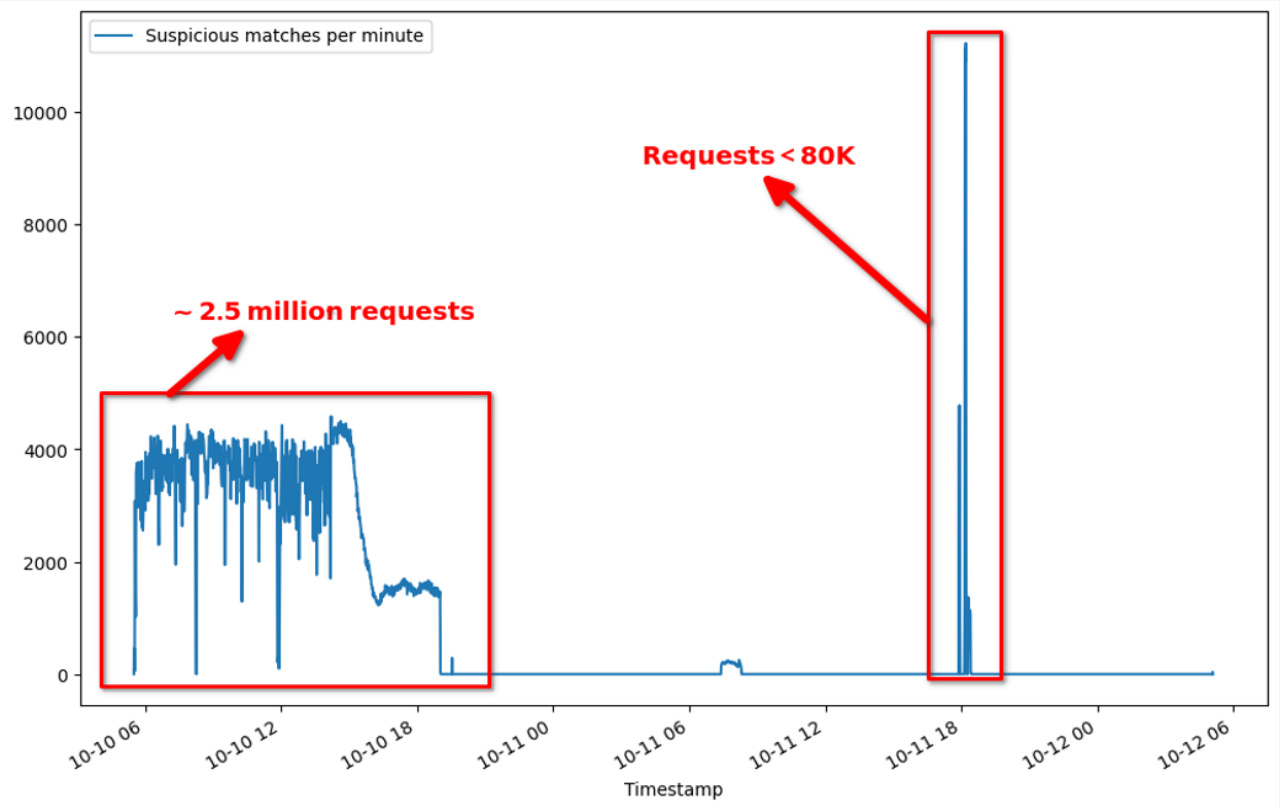
This led us to believe that this was a targeted attack trying to bypass the current protections that were in place. Moreover, this behavior was not seen by any other customer; therefore, we have another way to fingerprint the bot.

Now we had some structure or pattern that we could act on, and we made some rules to ensure that this traffic is always tracked and identified. Once we made sure the new protections were in place and working, it was time to go a bit on the offense and analyze how this bot works.

The Real Hunt: Analyzing the Scanner/Bot Network Traffic

Within hours of the attack starting point, the suspicious user agent showed up in more than 2.5 million requests against the same customer! Figure 2 shows how the bot attacked twice in less than 48 hours, with the second one being shorter due to the protections we developed from the first one.

Figure 2: Rate of attack requests from fingerprinted bot within 48 hours



That's a lot of requests in a relatively short amount of time, and why we started classifying this as a bot, but we needed more data to confirm it. Therefore, we started looking at how many IP addresses the bot was using, which location the requests were coming from, and what providers were being used in the bot's network.

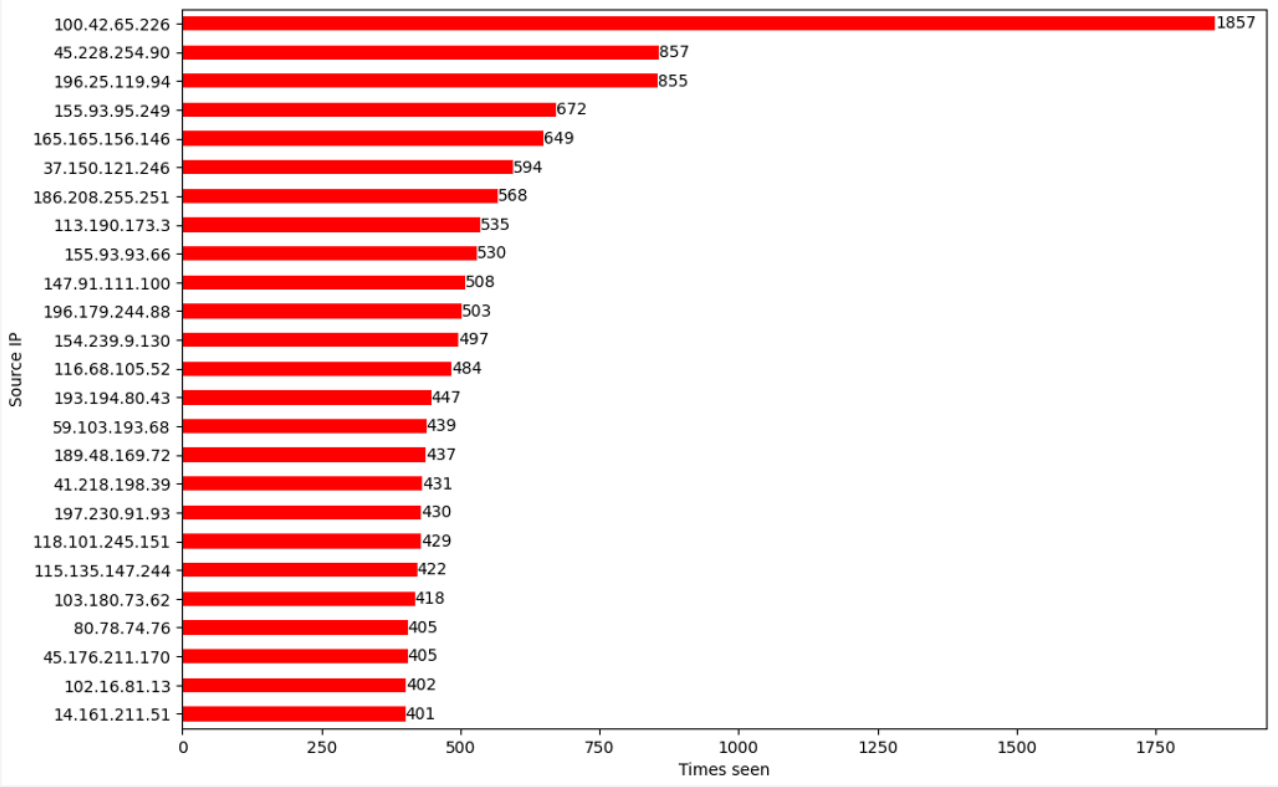
Source IP Analysis

Let's start looking at the amount of source IPs:

- Total unique IP addresses: **86,266**
- Countries: **176**
- ASNs/ASOs: **5021**

There were around 86K IPs used in this attack! Figure 3 highlights the top 25 IPs used.

Figure 3: IPs used in credential stuffing attack



Those numbers gave us more confidence in our classification. Figure 4 displays the top 25 countries associated with the malicious IPs, and Figure 5 highlights the autonomous system organizations these IPs belong to.

Figure 4: Top 25 countries associated with credential stuffing source IPs

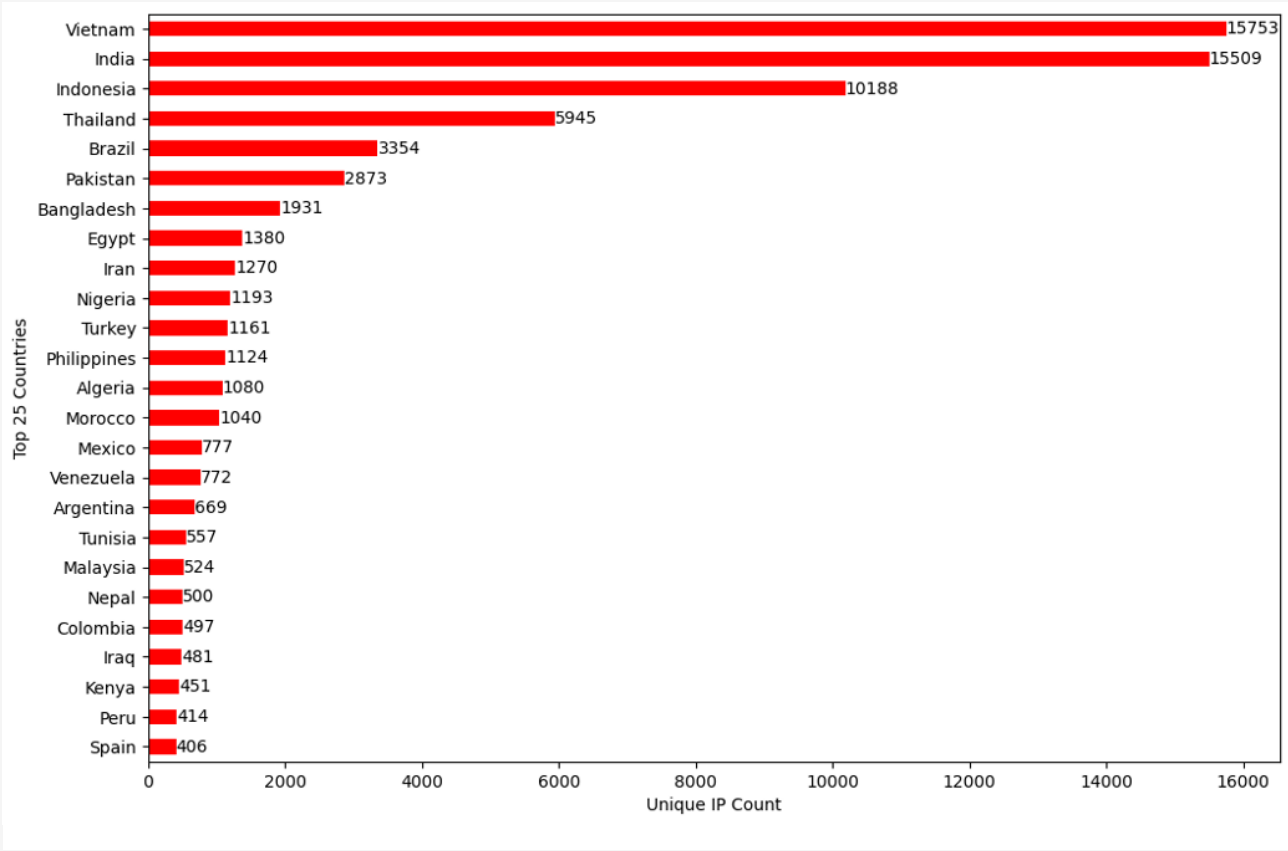
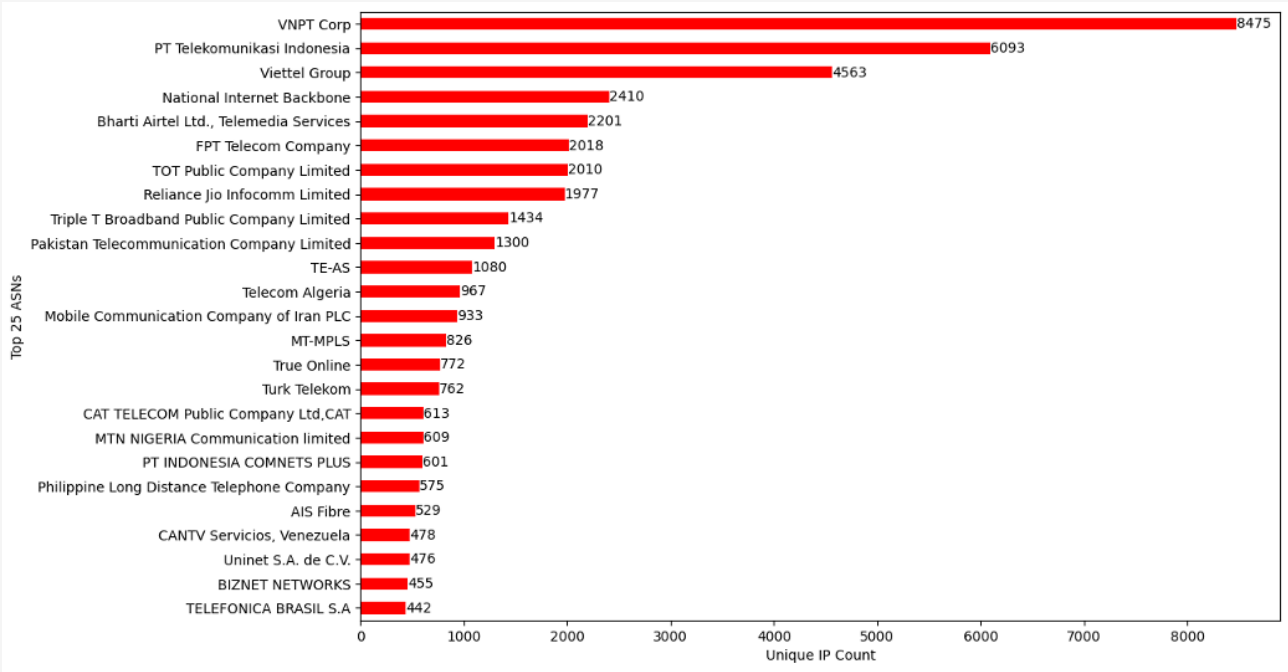
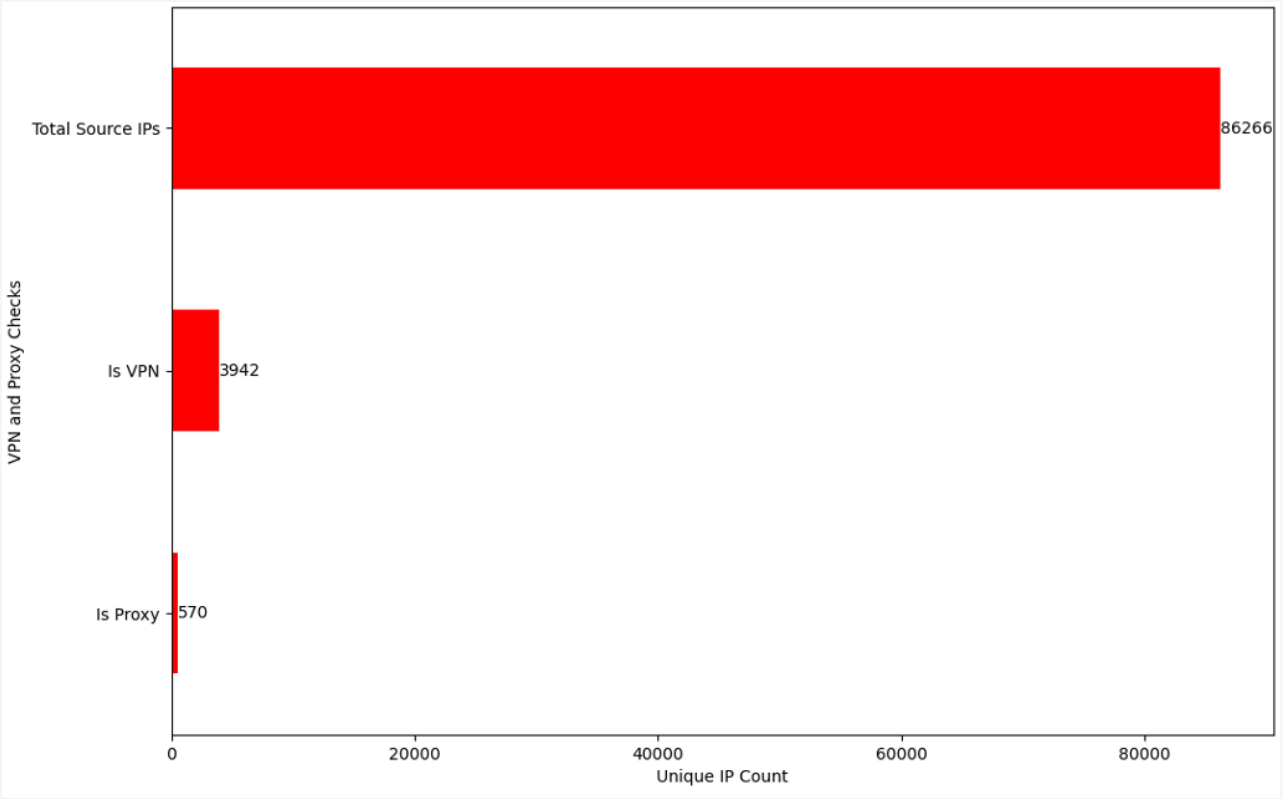


Figure 5: Top 25 autonomous system organizations associated with credential stuffing source IPs



It seemed like the IP addresses were mostly associated with non-US locations and ASOs, so we can add another protection layer based on where the traffic is coming from. We also checked for known proxy and VPN servers that are hosted on the observed sources, and the results are displayed in Figure 6.

Figure 6: Associating source IPs with known VPN and proxy servers



Since more than half of the IP addresses seemed to be flagged as businesses and less than 1K were flagged as compromised servers, we did not attempt to actively scan and extract more information out of them.

Suspicious IPs in ClientTimeZone

What about the IP addresses in the ClientTimeZone argument? Those were a bit more interesting:

- Total unique IP addresses: **1,171**
- Countries: **55**
- ASNs/ASOs: **227**

That IP space is significantly smaller, so we decided to repeat the analysis done on the source IPs to see if there were different patterns. First, we looked at the occurrence of these unique IPs being used as an argument.

We then pulled location and ASN/ASO details, as shown in Figures 7 and 8.

Figure 7: Top 25 countries associated to ClientTimeZone IPs

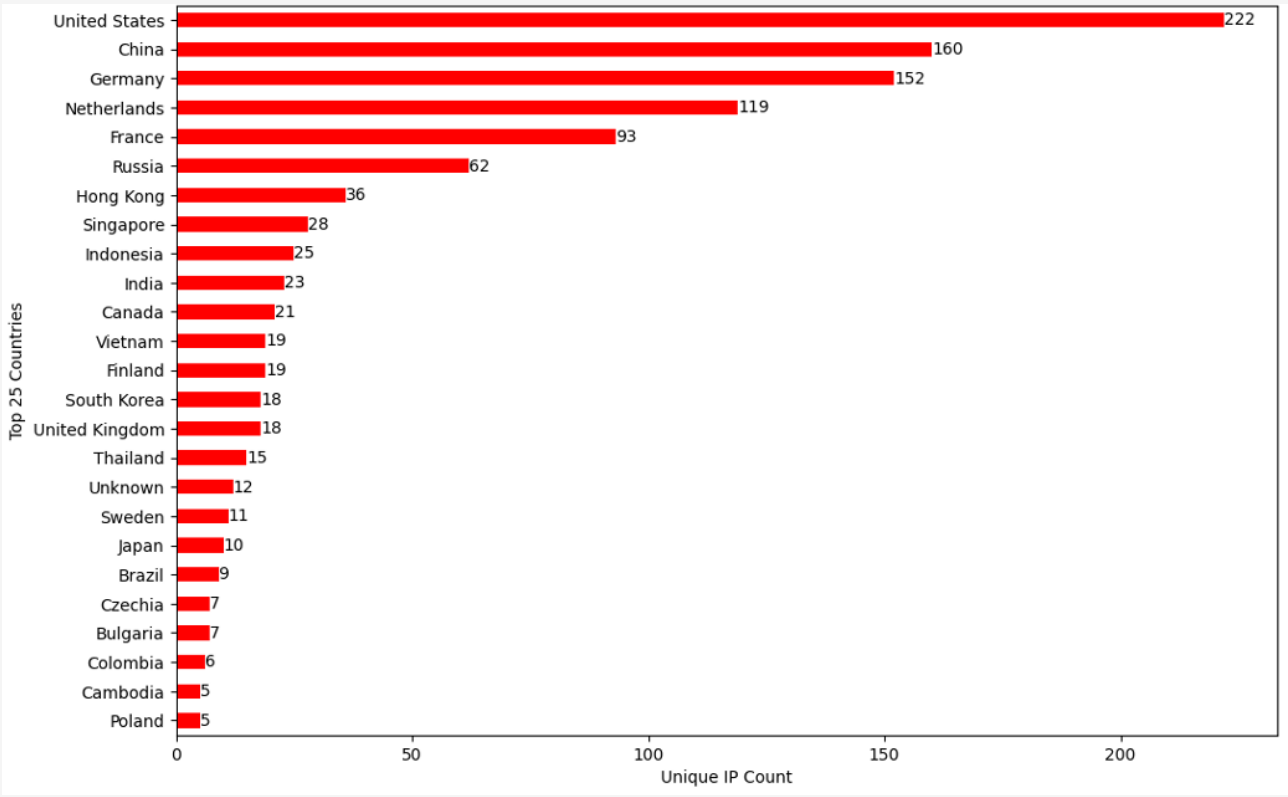
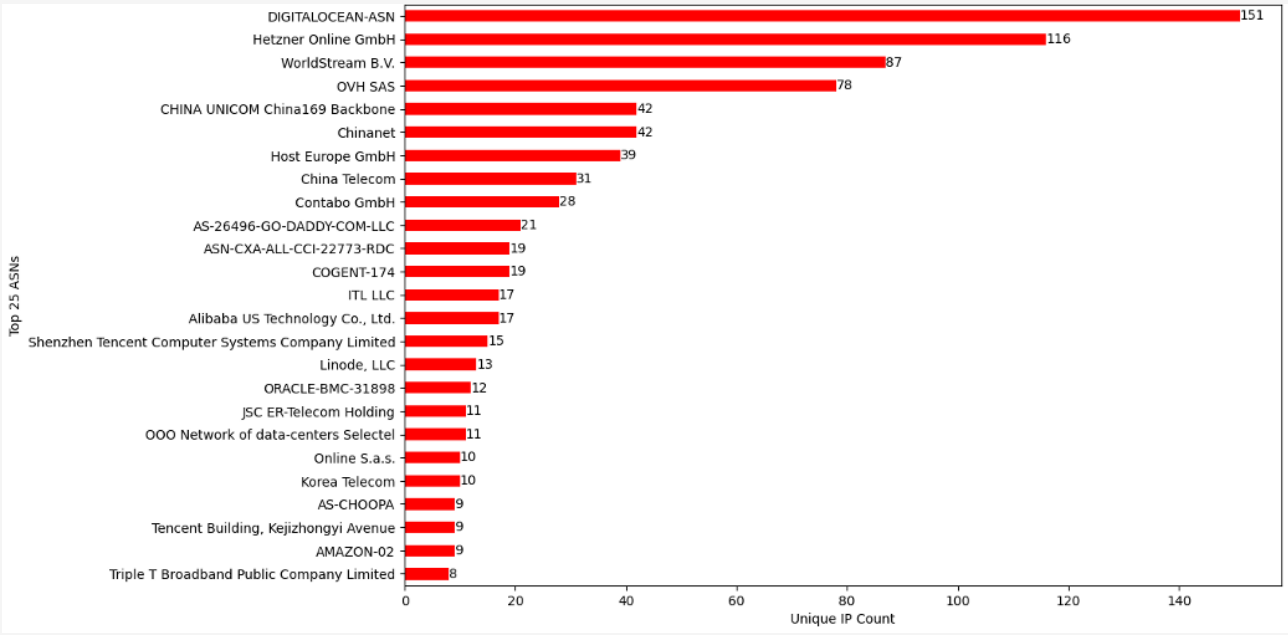
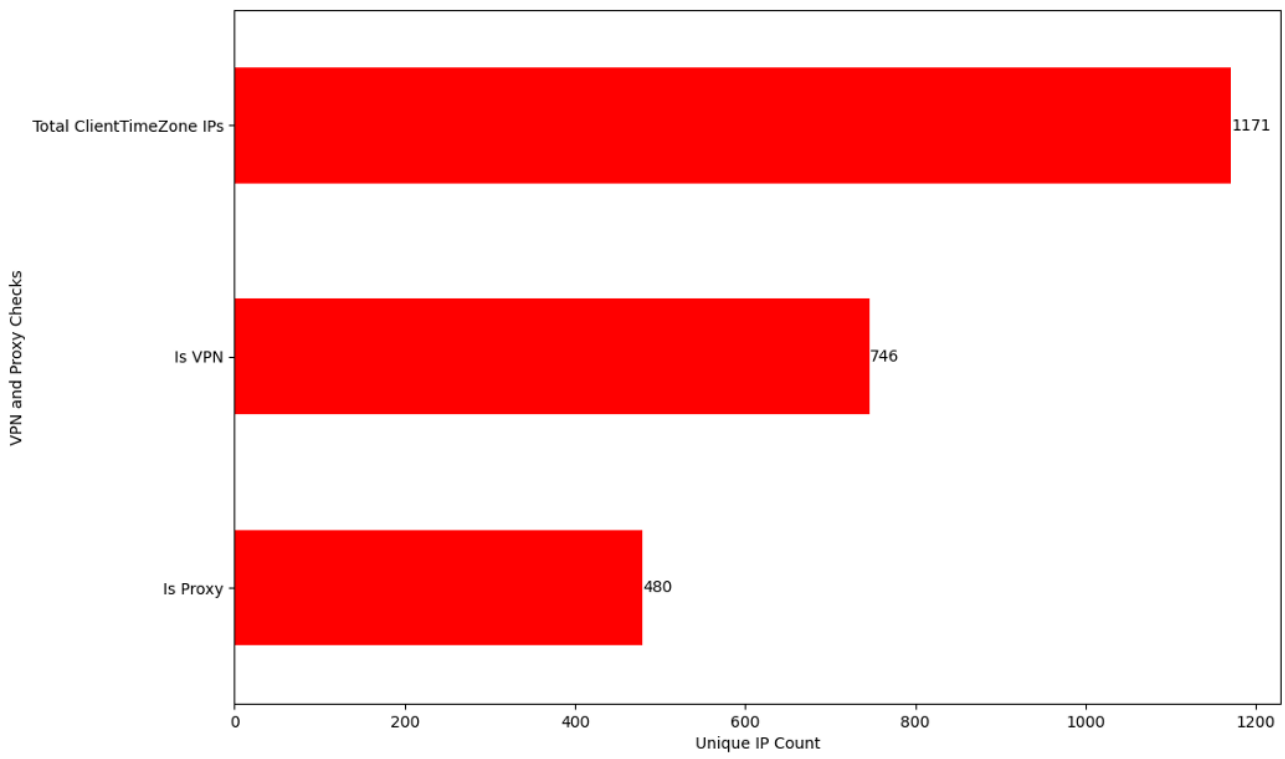


Figure 8: Top 25 autonomous system organizations associated to ClientTimeZone IPs



Finally, we ran the VPN and proxy checks, as seen in Figure 9.

Figure 9: Associating ClientTimeZone IPs with known VPN and proxy servers



This was more interesting to us, since most of them were classified as VPN or proxy servers, and locations that did not show up before now became the most prevalent. We discovered that most of them followed the pattern shown in Figure 10.

Figure 10: Sample port scan of an ClientTimeZone IP

```
Nmap scan report for [redacted].hosted-by-worldstream.net ([redacted])
Host is up (0.14s latency).
Not shown: 992 closed ports
PORT     STATE    SERVICE
22/tcp   filtered ssh
25/tcp   filtered smtp
80/tcp   open     http
777/tcp  open     multiling-http
888/tcp  open     accessbuilder
999/tcp  open     garcon
3001/tcp open     nessus
8000/tcp open     http-alt

Nmap scan report for static.[redacted].clients.your-server.de ([redacted])
Host is up (0.14s latency).
Not shown: 992 closed ports
PORT     STATE    SERVICE
22/tcp   filtered ssh
25/tcp   filtered smtp
80/tcp   open     http
777/tcp  open     multiling-http
888/tcp  open     accessbuilder
999/tcp  open     garcon
3001/tcp open     nessus
8000/tcp open     http-alt
```

Note how TCP ports 80, 777, 888, 999, 3001 and 8000 are open. This was confirmed to be the case for 1,042 out of the 1,171 IPs under investigation, but what could we get out of it? Not much since getting full scan samples for some of them would point to way too many ports open, which confirms some of these proxy and VPN servers' common behavior of obfuscating their users' traffic source infrastructure.

What Is Next?

One could argue that there are more rabbit holes we can go down, like finding out how the IPs from the ClientTimeZone arguments relate back to the source ones or run more complex active scans. On the other hand, the injection of IP addresses might have been a bypass mechanism to timezone checks. With that said, there will be times when either the data available is not enough to make a definitive conclusion about the real intent or actor behind an attack, but with the right tools in place, a reasonable assumption can be made. There are many ways to approach network traffic analysis and threat hunting, but hopefully this gives you a baseline process to start conducting your own investigations (or improve on existing ones).