Analyse der Proxy-Server-Sperre

Können APIs in Proxy-Servern GFW-Sperren vermeiden?

Ich betreibe einen einfachen Server auf meiner Shadowsocks-Instanz mit dem folgenden Code:

```
from flask import Flask, jsonify
from flask_cors import CORS
import subprocess

app = Flask(__name__)
CORS(app)  # CORS für alle Routen aktivieren

@app.route('/bandwidth', methods=['GET'])
def get_bandwidth():
    # Führe den vnstat-Befehl aus, um die 5-Minuten-Intervall-Traffic-Statistiken für eth0 zu erhalten
    result = subprocess.run(['vnstat', '-i', 'eth0', '-5', '--json'], capture_output=True, text=True)
    data = result.stdout

    # Gib die erfassten Daten als JSON-Antwort zurück
    return jsonify(data)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Und ich verwende nginx, um Port 443 zu bedienen, wie unten gezeigt:

```
server {
    listen 443 ssl;
    server_name www.some-domain.xyz;

    ssl_certificate /etc/letsencrypt/live/www.some-domain.xyz/fullchain.pem; # verwaltet von
    # ...
    location / {

        proxy_pass http://127.0.0.1:5000/;
        # ...
```

```
    }
}
```

Dieses Serverprogramm stellt Netzwerkdaten bereit, und ich verwende den Server als meinen Proxy-Server, wodurch ich meinen Online-Status auf meinem Blog mithilfe der Netzwerkdaten anzeigen kann.

Interessanterweise wurde der Server seit mehreren Tagen nicht von der Great Firewall (GFW) oder anderen Netzwerkkontrollsystemen gesperrt. Normalerweise würde der von mir eingerichtete Proxy-Server innerhalb von ein oder zwei Tagen gesperrt werden. Der Server führt ein Shadowsocks-Programm auf einem Port wie 51939 aus, sodass er mit Shadowsocks-Datenverkehr gemischt mit regulärem API-Datenverkehr arbeitet. Diese Mischung scheint die GFW dazu zu verleiten, zu glauben, dass der Server kein dedizierter Proxy ist, sondern ein normaler Server, was verhindert, dass die IP gesperrt wird.

Diese Beobachtung ist faszinierend. Es scheint, dass die GFW spezifische Logik verwendet, um Proxy-Datenverkehr von regulärem Datenverkehr zu unterscheiden. Während viele Websites wie Twitter und YouTube in China blockiert sind, bleiben zahlreiche ausländische Websites – wie die internationaler Universitäten und Unternehmen – zugänglich.

Dies deutet darauf hin, dass die GFW wahrscheinlich auf Regeln basiert, die zwischen normalem HTTP/HTTPS-Datenverkehr und Proxy-bezogenem Datenverkehr unterscheiden. Server, die beide Arten von Datenverkehr handhaben, scheinen Sperren zu vermeiden, während Server, die nur Proxy-Datenverkehr handhaben, eher blockiert werden.

Eine Frage ist, welchen Zeitraum die GFW verwendet, um Daten für eine Sperre zu sammeln – ob es ein Tag oder eine Stunde ist. Während dieses Zeitraums wird festgestellt, ob der Datenverkehr ausschließlich von einem Proxy stammt. Wenn dies der Fall ist, wird die IP des Servers gesperrt.

Ich besuche oft meinen Blog, um zu überprüfen, was ich geschrieben habe, aber in den kommenden Wochen werde ich mich auf andere Aufgaben konzentrieren, anstatt Blogbeiträge zu schreiben. Dies wird meinen Zugriff auf die `bandwidth`-API über Port 443 verringern. Wenn ich feststelle, dass ich erneut gesperrt werde, sollte ich ein Programm schreiben, das regelmäßig auf diese API zugreift, um die GFW auszutricksen.

Hier ist die überarbeitete Version Ihres Textes mit verbesserter Struktur und Klarheit:

Wie die Great Firewall (GFW) funktioniert.

Schritt 1: Anfragen protokollieren

```
import time

# Datenbank zur Speicherung von Anfragedaten
request_log = []

# Funktion zur Protokollierung von Anfragen
def log_request(source_ip, target_ip, target_port, body):
    request_log.append({
        'source_ip': source_ip,
        'target_ip': target_ip,
        'target_port': target_port,
        'body': body,
        'timestamp': time.time()
    })
```

Die Funktion `log_request` zeichnet eingehende Anfragen mit wesentlichen Informationen wie der Quell-IP, Ziel-IP, Ziel-Port, Anfragekörper und Zeitstempel auf.

Schritt 2: Überprüfen und Sperren von IPs

```
# Funktion zur Überprüfung von Anfragen und Sperren von IPs
def check_and_ban_ips():
    banned_ips = set()

    # Iteriere über alle protokollierten Anfragen
    for request in request_log:
        if is_illegal(request):
            banned_ips.add(request['target_ip'])
        else:
            banned_ips.discard(request['target_ip'])

    # Wende Sperren auf alle identifizierten IPs an
    ban_ips(banned_ips)
```

Die Funktion `check_and_ban_ips` iteriert durch alle protokollierten Anfragen, identifiziert und sperrt IPs, die mit illegalen Aktivitäten in Verbindung stehen.

Schritt 3: Definition, was eine Anfrage illegal macht

```
# Funktion zur Simulation der Überprüfung, ob eine Anfrage illegal ist
def is_illegal(request):
    # Platzhalter für die tatsächliche Logik zur Überprüfung illegaler Anfragen
    # Zum Beispiel Überprüfung des Anfragekörpers oder Ziels
    return "illegal" in request['body']
```

Hier überprüft `is_illegal`, ob der Anfragekörper das Wort “illegal” enthält. Dies kann auf anspruchsvollere Logik erweitert werden, je nachdem, was illegale Aktivitäten ausmacht.

Schritt 4: Sperren identifizierter IPs

```
# Funktion zum Sperren einer Liste von IPs
def ban_ips(ip_set):
    for ip in ip_set:
        print(f"Sperre IP: {ip}")
```

Sobald illegale IPs identifiziert sind, sperrt die Funktion `ban_ips` sie, indem sie ihre IP-Adressen ausgibt (oder in einem echten System blockiert).

Schritt 5: Alternative Methode zur Überprüfung und Sperrung von IPs basierend auf 80% illegalen Anfragen

```
# Funktion zur Überprüfung von Anfragen und Sperrung von IPs basierend auf 80% illegalen Anfragen
def check_and_ban_ips():
    banned_ips = set()
    illegal_count = 0
    total_requests = 0

    # Iteriere über alle protokollierten Anfragen
    for request in request_log:
        total_requests += 1
```

```
        if is_illegal(request):
            illegal_count += 1

    # Wenn 80% oder mehr der Anfragen illegal sind, sperre diese IPs
    if total_requests > 0 and (illegal_count / total_requests) >= 0.8:
        for request in request_log:
            if is_illegal(request):
                banned_ips.add(request['target_ip'])

    # Wende Sperren auf alle identifizierten IPs an
    ban_ips(banned_ips)
```

Diese alternative Methode bewertet, ob eine IP basierend auf dem Prozentsatz illegaler Anfragen gesperrt werden sollte. Wenn 80% oder mehr der Anfragen von einer IP illegal sind, wird sie gesperrt.

Schritt 6: Erweiterte Überprüfung illegaler Anfragen (z.B. Shadowsocks und Trojan-Protokoll-Erkennung)

```
def is_illegal(request):
    # Überprüfe, ob die Anfrage das Shadowsocks-Protokoll verwendet (Anfragekörper enthält binäre Daten
    if request['target_port'] == 443:
        if is_trojan(request):
            return True
    elif is_shadowsocks(request):
        return True
    return False
```

Die Funktion `is_illegal` überprüft nun auch spezifische Protokolle wie Shadowsocks und Trojan:
- **Shadowsocks**: Wir könnten auf verschlüsselte oder binäre Daten im Anfragekörper prüfen.
- **Trojan**: Wenn die Anfrage über Port 443 (HTTPS) kommt und bestimmte Muster (z.B. Trojan-Datenverkehrmerkmale) aufweist, wird sie als illegal markiert.

Schritt 7: Beispiel für legale Anfragen

Zum Beispiel sind Anfragen wie `GET https://some-domain.xyz/bandwidth` sicherlich legal und lösen den Sperrmechanismus nicht aus.

Schritt 8: Datenverkehrsmerkmale von Proxy-Servern

Proxy-Server haben sehr unterschiedliche Datenverkehrsmerkmale im Vergleich zu regulären Web- oder API-Servern. Die GFW muss zwischen normalem Web-Server-Datenverkehr und Proxy-Server-Datenverkehr unterscheiden, der völlig anders aussehen kann.

Schritt 9: Maschinelles Lernen und KI-Modelle für intelligente Erkennung

Angesichts der breiten Palette von Anfragen und Antworten, die durch das Internet gehen, könnte die GFW KI- und Machine-Learning-Modelle einsetzen, um Datenverkehrsmuster zu analysieren und illegales Verhalten intelligent zu erkennen. Durch das Trainieren des Systems auf eine Vielzahl von Datenverkehrstypen und den Einsatz fortschrittlicher Techniken könnte es effektiver Datenverkehr basierend auf beobachteten Mustern sperren oder filtern.

Update

Trotz meiner Bemühungen wird mein Proxy-Server weiterhin gesperrt. Um dies zu mildern, habe ich eine Problemumgehung mit der umgekehrten IP-Funktion von Digital Ocean implementiert, die es mir ermöglicht, schnell eine neue IP-Adresse zuzuweisen, sobald eine Sperre auftritt.