A Large-scale Analysis of Content Modification by Open HTTP Proxies

Georgios Tsirantonakis

Thesis submitted in partial fulfillment of the requirements for the

Master of Science degree in Computer Science

University of Crete
School of Sciences and Engineering
Computer Science Department
University Campus, Voutes, Heraklion, GR-70013, Greece

Thesis Advisors:
Prof. Evangelos Markatos,
Dr. Sotiris Ioannidis

UNIVERSITY OF CRETE
COMPUTER SCIENCE DEPARTMENT

A Large-scale Analysis of Content Modification by Open HTTP Proxies

Thesis submitted by
Georgios Tsirantonakis
in partial fulfillment of the requirements for the
Master of Science degree in Computer Science

THESIS APPROVAL

Author: _____
Georgios Tsirantonakis

Committee approvals: _____
Evangelos Markatos
Professor, Thesis Supervisor

_____
Sotiris Ioannidis
Research Director, Committee Member

_____
Xenofontas Dimitropoulos
Professor, Committee Member

Departmental approval: _____
Antonios Argyros
Professor, Director of Graduate Studies

Heraklion, 29/08/2017

Abstract

Open HTTP proxies offer a fast and convenient solution for routing web traffic towards a destination. In contrast to more elaborate relaying systems, such as anonymity networks or VPN services, users can freely connect to an open HTTP proxy without the need to install any special software. Therefore, open HTTP proxies are an attractive option for bypassing IP-based filters and geo-location restrictions, circumventing content blocking and censorship, and in general, hiding the client's IP address when accessing a web server. Nevertheless, the consequences of routing traffic through an untrusted third party can be severe, while the operating incentives of the thousands of publicly available HTTP proxies are questionable.

In this work, we present the results of a large-scale analysis of open HTTP proxies, focusing on determining the extent to which user traffic is manipulated while being relayed. We have designed and implemented a methodology for detecting proxies that, instead of passively relaying traffic, actively modify the relayed content. Beyond simple detection, the framework is capable of macroscopically attributing certain traffic modifications at the network level to well-defined malicious actions, such as ad injection, user fingerprinting, and redirection to malware landing pages, to name a few.

We have applied our methodology on a set of nearly 65,000 open HTTP proxies, which we monitored for a period of two months. Our findings are alarming. A significant fraction (5.15%) of the proxies we tested were found to perform some form of content injection in the retrieved HTML page, which can be considered as malicious or unwanted. Specifically, in 47% of the cases the injected code injected advertisements, 39% collected user information that can be used for fingerprinting and tracking and 12% attempted to redirect the user to pages that contained malware.

Our study reveals the true incentives of many of the publicly available web proxies. Our findings raise several concerns, as we demonstrate multiple cases where the user can be severely affected by connecting to an open proxy. In addition, we have generated a list of currently pinpointed malicious servers that should be strongly avoided and black-listed. Last but not least, our framework can stand as an open monitor for detecting additional malicious proxies in the future.

Περίληψη

Οι ανοικτοί διακομιστές μεσολάβησης πρωτοκόλλου μεταφοράς υπερκειμένου (HTTP) προσφέρουν μια γρήγορη και βολική λύση για τη δρομολόγηση της διαδικτυακής κίνησης προς έναν προορισμό. Σε αντίθεση με πιο περίπλοκα συστήματα αναμετάδοσης, όπως δίκτυα ανωνυμίας ή υπηρεσίες εικονικών ιδιωτικών δικτύων (VPN), οι χρήστες μπορούν να συνδεθούν ελεύθερα σε ένα ανοικτό μεσολαβητή χωρίς την ανάγκη εγκατάστασης ειδικού λογισμικού. Επομένως, οι ανοιχτοί διακομιστές είναι μια ελκυστική επιλογή για την παράκαμψη φίλτρων με βάση την διεύθυνσης διαδικτυακού πρωτοκόλλου (IP)και Περιορισμούς γεωγραφικής θέσης, παρακάμπτοντας την παρεμπόδιση περιεχομένου και τη λογοκρισία και, γενικότερα, απόκρυψη της διεύθυνσης IP του πελάτη κατά την πρόσβαση σε έναν διακομιστή ιστού. Παρ ́όλα αυτά, οι συνέπειες της δρομολόγησης της κυκλοφορίας μέσω ενός μη αξιόπιστου τρίτου μέσου μπορεί να είναι σοβαρές, ενώ τα κίνητρα λειτουργίας των χιλιάδων διαθέσιμων στο κοινό HTTP διακομιστών είναι αμφισβητήσιμα.

Στην παρούσα εργασία παρουσιάζουμε τα αποτελέσματα μιας ανάλυσης μεγάλης κλίμακας των ανοιχτών διακομιστών HTTP, εστιάζοντας στον καθορισμό του βαθμού στον οποίο χειραγωγείται η κίνηση του χρήστη ενώ αναμεταδίδονται. Έχουμε σχεδιάσει και εφαρμόσει μια μεθοδολογία που ανιχνεύει διακομιστές που αντί να μεταδίδουν παθητικά την κυκλοφορία, τροποποιούν ενεργά το αναμεταδιδόμενο περιεχόμενο. Πέρα από την απλή ανίχνευση, το πλαίσιο είναι ικανό για την απόδοση ορισμένων μικροσκοπικών αλλαγών κυκλοφορίας σε επίπεδο δικτύου σε σαφώς καθορισμένες κακόβουλες ενέργειες, όπως η τοποθέτηση διαφημίσεων, η αποτύπωση ηλεκτρονικών δακτυλικών αποτυπωμάτων των χρηστών και η ανακατεύθυνση σε σελίδες προορισμού κακόβουλου λογισμικού, για να αναφέρουμε μερικές.

Έχουμε εφαρμόσει τη μεθοδολογία μας σε ένα σύνολο σχεδόν 65,000 ανοιχτών HTTP διακομιστών, τους οποίους παρακολουθούσαμε για περίοδο δύο μηνών. Τα ευρήματά μας είναι ανησυχητικά. Ένα σημαντικό ποσοστό (5.15 %) των διακομιστών που δοκιμάσαμε βρέθηκαν να κάνουν μια αλλαγή περιεχομένου στην ανακτημένη σελίδα HTML, η οποία μπορεί να θεωρηθεί ως κακόβουλη ή ανεπιθύμητη. Συγκεκριμένα, στο 47% των περιπτώσεων η αλλαγή περιεχομένου είχε να κάνει με διαφημίσεις, 39% συγκέντρωσαν πληροφορίες χρηστών που μπορούν να χρησιμοποιηθουν για ηλεκτρονικά δακτυλικά αποτυπώματα και παρακολούθηση και το 12% προσπάθησε να ανακατευθύνει τον χρήστη σε σελίδες που περιείχαν κακόβουλο λογισμικό.

Η μελέτη μας αποκαλύπτει τα αληθινά κίνητρα πολλών από των διαθέσιμών στο κοινό διαδικτυακών διακομιστών. Τα ευρήματά μας εγείρουν αρκετές ανησυχίες, καθώς καταδεικνύουμε πολλές περιπτώσεις όπου ο χρήστης μπορεί να επηρεαστεί σοβαρά από τη σύνδεση με ανοικτό διακομιστή. Επιπλέον, έχουμε δημιουργήσει μια λίστα των κακόβουλων διακομιστών που έχουν εντοπιστεί αλλά και εντοπίζονται αυτήν τη στιγμή, οι οποίοι πρέπει να αποφεύγονται και να δημοσιεύονται σε μαύρες λίστες. Τέλος, το πλαίσιο μας μπορεί να σταθεί ως ένας ανοιχτός ελεγκτής για την ανίχνευση πρόσθετων κακόβουλων διακομιστών στο μέλλον

Acknowledgments

I would like to thank my Advisor Dr. Sotiris Ioannidis for his guidance and support during my studies and this work. I would also like to thank my Supervisor, Professor Evangelos Markatos for his overall assistance and interesting discussions all these years. Additionally i would like to thank Professor Michalis Polychronakis and Panagiotis Ilia for providing key insights during the development of this work and Elias Athanasopoulos for his overall help. Many thanks to Professor Xenofontas Dimitropoulos, for being the third committee member in the evaluation of this work.

I need to also express my appreciation towards the people that supported me all those years, my labmates: Thanasis Petsas, Evangelos Ladakis, Dimitris Deyannis, Giorgos Christou, Eva Papadogiannaki, Eirini Degkleri, Kostis Kleftogiorgos, Giorgos Vassiliadis, Rafail Tsirmpas, Nick Christoulakis, Kostas Solomos, Kostas Plelis, Lazaros Koromilas, Antonis Krithinakis and Christos Papachristos and my friends: Dimitri, Giorgo, Lefa, Panteli, Kosta, Maria-Xristina, Ioanna, Flora, Mimi, Eirini, Panito, Xristina, Dano, Katerina, Foti, Mariellen.

Finally i would like to extend a big thank you to my family for believing in me and supporting me all these years, my father Sofoklis, my mother Dafni and my sister Smaragdi. This work would have never been possible without them

This work has been performed at the **Distributed Computing Systems laboratory, Institute of Computer Science, Foundation of Research and Technology – Hellas (FORTH)**. In addition, this work has been supported by the European Commission via the H2020 ICT-32-2014 Project SHARCS under Grant 644571.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Internet users often place their trust on systems that are not under their control. From a security and privacy perspective, a particularly critical class of such systems is HTTP proxies that act as "stepping stones" between clients and servers. By relaying their traffic through a proxy, users can access otherwise blocked content and services due to geographical restrictions, content filtering policies, or censorship, and to some extent preserve their anonymity, by hiding the originating IP address from the final destination (although still exposing it to the proxy).

Once the user traffic reaches the proxy towards its actual destination, unless end-to-end encryption is used, a rogue or compromised proxy can tamper with the transmitted content or snoop for sensitive user data [25]. Even when end-to-end encryption is used, however, the problem is not alleviated, as man-in-the-middle attacks are still possible using fake or even valid—obtained through compromised CAs or generated by powerful adversaries—certificates, or SSL-stripping attacks [19]. The potential harm due to network traffic interception attacks can be severe, ranging from mere annoyance and inconvenience to system compromise and theft of private information.

As the majority of web content is still transmitted unencrypted over HTTP, rogue web proxy operators can monetize their traffic by altering the relayed content to inject ads and affiliate links, prompt users to download spyware and other unwanted software, or mount phishing attacks [16]. Even more deviously, instead of placing extra ads that may annoy users, miscreants can replace *existing* ads in the page with their own ads. This can be as simple as replacing a website's ad network identifier with the attacker's own affiliate identifier, essentially stealing the revenue of the original website. A more severe form of network traffic manipulation is the inclusion of malicious JavaScript code for mounting XSS, CSRF, or DDoS attacks, the injection of exploits against the browser or other client-side software [30, 26], or the infection of downloaded executables [22], which can all result in full system compromise.

The proliferation and widespread use of web proxies necessitate the need for understanding and measuring the extent of content modification by rogue web proxies. The ease of setting up a free online proxy (e.g., on a cloud-hosted virtual machine) and registering it on the numerous "proxy list" websites, raises the question of whether miscreants

employ these tactics to attract and gain access to user traffic, and then monetize it or cause further harm. Sporadic evidence so far has shown that this is indeed happening in various types of network relays, including VPN servers and anonymity network relays [33, 22, 16, 32, 9, 6, 3], but the extent of the problem in the front of web proxies remains unknown.

As a step towards understanding and measuring the extent of content modification by rogue HTTP proxies, in this work we have conducted a large-scale study of open HTTP proxies, focusing on the detection of content alteration or injection events. During a period of two months, we periodically collected publicly available proxies by crawling a set of "proxy list" websites, and probed them using a novel technique based on *honeysites* under our control. Instead of visiting real web sites with and without a proxy, and comparing the difference in the retrieved content, the use of honeysites allows us to avoid any false positives issues due to highly dynamic content (e.g., news updates, rotating ads, and time-related information), or content localization (e.g., due to the proxy's different location than the client). We employed a set of honeysites of increasing complexity and content diversity, and implemented a content modification technique that operates at the level of a page's DOM tree to detect even slight object modifications.

After successfully probing 65,871 proxies, our results indicate that 1004 of them (5.15% of those we found to be working) performed some form of malicious or unwanted content modification. The observed alterations included the injection of extra (or the modification of existing) ads, the inclusion of tracking libraries and fingerprinting code, the collection of data from social networking services on which the user is already authenticated, and the injection of malicious code. We provide a detailed analysis of our measurements in terms of the responsiveness rate of the probed proxies, their location, and their content modification characteristics. We also discuss in detail indicative cases of malicious content modification, after an automated clustering phase that groups together proxies that exhibited similar injection patterns. Besides the injection of ads and user tracking and fingerprinting code, we also discovered more severe and sophisticated instances of malicious behavior, such as SSL stripping, and redirection to servers that have been reported to host malware.

In summary, our work makes the following main contributions:

(i) We present an approach for the detection of unwanted or malicious content modification by rogue HTTP proxies, based on the observation of discrepancies in the retrieval of content through the probed proxies. Our technique uses a set of decoy web sites, dubbed *honeysites*, which are created with an incremental degree of complexity and external content dependencies, to allow differentiating between different types of content injection or alteration.

(ii) We have performed a large-scale measurement study of open HTTP proxies retrieved from public "proxy list" websites, and used our technique to assess the extent of malicious content modification by rogue proxies. Our findings suggest that 5.15% of the working proxies engaged in some form of malicious or unwanted content modification.

(iii) We provide an in-depth analysis of indicative content modification cases that we observed, which involved the injection of ads, tracking and fingerprinting code, private information collection, and malware distribution.

Chapter 2

Background and Related Work

2.1 HTTP Proxies

Web proxies are one of the most widely used types of network relays mainly due to their ease of use, especially for users that simply want to bypass filtering restrictions or access content that is not available in their country. Many web proxy sites allow users to conveniently type in the URL of the page they want to visit, which is then rendered directly, usually within a frame. Legacy HTTP or SOCKS proxies require users to manually configure their browser to explicitly use the proxy by entering the IP address and port of the proxy into the browser's settings window. Many browser extensions facilitate this process, offering single-click buttons for toggling a configured proxy on and off.

Both types of web proxies are widely used, as evident from the numerous websites dedicated to providing up-to-date lists of working web proxies around the world. Besides the URL or IP:port of each proxy, these lists typically also provide information about the geographic location of the proxy (important for users who want to access content available only in certain countries), its supported protocols, the last time the proxy was checked for responsiveness, and even statistics about its uptime and access latency.

It is common for each proxy list website to provide information about hundreds or even thousands of recently checked proxies. Considering that there are numerous independent proxy list websites, we estimate the number of publicly accessible web proxies to be in the order of tens of thousands. This conservative estimation is based on the number of proxies we collected from multiple sources.

2.2 Detecting Web Content Alteration and Fraudulent Ad Traffic

A large body of prior research has focused on security issues related to the ad serving ecosystem. Li et al. [27] shed light to fraudulent activities in on line ad exchanges. In the front of click fraud, Dave et al. [11] proposed a methodology for measuring the extent of click spam, and proactively detecting different simultaneous click-spam attacks. To

counter click fraud, Hamed Haddadi [15] proposed the concept of "Bluf ads," which are ads meant to be detected and clicked only by automated crawlers or click-fraud workers.

Malvertising is also an important threat that has received a lot of attention, mostly focusing on methods to detect and block malicious ads. Li et al. [18] performed a large-scale study of malicious ads, and built MadTracer, a defense that automatically generates detection rules and uses them to block malicious ads. Ford et al. [13] studied the particular class of malware delivery that leverages malicious Flash advertisements to infect web site visitors, and developed an automate Flash content analysis technique for the detection of malicious behavior

Once malware has infected a user system, it often uses ad injection to monetize the victim's web browsing activity. In their large-scale analysis of ad injection on infected machines, Thomas et al. [28] followed a similar approach to ours by inspecting a page's DOM tree to identify the injection of malicious content. The actual identification was based on a whitelist of legitimate content, similar in spirit to content secure policy (CSP) used by modern browsers for whitelisting JavaScript code. This approach has the limitation that if the injected code involves domains that are included in the whitelist (e.g., when replacing existing ads with malicious ads from the same ad network), then the injection will go unnoticed. Our content modification detection approach does not rely on a whitelist, and thus can identify any addition or alteration of an element in the DOM tree, even if it involves the same ad network. Their results indicated that 5% of unique IP addresses accessing Google were infected with malware that injected ads. Operation Ghost Click, one of the largest cybercriminal takedown efforts in history, took down an ad fraud infrastructure that affected 4 million users infected by the DNS changer malware, and made its owners 14 million USD over a period of four years [5].

Many efforts have focused on solutions to detect and prevent content modification. Vratonjic et al. [29] anticipated the problem of in-line ad replacement or modification through man-in-the-middle attacks, and proposed a collaborative technique to preserve the integrity of ads—our findings highlight the need for such schemes, and the necessity of preserving network traffic integrity in general. Reis et al. [24] proposed "web tripwires", a method that allows publishers to include a verification check in their page that compares the DOM a user received with the DOM the publisher sent. Arshad et al. [7] developed OriginTracer, a tool that notifies the user about which party is responsible for any modifications in a loaded page's content by tracking DOM element alterations (e.g., due to browser extensions). The same authors have developed Excision [8], an in-browser mechanism that relies on an enhanced version of the DOM tree to keep track of the relationships between the resources of a page in order to block malicious third-party content.

Although individual websites can detect to a certain extent modifications to their client-rendered content using the above approaches, these mechanisms are not widely adopted. In contrast, our efforts focus on the detection of content modification by scanning publicly accessible HTTP proxies to uncover various forms of traffic modification.

2.3 HTTP Proxy Studies

As discussed in the above, recent research has shown that ad injection and hijacking has been impacting tens of millions of users worldwide through malicious browser extensions and malware infections [28, 5]. It is only natural to expect miscreants to employ similar monetization strategies through the deployment of rogue network relays. Sporadic evidence so far has shown that traffic interception by rogue network relays is indeed happening [33, 22, 16, 32, 9, 6, 3].

O'Neill et al. [21] measured the prevalence of TLS proxies by developing a probe tool that was deployed through Google AdWords campaigns. They found that one in 250 TLS connections are TLS-proxied, and identified over 3,600 cases of malware intercepting a TLS communication. Holz et al. [17] developed the Crossbear system to discover TLS man-in-the-middle attacks by tracking IP routes and comparing certificates with distributed nodes they have deployed across the internet. Carnavalet et al. [12] uncovered security vulnerabilities in TLS proxies used by antivirus and parental control applications, allowing attackers to mount man-in-the-middle attacks. Weaver et al. [31] leveraged the ICSI Netalyzr client base to measure the prevalence of HTTP proxies, finding that we see 14% of the tested clients made use of a web proxy. They detected and classified several kinds of proxies, and observed cases of content modification due to client-side security software (e.g., antivirus and firewall products) or server-side compression and transcoding. Chung et al. [10] used a paid proxy service whereby users can route their traffic via other users to uncover content manipulation in end-to-end connections.

In contrast to the above studies, our research focuses on publicly available HTTP proxies that users knowingly employ (for various reasons, such as to access otherwise blocked content, seek protection when using untrusted networks, and preserve their anonymity), and which are available via numerous "free proxy list" websites. Furthermore, our methodology is different, since we focus on detecting modifications in the DOM tree of the page served to the user. Besides some limited evidence that free HTTP proxies are indeed involved in suspicious activity [16] and some work that shows potential attacks by someone controlling a proxy [4], our effort is the first to perform a large-scale, systematic analysis of the extent and nature of content modification performed by free HTTP proxies.

Chapter 3

Methodology

In this section we present the methodology we followed to collect a large representative sample of the open HTTP proxies ecosystem, and determine whether these proxies modify the user traffic they relay. Our design allows us to identify their behavior as well as any content modification activity performed by the proxies and to categorize these modifications according to different types of malicious behavior.

3.1 Crawling for Proxies

At first, for collecting a set of open HTTP proxies, we built and deployed a web crawler that visits daily 11 popular websites that provide lists of publicly accessible web proxies. Our crawler visits and collects all the proxies listed in each page of each of these websites. This task was performed on a daily basis for a period of two months.

To make sure our dataset was as complete as possible, alongside the 11 sites we crawled automatically we also collected proxies from 5 other websites. These websites imposed restrictions on the users in order to provide access to their proxy lists such as login and/or captcha. To include them in our study we visited these sites every 10 days in the duration of our experiment and added their proxies in our database manually. Lastly for the duration of a month we purchased shared proxies from another website to be able to make comparisons between the bought and non bought ones. This process resulted in the collection of 65,871 different HTTP proxies in our database collected over a period of two months.

Each day after crawling and updating our base with new proxies we probed them with nmap to see which of them were listening to incoming connections. We sent probes to all of our known proxies almost hourly (22 times per day) with no more than 10 maximum retries for each probe. If a proxy was found to be alive in any of the probes it was added to our daily list of proxies to be tested. By using nmaps we were able to save us some time by only checking the proxies that appear alive and test the reliability and lifetime of our proxies.

Using the daily list of alive proxies constructed by our probes we tested our proxies against three sites. More specifically we request through Selenium and Firefox two hon-

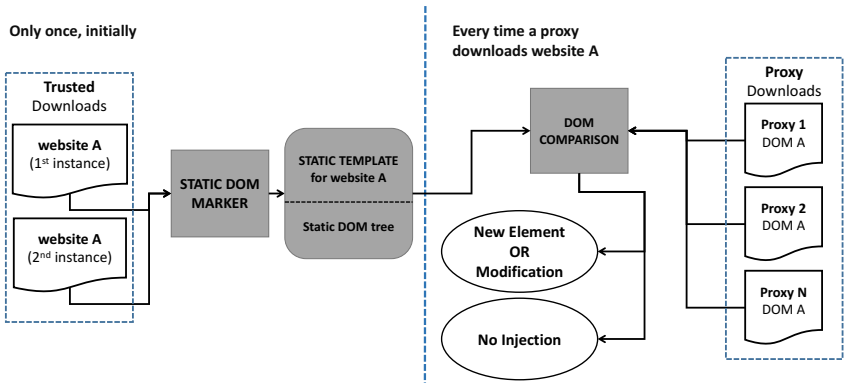

Figure 3.1: Diagram of the methodology followed for detecting content modifications in websites downloaded by proxies.

eysites created by us and a real site. These requests are done in parallel tabs with their timeout set at 3 minutes. The reason for this high timeout is that through tests we found that due to the public nature of the proxies a lot of them tend to be slow. Per day we tested all of them two or more times depending on the number of proxies we found to be alive. If a proxy managed to download our sites successfully it was removed from the daily list since we did not want to retest a working proxy a second time in the same day.

3.2 Honeysites

Conceptually, a simple approach for detecting content modification by rogue network relays is to visit a given public web page twice, once connecting directly and once through the relay, and comparing the fetched content. In practice, however, this naive approach is not effective for two main reasons. First, parts of the page may change due to inherently dynamic content, such as news updates, time-related information, rotating ads, and various forms of content personalization. Second, the geographic distance between the client and the proxy may also result in differences due to content localization, as many websites switch to pages of different language (and possibly region-specific content) based on the visitor's origin. On the other hand, selecting only simple static web pages for probing may miss rogue proxies that focus only on specific types of content that can be monetized, i.e., ads that already exist in the page.

To avoid the complexity of having to distinguish between legitimate content modification due to the inherent dynamic behavior of a fetched web page, and any kind of content modification performed by the proxy itself, we opted for an alternative approach that relies on decoy web sites under our control. Specifically, we set up two different testing websites, to which we refer as *honeysites*, of increasing complexity and dependence on (controllable) third-party content (e.g., ads).

The reason for using multiple honeysites is to not only detect content modification events, but also examine whether the behavior of the tested proxies changes according

to the type of content relayed through them (e.g., detect whether a proxy modifies only JavaScript files). The main difference between the two honeysites lies in the degree of dynamic content generated by the server, and on the third party resources they use. The first honeysite (h_1) consists of a relatively simple and completely static web page (a copy of the content of the testing website `example.org`). The second one (h_2) is a website created in WordPress,[1] which contains JavaScript elements and resources. In addition of an *iframe* that loads some external content, and three *fake ads*.

The included ads in h_2 were created based on actual source code provided by three popular ad networks: Google AdSense,[2] Media.net[3] and BuySellAds.[4] We did not actually register for displaying ads from these networks, but just used the example ad inclusion code snippets they provide by setting the publisher's ID to an invalid, non-existent value. This ensures that the included ads will not be rendered, as the requests for fetching them will typically fail. What is important, however, is that these fake ads look (from the perspective of the proxy) identical and indistinguishable from legitimate ones. Having this extra honeysite with embedded ads allows us to differentiate between proxies that perform unconditional content modification, from proxies that inject content depending on the presence of ad networks in the fetched web page (e.g., by replacing the existing legitimate publisher's ID with the attacker's registered ID, by injecting extra ads, or by completely replacing the honeysite's fake ads.

For a given proxy to be tested, we automatically browsed to all our honeysites through the proxy, and compared the retrieved content with the original content that was served by our web server. A benefit of this approach is that, by having control of both the client and the server, we can precisely identify whether any part of the page was modified, or whether any new additional content was injected.

To prevent any potentially sophisticated malicious proxies from suspecting the true purpose of our honeysites, and thus avoid exhibiting any inappropriate behavior, as an initial measure we decided to use Amazon's S3 cloud service[5] for hosting the honeysites, instead of using any other infrastructure (e.g., servers hosted in our academic institution's network).

3.3 Detecting Content Modification

Detecting any content modification by a proxy for the static honeysite h_1 is straightforward, given that its "ground truth" content never changes. The dynamic content of h_2 however, makes content modification more challenging, as we have to distinguish between changes due to the dynamic content itself and changes due to potential content injection or modification by a proxy. To achieve that, we initially downloaded the dynamic honeysite multiple times from a trusted computer (without using a proxy) and compared its

[1] https://wordpress.com
[2] https://www.google.com/adsense
[3] http://www.media.net
[4] https://www.buysellads.com
[5] https://aws.amazon.com/s3

content. Specifically, our approach constructs and traverses the DOM tree of the trusted downloads, and compares the DOM elements to identify which elements remain the same between different downloads. By comparing the DOM elements and their values, we are able to identify and *mark* the static and dynamic elements of each honeysite, and then create a static version of the honeysite that can be used as a *template* for identifying content modification by malicious proxies.

After creating the static template of each honeysite, we use an approach similar to the process followed for generating the templates themselves to compare the DOM tree of the honeysites downloaded through each probed proxy to the corresponding template. This approach can effectively identify content modification in most cases, as the malicious proxies typically inject new rogue elements or try to modify existing ones. Our system is able to detect changes in the DOM tree when new rogue elements are inserted, and also can detect the modification of the value of the elements that are marked as static. Additionally, it should be noted that the approach followed for the generation of the honeysite templates takes into consideration the position of each dynamic element in the DOM tree. Consequently, during the comparison of a page fetched through a proxy with the corresponding template, our system expects to identify specific dynamic elements at particular positions of the page's DOM tree.

A limitation of our system is that it cannot detect changes in the value of any elements that have been marked as dynamic in the generated template, as these values legitimately change between different downloads. Nevertheless, our system is still able to detect cases of malicious content injection that are missed by existing methodologies for detecting client-side DOM modification. For example, Thomas et al. [28] use a whitelist containing all the domains that are anticipated (i.e., allowed) to appear in a web page, which means that an injection will go unnoticed if the malicious proxy replaces or adds extra advertisements from an ad network that already exists in the original page. Using our approach, a modification may only go unnoticed if the proxy changes the value of a dynamic element.

For our experiments, we dealt with this problem by carefully selecting the types of dynamic content so that it is generated only in a predictable way, allowing us to still detect any unanticipated alterations in it. In the general case, to make our system more robust against potentially sophisticated proxies that may attempt to exploit this limitation and focus their injection only in the dynamic parts of the page, as part of our future work we plan to incorporate an additional safeguard using a template-based approach for specifying the anticipated ranges or patterns of the expected dynamic content.

3.4 Clustering of Content Modification Incidents

After some initial experimentation using the methodology described in the previous section, we observed a larger than anticipated number of proxies modifying the content of the probed honeysites, either by injecting new rogue elements or by modifying existing ones. In particular, our methodology revealed DOM tree modifications in thousands of cases from both honeysites.

To facilitate the analysis of all these individual content modification incidents, we

designed a *content modification clustering* approach for identifying similar patterns of injections and grouping them together. Initially, we categorize the downloads according to the observed changes in the DOM tree, by comparing each download through a proxy with the template of the corresponding honeysite. As an example, if an *iframe* element is expected at a particular position according to the DOM tree of the template, but a script element occurs in the corresponding DOM tree location of a downloaded instance, this means that this script is a potential rogue element inserted by the proxy. Thus, at this stage, we perform an initial grouping of the downloaded pages according to the position in the DOM tree of any injected or altered page elements.

Then, we compare the DOM tree of each downloaded instance with all other downloaded instances in the same cluster (according to the first step). If the DOM tree is exactly the same, which means that the same elements appear in the same order and have the exact same content, then they are added in the same cluster. It should be noted that we do not take into account the dynamic elements of the DOM tree, which also change in the trusted downloads. By performing this clustering process we minimize the effort needed for analyzing the observed content modification incidents, as we only need to manually inspect a single download from each cluster to understand that particular proxy behavior. Also this way it was easy to spot clusters of proxies that modified our honeysite by removing on or more of the existing ads but did not inject anything new.

In addition to the manual inspection of representative content modification cases from each cluster, we also employ dynamic analysis of samples from each cluster, to gain a better understanding of the functionality of any injected code. To that end, we configured the Firefox browser with the Firebug plugin, and manually ran one download from each cluster, while collecting and observing any outgoing requests and fetched resources. Overall, through this process, we detected malicious proxies setting tracking pixels and cookies, injecting ads, and performing browser fingerprinting, among many other cases. A detailed analysis of our findings is presented in Chapter 5.

Chapter 4

Data Analysis

In this section we present the characteristics of the HTTP proxies we analyzed, and the main differences between the benign and the malicious ones. Moreover, we focus on understanding the behavior of proxies, by further analyzing the reasons behind their life cycle and and where they are located. We also aim to ”group” the most common malicious proxies characteristics such as location and which domains they come in contact with.

4.1 Proxy Characteristics

As discussed in Section 3, our system visited daily all the pages of 11 popular websites that list publicly available HTTP proxies for a period of two months, and collected information about the proxies listed in all of the pages of each of these websites. For each unique proxy (given that the same proxy may be listed in more that one websites), the system then fetched our two honeysites and one real site using the proxy, and inspected the retrieved content for possible modifications. In total, we collected and tested 65,871 unique HTTP proxies, 49,444 of which were found to be accepting incoming connections by nmap as it can been seen in Table 4.1.

Interestingly, our attempts to retrieve the honeysites were only successful for only 19,473 proxies (39.38% from those found alive). We look into the reasons for our ”low” number of working proxies in Section 4.1.2 and show that it is mainly due to DNS errors caused by the public and shared usage of the proxies. We made sure to send repeated requests to our known and alive proxies daily with a high 3 minute timeout to be able to ascertain their status.

By following the previously described methodology for detecting content modification, we observed that 7,441 of the proxies (38.21% of the ones that responded) altered in some way the content of a retrieved web page. As the HTML DOM of the fetched pages may be modified by the proxy for various legitimate reasons, the observation of a modification event does not necessarily mean that the probed proxy is malicious, or that its actions can negatively impact a user. The main type of anticipated content alteration that we observed was due to “privacy-protecting” proxies that block any trackers or ads that exist in the retrieved page. We also observed proxies that inject elements or iframes

GROUPS OF PROXIES	NUMBER OF PROXIES
Total Tested:	65871
Found Alive:	49444
Found Working:	19473
Found Injecting:	7441
Found Malicious:	1004

Table 4.1: Total proxies tested and analyzed for the duration of our experiment and the sub categories they belong.

that are empty or modified the DOM of the page in such a way that cannot be classified as malicious. After employing our clustering method (described in Section 3.4) and manually inspecting the clusters that were generated, we identified 1,004 (5.15%) proxies that performed some form of malicious or unwanted content modification. We analyze and present in detail cases of such malicious behavior in Chapter 5.

4.1.1 Collecting and Testing

For our collection process as mentioned in Section 3.1 along with the 11 sites we crawled for proxies, we systematically manually added proxies from 5 other sites where crawling was not feasible. We found that most of these sites would ask for a money fee to generate proxy lists ready to be used in file format and on occasion claim to offer even more proxies to users that pay. By crawling the pages of these sites and collecting their proxies we discovered that some of them claimed to own more proxies than they really had and less surprisingly some proxies existed in multiple sites.

To make sure our database contained a large and representative sample of shared HTTP proxies we even purchased access to a site that claimed to own more than 50,000 shared proxies. As we analyze in more detail in Section 4.1.5 we only received less than 7,436 proxies from this site in total and only 3,278 of them were not already known to our database.

In Figure 4.1 we can see the total number of proxies that existed in our database every day of our experiment for two months. Each single day we sent hourly probes with nmaps to all of our proxies to see if they are alive (accepting incoming connections). If they did we would test them at least two times per day against our sites to see if they are ”working”. It can be derived from the Figure that proxies being found alive start at half of our total proxies but as the days pass and we discover more proxies it drops to one third while the number of working proxies stays about the same.

We visualize the number of new proxies we discovered per day for the duration of our experiment in Figure 4.2. We collect close to 10 thousand proxies every day from the sites we crawl automatically with only a 4% to 6% from them being new to our database. From this small group of new proxies according to our statistics we expect less than one third of them to work. The two spikes in the figure were caused by one of our biggest sites that we crawled that happened to be inaccessible these two days. Interestingly it did not affect

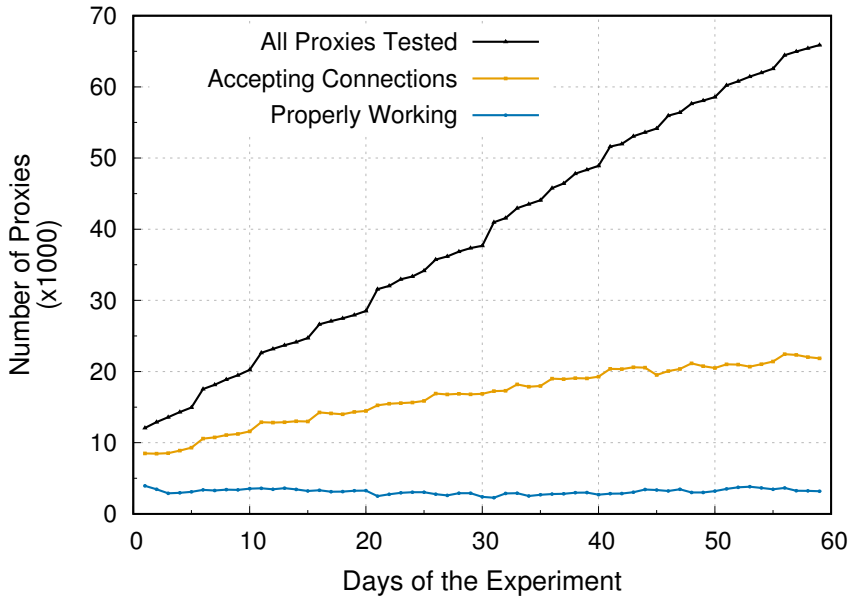

Figure 4.1: Total proxies we tested each day of our experiment and how many of them were found to be alive and working.

the number of new proxies found, meaning that the particular site did not refresh its list with new proxies.

4.1.2 Non Working Proxies

As we mentioned before from the proxies we found to be alive only 39.38% downloaded at least one of our sites successfully. Thus we looked into what response Selenium and Firefox was bringing for those non working proxies. In certain cases there was no response or it was just a blank page. However 16,336 of these non working proxies generated error messages related to their causes of failure.

We were able to look into these messages since a lot of them were similar across proxies and the categories we made for them can be seen in Table 4.2. We discovered that most of the proxies (92.63%) were not working due to internal problems they encountered while trying to send our request. Among socket and network errors the most common ones in this category were the DNS errors. In most cases the proxy had trouble getting an answer from the DNS server regarding the location of the site we were looking for.

The second biggest category (6.65%) had to do with error messages from proxies that were not allowing us to use them such as ”Authentication Required” or ”Access Denied”. Proxies in this category authenticated their users by their IP address or required some kind of other authentication meaning they were not for free. The missclassified category is related to cases where they were not HTTP proxies but something else (e.g Tor exit nodes) and the partial downloads is for proxies that did not manage to make a complete download of any of our sites.

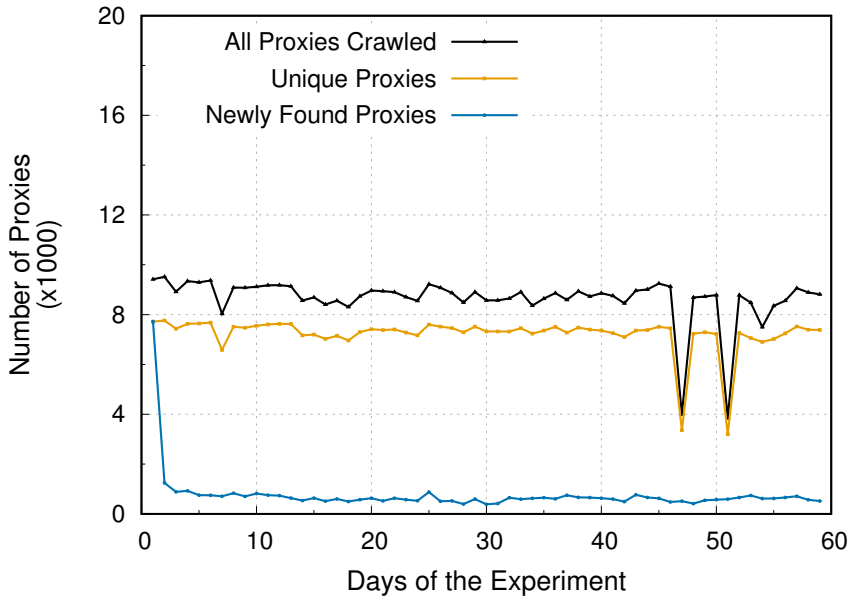

Figure 4.2: Number of proxies our crawler automatically collected each day and how many of them were new to our dataset.

TYPES OF FAILURE	NUMBER OF PROXIES
Total Non working proxies investigated:	16336
Proxies failing due to various internal errors:	15133
Proxies not allowing us to use them:	1087
Proxies misclassified::	89
Proxies partial downloads:	27

Table 4.2: Results from the error messages collected from non working proxies

4.1.3 Blacklists

A question that arises regarding the HTTP proxies is whether they are also involved in other malicious activities. To that end, we examined if they are included in any online blacklists of sites that have been observed to engage in malicious activity. To accomplish that, we leveraged *dnsbllookup.com*, a service that checks if an IP address exists in 66 DNS-based blackhole lists (DNSBL). These lists contain IP addresses associated with compromised or otherwise malicious systems that perform illicit activities. These lists are also commonly used by email services to identify and block spam and abuse.

As shown in Figure 4.3, most proxies are indeed included in at least one DNSBL, with only 26.13% of them not being found in any of the lists. We believe it is safe to assume that a portion of the proxies are likely to be hosted on compromised computers that are also involved in other malicious activities, and are probably parts of botnets.

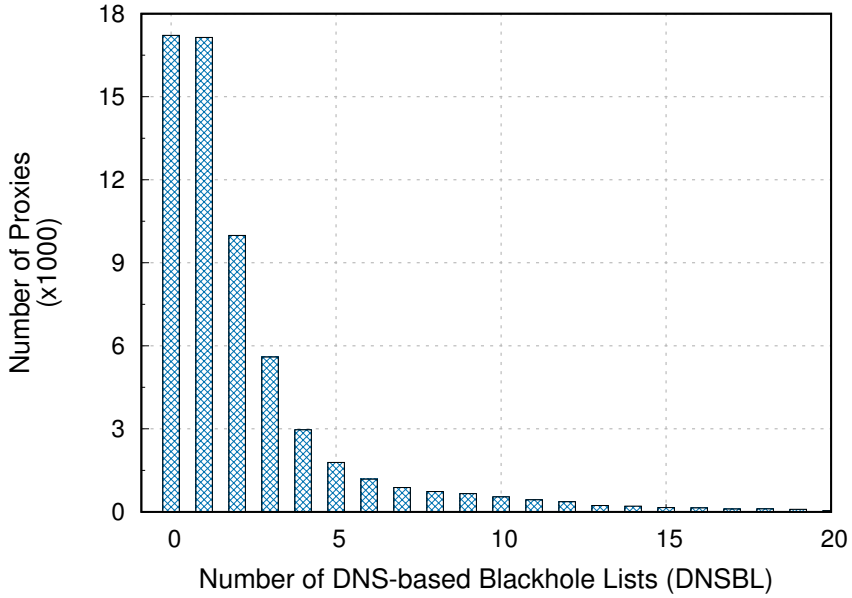

Figure 4.3: Appearance of proxies in DNSBL lists.

4.1.4 Lifetime and Reliability

In this section we analyze further the behavior of proxies. To see the longterm lifespan of the proxies we visualized Figure 4.4 where we can see the life expectancy of the 12,078 proxies we discovered on the first day of our experiment for the duration of our experiment. In is clear that while in the first day 46.40% of those alive are working after one and two months it drops to 21,40% and 23,08% respectively. The percentage of alive proxies starts at 70.25% then drops to 52.06% after a month and at the end its at 44.93%.

For Figure 4.6 we counted the percentage of hours we found each proxy alive from the day we knew of its existence. We only took into account the proxies that we had discovered to be alive at least once in the duration of our experiment. It can be seen that 40% of our proxies are less than 10% of the hours we observed them alive, while 20% of them are almost always alive (more than 90%). This shows us that while most of the proxies are unstable a minority of them offer their services continuously through time.

Another part of proxy behavior we wanted to study is their reliability meaning how often they go from being alive to being dead. For Figure 4.5 we only took into account the proxies we found to be alive at least once. Then we checked their daily logs we had from probing them 22 times per day to discover how often they change status from being alive to dead and visa versa. Then after finding the sum of status changes for each proxy we divided it with the days we knew it to be alive and probed it. In the figure it can be seen that only a small percentage of proxies (14.75%) is likely to change its status often.

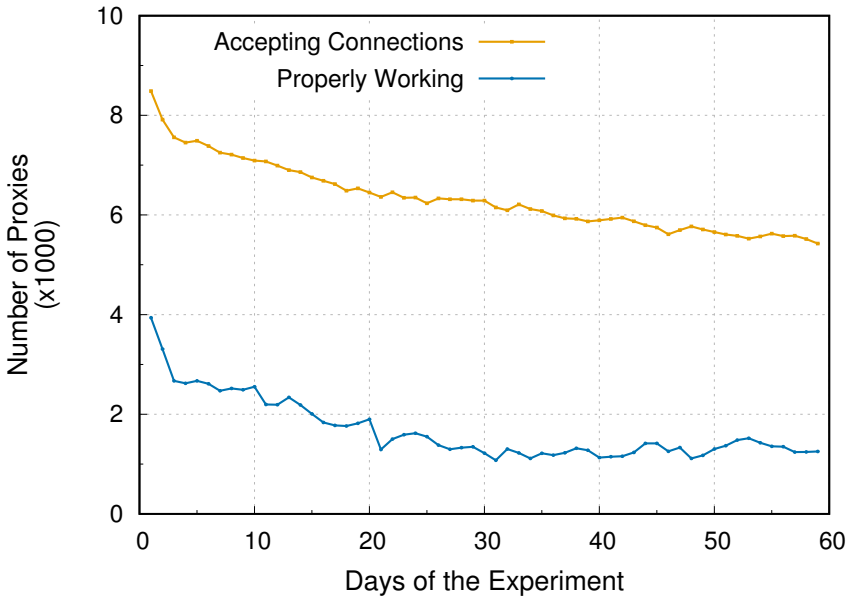

Figure 4.4: Lifetime of the proxies we discovered on the first day for the duration of our experiment .

4.1.5 Bought Shared Proxies

Some sites offered public shared HTTP proxies in exchange for money. We bought access to one of those sites that claimed to own the most proxies (over 50 thousand) to be able to make comparisons with the proxies we found from other sources. We gained access to the site's proxy lists for a month and we visited it 6 times in total every 5 days to collect proxies.

We discovered that the site in reality only offered to us about 2 thousand proxies with each visit and claimed that only these were available from their database at the time. In total we received 7,436 unique proxies and only 3,278 of them were not already known to us from other sites.

As it can be seen in Table 4.3 the percentage of working proxies is the same at the bought and not bought proxies at 30.20% and 25.62% respectively. The percentage of malicious proxies is the same at both categories at 6.16% and 5.82%. Interestingly the proxies found in both groups have a high working rate (84.41%) and the smallest malicious rate (2.02%).

4.1.6 Origins

To get some further insight about our proxies we checked them against "ip-api.com" which offered us information on the proxies regarding their country, isp and autonomous system they belong. In Table 4.4 we present the most common autonomous systems we found proxies to belong to in the wild. From the 65,871 proxies we crawled in total, we

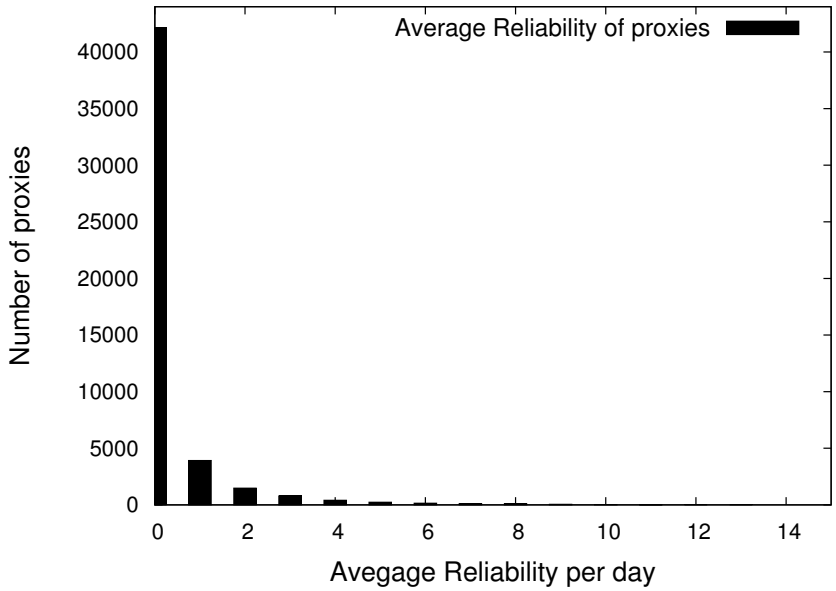

Figure 4.5: Status changes of HTTP proxies per day.

TYPE	TOTAL	WORKING	MALICIOUS
Not Bought:	58435	14973	872
Common:	4158	3510	71
Bought:	3278	990	61

Table 4.3: Comparison between shared proxies we found in the wild for free and those we purchased.

found 60,708 unique IP addresses. Thus 11.25% of the total proxies we found belong to the same AS in China. The AS from China, USA and Russia that can be seen in the table account for the 33.74% of our total proxies are located.

As our main focus in on the analysis of malicious proxies we build Table 4.5 to draw some possible correlation between the behavior of proxies and their location. It can be seen that Chinese AS dominate our list with 50.79% of our malicious proxies belonging to three on the table. Interestingly another big batch of our malicious proxies 23.3% belong to an Indonesian AS that is also visible in Table 4.4. Evidently proxies from China and Indonesia while being most common in the wild have also greater chances to be malicious than a proxy located to USA or Russia.

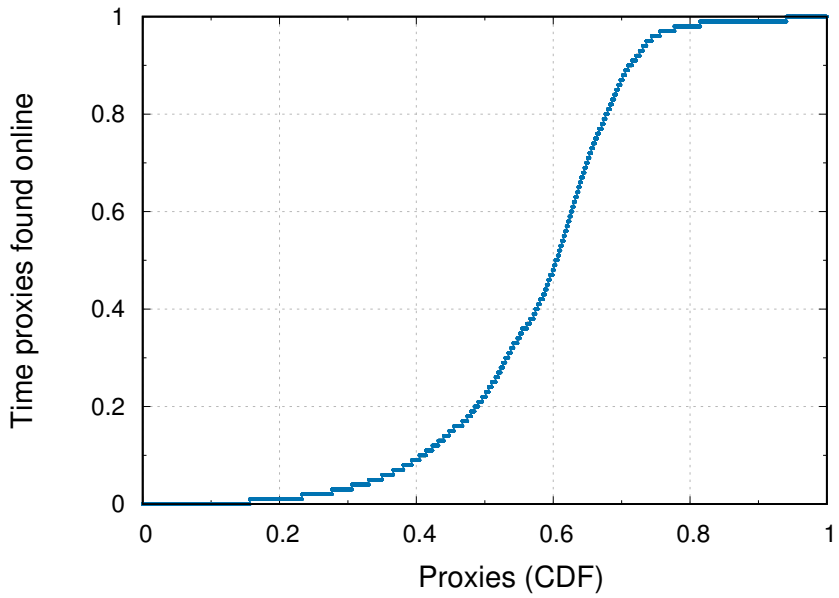

Figure 4.6: CDF of hourly lifetime of proxies.

4.2 Malicious Proxies

In this section we focus our analysis on the malicious proxies that were detected. Specifically, we first examine the variations of size of the injected HTML pages by the malicious proxies, and then study their behavior in terms of the requests sent to third-party domains due to the injected content. We also look into proxies that modify their injections based on the content they found.

4.2.1 Fetched Content Size Variation

A basic way malicious proxies differ from legitimate ones is in relation to the size of the downloaded HTML file. Since new code is typically injected, one would expect the size of the downloaded content to always be larger than the original. As discussed in the previous section, however, proxies that only block certain kinds of content, such as trackers and ads, which cannot be considered as malicious, may actually result in size reduction. Another type of proxies that we observed are proxies that block trackers and ad networks only to replace them with their own. Thus a downloaded HTML file belonging to these types of proxies does not have to differ much in size of that of the original page. This effect would be more evident for rich web pages that already contain trackers and ads, similar to our own honeysite h_2 (in order for the proxy to be able to remove some content, and thus reduce the size of the retrieved page)

Figure 4.7 shows the distribution of the downloaded content size when retrieving our most complex honeysite h_2, which has dynamic content due to the inclusion of trackers and the fake ad networks. The expected download size for honeysite h_2 ranges around

-	AUTONOMOUS SYSTEM	PROXIES	COUNTRY	OWNER
1	AS4837	6834	CHN	China Unicom
2	AS4134	4887	CHN	China Telecom
3	AS13335	2578	USA	CloudFlare
4	AS8048	2372	VEN	Cantv
5	AS17974	1830	IDN	PT Telkom Indonesia
6	AS50896	1399	RUS	MediaServicePlus LLC
7	AS200557	1399	RUS	Petersburg Internet Network
8	AS15169	1272	USA	Google Cloud
9	AS15003	1076	USA	SpeedVM Network Group
10	AS14061	1043	USA	Digital Ocean

Table 4.4: Autonomous Systems in which most of HTTP proxies are located (top 10 Autonomous Systems).

-	AUTONOMOUS SYSTEM	PROXIES	COUNTRY	OWNER
1	AS4837	433	CHN	China Unicom
2	AS17974	234	IDN	PT Telkom Indonesia
3	AS4134	61	CHN	China Telecom
4	AS56041	16	CHN	China Mobile Guangdong
5	AS131269	14	IND	PT Beam Telecom country

Table 4.5: Autonomous Systems in which most of malicious HTTP proxies are located (top 5 Autonomous Systems).

21KB. It can be seen that both legitimate and malicious proxies have downloads falling within the expected size of the page, while also both retrieve fewer content, mostly due to proxies that block tracking and advertising content. Approximately half of the malicious proxies have downloads with bigger size than expected while about 20% of them double or more the size of the page.

4.2.2 External Domains

In order to track the culprits of the identified injections and gain a better understanding of the behavior of the injected content, we manually ran the code injected by the malicious proxies to the fetched HTML content. We ran this experiment locally, in a controlled environment, and we kept track of all the contacted IP addresses and external domains. In order to accomplish this, we set our own trusted web proxy for logging all HTTP requests and forwarding the traffic.

Additionally, we used a whitelist that contained all the domains that were expected

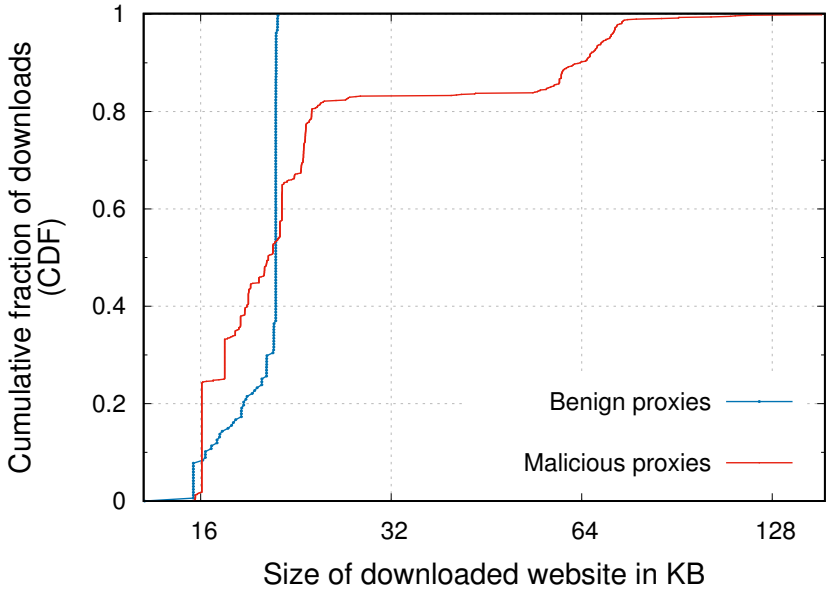

Figure 4.7: CDF of the downloaded content size when fetching honeysite h2.

to be contacted by our honeysites, thus identifying the requests sent due to maliciously injected code. Two limitations of this approach are that some of the requests to such domains were no longer working, and that our whitelist had the domains of the fake ad networks we used, which means that any requests from proxy-injected content to these domains would be ignored.

By following this methodology we were able to collect information for all the domains the injected code requests, as well as the IP addresses that corresponded to these domains. Table 4.6 shows a list of the top-20 most contacted domains by injected code. In total we observed requests towards more than 362 different domains.

Additionally, we identified that about half (181) of the domains were contacted by 5 or more malicious proxies. We identified 98 domains that were contacted only by a single proxy each, while the most popular domain (tongji.baidu.com) was requested by the content injected from 556 different proxies. We discuss the purpose of these injected domains in detail in Section 5. Based on this data, we observe that in most cases different proxies contact the same domains, which may indicate that the use the same tools for launching their attacks.

4.2.3 Content-Dependent Injections

After following the previously described methodology for detecting the malicious proxies and documenting their behavior, we wanted to investigate in more depth if the exhibited content modification behavior changes depending on the content of the fetched website. Here we will present our results from our honeysites methodology like we explained in Section 3.2. By comparing the domains the proxies repeatedly (as in more than once)

-	DOMAIN	PROXIES
1	tongji.baidu.com	556
2	cfs.uzone.id	140
3	a01.uadexchange.com	124
4	up.filmkaynagi.com	113
5	a.akamaihd.net	109
6	urlvalidation.com	107
7	i.qkntjs.info	106
8	adnotbad.com	106
9	ratexchange.net	105
10	i.tonginjs.info	105
11	www.onclickcool.com	104
12	agm.abounddinged.com	104
13	yellow-elite.men	103
14	qnp.demisedcolonnaded.com	102
15	intext.nav-links.com	102
16	www.tr553.com	101
17	ruu.outputsteddy.com	101
18	s.lm15d.com	74
19	rtax.criteo.com	72
20	www.donation-tools.org	69

Table 4.6: Most requested third party domains from malicious proxies.

requested for each honeysite, we found 37 proxies requesting 63 domains in total for (h_2) but not (h_1). This behavior confirms our hypothesis that proxies change their behavior depending on the content they encounter.

More specifically we detected 10 proxies that injected both our honeysites and real site but injected an unique domain inside the iframe of the adsense ad for (h_2) and the real site. The injection seems to try to bring an ad where the adsense ad should have been.

From the 362 domains injected in total in our honeysites we only found 48 of them ”missing” from our real site downloads. The missing domains could be explained by the fact that proxies request different domains if they request an ad, or change their injection based on content. This proves that these injections exist in the real world.

Since our methodology could not track changes to dynamic elements in the DOM tree we created a second script to verify whether the proxies attempted to modify the advertising IDS of any of the fake ads we had in our (h_2). We detected two proxies targeting our advertisement from Media.net and successfully changing our publisher's ID with theirs in 9 and 11 times respectively. Interestingly these two proxies did not inject anything else on the page or our other honeysite and went by undetected from our system, the first one even used two different publisher's IDS on different occasions when it replaced ours.

Surprisingly they did not always perform the replacement in (h_2). The first one injected the Media.net iframe for 9 days in a row and then the next two days it fetched the page with no modifications. While the second one brought a ”normal” version of the page for 18 days and then for the next 11 days it modified the ad. The next day after that that we found him to be working for the last time it reversed back to being benigh.

Chapter 5

Rogue Proxy Behaviors

In this section we thoroughly discuss the type of content malicious proxies inject to relayed traffic. We have already concluded our analysis to unique clusters (see Section 3.4), which are characterized by some common practices in injecting content (e.g., by leveraging a particular code template). In this section, we further discuss the semantic nature of each injection, for shedding more light to the mechanics of a malicious proxy's operation. The following analysis had to overcome many practical challenges, such as non-existing links, obfuscated JavaScript, etc.

5.1 Generic Injections

Advertisements: Malicious proxies often inject ads. We perform a manual analysis by visually inspecting the output of a web site (when the proxy is enabled and when it is not) for determining what type of ads are injected. Several advertisements are displayed in a language compatible with the proxy's geographical location, while sometimes ads can be displayed in a language compatible with the client's location. Certain ads exercise a legitimate behavior, without tricking the user into clicking them. Some of them completely cover the target web site, while others are much more aggressive, embedding videos from YouTube. For example, *Fyne.in* covers the whole page and displays video messages, while several others display fake rewards and alerts instructing the user to fix their infected computer. A fraction of advertisements embed adult content and services for meeting people at the client's geographical area. We clicked some of the ads and found that they could potentially redirect you to sites that have been already blocked by Google safe browsing and identified as malware distributors.

Tracking: As described in Section 3, we recorded all requests made with Firefox/-Firebug, which are associated with a particular proxy injection. We observed that injected domains attach third-party cookies to users' requests. In some cases, these cookies share the same `ID`, which means that some of the injected libraries are cooperating to better track the user through cookie syncing [14]. Malicious proxies were found to inject JavaScript files from Mixpanel [1], which is a library used to track user actions in websites.

Fingerprinting: Malicious proxies often attempt to maintain tabs open on the user's browser for launching fingerprinting attacks. Certain injected content was found to make requests to third-parties which emit JavaScript files in the HTML code of the page that uses fingerprintjs2 [2], which is a well known library for fingerprinting. In a similar fashion, libraries that perform canvas fingerprinting [20] were also detected.

Privacy leaks: Malicious proxies can easily inject scripts, which, once rendered, trigger requests to third-parties for exfiltrating sensitive user information. This information may include static fields (like geolocation data) or may be associated with the product of a fingerprinting methodology. For example, we found on a regular basis an injection, emitting requests to third parties affiliated with the ad market. Third parties like: *BlueKai* and *eyeota.com* which are a data management platform and a data supplier. We also uncovered injection libraries that target sensitive user information that exists in certain pages the user may or may not visit, as well as JavaScript that, based on regular expressions, harvests e-mails and phone numbers. In rare cases, we identified scripts that search for specific DOM elements of popular web pages (like QZone, a Chinese social network) for extracting personal information, such as the user's blood type. Therefore, we can undoubtedly infer that malicious proxies and their injection libraries are seriously affecting users' privacy.

Malware: Several injected libraries contain JavaScript code that triggers requests towards sites that are blacklisted and blocked by Google safe browsing. Upon inspection we found that malicious proxies often attempt to land malware on the user's host or redirect the browser to a drive-by download page [23].

Unclassified behavior: We had difficulties in understanding certain code injections in a small fraction of cases. This is mainly because of recorded requests towards targets that were no longer valid during the analysis phase. In certain cases, proxies inject a `script` or an `iframe` element with no target (i.e., `src` is empty). This may be due to a template framework used, which mistakenly or on purpose adds this element, while a higher-level script chooses when to include a payload or not. Also, in some other cases instead of fetching data, injected scripts use resources from the local storage of the browser. Proxies of all the aforementioned cases are marked as *suspicious* and they are excluded from the set of the malicious proxies, which perform attacks, such as fingerprinting and tracking.

We summarize all different types of injections performed by malicious proxies in Figure 5.1. We have grouped injections according to the high-level semantic goal of the malicious proxy (i.e., fingerprinting, device tracking). A particular proxy can exist in multiple categories (i.e., can be counted in multiple bars in Figure 5.1), since the proxy can launch several malicious attacks at a time. Also, note, that *suspicious* proxies are not counted at all, unless they simultaneously perform additional attacks.

As projected in Figure 5.1, we have discovered that 472 of our 1004 malicious proxies inject images in the page served to the user. Some of these images are brought by legitimate ad networks. We consider a proxy to be tracking user information if they attempt to extract information on the device, the operating system, the browser of the user, or a combination of those. Such behavior can be detected by looking for functions in the injected JS code that extract information on the language and timezone of the user, as well as the user agent, system language, platform, plugins, width and height of the screen, device

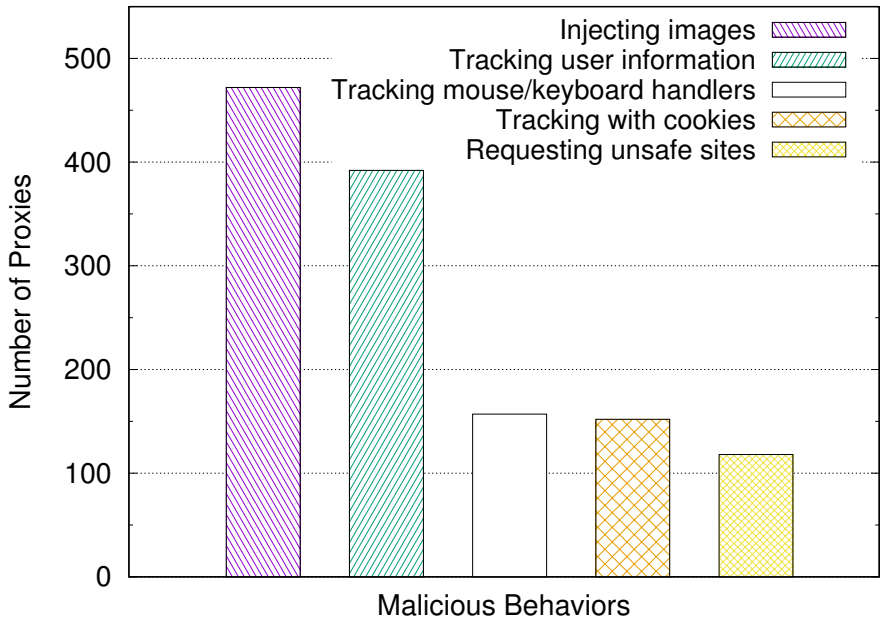

Figure 5.1: High-level categorization of malicious proxy behavior.

pixel ratio, color depth and others. We also include cases where the proxies use known tools like `fingerprintjs`. In total we discovered 392 proxies guilty of this behavior. We looked in the injected JS code for functions related to tracking mouse and and keyboard movements as well as those responsible for cookies. We found 157 and 152 proxies that call these functions respectively. Moreover, our set contains 118 proxies that make requests to sites blocked by Google's safe browsing potentially for installing malware on the user's host, as well as malicious behavior that belongs in the other categories shown in our Figure.

5.2 Interesting Injections

As we have discussed so far, in the majority of cases, malicious proxies inject scripts that interact with third-party sites. It is interesting to explore the distribution and the popularity of the domains used by these third-parties. We present a list of the most common extracted domains that were contacted through injected scripts in Table 4.6.

Our most accessed domain is owned by a popular Chinese search engine, namely Baidu. These are *baidu.com* and *tongji.baidu.com*; the latter is a traffic analysis service similar to Google analytics. In this case, a `div` element is injected in the DOM tree. and tries to load *tongji.baidu.com/logstat.swf*. The `allowscriptaccess` tag is set to *always* meaning that the `.sfw` files are granted access to the HTML DOM regardless of where they are hosted. Almost half of the malicious proxies we detected were found to be in contact with that domain. *cfs.uzone.id* and *a01.uadexchange.com* belong in the same

cluster of injections. In this case a `script` element is injected at the end of the DOM tree. This element contains a JS function that brings advertisements.

The rest of the domains presented in our table all have a common factor. They all resulted from injected JS code in the page served to the user. We tracked this code and we were able to find it used in injections by 141 different malicious proxies, some of them used different variations of the code resulting in different domains being requested but the core stayed the same. The code has a lot of functions and can be used as a toolkit. Those functions have different 'roles' such as setting and getting cookies, getting information from the user (browser, os, etc) that can be used to fingerprint the user, deploying advertisements, hashing, sending AJAX requests and others. 113 of the proxies contact *up.filmkaynagi.com* which according to Google safe browsing contains harmfull content.

ratexchange.net loads JS code and injects the page depending on the site the user has visited. For example if the user visits *ok.ru* (Russian social network) it injects with ads from *cdnpps.us* with the attackers publisher id. Alternatively if it detects the user using the *Yahoo search engine* it hijacks the url and redirects to *sugabit.net* which is a site that looks like a search engine but is a browser hijacker. Similar attacks exist for other different hostnames.

Another interesting cluster of malicious proxies fetches JS code from *d.chaoliangyun.com*. It strips all of the users HTTPS connections and tries to extract information about the user as well as possible information the page may contain like phone numbers or e-mails by using special regexes. Additionally it monitors the sites the user visits and searches for site specific information that could reveal very sensitive to the user information.

Chapter 6

Discussion and Limitations

The work presented here is our initial attempt to assess the extent of malicious content modification by rogue open HTTP proxies. We opted for a simple but effective scheme that relies on decoy web pages under our control to detect potential content modification without false positives. This approach, however, is not robust against determined rogue proxy operators who may anticipate our probes, and refrain from performing any suspicious activity. Currently, detecting our probes is relatively easy, since we use the same set of honeysites hosted on a single server, making them easily identifiable. A first step for mitigating this issue is to significantly expand our set of honeysites, and randomly expose only a few of them to each proxy.

Eventually, however, a more generic strategy is needed against sophisticated proxies who may choose to target only a subset of pages, e.g., only those containing ads from certain ad networks, high-profile and popular websites, websites of certain interests, and so on. Consequently, a broad, diverse, and constantly evolving set of pages should be used as targets of probe requests to achieve the required stealthiness and coverage.

Although we used the proposed probing scheme as part of a large-scale measurement effort, the core content modification logic in conjunction with honeysites can also be used as a standalone tool, for on-demand testing of any proxy that users are about to trust their traffic with. Using our existing implementation our crawler keeps visiting the proxy sites daily to keep our information up to date. Then we test these proxies against our honeysites and compare their downloads to our own static version.

As an extra contribution of this work we publish the proxies we find to be safe in our site for anyone to use. In order to offer usable proxies we have decreased the time we wait for the honeypage to load to 30 seconds. A limitation of our methodology is that as we show in the clustering section we will "discard" proxies that modify the page but are not found to be doing anything malicious.

Chapter 7

Conclusion

In this work, we carried out a large-scale analysis of open HTTP proxies, focusing on the detection of rogue proxies that engage in malicious or unwanted content modification. During a period of two months, we made multiple requests to 65,871 proxies, requesting honeysites we created, and developed a DOM comparison method to detect HTML alterations from third parties. Our findings suggest that 38.21% of the working proxies modify the page in some way while 5.15% of the proxies we tested performed malicious injections. We presented a detailed analysis of the characteristics of legitimate and malicious proxies, and identified and analyzed various forms of malicious injections. Our results indicate the important privacy implications of trusting user traffic to unknown open proxies, operated under questionable motives. Among the observed behaviors, we encountered many privacy compromising incidents, with many proxies collecting and sharing private user information with third parties, let alone making requests to pages that push malware.

Bibliography

[1] https://mixpanel.com/.

[2] https://github.com/Valve/fingerprintjs2.

[3] How hackers abused Tor to rob blockchain, steal bitcoin, target private email and get away with it, February 2015. http://www.forbes.com/sites/thomasbrewster/2015/02/24/blockchain-and-darknet-hacks-lead-to-epic-bitcoin-losses/.

[4] Chema Alonso and Manu T̈he Sur.̇ Owning bad guys { & Mafia } with javascript botnets. DEF CON 20, 2012.

[5] Sumayah A. Alrwais, Alexandre Gerber, Christopher W. Dunn, Oliver Spatscheck, Minaxi Gupta, and Eric Osterweil. Dissecting ghost clicks: Ad fraud via misdirected human clicks. In *Proceedings of the 28th Annual Computer Security Applications Conference (ACSAC)*, 2012.

[6] Jacob Appelbaum, Marsh Ray, Karl Koscher, and Ian Finder. vpwns: Virtual pwned networks. In *Presented as part of the 2nd USENIX Workshop on Free and Open Communications on the Internet*, 2012.

[7] Sajjad Arshad, Amin Kharraz, and William Robertson. Identifying extension-based ad injection via fine-grained web content provenance. In *Research in Attacks, Intrusions, and Defenses - 19th International Symposium, RAID 2016, Paris, France, September 19-21, 2016, Proceedings*, pages 415–436, 2016.

[8] Sajjad Arshad, Amin Kharraz, and William Robertson. Include me out: In-browser detection of malicious third-party content inclusions. In *Proceedings of the 20th International Conference on Financial Cryptography and Data Security (FC)*, 2016.

[9] Sambuddho Chakravarty, Georgios Portokalidis, Michalis Polychronakis, and Angelos D. Keromytis. Detecting traffic snooping in Tor using decoys. In *Proceedings of the 14th International Symposium on Recent Advances in Intrusion Detection (RAID)*, pages 222–241, September 2011.

[10] Taejoong Chung, David Choffnes, and Alan Mislove. Tunneling for transparency: A large-scale analysis of end-to-end violations in the internet. In *Proceedings of the*

2016 ACM SIGCOMM Conference on Internet Measurement Conference, IMC '16. ACM, 2016.

[11] Vacha Dave, Saikat Guha, and Yin Zhang. Measuring and fingerprinting click-spam in ad networks. *SIGCOMM Comput. Commun. Rev.*, 42(4):175–186, August 2012.

[12] Xavier de Carné de Carnavalet and Mohammad Mannan. Killed by proxy: Analyzing client-end tls interception software. In *23nd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24*, 2016.

[13] Sean Ford, Marco Cova, Christopher Kruegel, and Giovanni Vigna. Analyzing and detecting malicious flash advertisements. In *Proceedings of the 2009 Annual Computer Security Applications Conference*, ACSAC '09, pages 363–372, Washington, DC, USA, 2009. IEEE Computer Society.

[14] Arpita Ghosh and Aaron Roth. Selling privacy at auction. In *Proceedings of the 12th ACM Conference on Electronic Commerce*, EC '11, pages 199–208, New York, NY, USA, 2011. ACM.

[15] Hamed Haddadi. Fighting online click-fraud using bluff ads. *SIGCOMM Comput. Commun. Rev.*, 40(2):21–25, April 2010.

[16] Christian Haschek. Why are free proxies free?, 2015. `https://blog.haschek.at/post/fd9bc`.

[17] Ralph Holz, Thomas Riedmaier, Nils Kammenhuber, and Georg Carle. X.509 forensics: Detecting and localising the SSL/TLS men-in-the-middle. In *Computer Security - ESORICS 2012 - 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings*, pages 217–234, 2012.

[18] Zhou Li, Kehuan Zhang, Yinglian Xie, Fang Yu, and XiaoFeng Wang. Knowing your enemy: Understanding and detecting malicious web advertising. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 674–686, New York, NY, USA, 2012. ACM.

[19] Moxie Marlinspike. sslstrip. `http://www.thoughtcrime.org/software/sslstrip/`.

[20] Keaton Mowery and Hovav Shacham. Pixel perfect: Fingerprinting canvas in HTML5. In Matt Fredrikson, editor, *Proceedings of W2SP 2012*. IEEE Computer Society, 2012.

[21] Mark O'Neill, Scott Ruoti, Kent E. Seamons, and Daniel Zappala. TLS proxies: Friend or foe? *CoRR*, abs/1407.7146, 2014.

[22] Josh Pitts. The case of the modified binaries, 2014. `http://www.leviathansecurity.com/blog/the-case-of-the-modified-binaries/`.

[23] Niels Provos, Panayiotis Mavrommatis, Moheeb Abu Rajab, and Fabian Monrose. All your iframes point to us. In *Proceedings of the 17th Conference on Security Symposium*, SS'08, pages 1–15, Berkeley, CA, USA, 2008. USENIX Association.

[24] Charles Reis, Steven D. Gribble, Tadayoshi Kohno, and Nicholas C. Weaver. Detecting in-flight page changes with web tripwires. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, NSDI'08, pages 31–44, Berkeley, CA, USA, 2008. USENIX Association.

[25] Len Sassaman. The faithless endpoint: How Tor puts certain users at greater risk. Technical Report ESAT-COSIC 2007-003, Katholieke Universiteit Leuven, 2007.

[26] Bruce Schneier. The further democratization of QUANTUM, April 2015. `https://www.schneier.com/blog/archives/2015/04/the_further_dem.html`.

[27] Brett Stone-Gross, Ryan Stevens, Apostolis Zarras, Richard Kemmerer, Chris Kruegel, and Giovanni Vigna. Understanding fraudulent activities in online ad exchanges. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, pages 279–294, New York, NY, USA, 2011. ACM.

[28] Kurt Thomas, Elie Bursztein, Chris Grier, Grant Ho, Nav Jagpal, Alexandros Kapravelos, Damon Mccoy, Antonio Nappa, Vern Paxson, Paul Pearce, Niels Provos, and Moheeb Abu Rajab. Ad injection at scale: Assessing deceptive advertisement modifications. In *Proceedings of the 2015 IEEE Symposium on Security and Privacy*, SP '15, pages 151–167, Washington, DC, USA, 2015. IEEE Computer Society.

[29] Neenav Vratonjic, Julien Freudiger, and Jean-Pierre Hubaux. Integrity of the web content: The case of online advertising. In *Proceedings of the 2010 International Conference on Collaborative Methods for Security and Privacy*, CollSec'10, pages 2–2, Berkeley, CA, USA, 2010. USENIX Association.

[30] Nicholas Weaver. In contempt of bulk surveillance: It's too easy, September 2015. `https://lawfareblog.com/contempt-bulk-surveillance-its-too-easy`.

[31] Nicholas Weaver, Christian Kreibich, Martin Dam, and Vern Paxson. Here be web proxies. In *Proceedings of the 15th International Conference on Passive and Active Measurement - Volume 8362*, PAM 2014, pages 183–192, New York, NY, USA, 2014. Springer-Verlag New York, Inc.

[32] Philipp Winter, Richard Köwer, Martin Mulazzani, Markus Huber, Sebastian Schrittwieser, Stefan Lindskog, and Edgar Weippl. Spoiled Onions: Exposing Malicious Tor Exit Relays. In *Proceedings of the Privacy Enhancing Technologies Symposium (PETS)*, 2014.

[33] Edward J. Zaborowski. Malicious proxies. DEF CON 17, 2009.