# Tunneling for Transparency: A Large-Scale Analysis of End-to-End Violations in the Internet

Taejoong Chung
Northeastern University

David Choffnes
Northeastern University

Alan Mislove
Northeastern University

## ABSTRACT

Detecting violations of application-level end-to-end connectivity on the Internet is of significant interest to researchers and end users; recent studies have revealed cases of HTTP ad injection and HTTPS man-in-the-middle attacks. Unfortunately, detecting such end-to-end violations at scale remains difficult, as it generally requires having the cooperation of many nodes spread across the globe. Most successful approaches have relied either on dedicated hardware, user-installed software, or privileged access to a popular web site.

In this paper, we present an alternate approach for detecting end-to-end violations based on *Luminati*, a HTTP/S proxy service that routes traffic through millions of end hosts. We develop measurement techniques that allow Luminati to be used to detect end-to-end violations of DNS, HTTP, and HTTPS, and, in many cases, enable us to identify the culprit. We present results from over 1.2M nodes across 14K ASes in 172 countries, finding that up to 4.8% of nodes are subject to some type of end-to-end connectivity violation. Finally, we are able to use Luminati to identify and measure the incidence of content monitoring, where end-host software or ISP middleboxes record users' HTTP requests and later re-download the content to third-party servers.

## 1. INTRODUCTION

End-user applications typically make the implicit assumption of *end-to-end connectivity* when using the network: that application-level data is delivered to the destination application in unmodified form. However, this assumption is often violated, either due to other software running on the end-host (e.g., malware, ad injectors) or specialized appliances deployed by ISPs (e.g., middleboxes). Malware often modifies content to steal user credentials or inject advertisements; middleboxes are deployed by ISPs for a variety of reasons, including security (e.g., firewalls, anti-virus systems), content policies (e.g., content blockers), and performance optimization (e.g., proxies, DNS interception boxes, transcoders).

While the implementation and impact on users varies widely, all of these cases represent violations of end-to-end principles in Internet connectivity. These pieces of software and middleboxes are especially concerning when they monitor and manipulate users' traffic, such as when they intercept and replace DNS `NXDOMAIN` responses with advertisements or inject tracking JavaScript code into web pages. Making this situation more dire is the fact that these end-to-end violations are often opaque to users; ISPs typically do not announce the presence or function of middleboxes, nor do they usually declare how traffic is monitored/manipulated.[1]

Unfortunately, it is challenging to understand end-to-end connectivity violations without access to devices or users in affected networks. To address this challenge, there have been a number of successful prior approaches that entail deploying software [16,30] or hardware [22,25] to enable network experiments, or leveraging the vantage point of a popular Internet destination to deploy custom web-based measurement code [13,31]. While these approaches have identified a number of different violations, they are typically difficult for others to replicate: dedicated hardware and software approaches are often difficult to scale as users must install their hardware or software, and web-based approaches require privileged access to a popular web site, which few researchers have. Thus, a scalable approach to quickly enable researchers to measure end-to-end connectivity violations remains an elusive goal.

In this paper, we explore an alternative approach to detecting end-to-end connectivity violations in edge networks, which allows us to achieve measurements from over 1M end hosts simultaneously without requiring



*IMC 2016, November 14 - 16, 2016, Santa Monica, CA, USA*

 ISBN 978-1-4503-4526-2/16/11…$15.00

DOI: http://dx.doi.org/10.1145/2987443.2987455

[1] We consider all end-to-end violations, even those that user opt into, due to their potential impact on protocols making end-to-end assumptions.

| Project | Nodes | ASes | Countries | Measurement Period | ICMP | DNS | HTTP | HTTPS |
|---|---|---|---|---|---|---|---|---|
| *Our approach* | 1,276,873 | 14,772 | 172[1] | 5 days | | ✓ | ✓ | ✓ |
| Netalyzr [16, 19] | 1,217,181 | 14,375 | 196 | 6 years | ✓ | ✓ | ✓ | ✓ |
| BISmark [4, 22] | 406 | 118 | 34 | 2 years | ✓ | ✓ | ✓ | ✓ |
| Dasu [30] | 100,104 | 1,802 | 147 | 6 years | ✓ | ✓ | ✓ | ✓ |
| RIPE Atlas [25] | 9,300 | 3,333 | 181 | 6 years | ✓ | ✓ | ✓ | ✓ |

**Table 1:** Comparison of the approach presented in this paper to other, complementary approaches, based on number of nodes, ASes, countries and different supported protocols. Not shown are web-based approaches [13,31] that require privileged access to a popular web site. While we face a restricted set of protocols, our approach allows for a similar level of scalability but much faster data collection.

users to install our software or hardware. We leverage the commercial Peer-to-Peer (P2P)-based HTTP/S proxy service, *Luminati*, which is based on the *Hola Unblocker* browser plugin. Hola allows users to route traffic via other peers in order to evade geo-blocking, providing strong incentives for users to install it; the developers claim that over 91M users have installed Hola. By using Luminati, we can route HTTP/S traffic via many of the Hola nodes, and gain visibility into their networks. As shown in Table 1, when compared to alternate approaches, our approach allows for similar scalability (over 1M nodes) but with much faster data collection (in 5 days versus 6 years).

However, our approach faces several technical challenges and limitations. As the measurement is based on HTTP/S proxy service, we can only send DNS requests, HTTP traffic (on port 80), and arbitrary traffic (on port 443). This makes it impossible to directly measure violations affecting other protocols such as SMTP; fortunately, HTTP, HTTPS, and DNS represent three of the most commonly used protocols and targets for end-to-end connectivity violations. Additionally, we have no control over the host's network configuration, and only limited visibility into network traffic (i.e., we can induce the node to make a DNS request, but are unable to observe the exact response the node receives). For example, if a host uses OpenDNS, a public DNS resolver, we must infer this by causing the host to make a DNS request to a domain we control and then examining the requests that arrive at our DNS server. Similarly, if a host has malware that modifies outgoing requests, we must infer this as well.

Overall, this paper makes four contributions: *First*, we demonstrate how a large-scale HTTP/S proxy service can be used to measure end-to-end connectivity violations in DNS, HTTP, and HTTPS. We develop techniques that allow us, in most cases, to identify the party responsible for the violations (e.g., the user's DNS resolver, an ISP middlebox, software on the user's machine). This allows researchers to conduct measurements at the scale of approaches deployed by popular web sites, and avoids the overhead of having to convince users to install custom software or hardware.

*Second*, we deploy such measurements to over 1.2M nodes across 14K ASes in 172 countries[3] and investigate numerous instances of end-to-end connectivity violations that result in content modification. With DNS, we observe that 4.8% of nodes have their `NXDOMAIN` responses tampered with, often by the users' ISPs who direct them to pages containing ads. While this general behavior was reported in previous studies from 2008 and 2011 [8, 36], we find different patterns of DNS manipulation. With HTTP, we find that 0.95% of nodes suffer from HTML modification (e.g., ad injection or content filtering) and 1.4% experience image transcoding. Within days, we found new cases of content modification that extend previous results that only applied to the U.S. [37] or that required privileged access to a popular web site [23]. With HTTPS, we observe that 0.5% of nodes suffer from certificate replacement (i.e., man-in-the-middle attacks), typically from anti-virus software or malware; these results are in-line with a recent Facebook study [13].

*Third*, during the course of our measurements, we discovered another unexpected end-to-end violation: *content monitoring*. Specifically, we observed unexpected requests arriving at our measurement server, indicating that the user's application-level traffic was being monitored, and that the content was refetched by a third party. We found that this affected 1.5% of all nodes, and that it is most commonly conducted by anti-virus software and ISP-level middleboxes. These findings raise significant security, privacy, and Internet freedom issues for users subject to this monitoring.

*Fourth*, we make all of our analysis code and data public to the research community at

`https://tft.ccs.neu.edu`

allowing other researchers to use a similar approach to detect end-to-end connectivity violations in DNS, HTTP, and HTTPS.

## 2. BACKGROUND

In this section, we provide background on *Luminati*, the *Hola Unblocker*, and approaches to detecting end-to-end violations.

[2]As described in Section 3.1, we use CAIDA's AS-to-organization mapping to determine the country of each AS.

[3]Throughout the paper, we infer country-level information based on where the networks are registered (see Section 3.1); thus, our country-level statistics are measuring ASes, not users.

## 2.1 Large-scale network measurements

Conducting large-scale measurements to detect end-to-end connectivity violations has been of significant interest to researchers for many years. In general, making such measurements requires the cooperation of machines in a variety of networks across the globe. As a result, most prior approaches fall into one of two classes:

**1. Dedicated hardware/software** The first class of approaches is based on having users deploy dedicated hardware or explicitly run measurement software. For example, the Netalyzr [16] project is a Java applet that diagnoses network problems, and the Dasu [30] project is a BitTorrent extension that tells users how their ISP performs; these both also collect measurements for researchers. Note that Netalyzr [16] has comparable coverage [19] with our approach, but their coverage was amassed over 6 years (instead of days in our case). Similarly, the RIPE Atlas [25] and BISmark [22] projects deploy dedicated hardware to a variety of networks, enabling researchers to send and receive traffic from different vantage points. These projects and others discussed in Section 8 have the benefit that they can generally send arbitrary traffic. However, they can be difficult for other researchers to emulate, as researchers must convince users to install the software, or build and deploy dedicated hardware.

**2. Web-based measurements** The second class of approaches is based on injecting JavaScript or Flash into Web pages that runs measurement code. This approach has been successful employed by Google to measure the incidence of ad injection [31], and by Facebook to measure the incidence of HTTPS certificate replacement [13] (i.e., man-in-the-middle attacks). Due to the popularity of these sites, these approaches can quickly gather data from a large number of diverse users. Unfortunately, these approaches are typically limited in the protocols and destinations they can measure (due to web browser sandboxing and the web security model), and they require privileged access to a popular web site.

Our goal is to develop an approach that achieves the best of both of these: allow researchers to conduct measurements without having privileged access, and without having to spend significant effort to develop software or hardware for users to install.

## 2.2 Hola Unblocker

The *Hola Unblocker* (`http://hola.org/`) is a system deployed by Hola Networks that allows users to route traffic via a large number of proxies across the globe. The software is provided in a number of different forms, including a Windows application, a Firefox add-on, a Chrome extension, and an Android application. Hola claims [14] that more than 91 million people across the globe installed the system.

When users install Hola, they have two options for how to "pay" for access to the service:

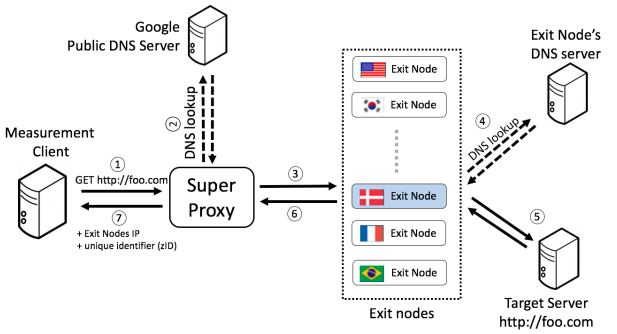


**Figure 1:** Timeline of a request in Luminati: the client connects to the super proxy and makes the request ①; the super proxy makes a DNS request ② and forwards the request to the exit node ③; the exit node makes a DNS request if desired ④ and then requests the actual content ⑤. The response is then returned to the super proxy ⑥ and finally to the client ⑦.

1. Users can choose to pay $5 per month (or $45 per year) for a "premium subscription."

2. Users can choose to allow Hola to route traffic via their machine, and then can use Hola for free.

If users choose the second (free) option, clients of *Luminati* (described below) are also allowed to route traffic via the user's machine. Through experimentation, we found that not all Hola clients actually are available in Luminati. In fact, if the user uses any version of the Hola software *other than* the version for Windows or Mac OS, the user is allowed to use Hola, but no traffic is routed via the user's machine. In the case of Windows or Mac OS, a separate service is installed on the user's machine, and this service maintains a persistent connection with the Hola servers. For more details on this service and the security implications, we refer the reader to the study by Vectra [34].

## 2.3 Luminati

*Luminati* is the *paid* HTTP/S proxy service that routes traffic via Hola nodes.[4] Clients of Luminati can use an API to automate requests, as well as express preferences over which Hola client will be selected to route their traffic. Luminati clients are charged on a per-GB basis, and all Luminati traffic is first routed via a Hola server before being forwarded to a Hola user's client. Below, we provide more details about the Luminati service and the protocols and control that Luminati clients are afforded.

**Architecture** Once a client signs up with Luminati, they are given a username and password to access the service. To route traffic via Luminati, clients make a proxy connection to a Hola server `zproxy.luminati.org` (called the *super proxy*); the super proxy then forwards the client's request to a Hola

[4]The exception to this policy is traffic sent to a few domains (e.g., Google); for these, traffic is forwarded directly from Hola servers.

client (called the *exit node*). The exit node then connects to the server the client wishes to connect to, makes the request, and returns the response back via the super proxy. Thus, the Luminati client interacts only with the super proxy. An overview is shown in Figure 1.

**Exit node selection** Luminati allows clients a measure of control over which exit node is picked to forward the traffic. *First*, the client is allowed to select the country that the exit node is located in by adding a `-country-XX` parameter to their username (where `XX` is the ISO country code). *Second*, the client is allowed to control whether the same exit node is used for subsequent requests by appending a `-session-XXX` parameter to their username. For example, if the client wished to make multiple requests via a single exit node, they select a random number (say, 429) and append `-session-429` to their username. Then, if the client makes another request to Luminati within 60 seconds using that same session number, it will be routed via the same exit node. If the client instead picks a different session number, Luminati will instead route via a new exit node.

**DNS request location** Luminati also allows clients to control where the DNS resolution is performed (recall that HTTP proxy requests take the form of `GET http://foo.com`). *First*, clients can request that the DNS resolution be done by the super proxy (using Google's DNS service). As this is normally faster, this is the default behavior. *Second*, clients can request that DNS resolution be done by the exit node (using the exit node's DNS server). This is done by appending a `-dns-remote` parameter to their username. Allowing the exit node to make the DNS request enables the client to observe any DNS localization that occurs based on requesting IP address (and, as we will show in Section 4, to measure any DNS content manipulation that the exit node experiences).

**Logging and debugging** In the HTTP response header (`X-Hola-Timeline-Debug` and `X-Hola-Unblocker-Debug`) Luminati provides debugging information about the request that is useful for understanding different events. Luminati includes a `zID` parameter that represents a persistent unique identifier for the exit node.[5] Thus, by recording these `zID`s, we can measure if we are accessing the same Hola exit node over long timescales, even if the exit node has changed IP address.

Luminati will also automatically "retry" requests with additional exit nodes if the first request fails, up to five times. If the request ultimately succeeds, the Luminati debugging response header will include the `zID`s of all exit nodes tried and why each request failed. This behavior is useful, as if the user requested a specific exit node be re-used for a subsequent request but that exit node went offline during the request, the debugging header will indicate that the first request failed but Luminati automatically retried with a different exit node.

**HTTPS** So far, we have described how the Luminati HTTP proxy works. Luminati also allows requests to be made over port 443. To do so, the client connects to the super proxy and issues a `CONNECT` ≪`IP`≫`:443` request to the super proxy. At this point, Luminati establishes a TCP-level tunnel via the exit node to the destination IP address. Luminati does not enforce that the client then initiates a TLS handshake—at that point the client can send any data it wishes—but Luminati only allows the `CONNECT` command to connect to port 443. The upshot of this behavior is that Luminati clients can collect the SSL certificates observed by exit nodes by starting the TLS handshake and requesting certificates.

# 3. METHODOLOGY AND DATASET

In this section, we describe how we use Luminati to collect data, detail the datasets we collected, and discuss the ethics of our measurement methodology.

## 3.1 Preliminaries

Throughout the paper, we look at exit nodes at the Autonomous System (AS) level, the Organization (ISP) level, and the country level. We map IP addresses to ASes using data from RouteViews [26] taken at the same time as our data collection. We map ASes to ISPs (as one ISP may operate many ASes) using CAIDA's AS-organizations dataset [6]. We determine the country of the ISP using the same dataset.

## 3.2 Selecting exit nodes

Hola nodes are unlikely to be representative of all Internet nodes, so we try to crawl as many as is feasible in order to obtain as representative a sample as possible. While Luminati gives us some control over which exit nodes are used to send traffic, it does not allow us to enumerate *all* exit nodes. As a result, we must iteratively request new exit nodes until we begin seeing many of the exit nodes we have already seen before (as identified by the `zID` value). Thus, our data collection methodology proceeds by picking a country (in proportion to the number of exit nodes Luminati reports in that country) and picking a random session number. We repeat this procedure until the rate of new exit nodes we discover drops significantly.[6]

## 3.3 Collected data sets

As each experiment requires a custom measurement methodology, we defer describing the exact requests we made to each exit node to the later sections. However, in Table 2, we present an overview of the number of exit nodes and their distribution. We collected our

[5] We verified this `zID` parameter is static by installing Hola on a machine we control and locating it with Luminati; the `zID` parameter included in the Luminati response is the same one that was located in the hola_svc.exe.cid file on our machine's drive.

[6] Note that the Luminati network is very dynamic, so there is no point at which we have crawled "all" exit nodes.

| | DNS (§4) | HTTP (§5) | HTTPS (§6) | Monitoring (§7) |
|---|---|---|---|---|
| Exit Nodes | 753,111 | 49,545 | 807,910 | 747,449 |
| ASes | 10,197 | 12,658 | 10,007 | 11,638 |
| Countries | 167 | 171 | 115 | 167 |

**Table 2:** Number of exit nodes, unique IP addresses, and corresponding ASes and countries for each of the four experiments in this paper.

datasets between April 13 and April 18, 2016 for the DNS and content monitoring studies, between April 14 and April 18, 2016 for the HTTPS study, and between May 4 and May 8, 2016 for the HTTP study. In most experiments, we are able to use over 650K exit nodes in over 165 countries (the HTTP and HTTPS experiments pose limitations that require us to study fewer exit nodes and countries, respectively; these limitations are described in those sections).

## 3.4 Discussion

**Ethics** Our methodology brings up a few ethical measurement issues, and we wish to discuss them explicitly before presenting our results. We first note that we paid the operators of Luminati for access to their proxy service, and were careful to not violate their Terms of Service. Additionally, using Luminati does not expose any PII of the exit nodes' users. We note the operators of the exit nodes agreed to allow Hola to route Luminati traffic via their nodes in exchange for free service[7]; these users have the ability to opt-out of such forwarding either by subscribing to Hola (for $5/month) or uninstalling the software. Regardless, we took great care to ensure that our measurements would not harm the users, either by us sending too much traffic or by visiting any potentially sensitive domains. For each exit node (identified by the `zID` parameter), we never downloaded more than 1 MB across all of our experiments. Additionally, we only requested content or DNS lookups from domains we created for the experiment, or from a small number of select sites (the Alexa top 20 domains in the user's country and the top 10 U.S. university domains). We believe that our methodology carefully balances the potential harm to the operators of the exit node with the scientific benefit of our results.

**Generality** Before presenting our results, we briefly discuss whether our network measurement techniques can be applied to other VPN services. There are several different types of VPN-like services offered today, each with its own characteristics. Luminati is notable in that it provides a very large number of exit IP addresses, but only HTTP and HTTPS protocols are allowed. Thus, our techniques are directly applicable to other HTTP/HTTPS-based VPN services; however, services that are implemented using a static set of centralized servers are likely to be less interesting from a network measurement perspective as they will likely cover a only small set of networks. Additionally, we could extend our methodologies for VPNs that allow arbitrary traffic to be sent, enabling us to capture end-to-end connectivity violations in protocols like SMTP; we leave exploring this further to future work.

[7] `https://hola.org/legal/sla`

# 4. DNS NXDOMAIN HIJACKING

We begin our look into end-to-end connectivity violations by investigating the prevalence of modifying DNS `NXDOMAIN` responses, which indicate that a given domain name does not exist. This practice is often referred to as "hijacking." Previous work [8, 36] showed that ISPs may hijack such responses to "assist" users by sending them to a "search help" page (or one simply filled with advertisements) instead of allowing the browser to show a connection error to the user. As this practice can create security vulnerabilities, break non-HTTP protocols, and confuse users, it has generated significant controversy [10, 28]. We first introduce our methodology, then describe our data set, and close by analyzing the causes and prevalence of `NXDOMAIN` hijacking.

## 4.1 Methodology

At first glance, measuring `NXDOMAIN` hijacking seems trivial: we could return a `NXDOMAIN` response to a Luminati exit node and see if the super proxy reports the error. However, in practice it more difficult, because (a) Luminati first checks that the requested domain name exists at the super proxy before forwarding the request to the exit node, and (b) returning an `NXDOMAIN` response does not allow us to identify the IP address of the exit node that receives it; rather, we only see the IP address of the exit node's DNS server. To address these issues, we need to ensure that Luminati's super proxy check passes, and that we can reliably retrieve the exit node's IP address.

Thus, for each exit node we wish to measure, we first select two unique domain names $d_1$ and $d_2$ for a domain whose authoritative server we control. We then proceed as follows, illustrated in Figure 2:

1. We configure our DNS server to always return a valid `A` record pointing to our web server for $d_1$. We also configure our DNS server to return a valid `A` record for $d_2$, but *only* if the request comes from Luminati's super proxy's DNS server (empirically determined to be one of Google's anycasted 8.8.8.8 DNS servers, located in 74.125.0.0/16). For all other source IP addresses, our DNS server returns `NXDOMAIN`. This is necessary to convince the super proxy to forward the request to the exit node.[8]
2. We then request that the exit node fetch `http://`$d_1$. We record the IP address of the exit

[8] The careful reader will note that this prevents us from measuring exit nodes that use the same anycasted Google DNS server; we filter these out after measuring the exit node's DNS server's IP address in step 2.

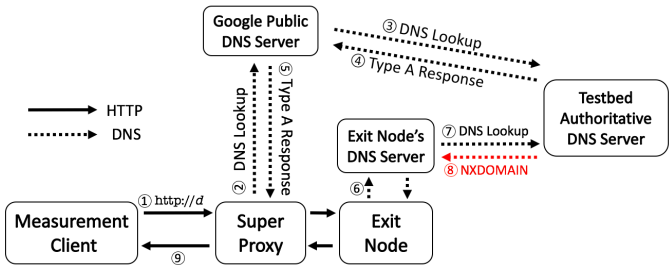


**Figure 2:** Timeline of measurement of `NXDOMAIN` hijacking: the client connects to the super proxy and makes the request ①; the super proxy makes a DNS request, our authoritative DNS server returns a `A` record, and the request is forwarded to the exit node ②∼⑤; the exit node makes a DNS request via its DNS server and our authoritative server returns `NXDOMAIN` response ⑥∼⑧. The error response (i.e., `NXDOMAIN` is not hijacked) or the resulting content (i.e., `NXDOMAIN` is hijacked) is returned to our client ⑨.

node's DNS server (from the incoming DNS request), the exit node's IP address (from the incoming HTTP request), and the exit node's `zID` (from the headers in the Luminati response). This allows us to establish the exit node's IP address for the subsequent `NXDOMAIN` test.

3. Using the same exit node, we request `http://`$d_2$. If we receive an `NXDOMAIN` error in the Luminati log, we know the exit node received the correct response. Otherwise, we record the content that was served to the exit node for later analysis.

## 4.2 Results

We use this methodology to measure a total of 753,111 unique exit nodes from 167 countries and 10,197 ASes. We find that these exit nodes are configured to use a total of 33,446 unique DNS servers. We observe that 717,311 of the exit nodes (95.2%) do not experience `NXDOMAIN` hijacking, but the other 35,800 exit nodes (4.8%) have their response intercepted.

We first obtain a macroscopic view of `NXDOMAIN` hijacking phenomena by grouping exit nodes according to country and AS, and focus on the groups where we have at least 100 exit nodes. This sample size allows us to draw strong inferences about the parties responsible for the hijacking. We observe a number of interesting trends: the exit nodes that experience hijacking are widely spread across the globe, with only 262 (40%) ASes and 15 (10%) countries having no exit nodes that hijacking. However, we do observe instances where hijacking is common among our exit nodes: we observe that in 20 ASes, more than one-third of exit nodes experience it. Table 3 shows the top 10 countries sorted by the fraction of exit nodes with hijacked DNS responses. For example, we found that in Malaysia, more than 52% of exit nodes we measured experienced hijacking.

For the rest of the section, we focus on individual DNS servers; We therefore consider all DNS servers where we observed at least 10 exit nodes, which leaves us 9,839 (29.4%) DNS servers.

| Rank | Country | Exit nodes Hijacked | Exit nodes Total | Ratio |
|---|---|---|---|---|
| 1 | Malaysia | 3,652 | 6,983 | 52.3% |
| 2 | Indonesia | 3,178 | 8,568 | 37.1% |
| 3 | China | 237 | 671 | 35.3% |
| 4 | U.K. | 9,553 | 37,156 | 25.7% |
| 5 | Germany | 4,703 | 19,076 | 24.7% |
| 6 | U.S. | 6,108 | 33,398 | 18.3% |
| 7 | India | 1,127 | 6,868 | 16.4% |
| 8 | Brazil | 3,190 | 24,298 | 16.4% |
| 9 | Benin | 90 | 716 | 12.6% |
| 10 | Jordan | 76 | 1,117 | 7.7% |

**Table 3:** Table showing the top 10 countries sorted by the ratio of hijacked exit nodes.

## 4.3 Causes of hijacking

Below, we investigate the sources of DNS hijacking. There are four locations where the request could be hijacked that we can identify: the ISP's DNS server, a public DNS server, a middlebox, or end-host software.

### *4.3.1 ISP's DNS server*

If the exit node is using the ISP's DNS server, then this server may be configured to hijack responses and direct users to an ISP-provided web server. This behavior has been observed at a variety of ISPs like AT&T [11] and Verizon [35]. If this were the case, we would expect to see that most of the exit nodes that use an ISP's DNS server would experience hijacking.

Our first task is to identify ISP-provided DNS servers. To do so, we group exit nodes by the DNS server that we observed them to use. We identify ISP-provided DNS servers as ones where all exit nodes and the DNS server belong to the same ISP; this results in 9,584 ISP-provided DNS servers. For statistical significance, we focus on those where we observe at least 10 exit nodes using the DNS server; this represents 534 of these servers. We then focus on those that are likely to be hijacking responses by selecting those among the 534 where at least 90% of the exit nodes using the server experienced hijacking; this represents 366 unique ISP-provided DNS servers (3.8% of all ISP servers) covering a total of 17,358 exit nodes.

In Table 4, we aggregate these 366 DNS servers into 19 ISPs from 9 countries. We observe that the majority of these ISPs and DNS servers are in the U.S., and their behavior varies by ISP. For example, we find that TMnet intercepts `NXDOMAIN` response and serves content that redirects the mistyped URL to `http://midascdn.nervesis.com`. This product's tagline is "We turn users' typing errors into your advertising advantage", clearly indicating that TMnet hijacks `NXDOMAIN` responses for advertising revenue.

As another example, we found that five ISPs used nearly identical JavaScript code in their hijacked response HTML: Cox Communication, Oi Fixo, TalkTalk, BT Internet, and Verizon. This code redirects users to a web page managed by each ISP, which typically includes search results for the `NXDOMAIN` domain. The

| Country | ISP | DNS Servers | Exit Nodes |
|---|---|---|---|
| Argentina | Telefonica de Argentina | 14 | 276 |
| Australia | Dodo Australia | 21 | 1,404 |
| Brazil | Oi Fixo | 21 | 2,558 |
| | CTBC | 4 | 290 |
| Germany | Deutsche Telekom AG | 8 | 1,385 |
| India | Airtel Broadband | 9 | 735 |
| | BSNL | 2 | 71 |
| | Ntl. Int. Backbone | 8 | 245 |
| Malaysia | TMnet | 8 | 1,676 |
| Spain | ONO | 2 | 71 |
| U.K. | BT Internet | 6 | 479 |
| | Talk Talk | 46 | 3,738 |
| U.S. | AT&T | 37 | 561 |
| | Cable One | 4 | 108 |
| | Cox Communications | 63 | 1,789 |
| | Mediacom Cable | 6 | 219 |
| | Suddenlink | 9 | 98 |
| | Verizon | 98 | 2,102 |
| | WideOpenWest | 1 | 39 |

**Table 4:** Table showing ISP DNS servers that hijack responses for more than 90% of exit nodes. Also shown is the number of DNS servers and exit nodes per ISP.

common JavaScript code suggests that these ISPs are using a common hardware device or software package to implement the hijacking.

In summary, `NXDOMAIN` hijacking by ISP DNS servers, while rare overall, affects a wide variety of networks globally and often causes users' browsers to visit sites with advertising. This raises serious concerns about privacy and deceptive business practices, as it is unclear what additional information is exposed to third-party advertisers, nor is it clear whether users have knowingly opted-in to participate in such advertising.

### 4.3.2 Public/External DNS server

If the exit node is configured to use a public DNS server external to the ISP—such as Google or OpenDNS—then this server may be hijacking responses (e.g., OpenDNS has been observed to do so [1]). Similar to the ISP DNS server hijacking, if this were the case, we would expect to see most of the exit nodes using a public DNS server receiving hijacked responses.

To identify public DNS servers, we use the same grouping by DNS server as in Section 4.3.1, and disregard the DNS servers with fewer than 10 exit nodes for statistical significance. We then identify public DNS servers as ones where we observe exit nodes coming from more than two countries.[9] We find 1,110 such public DNS servers, and we observe 21 (1.89%)[10] of them to be hijacking more than 90% of the exit node responses; these 21 servers are used by a total of 1,512 exit nodes.

Next, we take a closer look at the operators of these 21 servers to determine whether they are, in fact, public DNS servers. Specifically, we (a) identify the owner of the DNS server's IP address based on the owner of its BGP prefix and (b) issue DNS queries directly to each DNS server to determine whether it responds. We find four public DNS services we can identify: (1) *Comodo DNS*[11], encompassing 9 DNS servers, (2) *UltraDNS*, encompassing 4 DNS servers, (3) *LookSafe*, a piece of malware that changes users' DNS settings,[12] encompassing 2 DNS servers, and (4) *Level 3*, encompassing 3 DNS servers. From the remaining 3 public DNS servers, we are unable to identify the operator.[13]

### 4.3.3 ISP middleboxes and malware

If the exit node received a hijacked response but we cannot attribute it to a DNS server, then there are two other potential vectors: somewhere along the network path (e.g., a transparent DNS proxy) or software on the exit node itself (e.g., malware). In general, disambiguating these cases is difficult, as we have little visibility into the network path or the software on the exit node. However, we can get clues as to the source of the hijacking by looking at the *content* of the HTML page returned.

We first focus on exit nodes where we *know* the server is not performing hijacking. Specifically, we focus exit nodes using Google's 8.8.8.8 DNS service[14], which is well-known to not hijack responses. We observe 927 (0.12%) exit nodes that use Google's DNS service and yet still receive a hijacked response.

Next, we look into the content returned, extracting the URL links that appear in the response. If we observe links to ISP-operated web sites, it is likely that the hijacking is occurring either somewhere along the path or due to ISP-provided software. If we instead observe links to known malware or advertising domains (e.g., affiliate programs), it is likely that the hijacking is occurring due to malware on the exit node.

Across these 927 exit nodes, we extract 119 URLs; Table 5 presents all domains from the URLs that we observed on at least 5 exit nodes. We observe a number of interesting phenomena: *First*, we find that 12 URLs are from exit nodes in a small number of ASes, and the URLs link to sites operated by the same ISP. For example, all 80 exit nodes who received the content including the `http://navigationshilfe.t-`

[9] Interestingly, we are also able to measure when ISPs are likely pointing their subscribers to public DNS servers. We identify 91 ASes where at least 80% of their exit nodes are using Google DNS services. For example, for AS 28683 (*OPT Benin*) in Benin, we observe 225 exit nodes out of 227 (99.1%) using Google DNS. These results align with a recent study [32] that reported that 16.2% of African ASes' DNS resolvers are located at outside of their network.

[10] In fact, this fraction is in-line with a previous study [8] from 2008, which reported that 2% of public DNS servers hijack `NXDOMAIN` response by querying to most of IPv4 space. However, they focused only on open resolvers, whereas we are measuring DNS resolvers that nodes are configured to use.

[11] `http://www.comodo.com/secure-dns/`

[12] `http://www.spyware-techie.com/looksafe-removal-guide`

[13] In fact, we are not able to issue DNS resolution queries to two of them, despite the fact that the exit nodes using them come from more than 6 countries.

[14] We look for DNS requests coming from Google's published netblocks.

| URL | Exit Nodes | ASes |
|---|---|---|
| navigationshilfe.t-online.de | 80 | 1 |
| www.webaddresshelp.bt.com | 73 | 1 |
| v3.mercusuar.uzone.id | 53 | 1 |
| error.talktalk.co.uk | 46 | 3 |
| dnserros.oi.com.br | 40 | 2 |
| dnserrorassist.att.net | 32 | 1 |
| searchassist.verizon.com | 30 | 1 |
| finder.cox.net | 17 | 1 |
| ayudaenlabusqueda.telefonica.com.ar | 16 | 1 |
| google.dodo.com.au | 13 | 1 |
| airtelforum.com | 14 | 1 |
| nodomain.ctbc.com.br | 7 | 1 |
| search.mediacomcable.com | 7 | 1 |
| midascdn.nervesis.com | 68 | 1 |
| nortonsafe.search.ask.com | 25 | 18 |
| securedns.comodo.com | 9 | 9 |

**Table 5:** Table showing domains present in URLs from hijacked NXDOMAIN responses served by Google's DNS server. The top 12 rows represent cases of likely ISP hijacking; the bottom two rows (shaded) represent cases of likely anti-virus software or malware.

online.de are in the AS 3320 (Deutsche Telekom). Similarly, all 46 exit nodes whose content contains http://error.talktalk.com are in ASes 43234, 13285, and 9105—all from Talk Talk. Thus, we can reasonably conclude that these ISPs are responsible for intercepting the NXDOMAIN response from our DNS server.

*Second*, we also identify cases where we suspect software on the exit node (anti-virus software or malware) is performing the hijacking. For example, 25 exit nodes received content including http://nortonsafe.search.ask.com, which appears with other 5 URLs containing "symantec" or "norton" in 18 different ASes and in 18 different countries. A similar pattern exists for content with the URL http://securedns.comodo.com. Given the large number of ISPs affected, we can infer that this is likely due to software at end hosts and not due to an ISP.

## 4.4 Summary

In this section, we developed techniques to use Luminati to explore the prevalence of and mechanisms behind NXDOMAIN hijacking across the world. Overall, we found that 4.8% of all exit nodes experienced such hijacking. If we look at the sources, we can attribute 89.6% of the hijacking to ISP DNS servers, 7.7% of the hijacking to public DNS servers, and 2.7% of the hijacking to either ISP-provided software or malware.

This fraction of vantage points affected by this behavior is higher than was reported in 2008 [8], but is also *substantially* lower than reported in a 2011 study; in the latter, 24% of Netalyzr sessions experienced NXDOMAIN *wildcarding* [36]. We believe one key difference is that we use six times more vantage points, and our results maybe somewhat less biased by users who run Netalyzr because they suspect problems with their network configuration.

# 5. HTTP CONTENT MODIFICATION

Another important type of end-to-end violation consists of a third party modifying HTTP contents between servers and clients. In this section, we use Luminati to investigate how HTTP content is modified by focusing on four types of content: (1) HTML, (2) images, (3) JavaScript, and (4) CSS.

## 5.1 Methodology

Our methodology is relatively straightforward: we simply fetch content from our Web server via an exit node, and check whether the content we receive is the same as what we sent. For this experiment, we fetch four different pieces of content through each exit node: a 9 KB HTML page, a 39 KB JPEG image, a 258 KB unminified JavaScript library, and a 3 KB un-minified CSS file. We initially tried using very small files to minimize the bandwidth consumption and load on exit nodes, but found that when fetched objects smaller than 1 KB, we observed much lower levels of content modification.

Because the content modification tests require significantly more bandwidth that the other tests, we use a methodology that minimizes the network traffic. We first measure three exit nodes in the same AS. If we detect that at least one exit node in an AS experiences content modification, we then return to that AS to measure more exit nodes to confirm whether the modification is more likely due to the network service provider or software running on an end host. This approach may underestimate content modification that ASes apply non-uniformly, but should detect AS-level modifications that apply to most nodes, as well as provide an estimate of the prevalence of modifications due to end-host software.

## 5.2 Measurement Results

Using this methodology, we measured 49,545 exit nodes in 12,658 ASes across 171 countries. We detected HTML content modification for 472 exit nodes (0.95%), image modification for 694 (1.4%), JavaScript modification for 45 (0.09%), and CSS modification for 11 (0.002%). We discuss each of these in detail below.

**HTML** We find that 472 exit nodes (0.95%) received modified HTML pages, which is in-line with the previous work [23]. We filter 32 cases that return pages such as "bandwidth exceeded" or "blocked" messages, leaving 440 exit nodes in 268 ASes. In all of these cases, JavaScript code is injected into the HTML page.

To understand the source of injection, we group exit nodes at the AS level and find the ratio of exit nodes affected. We do this only for ASes where we measured at least 10 exit nodes, leaving us with 272 exit nodes spread across 65 ASes.

We find that only in one AS (discussed below) do all nodes receive injected content and only in one other do more than 10% receive injected content. Thus, for

| URL or Keyword | Exit Nodes | Countries (ASes) |
|---|---|---|
| NetSparkQuiltingResult | 21 | 1 (1) |
| d36mw5gp02ykm5.cloudfront.net | 201 | 44 (99) |
| msmdzbsyrw.org | 97 | 4 (76) |
| pgjs.me[14] | 16 | 1 (12) |
| jswrite.com/script1.js[15] | 15 | 9 (10) |
| var oiasudoj; | 11 | 1 (11) |
| AdTaily_Widget_Container | 11 | 8 (9) |

**Table 6:** Table showing the most commonly appearing 7 URLs (or keyword) of injected JavaScript. The first row (shaded) represent the keyword used for web filtering in AS 42925 (*Internet Rimon ISP*).

exit nodes in the remaining ASes, the culprit is likely software running on the exit node.

We observe that all exit nodes in AS 42925 (*Internet Rimon ISP*), received modified HTML content. Through manual inspection, we find that their source codes share the same `meta` tag, *NetsparkQuiltingResult*. We find that this tag is generated by *NetSpark*'s[17] Web filtering software, which shares the same parent company as Internet Rimon.

For the remaining cases, we investigated the JavaScript code injected into HTML content by manually extracting URLs or keywords that characterize the code. This process identified 21 URLs or keywords from 416 exit nodes (94.5% of all injected content). Table 6 shows the most common of these. Most injected content comes from malware. For example, when the modified HTML contains `class id AdTaily_Widget_Container`, an additional 335 KB of advertisements are included. Similarly, if the injected JavaScript includes a variable named `oiasudoj`, the modified response size increases by 23 KB and the page loads more than 170 ads.

**Images** We find that 694 exit nodes in 22 ASes receive modified images. As with the HTML analysis, we first group exit nodes by AS, then calculate the fraction of exit nodes per AS that are affected. We filter out ASes with fewer than 10 exit nodes, yielding 604 exit nodes in 12 ASes. Interestingly, via manual verification, we observe that *all* of 12 ASes correspond to mobile ISPs. Since Luminati exit nodes are not attached to mobile networks, we believe that most are temporarily connected via tethering, enabling us to measure mobile ISPs as well.[18] We also found that in all cases, the images were compressed to lower quality levels, which is consistent with recent prior work showing this behavior in the US [37].

In addition to the US, our measurements reveal significant image compression in other countries. To validate that image compression is due to the ISP, we use two approaches. First, we compare how many exit nodes observe image compression in each AS. Table 7 shows detailed results. Interestingly, we first notice that *Vodafone* is present in four different countries, indicating that they use compression in their networks globally. We also observe that not every exit node experiences compression. For example, only 6% of exit nodes in AS16135 received compressed content. We do not currently have an explanation for this phenomenon, but it may be due to different subscriber plans.

| AS | ISP (Country) | Exit Nodes Mod. | Total | Ratio | Cmp. |
|---|---|---|---|---|---|
| 15617 | Wind Hellas (GR) | 10 | 10 | 100% | 53% |
| 29180 | Telefonica (GB) | 17 | 17 | 100% | 47% |
| 29975 | Vodacom (ZA) | 83 | 88 | 94% | M |
| 25135 | Vodafone (GB) | 15 | 18 | 83% | 54% |
| 36935 | Vodafone (EG) | 62 | 81 | 77% | M |
| 36925 | Meditelecom (MA) | 87 | 128 | 68% | 34% |
| 16135 | Turkcell (TR) | 44 | 65 | 68% | 54% |
| 15897 | Vodafone (TR) | 14 | 25 | 56% | 53% |
| 12361 | Vodafone (GR) | 11 | 23 | 48% | 52% |
| 37492 | Orange (TN) | 97 | 331 | 29% | 34% |
| 132199 | Globe (PH) | 197 | 1,374 | 14% | 51% |
| 12844 | Bouygues (FR) | 34 | 615 | 6% | 53% |

**Table 7:** Exit nodes that received compressed images, divided by ISP. Also show in the final column is the compression ratio observed (Cmp.); "M" indicates multiple compression ratios were observed.

We also analyze the size of compressed images that exit nodes received. The rationale behind this is that if image is compressed by the ISP, all exit nodes should see similar sizes. Table 7 also shows the image compression ratios for each AS. We find that different ISPs use different compression ratios (suggesting different implementations or settings), but are consistent for all exit nodes *except* in two ASes. In these, we observe two different compression ratios spread across many exit nodes.

These results strongly suggest that the ISP is responsible for transparently compressing images, and that the exit nodes generally see consistent compression. The impact of image compression is potentially significantly reduced image quality, which not only violates net neutrality principles but also reduces quality of experience for affected users.

**JavaScript and CSS** We observe 45 exit nodes and 11 exit nodes received JavaScript and CSS content replaced by different content, respectively. Manually inspecting these revealed they all consisted of error pages or empty responses. We did not observe any modification to the original content such as minification or injection.

## 5.3 Summary

In this section, we examined how HTTP objects are modified in-flight, finding significant modifications for both HTML and image objects. Our findings complement results from previous studies that looked into the source of ad-injection by investigating Chrome exten-

[14] reported at `http://www.freefixer.com/b/remove-pgjs-me-from-firefox-chrome-and-internet-explorer/`

[15] reported at `https://www.herbiez.com/?p=218`

[17] `http://www.netspark.com`

[18] Luminati also uses mobile devices running the Hola Mobile App, but they become exit nodes only when they are on WiFi and charging [15].

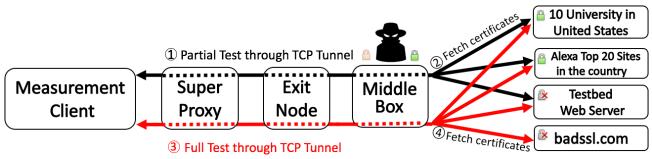


**Figure 3:** Timeline of measurement of the middleboxes man-in-the-middling: the client iteratively establishes a TCP-level tunnel via the exit node to the three target servers on port 443 ① and fetches their certificates ②. If our measurement client finds at least one of them to be modified, it does the same procedure of ① and ② but fetches the certificates from all target servers ③ and ④.

sions/Windows binaries [31] or local ISP servers [38]. We also observed images to be modified only by mobile ISPs, in-line with other recent research [37]; however, we present results from many additional ISPs and countries.

# 6. SSL CERTIFICATE REPLACEMENT

Next, we investigate the end-to-end violations in HTTPS by looking for cases where a third party intercepts a SSL/TLS connection and presents a certificate that is different from the one provided by the server. This attack, known as a man-in-the-middle (MITM) attack, is typically conducted to allow inspection of otherwise-encrypted content.

## 6.1 Methodology

To detect certificate replacement using Luminati, we use the `HTTP CONNECT` method with the super proxy, which tunnels all TCP port 443 via the exit node, allowing our measurement client to conduct a TLS handshake with arbitrary servers. For each site we measure, we complete a TLS handshake and record the SSL certificates presented; we then terminate the connection without actually requesting any content.

As certificate replacement may target individual web sites, we choose three different classes of sites to test:

1. **Popular sites** We choose the 20 most popular sites that support HTTPS from each country's Alexa Ranking [3].
2. **International sites** We choose the web sites of 10 U.S. universities where IMC'16 PC members are affiliated.
3. **Invalid sites** We create three sites under our control with intentionally invalid certificates: a self-signed certificate, an expired certificate, and a certificate having incorrect `Common Name`.

For each exit node we wish to measure, we conduct a two-phase scan. *First*, as an initial phase, we select one site randomly from each of these three groups. We connect to these three sites with the exit node and then download and verify the certificates. For the first two classes of sites, we check for certificate replacement by validating the certificate chain.[19,20] For third class of site, we check whether the invalid certificate matches exactly (because we know exactly which certificate was sent). *Second*, if any of these checks fail, we then download certificates for all 33 sites via the exit node. Figure 3 presents a diagram of our methodology.

| Issuer Name | Exit Nodes | Type |
|---|---|---|
| Avast | 3,283 | Anti-Virus/Security |
| AVG Technology | 247 | Anti-Virus/Security |
| BitDefender | 241 | Anti-Virus/Security |
| Eset SSL Filter | 217 | Anti-Virus/Security |
| Kaspersky | 68 | Anti-Virus/Security |
| OpenDNS | 64 | Content filter |
| Cyberoam SSL | 35 | Anti-Virus/Security |
| Sample CA 2 | 29 | N/A |
| Fortigate | 17 | Anti-Virus/Security |
| *Empty* | 14 | N/A |
| Cloudguard.me | 14 | Malware |
| Dr. Web | 13 | Anti-Virus/Security |
| McAfee | 6 | Anti-Virus/Security |

**Table 8:** Table showing the most commonly appearing 13 issuers of replaced certificates. The types refer to anti-virus/security products, content filters, and malware.

## 6.2 Measurement Results

Using this methodology, we measured a total 807,910 exit nodes in 10,007 ASes and 115 countries.[21] Among these exit nodes, we find that 4,540 of them (0.05%) received at least one modified certificate. Interestingly, we find that not every certificate is modified, indicating that certificates can be selectively replaced.

Looking at the AS distribution of these exit nodes, we find the majority of nodes in all ASes do *not* experience certificate replacement (e.g., only 1.2% of ASes have more than 10% of exit nodes experience replacement). Since the observed certificate replacement does not strongly depend on AS, we infer that the cause of replacement is likely software on the exit nodes [13, 20].

To investigate the source of the certificate replacement, we focus on the `Issuer Common Name` of the certificates.[22] We find 320 unique `Issuer Common Names`, and Table 8 details the 13 groups that have at least five exit nodes (these cover 93.6% of all exit nodes experiencing certificate replacement). We manually investigated these 13 issuers and found three primary causes: anti-virus software, content filtering services, and malware.

[19] We check the validity using `openssl verify`, configured to trust the OS X 10.11 root store [21]; this includes 187 unique root certificates.

[20] We cannot do an *exact match* check on the certificate, as many sites use content delivery networks and end up using different certificates on different servers.

[21] We measured fewer countries in this experiment than the others as we were unable to get Alexa rankings for many of the countries.

[22] Leaf certificates are signed by a Certificate Authority (CA), and the identity of the CA that signed the certificate is present in the `Issuer` field. Thus, the `Issuer` field is likely to provide clues about who is conducting the certificate replacement.

**Anti-virus** The most commonly occurring cause of certificate replacement is anti-virus software (or, other types of security software) that appear to be running on the exit node; these account for 9 of the top 13 issuers, which is in-line with a previous study [13]. We find that all of these appear to generate spoofed leaf certificates on demand, suggesting that as part of the install process, they installed a custom root certificate into the browser's or OS's trust store to avoid browser warnings.[23] Interestingly, we find that, with the exception of Avast, each system uses the same public keys on all certificates on a given exit node (i.e., every spoofed certificate uses the same public key on the same exit node running the software), which was not previously reported [12].

We also observe that Cyberroam, ESET SSL Filter, Kaspersky, McAfee, and Fortigate replace originally *invalid* certificates with seemingly *valid* spoofed certificates. In fact, all valid and invalid certificates share the same public key and other attributes in the `Issuer` field such as name, organization, and country, which strongly indicates that all of them are signed by the same root certificate. Thus, unless these AV systems have some other mechanism to alert users, their browsers would not raise an alert for sites with invalid certificates, potentially exposing users to security vulnerabilities like phishing attacks. Avast,[24] BitDefender, and Dr. Web also also generate new certificates for sites with invalid certificates, but they do so using a different `Issuer`.

**Content filter** We observe one content filter, OpenDNS, that replaces TLS certificates. OpenDNS provides a service called *Block Page and Block Page Bypass*[25] that presents a block page—depending the network administrator's list of blocked sites—over both HTTP and HTTPS connections. To prevent browser warnings from occurring on blocked pages, OpenDNS must install the "OpenDNS Root Certificate Authority (CA)" certificate into their root store. In our collected data, we find that OpenDNS uses this certificate to MITM secure connections, but only if the server's certificate is valid (i.e., they do not replace certificates that were originally invalid).

**Malware** Finally, we observe one piece of malware that is present on 14 exit nodes: *Cloudguard.me*. We observe that the replaced certificates copy most of the fields from the original, valid certificate, presumably to make them appear more legitimate to users. We also find these exit nodes experience HTTP content injection,[26] further evidence that Cloudguard.me is malware. Interestingly, all of these exit nodes are in Russian ISPs.

[23] Carné de Carnavalet and Mannan [12] investigated six of the anti-virus systems (Avast, AVG, Eset, Kaspersky, Dr. Web) and found that this is indeed the case.

[24] Avast has multiple issues, including: Avast! web/mail shield root, Avast! web/mail shield self-signed root, Avast! web/mail shield untrusted root, Avast trusted CA, Avast untrusted CA

[25] `https://support.opendns.com/entries/98279288-Block-Page-Errors-Installing-the-OpenDNS-Root-CA`

[26] `http://www.spyware-techie.com/cloudguard-removal-guide`

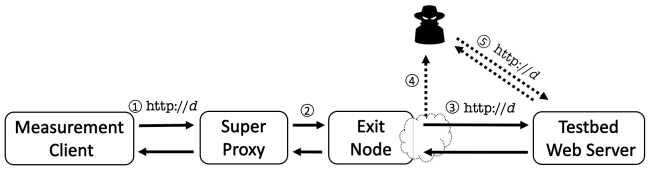


**Figure 4:** Timeline of measurement of the middleboxes monitoring content: the client connects to the super proxy and makes a request ① for a unique domain; the proxy forwards the request to the exit node ②; who makes the actual request ③. If someone is monitoring the request ④ they later make the same request to our server ⑤.

## 6.3 Summary

In this section, we examined how Luminati's exit nodes' HTTPS connections are intercepted. While we only found that only 0.05% of all exit nodes experience certificate replacement, it is an extremely serious attack: when SSL is intercepted, users' passwords, banking details, and other sensitive information are exposed to whomever is replacing certificates. Further, we observe than even in the case of the "legitimate" uses of certificate replacement (i.e., anti-virus and other security products), many follow poor security practices by sharing private keys and replacing invalid certificates with ones the browser trusts.

# 7. CONTENT MONITORING

All of our analysis thus far has focused on end-to-end violations where content was modified. Another concerning form of end-to-end violation is *content monitoring*, or cases where middleboxes are silently observing content that users are downloading for the purpose of scanning content or otherwise controlling access. While content modification is easy to detect (e.g., via block pages), content monitoring is significantly more difficult to detect, as there is (by definition) no change to the content itself. However, we discovered we can detect certain types of content monitoring based on unexpected requests arriving at our measurement server.

## 7.1 Methodology

In the previous experiments, recall that we generated a unique domain name for each HTTP request. When analyzing the results, we sometimes observed *multiple* requests for a given domain, even though our client only ever requested each domain once. We explored this behavior in more detail by generating a unique per-exit-node domain name $d$ whose IP address pointed to our web server. Then, we requested that the exit node fetch `http://`$d$, which we expect to generate a single request at our web server. We monitored the Web server for up to 24 hours after generating the initial request to see if additional requests for $d$ arrived from different IP addresses; if so, it would indicate that either a middlebox

| Monitoring entity | | Monitored users | | |
|---|---|---|---|---|
| Name | IPs | Exit nodes | ASes | Countries |
| Trend Micro | 55 | 6,571 | 734 | 13 |
| TalkTalk | 6 | 2,233 | 5 | 1 |
| Commtouch | 20 | 1,154 | 371 | 79 |
| AnchorFree | 223 | 461 | 225 | 98 |
| Bluecoat | 12 | 453 | 162 | 64 |
| Tiscali U.K. | 2 | 363 | 6 | 1 |

**Table 9:** Table showing the top six ASes where unexpected requests originated, indicating content monitoring.

along the path or software on the exit node monitored the request and requested the content itself. Figure 4 presents a diagram of our methodology.

## 7.2 Measurement Results

Using this methodology, we measured a total of 747,449 exit nodes, and observed that 11,234 (1.5%) of them resulted in multiple, unexpected requests. In general, it is challenging to determine the cause of the unexpected requests as we have little visibility into the network path or the software on the exit node. However, we are able to get clues by manually looking at features of the unexpected request, including the `User-Agent` field in HTTP request headers and the AS from which the request came.

Overall, we found that the unexpected requests came from a total of 424 unique IP addresses that were different from the exit nodes'; we grouped these by AS, resulting in 54 groups. Table 9 provides more details on the most frequent groups out of these 54; all together, these six sources generated 11,235 (94.0%) of the unexpected requests. We also calculated the time between the exit node's request and the unexpected request; Figure 5 presents the cumulative distribution of the delay for each of these six. We manually investigated these requests and found two primary culprits: anti-virus software and ISP-level services.

### 7.2.1 *Anti-virus and VPN software*

**TrendMicro** The most commonly observed source of the unexpected requests came from IP addresses owned by TrendMicro, an anti-virus software company; we observed unexpected requests from 6,571 (47.8%) exit nodes across 734 ASes and 13 countries. Interestingly, we found that TrendMicro almost always makes *two* unexpected requests: the first request typically arrives between 12 and 120 seconds after the exit node's request, while the second request typically arrives between 200 and 12,500 seconds after the exit node's request. This is immediately visible in Figure 5, with the two parts of the distribution separated by a step at y=0.5. Due to the wide distribution of ASes where exit nodes are affected, we believe this is likely to due to TrendMicro monitoring software (called Web Reputation Services [33]) running on the exit nodes.

**Commtouch** We observe a similar pattern with

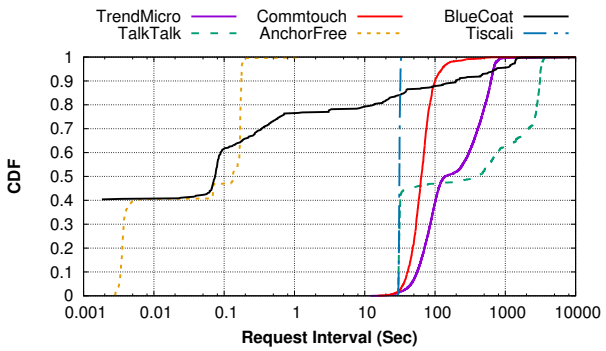


**Figure 5:** Cumulative distribution of delay between the exit node request and the additional, unexpected requests for the top six sources. Note that $x$ axis in log scale.

Commtouch, the former name of CYREN Ltd., a software company making anti-virus software. We observe that 20 IP addresses linked to Commtouch make unexpected requests from 1,154 exit nodes across 371 ASes and 79 countries. We also observe that unlike TrendMicro, Commtouch makes the unexpected requests between 1 and 10 minutes after the exit node's request.

**Anchorfree** The fourth-most-common source of additional requests comes from IP addresses owned by Anchorfree—a "freemium" VPN service for web browsing—and covers 461 exit nodes. We can observe that these exit nodes are using Anchorfree's VPN service, as the IP address of the exit node's request does not match the IP address of the exit node as reported by Luminati (instead, the IP address is from Anchorfree's AS). However, with these nodes, we also observed an *additional* request from a separate IP address in Anchorfree's network, suggesting that Anchorfree is monitoring the content browsed. In fact, we observe that the first request comes from one of 10 different locations around the globe, but the second request always comes from Menlo Park, California. Additionally, the two requests are extremely close in time; 99% of them are separated by under 1 second. This is likely due to their "malware protection" feature provided as part of the "Hotspot Shield" service.

**Bluecoat** Finally, we observe requests from 12 IP addresses owned by Bluecoat Systems (a computer security company) covering 453 exit nodes in 162 ASes across 64 countries. Bluecoat appears largely similar to TrendMicro and Commtouch, with two unexpected requests per exit node. However, we observe that the first of these unexpected requests comes in *before* the exit node's request 83% of the time (this is why Bluecoat's CDF starts at 41%). Thus, it appears that Bluecoat first downloads the content before allowing the exit node's request to proceed.

The behavior of these applications has significant security, privacy, and performance implications for end users. Users' web browsing history is being uploaded in

real-time to the anti-virus company's servers, who are then re-downloading the content fetched by the user. Similarly, Anchorfree is duplicating users' requests, and Bluecoat is holding Web requests until it first fetches it on the user's behalf for analysis.

#### 7.2.2 ISP-level monitoring

**TalkTalk** The second-most-commonly observed source of unexpected requests comes from six IP addresses owned by TalkTalk (a U.K. ISP); these requests were generated by 2,233 exit nodes. We also observe a similar pattern to TrendMicro, in that TalkTalk typically generates two unexpected requests, with the first request arriving almost exactly 30 seconds after the exit node's request and the second request arriving at our measurement server over the next hour (this pattern can be observed in Figure 5). Interestingly, we find that *all* of the exit nodes who generated these unexpected requests are in TalkTalk; we therefore believe that these requests are due to ISP-level content monitoring. Adding further weight to this theory is the fact that the 2,233 exit nodes in TalkTalk that generated unexpected requests represent 45.2% of all the exit nodes that we measured in TalkTalk.

**Tiscali U.K.** We also observe that another ISP, Tiscali U.K., has a similar pattern (in fact, Tiscali U.K. was acquired by TalkTalk in 2009, but continues to be run as a separate entity). We observe two IP addresses in Tiscali U.K. that generated unexpected requests for 363 exit nodes (representing 11.4% of all Tiscali U.K. exit nodes). Unlike TalkTalk, we observe only one unexpected request; this request almost always comes in exactly 30 seconds after the exit node's request.

We cannot say for sure why some, but not all, exit nodes in both ISPs experience content monitoring. Potential explanations are that content monitoring could be done non-deterministically (e.g., only 10% of requests are monitored), or it may be due to ISP-provided additional services like parental content controls.[27] Regardless, monitoring and its potential for controlling open access to content has significant implications for Internet users, and should be made transparent.

### 7.3 Summary

In this section, we developed and deployed techniques that can detect certain instances of content monitoring. We found that over 1.5% of all exit nodes suffered from their HTTP requests being collected and re-requested by a third party, and that the most common causes were anti-virus software products, VPN services, and the user's ISPs. All of these instances have significant privacy implications, as these users are likely unaware that their HTTP browsing history is being duplicated in near-real-time.

[27]For example, TalkTalk's opt-in SuperSafe feature (`https://help2.talktalk.co.uk/supersafe-boost-overview`).

## 8. RELATED WORK

Before concluding, we provide an overview of related work on end-to-end connectivity violations (we discussed other measurement approaches in Section 2.1).

**DNS Manipulation** Because DNS provides no built-in security, DNS traffic has been the vector for a large number of different attacks [24,27,29]. In parallel, other work has explored how different DNS resolvers or ISPs manipulate DNS traffic. Kuhrner et al. [17] classified open DNS resolvers using fingerprints of DNS software and found that millions of the resolvers deliberately manipulated DNS resolutions and returned unexpected IP address information. Dagon et al. [8] found in 2008 that 2.4% of DNS queries to open DNS servers are returned with incorrect answers; Weaver et al. [36] used Netalyzr data in 2011 to perform a similar study, finding up to 24% of responses manipulated.

Our `NXDOMAIN` hijacking study shares many goals and is complementary to these previous studies, but with the following key differences. First, using Luminati allows us to measure in-use DNS servers, rather than having to scan for open resolvers as some approaches have done. Second, our approach provides measurements at greater scale and in less time than previous work; the Netalyzr data set, while incredibly useful, took months to years to be created; we are able to measure a similar number of nodes in a matter of days. Finally, our results present a new look at `NXDOMAIN` hijacking, as the two previous studies are now both over five years old.

**HTTPS content manipulation** SSL and TLS secure a large portion of the Internet's traffic today; together with a public key infrastructure, they provide authentication and encryption. However, the increasing fraction of HTTPS traffic has led to renewed interest in trying to gain visibility into such encrypted traffic, typically via man-in-the-middle (MITM) attacks [7]. There has been a series of work [2,5,12,18] that focuses on how MITM attacks are conducted in different scenarios, including mobile networks [18], using invalid certificates [5], when authentication protocols are tunneled [2], or via anti-virus software [12]. Recently Carné de Carnavalet and Mannan [12] analyzed eight commercial antivirus software and parental control applications, which interpose a TLS proxy in between end hosts' communications. Our results complement theirs; while they were concerned with understanding how anti-virus applications work, we demonstrated how wide-spread they are. Additionally, Huang et al. [13] studied SSL MITM attacks by injecting a Flash object into Facebook's Web pages. Similar to our results, they find that 0.2% of hosts receive forged certificates. In comparison, we were able to measure a similar number of users, but did so without requiring access to a popular Web site.

**HTTP content manipulation** Because HTTP by itself has no integrity checks, violations of the end-to-end

connectivity of HTTP traffic have been occurring for many years. Much of this has occurred due to the use of proxies, both in both broadband [9, 16] and mobile networks [37]. For example, the Netalyzr project [16] revealed HTTP proxies by monitoring request and response headers, and also used this to identify proxy caching policies and content transcoding.

Other forms of HTTP tampering are more pernicious. Recently, Thomas et al. [31] found that more than 5% of unique daily IP addresses accessing Google experience ad injection due to malicious Chrome extensions or Window binaries. Also, Zhang et al. [38] used software installed on users' Microsoft computers to identify nine ISPs in U.S. that redirect users to rogue servers that serve modified content. In mobile networks, Xu et al. [37] recently investigated transparent proxies and identified content manipulation and caching behavior by studying four major mobile carriers in the US.

Our approach to measuring HTTP end-to-end connectivity violations largely complements these. Our results show similar patterns of malicious software and web proxies, but our approach allows us to examine this behavior for millions of hosts in networks worldwide.

# 9. CONCLUSION

In this paper, we proposed a new approach to measuring end-to-end connectivity violations in DNS, HTTP, and HTTPS, based on the Luminati proxy service. We developed techniques to be able to detect content manipulation in all three protocols, and used these techniques to measure over 1.2M hosts across 14K ASes in 172 countries. Our results, at various points, confirm prior findings, update prior studies with new measurements, and reveal new ways in which content manipulation is leading to security vulnerabilities. As part of our study, we also identified a new content monitoring attack, where users' URL requests via HTTP are uploaded to third party servers, who unexpectedly later fetch the same content.

We demonstrated our methodology allows researchers to quickly conduct measurement studies that previously required months to years of user-recruitment effort or privileged access to a popular Web site. This opens the door to continuous measurements worldwide, with the ability to see how various types of violations evolve over time. We believe this will be useful not only for improving transparency, privacy, and security, but also for informing regulators and policymakers.

## Acknowledgments

We thank the anonymous reviewers and our shepherd, Amogh Dhamdhere, for their helpful comments. This research was supported in part by NSF grants CNS-1421444 and CNS-1563320.

# 10. REFERENCES

[1] A new reason to love OpenDNS: No more ads. `https://www.opendns.com/no-more-ads/`.

[2] N. Asokan, V. Niemi, and K. Nyberg. Man-in-the-middle in Tunnelled Authentication Protocols. *Security Protocols*, Springer, 2005.

[3] Alexa Top 500 Global Sites. `http://www.alexa.com/topsites`.

[4] BISMark Network Dashboard. `http://networkdashboard.org/`.

[5] F. Callegati, W. Cerroni, and M. Ramilli. Man-in-the-Middle Attack to the HTTPS Protocol. *IEEE Security and Privacy*, 7(1), 2009.

[6] CAIDA AS Organizations Dataset. `http://www.caida.org/data/as-organizations/`.

[7] D. Dolev and A. C. Yao. On the Security of Public Key Protocols. *IEEE Transactions on Information Theory*, 29(2), 1983.

[8] D. Dagon, C. Lee, W. Lee, and N. Provos. Corrupted DNS Resolution Paths: The Rise of a Malicious Resolution Authority. *ndss*, 2008.

[9] G. Detal, B. Hesmans, O. Bonaventure, Y. Vanaubel, and B. Donnet. Revealing Middlebox Interference with Tracebox. *IMC*, 2015.

[10] T. Dziuba. When ISPs hijack your rights to NXDOMAIN. *The Register*, 2009. `http://www.theregister.co.uk/2009/08/17/dzuiba_virgin_media_opendns/`.

[11] DNS Error Assist. `http://dnserrorassist.att.net`.

[12] X. de Carné de Carnavalet and M. Mannan. Killed by Proxy: Analyzing Client-end TLS Interception Software. *NSDI*, 2016.

[13] L.-S. Huang, A. Rice, E. Ellingsen, and C. Jackson. Analyzing Forged SSL Certificates in the Wild. *IEEE S&P*, 2014.

[14] Hola. `https://hola.org`.

[15] Hola. Personal Communication.

[16] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson. Netalyzr: Illuminating the Edge Network. *IMC*, 2010.

[17] M. Kührer, T. Hupperich, J. Bushart, C. Rossow, and T. Holz. Going Wild: Large-Scale Classification of Open DNS Resolvers. *IMC*, 2015.

[18] U. Meyer and S. Wetzel. A Man-in-the-middle Attack on UMTS. *WISE*, 2004.

[19] Narseo Vallina-Rodriguez. Personal Communication.

[20] M. O'Neill, S. Ruoti, K. Seamons, and D. Zappala. POSTER: TLS Proxies: Friend or Foe? *CCS*, 2014.

[21] OS X El Capitan: List of available trusted root certificates. `https://support.apple.com/en-us/HT205204`.

[22] Project BISmark. `http://projectbismark.net`.

[23] C. Reis, S. D. Gribble, T. Kohno, and N. C. Weaver. Detecting in-flight Page Changes with Web Tripwires. *NSDI*, 2008.

[24] V. Ramasubramanian and E. G. Sirer. Perils of Transitive Trust in the Domain Name System. *IMC*, 2005.

[25] RIPE NCC Annual Report 2015. `https://www.ripe.net/publications/docs/ripe-665`.

[26] University of Oregon RouteViews project. `http://www.routeviews.org/`.

[27] J. Schlamp, J. Gustafsson, M. Wählisch, T. C. Schmidt, and G. Carle. The Abandoned Side of the Internet: Hijacking Internet Resources When Domain Names Expire. *Int. Workshop on Traffic Mon. and An.*, 2015.

[28] R. Singel. ISPs' Error Page Ads Let Hackers Hijack Entire Web, Researcher Discloses. *WIRED*, 2008. `https://www.wired.com/2008/04/isps-error-page`.

[29] S. Son and V. Shmatikov. The hitchhiker's guide to DNS cache poisoning. *Security and Privacy in Communication Networks*, Springer, 2010.

[30] M. A. Sánchez, J. S. Otto, Z. S. Bischof, D. R. Choffnes, F. E. Bustamante, B. Krishnamurthy, and W. Willinger. Dasu: Pushing Experiments to the Internet's Edge. *NSDI*, 2013.

[31] K. Thomas, E. Bursztein, C. Grier, G. Ho, N. Jagpal, A. Kapravelos, D. McCoy, A. Nappa, V. Paxson, P. Pearce, N. Provos, and M. A. Rajab. Injection at Scale: Assessing Deceptive Advertisement Modifications. *IEEE S&P*, 2015.

[32] R. F. a. G. Tyson, P. Francois, and A. Sathiaseelan. Pushing the Frontier: Exploring the African Web Ecosystem. *WWW*, 2016.

[33] TrendMicro Web Reputation Services. `http://esupport.trendmicro.com/solution/en-US/1058991.aspx`.

[34] Vectra Threat Labs. Technical analysis of Hola. `http://blog.vectranetworks.com/blog/technical-analysis-of-hola`.

[35] Verizon Search Assist. `http://searchassist.verizon.com`.

[36] N. Weaver, C. Kreibich, B. Nechaev, and V. Paxson. Implications of Netalyzr's DNS Measurements. *SATIN*, 2011.

[37] X. Xu, Y. Jiang, T. Flach, E. Katz-Bassett, D. Choffnes, and R. Govindan. Investigating Transparent Web Proxies in Cellular Networks. *PAM*, 2015.

[38] C. Zhang, C. Huang, K. W. Ross, D. A. Maltz, and J. Li. Inflight Modifications of Content: Who Are the Culprits? *LEET*, 2011.