# Silent Observers Make a Difference: A Large-scale Analysis of Transparent Proxies on the Internet

Rui Bian*, Lin Jin*, Shuai Hao†, Haining Wang‡, Chase Cotton*
*University of Delaware, Newark, DE, USA
†Old Dominion University, Norfolk, VA, USA
‡Virginia Tech, Arlington, VA, USA
{bianrui, linjin, ccotton}@udel.edu, shao@odu.edu, hnw@vt.edu

*Abstract*—**Transparent web proxies have been widely deployed on the Internet, bridging the communications between clients and servers and providing desirable benefits to both sides, such as load balancing, security monitoring, and privacy enhancement. Meanwhile, they work silently as clients and servers may not be aware of their existence. However, due to their invisibility and stealthiness, transparent proxies remain understudied for their behaviors, suspicious activities, and potential vulnerabilities that could be exploited by attackers. To better understand transparent proxies, we design and develop a framework to systematically investigate them in the wild. We identify two major types of transparent web proxies, named** `FDR` **and** `CPV`**, respectively. FDR is a type of transparent proxy that independently performs *F*orced *D*NS *R*esolution during interception. CPV is a type of transparent proxy that presents *C*ache *P*oisoning *V*ulnerability. We perform a large-scale measurement to detect each type of transparent web proxy and scrutinize their security implications. In total, we identify 32,246 FDR and 11,286 CPV transparent proxies. We confirm that these two types of transparent proxies are distributed globally — FDRs are observed in 98 countries and CPVs are observed in 51 countries. Our work highlights the issues of vulnerable transparent proxies and provides insights for mitigating such problems.**

*Index Terms*—**Web Proxy, DNS, Cache Poisoning**

## I. Introduction

Transparent proxies [1]–[5] are one type of web proxy servers [6]–[14] that relay the traffic between clients and servers. Transparent proxies intercept requests and responses between clients and web servers, but clients and web servers may not be aware of the existence of transparent proxies. The transparent proxies are typically deployed by ISPs (Internet Service Providers) and enterprises or are enabled as a function on the user-side devices such as home routers so that the proxy servers can monitor, filter, and censor the traffic [15]–[18]. Moreover, by caching the contents [19]–[23], transparent proxies can reduce the traffic volume effectively. However, transparent proxies may be legacy proxies that are not well-managed and updated. Transparent proxies may be vulnerable to known attacks such as cache poisoning [24] and Denial of Service attacks. There are only a few prior types of research measuring and studying transparent web proxies [2], [3].

Previous research has delved into the analysis of transparent proxies and their effects on network traffic. Xu *et al.* [2] examined the behavior of transparent web proxies in major US cell carriers, while Zhang *et al.* [3] investigated HTTP traffic manipulation by transparent proxies in China-wide networks, including the injection of advertisements and privacy concerns. In a similar vein, Mi *et al.* [25] and Yang *et al.* [26] explored residential proxy ecosystems, uncovering security issues associated with potentially unwanted programs. However, our study distinguishes itself by focusing specifically on a global measurement study of residential transparent proxies. By examining their prevalence and behavior on a broader scale, we aim to provide a comprehensive understanding of the security implications and impact of residential transparent proxies in the modern Internet landscape.

In this study, we investigate an overlooked issue of web browsing, the stealthy interception by on-path devices, especially transparent proxies, which is not yet thoroughly studied and well understood. HTTP queries from clients would be ultimately handled by the requested web servers. However, if intermediate transparent proxies intercept the queries but understand/process the requests differently from the web servers, the responses could be different from the desired results, which may cause potential risks. More importantly, such HTTP interceptions performed by transparent proxies are not authorized by users and are difficult to detect on the user's side, which leads to security and ethical concerns. Such proxies often lack proper maintenance (*e.g.*, equipped with outdated software), in comparison to those web servers of a well-known domain. Moreover, users' personal information may be exposed to rogue transparent proxy owners, thereby causing privacy leakage.

To this end, we develop novel techniques to detect transparent proxies on the Internet on a large scale. In particular, we scrutinize the transparent proxies that could be vulnerable to malicious attacks like various cache poisoning attacks. Our study investigates the magnitude of such problems, characterizes different aspects of transparent proxies, and assesses the impact on end-users. Furthermore, we provide insights into the mitigation of potential vulnerabilities.

**Challenges.** There are three main challenges that we need to address for systematically analyzing transparent proxies. (1) It is difficult to detect the presence of the transparent proxy because its IP address may only be visible to the backend servers, not the clients. In other words, we can only get transparent proxy IP addresses from the server side. (2)

Another challenge is to acquire clients belonging to different locations and Autonomous Systems (ASes) to perform large-scale measurements, which also should allow fine-tuning of the measurement parameters. A suitable vantage point platform is needed for providing comprehensive coverage. (3) To detect the caching effects of transparent proxies, we need to carefully choose the domain names because not all domain names will be cached by transparent proxies. The requested URLs should also be carefully crafted to avoid affecting normal users and web servers.

**Our approach.** To address these challenges, we design and develop a novel measurement methodology and apply it to a large-scale experiment. We utilize a residential proxy network based on `TCP SOCKS` which provides over 230,000 unique residential IP addresses across more than 200 countries. This comprehensive coverage allows us to understand transparent proxies from a worldwide point of view.

To verify the interception of transparent proxies, we deploy several web servers and domain names. Each vantage point is instructed to send HTTP requests to a list of domains and query non-existent files under and without our controls, but the destination IP address is our controlled server, *e.g.*, URL: `http://a.b.c.d/`*`UUID`*`.[css|jpg]` and host: `example.com`, where `a.b.c.d` is our controlled server IP address. Since each requested file `UUID.[css|jpg]` is non-existent, it cannot be cached by transparent proxies when the first request is received. In addition, because of the non-existent requested file, it does not affect the other clients. To increase the success rate of caching detection, Alexa's [27] top 200 domain names are selected as the domain test list because of their popularity. We also added one of our controlled domain names to the domain test list to obtain the IP addresses of transparent proxies.

**Contributions.** The major contributions of this work are summarized below:

- Understanding: We systematically measure HTTP interceptions by transparent proxies and investigate their scale, behaviors, and security implications.
- Methodology: We design novel approaches to conduct a large-scale analysis to characterize HTTP interception through 951,877 residential IP addresses worldwide.
- Findings: We identify that thousands of transparent proxies perform forced DNS resolution to intercept HTTP traffic and are vulnerable to cache poisoning attacks. Besides the vulnerability we examined, transparent proxies are also vulnerable to other malicious attacks, such as CPDoS (Cache Poisoned Denial of Service).

## II. Background and Threat Model

In this section, we begin with an overview of how transparent proxies intercept HTTP requests. Then, we present our threat model, which focuses on vulnerable transparent proxies.

### A. HTTP and Transparent Proxy

**HTTP.** HTTP is an Internet protocol enabling data transfer between a client and a web server. Clients make resource requests via a Uniform Resource Locator (URL) such as '`http://www.example.com/sample.css`', or an IP address, like '`http://127.0.0.1/sample.css`'.

Upon receipt of a request, the server selects an appropriate variant of the resource – known as a representation – to return to the client. The selection process is guided by HTTP headers, fields that contain additional context and metadata about the request or response.

For example, a client may specify preferred media formats through headers, and the server may indicate the media format of the returned resource. The `Host` request header is particularly significant as it identifies the server's host and port number. If the port number isn't explicitly provided, default values (*e.g.*, 80 for HTTP, 443 for HTTPS) are used.

**Transparent Proxy.** A transparent proxy, also known as an intercepting proxy, inline proxy, or forced proxy, intercepts normal application-layer communication without requiring any special client configuration. The existence of the proxy need not be made known to clients as it is typically located between the client and the Internet, performing some of the functions of a gateway or router.

Intercepting proxies are often used in businesses to enforce acceptable use policies and to reduce administrative overhead as no client browser configuration is required.

ISPs in some countries use intercepting proxies to save upstream bandwidth and improve customer response times by caching [2], [3], [28], [29]. This is especially prevalent in countries where bandwidth is limited or must be paid for.

### B. CPDoS: Cache Poisoned Denial of Service

In this study, we also studied transparent proxies that could be exploited by adversaries. In particular, we identified that many proxies are vulnerable to the CPDoS (Cache Poisoned Denial of Service) attacks [24], a new category of web cache poisoning attacks aimed at disabling web resources.

CPDoS attacks operate in five steps: 1) An attacker sends an HTTP request containing a malicious header toward a resource hosted on web servers. This request passes through an intermediate cache with the malicious header remaining unnoticed. 2) The cache forwards the request to the origin server if it lacks a fresh copy of the targeted resource. Upon arrival, the origin server encounters an error while processing the request due to the embedded malicious header. 3) Consequently, the origin server returns an error page that the cache stores in place of the requested resource. 4) The attacker can then ascertain the success of their attack by retrieving the cached error page from the cache. 5) Subsequent requests from legitimate users for the target resource result in the cache serving the stored error page instead of the original content.

CPDoS attacks can be exploited through various vectors of HTTP header semantics, including HTTP Header Oversize (HHO), HTTP Meta Character (HMC), and HTTP Method

Override (HMO). HHO involves sending oversized headers that exceed cache limits, causing error messages. HMC entails sending HTTP headers carrying harmful meta-characters like line break (`\n`), carriage return (`\r`), or bell (`\a`). HMO entails sending headers that override methods such as DELETE and PUT, which are typically prohibited by web servers. In this study, we employ these three CPDoS attack vectors to scrutinize transparent proxies.

### *C. Threat Model*

Our threat model is shown in Figure 1 and Figure 2. We assume that transparent proxies are present on the path and can intercept the HTTP requests that are originally sent to the web servers. The transparent proxies forward the requests based on their own HTTP understanding and configuration (*e.g.*, forward requests based on IP addresses in original requests or forward requests based on the IP address from their forced DNS resolutions using the domain name in the host header). After the responses are received by transparent proxies, the transparent proxies send the responses to the clients. To this end, from a client's perspective, HTTP responses appear to come from the original web servers, making the actual interception behaviors difficult to discern.

Transparent proxies handle HTTP requests differently, which may result in clients receiving responses from different web servers. In Figure 1, if transparent proxies simply forward the client's request, the request will reach the destination specified in the HTTP request (*i.e.*, Server A). However, in this study, we identify that many transparent web proxies would perform their own DNS resolution and replace the destination IP address based on the resolution answer, resulting in the request being sent to a different server (*i.e.*, Server B). We refer such case as *F*orced *D*NS *R*esolution (FDR) in this work.

Moreover, transparent proxies usually cache frequently used static content such as HTML, images, or CSS files, which can cause significant issues. As shown in Figure 2, attackers can inject content into transparent proxies and deceive them into caching the malicious content. In doing so, attackers specify the victim domain name (hosted by Server A) in the Host header but craft the IP packet with the destination as a server (Server B) under the attacker's control. If the proxy does not perform a separate DNS resolution like the FDR case in Figure 1, the proxy will contact Server B to fetch the required content. As a result, such content offered by the attacker will be cached and then returned to subsequent clients who request the content originally hosted by Server A. Figure 4 illustrates the detailed request/response flow.

The objective of this study is to conduct large-scale data analysis to measure and characterize transparent proxies. During our research, we could inject selected content into transparent proxies and deceive them into caching the malicious content. If other users utilize the same transparent proxy and request the same content, the proxy may return our injected content from its cache. Therefore, we need to carefully manage our experiments to reduce the impact on other users. We discuss ethical considerations in Section III-E.

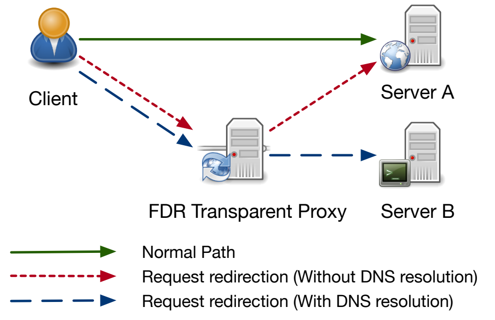
Client
Server A
FDR Transparent Proxy
Server B
Normal Path
Request redirection (Without DNS resolution)
Request redirection (With DNS resolution)

Fig. 1. HTTP interception caused by transparent proxy

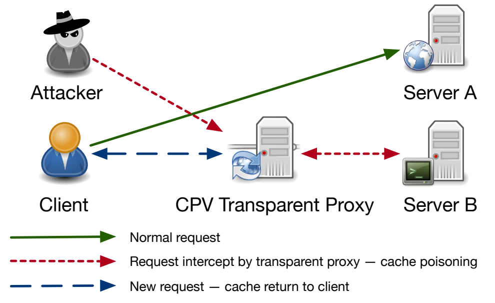
Attacker
Server A
Client
CPV Transparent Proxy
Server B
Normal request
Request intercept by transparent proxy — cache poisoning
New request — cache return to client

Fig. 2. Cache poisoning caused by transparent proxy

## III. Methodology and Data Collection

This section presents our proposed methodology and data collection, which can address the challenges outlined before. We begin by providing an overview of our approach and the design requirements. Next, we delve into the details of our measurement framework, as well as the workflow to identify vulnerable transparent proxies. Finally, we consider potential ethical issues.

### *A. Overview*

First, we describe our methodology for detecting transparent proxies on the Internet in the wild and identifying potential interceptions by those proxies that are vulnerable to such attacks.

**Approach.** Transparent proxies are able to monitor and intercept HTTP requests and forward them to the web servers. During interception, transparent proxies parse and reconstruct HTTP request messages before sending them to the web server. Ordinarily, the Host header is mapped to the destination IP address, ensuring that requests reach the correct server. However, if the Host header is not correctly mapped to the destination IP address, the situation becomes complicated. When there are no transparent proxies, requests are directly sent to the correct server based on the destination IP address. However, if there is an on-path transparent proxy, it may forward requests to a web server that matches the domain name by performing DNS requests and redirecting them to the IP address of the requested domain based on its DNS resolution result (*i.e.*, an FDR case).

Figure 3 illustrates the request/response flow produced by such FDR transparent proxies. In packet ①, a client sends a request using Server A's IP address as the destination IP

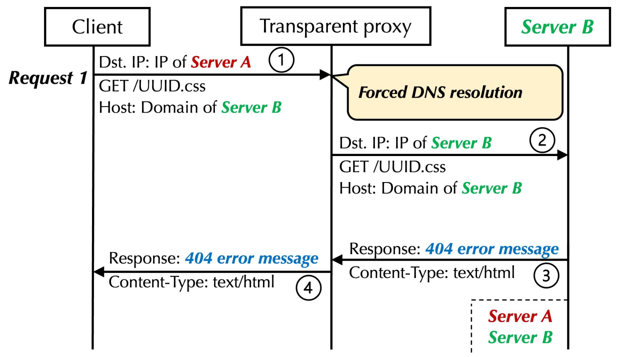
Client
Transparent proxy
Server B
Request 1
Dst. IP: IP of Server A ①
GET /UUID.css
Host: Domain of Server B
Forced DNS resolution
Dst. IP: IP of Server B ②
GET /UUID.css
Host: Domain of Server B
Response: 404 error message
Content-Type: text/html ③
Response: 404 error message
Content-Type: text/html ④
Server A
Server B

Fig. 3. Request/Response flow produced by FDR proxies

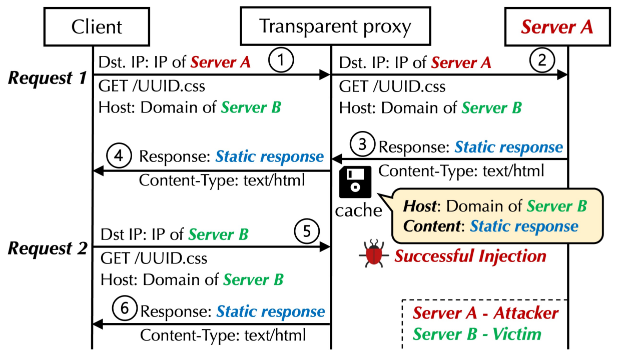

Fig. 4. Request/Response flow produced by CPV proxies

and Server B's domain as the Host header. The transparent proxy performs DNS resolution and changes the destination IP to Server B's IP address in packet ②. Server B's content is returned to the client through packets ③ and ④.

In addition, we discovered another type of transparent proxy that is more vulnerable. This type of proxy forwards requests based on the destination IP address and caches the responses using the Host header as the cache key, which is a unique identifier for each object in the cache. When there is a cache hit, the requested object is served to clients from the proxy's cache, which can be used to inject content into the cache. To test for this vulnerability, we send an HTTP request with our server as the destination IP address and a different domain name as the Host header to inject our content into the proxy's cache. Next, we send a second request with the earlier Host headers but a matching destination IP address to validate if we can receive the cached content for normal HTTP queries. If we receive the same responses as before, we can infer the presence of a transparent proxy with cache poison vulnerability. We refer to this type of transparent proxy as `CPV` – transparent proxy with Cache Poison Vulnerability.

Note that we could inject our contents into any web server using transparent proxies by changing the Host headers to any domain name. The request/response flows produced by CPV transparent proxies are shown in Figure 4. Packet ① shows a client sending an HTTP request to the transparent proxy. The request includes Server A's IP address as the destination IP and Server B's domain as the Host header. The transparent proxy forwards this request to Server A in ②. Server A returns a response to the transparent proxy, which is then stored in the proxy's cache. Packet ④ shows the transparent proxy returning the cached content to the original client. Later, another client sends a request with Server B's IP address and domain name, as shown in packet ⑤. The transparent proxy recognizes the request as a cache hit since it matches the cached object's cache key (which is based on the Host header). The proxy returns the cached content to the client using packet ⑥.

Given these two types of transparent proxies, we perform probing tests to identify their presence by the following steps:

*1).* Instruct a client to send an HTTP request with the victim server's Host header and our controlled attacker server's IP address as the destination IP address. We record the corresponding responses on the client side and both the attacker and victim server side.

*2).* If the request reaches the victim server, we compare the IP address in the victim server's log with the client's IP address. If the two IP addresses are not the same, we classify it as `FDR` – a transparent proxy with forced DNS resolution.

*3).* Instruct a client to send an HTTP request with the victim server's Host header to the victim server.

*4).* If the client receives the attacker server's content, we compare the IP address in the attacker server's log with the client's IP address. If the two IP addresses differ, we classify it as `CPV` – a transparent proxy with cache poisoning vulnerability.

**Design requirements.** Our methodology must meet several requirements to ensure valid results. First, to avoid caching, each request from the client must query a different resource. Second, since we capture packets separately from clients and web servers, we must be able to correlate a request from a client with one captured by our web server. We address this issue by uniquely prefixing each requested file name. Third, our study requires diverse clients capable of sending HTTP packets directly to the specified web servers with the specified domain name. Fourth, to study interception characteristics in-depth, the vantage points must issue diversified HTTP requests, including requests of different methods and headers. The measurement infrastructure used by previous works, including advertising networks, HTTP proxy networks, and Internet scanners, does not meet the requirements.

## B. Methodology

*1) Experiment Setup:* To perform the experiments to identify the transparent proxies, we arranged two distinct servers: Server A, a reference web server under our control, and Server B, a victim server that may or may not be under our control. Server A is configured to consistently deliver static text content in response to any request, irrespective of whether the requested host and file were found or not. This operation aims to deceive transparent proxies into caching this content.

As part of the experiments, we initiated HTTP requests to our controlled servers as illustrated in Figure 3 and 4. If the transparent web proxies cached the content and primarily used the Host header as the cache key instead of the IP address, clients might receive content from the attacker's server rather than the victim's. However, this was not our desired outcome. To boost the chance of caching, we set Server A to emit Cache-Control headers, marking the content as cacheable for an extended duration.

On the other hand, Server B acts as a typical web server, returning standard responses. In cases where the requested host and files were not identified, Server B responded with corresponding error messages. This server could either be one within our control or a web server beyond our management. It is noteworthy that transparent proxies have specific policies and configurations regarding domain selection for caching, and higher-traffic websites have a greater likelihood of being cached. To accommodate this factor, we used domain names from Alexa's top 200 list, and a domain name under our control for Server B.

*2) Generating HTTP Requests:* In this study, we address the issue of inconsistent source IP addresses between a client's original request and the corresponding request relayed by the transparent proxy. We tackle this by linking these requests by creating a unique file name that includes a distinct UUID (universally unique identifier) generated for each client and a file extension such as CSS or JPG.

To conduct our experiments, we constructed two types of HTTP requests. For the first type of request, the destination IP is set as our controlled server's address (*i.e.*, the attacker) and the control server would reply with a static response to all HTTP requests. The Host field is set to the domain of a different web server (*i.e.*, the victim server) and the Path field is set to the UUID + file extension (*e.g.*, `[UUID].css`). The UUID labels requests and vantage points (clients), while the file extension indicates the file types that can be cached by proxies. One example request is

```
HTTP Query:
  DST IP:  IP address of the Server A
  Host:    Domain name of the Server B
  Path:    /UUID.css
```

For the second request, the host is the victim server's domain name, and the destination IP address is changed to the victim server's IP address. The path is the same as in the first request: UUID + file extension. The UUID is identical to that used in the first request, while the file extension remains the same as in the first request. One example request is

```
HTTP Query:
  DST IP:  IP address of the Server B
  Host:    Domain name of the Server B
  Path:    /UUID.css
```

## *C. Vantage Points*

Our study requires a large number of globally distributed clients capable of sending customized HTTP requests. To achieve this, we leveraged a residential proxy network called ProxyRack [30] which is based on TCP SOCKS.

SOCKS proxy networks allow us to send HTTP packets directly from globally distributed clients, providing a global landscape of distributed transparent proxies. When our requests are sent by our client, an entry node receives and forwards the requests to exit nodes distributed across the world. The exit nodes, which serve as our vantage points, will be responsible for sending requests to the servers and relay the responses back to the client.

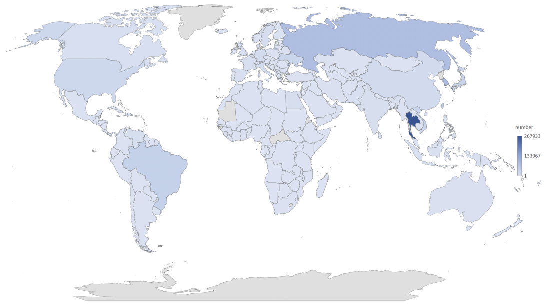

Fig. 5. Geo-distribution of vantage points

As we can only interact with the SOCKS proxy platform, we cannot directly obtain the IP addresses of the vantage points (*i.e.*, the exit nodes). In this study, we use an indirect method to obtain the vantage point IP addresses. Before each test, we send a request to the IP geolocation service (*e.g.*, IP-API.com [31]) through ProxyRack, which will typically return a response including the query IP address (vantage point IP), its geolocation, AS number, and other information.

## *D. Data collection and Dataset*

Our dataset, as summarized in Table I, consists of HTTP traffic collected from 951,877 residential IP addresses globally.

**Format of dataset.** To perform the correlation analysis and identify the existence of transparent proxies on the path, we issue HTTP requests from clients, monitor web server logs, and capture HTTP requests. For each HTTP request, we store the collected data in a JSON format which includes the source IP address, timestamp, method, URL, headers, user agent, and requested domain name at our controlled web servers.

**Geo-distribution of clients.** Using the ProxyRack proxy network, we are able to obtain vantage points globally. The geo-distribution of distinct IPs provides an evaluation of our clients. Our collected clients span more than 951,877 unique addresses across 205 countries and 10,145 ASes. As shown in Figure 5, our clients cover most countries in the world, with Thailand, South Korea, Russia, Japan, and the United States having the highest numbers of clients.

## *E. Ethical Considerations*

In designing and conducting our study, we have taken ethical considerations seriously to protect users from potential side effects that may arise from our experiments.

The residential proxy network, ProxyRack, which we utilized in this study, is a commercial service that invites users to participate in the business for profit. The owners of exit nodes have an agreement with ProxyRack that authorizes ProxyRack traffic to exit from their hosts. Therefore, launching

TABLE I
STATISTICS OF COLLECTED DATASET

| # IP | # AS | # ISP | # /24 pref | # /16 pref | # country |
|---|---|---|---|---|---|
| 951,877 | 10,145 | 14,657 | 320,444 | 22,672 | 205 |

TABLE II
DISTRIBUTION OF IDENTIFIED FDR AND CPV PROXIES

| | FDR | CPV |
|---|---|---|
| IP | 32,246 | 11,286 |
| AS | 1,458 | 226 |
| ISP | 2,018 | 257 |
| Countries | 98 | 51 |
| /24 prefix | 21,437 | 2,542 |
| /16 prefix | 3,690 | 474 |

HTTP requests from ProxyRack complies with the permission granted by the exit node owners.

We also carefully crafted our HTTP requests and restricted their quantity to prevent excessive network traffic. In our experiments, traffic goes to the victim web server only when FDR transparent proxies exist. Otherwise, most traffic generated by our tests is directed to our controlled web servers. Additionally, our controlled attacker server returns only static contents that are harmless. We use UUID as the requested file name to avoid interfering with other users who share the same transparent proxy.

By adopting these approaches, we believe we have largely mitigated potential impacts on users' privacy and security in our experiments.

## IV. Transparent Proxy Interception Analysis

In this study, we aim to conduct a global measurement of transparent proxies. We present our measurement results and analysis, showcasing the landscape and characteristics of transparent proxies.

### A. Scope and Magnitude

We conducted our global-scale measurement from September 2021 to June 2022, performing a total of 1,401,567 scans. During these scans, we identified 114,310 instances of `FDR` transparent proxies with Forced DNS Resolution and 29,971 instances of `CPV` transparent proxies with Cache Poison Vulnerability. We found a total of 32,246 IPs vulnerable to FDR transparent proxies in 1,458 ASes, and 226 ASes are vulnerable to CPV transparent proxies. For ISPs, we found 2,018 ISPs vulnerable to FDR and 257 ISPs vulnerable to CPV. Furthermore, we identified 98 countries vulnerable to FDR and 51 countries vulnerable to CPV. Additional details are presented in Table II. Based on these findings, we observed more instances of FDR cases than CPV cases, and FDR cases are more widely distributed than CPV cases.

### B. AS-level Analysis

We observed FDR cases in 1,458 ASes globally. The statistics of AS distribution indicate that FDR transparent proxies are spread across many ASes. However, the distribution is also highly imbalanced, with most of the observed cases located in only a few ASes. This is reflected in the long-tail distribution of ASes, where 681 (46.7%) of ASes have only one observed transparent proxy.

TABLE III
TOP 10 ASES THAT HAVE THE MOST FDR TRANSPARENT PROXIES

| AS number | Organization | #IP |
|---|---|---|
| AS9318 | SK Broadband Co Ltd | 7,833 |
| AS17552 | True Online | 3,586 |
| AS45758 | Triple T Internet | 1,850 |
| AS45629 | JasTel Network International Gateway | 1,417 |
| AS4766 | Korea Telecom | 758 |
| AS45758 | Triple T Broadband | 758 |
| AS58224 | Iran Telecommunication Company PJS | 631 |
| AS44208 | Farahoosh Dena PLC | 626 |
| AS25019 | Saudi Telecom Company JSC | 600 |
| AS17858 | LG POWERCOMM | 509 |

TABLE IV
TOP 10 COUNTRIES WHICH HAVE THE MOST OBSERVED FDR TRANSPARENT PROXIES

| Country | #IP |
|---|---|
| South Korea | 10,265 |
| Thailand | 8,942 |
| Iran | 4,137 |
| Russia | 2,924 |
| India | 1,447 |
| Bangladesh | 782 |
| Saudi Arabia | 775 |
| United Arab Emirates | 490 |
| Taiwan | 389 |
| China | 376 |

Table III shows the top 10 ASes that FDR transparent proxies belong to. AS9318 SK Broadband Co Ltd, AS17552 True Online, and AS45758 Triple T Internet/Triple T Broadband are the top three ASes with the most observed FDR transparent proxies. The distribution of ASes of FDR transparent proxies indicates that HTTP interception by transparent proxies is prevalent on the global Internet.

### C. Country-level Analysis

We observed FDR transparent proxies in 98 countries and regions globally, indicating that almost half of the countries are affected by this type of vulnerable transparent proxies. The wide distribution of vulnerable transparent proxies highlights that this vulnerability is a global security issue, not confined to a single region. Table IV presents the top 10 countries with the most observed FDR transparent proxies. The majority of the observed transparent proxies are in South Korea, Thailand, Iran, Russia, and India.

### D. Domain Selection Analysis

In our experiment, we selected 201 domain names to detect FDR transparent proxies, including Alexa top 200 domains and one of our controlled domains.

Table V presents the top domain names that trigger FDR transparent proxies. Interestingly, some of the top domain names are related to adult content, suggesting that transparent

TABLE V
TOP 10 DOMAIN NAMES WHICH TRIGGER FDR TRANSPARENT PROXIES

| Domain name | #Detection scans |
|---|---|
| xhamster.com | 116,660 |
| chaturbate.com | 113,063 |
| xnxx.com | 108,724 |
| bet365.com | 108,476 |
| bongacams.com | 108,040 |
| pornhub.com | 107,805 |
| xvideos.com | 101,129 |
| bet9ja.com | 22,896 |
| livejasmin.com | 18,670 |
| 6.cn | 12,412 |

TABLE VI
TOP 10 ASES THAT HAVE THE MOST OBSERVED CPV TRANSPARENT PROXIES

| AS number | Organization | #IP |
|---|---|---|
| AS45629 | JasTel Network International Gateway | 8,255 |
| AS45758 | Triple T Internet/Triple T Broadband | 1,739 |
| AS45758 | Triple T Broadband Public Company Limited | 596 |
| AS23969 | TOT Public Company Limited | 86 |
| AS4766 | Korea Telecom | 78 |
| AS30722 | Vodafone Italia S.p.A. | 58 |
| AS131090 | CAT TELECOM Public Company Ltd | 31 |
| AS133481 | AIS Fibre | 26 |
| AS4760 | HKT Limited | 21 |
| AS17552 | True Online | 21 |

proxy owners may use these domain names to censor and block such content.

## V. TRANSPARENT PROXY CACHE POISONING ANALYSIS

In this section, we analyze observed CPV transparent proxies in our measurements. In total, 11,017 IPs are identified to be exposed to the vulnerability of CPV transparent proxies.

### A. AS-level Analysis

We observed CPV transparent proxies in 226 ASes, showing that such a problem is prevalent. On the other hand, the distribution is very imbalanced with most of the observed cases located in only a few ASes. The distribution of ASes follows a long-tail distribution, with 149 (65.5%) of ASes having only one observed vulnerable transparent proxy. Table VI shows the top 10 AS that CPV transparent proxies belong to. AS45629 JasTel Network International Gateway, AS45758 Triple T Internet/Triple T Broadband, and AS23969 TOT Public Company Limited are the top three ASes with the most observed transparent proxies, and all these ASes are from Thailand. AS4766 Korea Telecom and AS30722 Vodafone Italia have also been observed with more than 50 transparent proxies. This means that clients in these ASes have a higher possibility of using those vulnerable CPV transparent proxies.

### B. Country-level Analysis

Our analysis revealed the presence of CPV transparent proxies in 51 countries and territories worldwide, indicating that approximately one-quarter of all countries are affected by this vulnerability. This widespread distribution underscores the fact that CPV transparent proxies constitute a global security issue, rather than a problem limited to certain regions.

TABLE VII
TOP 10 COUNTRIES WHICH HAVE MOST OBSERVED CPV TRANSPARENT PROXIES

| Country | #IP |
|---|---|
| Thailand | 10,772 |
| South Korea | 87 |
| Italy | 64 |
| Ukraine | 41 |
| Russia | 40 |
| Japan | 36 |
| Hong Kong | 28 |
| United States | 23 |
| Canada | 22 |
| Iran | 16 |

Table VII provides an overview of the top 10 countries with the highest number of observed CPV transparent proxies. Thailand has the highest percentage of CPV transparent proxies, accounting for 95.44% of all observed instances, followed by South Korea, Italy, Ukraine, and Russia, respectively.

### C. Domain Selection Analysis

TABLE VIII
TOP 10 DOMAIN NAMES WHICH TRIGGER VULNERABLE TRANSPARENT PROXIES

| Domain name | #Detection scans |
|---|---|
| netflix.com | 14,854 |
| spotify.com | 7,625 |
| speedtest.net | 2,374 |
| instagram.com | 1,708 |
| microsoft.com | 1,079 |
| vk.com | 875 |
| wordpress.com | 638 |
| ikea.com | 628 |
| tribunnews.com | 177 |
| csdn.net | 175 |

For our experiment, we selected 201 domain names to detect vulnerable transparent proxies. Table VIII displays the top domain names that triggered CPV transparent proxies during our analysis. Notably, Netflix.com was the most commonly affected domain, followed by Spotify.com, Speedtest.net, Instagram.com, and Microsoft.com.

Several of these websites offer audio/video streaming or file-sharing services, which typically generate significant amounts of Internet traffic. Thus, we speculate that ISPs may configure transparent proxies to cache the content of these domains to conserve bandwidth, decrease traffic, and reduce costs.

### D. Transparent Proxy Server Analysis

To gather more information on transparent web proxies, we utilized Nmap [32] to conduct scans and collect data. In the following sections, we will present the data we collected on the transparent proxy servers, including information on the operating system, services, and products (device information).

**Operating Systems.** Our analysis revealed the presence of 139 different operating systems among the transparent proxies we scanned. Table IX displays the top 10 most common

TABLE IX
TOP 10 OPERATING SYSTEMS OF TRANSPARENT PROXY SERVERS

| Operating System | # of IP |
|---|---|
| HP P2000 G3 NAS device | 419 |
| MikroTik RouterOS 6.36 | 227 |
| Linux 2.6.18 - 2.6.22 | 94 |
| OpenWrt Kamikaze 7.09 (Linux 2.6.22) | 75 |
| Linux 3.10 - 4.11 | 62 |
| Fortinet FortiGate 100D firewall | 41 |
| ProVision-ISR security DVR | 41 |
| Linux 2.6.32 - 3.13 | 32 |
| OpenWrt 0.9 - 7.09 (Linux 2.4.30 - 2.4.34) | 25 |
| Crestron XPanel control system | 24 |

TABLE X
SERVICE AND PRODUCTS OF CPV TRANSPARENT PROXY SERVERS

| Service | # IP | Product | # IP |
|---|---|---|---|
| http | 723 | MikroTik bandwidth-test server | 289 |
| domain | 294 | MikroTik router config httpd | 148 |
| bandwidth-test | 289 | MikroTik | 135 |
| unknown | 232 | Hikvision IPCam control port | 123 |
| tcpwrapped | 195 | Huawei Home Gateway telnetd | 104 |
| rtsp | 191 | Apache httpd | 101 |
| telnet | 184 | Apple AirTunes rtspd | 93 |
| pptp | 167 | Hikvision Network Video Recorder | 84 |
| ssh | 128 | Dropbear sshd | 73 |
| ipcam | 123 | nginx | 55 |

operating systems, with the HP P2000 G3 NAS device OS being the most prevalent.

Notably, Linux and Microsoft Windows emerged as the most widely used operating systems among transparent proxy servers. However, a significant number of these systems were found to be outdated and vulnerable to attacks. These vulnerabilities, such as those documented in CVE (common vulnerabilities and exposures), can be exploited by attackers to compromise the security of transparent proxies.

**Services and Devices.** Although we were only able to identify a limited number of transparent proxies, we were still able to gather some valuable information through our Nmap scans. Detecting service and product information for ISP transparent proxies can be challenging, but we were able to identify some client-side transparent proxies.

Our analysis revealed that HTTP was the most common service utilized by transparent proxy servers. Additionally, we found that a set of products from MikroTik, Huawei, Apple, and Hikvision demonstrate the behavior of CPV transparent proxies, as shown in Table X.

## *E. Case Study: CPDoS Attacks on Transparent Proxy*

**CPDoS detection methodology.** In our study, we designate a server as the target server to receive requests. When the target server receives CPDoS requests, it returns default error messages. For normal requests, the server returns designed static content. During our experiments, we sent a pair of requests - one for a CPDoS attack and the other for a normal request - and compared the two responses for each pair. We labeled a CPDoS attack as successful if the first response matched the second response and both responses matched the default error message. To determine whether these successful attacks were caused by transparent proxies, we compared the IP address of the vantage point with the IP address in the Apache log of the target server. If these IP addresses were different, we concluded that a transparent proxy caused the successful attack.

**Result.** In this study, we identified two types of CPDoS attacks on transparent proxies: HMC and HHO. Our analysis revealed 434 HMC cases and 32 HHO cases in the transparent proxy study. Our findings suggest that transparent proxies are susceptible to CPDoS vulnerabilities, and therefore, transparent proxy owners should take immediate measures to mitigate such vulnerabilities.

## *F. Summary of Findings*

Our measurement findings in the global analysis are summarized below.

- A large number of transparent proxies are performing potentially harmful HTTP interceptions. More seriously, thousands of transparent proxies are vulnerable to cache-poisoning attacks.
- HTTP interceptions are distributed globally, and we find FDR transparent proxies in 1,458 ASes and 98 countries.
- CPV transparent proxies are found to exist in 51 countries and 226 ASes.
- CPV transparent proxies may cause serious damage. Damage might be significant if attackers target popular websites and the vulnerable transparent proxies serve many clients.
- Transparent proxies are also vulnerable to other attacks, such as CPDoS (Cache Poisoned Denial of Service). We identified 434 HMC and 32 HHO cases in our study.

## VI. DISCUSSION

## *A. Security Implications*

A transparent proxy is difficult to be detected on the client side, and thus Internet users might not realize their traffic is intercepted. First, when HTTP requests from clients are handled by transparent proxies, it is possible to monetize illegally from the traffic. Second, as it is difficult for Internet users to detect transparent proxies merely from clients, requested websites could be wrongly blamed when undesired results (e.g., advertisement sites or even malware) are returned. Third, `CPV` brings severe cache poisoning vulnerability. Attackers might utilize such a vulnerability to inject designed content into transparent proxies. Other clients who share the same transparent proxies may also not get the original content. Moreover, if attackers inject content similar to an online bank or other financial websites, it may cause significant financial damage to clients. Finally, intercepted HTTP requests could be snooped on by untrusted third parties, leading to the leak of private data. Therefore, we believe that transparent proxies potentially induce ethical, privacy, and security risks to Internet users.

### B. Mitigation

In our study, we observe a significant vulnerability in the current state of HTTP traffic, as the majority of HTTP packets are sent unencrypted, making them susceptible to snooping and manipulation. Recognizing this issue, the Internet community has taken steps to address it, resulting in the release of RFC 2818 [33], which outlines the HTTP specification over Transport Layer Security (TLS). By employing HTTPS, websites can authenticate their identity, protect the privacy and integrity of data exchanged during transit, guard against man-in-the-middle attacks, and ensure bidirectional encryption to prevent eavesdropping and tampering. We strongly recommend the use of HTTPS over HTTP to safeguard against potential interceptions.

However, despite the availability of modern browsers offering HTTPS-Only mode for secure browsing (*e.g.*, [34], [35]), HTTP is still predominantly allowed on HTTPS-enabled websites [36], and outdated browsers remain prevalent [37]. Additionally, modern web services heavily rely on third-party libraries and services, resulting in HTTPS webpages issuing additional HTTP requests to acquire resources. Therefore, we believe that the issues associated with HTTP are still of critical importance and warrant attention.

Furthermore, according to RFC 2616 [38], transparent proxies should not modify requests or responses beyond what is required for proxy authentication and identification. However, our study reveals that a significant number of transparent proxies deviate from this standard. These proxies perform DNS resolutions to obtain the destination IP address but ignore the destination IP address in the request. Such behavior can lead to substantial damage to both clients and servers. Transparent proxy administrators should exercise great caution when configuring the proxy server to avoid unintended consequences. Moreover, we have identified that many transparent proxies operate on outdated operating systems and software, exposing them to vulnerabilities and exploits such as Common Vulnerabilities and Exposures (CVE). This makes it easier for attackers to exploit the proxy server. Transparent proxy owners should prioritize keeping the operating systems and software up-to-date to mitigate such attacks.

## VII. Related Work

Xu *et al.* [2] conducted an analysis of transparent proxy behavior and their interaction with HTTP traffic in four major US cell carriers. They observed that all four carriers employed transparent proxies to interpose on HTTP traffic, but noted variations in their behaviors. In our study, we specifically focus on residential transparent proxies.

Zhang *et al.* [3] investigated the manipulation of HTTP traffic by transparent proxies. Our work extends this research by performing a global measurement study on transparent proxies, providing a broader perspective on their prevalence and behavior.

Fanou *et al.* [28] explored the web infrastructure in Africa and conducted a mapping of middleboxes in the region.

Mi *et al.* [25] conducted a study on residential proxy (RESIP) ecosystems, focusing on the measurement and testing of 6 million RESIP IPs. Their research identified potential security issues, such as the presence of potentially unwanted programs. Similarly, Yang *et al.* [26] conducted an extensive study on the Chinese RESIP ecosystem, providing a larger dataset and additional insights based on their findings. It is worth noting that our work primarily focuses on transparent proxies, which distinguishes it from the topics covered in these prior papers.

Nguyen *et al.* [24] introduced and analyzed a new class of web cache poisoning attacks known as Cache Poisoned DoS (CPDoS) attacks. In our work, we explore the vulnerability of transparent proxies to CPDoS attacks.

Mirheidari *et al.* [39] conducted a large-scale study on web cache deception (WCD), where attackers trick caching proxies into erroneously storing private information transmitted over the Internet, subsequently gaining unauthorized access to the cached data. Additionally, Mirheidari *et al.* [40] proposed a novel WCD detection methodology applicable to any website, expanding our understanding of WCD attacks, their spread, and their implications. Tyson *et al.* [41] investigated HTTP header manipulation by proxies and middleboxes, analyzing the factors influencing header manipulation. Chung *et al.* [42] detected end-to-end violations of DNS, HTTP, and HTTPS through a paid residential proxy service, revealing that up to 4.8% of nodes were subject to such violations.

Nguyen *et al.* [43] developed a cache testing environment for analyzing shared caches and evaluated seven different shared caching systems. However, our work focuses on identifying real-world cache security issues and analyzing their actual impact on the modern global Internet.

## VIII. Conclusion

In this paper, we perform a large-scale study on HTTP interceptions by transparent proxies, which induce security, privacy, and performance issues. We develop a set of techniques to detect the stealthy behaviors of transparent proxies by utilizing a well-maintained proxy platform with numerous vantage points. Based on our collected dataset, we observe that HTTP interceptions by transparent proxies exist in 1,458 ASes and 98 countries. In addition, the interception characteristics are further analyzed. Our analysis results indicate that stealthy HTTP interceptions by transparent proxies can potentially introduce new threats in the web ecosystem. Furthermore, we investigate the security problems around transparent proxies, such as caching poisoning attacks and CPDoS. We uncover cache-poisoning-prone transparent proxies in 226 ASes and 51 countries. For CPDoS, we identify 434 HMC and 32 HHO cases in our study. Ultimately, we analyze the threats caused by transparent proxies and discuss mitigation solutions.

## Acknowledgements

We thank the anonymous reviewers for their insightful comments, which help to improve the quality of this paper. This work is supported in part by NSF grant CNS-2317829.

## References

[1] R. Crandell, J. Clifford, and A. Kent, "A Secure and Transparent Firewall Web Proxy," in *LISA*, 2003.

[2] X. Xu, Y. Jiang, T. Flach, E. Katz-Bassett, D. Choffnes, and R. Govindan, "Investigating Transparent Web Proxies in Cellular Networks," in *Passive and Active Network Measurement (PAM)*, 2015.

[3] M. Zhang, B. Liu, C. Lu, J. Zhang, S. Hao, and H. Duan, "Measuring Privacy Threats in China-Wide Mobile Networks," in *8th USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2018.

[4] X. Jia, D. Li, H. Du, and J. Cao, "On Optimal Replication of Data Object at Hierarchical and Transparent Web Proxies," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 8, 2005.

[5] T.-c. Chiueh, H. Sankaran, and A. Neogi, "Spout: a Transparent Proxy for Safe Execution of Java Applets," *IEEE Journal on Selected Areas in Communications*, vol. 20, no. 7, 2002.

[6] A. Luotonen, *Web Proxy Servers*. Prentice-Hall, Inc., 1998.

[7] R. Bian, S. Hao, H. Wang, and C. Cotton, "Shining a Light on Dark Places: A Comprehensive Analysis of Open Proxy Ecosystem," *Computer Networks*, vol. 208, 2022.

[8] R. Bian, "Using stand-off observation and measurement to understand aspects of the global internet," Ph.D. dissertation, University of Delaware, 2022.

[9] W. Jin and D. Kim, "Development of Virtual Resource based IoT Proxy for Bridging Heterogeneous Web Services in IoT Networks," *Sensors*, vol. 18, no. 6, 2018.

[10] N. Weaver, C. Kreibich, M. Dam, and V. Paxson, "Here Be Web Proxies," in *Passive and Active Measurement (PAM)*, 2014.

[11] 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 (S&P)*, 2019.

[12] J. Choi, M. Abuhamad, A. Abusnaina, A. Anwar, S. Alshamrani, J. Park, D. Nyang, and D. Mohaisen, "Understanding the Proxy Ecosystem: A Comparative Analysis of Residential and Open Proxies on the Internet," *IEEE Access*, vol. 8, 2020.

[13] A. Tosun, M. De Donno, N. Dragoni, and X. Fafoutis, "ResIP Host Detection: Identification of Malicious Residential IP Proxy Flows," in *2021 IEEE International Conference on Consumer Electronics (ICCE)*, 2021.

[14] R. Bian, S. Hao, H. Wang, A. Dhamdere, A. Dainotti, and C. Cotton, "Towards Passive Analysis of Anycast in Global Routing: Unintended Impact of Remote Peering," *ACM SIGCOMM Computer Communication Review*, vol. 49, no. 3, 2019.

[15] N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, and D. Karger, "Infranet: Circumventing Web Censorship and Surveillance," in *Proceedings of the 11th USENIX Security Symposium*, 2002.

[16] L. Jin, S. Hao, H. Wang, and C. Cotton, "Understanding the Practices of Global Censorship through Accurate, End-to-End Measurements," *Proceedings of the ACM on Measurement and Analysis of Computing Systems (SIGMETRICS)*, 2022.

[17] A. Chaabane, T. Chen, M. Cunche, E. De Cristofaro, A. Friedman, and M. A. Kaafar, "Censorship in the Wild: Analyzing Internet Filtering in Syria," in *Proceedings of the 2014 ACM Conference on Internet Measurement Conference (IMC)*, 2014.

[18] T. K. Yadav, A. Sinha, D. Gosain, P. K. Sharma, and S. Chakravarty, "Where the Light Gets in: Analyzing Web Censorship Mechanisms in India," in *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2018.

[19] S. Chen, H. Wang, X. Zhang, B. Shen, and S. Wee, "Segment-based Proxy Caching for Internet Streaming Media Delivery," *IEEE MultiMedia*, vol. 12, no. 3, 2005.

[20] M. Xie, I. Widjaja, and H. Wang, "Enhancing Cache Robustness for Content-centric Networking," in *Proceedings of IEEE INFOCOM*, 2012.

[21] R. Caceres, F. Douglis, A. Feldmann, G. Glass, and M. Rabinovich, "Web Proxy Caching: The Devil is in the Details," *ACM SIGMETRICS Performance Evaluation Review*, vol. 26, no. 3, 1998.

[22] I. Cooper and J. Dilley, "Known http proxy/caching problems," Tech. Rep., 2001.

[23] M. Arlitt, L. Cherkasova, J. Dilley, R. Friedrich, and T. Jin, "Evaluating Content Management Techniques for Web Proxy Caches," *ACM SIGMETRICS Performance Evaluation Review*, vol. 27, no. 4, 2000.

[24] H. V. Nguyen, L. L. Iacono, and H. Federrath, "Your Cache Has Fallen: Cache-poisoned Denial-of-Service Attack," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2019.

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

[26] M. Yang, Y. Yu, X. Mi, S. Tang, S. Guo, Y. Li, X. Zheng, and H. Duan, "An Extensive Study of Residential Proxies in China," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2022.

[27] Amazon, "Alexa top domain list," https://www.alexa.com/.

[28] R. Fanou, G. Tyson, E. L. Fernandes, P. Francois, F. Valera, and A. Sathiaseelan, "Exploring and Analysing the African Web Ecosystem," *ACM Transactions on the Web (TWEB)*, vol. 12, no. 4, 2018.

[29] R. Baumgart and Y.-Y. Chan, "On Privacy Issues of Internet Access Services via Proxy Servers," in *Proceedings of International Exhibition and Congress on Secure Networking*, 1999.

[30] ProxyRack, https://www.proxyrack.com/, 2022.

[31] "IP-API," https://ip-api.com/.

[32] Nmap, "Nmap: the Network Mapper - Free Security Scanner," https://nmap.org/.

[33] E. Rescorla, "RFC2818: HTTP Over TLS," 2000.

[34] C. Blog, "A Safer Default for Navigation: HTTPS," https://blog.chromium.org/2021/03/a-safer-default-for-navigation-https.html.

[35] C. Kerschbaumer, J. Gaibler, A. Edelstein, and T. van der Merwe, "HTTPS-Only: Upgrading All Connections to HTTPS in Web Browsers," in *Proceedings of the Workshop on Measurements, Attacks, and Defenses for the Web*, 2021.

[36] K. Shen, J. Lu, Y. Yang, J. Chen, M. Zhang, H. Duan, J. Zhang, and X. Zheng, "HDiff: A Semi-automatic Framework for Discovering Semantic Gap Attack in HTTP Implementations," in *Proceedings of the 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2022.

[37] M. Musch, R. Kirchner, M. Boll, and M. Johns, "Server-Side Browsers: Exploring the Web's Hidden Attack Surface," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security (AsiaCCS)*, 2022.

[38] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "RFC2616: Hypertext Transfer Protocol–HTTP/1.1," 1999.

[39] S. A. Mirheidari, S. Arshad, K. Onarlioglu, B. Crispo, E. Kirda, and W. Robertson, "Cached and Confused: Web Cache Deception in the Wild," in *Proceedings of the 29th USENIX Security Symposium*, 2020.

[40] S. A. Mirheidari, M. Golinelli, K. Onarlioglu, E. Kirda, and B. Crispo, "Web Cache Deception Escalates," in *Proceedings of the 31st USENIX Security Symposium*, 2022.

[41] G. Tyson, S. Huang, F. Cuadrado, I. Castro, V. C. Perta, A. Sathiaseelan, and S. Uhlig, "Exploring Http Header Manipulation In-the-Wild," in *Proceedings of the 26th International Conference on World Wide Web (WWW)*, 2017.

[42] T. Chung, D. Choffnes, and A. Mislove, "Tunneling for Transparency: A Large-scale Analysis of End-to-End Violations in the Internet," in *Proceedings of the 2016 ACM Internet Measurement Conference (IMC)*, 2016.

[43] H. V. Nguyen, L. L. Iacono, and H. Federrath, "Mind the Cache: Large-scale Explorative Study of Web Caching," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, 2019.