

# RESIP Host Detection: Identification of Malicious Residential IP Proxy Flows

**Tosun, Altug; De Donno, Michele; Dragoni, Nicola; Fafoutis, Xenofon**

*Published in:*
Proceedings of the 39th IEEE International Conference on Consumer Electronics

*Link to article, DOI:*
10.1109/ICCE50685.2021.9427688

*Publication date:*
2022

*Document Version*
Peer reviewed version

# RESIP Host Detection: Identification of Malicious Residential IP Proxy Flows

Altug Tosun, Michele De Donno, Nicola Dragoni and Xenofon Fafoutis
DTU Compute, Technical University of Denmark (DTU), Denmark
Email: altugtosun94@gmail.com, {mido,ndra,xefa}@dtu.dk

***Abstract*—The number of residential proxies as a service has grown as an emerging gray-area business, in which a provider utilises (compromised) hosts inside residential networks in order to provide anonymity services, effectively hiding the IP addresses of their clients behind other, seemingly normal, Residential IP (RESIP) addresses. This paper investigates commercial RESIP proxy service providers and looks into their host recruitment practices, which are often suspicious or borderline legal. In turn, the paper proposes a detection mechanism for identifying RESIP proxy flows on compromised consumer electronic devices, whereby the proxy software may operate without the user's knowledge or explicit consent.**

***Index Terms*—Residential Proxy, Proxy Detection, Traffic Analysis, IoT Security**

## I. Introduction

A proxy is software that has the purpose of mediating access between the client's machine and the destination server [1]. Immediately upon a client application generates a request to access a particular resource, the request is transferred through the network to a proxy server. After the proxy server obtains the request, the desired resource accompanied by its machine or appointed server is decided by the proxy server, along with any supplementary information that needs to be relayed. Once everything settled on the proxy server, the request is forwarded to the target server and the proxy server starts to wait for a response. Upon fetching the response, it forwards the response back to its end client [2]. The proxy may also implement caching to improve the performance.

Similarly to Virtual Private Networks (VPN), proxy servers allow a user to masquerade their internet traffic to seem as it is coming from a different Internet Protocol (IP) address. If a user is utilizing one of these services, the source IP address of the incoming traffic on the end node will belong to the proxy server. Since proxies typically reside in data centers, some web services started to block the network traffic coming from them in order to mitigate illegal activity [3]. Moreover, as the IP ranges of data-centers usually start with the same few integers, they are easy to block.

Consequently, proxy providers started to use residential hosts which have Residential IP (RESIP) addresses. A residential IP address is an IP address that is typically tied to personal devices, for example, a mobile phone or a desktop computer, or home networks. In contrast to non-residential IP addresses, residential ones are directly controlled and appointed by an Internet Service Provider (ISP) [4]. Although having a high level of privacy over the Internet is a privilege, directing one's network traffic through another person's device without the knowledge or consent of the device holder generates a lot of problems, such as advertisement and bank fraud. RESIP proxy service providers utilize residential hosts, which are recruited in dubious ways, as proxy servers, and detecting such malware-like applications constitutes a new challenge.

*ICCE'2021, January, 10-12, 2021, Las Vegas, USA. XXX-X-XXXX-XXXX-X/XX/$XX.00* 

The contribution of this paper is twofold. Firstly, we investigate commercial RESIP proxy service providers, providing further insight on their recruitment practices. Secondly, we propose a RESIP host detection algorithm that aims at identifying malicious proxy flows that operate on a system without the knowledge or explicit consent of its owner. The proposed detection mechanism is implemented and evaluated on traffic collected in a virtual and a real world environment.

The remainder of the paper is structured as follows. Section II summarizes the state of the art. Section III looks into commercial RESIP proxy service providers. Section IV presents our proposed proxy flow detection algorithm. Section V experimentally evaluates the proposed algorithm. Finally, VI concludes this paper.

## II. Related Work

The security aspects of web proxy services have been largely investigated in the literature. Weaver et al. [5] investigated the purpose of free proxy services based on how they modify traffic. The study used controlled clients and servers to exchange known HTTP messages and then looked for deviations from the expected. O'Neill et al. [6] investigated the existence of TLS proxies and identified thousands of malicious TLS proxies intercepting TLS communications. Carnavalet et al. [7] proposed a framework for the evaluation of client-end TLS proxies, by addressing limitations of regular TLS test suites. Perino et al. [8] created a distributed measurement platform in order to monitor the free proxy ecosystem. Tsirantonakis et al. [9] have designed a methodology for detecting proxies which, instead of passively relaying traffic, actively modified the relayed content. In contrast to the web proxy studies, this paper investigates RESIP services and focuses on the residential hosts as a way of forwarding traffic.

VPN detection has also been a hype research topic. Draper-Gil et al. [10] studied the effectiveness of flow-based time-related features to detect a VPN traffic, as well as character-

izing encrypted traffic into different categories with respect to the type of traffic. Abideen et al. [11] suggested a new approach to detect a VPN activity in an organization network. Unlike VPN detection studies, this paper will focus on RESIP proxies which does not use VPN technology in order to increase their stealthiness, even though there exists some VPN services that use residential hosts as a server.

The RESIP proxy approach is a fairly new concept that has not been investigated as much as other proxy approaches. Yet, there exists a few studies utilizing RESIP services. Chung et al. [3] studied a paid RESIP proxy service to uncover content manipulation in end-to-end connection. Chung et al. [12] performed extensive measurement study about how is Public Key Infrastructure (PKI) of Domain Name System's Security Extensions (DNSSEC) managed. Feng et al. [13] reported a first systematic study on RESIP services and provided information regarding their behaviours and the ecosystem of these dubious services. In contrast to prior studies that investigated RESIP services, this paper aims to detect RESIP proxy involvement on the residential host side rather than classifying them on the web server side.

## III. The Dark Side of RESIP Proxies

A residential proxy is an intermediary that runs on the devices (*e.g.* laptops, smartphones) of Internet users, exploiting the residential IP addresses that is provided by their ISP. Residential proxy users connect to the target on the web through residential hosts. These proxies are the most difficult to detect since they appear as real Internet users from the web server's perspective. Moreover, since legitimate traffic and proxy traffic appear to come from the same source, it is difficult for a detection mechanism to differentiate between these two. Residential proxy services do not let tracking tools recognize the real location of the user.

RESIP proxy approach is an emerging online gray business, whose security implications have never been studied before [13]. Therefore, this section attempts to reveal the evil side of RESIP services in detail. When residential proxy services are abused, they can outmatch traditional public proxies or even anonymity networks to assist their clients masquerade as clean and benevolent sources to communicate with the targets.

As the number of proxy server users grows each day, there are various legitimate reasons for organizations and individuals to make use of these services. Even though most of the users of proxy services utilize these functionalities for benign purposes, there exists a fair amount of malicious parties that is utilizing proxy services for illegal activities. In particular, anonymous IP addresses are frequently utilized in order to boost income that is gained from online advertising. Malicious programmers can develop programs to create fabricated online clicks, actions and data. With the help of anonymous IP addresses, cyber-criminals can change the source IP address of incoming traffic to any desired geographical location in order to have the correct location for getting payouts from advertisement [9]. Furthermore, anonymous IP addresses open up a way to dodge network traffic control tools that check for repeated views and clicks from one or few IP addresses. Similar approaches are also utilized to promote products, for example, to encourage the installation of application by increasing the number of downloads [14].

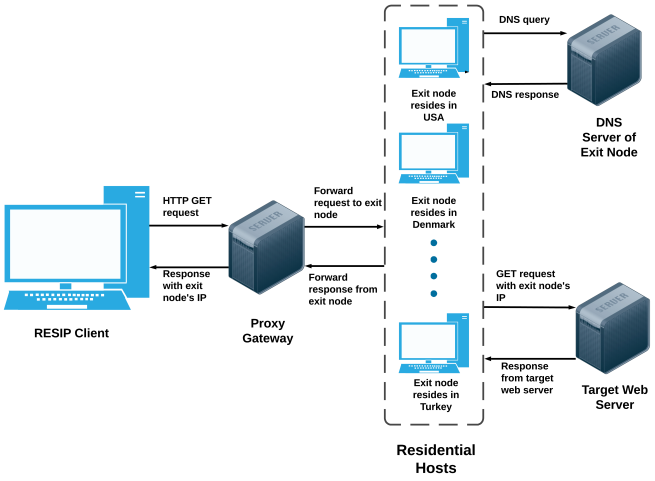


Fig. 1. RESIP proxy service architecture from an outsider's point-of-view.

Another illegal activity that is carried out with the help of the anonymity provided by proxy servers is to make a transaction with stolen bank account information. Indeed, in such transactions, the billing address is often checked against the IP address location of the customer. Malicious users can utilize anonymous IP addresses to spoof geographical locations near to the billing addresses of the victims [15].

As the last example of malicious use of proxy services, users of media streaming services are able to access the media content that is unavailable in their region, bypassing IP-based geo-blocking [8]. Since media streaming companies are responsible to make sure that digital rights are protected regarding the geographical locations, this mischievous behaviour of users can lead to loss of content licenses which can result in loss of reputation, market share and profit for the company. As a result, for example, Netflix, is blocking connections from all proxies and VPNs to enforce its copyright obligations [16].

### A. RESIP Proxy Providers

The existence of new RESIP proxy services has been more prevalent than ever in the recent years. These services managed to gain the upper hand over all of the traditional proxy services [13]. Fig. 1 illustrates a typical RESIP service architecture, as also shown in the prior studies [12], [3], [13]. The client's traffic is forwarded through residential hosts which have residential IP addresses supplied by ISPs. ISPs register residential IP addresses in public databases, which enables web servers to determine the client's location and ISP. For that reason, most online services recognize residential IP addresses as real users, as opposed to those who use data center IP addresses [17]. To make things worse, RESIP proxy services hide the user behind a pool of proxies instead of a single IP address. This technique, called "rotating residential proxies"

(also known as "back-connect proxies"), changes the exit node on every session or at regular intervals.

Consequently, modern RESIP proxies are very resilient to server-side detection and blocking. Even if a suspicious connection is detected by a server, it does not make sense to block the IP address for two main reasons. Firstly, ISP providers are assigning the same IP address to multiple clients behind Network Address Translation (NAT), since the IPv4 addresses have been depleted in many regions. Therefore, blocking an IP address means that the services will be unavailable for real users along with the proxy host. Secondly, if IP rotation is applied, blocking the IP address of a single RESIP proxy connection is completely futile.

Commercial RESIP proxy providers are in possession of huge pools of different RESIP addresses. These services have IP addresses from all around the world. Due to this global portfolio of residential IP addresses, RESIP service clients can masquerade their web connections and can leave fingerprints from any city, country and region as desired. Table I provides an overview of the major RESIP proxy providers, summarizing information collected from their websites as of July 2020.

TABLE I
RESIP POOL OF PROXY SERVICE PROVIDERS (DATA RETRIEVED ON JULY 2020, NUMBERS MIGHT BE DIFFERENT AT THE TIME OF READING)

| Proxy Provider | IP Addresses | Countries |
|---|---|---|
| luminati.io | ≈ 72,000,000 | 215 |
| smartproxy.com | ≈ 40,000,000 | 235 |
| shifter.io | ≈ 31,000,000 | Claims every city is available |
| oxylabs.io | ≈ 72,000,000 | 186 |
| geosurf.com | ≈ 2,500,000 | 94 |
| homeip.io | ≈ 13,000,000 | 157 |
| ipburger.com | ≈ 60,000,000 | 195 |
| netnut.io | ≈ 500,000 | 50 |
| proxyrack.com | ≈ 500,000 | 140 |
| scraperapi.com | ≈ 40,000,000 | 12 |
| intoli.com | unspecified | unspecified |
| blezingseollc.com | unspecified | 10 |
| stormproxies.com | ≈ 70,000 | Seperated as EU and USA |

The majority of the RESIP proxy providers do not provide any information on their recruitment practices. Only Luminati (`luminati.io`) provides a description of the recruitment process [18]. Luminati has a Software Development Kit (SDK) which is integrated into various types of applications. Luminati is making agreements with application developers to place this SDK in their application. In turn, the users of applications can choose become a Luminati exit node, in return for an advertisement-free application. For every user that opts in to the network, Luminati pays a monthly fee to the application vendor. The HolaVPN application, which is considered as the biggest contributor to Luminati's host pool, can be given as an example. Once a user installs the free HolaVPN, she is recruited as one of Luminati's exit nodes [19].

Apart from Luminati, the recruitment techniques of other RESIP services remain unknown. Yet, it is reasonable to assume that they have followed a similar approach or worse. Even though some of the providers may claim that the user is consented to be part of residential proxy network, in reality, the user is often unaware of the dangerous consequences of being part of such traffic forwarding activity.

Proxies are being heavily utilized by cyber-criminals, and RESIP proxies only makes the matter worse. Indeed, using some other individual's IP address to hide the real IP address is way more dangerous than IP spoofing, and is already identified as a significant threat by several studies in the literature [20], [21], [22]. Prior work on RESIP proxies [13], revealed that residential hosts are abused as attack intermediaries to be part of illegal activities such as fast fluxing, phishing, malware hosting, blackhat SEO, and botnet campaigns. Surprisingly, even in the RESIP pool of Luminati, which is claiming to have compliance officers to handle every report of abuse including investigating, warning, and blocking suspicious clients, 2.32% of the RESIPs were found as blacklisted in the study.

In summary, residential hosts are not protected by proxy providers, instead they are being abused by them, similarly to how a botmaster control botnets. On that account, these IP addresses are the primary choice for evil actors for bypassing geographic filters, submitting fake data through lead generation channels, exploiting free trials, creating duplicate accounts, engaging in click or ad fraud, amongst others [23].

## IV. RESIP HOST DETECTION

We define as malicious RESIP proxy a situation whereby a residential device participates in a RESIP proxy network without the explicit consent of its owner. Next, we propose a mechanism for the detection of malicious RESIP proxy flows on a residential host. The detection mechanism is composed of three anomaly detection algorithms [24], detailed as follows.

Intuitively, the proxy traffic is expected to have a specific ordering among the packets in the network traffic flow. This *Suspicious Flow* is expected to appear on the residential hosts that are acting as a proxy in a proxy network. The suspicious flow is described as follows: (i) the client needs to send to the proxy the information regarding the resource on the web that it is trying to get; (ii) as the resource is retrieved, it is expected to generate a flow of packets between the compromised host and the web server; (iii) the compromised host needs to forward the resource that is fetched from the web server to the client. The algorithm for detecting flows with these suspicious patterns is provided in Algorithm 1.

A second anticipated characteristic of a proxy server network flow is that the amount of data that is sent back to the client needs to be at least as big as the data that is fetched from the target web server. One might argue that residential proxy providers might make changes on the data, however, as discussed in Section II, such modifications will be recognized by traditional proxy detection systems and reduce the stealthiness of the residential proxies. The algorithm for this *Data Check* is provided in Algorithm 2.

**Algorithm 1** Suspicious Flow Check

$device\_ip \leftarrow$ IP address of the device
$network\_flow \leftarrow$ List of packets extracted from pcap file
$threshold \leftarrow$ Chosen threshold for suspicion
Initialize $suspicious\_connections$ as an empty list
Initialize $inc\_flow\_ips$ as an empty list
Initialize $out\_flow\_ips$ as an empty list
Initialize $prev\_inc\_pckt$ as an empty string
Initialize $prev\_out\_pckt$ as an empty string

**for** each packet in $network\_flow$ **do**
    **if** destination ip address of packet = $device\_ip$ **then**
        **if** source ip address of packet = $prev\_inc\_pckt$ **then**
            skip to processing next packet
        **else if** source ip address of packet not in $inc\_flow\_ips$ **then**
            insert source ip address of packet into $inc\_flow\_ips$
        $prev\_inc\_pckt \leftarrow$ source ip address of packet

    **else if** source ip address of packet = $device\_ip$ **then**
        $distance \leftarrow$ CALCULATEDISTANCE(position of packet, destination ip of packet)
        **if** destination ip address of packet = $prev\_out\_pckt$ **then**
            skip to processing next packet
        **else if** destination ip address of packet not in $out\_flow\_ips$ **then**
            insert destination ip address of packet into $out\_flow\_ips$
        **else if** destination ip address of packet in $inc\_flow\_ips$ **and** $distance > threshold$ **then**
            insert suspicious connection into $suspicious\_connections$
        $prev\_out\_pckt \leftarrow$ destination ip address of packet

**procedure** CALCULATEDISTANCE($pos, ip$)
    $distance \leftarrow 0$
    **for** i $\leftarrow pos$ to 0 **do**
        **if** $ip$ = source ip of $network\_flow[i]$ **then**
            $distance \leftarrow pos - i$
            **break**
    **return** $distance$

**Algorithm 2** Data Check

$device\_ip \leftarrow$ IP address of the device
$suspicious\_connections \leftarrow$ List of the suspicious connections detected in the algorithm 1
$network\_flow \leftarrow$ List of packets extracted from pcap file
$time\_threshold \leftarrow$ Chosen threshold for required time to send response back

**for** each connection in $suspicious\_connections$ **do**
    $total\_data\_outgoing \leftarrow$ TOTALDATASENT(position of connection, IP address of connection)
    $total\_data\_incoming \leftarrow$ TOTALDATARECEIVED(position of connection, number of packets flowed during suspicion)
    **if** $total\_data\_incoming \leq total\_data\_outgoing$ **then**
        update $suspicious\_connections$ by adding violation of data to current connection

**procedure** TOTALDATASENT($pos, ip$)
    $total\_data \leftarrow 0$
    $start\_time \leftarrow$ time of $network\_flow[pos]$
    **for** i $\leftarrow pos$ to size of $network\_flow$ **do**
        $time\_diff \leftarrow$ time of $network\_flow[i]$ - $start\_time$
        **if** $time\_diff < time\_threshold$ **and** ip = destination ip of $network\_flow[i]$ **then**
            $total\_data \leftarrow total\_data +$ size of $network\_flow[i]$
        **else if** $time\_diff \geq time\_threshold$ **then**
            **break**
    **return** $total\_data$

**procedure** TOTALDATARECEIVED($pos, num\_of\_packets$)
    Initialize $incoming\_data$ as an empty dictionary
    **for** i $\leftarrow pos$ - $num\_of\_packets$ to $pos$ **do**
        **if** destination ip of $network\_flow$[i] = $device\_ip$ **then**
            key = source ip of $network\_flow$[i]
            **if** key in keys of $incoming\_data$ **then**
                value of $incoming\_data$[key] $\leftarrow incoming\_data$[key] + size of $network\_flow$[i]
            **else**
                insert [key, size of $network\_flow$[i]] into $incoming\_data$
    $total\_data \leftarrow$ select maximum total data sent by single IP from $incoming\_data$
    **return** $total\_data$

Finally, as it can be seen in Fig. 1 and also in several similar studies (see [12] [3] and [13]), before fetching the data from the target web server, the compromised host is expected to make a DNS lookup (unless the web server is already in the local DNS cache). The algorithm for conducting a *DNS Check* is provided in Algorithm 3.

## V. EVALUATION

The proposed RESIP proxy detection mechanism is implemented in Python[1]. To validate the proposed algorithms, we collect data in a virtualized environment, whereby a virtual machine plays the role of a client of a RESIP proxy service (Client) and a second virtual machine plays the role of a compromised host (Proxy), which is seized to be used as an exit node in a RESIP proxy service. A request from the client will be forwarded through the compromised host. Upon receiving the response from the target web server (Web Server), it will be forwarded back to the Client by the Proxy. In parallel, the legitimate user of the compromised host is also generating (legitimate) HTTP requests to web servers. In this environment, we periodically generate legitimate and proxy requests to a random web server (from the Alexa's top 100 most popular websites[2]) at various rates. The network traffic is collected at the compromised host (Proxy). Studies on the proxy detection (see [5], [8], [9]) revealed that if a proxy server applies content modification, it can be detected. We thus assume that the Proxy does not apply any modification to the requests and the responses. Using this experimental setup, we consider four 30-minute scenarios (Case 1-4) using various rates (requests per minute), as summarized in Table II.

For validation, we also repeat the experiment on two real computers (Real World). The key difference compared with the virtual environment is additional noise in the networking interface which makes the detection more challenging. Even though connection and proxy setup between the two computers are established with the same approach of the cases with virtual machines, the computer that is acting as the compromised host has different network traffic flow characteristics than a virtual machine. The request rates of the Real World scenario are also shown in Table II.

TABLE II
EVALUATED SCENARIOS (RATE IN REQUESTS PER MINUTE)

| Scenario | Legitimate Flows | | | Proxy Flows | | |
|---|---|---|---|---|---|---|
| | intensity | rate | total | intensity | rate | total |
| Case 1 | moderate | 16.9 | 509 | moderate | 2.3 | 70 |
| Case 2 | moderate | 17.4 | 521 | high | 56.1 | 1684 |
| Case 3 | high | 107.7 | 3231 | moderate | 1.67 | 50 |
| Case 4 | high | 118.2 | 3546 | high | 63.4 | 1902 |
| Real World | moderate | 15.8 | 473 | moderate | 2.6 | 79 |

In turn, we apply the proposed RESIP proxy detection

[1]Source code available at https://github.com/dtu-ese/resip-proxy-detection.
[2]https://www.alexa.com/topsites

**Algorithm 3** DNS Check

```
suspicious_connections ← List of the suspicious connections updated in the algorithm 2
network_flow ← List of packets extracted from pcap file
time_threshold ← Chosen threshold for required time to make a DNS lookup

for each connection in suspicious_connections do
    time_diff ← DNSCHECK(position of connection, number of packets flowed during suspicion)
    if time_diff ≥ time_threshold then
        skip to the next iteration of connections
    else
        update suspicious_connections by adding violation of DNS to current connection

procedure DNSCHECK(pos, num_of_packets)
    iterator ← pos − num_of_packets
    start_time ← time of network_flow[iterator]
    time_diff ← maximum integer

    while True do
        packet ← network_flow[iterator]
        if iterator = pos then
            break
        if protocol of packet = DNS then
            time_diff = time of packet − start_time
            break
        iterator ← iterator + 1
    return time_diff
```

algorithms on the collected datasets[3]. To further understand the effectiveness of each of the proposed algorithms (Algo), we apply them in four different combinations, namely 1, 1-2, 1-3 and 1-2-3. For example, in 1-2, a flow is predicted to be a proxy flow if both Algorithm 1 and Algorithm 2 return true.

TABLE III
EXPERIMENTAL RESULTS: ACCURACY, PRECISION AND RECALL IN %, FOR EACH CASE THE BEST PERFORMANCE IS MARKED IN BOLD

| | Algo | TP | TN | FP | FN | Acc | Prec | Rec |
|---|---|---|---|---|---|---|---|---|
| Case 1 | 1 | 69 | 314 | 195 | 1 | 66.2 | 26.1 | **98.6** |
| | 1-2 | 67 | 509 | 0 | 3 | **99.5** | **100** | 95.7 |
| | 1-3 | 69 | 383 | 126 | 1 | 78.1 | 35.4 | **98.6** |
| | 1-2-3 | 67 | 509 | 0 | 3 | **99.5** | **100** | 95.7 |
| Case 2 | 1 | 1683 | 212 | 309 | 1 | 85.9 | 84.5 | **99.9** |
| | 1-2 | 1605 | 515 | 6 | 79 | 96.2 | **99.6** | 95.3 |
| | 1-3 | 1683 | 354 | 167 | 1 | 92.4 | 91 | **99.9** |
| | 1-2-3 | 1605 | 521 | 0 | 79 | **96.4** | 95.3 | **99.9** |
| Case 3 | 1 | 49 | 2036 | 1195 | 1 | 63.6 | 3.9 | **98** |
| | 1-2 | 43 | 3230 | 1 | 7 | **99.8** | **97.7** | 86 |
| | 1-3 | 49 | 2316 | 915 | 1 | 72.1 | 5.1 | **98** |
| | 1-2-3 | 43 | 3230 | 1 | 7 | **99.8** | **97.7** | 86 |
| Case 4 | 1 | 1780 | 1794 | 1752 | 122 | 65.6 | 50.4 | **93.6** |
| | 1-2 | 1548 | 3192 | 33 | 354 | 92.5 | 97.9 | 81.4 |
| | 1-3 | 1775 | 2255 | 1291 | 127 | 74 | 57.9 | 93.3 |
| | 1-2-3 | 1545 | 3536 | 10 | 357 | **93.3** | **99.4** | 81.2 |
| Real World | 1 | 77 | 289 | 184 | 2 | 66.3 | 29.5 | **97.5** |
| | 1-2 | 73 | 465 | 8 | 6 | **97.5** | **90.1** | 92.4 |
| | 1-3 | 19 | 414 | 59 | 60 | 78.4 | 24.4 | 24.1 |
| | 1-2-3 | 19 | 466 | 7 | 60 | 87.9 | 73.1 | 24.1 |

[3]Data available at https://www.doi.org/10.11583/DTU.12729530.

The results of the experiments are summarized in Table III. Specifically, the table presents the outcome of the malicious proxy flow detection for each of the examined scenarios (Case 1-4 and Real World) and for each of the algorithm combinations (Algo), shown as the total number of True Positives (TP), True Negatives (TN), False Positives (FP) and False Negatives (FN). In addition, the table shows the Accuracy (Acc), Precision (Prec), and Recall (Rec) as a percentage.

The results suggest that Suspicious Flow Check (Algorithm 1) is the least accurate one. However, considering that it is the prerequisite condition for the other checks, it is sensible to have a large amount of false positives. If this check was designed to be too selective with suspicious connections, then there would have been a considerable amount of misses for the further checks because suspicious connections would not be passed to these checks at all.

The experiments also suggest that the Data Flow Check (Algorithm 2) is the one that contributes most to an accurate detection. Even without the DNS check, *i.e.* combination 1-2, the accuracy and precision of the results increase significantly for all of the scenarios, including the real world scenario. Also, the Data Check algorithm was the one that resulted with the least false positives which can be seen as the reason of the increase in the accuracy and precision.

The DNS Check (Algorithm 3) helped with the accuracy and precision to the some extent, but not as much as the Data Check. Still, it appears to be more resilient to false negatives than the data check, and it rarely missed an actual proxy connection. For the case with the real computers, however, the DNS Check did not perform well, due to caching: no DNS lookup made for the websites that were already in the DNS cache of the computer representing the proxy host.

Let us now compare the simulated scenarios (Case 1-4) against the Real World scenario. Unlike the simulation environment, in the case with the real computers, there exists a lot of network traffic generated by the machine, beyond our control. Yet, the results for the Suspicious Flow Check (Algorithm 1) and Data Check (Algorithm 2) perform similarly to the cases with virtual machines. This increases our confidence that these algorithms are solid enough and not affected by the uncontrolled network traffic flow.

However, for the DNS Check (Algorithm 3), as stated before, the performance decreased significantly because of the DNS caching of the real computer. We note, however, that in this experiment we randomly request very popular websites (taken from Alexa's top 100) and popular websites are more likely to appear in DNS caches. As a result, in a real life scenario, we expect to see more DNS lookups and, thus, Algorithm 3 to be more useful. In other words, DNS lookups are expected to happen more frequently than the Real World scenario, but not as frequently as in the virtual environment.

Next, let us have a look at some of the failures of the Data Check (Algorithm 2). For some websites (such as `blogger.com`, `telegram.org` and `aliexpress.com`), the total data that is forwarded back to the Client is less than the total data received by the compromised host (Proxy) during

the connection with the web server. For such websites, it is suspected that they send big certificates for the connection establishment. Hence, the encryption between the compromised host and the target website might increase the data size; as a result, the amount of the total data received from the target web server becomes bigger than the total data sent back to the client. In future work, the encryption implementations of these websites could be investigated further to test this hypothesis.

To sum up, the detection mechanism performed well even under the high legitimate and high proxy network traffic in the virtualized environment, which can be considered as a really challenging case. In addition, the detection mechanism is proven to be consistent when it is used in the real environment, with the exception of the DNS check. Therefore, it can be concluded that the detection mechanism will provide a solid groundwork for further studies to detect suspicious network traffic activity on the device acting as a RESIP proxy.

Finally, it can be argued that it is possible that RESIP proxy services could be engineered to circumvent our checks. Yet, in order to keep network traffic ordinary and undetectable, options are limited for them. For every effort that is made to escape from detection, it would also mean that network traffic will be less natural, and thus more vulnerable to traditional proxy detection techniques (see Section II). In addition, the algorithmic checks that are proposed in this study could be tailored to different versions of RESIP systems in order to be utilized in the detection.

## VI. Conclusion

Proxy services are rising online businesses, whose security aspects are being analyzed in the literature for some time. In this study, we shed light on RESIP proxy services, which represent the latest approach among the proxy techniques. It is unveiled that residential proxy services can outmatch traditional public proxies or even anonymity networks in assisting their clients masquerade as clean and benevolent sources. Indeed, RESIP services are almost undetectable on the server side. In this paper, we proposed a RESIP proxy detection mechanism that operates on the compromised host (*i.e.* the proxy). The mechanism is composed of three algorithms that perform different checks on the network traffic. We evaluated and compared various combinations of these algorithms in a virtual and real world environment. The results indicate that our proposed mechanism can successfully detect malicious RESIP proxy services that run on a compromised host.

Even though the detection mechanism was able to find malicious proxy connections with a decent accuracy and precision, further research on RESIP services is needed to get a more comprehensive and complete system able to detect a residential host involvement in a RESIP service. From now onward, these uncontrolled RESIP services indeed endanger the Internet users and certain regulations are necessary.

## References


[1] M. Pannu, B. Gill, R. Bird, K. Yang, and B. Farrel, "Exploring proxy detection methodology," in *2016 IEEE International Conference on Cybercrime and Computer Forensic (ICCCF)*. IEEE, 2016, pp. 1–6.

[2] D. Stuttard and M. Pinto, *The Web Application Hacker's Handbook: Discovering and Exploiting Security Flaws*, 2nd ed. John Wiley & Sons, 2011.

[3] T. Chung, D. Choffnes, and A. Mislove, "Tunneling for Transparency: A Large-Scale Analysis of End-to-End Violations in the Internet," in *Internet Measurement Conference (IMC)*, 2016, pp. 199–213.

[4] E. J. Hernandez-Valencia, "Architectures for broadband residential IP services over CATV networks," *IEEE Network*, vol. 11, no. 1, pp. 36–43, 1997.

[5] N. Weaver, C. Kreibich, M. Dam, and V. Paxson, "Here be web proxies," in *Passive and Active Measurement*, 2014, pp. 183–192.

[6] M. O'Neill, S. Ruoti, K. Seamons, and D. Zappala, "TLS Proxies: Friend or Foe?" in *Proc. Internet Measurement Conference (ICM)*, 2016, pp. 551–557.

[7] X. d. C. de Carnavalet and M. Mannan, "Killed by proxy: Analyzing client-end TLS interception software," in *NDSS*, 2016.

[8] D. Perino, M. Varvello, and C. Soriente, "ProxyTorrent: Untangling the Free HTTP(S) Proxy Ecosystem," in *Proc. 2018 World Wide Web Conference (WWW)*, 2018, pp. 197–206.

[9] G. Tsirantonakis, P. Ilia, S. Ioannidis, E. Athanasopoulos, and M. Polychronakis, "A Large-scale Analysis of Content Modification by Open HTTP Proxies." in *NDSS*, 2018.

[10] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related," in *Proc. 2nd international conference on information systems security and privacy (ICISSP)*, 2016, pp. 407–414.

[11] M. Z. ul Abideen, S. Saleem, and M. Ejaz, "VPN Traffic Detection in SSL-Protected Channel," *In Security and Communication Networks*, pp. 1–17, 2019.

[12] T. Chung, R. van Rijswijk-Deij, B. Chandrasekaran, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson, "A Longitudinal, End-to-End View of the DNSSEC Ecosystem," in *26th USENIX Security Symposium*, 2017, pp. 1307–1322.

[13] X. Mi, X. Feng, X. Liao, B. Liu, X. Wang, F. Qian, Z. Li, S. Alrwais, L. Sun, and Y. Liu, "Resident Evil: Understanding Residential IP Proxy as a Dark Service," in *IEEE Symposium on Security and Privacy*, 2019, pp. 1185–1201.

[14] M. Atienza, "Types of anonymous ips and how they affect your business," https://blog.maxmind.com/2019/01/24/types-of-anonymous-ips-and-how-they-affect-your-business, 2019.

[15] J. Taylor, J. Devlin, and K. Curran, "Bringing location to IP Addresses with IP Geolocation," *Journal of Emerging Technologies in Web Intelligence*, vol. 4, no. 3, pp. 273–277, 2012.

[16] "The Netflix proxy error and how to bypass it with a VPN," https://www.comparitech.com/blog/vpn-privacy/netflix-proxy-error.

[17] "What are residential proxies?" https://smartproxy.com/blog/what-is-a-residential-proxies-network, 2020.

[18] R. Hollander, "How luminati obtains its residential IPs," https://luminati.io/blog/luminati-network-sdk?cam=tb_re_23, 2019.

[19] "Shining a light on the risks of holavpn and luminati," www.trendmicro.com/vinfo/hk-en/security/news/cybercrime-and-digital-threats/shining-a-light-on-the-risks-of-holavpn-and-luminati, 2018.

[20] Q. Wang, X. Gong, G. T. Nguyen, A. Houmansadr, and N. Borisov, "CensorSpoofer: Asymmetric Communication Using IP Spoofing for Censorship-Resistant Web Browsing," in *Proc. ACM Conference on Computer and Communications Security (CCS)*, 2012, pp. 121–132.

[21] T. W. Doeppner, P. N. Klein, and A. Koyfman, "Using router stamping to identify the source of IP packets," in *7th ACM Conference on Computer and Communications Security (CCS)*, 2000, pp. 184–189.

[22] C. Jin, H. Wang, and K. G. Shin, "Hop-count filtering: an effective defense against spoofed DDoS traffic," in *10th ACM conference on Computer and Communications Security (CCS)*, 2003, pp. 30–41.

[23] "How residential proxies enable fraud," https://www.ipqualityscore.com/articles/view/13/how-residential-proxies-enable-fraud, 2018.

[24] M. Thottan and Chuanyi Ji, "Anomaly detection in IP networks," *IEEE Transactions on Signal Processing*, vol. 51, no. 8, pp. 2191–2204, 2003.