# MegaMedusa, RipperSec's Public Web DDoS Attack Tool

radware.com/blog/security/2024/08/megamedusa-rippersec-public-web-ddos-attack-tool/

Pascal Geenens

### Key insights

- RipperSec is a pro-Palestinian, pro-Muslim hacktivist group operating from Malaysia
- RipperSec has been operating on Telegram since June 2023 and accumulated over 2,000 members in a little over a year
- MegaMedusa is a publicly available Web DDoS attack tool created and maintained by a member of the RipperSec group
- MegaMedusa can be installed in just five simple commands allowing anyone to launch highly scalable Web DDoS attacks against targets of their choice
- The MegaMedusa attack tool uses 10 randomization techniques to diversify its attack requests and make the detection and mitigation of its attacks harder
- MegaMedusa makes some rudimentary attempts to evade CAPTCHA triggers through randomization and proxy use, but it does not include advanced CAPTCHA-solving capabilities
- RipperSec's threat and scale do not come from a large and sophisticated attack infrastructure but from its community. Community has always been activists' and hacktivists' most powerful weapon.

## RipperSec Background

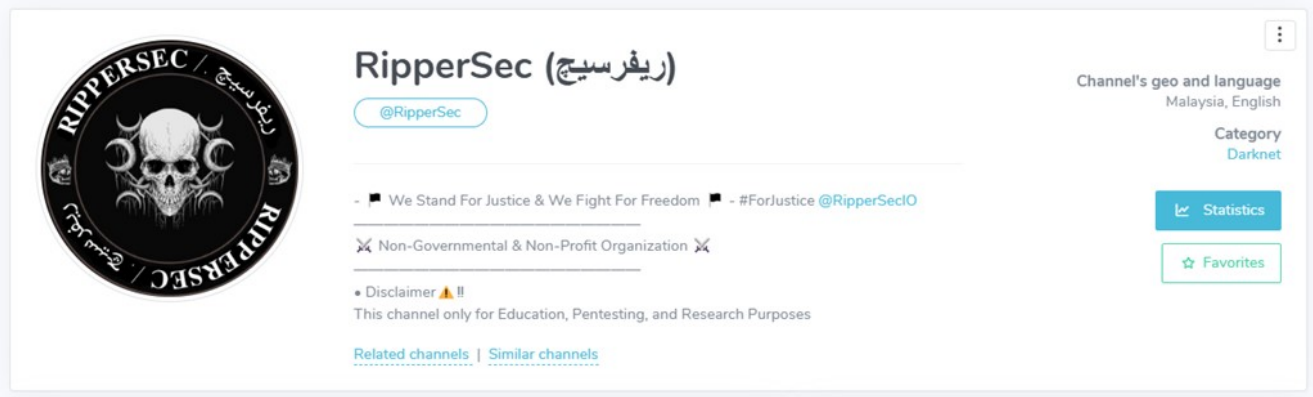


**Figure 1: RipperSec Telegram profile (source: TGStat)**

RipperSec is a pro-Palestinian and pro-Muslim Malaysian hacktivist group. Their Telegram channel @RipperSec was created in June 2023 and accumulated over 2,000 subscribers by August 8, 2024.

RipperSec often works in alliance with other like-minded hacktivist groups and hackers in and outside of the region, including Tengkorak Cyber Crew, Eagle Cyber Crew, Stucx Team, 4Exploitation, Khalifah Cyber Crew, Helang Merah Group, Rex AnonSaven7, Team Cyber Ababil, Malaysia Hacktivist, Zenimous Crew, Laskar Pembebasan Palestina aka the Palestine Liberation Army, Garruda From Cyber (GFC), Holy League, Morrocan Cyber Black Army and several others. Most hacktivists from Malaysia do not agree with the actions taken by Israel and consider all countries that support Israel as enemies.

RipperSec's attack activity includes data breaches, defacements and DDoS attacks—anything that creates chaos, attracts attention and causes disruption that's typical for a hacktivist group.

## RipperSec DDoS Attack Claims

Between January 1 and August 8, 2024, RipperSec claimed 196 DDoS attacks. Almost a third of the attacks targeted Israel. India, the United States, the United Kingdom and Thailand were other countries with significant attack activity claimed by RipperSec in 2024.

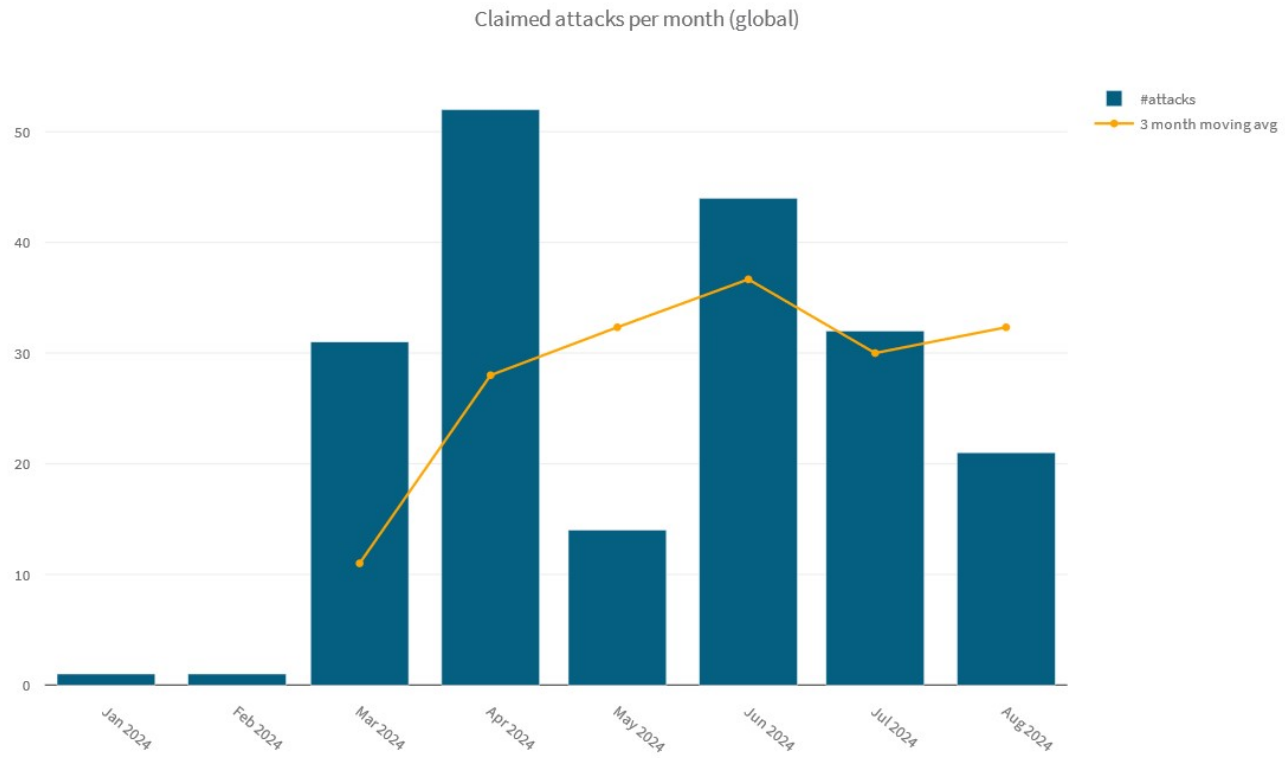


**Figure 2: Number of attacks claimed per month by RipperSec (source: Radware)**

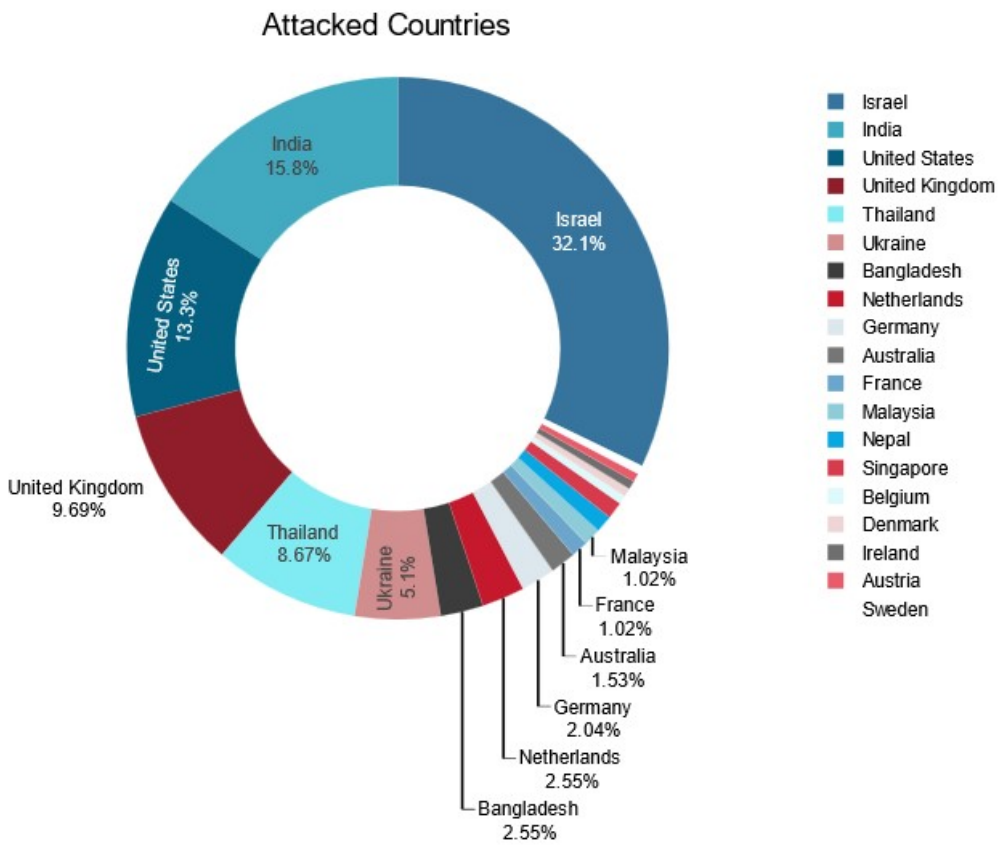


**Figure 3: RipperSec's top targeted countries (source: Radware)**

RipperSec targeted mostly government and educational websites, followed by business, society and financial services.

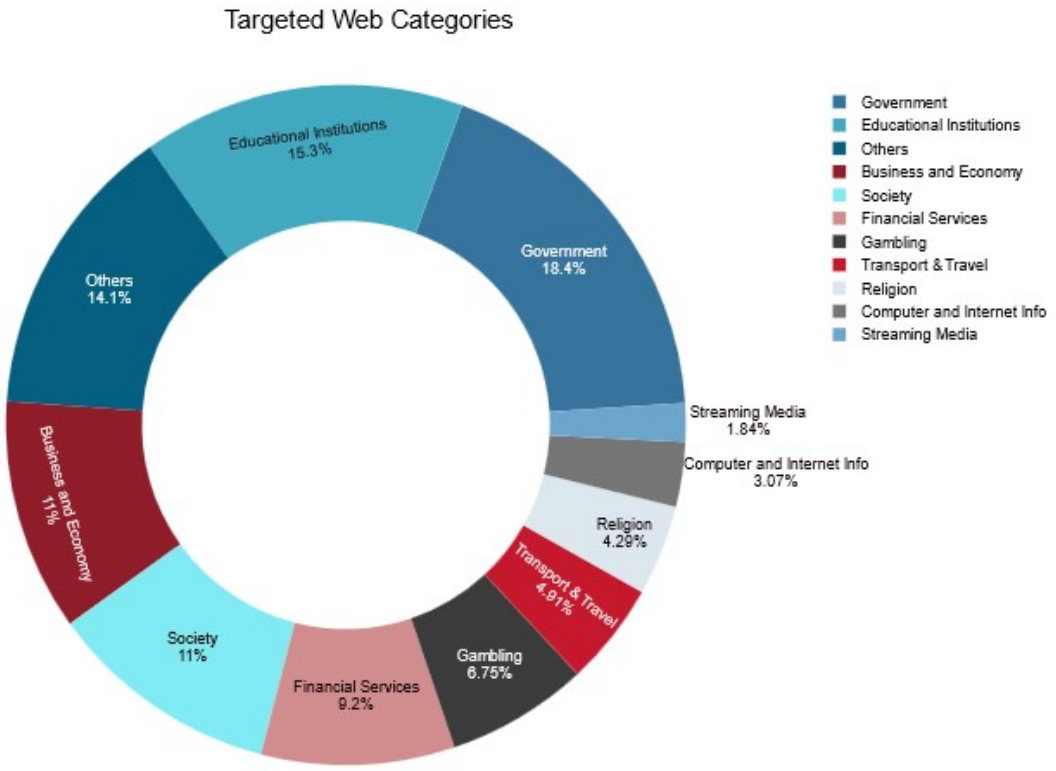


**Figure 4: RipperSec's top targeted website categories (source: Radware)**

## MegaMedusa, a Web DDoS Attack Tool

MegaMedusa is a publicly available Web DDoS attack tool created and maintained by a member of the RipperSec group. The tool's source code is published on GitHub and while its JavaScript code is obfuscated, it is easy enough to deobfuscate and recover readable code. While written in JavaScript, MegaMedusa is a command-line tool to be executed using the Node.js cross-platform JavaScript runtime environment. Node.js provides asynchronous and non-blocking I/O, allowing it to handle multiple requests concurrently. This makes it highly efficient for I/O-bound tasks such as managing large amounts of network connections. Node.js applications also can run on multiple platforms (Windows, macOS, Linux) without the need for platform-specific code.

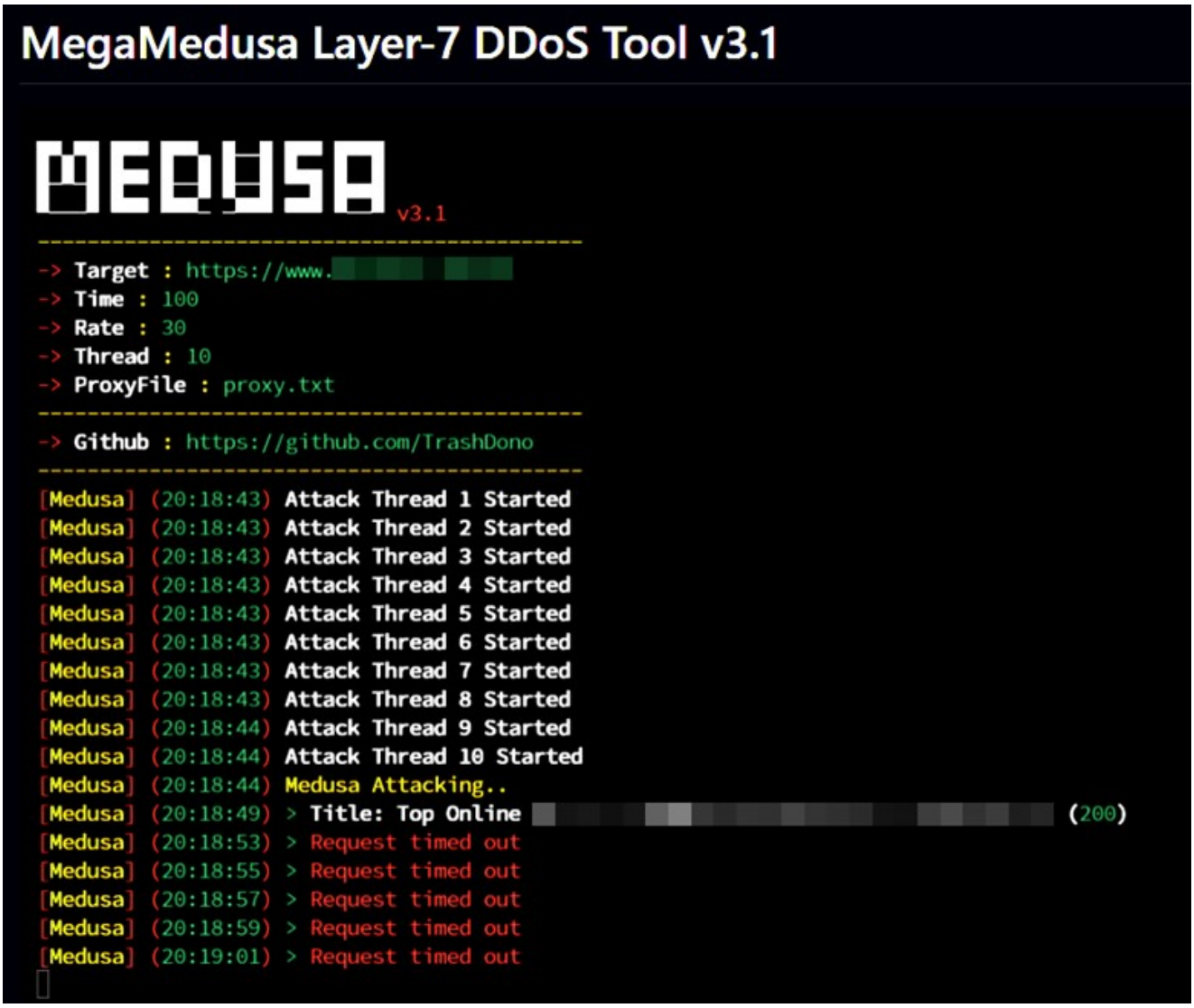


**Figure 7: MegaMedusa Layer 7 DDoS attack tool (source: GitHub)**

## Installation and Operation

The author of MegaMedusa provides an installation script and steps to install and run the tool on the code repository's GitHub landing page. Only five simple commands are needed to download and install the Node.js runtime environment and all required dependencies. Anyone who runs a Linux-based system at home or rents a Linux-based virtual private system in a public or bulletproof cloud can gear up to launch highly scalable Web DDoS attacks against targets of their choice in only a few minutes.

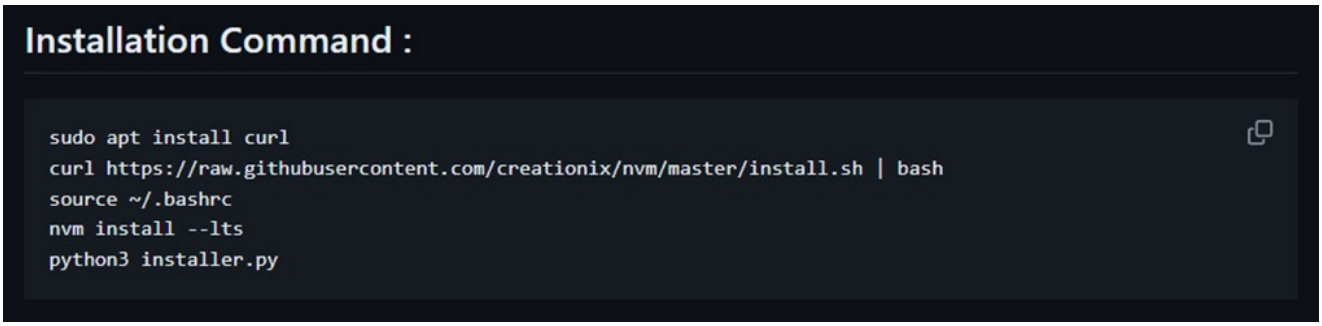


**Figure 8: MegaMedusa installation instructions (source: GitHub)**

The publicly available version of MegaMedusa allows attacks directed at any online web application or API. On the command line, users can specify the number of simultaneous threats that should be executing web requests, the attack rate expressed in requests per second (RPS) and the duration of the attack in seconds. Specifying a text file with open proxies allows the attack traffic to be distributed across proxies. For each new request, the proxy is randomly chosen from the list of proxies provided in the command line.

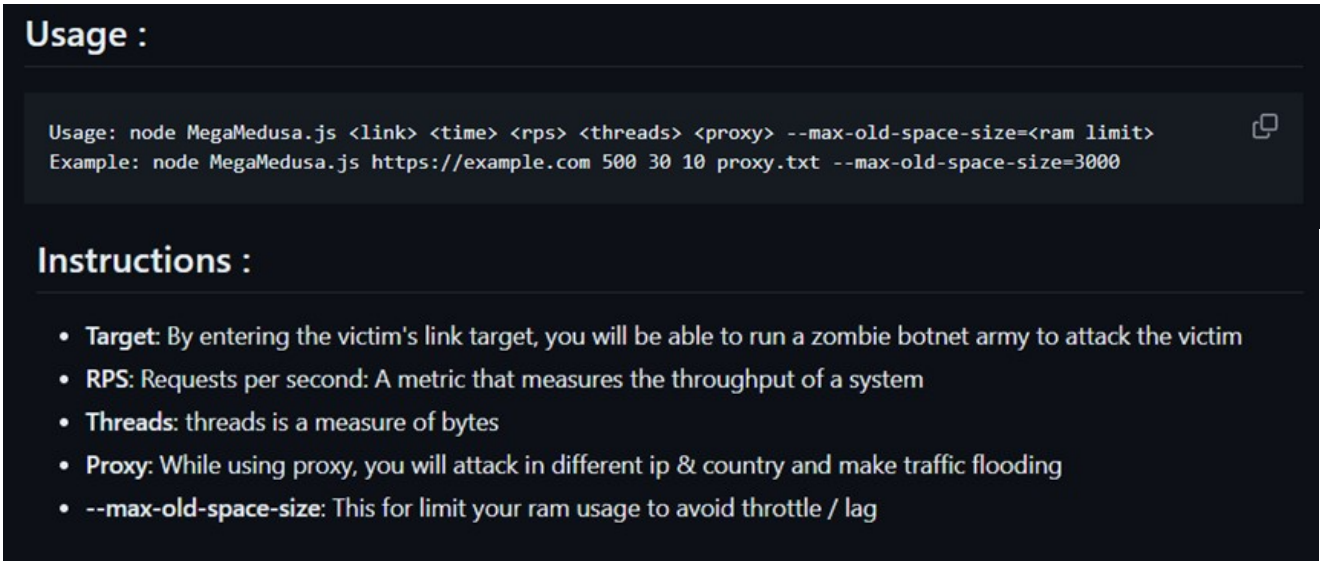


**Figure 9: MegaMedusa command line usage and description of command line arguments (source: GitHub)**

## Randomization of Web Requests

The MegaMedusa attack tool leverages several levels of randomization to diversify its attack requests and make detection and mitigation of its attacks harder. Here are the key randomization techniques used by MegaMedusa:

1. **Randomized Headers:**
    - User-Agent Strings: The script randomizes the User-Agent header, which represents the client's browser and operating system. Different user agents make the requests appear to come from various types of devices and browsers.
    - Accept-Language, Accept-Encoding, Cache-Control, etc.: These headers are randomized to simulate requests from different regions, clients and configurations. For example, the Accept-Language header is randomized to suggest different languages.
    - Referrer Header: The Referrer header is randomized to make it appear as if the request is coming from various legitimate pages like Google Search.
    - Connection Header: The script randomly alternates between keep-alive and close in the connection header.
2. **Randomized Request Paths:**
    - Query Parameters: The script appends random query strings and parameters to the URL path to create unique URLs. For example, it might add '?s=', '?page=', or other parameters with random values to the URL.
    - Path Segments: It also includes random path segments such as '/' and '/.lsrecap/recaptcha?' to diversify the request paths or target specific functionalities of the website like reCAPTCHA to increase the impact on the backend infrastructure of the website.
3. **Randomized Request Methods:** The HTTP request method (GET, POST, HEAD, etc.) is randomized. Each request might use a different method, making it more difficult to identify patterns.
4. **Randomized Cookies:** Randomly generated cookies make each request appear as though it comes from a different session or user. Some cookies such as 'cf_clearance' are randomly generated within specific patterns to attempt to bypass security features.
5. **Random IP Addresses (IP Spoofing):** The X-Forwarded-For, Client-IP, Real-IP, X-Forwarded-Host, and other headers are filled with random IP addresses used to spoof the origin of the request.
6. **Randomized TLS/SSL Configurations:** The script randomizes the use of different TLS/SSL ciphers and protocols when establishing a connection. This makes the TLS handshake appear unique.
7. **Randomized HTTP/2 Settings:** HTTP/2 settings such as headerTableSize, maxConcurrentStreams, initialWindowSize, etc., are randomized to vary the HTTP/2 session characteristics.
8. **Proxy Randomization:** The tool selects a random proxy from a list of proxies for each request, which diversifies the apparent source IP addresses and geolocations of the requests.
9. **Random Timing Intervals:** The attack tool uses setInterval() to send requests at intervals, which can be configured and randomized to vary, making the timing of requests less predictable.

10. **Randomized Header Values:** Random values are inserted into certain headers, such as Sec-WebSocket-Key, Sec-WebSocket-Version, etc., making each request appear unique even at the WebSocket level.

These techniques combined make the requests appear unique and diversified, which helps avoid detection or blocking by standard web application firewalls (WAFs) and other common security measures designed to mitigate suspicious or malicious traffic patterns.

## Open Proxy Support

While MegaMedusa provides support for open proxies and includes a tool to scrape fresh lists of open proxies from publicly available open proxy lists, commercial proxies and private proxies that require authentication are not supported. Adding support for authentication, however, is not a big task and the code shows that the author is not an inexperienced Node.js developer. There is also evidence that the core members of RipperSec use several more evolved and improved versions of MegaMedusa. It is not improbable that certain attacks leverage commercial proxy lists that allow diversifying attack traffic across several hundreds of thousands of IP addresses compared to several thousands of open proxies than can be freely scraped.

The Node.js scrapers provided in the repository are very basic, but they do provide a list of over 30 different resources on the internet that can be used to create custom proxy files. Moreover, searching the internet for open proxy lists is easy enough. Most open proxy websites provide a download function to save a filtered list of proxies in text format that can be directly used with the MegaMedusa tool.

## Security Challenge Bypasses

The author of MegaMedusa claims challenge bypass support for several security vendors including Cloudflare’s Under Attack Mode and NoSec, DDoS Guard, vShield and ShieldSquare Captcha (Radware).

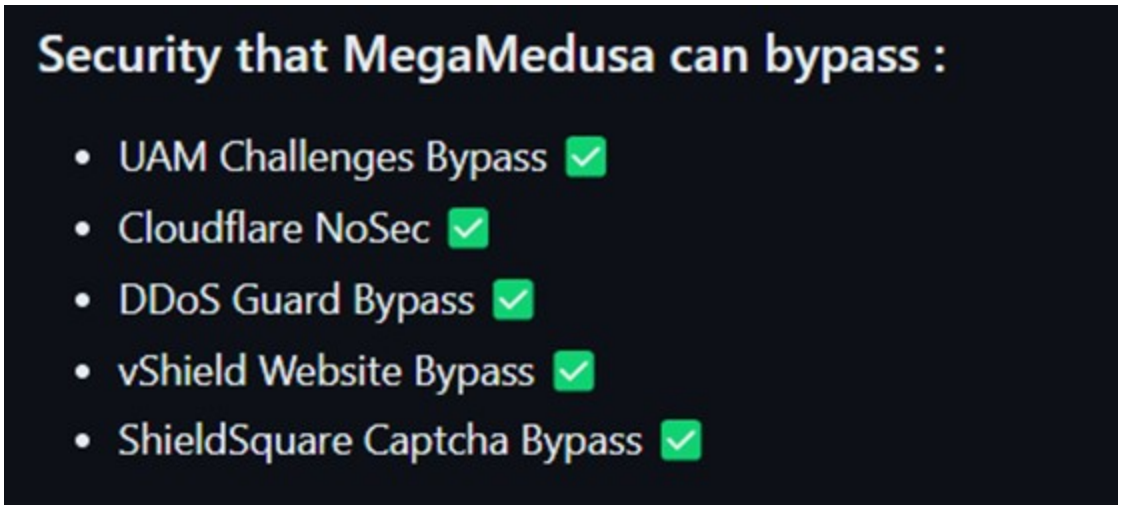


**Figure 10: MegaMedusa challenge bypass features (source: GitHub)**

While MegaMedusa makes some rudimentary attempts to evade CAPTCHA triggers through randomization and proxy use, it does not include advanced CAPTCHA-solving capabilities. It mainly focuses on making the requests appear diverse and less detectable rather than directly bypassing CAPTCHA challenges. True CAPTCHA bypass typically requires more sophisticated approaches, such as solving the CAPTCHA or using pre-solved tokens, neither of which is present in this script.

The implementation of security challenge bypasses in MegaMedusa will not cut through current modern security challenges. The internet is riddled with information, misinformation and all levels of working and broking examples to perform bot detection bypass based on HTTP request headers. It is not impossible to bypass security challenges and fingerprinting detections but relying on static bypass and using randomly generated and pre-provisioned HTTP request headers as implemented in MegaMedusa and shown in the code below has a very limited success rate.

The most efficient way to circumvent protections and challenges is to find the server's origin IP address. This, however, is only possible in scenarios where the origin server is exposed on the internet and does not adequately filter traffic other than from known and authorized secure gateways.

```
headers[":method"] = RipperSec;
headers[':authority'] = parsedTarget.host;
headers[":path"] = parsedTarget.path + paths[Math.floor(Math.randdom() * paths.length)] + '&' + randstr(0xa) + queryString + randstr(0xa);
headers[":scheme"] = "https";
headers["x-forwarded-proto"] = "https";
headers["cache-control"] = control;
headers['X-Forwarded-For'] = spoofed;
headers['sec-ch-ua'] = versi;
headers["sec-ch-ua-mobile"] = sechuas[Math.floor(Math.random() * sechuas.length)];
headers["sec-ch-ua-platform"] = browsers[Math.floor(Math.random() * browsers.length)] + platform1;
headers["accept-language"] = lang;
headers["accept-encoding"] = encoding;
headers["upgrade-insecure-requests"] = Math.random() > 0.5;
headers.Connection = Math.random() > 0.5 ? 'keep-alive' : "close";
headers.accept = accept;
headers["sec-fetch-mode"] = 'navigate';
headers["sec-fetch-dest"] = dest1;
headers["sec-fetch-user"] = '?1';
headers.TE = "trailers";
headers.cookie = 'cf_clearance=' + randstr(0x20) + '.' + randstr(0xa) + '-' + randstr(0xa) + "-1.0.1.1-" + randstr(0xb) + "_vs_V." + randstr(0x1
headers.cookie = "?__cf_chl_tk=" + randayat(0x2b) + '-' + randnombor(0xa) + "-0.0.1.1" + randnombor(0x4);
headers.cookie = "cf_clearance=jJON9OFgfi5                                                                                    bzDi7Nt09
headers["sec-fetch-site"] = site1;
headers["x-requested-with"] = "XMLHttpRequest";
headers['alt-svc'] = randomHeaders["alt-svc"];
headers.Via = spoofed;
headers.sss = spoofed;
headers["Sec-Websocket-Key"] = spoofed;
headers["Sec-Websocket-Version"] = 0xd;
headers.Upgrade = websocket;
headers["X-Forwarded-For"] = spoofed;
headers["X-Forwarded-Host"] = spoofed;
headers["Client-IP"] = spoofed;
headers["Real-IP"] = spoofed;
headers.Referer = randomReferer;
headers.Referer = Ref;
```

**Figure 11: HTTP header parameters, including captcha bypasses (source: MegaMedusa.js)**

MegaMedusa does not include any functionality to automatically solve CAPTCHAs such as image recognition and reCAPTCHA v2/v3 tokens. Sophisticated CAPTCHA bypassing typically involves using machine learning models, CAPTCHA-solving services, or leveraging browser automation tools like Puppeteer or Selenium to interact with CAPTCHAs in real-time, which MegaMedusa does not do.

The code does include some basic CAPTCHA handling elements that interact with CAPTCHA challenges, such as using random __cf_chl_tk tokens, which are related to Cloudflare's CAPTCHA challenges. However, these tokens are randomly generated and not tied to an actual CAPTCHA-solving process. Without solving the CAPTCHA correctly, these tokens are unlikely to be valid.

## RipperSec and MegaMedusa

While a member of the RipperSec group made a version of MegaMedusa publicly available for volunteers and subscribers of the RipperSec Telegram channel, there is evidence that shows the use of improved and more capable custom versions of the tool used by RipperSec group members. Below is a screenshot taken from an advertisement video shared by RipperSec. While not the best quality, it is possible to see several alleged Node.js attack programs in the directory listing. It is reasonable to assume that core RipperSec members have access to more capable tools than the one that is shared publicly and used by volunteers and allies of the group.

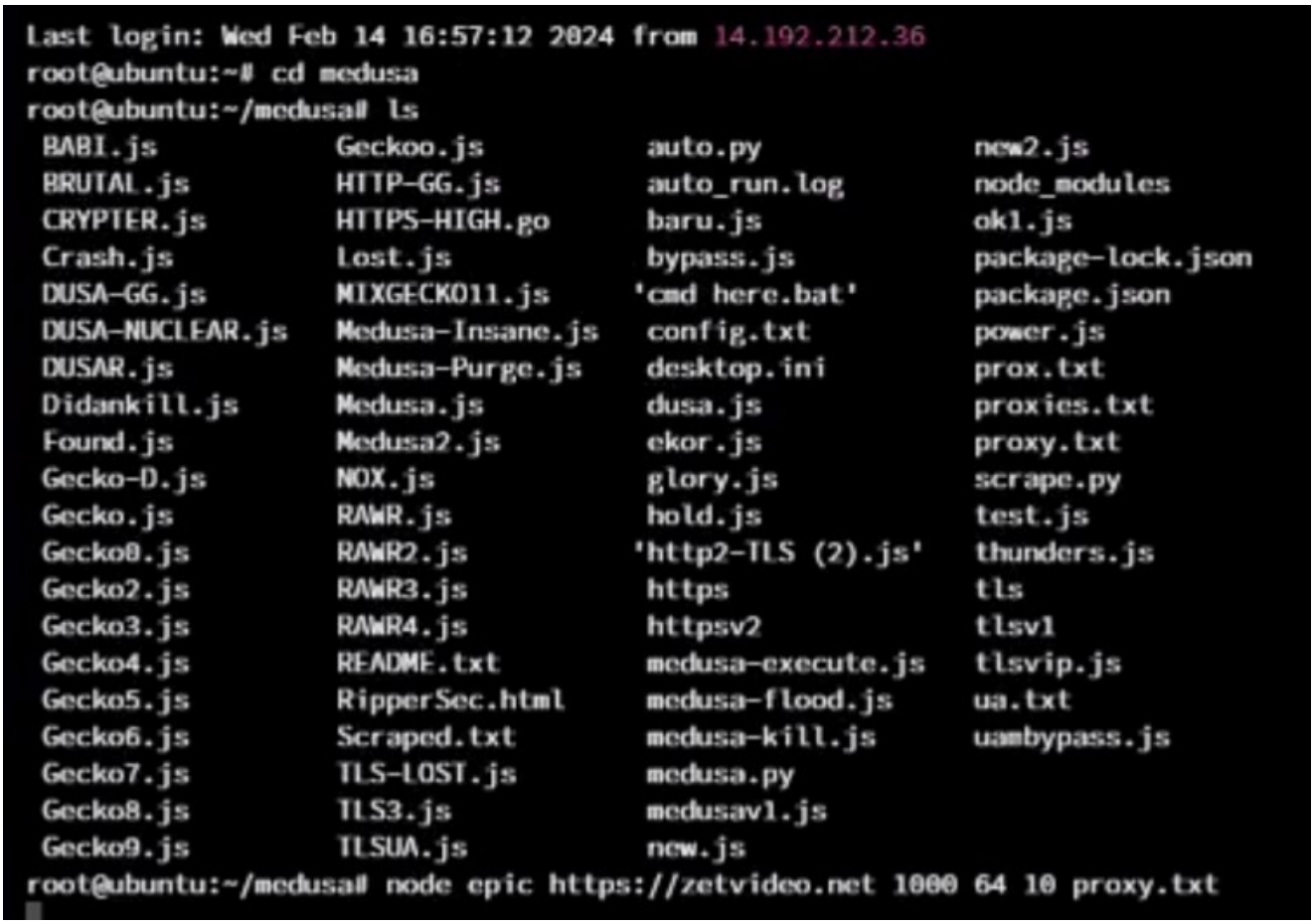


**Figure 12: Screenshot demonstrating an attack and more advanced attack tools (source: RipperSec)**

## Generating Web DDoS Attacks (for educational purposes)

The name Web DDoS attack refers to a high rate of web requests directed at an online web application or API, leveraging different methods of request randomization and proxies. Proxies conceal the origins of web requests and make detection and mitigation more difficult by making it look like the requests are coming from many different locations. Proxying requests can also circumvent measures like geo-blocking by leveraging proxies inside the country of the victim.

### Proxying Requests

Tunneling web requests is performed by creating a TCP connection with a proxy and requesting the proxy to *'connect'* to the target (see image below). The proxy handles the SSL and HTTP connection with the target and responds with a status 200 when the connection is established successfully. The attacker node is now able to send requests to and receive responses from the target through its TCP tunnel connection with the proxy. From the perspective of the target, the communication is established from the proxy and not from the attack-generating node, concealing the location of the attacker's infrastructure.

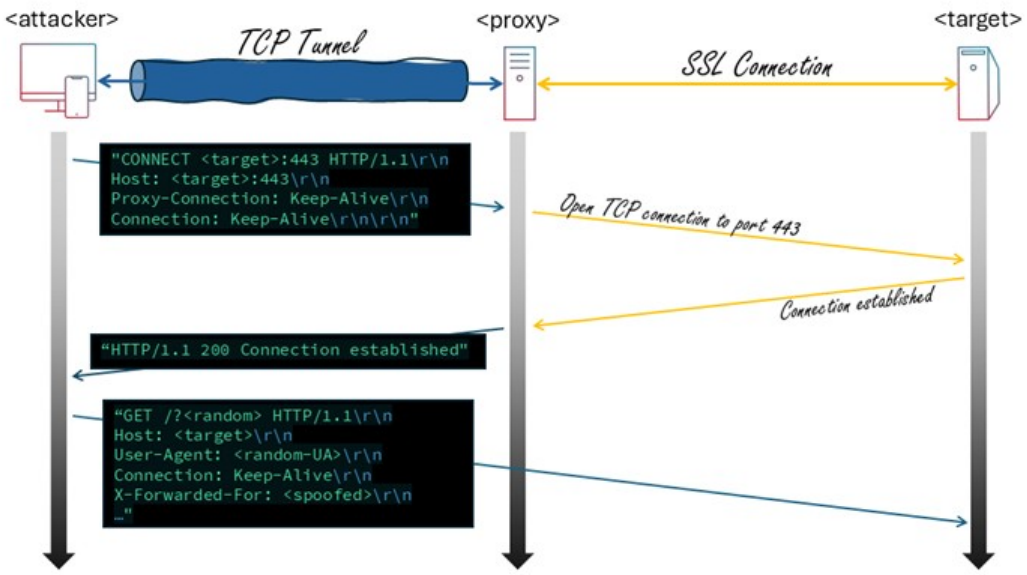


**Figure 13: A HTTPS request tunneled through a Proxy (source: Radware)**

Node.js provides several libraries that implement web requests through proxies such as the NPM (Node Package Manager) modules http-proxy-agent and https-proxy-agent. Proxying web requests using the https-proxy-agent module is as easy as creating an agent object using the statement “agent = new HttpsProxyAgent(‘http://<proxy-ip>:<proxy-port>’)”, followed by a regular HTTPS request using the https standard library module and passing the agent object as an argument (see image below).

**https module example**

```
import * as https from 'https';
import { HttpsProxyAgent } from 'https-proxy-agent';

const agent = new HttpsProxyAgent('http://168.63.76.32:3128');

https.get('https://example.com', { agent }, (res) => {
  console.log('"response" event!', res.headers);
  res.pipe(process.stdout);
});
```

**Figure 14: https-proxy-agent module example (source: GitHub)**

MegaMedusa, however, does not leverage the https-proxy-agent module. It implements proxy support natively by first creating a proxy socket, followed by the connect command to the proxy and then passing the established socket to the https module for performing the

GET/POST/HEAD/… request. Native implementations reduce the dependencies on third-party modules and leave the author in control of the connection options and the lifetime of the proxy connection.

## Optimizing Attack Request Rates

To support a world that has become more reliant on web and mobile applications providing real-time access to data combined with great user experience, the protocol specifications for HTTP were improved over time to include mechanisms that reduce the latency and increase the throughput, while decreasing resource requirements on both client and server. While these improvements were meant to improve the efficiency of applications and server infrastructure, they also serve malicious actors by making their attacks more efficient.

HTTP/1.1 pipelining is used primarily to improve the efficiency of sequential requests but suffers from limitations like head-of-line blocking. While not widely adopted due to its limitations, it is much liked by attackers because it allows them to increase the number of RPS from the same infrastructure without too much impact on resources and it is very easy to implement.

HTTP/2 introduced multiplexing, which allows multiple HTTP requests and responses to be performed concurrently over a single TCP connection. This implementation overcomes the limitations of HTTP/1.1 pipelining and provides a more efficient use of network resources, reducing latency and improving page load times for users. As with pipelining, multiplexing also allows attackers to efficiently increase their RPS numbers. To make it more evident, in October 2023, the HTTPS/2 Rapid Reset vulnerability was disclosed. This resulted in a large number of DDoS-for-hire services introducing this technique as a new attack vector in their arsenal of attack tools. Another notable vulnerability is the HTTP/2 Continuation vulnerability, which was disclosed in April 2024.

## Open and Commercial Proxies

Open proxies typically consist of compromised residential routers and servers. IoT botnets compromise vulnerable modems, routers and weakly secured cloud servers not only to generate attacks, but the same enslaved devices can also be leveraged as HTTP or SOCKS proxies by incorporating proxy functionality in the bot, installing a proxy server on the compromised device or just by reconfiguring the device and living off the land (LOTL) from services offered by the router or gateway. By leveraging devices located in residential IP ranges, attacks are less prone to be filtered as they share the same location as legitimate users.

Commercial proxy vendors deploy servers behind residential IP addresses and lease unused ranges from internet service providers. They consolidate the proxy service on a single or several Linux servers running, for example, a Squid proxy.

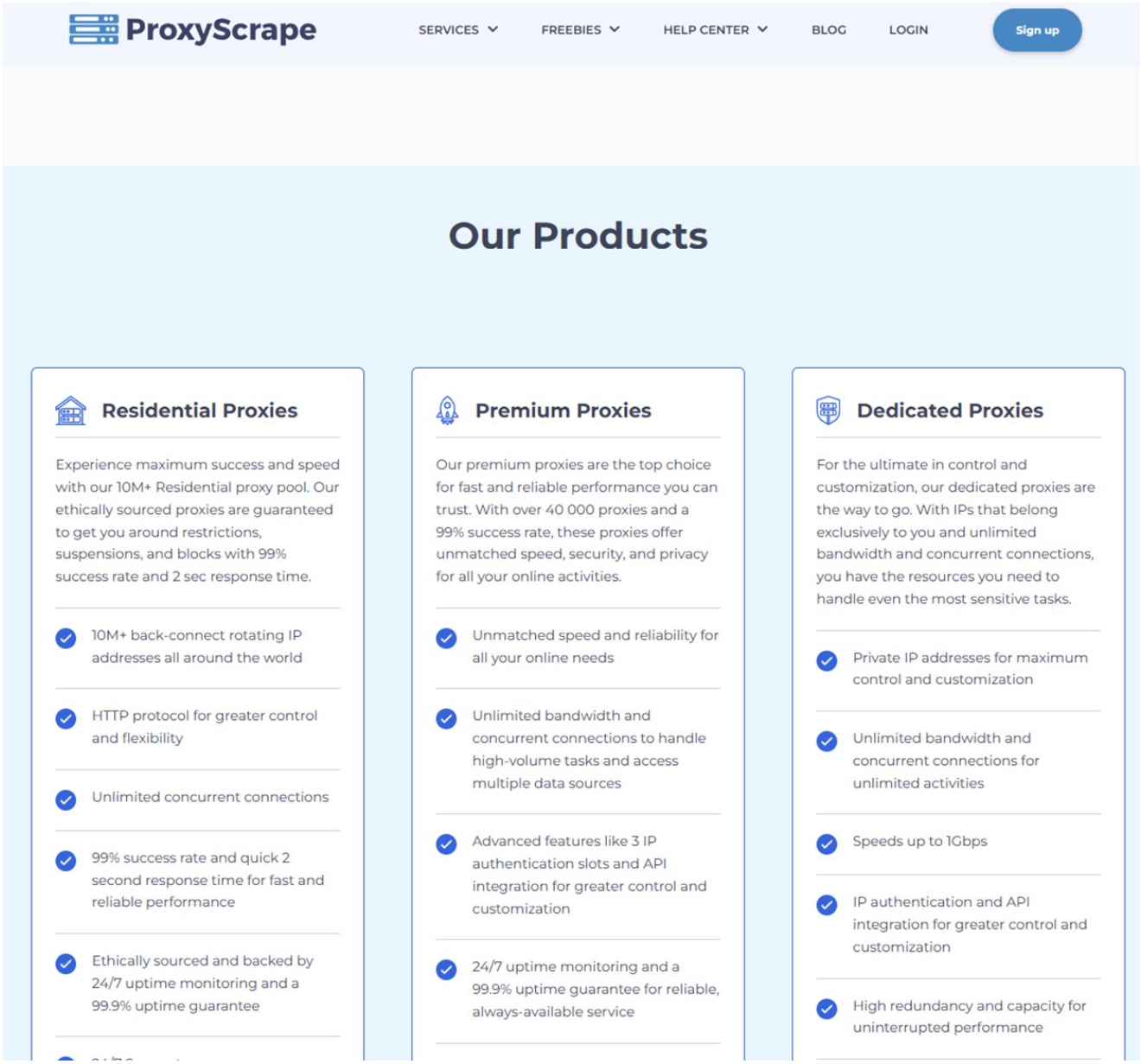


**Figure 15: Services offered by commercial proxy services (source: internet)**

Most commercial proxy services offer features like daily rotating IP addresses, which change 30% or more of the IP addresses in the pool of proxies every day. Attackers can leverage this to avoid blocking from IP feeds that provide longer-term analysis for the collection of malicious IP addresses.

Many commercial proxy providers also offer a free list. These lists can be leveraged without additional cost but with limitations in time or in the service rendered. There are also several unmanaged lists scattered around the internet, hosted publicly on GitHub and in other places, containing free and open proxies. Proxies from those lists are typically hit and miss, but by leveraging a proxy checker tool it is still possible to generate working lists of free and open proxies to perform attacks. The IT Army of Ukraine, for example, as part of its IT Army Kit, provides a curated list of proxies scraped from several locations on the internet and

includes it with their improved version of MHDDOS. More information about the IT Army Kit and DDoS tools offered by the IT Army of Ukraine to volunteers supporting its cause can be found here.

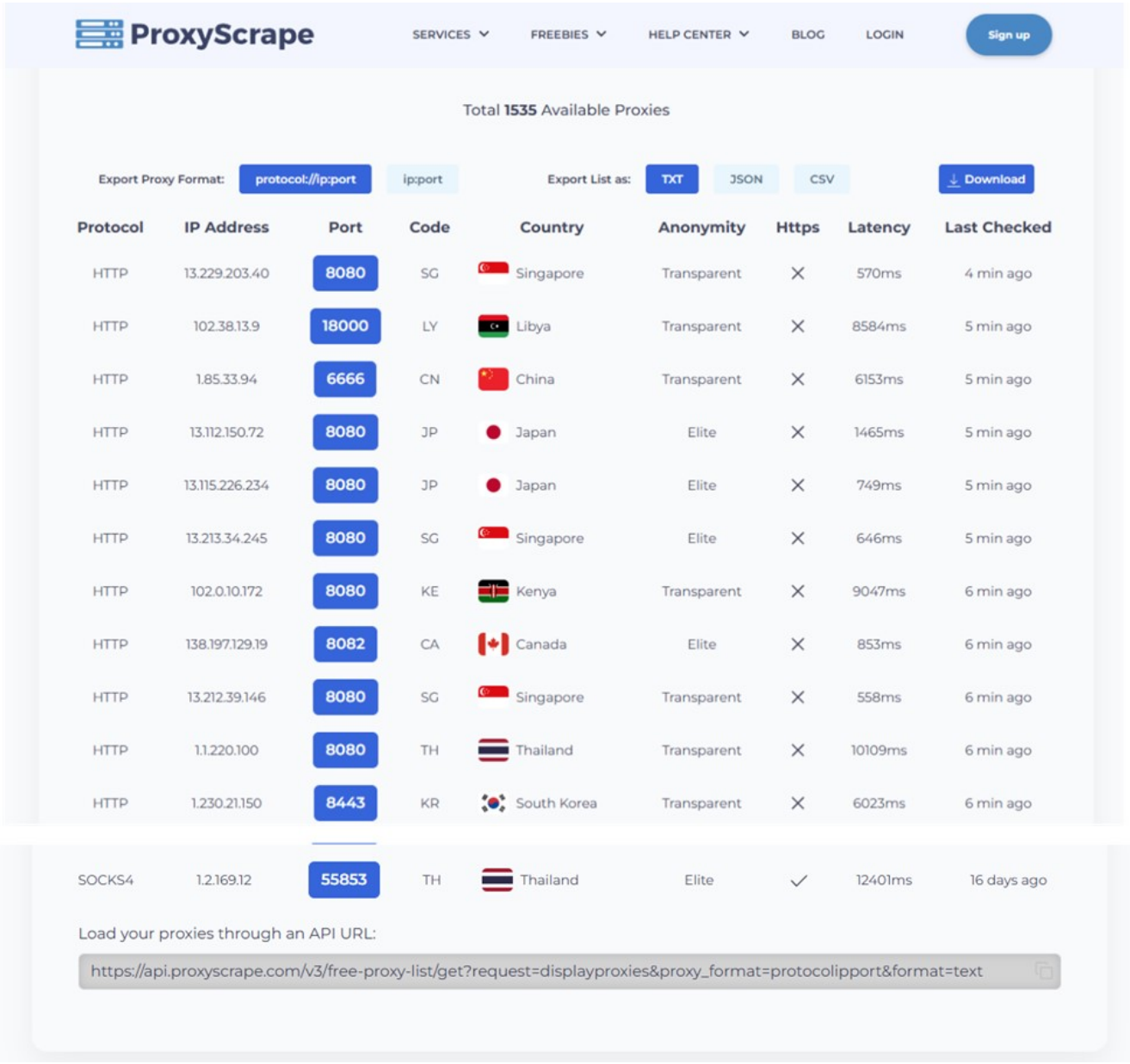


**Figure 16: Free proxy list offered by commercial provider, categorized by protocol, country and latency and providing an API URL for dynamically loading proxies in software (source: internet)**

## Advanced DDoS Attacker Infrastructure

Sophisticated DDoS attackers and DDoS-for-hire service providers leverage a whole arsenal of attack tools ranging from several variants of network DDoS amplification and reflection to Web DDoS with different randomizations of payloads and protection evasions. Depending on the attack, attackers conceal their infrastructure and increase their reach by leveraging source IP spoofing, proxy and SOCKS services and, in some cases, routing through the Tor overlay network.

Threat actors, like businesses, find agility and scale in the cloud. By transitioning from internet of things (IoT) devices to a cloud-centric model, they enhance scalability, reliability and ease of management compared to trying to control and maintain a huge number of infected IoT devices. The scanning and infecting of IoT devices is also more prone to getting noticed by honeypots, resulting in exposure and potentially the takedown of the command and control (C2) infrastructure.

Bulletproof hosters provide more lenient content policies and are often used for hosting content or services that regular hosting providers would not permit due to their policies. While they provide more freedom, they are often associated with nefarious activities and are scrutinized for hosting malicious content. They also provide malicious actors with a more stable environment as the providers are not responding to abuse and take down messages from the security community.

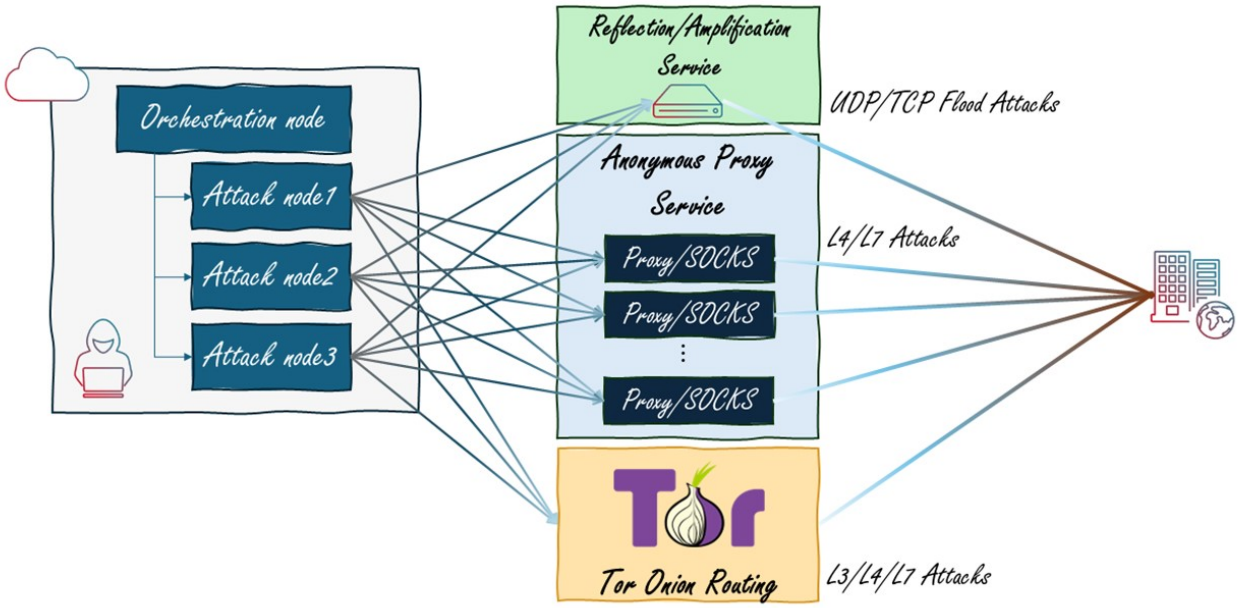


**Figure 17: Advanced DDoS attackers' cloud infrastructure (source: Radware)**

While the cloud provides many benefits for attackers' infrastructure, there are still plenty of botnets in use and leveraged for DDoS attacks. Botnets are good at creating highly distributed attacks. A botnet is the better platform to launch, for example, highly impactful DNS Water Torture or Pseudorandom Subdomain (PRSD) attacks as it leverages the trust relation between DNS forwarders and the residential IP ranges of the internet service providers (ISP). Some DDoS-for-hire services start with botnets as a cheaper alternative for their attack infrastructure. As they grow and need to provide more scalable and stable services to their customers, we see them transition to a hybrid infrastructure consisting of both botnets and cloud attack nodes.

The same IoT botnets used for DDoS attacks can also be leveraged for proxy and SOCKS services. Once IoT devices are compromised, it is rather easy for a bot herder to change or diversify his services by updating his bot's functionality and increasing and diversifying their revenue streams.

## Reasons for Concern

MegaMedusa is not the ultimate DDoS attack tool, but it is certainly good enough for a person with limited knowledge to perform attacks reaching levels that most websites will not be able to withstand without adequate Web DDoS protections.

As an attacker group, RipperSec can count on volunteers and allied groups to orchestrate devastating attacks. RipperSec's threat and scale do not come from a large and sophisticated attack infrastructure but from its community, which has always been the most powerful weapon of activists and hacktivists.

Posted in: Security Threat Intelligence

**Pascal Geenens**

As the Director, Threat Intelligence for Radware, Pascal helps execute the company's thought leadership on today's security threat landscape. Pascal brings over two decades of experience in many aspects of Information Technology and holds a degree in Civil Engineering from the Free University of Brussels. As part of the Radware Security Research team Pascal develops and maintains the IoT honeypots and actively researches IoT malware. Pascal discovered and reported on BrickerBot, did extensive research on Hajime and follows closely new developments of threats in the IoT space and the applications of AI in cyber security and hacking. Prior to Radware, Pascal was a consulting engineer for Juniper working with the largest EMEA cloud and service providers on their SDN/NFV and data center automation strategies. As an independent consultant, Pascal got skilled in several programming languages and designed industrial sensor networks, automated and developed PLC systems, and lead security infrastructure and software auditing projects. At the start of his career, he was a support engineer for IBM's Parallel System Support Program on AIX and a regular teacher and presenter at global IBM conferences on the topics of AIX kernel development and Perl scripting.