Internet Censorship Detection: a Survey

Giuseppe Aceto and Antonio Pescapé
University of Napoli Federico II (Italy)
{giuseppe.aceto, pescape}@unina.it

***Abstract*—Internet Censorship is a phenomenon that crosses several study fields, from computer networking and computer security to social sciences; together with censorship detection and censorship circumvention it has impact on Internet infrastructure, protocols and user behaviors. Detection of Internet Censorship is the basis for the study of this phenomenon, and recently it has received focus from a technical point of view. Due to the heterogeneity of study fields and approaches, the scientific corpus on these topics is still in need of an overall analysis, based on coherent framework and lexicon to describe the experimented approaches and findings.**

In this paper we present a survey on Internet Censorship detection. We propose a reference for censoring techniques and a characterization of censoring systems, with definitions of related concepts. Based on the censoring techniques investigated in literature, we propose an analysis and discussion of censorship detection techniques and architectures and we present a chronological synopsis of the literature adopting or introducing them. Then we collect, study, and discuss available tools and platforms for censorship detection, and propose a characterization scheme to analyze and compare them. Finally, we compare and discuss detection architectures, tools and platforms, and we use the results to infer current challenges and for proposing new directions in the field of censorship detection.

***Index Terms*—Internet Censorship, Network Monitoring, Communications Surveillance, Privacy, Network Security.**

1 Introduction and Motivations

Internet Censorship is a complex phenomenon that is deeply discussed and analyzed in the field of social sciences, and in recent years has attracted attention also from other study fields such as computer security and computer networking due to the widespread adoption of ICT for information control, previously focused on analog mass media. Putting aside the social and political aspects evidently related to censorship, we focus on the technical aspects only: regardless of the aims, scope or legitimacy of it, we consider "Internet Censorship" as the intentional impairing or blocking of access to online resources and services. The design principles of the Internet as an open and distributed system contrast with the controls required by censorship. Therefore the technical means adopted to this end almost invariably imply the interference with—or disruption of—standard network protocols and expected behavior of network applications. This has practical consequences for all the stakeholders of the Internet: the end users, which are subject to restrictions and impairments with varying degrees of transparency; the ISPs, that face the complicated trade-off among complying with the law, building and managing the censorship infrastructure, and providing the best service to their customers; the transit operators, that potentially experience unexpected traffic patterns; finally the online service providers, that may have to deploy and operate censorship systems as demanded by the law of their own country, and whose global user base (up to whole countries at a time) can be subjected to impairments or complete blocking. Moreover, due to the complexity and the non-standard nature of censorship techniques, unforeseen side effects can strike third parties (as actually already happened [110]). Summarizing, even if adopted for legitimate and embraceable aims censorship requires mangling of several components of the Internet and this has an impact on all its stakeholders. Several systems for *circumvention* and *detection* of Internet Censorship have been developed over the years; these too are of interest for the different Internet stakeholders, according to their roles. In fact surveillance needs to recognize the related traffic, and prevent both false negatives (when *circumvention* is effective) and false positives (when *censorship detection* triggers the alarms); users and online service providers may be interested in *circumvention* techniques to prevent side effects or unlawful interference (besides illicitly eluding the restrictions). In addition to the aforementioned reasons, censorship *detection* in its turn is of central importance for different actors. For academy and industry researchers, the study and employment of censorship *detection* is functional to understanding if, to what extent, and with which method censorship is enforced. Significant aspects of censorship, such as its enforceability, effectiveness, and transparency, as well as the possible unwanted side effects, strongly depend on the technical details of the adopted censorship technique and thus evolve with the technology and real usage of it. For the creators of circumvention systems, the mechanics of censorship revealed by *detection* are at the basis of the design and development of their tools. Finally, for the operators and users that perform network diagnostics, the *detection* of censorship can provide the explanation for apparent outages and malfunctioning, discharging the inculpable application, network administrator, ISP, or online service provider. In brief, for many different actors censorship *detection* is either very valuable or strictly necessary.

Despite this, to the best of our knowledge, no peer-reviewed survey has investigated such topics, moreover no survey is available that specifically addresses Internet Censorship *detection*. Previous works have surveyed and analyzed Internet Cen-

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

sorship and circumvention techniques and tools (Leberknight et al. [99], Elahi and Goldberg [50]). An analysis of the different phases of the application of Internet Censorship and the citizens' perception and reaction to it is presented by Bambauer [16]. The studies on Internet Censorship and papers proposing detection and circumvention techniques often report a technical analysis of selected censorship techniques and related works; these however are not meant to be surveys, and thus are selective or partial, do not adopt a shared lexicon, and suffer from the diversity of venues and goals characterizing the papers.

In this paper we aim to fill this gap in the literature by means of this survey, focused on Internet Censorship *detection* techniques and architectures, and their implementations in tools and platforms. We also propose and adopt a coherent reference framework for the description of Internet Censorship enforcement techniques and for the characterization of censoring systems. In studying the literature and the available tools and platforms for censorship detection we have considered top conferences and journals in the field of computer networks and computer security, searching for works discussing Internet *censorship*, censorship *detection* and censorship *circumvention*; without the aim of being exhaustive, we have adopted an inclusive approach considering also minor venues and technical reports when we deemed the contribution worth mentioning. From the selected literature we have derived the references to the analyses of censorship systems and case studies, and techniques, architectures, tools and platforms used for the task of censorship *detection*. Then each reference has been analyzed to extract and discuss, where applicable: the considered censoring techniques, the adopted *detection* techniques, tools, and platforms, and the proposed architectures. In the analysis and throughout the survey we have considered as *architectures* the detection systems for which only a structural and functional description is provided, and there is no publicly available implementation (possibly excluding proof-of-concept or study-specific prototypes). In the case an implementation is available, we adopt the term *tool* when the aim of the application is on one-spot measurement and can be operated from a local installation, and call *platform* an application that automatically takes care also of a number of accessory tasks, and is usually distributed in nature, requiring administrative access to multiple hosts. For tools and platforms not presented in academic papers, the online documentation has been leveraged, and when available also the application itself or its source code have been analyzed. Collectively we refer to architectures, platforms and tools devoted to detection of Internet Censorship as "detection systems".

With this survey we aim to fill a gap in academic literature regarding Internet Censorship *detection*, also offering a reference frame for the description of Internet censorship systems. The outcome of our research and the contributions of this work can be summarized with:

- a reference description of censorship techniques and systems;
- a survey of censorship *detection* systems;
- a characterization frame for the comparison of censorship *detection* systems;
- a discussion of considered *detection* architectures, tools and platforms;
- an analysis of challenges and future directions of censorship *detection* systems.

This survey is structured as follows. Section 2 constitutes an **overview and background** on different technical aspects of Internet Censorship and related concepts, introducing and defining the terms that will be used in the survey.

In Section 3 we provide a **reference for censoring techniques**, discussing the types of actions and evidences of censorship with the related bibliography, in order to describe the related *detection* techniques. The censoring techniques are ordered according to the phases of an (ideal) communication sequence necessary to reach an online resource: starting with path establishment by the routing protocol and ending with the resource retrieval by the application.

Building on the introduced background, definitions and references, we present a **survey of censorship *detection* techniques** in Section 4, with a subsection specifically devoted to **detection architectures** as proposed in the relevant literature. The considered papers are characterized along the lines of the communication sequence phases, as in previous section, and properties of the detection technique, and are reported as a **chronological grid** in Table I.

The whole Section 5 is devoted to tools and platforms for censorship detection whose implementation is available to the public, describing and discussing their specific peculiarities; a **comparison of available tools and platforms for censorship detection** in terms of censorship techniques detected is shown in Table II.

Section 6 presents a **characterization frame for censorship detection architectures, platforms, and tools**, based on their most relevant properties, shown graphically in Fig. 17. In the same section a comparison is provided of **censorship detection architectures, platforms, and tools** considered, as shown in Table III, adopting the characterization frame introduced before.

Finally, in Section 7, **conclusions drawn from this survey** are discussed and the inferred open challenges and future directions are proposed.

2 Background: technical aspects of Internet Censorship and definitions

Terminology in the matter of Internet Censorship is not well defined, being often ambiguous and inconsistent across different papers. The phenomenon under analysis also lacks a common definition, being named filtering, blocking, interference, tampering, surveillance, referring possibly to different aspects of it, with little or no formalisms. This does not help

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

scientific discussion on the topic and the sharing of technical advancement in detection and circumvention. Valuable contributions in the direction of clearly defining the concepts related to Internet Censorship have been provided by Verkamp and Gupta [147] and Filastò and Appelbaum [57], even though they were not driven by this goal, their approaches aiming either at reporting the state of art or providing the researchers with detection tools. One of the first technical characterization of Internet Censorship can be found in [48], where it is defined as *" [...] refusing users access to certain web pages without the cooperation of the content provider, the hosting provider and the owner of the client machine being used to access these pages"*. A somehow more generic definition is proposed by Elahi and Goldberg [50]: *"Internet censorship is the intentional suppression of information originating, flowing or stored on systems connected to the Internet where that information is relevant for decision making to some entity"*. This definition, while having merits in highlighting the *motivation* behind the censorship and its intentionality, goes beyond the scope of this survey, that is focused on the detection of censorship and has little practical use for the modeling of the censor's decision making processes.

While most research on Internet Censorship has focused on the Web, and thus on censorship of web resources, censorship has been found also on other Internet applications and services. Therefore, extending the definition provided in [48] for web pages and including what we will describe as *soft censorship* techniques, we define "Internet Censorship" the intentional impairing of a client application in communicating with its server counterpart, enforced by a third party (neither the user, nor the server operator), named hereafter "censor". The impairing can act both on the communication control and on the informative content that the communication is meant to convey. The intentionality differentiates censorship from outages and the selective censor behavior (affecting the censored resources or services and not others) is the condition necessary to detect it. We note that the adoption of a client-server terminology does not restrict the definition to applications implementing exclusively this communication paradigm, as also in peer-to-peer applications each communication sees at least one node initiating the exchange thus qualifying itself as "client" for that exchange. In this section we will provide definitions for concepts related to Internet Censorship techniques.

There are many such techniques that deliberately interfere with access to online resources; from a network topology point of view, we propose a coarse-grain classification with regards to the components of the communication system that are employed for censorship: client-based, and server-based, if censorship is applied at the ends of the communication path, network-based, if it happens in between.[1] While most of literature surveyed in this work is focused on detection of network-based censorship, in this section an overview of client-based and server-based censorship techniques is also given; moreover the concepts of self-censorship and circumvention are briefly discussed with bibliographic references providing a more comprehensive context.

[1] As previously noted, the case of peer-to-peer applications fits in this classification considering each communication, with the node initiating it acting as the client of a client-server scenario.

2.1 Definitions

In the following we define terms and concepts related to Internet Censorship that will be useful in describing censoring techniques and censorship detection techniques, elaborating on definitions (explicit or implicit) from the related literature.

target an online resource or service; it is characterized by information needed by a client application to access it: e.g., for a document its URL, or for a service its application protocol (with signatures potentially triggering keyword-based censorship) or a list of servers (characterized by hostname, IP addresses, transport port).

trigger the element or group of elements, in the client request to access the target, that cause the censoring system to act; if the trigger is absent (where possible) or substituted with a non-trigger then censorship does not happen i.e. the requested target is reachable; it is characterized by the phase of communication it affects, the levels of the network protocol stack that are involved and possibly specific protocols; implies the presence of a *surveillance device*.

surveillance device the device that analyzes the user traffic looking for *triggers*; it is characterized by the phase of communication it affects, the (topological) position in the network path, the levels of the network protocol stack that are inspected;

action the action performed by the censor to block, mangle or impair the access to the target; it is characterized, like the *trigger*, by the phase of communication it affects and the levels of the network protocol stack that are involved; implies the presence of a *censoring device* performing it.

censoring device a device that applies the censoring *action* by tampering with the communication between the user and the *target* so that it is impaired or altogether prevented; the surveillance and censoring devices can be co-located and coincide;

censoring system the *surveillance* and *censoring* devices;

symptom what the user experiences as result of the censor *action*; it can range from the access to a modified *target*, to an annoying worsening of quality of experience, to complete unreachability of the *target* possibly accompanied with an error; in case an error is provided, it characterizes the symptom with the phase of communication and the level of the network stack that appears to issue the error.

circumvention the process of nullifying the censoring *action*, i.e. accessing the unmodified *target*—or an equivalent copy—despite the presence of a censoring system; this can be done by preventing the *trigger* from being seen

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

by the *surveillance device* or by countering the effects of the *action*.

2.2 Client-based Censorship

We consider *client-based censorship* as the blocking of access to online content by means of applications running on the same system of the network application client. It can be implemented by different means, such as an independent application, akin to a *keylogger*, that terminates the applications whose keyboard input matches a blacklisted keyword—such apparently is the technology employed in Cuba, as reported in [150].[2] Another form for this kind of censorship is a network filter like parental control or company policy control enforcement filters, running as a personal firewall (see [18] for a recent survey, and [144] for a list of websites of parental control software).

Finally, it can be enforced as a modified version of the network application client itself, added with surveillance "features", as the case of the VoIP / telepresence / instant messaging application TOM-Skype (Chinese clone of Skype) analyzed by Villeneuve [148], Knockel et al. [91], and the Chinese instant messaging application *SinaUC* considered by Aase et al. [2].

The characterizing property of *client-based censorship* is that the functionalities of the censorship system are bound to the client system; this poses additional constraint to its detection, requiring tests to be performed within the eavesdropping possibilities of the censoring application: e.g., if input mimicking user actions is not provided through the client keyboard, a *keylogger* can not intercept it, and the detection test will be unable to trigger the censorship and thus detect it. On the other hand, having access to a component of the censoring system allows for more direct means of reverse-engineering (such has been the method adopted by Aase et al. [2]). Moreover, based on executable code of the surveillance or censoring application, detection methods and tools adopted to find malware presence can be applied, such as Detekt [29].

2.3 Server-based Censorship

The final node of the communication path, the server, is the component where server-based censorship is enforced, with no disruption of the communication mechanics: the censor selectively removes, hides, or impairs access to specific content directly in the server, employing management facilities provided by the service itself. The censoring action can be enforced ordering the server manager to comply with the request.

The existence of this kind of censorship is sometimes acknowledged by the Online Service Providers themselves. One such case is Google Transparency Report—Removal Requests[3] by which Google discloses a summary of requests from governments or from copyright owners to block access to specific content. While the actual removed targets are not disclosed, a categorization of removal requests is done according to the reason and the type of requester and the related statistics are provided. An independent observatory for removal requests of online content is maintained by The Berkman Center for Internet & Society, that provides search access to *Chilling Effects* [143]: an online database of complaints and removal requests classified according to several aspects among which topic, sender, recipient, and action taken.

Server-based censorship and its consequences are analyzed under the term "intermediary censorship" by Deibert [43, chap. 5]. This form of censorship is specifically hard to be analyzed, as its mechanics are internal to the service and not exposed to the users; a recent quantitative analysis of it has been performed by Zhu et al. [160], that reported several censoring evidences of different type (divided as proactive or retroactive mechanisms), and proposed hypotheses on how these mechanisms are actually enacted.

2.4 Network-based Censorship

In between the host running the application client and the host running the respective server part is where *network-based censorship* is enforced.

With respect to client-based censorship it provides the censor a much wider coverage of the network and with more efficiency, allowing the control of high number of communications through the management of a relatively few gateways or hubs (instead of one installation for each user system). On the contrary, client-based censorship implies the control of the user terminal or the compliance of the user herself, for each user subject to censorship.

Similar considerations can be done with respect to *server-based* censorship, that in turn requires control or compliance of server host managers. The relatively small number of popular services helps the censor that wants to control them, but there is the possibility that such servers are located abroad or otherwise outside of the influence of the censor, thus nor direct control nor compliance can be forced.

These comparisons highlight the pivotal importance of the detection of network-based censorship, that is the central phenomenon considered in this survey, and will be analyzed in more detail in the following sections. Unless explicitly stated differently, hereafter by "censorship" will be intended "*network-based* Internet Censorship", and similarly by "detection" will be intended "detection of *network-based* Internet Censorship".

2.5 Other related concepts

[2] A *keylogger* is an application that covertly intercepts and processes key strokes directed to other applications running on the same system. A documented case of a *keylogger* being used in conjunction with a censorship *circumvention* software is described by Marquis-Boire [107]: no direct censorship was enacted, but all text typed by the user was intercepted and reported to a remote server.

[3] http://www.google.com/transparencyreport/removals/government

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

2.5.1 Self Censorship: Users can self-restrict their possibilities of expression due to fear of punishment, retaliation, or other negative consequences. Changes in the degree of self-censorship have impact on detection techniques that are based on independent users traffic (*passive* detection techniques) that in absence of traffic that engages the censoring system are not able to detect its presence; this is an issue because the existence and the extent of this phenomenon are hard to estimate. One rare case for such analysis has been provided by the introduction of the "Real Name Policy" in South Korea in 2007: the obligation to register to online social network and (micro-)blogging systems providing one's own real name, thus allowing for personal identification and accountability for the expressed opinions. When protected by anonymity the user could neglect the consequences of her words: in absence of this protection the actual possibility of punishment can prevent her expression leading to self-censorship. The estimation of the effects of the new policy has been presented by Cho et al. [25], that indeed evidenced a global decrease of uninhibited behavior (most affected users where the ones with low usage of the online media, while heavy users seemed to be unaffected). Similar analysis has been carried by Fu et al. [61] related to the introduction of analogous regulation in China, officially launched in March 2012: the authors do not find a significant shift in the volume of microblog posts, but for some classes of users they detect a shift in the topics, moving away from politically sensitive ones, and thus infer—with caveats—a possible chilling effect on the freedom of expression specifically for the political debate.

2.5.2 Circumvention: The awareness of censorship and the progress on understanding its working details has led to the development of methods to elude it, collectively named *censorship circumvention* (just "circumvention" in the following). Besides being strongly based on results of *detection*, circumvention techniques and tools in their turn constitute a class of applications potentially subjected to censorship (applied to the websites describing them or providing them for download, or to the network nodes that compose their system or to the network protocol they adopt). Moreover some censorship *detection* systems leverage circumvention tools to have a supposedly uncensored access to the Internet to use as a Ground Truth in comparisons. Methods, tools and platforms have been specifically designed to counter censorship: in [50] a taxonomy is presented that characterizes thirty among circumvention tools, platforms and techniques according to a number of properties, including cost-benefit analysis for the actors. A recent field survey on circumvention techniques in China has been published as technical report by Robinson et al. [130]. Another valuable source for scientific literature on censorship and circumvention is the webpage "Selected Papers in Censorship" [155]. Besides papers focused on circumvention itself, often papers discussing censorship and censorship detection add also the related analysis of possible circumvention methods; e.g., in the early analysis of network-based censorship techniques Dornseif [48] cites and discusses a number of possible circumvention techniques, concluding that they are not easy to be applied for a common user; in [31] a technique is presented to circumvent a specific censorship (TCP-RST communication disruption) by identifying and ignoring the forged RST packets; a few techniques for circumvention of application-level keyword-based censorship are suggested by Crandall et al. [33]. Even if not specifically designed for censorship circumvention, anonymity technologies can and have been used to circumvent censorship: a recent survey on usage of several technologies including proxy servers, remailers, overlay networks such as JAP (Köpsell et al. [92]), I2P (Schomburg [135]), and Tor (Dingledine et al. [46]) is provided by Li et al. [102].[4] Specifically dealing with web browsing activities, a survey on "privacy enhancing" tools is presented by Ruiz-Martínez [133]. A comprehensive source for academic literature on anonymity is the structured bibliography page of the *Free Haven* project [59].

3 Censorship techniques

The techniques employed by the censors—as analyzed in the considered sources—can be characterized according to different properties: e.g., in terms of the *trigger* that initiates the censoring process (and thus implicitly the phase of communication in which the trigger is sent), the *action* itself, and the *symptom* experienced by the user. A general distinction is between *stateless* and *stateful* censoring technique or system: in the first kind the censoring *action* is performed deciding on a per-packet basis (presence of the *trigger*); in the latter the decision depends on both the information on past packets (status) and the presence of the related *trigger*: what composes a *trigger* changes over time according to the sequence of packets that are seen by the *surveillance device*. The overall censoring system can operate in a single step, or may be designed as *multi-stage*, involving a cascade of two or more devices (or software modules) processing the inspected traffic. Building upon the definitions we have given and on literature cited in the previous section we introduce hereafter a characterization of analyzed censorship techniques; a graphic overview of such characterization is depicted in Fig. 1, where the defining properties of one of the techniques are highlighted (two-stage DNS hijacking and HTTP injection, Section 3.10.2); the meaning of the characterization axes and of the links connecting them is explained in detail hereafter.

In the following the techniques are described, grouped according to the type of *action* (and the possible setups) adopted by the censor, also discussing the remaining elements. The presentation order follows the phases of an (ideal) communication sequence necessary to allow an application to retrieve a resource or access a service, and ends with *multi-stage* systems, that operate at several such phases.

[4] https://geti2p.net/en/

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

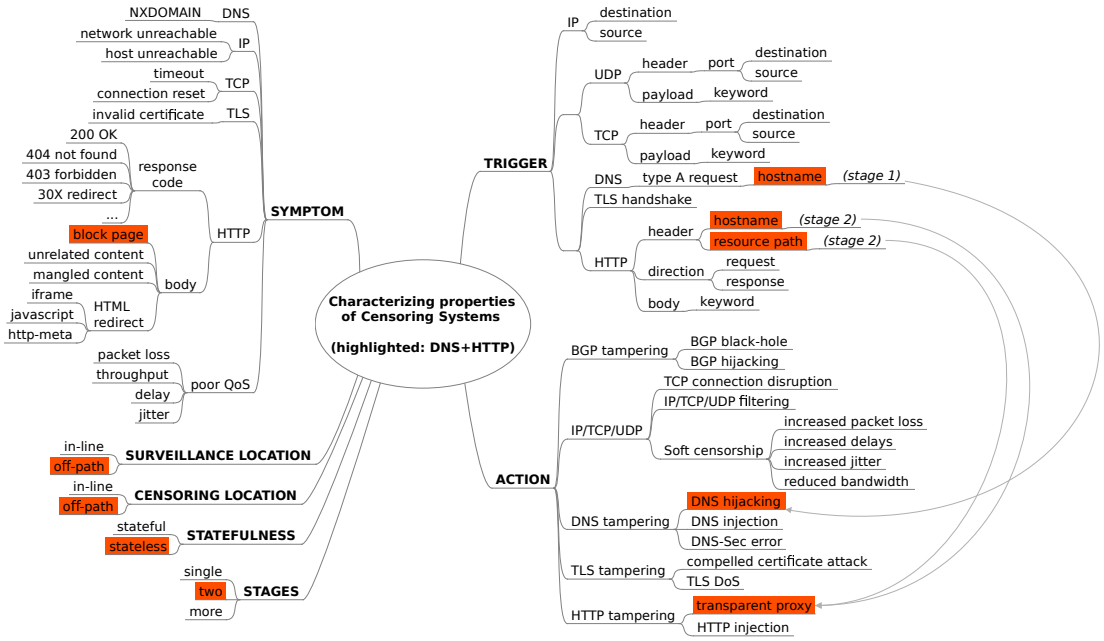

Fig. 1: Characterization of Network-based Censoring Systems. An example of two-stage technique (DNS+HTTP) is described in terms of the different properties; fulfilled ones are highlighted, with stage specification for the association *trigger-action*.

3.1 BGP tampering

Packet forwarding—the basic functionality of packet-switched networks—is performed according to criteria set by a routing algorithm. In the Internet the routing algorithm that is used by routers to coordinate across different administrative boundaries (Autonomous Systems, "AS") is the Border Gateway Protocol (BGP) [126]. Interfering with the intended functionality of the BGP protocol has potential impact to whole sub-networks up to the scale of whole countries. Such has been the case documented during the so-called "Arab Spring", the events of social unrest manifestation and civil protests occurred during the first part of 2011 in Libya and Egypt (Dainotti et al. [39]). In these cases the countries were made unreachable from outside the country ASes by withdrawing their presence from the BGP network view. When these techniques escape the control of the censor, dramatic side effects can verify, as happened in the 2009 incident that made *YouTube* unreachable also from *outside* the controlled network of Pakistan or a similar event caused by China in 2010 (Mueller [110, chap. 9]). The mechanics of such accidental misuse of BGP—that can also be intentional tampering (Feng and Guo [54])— have been studied by Ballani et al. [15], that have estimated that a significant part of the Internet was potentially subject to *prefix hijacking*: the condition in which packets that an AS T should forward to an AS B are erroneously diverted to another AS E because E advertised to T a fake shorter path towards B (see Fig. 2).

In the characterization of censorship that we have adopted, this technique presents as *trigger* the destination *or* source IP addresses; in fact by diverting to a black hole one direction of traffic consequently makes bidirectional exchange impossible: TCP connections are surely affected and only one-way traffic (notably related to malicious activities – scans, backscatter [39]) is allowed through. The *symptom* a user would experience is a `network unreachable` error in case of *prefix withdrawal*, and `time exceeded` TTL expiration in case of *prefix hijacking* that leads to loops or too long paths.

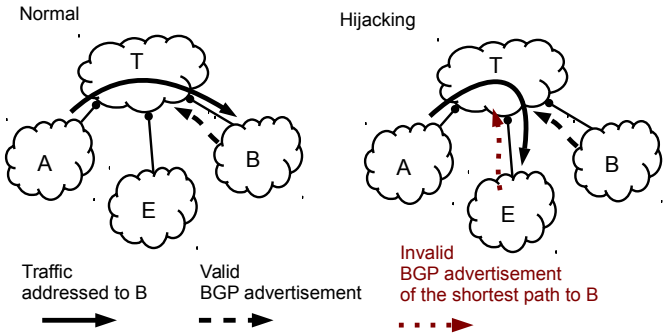

Fig. 2: BGP tampering – *prefix hijacking*. Traffic from AS A to AS B is erroneously diverted to AS E. Figure inspired by Ballani et al. [15, fig. 2].

3.2 DNS tampering

The access to a *target* usually implies the translation from the symbolic name of the server hosting the resource to its IP address. The communications related to this phase can be subject to different censorship techniques, first analyzed in detail in [48]. The sequence diagram of a DNS request is

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

shown in Fig. 3: the software module on the client (referred to as "stub resolver") issues a `type A` query to the *recursive resolver*, that is in charge of performing the resolution asking the authoritative servers (or providing a previously cached response).[5]

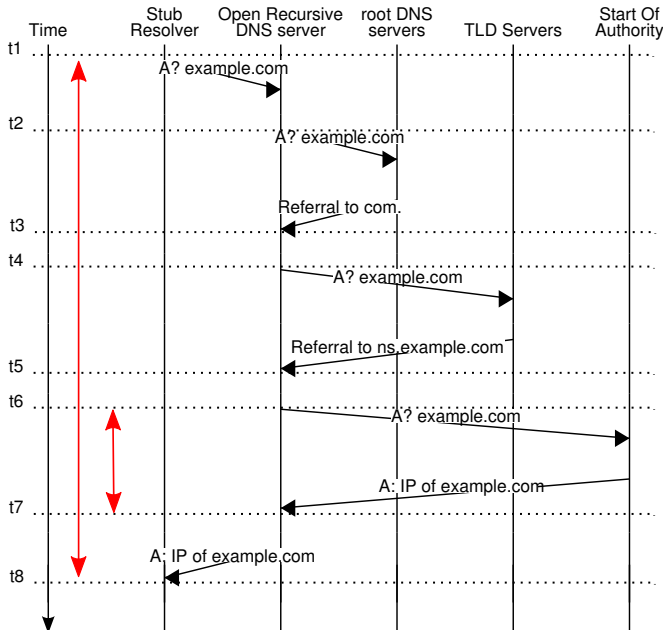

Fig. 3: DNS tampering – Comparison of time window in which a fake response can be provided to an Open Recursive DNS resolver (between instants t6 and t7) and the time window in which the stub resolver is vulnerable to *injection* (between instants t1 and t8). Figure inspired by Dagon et al. [37, fig. 2].

We have adopted the umbrella term "DNS tampering" to avoid confusion with the term "DNS redirection" found in literature (Gill et al. [63]) to specify one of the possible effects (thus a *symptom* in the lexicon defined in this survey) of these censoring techniques, while there are different variants involved with this process, according to (i) the presence of *surveillance devices* on the path between the client (stub resolver) and the recursive resolver and (ii) the kind of response that is provided back. These variants are described in the following.

3.2.1 DNS hijacking: According to the DNS protocol definition (Mockapetris [109]), when a DNS recursive server is queried for a resource record it should fetch it from the authoritative servers (if a cached entry is not found): censoring servers instead reply with a forged response, not corresponding to the legitimate DNS database entry. Having administrative control on the DNS server allows to alter its behavior diverting it from the standard [108]; the following possible responses are given to a query of `type A` in lieu of the expected Resource Record:

`NXDOMAIN` an error response of type "no such domain" – the domain name referenced in the query does not exist;

forged Resource Record a Resource Record of `type A`, `Class IN` with an IP falling in one of the following cases:

- *Block Page* the returned IP address hosts a webserver that invariably serves a page telling that the requested hostname *has been deliberately blocked*.
- *Error Page* the returned IP address hosts a webserver that invariably serves a page telling that *the requested hostname is not existent or misspelled*;
- *Failing IP* a non-Internet-routable addresses such as *private address space* [125] or *shared address space* [154] as well as ordinary assigned IP addresses unrelated to the requested resource;
- *Surveillance IP* the returned IP address is assigned to a surveillance device that inspects higher layer protocols;

The *symptom* that the client experiences is therefore different according to the replies it gets: only in the "Block Page" case the censorship is clearly notified, possibly providing a motivation or a law demanding it, and in some cases a reference to appeal the censorship. If a DNS error is returned, a tech-savvy user can infer that something went wrong with the name resolution phase (which is indeed true to some extent). The same happens for the "Error Page" case: the error is surfaced to the user (potentially fooling applications that rely on protocol-compliant behavior, see [48]) but still gives hints about the phase that failed. When a "Failing IP" is provided, an error of type `network unreachable` or `host unreachable`, or a TCP connection error will be returned to the user, giving no information about the real event. In case the returned IP address is assigned to a host that is reachable by the client, the *symptom* will be different according to whether a service is listening at the transport port (usually 80, for HTTP requests) and how it will react to the HTTP request (likely replying with a `HTTP 404` resource not found error). The case of "Surveillance IP" has been detected for two-stages censoring techniques, where DNS is used to divert selected traffic to a *surveillance device* that will inspect and possibly cause censorship (see Section 3.10.2).

From the variability of the possible outcomes we observe how little transparency a user would experience when dealing with censored resources, and how simple detection tests, such as the one performed through the *Herdict* platform (Sect. 5.2), can not give reliable information about the censorship technique.

Censorship techniques based on DNS have been documented since the early analysis of Internet Censorship [48] and confirmed in most of the field tests worldwide [79] up to the time of writing.

The *trigger* for this censorship technique is a `UDP` port 53 packet directed to the IP address of the misbehaving DNS

[5]The recursive resolver IP address is provided by the network administrator or obtained by automatic host configuration, and usually corresponds to a server of the Internet Service Provider that gives the client host connectivity to the Internet.

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

recursive resolver and containing a DNS query of `type A` including the hostname belonging to the blacklist. In order to enforce censorship with this technique it is not necessary to have a *surveillance device* on the network path between the client and the *target*, as the client will issue a direct request to the recursive resolver (that acts as the *censoring device*). This on the other hand makes the circumvention of this censorship technique straightforward: changing the default recursive resolver will avoid the *censoring device* and thus leave open access to the Internet. Actually the change of the default (ISP-provided) DNS recursive resolver with other "open" resolvers is not rare, but can adversely impact the user experience [6].

3.2.2 DNS injection: Differently from DNS hijacking—performed directly at the recursive resolver—a more sophisticated technique is the *injection* of forged packets that imitate the legitimate response of the queried DNS server but providing fake data. Injection can happen at different locations of the network, not necessarily on the path between the client and the *target* and requires a *surveillance device* on the network path between the stub resolver and the recursive resolver or between the latter and the authoritative server that should provide the requested Resource Record.

Looking at Fig. 3 we can see how a *censoring device* has the opportunity of replying to the stub server faster than the queried resolver: the first well formed message arriving to the client will be accepted, and a possible subsequent legitimate one will be ignored. The *trigger* for this technique is similar to the hijacking performed at the ISP recursive resolver (`UDP` packet with destination port 53, carrying a DNS query of `type A` with the blacklisted hostname) but in this case there is no need to have as the packet destination address the IP of the default DNS recursive resolver: as long as the packet reaches the *surveillance* device the censorship can be applied. This makes ineffective the simple circumvention technique adopted against the DNS hijacking described before, as also the queries addressed to third party resolvers will be intercepted and replied with the tampered data.

The types of responses that are injected are the same as the aforementioned ones, and thus the same *symptoms* are experienced by the client. Due to the different mechanics, however, this technique can have much broader impact than intended, as found in [10], where it is shown how censorship applied by transit ASes affects also DNS queries originating from foreign countries, finally censoring the *target* for peoples that are not subject to the censor's jurisdiction.

3.2.3 DNS-Sec DoS: The original design of DNS did not assume a hostile network environment, hence the weakness of this protocol in the face of tampering; to extend it while retaining compatibility with the existing infrastructure the Secure DNS (DNS-Sec) specification has been proposed (Atkins and Austein [14]).

The adoption of DNS-Sec provides secure authentication of server response, thus preventing the possibility of injecting a forged response that could be accepted as valid. Even if authentication prevents the censor *action* to succeed unnoticed, yet the failure of the DNS resolution causes the impairment of access to the *target* whenever an untampered response can not be received – creating a Denial of Service. We refer to Vixie [149] and Crocker et al. [35] for a discussion on other possibilities for the censor to work-around DNS-Sec, where such options are all discarded concluding that current and envisioned DNS tampering techniques are not compatible with DNS-Sec adoption. Moreover, DNS-Sec makes impossible the redirection to warning pages, preventing "informed blocking": the only *symptom* a client would experience is a DNS error—or possibly a DNS-Sec validation error reporting that the name resolution system has been tampered with. In this case the expert user could be aware of the *Man-In-The-Middle* (MITM) attack she is undergoing, but a user ignoring the network technicalities could as well blame the *target* for the failure, mistaking the censorship for outage or downright nonexistence of the *target*. This would negatively affect the transparency of censorship practices, at least for non tech-savvy users. On the other hand, DNS-Sec forces the censor to operate an on-path censoring device that must drop either the DNS request or the untampered DNS response: a more restrictive setup potentially raising the cost of censorship enforcement. Though current deployment of DNS-Sec may be limited (Lian et al. [103]), it is expected to grow as security concerns mandate it, thus having more impact on effectiveness and transparency of censorship enforced by *DNS tampering*.

3.3 Packet filtering

We group under the term "packet filtering" all the censorship techniques whose *action* is to simply discard packets. With this technique the triggering packet is silently dropped causing a *symptom* of type `connection timed out` error. The *trigger* is data from the headers of *up to* the fourth level of the TCP/IP network stack (thus including also network layer). The motivation for associating *triggers* of different layers to a single censoring technique (while in principle they should be used to tell apart different setups) is that the enforcement of censorship based on the headers of these two protocols have little practical differences: packet filtering of varying complexity is a standard functionality provided by network devices ranging from switches to dedicated security appliances.

This technique requires a *surveillance device* on the path between the client and the *target* (as opposed to BGP tampering and DNS hijacking), and the *censoring device* must be in-line too (as noted, they are possibly the same device). In a stateless censoring system this technique can be used to block IP addresses of targets that are also subject to DNS tampering, so that if the client circumvents censorship in the DNS resolution phase it is caught on the first packet of the TCP handshake. It has been found *in the wild* [48, 112], but on the one side it has the shortcoming of blocking all services reachable at the same IP address (thus "overblocking" virtual

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

hosts that, though having a different hostname, are hosted on the same IP), on the other side it requires the censor to collect all the IP addresses associated with the targeted resource and configure the *surveillance device* with rules for all of them. The same setup and *trigger* can be used to selectively route packets towards an off-path *surveillance device* in multi-stage censoring systems (see Section 3.10).

3.4 TCP connection disruption

An intentional disruption of the communication can happen during either the setup of the TCP connection or during the subsequent packet exchange belonging to the same connection, i.e. sharing the same 5-tuple (source IP, destination IP, protocol=TCP, source port, destination port).

Techniques enforcing this *action* leverage the connection-oriented nature of the TCP protocol: a notion of "state" of the connection is held at each endpoint and signals are used to set-up and tear-down the connection. The *censoring device* sends to the client packets that have the source IP of the *target* and the `RST` flag set, indicating that the connection is in a wrong state and thus has to be immediately terminated. The client will experience a *symptom* of type `connection reset` error. The same can be done towards the *target* sending forged packets seemingly originated by the client and with the `RST` flag set. Also in this case the client will experience a *symptom* of type `connection reset` error, but generated by legitimate `RST` packets coming from the *target*.

The *trigger* for this technique contains the destination IP address or of the *target* (Clayton et al. [31]) and possibly the transport level port numbers, to limit censoring to a specific application such as HTTP (port 80), HTTPS (port 443), SSH (port 22); this requires a *surveillance device* on the path between the client and the *target* (as opposed to BGP tampering and DNS hijacking).

This *action* can be used as in two-stage techniques (see Section 3.10).

3.5 Soft Censorship

Blocking the access to an online resource or service is an evident action, that if prolonged in time stands out as intentional, and possibly draws attention and strong complains. Instead, gradually reducing the Quality of Service (QoS, Kurose [95]) and thus the perceived performances is much less evident, and it is harder to prove intentionality. Inconstancy of performance also adds up to the difficulties in measuring this action, and also on the frustration of the user, that ultimately will look for—allowed—alternatives: such is allegedly the case for the Chinese offer of online services replicating foreign analogous (Mumm [111]). The intended effect thus, i.e. preventing the user from accessing some resource or service, is reached anyway.

In order to enforce this kind of *soft censorship*, also called *throttling* (Anderson [9]), tools initially devised to guarantee QoS—and later used to violate *network neutrality* for economical advantage—are being employed (Aryan et al. [13]).

The *action* corresponds to the worsening of QoS parameters:

- increased packet loss
- increased delays
- increased jitter
- reduced bandwidth

One simple method to achieve these results (as detected in [13]) would be to filter random packets along the path. In the case of communications relying on TCP, this would significantly impact the delays and throughput due to the connection-oriented nature of the protocol, bound to retransmitting lost packets and waiting for in-order reassembly. The UDP protocol instead would not suffer additional damage besides the packet loss, but the communication would still be heavily affected in case the application protocol that is carried in UDP has its own loss recovery mechanisms.

Such *actions* can be implemented in routers and thus classified as a special case of "TCP/IP" filtering, and can adopt both a stateless or a stateful paradigm.

As for the other cases analyzed so far, a preferential location for the *censoring* device would be on national border connection gateways.

The *trigger* can be, like ordinary (blocking) censorship, related to specific *targets*, but a notable difference is that in this case the *trigger* can also be void, i.e. all communications—no matter the *target* or protocol, or content—will be subject to the *action*. This scenario is possibly based on external events (not elicited from the network) that cause a curfew-like control of Internet access [13] reminding of the nation-wide disconnects experienced in the events of the "Arab Spring"[39].

3.6 TLS tampering

One widely adopted defense against various kinds of MITM attacks in accessing content on the Internet is provided by Transport Layer Security / Secure Sockets Layer (TLS/SSL, see Dierks and Rescorla [45]). This protocol operates over the transport layer of the TCP/IP stack and offers an authenticated and private channel to the application protocol (thus behaving as the lowest sub-layer of the application layer). Besides HTTP, it can be adopted to secure other application layer protocols such as FTP, SMTP, XMPP, based on TCP.[6] Basically TLS performs a session setup phase (Fig. 4) using asymmetric cyphers whose keys (in form of SSL "certificates") are verified according to a trusted third party, the "Certification Authority" (CA). Once the negotiation of a symmetric cypher and session key is completed, the remaining communication will convey the encrypted data of the application level protocol.

Such technology has a number of general shortcomings, briefly discussed by Laurie [98] with a proposed solution. From the censorship point of view, the TLS handshake provides elements that can be used to identify some specific

[6]For applications relying on the UDP transport protocol there is a dedicated version, Datagram Transport Layer Security (Rescorla and Modadugu [127]).

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

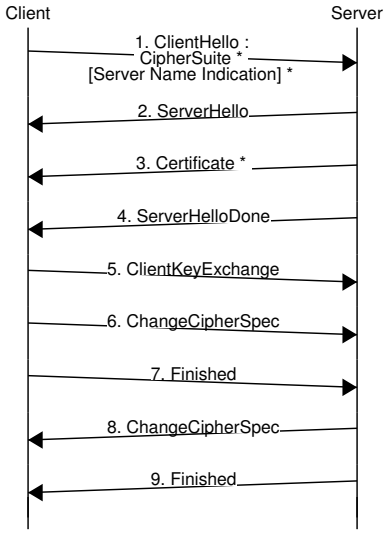

Fig. 4: TLS tampering – TLS handshake for server authentication. Potential triggers for censorship are marked with a *, square brackets denote optional data.

application or service and thus can become *triggers* for censorship; Winter [156] cites the *client cipher lists*, the *server certificates* and the (randomly generated) *Server Name Indication* as examples of known *triggers* for the Tor [46] application.

Besides the interception of the aforementioned *triggers*, other censorship possibilities exist, described hereafter.

3.6.1 TLS compelled certification creation attack: Although a number of weaknesses and countermeasures have been proposed in the past, a recent "politically sensitive" attack scenario has been presented by Soghoian and Stamm [139] that could be easily employed by a censor in order to tamper with the communication (e.g., editing content) in ways unnoticeable to the user. The effectiveness of the attack is based on providing an SSL certificate for the site attesting a false identification, but still results as valid because it is signed by a trusted (but actually *misbehaving*) CA. In the attack scenario introduced in [139] this CA can be compelled by government institutions to provide a site-specific certificate or even an intermediate certificate that can be used to generate valid ones for any website (hence the name of *compelled certification creation attack*).

While injecting an *in*valid certificate would warn the user that some kind of issue is happening (still not declaring that it is intentional censorship), the use of a rogue certificate would completely hide the tampering and still provide the user with both tampered content and a false sense of confidence in its authenticity.

The *trigger* of this technique is an HTTPS request with a hostname for which the compelled CA has generated a rogue (fake but valid) certificate. The *symptom* is the reception of mangled content without the TLS protocol being able to alert for it.

The authors of [139] state that to the best of their knowledge there is no documentation about the use of MITM attacks by *compelled certification creation* aimed at censoring content, but only at accessing encrypted communications (e.g., in the Iranian GMail incident [101]); nonetheless there is no evident theoretical obstacle to adoption for censorship in addition to the known surveillance application.

3.6.2 TLS DoS: Analogous to the DNS-Sec case, if the censor tries to intercept the beginning of the communication that is protected with TLS and fails to provide a valid certificate then the TLS layer will warn the application that the validation of the certificate has failed and thus a MITM attack could have been attempted. The *trigger* of this technique is an HTTPS request with a hostname for which the censor does not provide a valid certificate, and the *symptom* is an error (usually presented with an evident pop-up offering the user to either abort the connection or to override the failed certificate check and continue). If the user refuses to continue will not have access to the *target*. Again, an informed user is able to understand what happened, while an ordinary user could blame the *target* itself or the browser security features. If the user decides to ignore the validation error then the censor has access to the cleartext communication and thus can apply the *surveillance* and selective *censoring* techniques described in the following (again preventing the user from accessing the unmodified *target*).

In surveying the related literature the authors have found no publications addressing the detection of this type of multi-step censoring technique.

3.7 Keyword blocking

A *surveillance device* that analyzes packets beyond the headers of IP and TCP is said to be performing Deep Packet Inspection. Different depths of inspection are possible depending on the payload that is analyzed (first packet, a given number of initial packets, all the packets); moreover different degrees of complexity in the analysis of such payload can be adopted (per-packet string matching, per-stream string matching, syntactical verification, semantic verification), progressing from a stateless to stateful inspection with increasing status information kept per connection. The more payload is analyzed and the more complex the analysis, the higher the resources required for the *surveillance device*.[7]

The *action* performed by the censoring system can be of the same kind of the ones adopted in TCP-level filtering, but having an established TCP connection between the client and the *target* there is also the chance to provide application data, e.g., to redirect to a blocking page, or performing varying degrees of content mangling.

[7]See Risso et al. [128] for a comparison of accuracy and resource consumption of different traffic classification methods.

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

An example of enforcement of such kind of *action* for applications using HTTP is *HTTP tampering*, described in the following section.

3.8 HTTP tampering

If a TCP connection is allowed to complete successfully, the censor has the opportunity of providing the client with an HTTP message that will be parsed by the client as it were coming from the queried *target*.

In the HTTP protocol (Fielding et al. [56]) the messages are divided in two parts: the *header* and (not for all types of message) a *body*; a fundamental part of the header for response messages is the *status code*, a numeric value of three digits characterizing the message type.[8] According to the different *triggers* that the *surveillance* device looks for in the request message, the different response messages that the *censoring* device sends back to the client, and the location of these components of the censoring system, the variants described in the following are possible.

The *action* of the *censoring* device in the case of HTTP tampering consists in sending an HTTP response message belonging to the following types, with some status codes grouped by the first two digits:

- codes `30X` redirect, signaling that the requested resource is at another URL, reported in the "location" header field; this will cause the web browser to begin a new communication sequence (possibly starting from DNS resolution of the symbolic hostname of the new URL) to reach the addressed resource; this performs an "HTTP redirect" and the experienced *symptom* for the user will depend on the result of the new communication;
- code `404` resource not found: the path of the requested URL does not correspond to a resource known to the server; the *symptom* experienced by the user will be a browser-provided error message describing the issue;
- code `403` forbidden: the request is valid but the resource will not be provided; the body of the message can contain a description of the reason; the *symptom* experienced by the user will be the content of the page if present, or a browser-provided error message describing the error
- code `200` no error, signaling that the requested resource is provided in the body of the message; the browser will parse the content.

3.9 HTTP proxy filtering

A special case of semantic stateful inspection is constituted by a transparent proxy located in-line with respect to the path between the client and the *target*, by all means performing a MITM attack, forwarding only content that does not match the blacklisting rules. The proxy fully understands the application level protocol, hence the keyword matching can be done on specific statuses of the application protocol automaton, and on specific data fields, thus being highly selective and reducing possibilities of overblocking. The downside is that the transparent proxy must be in-line and that it is required significant processing power, otherwise performance impairing or altogether blocking could occur as a byproduct also for traffic that should not be censored.

An alternative is to have multi-stage censorship system, with the preceding stages in charge of pre-filtering (and blocking) connections or "hijacking" towards the *censoring device* the suspicious connections only (see Section 3.10).

3.10 Multi-stage blocking

Having a *surveillance / censoring* device working in-line requires that it has to be deployed at country borders in order to intercept traffic directed to foreign countries (hosting *targets* that are legally out of direct reach for the censor). This setup poses two technical problems: (i) it has to process all the cross-border exchange traffic, potentially suffering performance issues; (ii) it represents a Single Point of Failure: any issue in this equipment would disconnect the served network from the Internet, with potentially high economic loss.

This problem has been solved in known censorship systems by employing multi-stage systems, e.g., preselecting at the first stage a much smaller fraction of "suspicious" traffic, discharging the second-stage device from processing large part of permitted traffic. This way the finer-grain blocking of the second-stage censoring device does not come at a high cost.

Another solution uses a deployment where the first stage is akin to a mirroring port just copying the traffic to an out-of-band *inspection* device, that therefore does not impair the transmission performance of the original flows. Such a setup has a drawback, though: the enforcement of injection techniques is more challenging for the censor if the *censoring device* is not in-line with the path between the client and the *target* and thus can not discard the request. In this setup the *censoring device* is engaged in a race condition against the legitimate response of the *target* server and its forged packets could arrive after the *target* reply; this would make the client ignore the censor packets due to the mismatch of the TCP sequence number.[9]

Borrowing from literature on traffic classification (Dainotti et al. [40]) and censorship circumvention, we can envision another case in which a multi-stage setup can be useful or necessary: the adoption of *behavioral* features as *trigger*. We group under this umbrella term: statistical analysis of network and transport level characteristics, e.g., packet size, inter-packet time; connection graph properties, e.g., number of IPs interested by outbound/inbound connections from/to a given host; reaction from active probing performed by the censor (see Winter and Lindskog [157]). In order to collect this kind of features a *surveillance device* must observe several

[8] The "response messages" are sent by the server when replying to "request messages", from the client.

[9] The initial sequence number is randomly generated to make harder for an off-path attacker to guess it, in compliance with Gont and Bellovin [64].

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

related packets, thus in general the *trigger* can be matched only after some traffic already has gone—back and forth—through it, hence the suitability of an off-path setup. To the best of our knowledge, no *detection* system supports this kind of *triggers*, besides cited [157] where an actual deployment of the censored application is adopted.

In the following some examples of multi-stage deployments are described.

3.10.1 BGP hijacking + HTTP proxy: One of the first descriptions of a multi-stage deployment is provided by Clayton [30]: here the aforementioned reasons in favor of this kind of deployment are stated as motivation for the design of the system. The first-stage is triggered by destination IP address and TCP port number. In case of match, the packet is routed (by means of BGP) to an HTTP proxy that can access the hostname and the specific resource (second-stage *trigger*). Allowed URLs are fetched from the original *target* and served back to the client, while blacklisted ones will trigger an *action* of the kind reported in Section 3.9 – in [30] requests are ignored, thus the client waits in vain until the *timeout* is struck.

3.10.2 DNS hijacking + HTTP proxy: This technique is triggered by DNS queries (UDP port 53) of *type A*, i.e. requiring the translation of a hostname belonging to a blacklist; the censoring DNS server replies providing an IP address that belongs to the second-stage surveillance device. Then the browser establishes a TCP connection directly with the second-stage surveillance device.[10] To an HTTP *GET* request including an URL belonging to a blacklist (secondary *trigger*) the second-stage censoring device replies with one of the *actions* seen in Section 3.9. If the requested URL was not blacklisted then the request is let pass.

3.10.3 Keyword blocking + TCP disruption: The *trigger* is a TCP/port 80 (HTTP) packet containing an `HTTP GET` request for the *target* URL and as IP destination address the *target*'s address.[11] If the URL contains a blacklisted keyword then the TCP connection disruption (see 3.4) is enacted.

This deployment has been found operating in China and has been analyzed in detail, showing evolution in complexity over time (Clayton et al. [31], Weaver et al. [151], Xu et al. [159], Polverini and Pottenger [119], Verkamp and Gupta [147], Feng and Guo [54]), with different levels of sophistication in the craft of `RST` packets.[12] The analysis revealed stateful implementations for both stages: only the HTTP request belonging to a correctly established TCP connection triggers the censorship, while after it has been activated all packets sharing the 5-tuple (IP source and destination addresses, TCP transport protocol, source and destination ports) generates the TCP connection disruption *action* (Xu et al. [159]).

[10] If HTTPS is employed, at this point the browser attempts a TLS session establishment, that will fail unless a TLS compelled certificate creation attack has been performed (see Section 3.6.1). If the user, uncaring or unaware of the risk, allows the browser to accept the invalid certificate anyway, then the process continues as if HTTP were used.

[11] The payload of the TCP packet contains the strings of the query part of the URL (after the `GET` string) and the hostname part of the URL (following the `Host:` string).

[12] See Weaver et al. [151] for an experimental survey of the types of packets forged by *censoring devices*.

4 Censorship detection techniques and architectures

In coherence with the definition of Internet Censorship we have adopted in Section 2.1, we consider the Internet Censorship Detection (hereafter also "detection", when no ambiguity arises) as "*the process that, analyzing network data, proves the existence of impairments in the access to content and services caused by a third party (neither the client system nor the server hosting the resource or service) and not justifiable as an outage*". In the process of *detection* we implicitly include the collection of the suitable network traffic data. The device used to collect said network traffic or related information (metadata) is hereafter called *probe*.

We also note that in the considered literature the focus is predominantly on *Network-based Censorship*, and thus *detection* in general refers, unless differently stated, to the related techniques (Section 2.4).

Detection is essentially based on the ability to tell the effect of the censorship from the "normal" uncensored result and from involuntary outages; for a class of detection methods (active detection), it requires also the possibility of intentionally triggering the supposed censorship system. The inference of the adopted censorship technique is inherent to identifying the type of third party causing the impairment and its differentiation from an outage.

With reference and in addition to definitions stated in Section 2.1, censorship detection techniques can be characterized considering two main aspects:

viewpoint the role of the *probe* host in a client-server communication model:

- **client-based** collected network traffic is initiated by the same host of the *probe*, i.e., the *probe* sees traffic that originates from IP addresses belonging to the same host;
- **gateway-based** collected traffic has neither source nor destination IP addresses belonging to the probe; in the scenario of interest the *probe* host is a gateway towards Internet: all traffic between the served network and the Internet passes through it;
- **server-based** collected network traffic is sent to IP addresses belonging to the *probe* host, initiated by several other network hosts;

collection method the collection of network traffic data can be performed with *active* or *passive* techniques

- **active collection** techniques that use client systems (*probes*) to generate network traffic purposely crafted for possibly eliciting a censorship response (to be recorded and analyzed);
- **passive collection** techniques that extract traffic data from network application logs or traffic traces captured

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

on a device (*probe*) in order to look for evidence of censorship events.

A special case crossing these definitions is the usage of *active* methods towards a controlled destination: as both sides of the communication are controlled (both qualify as "probe") then traffic can be collected also at the receiver side (*passive* collection method). As for the *viewpoint*, if both edges are controlled then once the server receives client traffic it can also respond with a purposely crafted reply (e.g., containing a *second-stage trigger* for a stateful censorship system); finally, probes can switch role, allowing directed testing of the network paths in-between. This setup is akin to the one used in the methods of *network tomography* [20], thus in analogy we name it "censorship tomography", or shortly "tomography" as a special case of *active* methods. In fact even though *censorship tomography* implies logging of received traffic, the possibility to generate traffic purposely forged to trigger some specific mechanisms is the essential property that characterizes *active* methods. In the following we describe *active* detection methods, then *passive ones*, ending with *architectures* adopting them.

4.1 Active detection methods

For *active* censorship detection tools, the algorithm consists in variations on the following steps:

- generate traffic towards a *target*, supposedly reachable;
- if the request returns no results (before a timeout) or an error, then *target* is censored, terminate;
- if the received content is equal to a Ground Truth or satisfies a similarity criterion, then the *target* is considered reachable and not subject to censorship; otherwise *target* is censored.

The Ground Truth can be either obtained by means of trusted communications, or inferred by leveraging multiple viewpoints.

In order to analyze the censorship technique, variations on the request can be made to pinpoint the *trigger*: by comparing the *symptoms* collected for the different tries, information about both the *trigger* and the *action* is obtained and then existence and properties of the *censoring system* can be inferred. We stress that, without the ability to selectively investigate the censoring technique mechanics, a detection tool can hardly tell censorship from outage, as it is the coupling (*trigger*,*symptom*) confronted with the case (*non-trigger*,*expected uncensored behavior*) that can surface the existence and nature of a censoring infrastructure and thus the intentionality of the communication impairment.

In the following an analysis of literature on active detection techniques is presented, grouped by the layer of the network stack that is affected by the detected censorship technique.

4.1.1 DNS resolution: A widespread censoring technique involves the DNS resolution process, that is called into action every time a symbolic name for the host is used (see Section 3.2). Possible variants involve ISPs altering their copy of the distributed DNS database so that a wrong reply is given to clients ("DNS hijacking"); or an in-line *censoring device* intercepting the DNS queries and replying in place of the intended DNS server ("DNS injection").

Detection of DNS tampering without a comparison with a ground truth can be performed by issuing a DNS query for resolving a nonexistent hostname (thus expecting a `NXDOMAIN` error): if a reply containing an address is returned instead of the error then DNS tampering is inferred.

Variants on this method are adopted by Dornseif [48], Gill et al. [63], Nabi [114] and in non-censorship-specific analyses: [6, 38, 152, 153]. The most notable ones are described hereafter.

From the detection point of view, the cases of the complacent ISP and that of the intercepting device can be told apart by querying alternative Name Servers hosted outside of the censored network (as done by Nabi [114]): if the forged Resource Record is only in the database of the ISP's default Name Servers, the alternative Name Server will provide different (correct) answers; in case an intercepting device is actively tampering the DNS traffic then all responses will be equivalent (and wrong), and detection will be possible only by comparing results with the ones collected from probes outside the censored network.

Due to the mechanics of the DNS system [109, 108], there is the possibility to detect *DNS hijacking* even without the aid of a probe inside the censored network: if the censoring DNS is configured as an *open resolver*, thus replying to queries regardless of their source address, it can be tested from any network host. Although not focused on censorship detection, the detection of DNS tampering by surveying and directly querying *open resolvers* has been performed in [38]. In this case another form of the "needle in a haystack" problem is present: how to find *open resolvers* in the whole Internet in order to directly probe them. The solution adopted in [38] leverages active probing and a kind of *tomography* setup, and is described in Section 6.

In case of DNS tampering a comparison with results of measurements from other countries will show that a given (censored) host name is resolved to two different sets of IP addresses. Unfortunately, this is also the case for DNS-based load balancing and more in general performance enhancement usually provided by Content Delivery Networks, or CDNs (see Pathan and Buyya [117] for a taxonomy on this technology). Therefore a detection test that compares the set of resolved addresses of a hostname among different countries would systematically suffer of false positive errors specially for high-traffic addresses (mostly likely to leverage CDNs). This test can instead be used as an exclusion criterion: if the two sets are identical, then **no DNS-based censorship** has been applied for the considered *target*.

In [63] a method is proposed to infer DNS tampering (there named "DNS redirection") by considering all the hostnames resolved to one same IP address in the test results, and count the number of different ASes the same hostnames are resolved

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

to by a trusted DNS server: if the AS count is greater than an empirically set threshold then the response of DNS tampering is given.[13]

A method to detect DNS injection is implicitly suggested in [119] where the authors notice that for DNS replies injected by the GFC the (UDP) checksum is wrong. This aspect is not analyzed in depth to the extent of a viable detection method, as no information about precision nor accuracy of detection is provided.

A simpler active detection method that is able to detect DNS injection is adopted in [10], where the IPv4 address space is scanned, in blocks of /24 networks, looking for one IP in each block **not** running a DNS server: such IP is used as a destination of a DNS query that should have nobody listening to, and thus never receive a reply *unless* an injector device intercepts the requests and replies with a forged response.

4.1.2 IP reachability: The testing of reachability of a host at the network level has been performed since the early days of the deployment of Internet using the standard utilities `ping` and `traceroute` (Jacobson [81]). The first one is a user-level interface to send ICMP `echo requests` messages to an host, receiving back an `echo reply` demonstrating the mutual reachability of the two, i.e. that directed paths forth and back exist between the sender and the receiver.

The original traceroute used UDP packets with an increasing TTL counter, so that at each expiration an ICMP `time-exceeded` error from the last reached router would be returned, altogether with the router IP address; basically collecting the sequence of the router addresses the path between the sender and the destination would be discovered hop-by-hop. More recent versions of `traceroute` allow probing packets of type *ICMP echo request* or *TCP SYN*.

These techniques can carry *triggers* of kind *source/destination IP address* and *source/destination transport port*, thus they are useful to pinpoint censorship techniques of type *packet filtering* (Section 3.3).

There are several possible causes of inaccuracy or unresponsiveness (Luckie et al. [104], Marchetta and Pescapè [105]) that would lead to false positives when using this techniques as censorship tests, thus using them alone has limited usefulness.

Both `ping` and `traceroute` have been used by Feng and Guo [54] as a preliminary active detection technique, inferring censorship when the path trace from inside the censored network towards a server abroad timed out for hops beyond an international gateway router.[14] More complex variations on the `traceroute` technique are described in Section 4.1.6.

4.1.3 TCP reachability: A basic test to detect if censoring techniques of type *packet filtering* or *TCP connection disruption* (Sections 3.3 and 3.4 respectively) are employed by the censoring system can be performed by tools logging transport and network level errors when sending TCP packets with the suitable *trigger*. A tool to start probing with the most basic *trigger* towards a *target* has just to try a three-way handshake: `netcat` [115], a mature and widely used tool for network diagnostic and security testing, has been adopted for this purpose (Clayton et al. [31]).

Often the detection techniques adopted in research papers or in the provided platforms and tools just leverage the application level log files and error reporting functionality to detect the disruption of TCP connection (Gill et al. [63], Aryan et al. [13], Nabi [114], Polverini and Pottenger [119]) e.g., triggering the system directly by sending a—possibly innocuous—`HTTP GET` request (see following section). At the other end of the spectrum of possibilities there is the highly specific technique adopted in [86], where different combinations of initial packets (with different flags set) are generated in order to test the statefulness of a *censoring device* and investigate the layers that it inspects to find the *triggers*. By applying techniques and methodologies used for Network Intrusion Detection Systems (NIDS) analysis and circumvention [151, 123, 69, 137], Khattak et al. [86] confirm and deepen previous characterization of the GFC, also finding possible circumvention techniques and estimating the cost for the censors for fixing them.

The analysis of the censoring system response also varies: the most basic test just detects network errors in response to the *trigger*. An unsupervised machine learning approach has been presented by Polverini and Pottenger [119] to clusterize network traffic, divided in evenly spaced time slices. The traces are collected actively eliciting censorship by probing *targets* known to be censored (as reported by Crandall et al. [33] and [1]), then the clustering algorithm is applied, and afterwards the time slices known to contain censorship evidences (as elicited) are labeled as "anomalous". The resulting classification algorithm is used to infer traffic patterns typical of censorship, and to pre-select time slices for manual inspection. Considered features for the clustering algorithm are: IP and TCP header field values and counting of TCP, UDP, ICMP, IGMP, "miscellaneous" packets. No results are given in terms of overall accuracy and precision of the classification approach, that is only proposed as a preprocessing stage before manual inspection.

A peculiar detection technique is presented by Ensafi et al. [51], proposing a variant of the stealth host scanning technique initially proposed by Antirez [11] to reveal open TCP ports while avoiding identification from the IDS. The proposed technique verifies TCP connectivity between a *client* host and a *server* host by probing both from a third *probe* host (in the text dubbed *measurement machine* or *MM*). The kind of censorship techniques that this detection method can identify are both client-to-server filtering and server-to-client filtering, compared in Fig. 5 with uncensored expected behavior. In the figure dashed lines represent packets that have as source IP address a *spoofed* address, specifically they falsely present

[13]The threshold is set to 32 ASes considering the rate of growth of the percentage of blocking detected with other methods (Gill et al. [63, appendix A.1]).

[14]The adopted traceroute version, shipped with the Windows operating system, generates by default `ICMP echo request` packets.

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

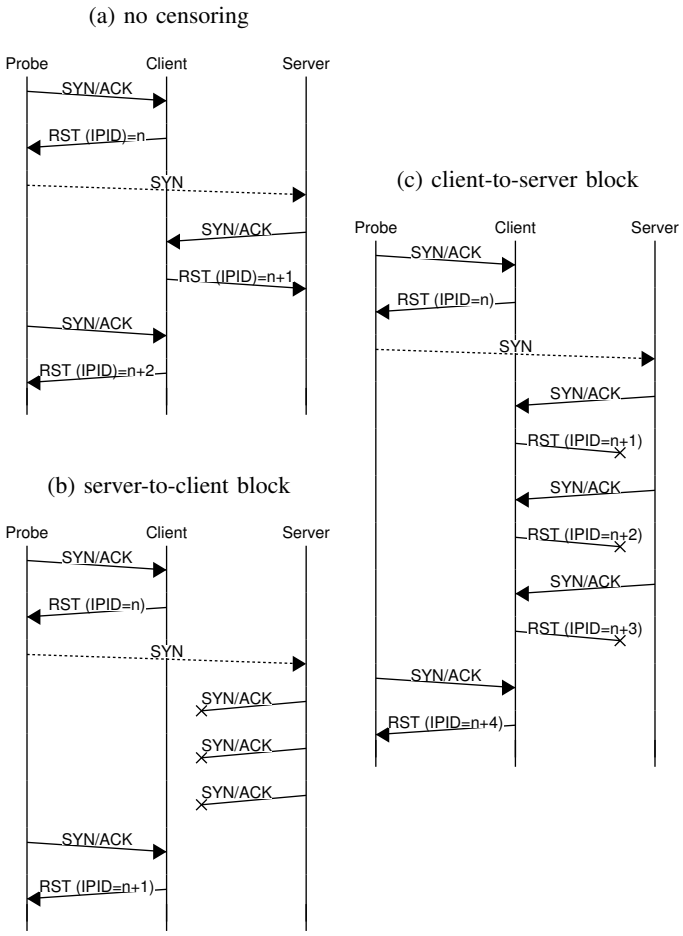

Fig. 5: Detection of TCP reachability – Hybrid IP-ID and SYN scan presented in [51]; dashed lines represent spoofed packets.

the *client* address instead of the *probe* address; they are meant to elicit a response from the *server* to the *client*. The consequences of such response will be learned by the probe indirectly, when engaging the client at the end of the exchange. Exploiting the increment of the value in the ID field of the IPv4 header ("IP-ID" hereafter) of the response packets, the difference between the IP-ID value in the last exchange and the one in the first exchange allow to distinguish among the three possible cases: no censorship is enacted (Fig. 5a), packets from the client to the server are blocked (Fig. 5c), and packets from the server to the client are blocked (Fig. 5b). Limitations in this detection technique lie in the necessity to find suitable *client* machines that provide a predictable IP-ID sequence and in the type of trigger it can use (just IP and TCP port addresses). The first condition according to Ensafi et al. [51] reduces to around 1% of reachable IP address space the potential hosts to be used as *clients* for this technique.[15]

4.1.4 HTTP tampering: The detection of censorship techniques of type *HTTP tampering* (Section 3.8) is the most frequent in literature, being considered virtually by all the censorship-related papers cited so far.

The common procedure is to use an application to request a web resource (the potential *target*) employing the HTTP protocol, the *trigger* being set in the `header` section of the HTTP request, either in the `query` part of the GET request—that is interpreted as a path to a resource on the server—or in the `Host:` header field—that is interpreted as the hostname and used to identify one specific website if many are hosted at the same IP address ("virtual hosting").

The main source of variation is the specific tool adopted to generate an `HTTP GET` request, comprising Python scripting (Gill et al. [63], Filastò and Appelbaum [57], Nabi [114]), the command line common unix utility `wget` (used by Polverini and Pottenger [119]), or more exotic tools such as `fragroute` [58, 123] (used by Park and Crandall [116]) and `scapy` [134] (adopted by Crandall et al. [33], Khattak et al. [86]).

In case a *censorship tomography* setup is adopted, an helper server is used to receive the requests (usually with no *trigger*) and reply back with a blacklisted keyword in the content of the response HTTP message. Such a setup has been proposed by Filastò and Appelbaum [57] using programmable back-ends, and previously adopted by Park and Crandall [116] to detect HTTP response filtering in the Great Firewall of China (found to be dismissed). Recently Khattak et al. [86] have adopted this setup to investigate the details of statefulness (and more in general the vulnerability to circumvention) of the GFC.

In case the censoring system successfully submits a—potentially mangled—content, the ultimate detection technique to check whether a *target* has been tampered with would be to compare the received content against a Ground Truth. If the content is an HTML page of a dynamic web site a verbatim comparison would almost surely fail, as there are variable components such as timestamps, localization effects, and in general every dynamic content managed through server-side scripting would vary the HTML code downloaded by different clients. A method to overcome this variability has been proposed by Sfakianakis et al. [136], that uses three tests on content: (i) an MD5 hash (Rivest [129]) is taken for both the retrieved and the Ground Truth version: if they coincide there is no censorship; (ii) otherwise the relevant content is extracted for both webpages by adopting the *Readability* [96] algorithm and (iii) the result is compared with a fuzzy hashing technique (Kornblum [93]) to obtain a similarity score to base the decision on.

4.1.5 TLS tampering: Considering censorship-specific literature, and tools, we have found explicit mention of tests aimed at detecting TLS/SSL tampering only by Filastò and Appelbaum [57], which in turn cites Holz et al. [74] where the *crossbear* platform is introduced. Such platform is released as open-source [141] and is composed of probes (implemented either as add-on for the Firefox Browser or as standalone executables) and a centralized server. The aim of the platform is to (i) detect tampering of the SSL/TLS chain of certificates

[15] An empirical study on IP-ID linear increase in IPv4 address space is presented by Keys et al. [85].

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

and (ii) locate the device that is performing the MITM attack. While the collection of *targets* is performed using the Firefox add-on, and is thus crowdsourced, periodic monitoring of high access websites is performed by unmanned probes deployed on the PlanetLab [28] infrastructure. Detection and localization are performed in two separate steps. The first step is performed by retrieving the TLS/SSL certificate offered by a *target* and uploading it to the centralized server, that stores the certificate and its metadata with a timestamp; the server compares such (meta-) data with the archived version regarding the same *target*: if a difference is found, the server requires a `traceroute` towards the *target* from the client and from other probes. In order to limit false positives, a score is calculated based on the historical log of certificates for the given *target*, weighting the longest period of observation for the same and previous certificate. Previous work on the detection of TLS tampering, similarly to the detection phase of *Crossbear*, relied on download of the certificate and comparison of it and its metadata with either previously stored version (Soghoian and Stamm [139]) or retrieved from multiple probes in different networks (Holz et al. [73]).

A new possibility for detection of TLS tampering will be available as a consequence of the adoption of the *Certificate Transparency* framework described by Laurie [98].[16] By design such framework provides multiple *monitor* servers that periodically validate the certificates and automatically spot the suspect, illegitimate or unauthorized ones, effectively detecting potential tampering with TLS authentication. The validation of a specific certificate can also be requested through the *auditor* components of the framework. For further details we refer to Laurie [98] and the Certificate Transparency project page [65].

4.1.6 Detection of Censoring Devices: Some detection techniques focus on the topology of the network when probing for censorship, aiming at finding the location of censoring devices. By generating `traceroute`-like traffic that carries the *trigger*, such detection methods are able to count the distance in hops from the *probe* and the censoring device. In fact variations of the ICMP-based `traceroute` technique leverage other kind of probing packets (TCP or UDP) using either varying port numbers (UDP) or different sequence numbers (TCP) to identify the hop that elicited the ICMP `time-exceeded` error. Thus a TCP packet initiating a connection (or carrying payload in an established connection) can be sent with increasing TTL to discover if and where (in terms of path hops) the blocking is enforced.

An example of these techniques can be found in [159], where the location of censorship-enforcing boxes is found by exploiting the behavior of the specific censoring system—namely, the Great Firewall of China (GFC). The peculiar behavior in discussion is related to the *statefulness* of the censoring system, as detected by Clayton et al. [31] and recalled briefly in the following.

[16] The *Google Chrome* browser is planned to require Certificate Transparency adoption for a subset of certificates issued after January 1st, 2015 (http://www.certificate-transparency.org/ev-ct-plan).

In the GFC not all the packets are inspected by the system, but only the ones occurring in a correctly established TCP connection to a webserver (TCP port 80); in such connections Deep Packet Inspection techniques are used to match character strings (keywords belonging to a blacklist): once a blacklisted string is found, forged RSTs directed to both endpoints shut down the connection, and every further connection between the same endpoints is replied with forged RSTs regardless of the packets content, for a fixed timespan. Thus the censoring system "remembers" the connection and behaves differently according to the kind of communication that happened before, in other words the *state* of the connection is kept, hence the *statefulness* of the system.

The statefulness of the Chinese censoring system has changed over time: Xu et al. [159], Polverini and Pottenger [119], Crandall et al. [33], Khattak et al. [86] found it being stateful with no exceptions, as a TCP packet containing an `HTTP GET` request with a blacklisted string would be a *trigger* only if sent after a valid TCP connection establishment.

The probing technique in discussion is aimed at detecting and topologically locating devices that enforce keyword-based censorship. It uses a *first-stage trigger* of type `TCP-port-80`, `HTTP-header-keyword` to activate filtering towards a website inside the censored network; after the activation a `traceroute`-like sequence of packets with *trigger* simply *IP-destination,TCP-port-80* is sent until the reception of an `RST` signals the finding of the censoring device. More detail is provided in the algorithm pseudocode in Fig. 6.

```
 1 for target in targetlist do
 2     check HTTP reachability with neutral GET
 3     if target is reachable then
 4         try HTTP GET with ''FALUN'' keyword
 5         wait 5 seconds
 6         if no RST is received then
 7             target is whitelisted
 8             next for
 9         else repeat:
10             hop=hop+1
11             send ACK to target with TTL=hop
12             if response is RST then
13                 censorIP=(last saved IP)
14                 censordistance=hop
15                 exit repeat
16             elseif response is ICMP Time Exceeded
then continue repeat
17           end repeat
18         end if
19     end if
20 end for
```

Fig. 6: Topology-aware detection: pseudocode describing the location algorithm for Chinese Great Firewall presented by Xu et al. [159] (interpreted from the textual description).

The characterization of the GFC in terms of its NIDS functionalities performed by Khattak et al. [86] is detailed to the point that could provide a fingerprint of a specific device or appliance.

Another traceroute-like detection and location technique is presented in [10], where a preliminary measurement campaign

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*, http://dx.doi.org/10.1016/j.comnet.2015.03.008

collects network paths subject to DNS injection (see Section 4.1.1), then for each path a series of DNS queries with increasing TTL and a known censored domain name as *trigger* is sent, thus identifying the hop (and potentially the IP) of the node hosting the surveillance device.

A classic ICMP-based traceroute is used by Holz et al. [74] in order to localize the device performing a MITM by TLS tampering. In this case multiple traces are collected from different probes towards the same—supposedly impersonated—*target*; each probe being characterized as affected or not by the attack. By comparing paths of probes affected and not, the IP corresponding to the *censoring device* can be revealed.[17]

Other detection techniques are aimed specifically at identifying the type of censoring device, leveraging characteristic text strings in the headers or the body of the responses. Such is part of the detection methodology adopted in [41], in which the HTTP headers and the directory structure of the device administration page (misconfigured as to be reachable from the public Internet) are exploited to identify the device as marketed by a given vendor. Through geolocation of the related IP address, the country hosting the proxy filter is identified. The methodology includes also the use of "reporting interfaces" provided by the filtering product firm to submit and categorize new URL to be blocked, along the following steps: an ad-hoc set of servers providing typically blocked services (namely proxying and adult content) are created ex-novo; from inside the censornet said set of servers is accessed and verified unblocked; half of the set is reported to the vendor interface; after a few days (less than 5), from inside the censornet the set of servers is accessed and verified blocked only for reported servers. While this methodology has been effective in giving insights on the devices adopted by censors, the non-triviality of its automation and its dependence on occasional misconfigurations and auxiliary services makes it not very robust. A similar methodology based on the characterization of the redirection methods has been previously adopted for the analysis of DNS tampering in a non censorship-specific scenario (`NXDOMAIN` hijacking for advertising and user profiling) performed by [153], where the authors were able to identify products of *monetization* companies that applied the tampering.

4.1.7 Soft-censorship detection: Detection of soft censorship activities requires the possibility to measure impairments in access to online services and resources. This kind of detection is rooted in techniques from the field of performance measurements and Quality of Service estimation and monitoring. An example of detection of *throttling* is found in [9], in which the author applies statistical analysis on measurement data collected by μTorrent clients whose IP addresses are geolocated in Iran. The performed measurements are executions of Network Diagnostic Tool (NDT, see Carlson [19]) tests against M-lab servers [49] outside the country under analysis (mostly Greece, U.S.A., and U.K.). Considered performance parameters are Round-Trip Time (minimum and average), Packet Loss, Network-limited Time Ratioand Network Throughput. Measurements are aggregated along three axes: country-level, ASN and network prefixes, control group (defined as "logical, coherent groups of networks and clients based on common characteristics, such as the nature of the end user or performance", assuming that sets of privileged users—government agencies, banks, commercial customers—are subjected to different policies than the rest). The detection of significant events (suggestive of censorship activities) is based on thresholds (trend-based minimum and maximum bounds) and variance among different aggregates (assuming that natural variance is high, while when external limitations are imposed variance will be low). A platform that detects soft-censorship is *Greatfire.org* (described in Section 5.5): if tested *targets* allow an average download rate below a given threshold they are considered as throttled and labeled "otherwise restricted".

4.1.8 Detection of server-based censorship: The detection of server-based *censorship* is an edge case of active detection methods. Most of the definitions proposed in Section 2.1 still apply, but collapse on each other, as the *target* at the same time is also the *surveillance and censoring device*. Another peculiarity of this case is that censorship is not elicited by *triggers* in the probe request, but are to be inferred in the resource data and metadata: e.g., for a blog post the presence of a sensitive keyword; moreover the censoring *action* (making the *target* unaccessible) does not happen in the phase of the *target* probing, and neither necessarily at the moment of submission of content to the online service: instead it can happen at any subsequent moment, caused by periodic inspection by censors or solicited by external events (third party requests for removal), all of them in general not under the control of the detection system.

Despite these differences, as in general the services based on user-provided content present a web interface, the phase of collection of evidences is analogous to HTTP tampering tests: the *probe* issues an HTTP GET request for the web resource potentially censored (identified, as *target*, by the related URL) and the outcome of the request is stored and compared against an earlier stored copy, looking for disappearance of messages. Such is the overall methodology applied by Zhu et al. [160], Fu et al. [61], King et al. [87], that differ mostly in how the users to be monitored are found (a variation of the "needle in a haystack" problem discussed in Section 6.5) and how the *trigger* (intended here as the element that caused censorship) is inferred. Most notably the analysis phase aimed at telling censorship from outages or other "natural" causes of message unavailability—such as deletion by the author—does not pose significant troubles: in the analyzed cases (regarding Chinese providers of user-generated-content) censorship was explicitly

[17] In [74] the localization algorithm efficacy is evaluated using a model of the Internet topology; a closed-form characterization is provided of the number of probes necessary to localize the *censoring device* with the precision of 1, 2 or 3 hops with a given probability. The model shows that as little as 100 probes allow for localization with *AS* precision—enough for nation-level censorship—while about $5K$ probes are necessary to locate with 50% probability the device with a precision of less than 4 hops, never reaching probability higher than 70% regardless of the number of probes recruited.

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

signaled with standard notices. This is the case of the platform *Greatfire.org* (Section 5.5) where the Chinese web search engine *Baidu* is tested by submitting queries with supposedly censored *keywords* and looking in the response for a known text that explicitly notifies censorship. Same criterion is adopted for the platform *Weiboscope* (Section 5.7), monitoring the Chinese microblogging platform *Sina Weibo*. It is evident that in case the practice of notifying the user with a warning message were discontinued, such simple tests would fail and other means (and possibly more complex analysis) would be needed. As a consequence of the current ease in telling server-side censorship from other unavailability causes, the literature on censorship detection monitoring server-based censorship is focused either on the selection of *targets* or on the inference of the censors' criteria for deleting content. These goals are performed applying text analysis and knowledge discovery techniques and tools such as Landauer et al. [97], Grimmer and King [68].

4.2 Passive detection methods

As introduced in the previous section, *passive* methods are characterized by their inability to intentionally elicit censorship, as such methods do not actively inject traffic in the network. Traffic data and metadata that are processed looking for censorship evidences have been independently generated by users or processes not under control of the detection mechanism. One specific limitation of these methods is the necessity that the users engage the censoring system: variations of the degree of self-censorship will impact on the censoring events that can be detected by these methods, resulting in a factor that is hard to account for.

According to the *viewpoint* they are applied to, passive detection methods can be divided in the following groups.

4.2.1 Server-based Detection: Server-based Internet Censorship Detection methods belong to the *passive detection* category, as they get the evidences of censorship from traffic traces and application logs. Such data, collected at a server, is independently generated by the customers when accessing the provided online services, thus it is limited in two aspects: (i) the considered *targets* are only those hosted on the server itself; (ii) the *triggers* are limited to the service protocol. Depending on the service architecture it can adopt a single viewpoint (service hosted on one single host) or multiple viewpoints (for a service hosted on a distributed platform, such as a Cloud system or a CDN). The main server-side method for detection of censorship is applying statistical analysis to the number and origin of clients connecting to the server. This kind of detection relies on the hypothesis that censorship events are country-wide in scope, and requires the possibility of tracking at least the country that is the source of the connections, e.g., relying on *IP address - to ISP - to country* mappings, also named *IP Geolocation*.[18] Examples of statistical server-based censorship detection activities are the *Google Transparency Report—Traffic* page [78] and the *TOR metrics portal* [121]. Another example of server-side censorship detection is the case of Google's analysis on reported malfunctioning of the search engine from mainland China users: the presence of some specific characters in a search query caused the connection to the main Google website to be interrupted for a minute or more, showing "The connection was reset" error.[19] The response from Google was to warn the user when the "sensitive" keywords (characters possibly contained in simple everyday use words) were used in a search query. This practice has been quietly discontinued, as reported by Greatfire.org (Greatfire.org News Blog [67]). A peculiar case to be considered is a form of server-based *detection* of server-based *censorship*. Explicit requests of blocking target content are issued to Online Service Providers: a notable example is given by Google Transparency Report—Removal Requests [77] that discloses a summary of requests from governments or from copyright owners to block access to specific content. While the actual targets are not disclosed, a categorization of removal requests is done according to the reason and the type of requester and the related statistics are provided. At the time of writing (June 2014) considered reasons are most prominently Defamation, Privacy and Security, and then (each accounting for less than 5% and decreasing): Government Criticism, Impersonation, Adult Content, Hate Speech, Violence, Copyright, National Security, Religious Offense, Trademark, Electoral Law; Other (about 18%). A breakdown is provided according to the origin of the request: "court orders" or "executive, police, etc.", the first type being prominent for category "Defamation". For each country also the extent of non-compliance with the requests is given, along with the reason for not abiding by the requests. An interesting "out-of-band" channel is offered to users willing to notify the unavailability of Google services (and possibly of Internet connection): three phone numbers are provided to leave a voice message that will be automatically tweeted with hashtag indicating which region the calling number is located (when detected).

4.2.2 Gateway-based Detection: The passive analysis of network traffic at gateways in general can reveal information on the existence and the extent of censorship. This kind of study is obviously limited both in having administrative access to such devices and in the privacy issues involved in accessing users' traffic and disseminate the analysis results. This provides justification to the lack of publicly available analyses of this kind, with one notable exception [3] to the time of writing. The study proposed in [3] is peculiar not only for the deployment (gateway-based data collection), but also for the nature of the collecting devices, that are the ones actually enforcing censorship (acknowledged by the equipment producer as filtering proxies). In fact collected data are obtained in the form of the monitoring devices logs,

[18] An example of service offering this mapping both as lower-quality publicly available version is given by MaxMind http://dev.maxmind.com/geoip/geolite.

[19] The analysis is described in the official *Google Search* blog [53].

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*, http://dx.doi.org/10.1016/j.comnet.2015.03.008

leaked by a *hacktivist* group [142]. From this privileged and specific viewpoint the techniques, the targets, and the possible overblocking are exposed, albeit for the limited timespan and geographic position covered by the leaked data (9 days across July and August 2011, from seven devices in Syria). The findings show that the *triggers* can be either IP ranges, domains, keywords (substrings of the URL in the GET requests), and the response for censored *targets* can be either a redirection (to a censoring notice) or the generation of a communication error.

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

TABLE I: Bibliography on Censorship detection techniques and architectures

		Censoring techniques detected									Properties			
Paper	Year	BGP	DNS	IP/port filter	TCP disrupt	Soft-cens.	TLS	Keyword	HTTP	Server-side	Passive	Topology	Tomography	Notes
Dornseif [48]	2003	-	X	-	-	-	-	-	-	-	-	-	-	in-depth discussion of DNS tampering
Clayton et al. [31]	2006	-	-	-	X	-	-	X	-	-	-	X	-	time; 1 probe outside censornet, targets inside
Clayton [30]	2006	-	-	X	-	-	-	-	X	-	-	X	-	detecting UK "clean-feed" system
Ballani et al. [15]	2007	X	-	-	-	-	-	-	-	-	X	-	-	analysis of control- and data- plane data
Crandall et al. [33]	2007	-	-	-	X	-	-	X	X	-	-	X	-	architecture: *ConceptDoppler*; time sensitive
Dagon et al. [38]	2008	-	X	-	-	-	-	-	-	-	-	-	X	DNS tampering ;
Weaver et al. [151]	2009	-	-	-	X	(p)	-	X	-	-	X	-	-	device fingerprint
Antoniades et al. [12]	2010	-	(p)	X	-	-	-	-	X	-	-	-	X	architecture: *MOR*; leverages *Tor*
Kreibich et al. [94]	2010	-	X	X	-	(p)	-	-	(p)	-	-	-	X	platform: *Netalyzr*; network neutrality analysis
Park and Crandall [116]	2010	-	-	-	X	-	-	X	X	-	-	-	X	web proxies as probes
Dainotti et al. [39]	2011	X	-	-	-	-	-	-	-	-	X	-	-	analysis of BGP, traceroute, unsolicited traffic
Holz et al. [73]	2011	-	-	(p)	-	-	X	-	-	-	-	-	-	port 443 reachability; in-depth TLS/SSL certificate analysis
Polverini and Pottenger [119]	2011	-	X	-	X	-	-	X	X	-	-	-	-	Machine Learning (clustering)
Sfakianakis et al. [136]	2011	-	X	X	-	-	-	X	X	-	-	-	-	architecture: *CensMon*; fuzzy hashing content
Soghoian and Stamm [139]	2011	-	-	-	-	-	X	-	(p)	-	-	-	-	SSL tampering detection
Weaver et al. [152]	2011	-	X	-	-	-	-	-	-	-	-	-	X	platform: *Netalyzr*; DNS tampering in-depth
Xu et al. [159]	2011	-	-	-	X	-	-	(1)	-	-	-	X	-	geographical target selection, probes outside censornet
Aase et al. [2]	2012	-	-	-	-	-	-	-	?	X	-	-	-	client-based cens.; manual Weibo probing; time analysis
anonymous [10]	2012	(a)	X	-	-	-	-	(d)	-	-	-	X	-	scan IPv4 space for DNS injectors
Feng and Guo [54]	2012	X	X	(p)	(p)	-	-	X	X	-	-	X	-	mainly focused on topology (traceroute analysis)
Filastò and Appelbaum [57]	2012	(p)	X	X	X	-	X	X	X	-	-	X	X	platform: *OONI*;
Holz et al. [74]	2012	-	-	(p)	-	-	X	-	(p)	-	-	X	-	platform: *Crossbear*; SSL certificate analysis
Verkamp and Gupta [147]	2012	-	X	X	X	-	-	X	X	-	-	-	-	manual inspection of packet traces, crafted servers
Winter and Lindskog [157]	2012	-	-	X	X	-	X	-	-	-	-	-	X	specific for Tor [46];
Anderson [9]	2013	-	-	-	-	X	-	-	-	-	-	-	-	NDT in μTorrent; RTT packet loss, throughput
Aryan et al. [13]	2013	-	X	X	X	X	-	X	X	-	-	-	X	obfuscation
Dalek et al. [41]	2013	-	-	-	-	-	-	-	X	-	-	-	-	identification of censoring devices
Esnaashari et al. [52]	2013	-	-	(p)	-	-	-	-	X	-	-	-	-	architecture: *WCMT*
Gill et al. [63]	2013	-	X	-	X	-	-	(p)	X	-	-		-	data from ONI, rTurtle
Hiran et al. [72]	2013	X	-	-	-	-	-	-	-	-	X	-	-	analysis of control- and data- plane data
Khattak et al. [86]	2013	-	-	(p)	X	-	-	X	-	-	-	X	X	black-box approach
Nabi [114]	2013	-	X	(p)	X	-	-	X	X	-	-	-	-	tool: *Samizdat*
Winter [156]	2013	-	X	X	X	-	-	(t)	X	-	-	X	-	all specific to Tor [46]
Zhu et al. [160]	2013	-	-	-	-	-	-	-	(c)	X	-	-	-	queries Weibo API
Fu et al. [61]	2013	-	-	-	-	-	-	-	(c)	X	-	-	-	platform: *Weiboscope*; Weibo API, differentiated probe frequencies
King et al. [87]	2013	-	-	-	-	-	-	-	(c)	X	-	-	-	extensive set of Chinese user-generated-content providers
Aceto et al. [4]	2013	-	X	X	X	(p)	X	(p)	X	-	-	-	-	architecture: *UBICA*; throughput, delay
Ensafi et al. [51]	2014	-	-	X	-	-	-	-	-	-	-	-	-	tool:*Spookyscan*; peculiar deployment/setup
Abdelberi et al. [3]	2014	-	-	X	-	-	-	X	X	-	X	-	-	analysis of (leaked) filtering devices logs

X yes; - no; ? undisclosed, unclear; (1) just the well known `falun` keyword; (a) collateral censorship due to routing; (c) evidence collection and content analysis; (d) forged variations of domain names are used as triggers; (p) potentially or partially; (t) SNI in TLS handshake ;

4.3 Detection architectures

Besides measurement in itself, detection and monitoring imply other accessory activities such as selection of *targets*, deployment and activation of probes, analysis and publishing of censorship evidences. The execution of these activities characterizes a censorship detection *platform* as opposed to a detection *tool* (this aspect will be analyzed in more detail in Section 6).

Architectures performing—up to different extent—the aforementioned activities have been proposed in literature; we describe in the following the most notable ones, in chronological order of publication, highlighting their specific contributions to the state of the art.

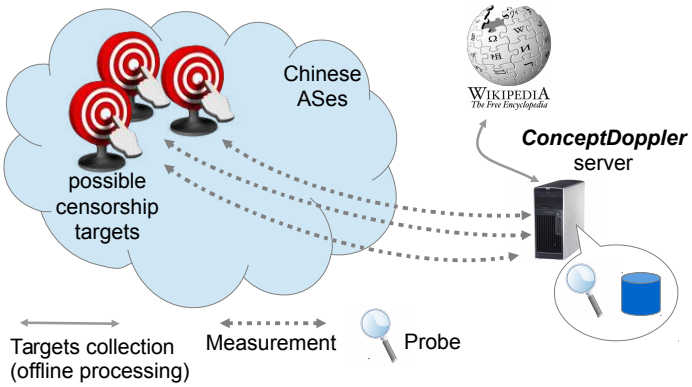

Fig. 7: The *ConceptDoppler* architecture diagram.

4.3.1 ConceptDoppler: An architecture is proposed by Crandall et al. [33], with the aim of providing a "weather report" for censorship, or in other words the description of evolution in time of *keywords* based censorship.

The architecture, whose prototypical implementation has been field tested in investigating the Great Firewall of China, is composed by a probing component, a database, and an—off-line—keyword extraction module. A representation of the deployment adopted in testing is shown in Fig. 7. This architecture does not manage multiple probes (all active measurements are performed from a probe outside of the censored network), and the control is centralized being local to the probe.

The probing method is both topology-aware and time-aware, as it locates the surveillance devices (by adopting an increasing TTL for probing packets that carry as a *trigger* the payload containing a blacklisted keyword) and measures the duration of the "blocking" state for the stateful GFC. To cover a diverse set of internal Chinese networks, probe packets have been addressed to IPs associated to the top 100 subdomains of the `.cn` top level domain returned by queries to the Google search engine, while the probe is located outside China at more than 10 hops from border routers belonging to Chinese ASes.

While advocating for continuous probing and reporting, and thus for more complexity and functionalities with respect to a bare censorship detection *tool*, ConceptDoppler as described by Crandall et al. [33] represents only one potential step towards a censorship monitoring architecture. The key contribution in presenting this architecture is the proposed solution to the "needle-in-a-haystack" problem of selecting the *keyword* to be used in probing the censoring system: this issue is analyzed in more depth in Section 6.5.

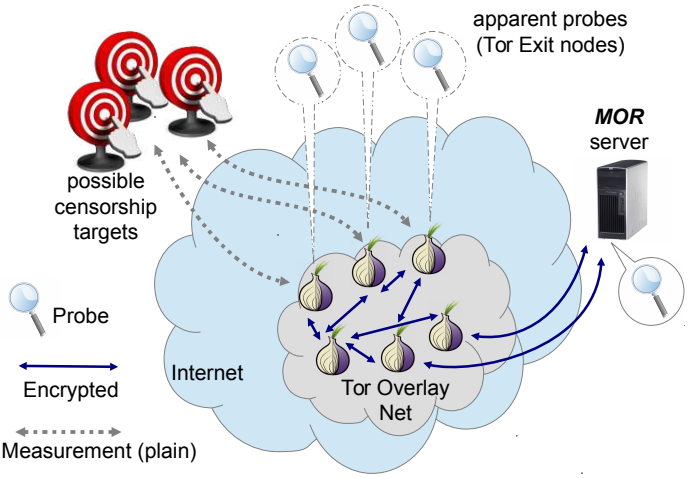

Fig. 8: The *MOR* architecture. Figure inspired by Antoniades et al. [12, fig. 1].

4.3.2 MOR: The Tor [46] platform is used by Antoniades et al. [12] as a free, globally distributed network of proxies to perform measurement of TCP filtering and HTTP tampering. The main property of the Tor overlay network that is leveraged to this aim is the possibility to choose the *exit node*, i.e. the proxy server that will generate and receive network traffic (DNS and TCP only) on behalf of the probe. The overall scheme of *MOR* deployment is depicted in Fig. 8. The probe traffic thus is encrypted and tunneled through the overlay network, and traverses the Internet eluding the possible *surveillance* devices until it comes out from the *exit node*. If the Internet path between the *exit node* and the *target* is free from *surveillance* devices then the unmodified *target* can be accessed, otherwise a censoring *action* will be triggered. In this architecture control is centralized, as all of the *viewpoints* (the Tor *exit nodes*) are contacted from a server accessing the Tor network, and the collection of measurement results is automatically performed through the overlay network again, when responses to the *exit nodes* are tunneled back to the controlling probe. While Antoniades et al. [12] does not focus specifically on censorship, it explicitly lists censorship detection (for "Web-Page Censorship", i.e. HTTP tampering) among the use cases of the proposed architecture. The detection tests and analyses discussed in the paper presenting the architecture are: blocking of the *Skype* Instant Messaging application (by checking reachability of the webserver employed for the initial log-in phase); the occurrence of TCP filtering (based on the port); HTTP header tampering (by leveraging controlled web servers that are used as *target* of the requests, thus employing a

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

tomography-like setup); given the features of the Tor platform (tunneling of DNS requests) also DNS tampering tests could have been performed, but the authors do not consider this kind of censorship technique, suggesting instead the analysis of DNS system dynamics as use case for the platform. No details are provided on the tasks of *target selection* and analysis and publication of results, leading to the conclusion that they are not automated, and performed ad-hoc manually.

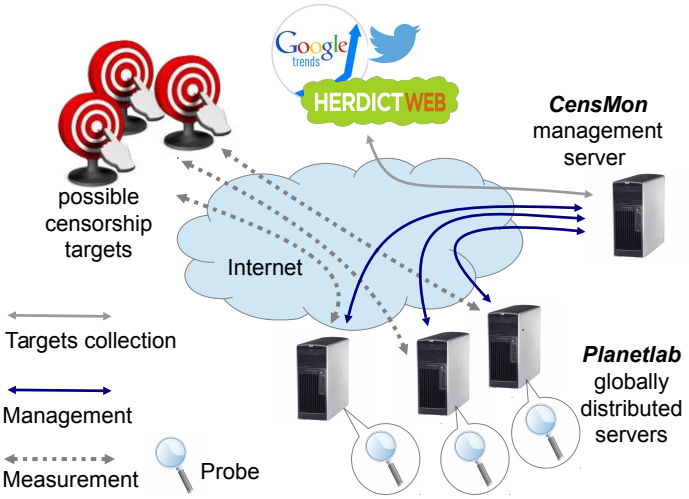

Fig. 9: The *CensMon* architecture.

4.3.3 CensMon: A complete architecture named *CensMon*, specifically designed for censorship detection, has been introduced by Sfakianakis et al. [136]. As a monitoring architecture, it is designed for continuous and automatic functioning (opposed to spot detection and manual control of simple tools), and it faces the problem of selecting *targets* worth checking among the entirety of the World Wide Web. The target selection problem has been addressed by feeding the system with URLs automatically harvested from online sources and is discussed in more detail in Section 6.5. An abstracted representation of the *CensMon* architecture is shown in Fig. 9. The deployment of this architecture leverages the *PlanetLab* platform [28] as probes (collectively named "Agent Overlay Network" in the architectural description), controlled centrally by a management server. The management server is in charge of target collection and selection, probe activation, collection of results, their analysis and storage in the local database. The detection procedure, split in the phases of *evidence collection* and *analysis* is listed in Fig. 10. A notable aspect of the considered tests is the usage of a "Web Helper", i.e. an external web server that is supposed not to host censored *targets*; such server is requested a web resource whose URL comprises part of a potentially censored *target* URL, in order to detect *keyword*-based censorship techniques.

Another notable characteristic is the analysis of HTML content, performed using MD5 hashing to weed out the case of no changes, then by extracting only readable content, applying fuzzy hashing on it and then compare such hashes across

INPUT: URL
1) distribution of URL to probes
2) DNS resolve → output: noerror, noxdomain, timeout, connrefused
3) TCP connect to resolved IP, port 80
4) URL filtering: keyword=URL postponed to WebHelper URL (expecting 404 not found)
5) HTTP access to URL, get status code, HTML (if redirection: last URL and IP visited)

Server-side analysis:
6) DNS tampering:
   a) (whois from server)
   b) coherence of DNS resolution from different probes
7) HTML content check:
   c) md5sum for coherence among probes
   d) arc90 readability extraction → fuzzy hashing → comparison among probes
8) inaccessibility event → probe that got it keeps monitoring the URL

Fig. 10: *CensMon* [136] detection procedure (inferred from the textual description).

multiple *viewpoints* (as described in Section 3.8).

Finally, the *CensMon* architecture leverages repeated evidence collection for a target that has been found unaccessible, in order to tell outages (temporary) from intentional filtering (persistent over the repeated checking). No detail is provided about how much time or how much evidence collection attempts are considered enough to discriminate censorship from outage.

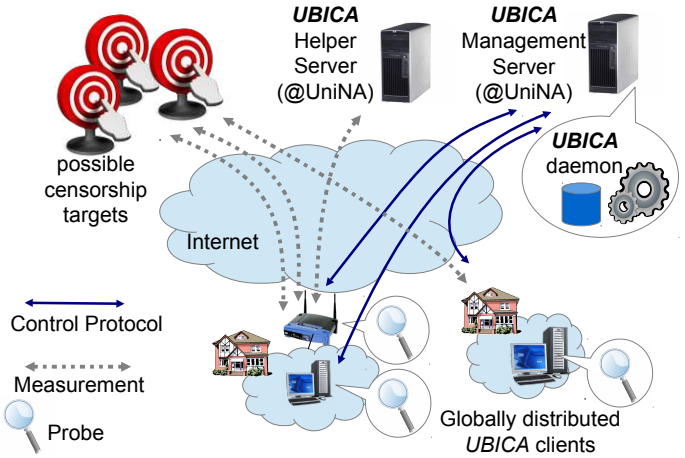

Fig. 11: The *UBICA* architecture diagram.

4.3.4 UBICA: The "User-Based Internet Censorship Analysis" (UBICA) platform is a research project from University of Napoli "Federico II" aimed at continuous monitoring of Internet Censorship.[20] With respect to previously described

[20]The authors of the present survey are the P.I. (prof. Pescapè) and the co-designer and co-developer (Dr. Aceto) of the platform.

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

architectures, and specifically with the most similar one, *CensMon*, UBICA presents the following properties:

- a diverse probe set (leveraging, in addition to PlanetLab nodes, also home gateway devices and desktop OS executables);
- complex test management (allowing for upgrade, selective test activation, per-probe test parameters tailoring, aggregates of probe - campaigns);
- incentives to user adoption (an aspect structurally missing from CensMon as its probes are not intended to be run on users premises);
- a reporting system, differentiated for the operators of the platform, its users, and the general public.

The *probes* are implemented in two versions: (i) an executable (for Mac-OS-X, Windows and Linux platforms) with GUI; (ii) a bundle of `ash` scripting and standard UNIX utilities (working as a daemon, with only a command-line interface).

The command-line-only version has been deployed on home gateways of the BISMark project (Sundaresan et al. [140]), in the form of pluggable OpenWRT package.[21]

The possibility of running tests from different types of networks (academic and research networks, residential houses, nomadic users) enhances the detection capabilities and the depth of analysis of censorship evidences.

A schematic diagram of the UBICA architecture in shown in Fig. 11. In the diagram the components of the architecture are shown: the Management server, the distributed *Probes*, the Helper server, and the *targets* to be checked from the *Probes*.

The UBICA management server operates following a control cycle that comprises several phases, namely:

1) Collection of Targets
2) Scheduling of evidence collection
3) Evidence collection
4) Censorship Tests
5) Evidence reporting and data export
6) Update Targets and Scheduling criteria

During these phases, in order to perform its operations, the management server interacts with external components such as sources of information, the platform probes, and clients of the portal services.

Tests performed by the UBICA platform are shown in Table I in correspondence to the reference Aceto et al. [4]. Results obtained by operating the platform are presented in [5].

4.3.5 WCMT: The "Web Censorship Monitoring Tool" as presented by Esnaashari et al. [52] consists of two different architectures, one devoted to detection of HTTP tampering and the other to detection of a port-based filtering (a subset of IP/port filtering). Despite the preliminary nature of the work, the paper proposes an interesting variation on the detection of HTTP tampering.

[21] Publicly available at https://github.com/sburnett/bismark-packages/tree/f383d68fdee2c5fbd027114c0cf355dc79e83f5a/utils/pakistan-censorship-testing .

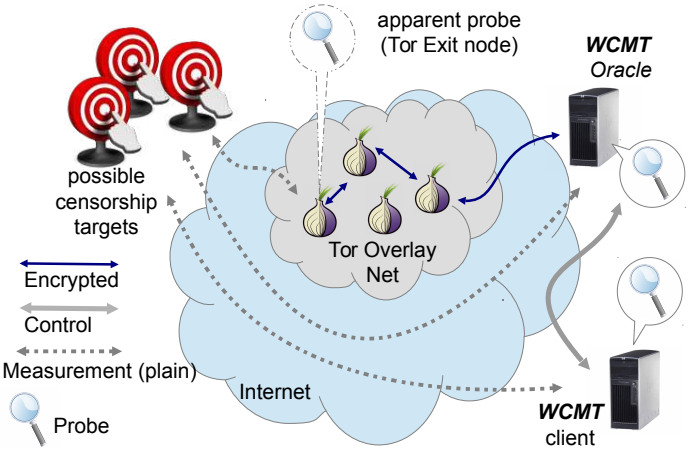

Fig. 12: The *WCMT* architecture diagram – *HTTP tampering* detection.

The architecture for HTTP tampering detection is shown in Fig. 12 and consists of three components: a client to be manually operated by the user, which provides the *targets* to be tested in the form of a list of URLs and activates the testing procedure; an "Oracle" server, that is prompted by the client to retrieve the same *target* it is testing, both directly and through the Tor ([46]) overlay network; a censorship analysis engine, that receives and compares the three versions of retrieved content respectively from the client, from the Oracle directly, and from the Oracle through Tor. The comparison is performed—in sequence—in terms of the following properties: overall length, header length, body length, header content, body content, number of and size of images; any of these tests showing differences is considered as censorship evidence. The algorithm compares first the couple of responses (client, Oracle-direct): a difference is considered as evidence of censorship enforced at organization level, i.e. the surveillance and censoring device is supposed located inside the client network or at its gateway to the Internet; if no difference is detected, then the couple of responses (client, Oracle-Tor) is compared: a difference is considered as evidence of censorship enforced at country level.

Implicit hypotheses for the proposed detection algorithm to work is to have the Oracle topologically located in the same country of the client, but outside of its organization network.

The architecture proposed for "service blocking" consists in a tomography deployment with an Helper Server accepting connections to whichever transport port the client tries to access, providing a report about the success or failure of the attempt. The evident limitation of this architecture is that known IP/port filtering is usually triggered also by the IP address of the *target*, not the transport port only: in this case the presented architecture would require to deploy helper servers on each potential *target* or fail detection systematically. Therefore it can be used only to detect indiscriminate blocking of a transport protocol port.

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*, http://dx.doi.org/10.1016/j.comnet.2015.03.008

TABLE II: Detection Platforms and Tools: detected techniques

	Detection system	rTurtle* [63]	Herdict	Alkasir◇	YouTomb	Greatfire.org	OONI [57]	Weiboscope [61]	Samizdat [114]	Spookyscan [51]	encore
	Year of first release	2009	2009	2009	2010	2011	2012	2013	2013	2014	2014
Detected technique	BGP tampering	-	-	-	-	-	-	-	-	-	-
	DNS tampering	X	-	◇	-	X	X	-	(c)	-	-
	DNS hijacking	-	-	-	-	X	-	-	X	-	-
	DNS injection	-	-	-	-	X	-	-	X	-	-
	Packet filtering	X	-	◇	-	X	X	-	X	(d)	(e)
	TCP disruption	(m)	-	?	-	X	(p)	-	-	-	-
	Soft Censorship	-	-	-	-	X	(p)	-	-	-	-
	TLS tampering	-	-	◇	-	-	X	-	-	-	(e)
	DPI / Keyword blocking	X	-	◇	-	X	X	-	(w)	-	-
	HTTP tampering	(p)	X	◇	-	(p)	X	-	(p)	-	(e)
	Server-side Censorship	-	-	-	X	X	-	X	-	-	-

X yes; - no; * distributed by researchers to volunteers; ◇ indirect inference: possibility to detect technique but just "blocked" in reports or documentation;
(c) uses also a CDN, in [114] no keyword blocking has been detected;
(d) detects also direction (client-to-server or server-to-host); (e) can detect unreachability, but does not infer the censorship technique; (p) potentially or partially;
(w) uses also web caches, in [114] no keyword blocking has been detected;
(m) to tell censorship from outages comparison with other probes and over time is used.

5 Censorship Detection Platforms and Tools

The tools and platforms employed for Internet Censorship Detection or Monitoring are not numerous but quite heterogeneous. A common baseline can be abstracted from their nature of tests: as such the detection can be broken down to the same components and phases described in the previous section. Besides this, they differ in the approaches that have been adopted in many aspects such as the degree of automation (ranging from testers having to run manual checks [54] up to virtually unmanned censorship monitoring platforms [136]), or the control paradigm, ranging between centralized control [70] to completely distributed [57]. We coarsely aggregate this variety in two general classes, distinguishing *tools* and *platforms* on the basis of the complexity and comprehensiveness of the activities they perform (see Section 6 for a more detailed discussion on this). In the following we describe the detection systems for which an implementation is available or known, listed in chronological order of publication as reported also in Table II, ending with a section on other diagnostic tools that have been employed for censorship *detection*.

5.1 rTurtle

The data backing the analyses performed by *The OpenNet Initiative* [79] has been collected by means of an undisclosed client, but a detailed analysis of the dataset has been published in [63]. In this paper an algorithm for the detection of censorship based on collected data is reported (see Fig. 13):

```
 1 IF NOT got DNS reply
 2     THEN outcome is ``DNS blocking''; END.
 3 check presence of DNS redirect
 4 IF NOT got reply to SYN
 5     THEN outcome is ``IP blocking''; END.
 6 ELSE IF NOT got response to HTTP request
 7         THEN outcome is ``No HTTP Reply''
 8         ELSE check whether outcome is:
 9                 ``RST'', or
10                 ``Infinite HTTP Redirect'', or
11                 ``Block Page''
12 END.
```

Fig. 13: Detection algorithm applied to *ONI* data: pseudocode as inferred from [63].

if the test finds evidence of blocking possible motivations are: DNS blocking, IP blocking, No HTTP Reply, RST, Infinite HTTP Redirect, Block page.

From the description of the tests it can be inferred that this tool performs HTTP GET requests towards given *targets* and collection of the HTTP replies and application-level logging of errors. The censorship techniques that are detected with this tool are reported in Table II. We notice that the detection of HTTP tampering is partial, as the analysis of the content of the returned HTTP response is limited to looking for a manually-compiled list of character patterns corresponding to block pages.

The deployment architecture shows no centralized control, with probes distributed manually into selected countries and operated by users. The reporting is not automated, and the

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

analysis is centralized and not automated. One of the detected techniques (TCP disruption) leverages results from different probes in a country and test repetition over a time span (a week).

5.2 Herdict

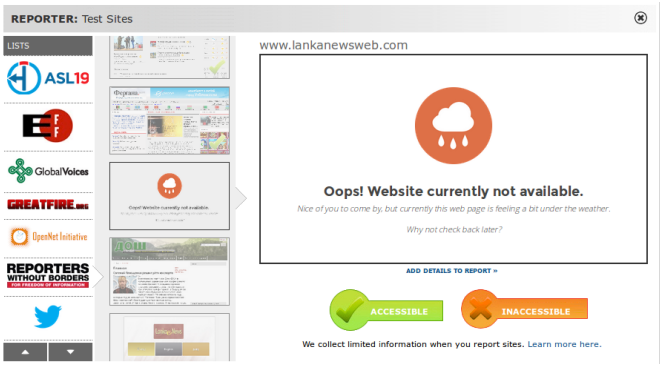

Fig. 14: The *Herdict* platform: a screenshot of the submission form. On the right a frame showing the *target* as retrieved by the browser (in an i-frame), and below the two buttons to report: "accessible" or "inaccessible". On the left a menu of URL lists proposed by affiliated associations is shown.

The "Herdict" platform [70] is a crowd-sourced censorship monitoring system. Its main interface consists of a website allowing users to report about "accessibility"of URLs from within their browser; this way the platform leverages crowd-sourcing both for the collection of *targets* of interest for the users, and for performing application-level censorship test. The URLs to be checked are both suggested by a limited number of affiliated organizations and provided by the users themselves. In Fig. 14 a screenshot [22] is shown of the form for the submission of a test verdict: by clicking either on "accessible" or "inaccessible" the user provides a verdict about the reachability at *application* level. The simple mechanics of this method consists in providing the user some URLs of interest and receive, together with her verdict on accessibility, also the IP address of the user (as her browser is connected to the Herdict website), and a few accessory browser-related data, so that the user location can be inferred through geolocation and the verdict can be aggregated by the *source IP* of the user/probe.

An additional interface is provided through a browser module ("plug-in") that: (i) collects and sends to the Herdict platform the URL of each website visited by the user, returning a response about the accessibility of that website according to the Herdict database; (ii) allows the user to quickly report whether the website is accessible or not. The first functionality can be disabled by the user. Moreover a registration form is provided to the user to describe her location (that will be associated with all the user's report). This constitutes another form of metadata collection about the condition of the accessibility measurement.

Given the mechanics of this detection method, it can be considered as testing the censorship of the targets at *application level*, as the whole protocol stack must have worked unimpeded in order to provide the final result of the webpage rendering (thus including HTML and script processing). Being more precise, as user judgment is involved in assessing whether the results she sees are to be considered an "access" or not, this should be considered an "user-level" [23] censorship test.

5.3 Alkasir

Alkasir is mainly meant as a circumvention tool dedicated to a restricted list of websites. The tool is part of a system comprising a website [8], a client application, and an undisclosed set of proxies that are used to tunnel the user traffic towards blocked websites. The tool is not open source, while it is downloadable for free. The list of URLs to which access is granted is managed on a per-country basis: users are allowed to tunnel their traffic through the Alkasir proxies provided that the following conditions are all met:

- the user herself or another user from the same country has reported the URL as blocked
- the operators of the Alkasir system have policied the URL as complying to the Alkasir policies for URL submission [145]
- the Alkasir system validates that the submitted URL is blocked from the user country

The submission of the URLs is allowed only through the Alkasir client, in the form of a URL list. The user will be notified if and which URLs have been validated as being blocked from her country and compliant with the URL submission policy.

A report of URLs considered as blocked by the system is given in the form of *google map* embedded in the product website, reachable from the platform website [8]. Web pages reporting per-country lists of blocked URLs are also provided.

The detection technique of the Alkasir tool is not disclosed. Nevertheless this system has been considered among censorship detection platforms as it actually can be used as a detection service, given the public availability of the results of blocking detection. By making the client update the information of blocked URLs and inspecting traffic traces generated by the client[24], we have verified that the *targets* are checked for reachability at DNS, TCP, HTTP and TLS levels, therefore the platform has the possibility to detect at least the phase of communication that is tampered with. Curiously only country-level granularity is actually considered to choose if tunneling through Alkasir proxies is to be used, while the

[22] as of March 2014

[23] a humorous expression with some popularity is "at layer 8", with reference to the 7 layers of the ISO/OSI network model [132], the upper one being the application protocol (roughly corresponding to the top sublayer of TCP/IP application layer).

[24] version 1.4 build 005

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

platform detects the user ISP and could restrict the list of tunneled URLs to those blocked from that specific ISP. These findings are confirmed by considering Alkasir blocking reports and comparing them with what is known in literature about censoring systems. Despite this in the reports no detail about the blocking technology is present, so as a detection platform it is of limited usefulness.

The shortcomings of the Alkasir system as a censorship monitoring platform derive mainly from its intended usage as a censorship *circumvention* tool: the analysis and reporting of censorship is intended as a mean to limit the traffic that users can impose to the Alkasir servers, and does not have a monitoring objective besides this. The reason, as stated in the website FAQs [8] is that having limited resources (bandwidth, proxy servers), the restriction of circumvention only to selected categories of *targets* that actually need it is necessary to provide the service to a larger user base.

5.4 YouTomb

The *YouTomb* platform is the product of a research project, created by the MIT chapter of the Free Culture Foundation[25], aimed at monitoring server-based censorship on the video publishing platform *YouTube*.[26] The project started in 2008 and has been discontinued since 2009 but the platform public interface is still online [23]. Most notably, the code of the platform is publicly available [22] under the Free Software license AGPL. The videos to be checked for availability (the *targets* for this platform) are periodically collected from a limited set of sources, performing data extraction from their web interfaces (including the most popular videos of *YouTube*). The detection method, as for similar server-based detection platforms, leverages the explicit censorship notices provided by the service itself. The set of monitored videos are periodically accessed until censorship is detected, then the time of publication before censorship ("takedown") as well as the takedown reason are recorded. For each monitored video the result of detection is represented by a "status" label, varying among

- `up` – not censored;
- `down:tos` – violation of Terms of Service;
- `down:user` – the user herself has removed the video;
- `down:copyright:`*holder* – the copyright holder in the status line has requested the takedown;
- `down:other` – down for `unknown reason`;
- `unknown:private` – status is unknown because video is private;
- `unknown:only18` – status is unknown because video is rated for adult audience.

The results of the censorship detection are exposed through the web interface, where the list of recently censored videos with a thumbnail image and some metadata are kept updated. The platform allows for search based on title text and copyright holder (provided as hypertext links in the status label), and results are shown ordered chronologically, or by the number of views of the video; for each reported video the shown metadata are: a link to the YouTube page of the video; YouTube user id (and link to profile); time of upload; video status (including the copyright holder, if censored); time of takedown ; the "Description" text and the Tags; along with a reduced-size version of 4 still frames of different parts of the video.

Though the monitoring has been discontinued, the reports of results for the period 2008-2009 and above all the availability of the source code, make this platform well worth considering.

5.5 Greatfire.org

The platform Greatfire.org provides a web interface [66] that shows the results of a monitoring campaign of censorship enacted in China. It also allows the user to submit URLs and keywords to be tested for blocking: blocked *targets* will be checked repeatedly. Although being restricted to one specific country and not disclosing much details about the detection methods and infrastructure *Greatfire.org* is a notable source of information about Internet Censorship.

Besides user-submitted URLs and keywords, documentation on the website states that the targets to be monitored are automatically gathered from "friend" projects, namely: Autoproxy [118], China Digital Times [24] and Herdict [70]. The reports are organized in the following categories: *Alexa Top 1000 Domains, Baidu Searches, Blocked, Domains, Google Searches, Google Sites, HTTPS, IP Addresses, Keywords, New York Times Chinese Articles, One Character Keywords, Triple Blocked Keywords, URLs, Weibo Searches, Wikipedia Pages*.

Both from the reported data and the textual descriptions in the FAQ and blog sections of the website it can be deduced that the detection technique is active, and the censoring techniques that are detected are the ones reported in Table II. With respect to the censorship detection tests as described in Section 4, some specializations are present: the platform is able to perform detection of one kind of server-based censorship, labeling as "self-censored" the keyword that generate an explicit notification of censorship from the *Baidu* Chinese search engine and the search facility of the *Sina Weibo* Chinese microblogging platform. In both cases detection is performed by looking for the explicit censorship notification text in the response triggered by a search query containing the keyword.

The detection architecture, as deduced from the report data, consists of at least one US-based probe and probes in Chinese networks. The tool that is used to generate the triggers is `curl`, and the tests to infer the censorship technique are performed by collecting the response to the triggers from inside the Chinese networks (up to 4 probes can be seen in a random sampling of the results) and comparing them with responses got from inside the USA.

Another notable characteristic is that the platform detects soft censorship or *throttling*, by evaluating the average down-

[25] http://freeculture.org
[26] http://youtube.com

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

load rate and labeling the target as ‘otherwise restricted‘ if it is smaller than $5kbps$.

5.6 OONI

The OONI project [146] has designed a censorship detection architecture that has been presented by Filastò and Appelbaum [57] and at the time of writing (June 2014) is in active development. It is a Free Software project, part of the wider *Tor Project* [120] with which it is tightly integrated. The main component of the architecture is the *ooniprobe* tool, that can perform several different tests in trying to access either the *target* or one of the *helper servers*. The other component is a back-end, named *oonib*, that is implemented as a deployment of servers with the role of helpers for tests that require the control of both sides of the communication. Moreover they serve as repositories for the report of test results. The whole architecture is being integrated in M-lab, under development at the time of writing.

The *ooni-probe* component can be used as an independent, locally controlled tool; it is written in Python, benefiting from the high-level libraries available for this language to deal with networking.

Censorship detection tests that are supported by *ooniprobe* are:

- Content Blocking Tests
  - DNS Consistency
  - HTTP Requests
  - TCP Connect
- Traffic Manipulation Tests
  - HTTP Invalid Request Line
  - DNS Spoof
  - HTTP Header Field Manipulation
  - Traceroute
  - HTTP Host

The first group (Content Blocking) requires in input one or more *targets* to be checked, while the second group (Traffic Manipulation) does not, and can be performed by interacting with an helper server.

The Content Blocking tests present two phases: (i) generation of probe traffic; (ii) analysis of outcome or comparison against a ground truth.

The ground truth is obtained using the Tor application [46], a transport-layer proxy providing access to an overlay network designed for privacy and used also for censorship circumvention. In fact the Tor application running on the same host of the probe offers to the local network applications a SOCKS 5 [100] proxy server that tunnels TCP and DNS traffic in an encrypted circuit used to traverse the *surveillance* device without exposing any *trigger*. The *targets* accessed through the tunnel are considered as not tampered.

5.7 Weiboscope

The *Weiboscope* platform has been presented in [61], where it has been leveraged to analyze server-side censorship of the Chinese microblogging platform *Sina Weibo*. The platform presents a web interface [83] where the latest censored messages are reported in form of a list of Chinese text (with automatic mouse-over translation to English) at the side of a screenshot of the deleted message together with the date of creation and of deletion (Fig. 15). From the aspect ratio of the screenshot and the icons (showing signal strength and battery) always present at the top of it we can infer that a mobile browser (possibly an emulated one) is used to render the message page and take the screenshot, and possibly also to retrieve the web page, in order to have a version that is lighter and easier to parse.

Fig. 15: *Weiboscope* home page with latest censored messages (detail of top left quarter); on the left a screenshot of a mobile browser rendering of the censored message is visible, on the right half can be read the post metadata and an automatic translation of part of text.

At the bottom of the page a graph is generated of the time variations of the “Censorship Index”, defined as $10^4 \frac{C}{N}$, where C is the number of censored posts and N is the total number of published posts; this number gives the ratio of censored messages per 10^4 published ones, and being shown with day granularity implies averages on the day (values for the first half of 2014 show a minimum of 14.11 and a maximum of 70.1). The geographical location of the original poster (as provided by the *Sina Weibo* API) is used to draw a heat map of the daily value of the Censorship Index per province (only for the last week), showing at a glance in which provinces the authors have been censored most.

Collected data is the result of tracking the publishing history of a selected sample of *Sina Weibo* users, according to the methodology described by Fu and Chau [60].

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*, http://dx.doi.org/10.1016/j.comnet.2015.03.008

The database exposed by the platform can be queried through a text form for terms appearing in the deleted messages, and the whole database of year 2012 (with anonymized username and ID for the authors of messages) is available for download [82] as CSV files (split per week). The fields of the exported database contain the anonymized IDs of message, user, and original message (if it is a *re-post*, i.e. a referenced quote, of another message), the text of the message and creation and deletion dates.

An example of reported statistics for the database is:

weibo messages	226,841,122
deleted messages	10,865,955
censored messages	86,083
unique weibo users	14,387,628

Differently from the *Greatfire.org* platform, *Weiboscope* does not allow the user to suggest sensitive terms to be checked, and acts only as a reporting front-end.

5.8 Samizdat

The tool used to gather the data analyzed in [114] has been made available by the author, and is publicly accessible on the distributed code management platform GitHub [113].

The tool consists in a Python script that downloads from a blog URL [122] the list of targets to check for censorship then applies a list of tests of type DNS resolution, alternate DNS resolver check, web content access, and a variation on keyword-based web access. In DNS Tampering detection, the cases of hijacking and injection are told apart by issuing name resolution requests to the probe default resolver and to a list of open resolvers. A third type of DNS tampering check is done by leveraging the Coral CDN: the original hostname is appended with "`.nyud.net`" making it a subdomain of the CDN, and the accessibility of the new URL is checked. No automated comparison of results is made between results obtained through CDN and direct access, but potentially the tool could tell if there is any difference in the content if both are reachable or, if one or both are censored only for specific URLs, this could provide more insight in the mechanics of the censoring system.

Similar considerations can be made for the access to the *target* using a web search engine cache such as *google cache*, that allows access to a copy of the web resource located at a given URL by querying the search engine (therefore carrying the URL of the target as a parameter in the request string of the HTTP request). In this case neither automated comparison of results is made between results obtained through the web cache and direct access, but again potentially content modification and insight in the censoring mechanics could be inferred.

5.9 Spookyscan

Spookyscan is an implementation of the detection technique presented in [51] and described in Section 4.1.3 (TCP-level reachability). The detection technique is designed to reveal TCP-level reachability between a client and a server, with the notable characteristic that none of the two has to be under control of the tester. The applicability of the technique is limited to clients presenting a specific behavior for IP protocol.

The source code is publicly available [89] under the GNU General Public License; it is written in Python and has been developed for the linux platform, but it should be easily portable to other unices. A web interface is also present [90], performing the test and reporting the results in real time.

The input data that has to be provided are:

- **server IP address** of the target whose accessibility from the probe we want to test
- **server TCP port** of the target
- **client IP address** of the probe

If the preconditions for the detection algorithm are satisfied (predictability of the IP ID field on the client) the report will show if the client can communicate at TCP level with the server, or which side of communication is blocked. Tests data are stored and accessible at a specific URL, valid for up to one week after the test request, if not explicitly deleted; thus no reports of other people tests are provided, nor repeated monitoring is performed.

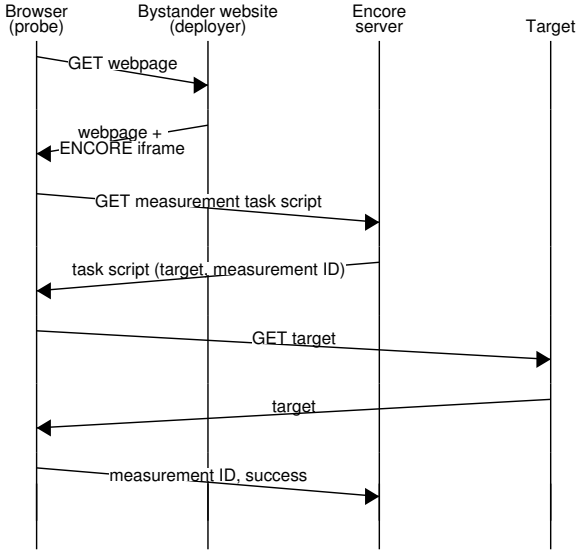

Fig. 16: The *encore* tool: sequence diagram of evidence collection (inferred from code [17]).

5.10 encore

The *encore* tool collects evidence of censorship from (possibly unaware) visitors of web pages. The design principle is borrowed from third-party tracking techniques (Roesner et al. [131]), where instrumented web pages make the client browser perform (network) activities by executing script from – and reporting data back to – third party servers. The application deployment, as inferred from the description on the project page [62] and the available code [17], is composed of a server

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*, http://dx.doi.org/10.1016/j.comnet.2015.03.008

that performs *target* selection and acts as repository, and a number of websites participating in the recruitment (named "bystanders" as they do not participate in any phase but the recruitment of probes). The "bystanders" host webpages that are instrumented with an HTML `iframe` element linking the evidence collection code. The *probes* are the JavaScript-enabled browsers of users that land on the instrumented webpage. The sequence diagram representing the evidence collection activity is shown in Fig. 16.

The only type of test performed by the probe is a reachability test at HTTP (application) level. The measurement phase consists in the script requiring the *embedding* of an external web resource (the *target*); this makes the browser issue a HTTP GET request for the target resource and try to render the returned content. If both the fetching and rendering succeed (no matter what content has actually been received), the "success" condition is reported back to the evidence collection server; if either the fetching or the rendering fails, a "failure" condition is reported instead. According to the characterization of censorship detection system that we have proposed in Section 4, this tool in the published implementation only operates the *target selection* and *evidence collection* phases. The peculiar recruitment method does not allow to schedule the time of activation and the network location of the probe, as these join the system according to the user visits to the "bystander" websites. The collection of *targets* is delegated to the *Herdict* platform. No analysis of results is reported on the project website, and from the published code it can be inferred that evidence collection results are stored and processed by means of a *cube* time-series analysis system [36] (in turn based on a *noSQL* approach [21]). In any case the user participation is not engaged in any of the phases, being the recruitment and measurement phases performed in background and without any notice.

As *encore* exposes only limited functionalities and is operated on a single server we have classified it among the tools, even though it indirectly leverages an heterogeneous set of distributed components. It is evident from the nature of the collected evidence (HTTP reachability and rendering of web resource) that using that alone to infer censorship is prone to both false positives (e.g., temporary unreachability) and false negatives (a blocking page, an error page, and a content-mangled one all would return "success"). As a consequence, to be considered an actual censorship detection platform, this system has to heavily rely on the analysis phase, that is not currently disclosed. The fundamental strength of this detection system is to mitigate the necessity to engage the users and keep them participating in a detection system, as no activity is explicitly requested to them, and they visit the instrumented web page out of their personal interest and desire. This shifts the need to engage the users to engaging web publishers, but as each website participating in the system potentially recruits all its viewers, there can be a multiplication effect (many clients even for few "bystanders") that can be extremely significant for high traffic websites. This introduces the necessity on the one side to scale in order to support traffic bounced by mass websites, and on the other to avoid to generate too much artificial traffic towards the tested targets, that could adversely affect the interested servers and networks. None of these aspects is currently discussed on the project website [62].

5.11 Other diagnostic tools

The selective degradation of Quality of Service has been shown to be applied as a censoring technique (see Section 3.5); performance measurements and network neutrality checks can be in principle employed to detect such impairments but associated to censored *targets*. In fact the analysis of Internet Censorship in Iran described in [9] has been performed leveraging a performance analysis tool, *Network Diagnostic Tool* (NDT) [80]. One notable characteristic of this deployment derives from the nature of the probes engaging in measurements: NDT is embedded in a freely downloadable application, an implementation of the peer-to-peer file sharing protocol bitTorrent. Other peculiar characteristic, that potentially limits the usage of the tool for detecting targeted censorship, is the need of specifically instrumented servers (linux boxes with ad-hoc network stack modifications).

Other network diagnostic or performance assessment platforms have been cited in censorship analysis papers, but no result obtained directly from their usage has been reported. One such example is *Netalyzr* [94], a network diagnostic platform implemented as a Java application that can be downloaded and run from inside a browser, performing tests to detect presence of HTTP transparent proxies, DNS query rewriting and QoS parameters. While tests performed by *Netalyzr* can be in principle used to detect censorship, the authors explicitly avoid it for ethical reasons; nevertheless the methodology for detecting DNS tampering, and HTTP tampering in a tomography setup, are relevant to the literature on censorship detection and have thus been listed in Table I.

Another tool, specifically aimed at detecting traffic shaping or blocking, is *Glasnost* [47]. This tool uses a *tomography* deployment, and probes the network replaying a pre-recorded TCP bidirectional traffic trace and evaluating the achieved throughput. Tests for different applications are available (using the respective traffic traces), namely: BitTorrent, eMule, Gnutella, Email (POP), Email (IMAP4), HTTP transfer, SSH transfer, Usenet (NNTP), Flash video (e.g., YouTube) .

6 Characterization of censorship detection architectures, tools and platforms

From the analysis of literature (Section 4) and of available detection tools and platforms (Section 5) we have derived a characterization of censorship detection architectures, platforms and tools (collectively "detection systems"). In the comparison and discussion we have considered as *architectures* the detection systems for which no implementation is publicly available; besides this, they are evaluated and compared with the other systems according to the same set of properties,

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

whenever they apply. In our research we have considered fifteen among tools, platforms and architectures: to be able to compare such diverse set of systems we have selected the list of properties that is depicted in Fig. 17. For each property, a definition is provided in the following together with comments on its variability across the considered systems; when the concepts and the definitions have been introduced and discussed in previous sections the related reference is reported.

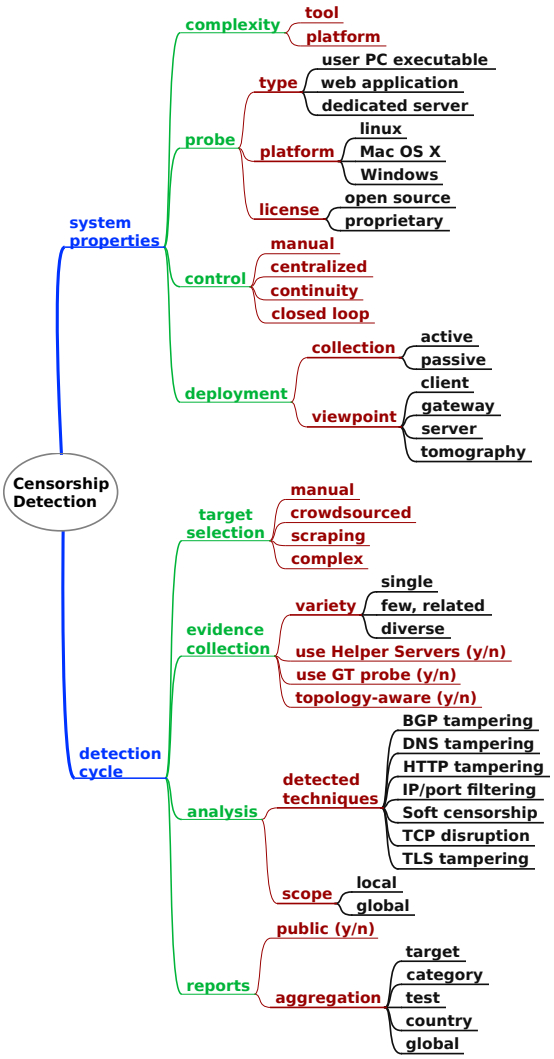

Fig. 17: Censorship Detection Architectures, Tools and Platforms: hierarchy of characterization properties.

The characterization and comparison of surveyed architectures, platforms and tools is also summarized in Table III, where for each of the considered detection system the relevant characterizing properties are evaluated, with the exception of the detected censoring techniques (already shown in Table II). Considered detection systems are listed in chronological order of release for the first implementation. For systems that have been presented in academic publications, such as *OONI, Samizdat, Weiboscope* and *Spookyscan*, the year of the paper has been considered; for others (namely *Herdict, Alkasir, YouTomb, Greatfire.org*), the historical or presentation information provided on the system home webpage has been taken in consideration; for *encore* the code publication time has been taken from the source repository; for *rTurtle* the date of the citation in [57, ref. 13] is considered as year of first release.

The list of properties definitions and discussion follows.

6.1 Complexity

The final goal of detecting and monitoring censorship is reached through several steps, of which the measurement is but one. Compared with detection platforms, tools are more limited in functionality, being focused mainly on the measurement aspect, and lack most if not all of the automation facilities offered by platforms; these in turn can offer different degrees of automation and completeness. The steps in which a Censorship Monitoring platform pursues its goal are:

- target collection and selection
- deployment and selective activation of probes
- collection of measurement results
- analysis of measurement results
- publishing of censorship detection
- update of detection tests and criteria

Moreover a censorship detection platform can provide management servers, repositories of test results and of analysis results, and helper servers to perform tests requiring *tomography* setups.

In platforms the analysis of measurement results can be performed leveraging multiple probes, while tools are usually bound to local probe results. This allows also to install a *tool* on a single host, acting as a *probe*, without the complication involved in setting up a distributed platform, where non-trivial system administration skills may be needed besides the need of administrative access to multiple geographically distributed hosts.

Referencing Table III, we can see that most of the considered systems qualify as platforms, as they not only collect evidences of censorship, but also analyze and publish the results (*Herdict, Alkasir, YouTomb, Greatfire.org, Weiboscope*) or perform automatic target collection or selection (*YouTomb, Greatfire.org, Weiboscope*). We highlight that in the latter case all three platforms detect *server-based censorship*.

6.2 Probe

An element that strongly characterizes a censorship detection platform is the nature of the employed *probes*. With the notable exception of the tool *Spookyscan* (Ensafi et al. [51]), all considered tools and platforms employ a client-based *viewpoint*, i.e. traffic data is collected from the client itself, and

[27]For Herdict Web 1.4.1 as of June 2014 the licensing terms explicitly forbid creation of derivative works and commercial use: https://addons.mozilla.org/bn-BD/firefox/addon/herdict-web/eula/ .

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*, http://dx.doi.org/10.1016/j.comnet.2015.03.008

TABLE III: Detection Platforms and Tools: characterization

	ConceptDoppler [33]	rTurtle [63]	Herdict	Alkasir$^{\diamond}$	YouTomb	MOR [12]	Greatfire.org	CensMon [136]	OONI [57]	Weiboscope [61]	Samizdat [114]	UBICA [4]	WCMT [52]	Spookyscan [51]	encore
Year of publishing	2007	2009$^{\diamond}$	2009	2009	2010	2010	2011	2011	2012	2013	2013	2013	2013	2014	2014
Implementation															
available for use	-	*	X	X	X	-	X	-	X	X	X	-	-	X	X
Complexity															
tool	-	X	-	-	-	-	-	-	-	-	X	-	-	X	X
platform	X	-	X	X	X	X	X	X	X	X	-	X	X	-	-
Probe type															
user PC exec.	-	X	-	X	-	-	-	-	X	-	X	X	X	X	-
web app.	-	-	(h1)	-	-	-	-	-	-	-	-	-	-	-	(e)
dedic. server	(px)	-	-	-	X	(tp)	X	(pl)	-	?	-	(pl)	X	(a)	-
other	-	-	-	-	-	-	-	-	-	?	-	(bm)	-	(a)	-
Probe platforms															
Windows	?	X	X	X	-	-	-	-	(v)	-	-	X	-	-	X
Linux	?	-	X	-	X	?	◇	X	X	-	X	X	X	X	X
Mac OS	?	-	X	-	-	-	-	-	(v)	-	-	X	-	-	X
Other	?	-	(p)	-	-	-	-	-	-	-	-	(bm)	-	-	(p)
Probe license															
open source	-	-	-	-	X	-	-	-	X	-	X	-	?	X	X
proprietary	?	?	(h2)	X	-	-	-	-	-	-	-	X	-	-	-
not released	-	-	-	-	-	X	X	X	-	X	-	-	-	-	-
Control															
manual	-	X	X	X	-	-	-	-	X	-	X	-	-	X	-
centralized	X	-	-	-	X	X	X	X	-	?	-	X	X	-	(t)
continuity	X	(p)	-	X	X	-	X	X	-	X	-	X	-	-	X
closed loop	-	-	-	(p)	-	-	X	(p)	-	-	-	X	-	-	-
Target selection															
manual	-	X	-	-	-	?	-	-	X	-	X	-	X	X	-
crowdsourced	-	-	X	X	-	-	X	X	-	-	-	X	-	-	-
scraping	-	-	-	-	-	-	X	X	-	-	-	X	-	-	X
complex	X	-	-	-	X	-	-	X	-	X	-	-	-	-	-
Test variety															
single	-	-	X	-	X	-	-	-	-	X	-	-	-	X	X
few, related	X	X	-	◇	-	X	-	-	-	-	-	-	X	-	-
diverse	-	-	-	-	-	-	X	X	X	-	X	X	-	-	-
Helper servers	-	-	-	?	-	X	-	X	X	-	-	(p)	-	-	-
Ground Truth Probes	-	X	-	?	-	-	X	-	(tp)	-	-	-	-	-	-
Analysis															
local	X	-	X	-	X	-	-	-	X	X	X	-	-	X	-
global	-	X	-	◇	-	X	X	X	-	-	-	X	X	-	?
Reporting															
public	-	-	X	X	X	-	X	-	-	X	-	X	-	-	-
Report aggregation															
target	-	-	X	X	-	-	X	-	-	-	-	X	-	-	-
category	-	-	X	X		-	X	-	-	-	-	-	-	-	-
test	-	-	(s)	(s)	(s)	-	-	-	-	-	-	-	-	-	-
country	-	-	X	X	-	-	(s)	-	-	(w)	-	X	-	-	-
global	-	-	X	X	-	-	(s)	-	-	(s)	-	X	-	-	-

X yes; - no; ? undisclosed, unclear; * distributed by researchers to volunteers;
◇ indirect inference; (a) peculiar deployment, see [51] and specific section; (bm) leverages the *BISMark* platform [140];
(e) a JavaScript code is downloaded while visiting 3rd-party websites; (h1) besides the web application, an add-on for the FireFox browser is provided; (h2) the browser add-on code can be inspected but its usage is restricted by a non-open-source license[27]; (p) potentially or partially; (pl) leverages the PlanetLab platform [28];
(px) from outside censornet leverages web proxies in censornet as probes; (s) one single option available;
(t) only target selection, while probe activation and location is not controlled; (tp) leverages the *Tor* overlay network [46];
(v) by means of virtualization and automated setup framework, instructions provided;
(w) only China is addressed, geolocation of message publisher shown with Province granularity;

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

with no exception perform *active* collection (see Section 4.) This holds true for the tool *encore*, that collects reachability results as seen by the browser that runs the measurement JavaScript code, and for the platforms that detect *server-based censorship* (*YouTomb,Greatfire.org, Weiboscope*) that actively request the *target* resources on the monitored websites.

One main difference is between probes deployed on common users workstations and ones deployed on dedicated servers. In the first case potentially a high number of probes with high geographic and administrative diversity can be employed; in the second case the platform can leverage full control of the probe, but requires access to servers in the networks that are under monitoring.

Another distinction is related to the implementation of the probe application, for which different programming languages can be used, supporting different development platforms. For several tools (namely *OONI, Samizdat, rTurtle, Spookyscan*) the Python scripting language has been employed: being an interpreted language with implementations of the interpreter available for different platforms, in principle all of them could be supported. The same can be said if C or C++ sources of the tool are available, as compilers for these languages are widespread. As porting to different systems can be a non-trivial task even if theoretically feasible, we have adopted a conservative definition of "supported platform" considering as such only platforms for which either a package is provided or specific instructions for the installation are given. One case of wide support is when the probe is a web application, thus it is run through a web browser. Restrictions to this case apply when JavaScript is required, or even more if the probe is an add-on (browser-specific).

6.3 Control

The control paradigm for censorship detection systems can vary: for stand-alone tools the direct user intervention is the only control, while for distributed platforms that leverage multiple probes either a centralized or a distributed approach is possible.

For systems leveraging crowdsourcing for the collection of evidences, the control is delegated to the single users, that manually decide the time and the targets to be tested. Other censorship detection phases such as the collection and selection of targets or the reporting of collected evidences can be performed by interacting with a centralized service.

A possibility shared among stand-alone tools and distributed platforms is the automatic repetition of the evidence collection phase: even for tools manually operated by users an automatic repetition of the probing would allow to collect evidences on a given time span with no need of human intervention.

A property related to the control paradigm is the automatic use of results from previous analyses to inform the scheduling of new evidence collection rounds. This "closed loop" control paradigm can be applied to minimize the probing impact over the network, raise the frequency of probing for recently changed conditions, deepen the complexity of analysis, all based on the detected censorship techniques and status. A model for this approach in the field of distributed monitoring platforms can be found in *autonomic computing* (see Huebscher and McCann [75] for a specifically dedicated survey).

6.4 Deployment

Detection techniques (and thus the systems that adopt them) are characterized by the ability of generating purposely crafted network traffic (*active* methods, opposed to *passive* ones). The *viewpoint*, i.e. the role of the probe that collects the evidences, in its own turn heavily affects the applicable detection methods and the censoring techniques that can be revealed. We refer to Section 4 for the related introduction and discussion. We also note that all considered detection systems adopt *active* collection methods and *client*-side viewpoint, therefore this property has not been reported in the comparison of Table III.

6.5 Targets selection

Censorship detection platform and tools, when checking for evidences of censorship, face the problem of considering a processable number of *targets* among the potential billions reachable on the Internet at the time of checking.[28] We refer to this as the "needle in a haystack" problem.

This is specifically evident for *active* evidence collection methods, that have to individually check each *target*, for each probe, for each test, for several times in each time interval. Similar difficulties are present when the detection algorithm checks for keyword-based censorship technique: the selection of the words to be used as *trigger* among all possible words is also a "needle in a haystack" problem. It constitutes an issue also for *passive* evidence collection because, in order to extract specific traffic of interest from the whole of inspected traffic, filtering rules must be used that depend on the considered *targets*.

A simple solution to this problem is manual harvesting of *target* lists. The criteria to select specific *targets* to be checked for censorship can vary depending on the interest of user of the detection system. The more general case is for researchers that are exploring the phenomenon of Internet Censorship at large; on the other end of the spectrum we can envision the publisher of a specific content or online service that wants to verify its accessibility from different countries.

A minority of considered platform and tools allow the user to input a specific *target* to check (in the form of an URL).

Some platforms allow crowdsourcing of *targets* (limited to web resources, identified by their URL), i.e. prompt the users to input the URL of the resource and collect and make available the full list for others to check. This is an effective method to collect *targets* that are both of interest for the

[28] Leveraging results count provided by search engines and statistical properties of words in text corpora, an estimation of indexed web pages is provided as close to 2 billions (June 2014) .http://worldwidewbsize.com .

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

users and potentially censored of recent. One example of such approach is provided by the Herdict Project [70].

In our characterization, the difference between *manual* and *crowdsourced* collection methods are that in the first case the subsequent phases will involve only the user, while in the second case the *target* will be shared with other users and potentially tested system-wide.

Another approach found in literature about censorship and censorship *detection* leverages general purpose search engines to obtain updated lists of URLs related with sensitive topics, selected on the basis of search keywords. The choice of the keywords can be guided by common sense of the tool/platform designers, informed by former experimental studies on censored contents for each investigated country. Similar criteria and processes are involved in retrieving lists of URLs or IP addresses provided by specialized online directory services, e.g., listing open HTTP proxies, open DNS servers, Tor nodes, VPN providers, etc.

We characterize as "scraping" the *targets* collection methods that are performed by accessing such online lists and documents, potentially involving some format conversion and parsing.

Other approaches, that we collectively refer to as "complex", adopt multi-step processes besides simple scraping of third party websites and services (e.g., they first collect potential keywords or topics, then use them in general search engines to find URLs related to them), or adopt some specific algorithm or process to build *target* lists, as described in the following.

One of the first solutions to the "needle in the haystack" problem is presented in [136] (see Section 4.3.3), where periodic scraping of information from a number of sources is performed in order to extract the list of *targets* to be considered. The authors do not dwell on this aspect, and simply describe the information used to gather *inputs* for the detection algorithm; a brief description of such sources is reported in the following.

User Input: collection of URLs provided by the users of the system through the user interface

OpenNet Initiative's Herdict [70]: periodical scraping of the online report of tested URLs.

Google Alerts[29]on selected topics: subscribing to the topics Internet Censorship, Net Neutrality, freedom of speech and human rights, a daily report by email on the relevant search results on the topics is fetched by IMAP client, processed and fed to the system.

Internet Trends: Google Hot Trends [30] and Twitter [31] are scraped for keywords; these are used to query Google and take the top-10 results for each trend. No details are given in [136] on how the keywords are inferred from the trends, we speculate that, besides the trivial tokenization of returned text, techniques for keyword extraction like Latent Semantic Analysis [33] can be applied.

Search of ONI categories: the list of categories that the Open Net Initiative [79] has defined for censorship targets is considered to extract 10 keywords (news outlets, freedom of speech, entertainment, government, terrorism, porn, gambling, religion, net neutrality and human rights); for each the top-100 results from the Google search engine are considered.

An implementation of extraction of data from web sources has been made available by the *YouTomb* project (see Section 5.4), where python scripts are used to extract URLs of YouTube videos from news aggregation platforms. More in general, the automation of the activity of *target* collection can benefit from the literature on extraction of web data (see [55] for a recent survey of techniques).

A formal approach for keywords extraction has been adopted in [33], where the detection method is based on *trigger* of type *keyword*. The authors of [33] propose the application of *Latent Semantic Analysis* (Landauer et al. [97]) to a corpus of documents in the language of the censored *targets* (in the proof of concept the Chinese section of Wikipedia, as the censoring system under analysis is the Great Firewall of China). By means of this technique, that in turn leverages *Singular Value Decomposition* (Klema and Laub [88]), corpus documents and keywords are mapped in a lower dimensionality "concept space", allowing to relate keywords with similar occurrence properties together in "concepts". Starting from known censored keywords these can be mapped to their related concepts and then lead to new potentially censored keywords. Similarly, starting directly from sensitive topics, the related concepts map to a set of keywords to be tested for censorship. While this approach has been adopted in [33] to collect a list of *keywords* for HTTP *keyword*-based censorship, it can as well be used to generate queries to retrieve URLs lists from web search engines.

In order to perform detection of DNS tampering the commonly used approaches require the presence of a probe inside the censored network, but in case the censoring device acts as an *open resolver* direct probing is possible from outside the censored network. Thus another form of "needle in a haystack" problem arises: finding *open resolvers* in the whole Internet (or in a given country / ISP / AS), in order to probe it. This has been addressed in [38] using an active methodology that leverages the control of a name server that is authoritative for a given zone: queries are sent to the whole publicly routable IPv4 space of addresses[32], asking for resolution of a purposely crafted subdomain of the controlled zone. By logging all the requests coming to the controlled authoritative server, the existence and specific type of *open resolver* can be inferred, along with its IP address as encoded in the

[29]http://www.google.com/alerts

[30]https://www.google.com/trends/hottrends

[31]www.twitter.com

[32]From the experimental survey have been excluded special-use addresses [76] (now obsoleted by [32]), as well as other "bogons" address intervals published by Team Cymru [44] and address blocks allocated to the military and government of U.S.A. (as advertised through BGP).

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

specifically crafted query (Fig. 18). The reason behind the use of a controlled Name Server is that it allows to distinguish between *open forwarders* sub-type (case (2) in Fig. 18) and the other sub-type, *open recursive*, likely a misconfigured and harmless Name Server. The subset of *open forwarders* is then considered as possibly related to malicious activities (phishing or malware-related in [38], or censorship as well) and become the destination of direct active probing.

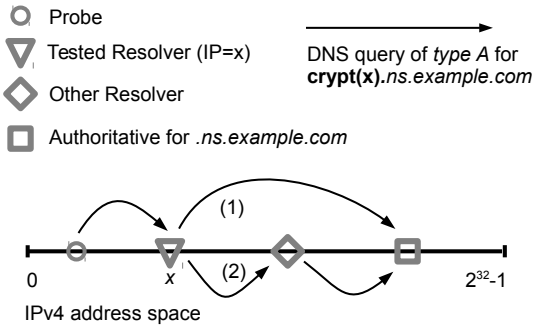

Fig. 18: Finding and characterizing Open Resolvers. The horizontal line represents IPv4 address space, the controlled authoritative resolver collects the queries; probed host with IP *x* is detected as (1) *open recursive* or (2) *recursive forwarding*. Figure inspired by [38, fig. 1(b)].

Keyword lists can be obtained by reverse-engineering the binaries of applications suffering client-based censorship such as *TOM-Skype* (Villeneuve [148], Knockel et al. [91]) and *SinaCU* (Aase et al. [2]). Most interesting, in the analyzed cases the lists were updated through the network, and in [2] the authors report to have extracted both the URL for the update and the decryption key, becoming able to track the changes in the blacklists over time.

6.6 Evidence collection and Analysis

The core functionality of detection systems is the collection of evidences of censorship by means of measurements and subsequent analysis. These two tasks are strictly connected and in some systems due to the simple analysis process can be considered as a single phase. Nevertheless, besides being tasks conceptually separated, there are detection systems that perform them at different times. The different measurements and tests aimed at collecting evidence for censorship detection have been extensively addressed in the Section 4 and summarized in Table I; the categories of evidences collected by the different platforms and tools is reported in Table II.

Besides the type of tests performed to collect evidences, another aspect characterizing a detection system is its specificity in addressing a type of censorship. According to this we separate detection tools in three groups: specialized systems performing one single test, intermediate systems performing a small number of strictly related tests, and generalist detection systems, performing a broad collection of evidence tests. From Table III we can see that some systems can perform multiple complex tests, and platforms can have single accessibility test; thus the complexity of the detection system is not directly related to the complexity of tests and analyses performed.

Another aspect of evidence collection activities is the usage of *helper servers*, i.e. the adoption of a controlled *target* whose incoming network traffic is collected or logged. These can be used for tomography setups.

Some of the platforms or tools are equipped with special probes used as a ground truth or oracle, by hypothesizing that it is not subject to the censorship technique that is under detection. Thus the results from the ground truth probe and the ones from the fields are compared, inferring censorship when they differ.

Finally the analysis of results can be performed on a local basis or a global one. For detection systems that operate a local analysis, the results from the probe are analyzed for known anomalies typical of censorship, or checked for consistency or compared with a ground truth that is accessed from the same probe collecting the evidences (a common mean is the Tor overlay network). All these cases imply operations limited to a single probe, not requiring the support of other components, and are thus considered "local analysis". The other possibility is to collect centrally the results from multiple probes, and analyze them together, leveraging a multi-probe view of the phenomenon to infer the presence and type of censorship.

6.7 Reporting

The results of the analysis of collected evidences are reported by the considered detection systems in different forms. Virtually all of them are able to produce the results of analysis for the single probe and the single detection test; the differences are in the public exposition of such results and their level of aggregation.

If the reports are publicly accessible, and thus available to users regardless of their participation in the measurement process, the reporting is flagged as "public" in Table III. The aggregation level of reports is divided as: "target", if platform-wide results are available for each tested *target*; "category", if *targets* are grouped in coarse thematic categories, such as *adult content, religion, news, circumvention, etc.*; "test", if results for each test is available, evidencing the detection of the specific censorship techniques; "country", if detection results are shown aggregated by country from which the tests have been performed.

7 Discussion and conclusions

In this survey we have collected and analyzed the literature on Internet Censorship, censorship circumvention and censorship detection in order to study the methods, techniques, architectures, platforms and tools available for censorship detection. We have provided an overview on Internet Censorship and related concepts, and we have proposed a characterization of

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

censoring systems and a reference description of the censoring techniques they adopt, as a basis for the subsequent discussion (Fig. 1). We have analyzed and discussed the techniques and architectures adopted and proposed in the considered literature, presenting a chronological summary bibliography (Table I), and we have researched and analyzed detection tools and platforms whose implementation is publicly available, both those described in academic literature and not. We have proposed a characterization of censorship *detection* systems, and described and discussed the considered systems in terms of such characterization (Table II and Table III).

Despite the relative youth of the surveyed topic, we have found that significant variability of deployment setups, detection techniques and accessory services have been adopted. In the following we present a discussion of the state of art represented by the considered detection systems.

Evolution over time: With reference to Table II and Table III we can notice no evident trend from the chronological sequence of considered properties: virtually all aspects present themselves again over time with no clear beginning or ending years that could have shown adoption or dismission of techniques and setups. One possible reason for such behavior is the limited time span resulting from the survey: while Internet Censorship appears as an academic research topic as early as year 2000 (see [71] for a general survey of academic literature on the Chinese Internet), the fields of study that addressed it were prominently social sciences; to find technical analyses detailed to the point of be considered as *detection* methodology or technique we have to wait until 2003 and 2006 (Dornseif [48], Clayton et al. [31], Clayton [30]), while for the availability of tools and platforms year 2009 must be reached. This limits the span of the time interval under analysis to just 6 years, testifying the relative youth of the topic.

The lack of evident trends in the censorship techniques that are detected can be explained considering that the censoring systems being monitored have evolved over time (e.g., *TCP disruption* in China has been found changing from 2006 to 2013 in several papers, adopting different detection and analysis methods). Moreover, despite the availability of new and more advanced censoring systems, the oldest censoring technique analyzed (DNS tampering) is still currently used and thus detected (Nabi [114]).

Results publishing: With the exception of *Weiboscope*, recently presented detection systems neglect the publishing phase, being centered mostly on novel measurement techniques and deployment setups. Moreover they focus on specific and limited evidence collection techniques: *encore* collects the success or failure outcome in rendering embedded components of web pages, but is unable to access the content of the tested resource, and no algorithm is proposed to infer censorship from such evidences; *spookyscan* due to the peculiar technique it adopts is able to collect evidence only of IP/port filtering.

The most informative reporting is provided by *Herdict*, that offers time-series graphs (with annotations of significant events such as Egypt disconnection on January 2011), lists of top-reported countries, URLs, categories of URLs, a world map with country-based totals shown as differently sized dots, and the possibility to download a CSV database of reports. Similarly *Greatfire.org* offers overall aggregated statistics for most popular worldwide websites[33] and a calendar report for each monitored *target*. The server-side censorship detection platforms *YouTomb* and *WeiboScope* both provide chronologically ordered lists of censored targets; *WeiboScope* offers also a color-coded map of China Provinces showing the location of censored authors.

For what concerns the reporting facilities *Herdict* can thus be considered the leading example, but due to the limited evidence collection procedure it does not allow to tell censorship from outages with a degree of confidence, nor is able to infer the censoring technique (that could help confirming intentional action and also suggest which actor is responsible of it, besides indicating possible collateral damage and possible circumvention techniques). In this direction *Greatfire.org* provides significant more information, but is specifically devoted to monitoring the Chinese censoring system; moreover, using a limited set of probes to perform evidence collection it could have limited visibility of censorship, that has been found to be applied in different parts of the networks (Xu et al. [159]) and differently across ISPs (Anderson [9]).

Detection features: The platform detecting the most diverse set of censoring techniques is *Greatfire.org*, including besides DNS hijacking and DNS injection and the common HTTP tampering check also the detection of server-side censorship. It is followed by *OONI* and *Samizdat*, both collecting evidence for a diverse set of censoring techniques; in particular *OONI* has been explicitly designed to easily add new evidence collection tests (with local analysis), thus the set of supported tests and detected techniques is expected to grow with the adoption of the platform by researchers.

All the analyzed detection systems have specific merits, specially the ones aimed at a narrow goal (*Spookiscan, encore, YouTomb, Weiboscope*) but even the ones with the broadest scope (*Herdict* and *OONI*) do not cover all the aspects of censorship detection, leaving open the need for a comprehensive solution. More specifically *OONI*, focused specially on the simplicity for the researchers to define and implement new evidence collection methods, lacks an analysis and publishing component, while *Herdict* partially solves target selection issues and extensively reports the results, but is very limited in the collection and analysis parts. Both lack detection of server-side censorship, that according to Bambauer [16] is a censoring approach expected to keep gaining importance. The platform *Greatfire.org* includes server-side censorship and does not depend on users terminals for probing, but is dedicated to one specific censoring system (the Great Firewall of China) and has issues related to the limited number of probes.

[33]The popularity ranking is provided by the Alexa online service [7].

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

Target collection: The target collection task has been explored in the literature with different approaches, but the implemented platforms rely mostly on crowdsourcing (*Herdict, Alkasir, Greatfire.org*) or on extraction of information from search engines and other web services, with a special case represented by monitors of server-side censorship (*YouTomb, Greatfire.org, Weiboscope*). The *Herdict* platform has become in turn a source of targets for other detection systems such as *encore, Greatfire.org* and is often one of the considered sources also in the architectures proposed in literature. Notably, crowdsourcing of *targets* collection has been applied only to web resources, and not to Internet applications in general. This can be partly because the Web and applications leveraging the HTTP protocol likely are the most commonly censored ones, but also because none of the considered detection systems supports checking a generic Internet application. (The closest to this goal is *OONI*, that has been designed as a framework for defining censorship detection tests.) Therefore it is expected that no facility is available for the users to submit a more general type of *targets*.

Other sources include blacklists either leaked from internal communications of involved entities (e.g., *Samizdat*) or extracted by client-based censorship enforcement tools. An aspect that has been briefly introduced in [136] but not deeply investigated nor implemented in available tools, is the *closed loop control* i.e. the automatic use of the results of analysis to update the system behavior in order to improve the coverage, the reliability or the granularity of censorship detection.

Active users participation: Besides *Greatfire.org* and the other platforms that detect server-side censorship, all considered tools and platforms rely on user engagement, either by repeatedly visiting a website, or installing and running an application. The necessity to attract and keep users is mitigated in the *encore* tool by inserting the probing code in third parties web pages, that the user may be motivated to access for her own reasons. Similar approach has been adopted in [9], where some measurements were performed automatically by users running the file sharing application *µTorrent*. An alternative solution can be derived by the *Alkasir* platform, that is mainly aimed at censorship circumvention, and provides detection as a secondary service (enabling circumvention only for selected targets actually found censored). From an abstract point of view, all induce the users to participate in the detection process as part of something else the users desire, but ethical issues are worsened, as the users are not aware of their involvement (this is not the case for *Alkasir*, that explicitly engages the users in the detection).

7.1 Challenges and final remarks

A number of issues arise when considering the monitoring of Internet Censorship, regarding both the complex nature of the phenomenon in itself, and the technical implications in its monitoring. We discuss the challenges and propose possible solutions and research directions in the following, distinguishing issues related to *analysis* from the ones related to *measurement*, further splitting the last in aspects common to wide-scale measurement systems, and the ones specific to censorship monitoring.

7.1.1 Challenges of censorship analysis: First of all **the phenomenon of Internet Censorship itself is hard to define**. We have addressed this issue by adopting a strictly technical approach, but it is evident that the motivations and intended goals behind the censoring activities have both ethical and practical consequences; e.g., denying access to child pornographic content with the aim of discouraging its production and thus the associated abuse is different from blocking information regarding political topics to prevent an organized response from a population. The global nature of the Internet and the variability across countries and in time of what is legal, tolerated, improper, harmful, make an in-depth and worldwide valid definition of Internet censorship potentially unfeasible.

Given that a complete definition of Internet Censorship is a still unsolved issue, and sticking to the purely technical definition that we have provided in this survey, we also argue that **there is a lack of significant metrics to quantify censorship**. A censorship detection system should be aimed at answering a number of questions, the most basic one can be worded as: *What is the extent of censorship—if any—that is enforced on a given set of users?* Metrics that quantify censorship extent in the surveyed detection systems do not go beyond the count of single *targets* detected as blocked, but this is hardly significant *per se* and is of limited practical use as index of intrusiveness of censorship in a given country, and even less for a comparison across different countries. We see this as an important open issue still in need of research. Possible directions could take into account the informative content of censored *targets*, their specificity in terms of topics, their relevance for political, cultural, health conditions of the affected population. Other possible metrics could consider the number of entities affected by censorship that belong to all affected parties: the producers of the censored content, the publishers, and the potential consumers.

Previous challenge is strictly related with the **complete lack of account for *overblocking*** in the surveyed detection systems. We have thoroughly discussed how each known censoring technique affects the user and interferes with intended functioning of the Internet, and how they are prone to side effects, such as the inaccessibility of legitimate *targets* ("overblocking"). None of the considered detection systems explicitly addresses or reports overblocking, despite it is an important aspect of censorship and is tightly linked with the measurement of censorship extent and implicit costs. A possible example in pursuing this extension of analysis can be the association of each *target* to a potential motivation for blocking it (e.g., on the basis of sensitive topics addressed in it); then, given the detected censoring technique, evaluate which other *targets* potentially present the same *trigger* or are affected by the same censoring *action*. This would provide the

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

overblocking in terms of other potential *targets* to test: actual overblocking can be measured considering among these *targets* the ones for which no apparent motivation for blocking can be found.

A fundamental aspect of Internet Censorship is the actual adoption of techniques available to enforce it, following their evolution in time, thus **in reporting censorship the specific censoring technique should be stated.** A detection system should be able to answer the question *How the censorship has been enforced?* Yet only one third of the surveyed systems performs a diverse set of tests necessary for inferring the actual censoring technique, and not all of these actually perform the inference on test results: we think that the available tools and platforms should both aim to support the broadest set of evidence collection tests, and leverage test results to infer the censoring techniques. On the other hand, novel detection techniques should be designed from the beginning with the objective of being integrated in a more general platform, otherwise resulting in limited usefulness.

By considering the literature on traffic classification and on circumvention, we have noticed that, to the best of our knowledge, **no detection system tries to elicit *triggers* of *behavioral* type**, i.e., based on statistical flow-level features, host-based connection graph features, or reaction to censor's active probing. As the technology to build this kind of *censoring systems* is known, and can be considered as the last step in the army race of Internet Censorship versus circumvention (see, e.g., [158]), we expect *detection* systems to keep pace with it. In order to reveal censoring systems that are triggered by such features, the mimicking of network protocol and overall application behavior is likely necessary, requiring *tomography* setups and testing procedures more complicated than the ones found in the survey, and less generalizable to different *target* services. This can be considered a clear challenge to *detection*, but also likely an unavoidable future step. Considering the inference of the censoring technique can be indirect, complex and error-prone, **an index of confidence in the resulting response should also be provided**. In fact telling intentional impairment from accidental performance problems or outages and telling content mangling from ordinary dynamic content and personalization are non-trivial tasks, unlikely to have sharp 100% or 0% certainty in the outcome. Despite this we have found very little attention to this aspect, as the few solutions provided adopt basic approaches, e.g., based on the percentage of access failures over a given testing time. Therefore we highlight this lack as a notable opportunity for improvement on the state of art.

Automatic time-based analysis with external events correlation and leveraging of news outlet for *targets* collection are two valuable features still in need of deployment in considered systems. While the set of censoring techniques documented in the surveyed papers did not significantly change over time, their adoption in specific countries or by specific ISPs, and the *targets* they were aimed at, did actually change. So another question to be answered by a detection system would be *When censorship of this* target *has begun, and how long it lasted?* Besides accounting for the dynamic nature of Internet Censorship, this kind of analysis would enable correlation with external events, either providing context (and possible motivation) for censorship, or ascribe the detected impairments to technical issues. While more than half of the considered detection systems are designed for continuous operation and can in principle perform time-based analysis, this has been performed manually, and related to external "real world" events as a subsequent validation of results.

Other less evident challenges lie in **ethical and practical consequences related to the detail of analysis.** All the analyses described so far, in order to be significant and useful, should be further broken down according to countries, geographical zone, adopted ISP, connection type (office, residential, academic, public). This requires collection of information not easily available to a third party, and could potentially result in a threat to privacy of users and to trade secrets of involved companies. Moreover, the explicit report about the censoring technique that has been detected informs the users on the circumvention techniques that can be successfully adopted. This on one hand is desirable, coherent with the idea that "security by obscurity" is a poor security paradigm, and reduces the advantage that technically skilled people have on the common citizen. On the other hand, in the case of—locally—lawful censorship, such reporting would be akin to suggesting how to elude laws, with the related ethical, legal and potentially practical consequences. The same act of performing active censorship detection tests could in principle be not legal in some countries or put in danger the user regardless of the official laws.

While not strictly technical, these aspects have been cited as significantly limiting the deployment of detection systems in countries where rule of law is not respected, and is an open issue for censorship detection. An analysis of challenges in studying *information controls* (of which Internet Censorship constitutes a subset) can be found in [34], where some background on the methodology adopted in the *OpenNet Initiative* project is provided and the related multidisciplinary approach is proposed to holistically investigate such complex phenomenons.

7.1.2 Challenges of censorship measurement: Due to their intrinsic nature, **censorship detection systems share challenges of wide-scale network measurement systems**. All the analyzed detection systems perform *active* measurements, i.e., they generate purposely crafted network traffic in order to collect information about the network behavior. Moreover such measurements are wide-scale, as they engage—in number of thousands and orders of magnitude more—globally distributed servers on the Internet. These characteristics qualify most of the considered detection platforms and architectures as wide-scale network measurement systems, sharing goals and challenges related to this nature, regardless of the specific network properties that are measured. Examples of wide-scale network measurement systems are *Trinocular* (Quan et al.

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

[124]) and *Hubble* (Katz-Bassett et al. [84]), both addressing Internet outages monitoring, *DIMES* (Shavitt and Shir [138]) and *MERLIN* (Marchetta et al. [106]), focused on Internet topology monitoring, and *HoBBIT*(de Donato et al. [42]), designed for network performance monitoring.

Common concerns include the choice of the probing frequency, the traffic load imposed on the network, the time-coherence among the globally distributed measurements, the coverage of networks or hosts necessary to obtain results representative of the rest of the Internet, the adaptation of such choices to the dynamic nature of the networks, and data management issues related to quantity and persistence of collected data. A number of challenges derive from such aspects, as many of them are mutually constrained by trade-off relationships, e.g., increasing the coverage of tested hosts/networks increases the imposed traffic load and affects negatively the time-coherence of global results; increasing the probing frequency enhances the time granularity but increases also the amount of measurements data to be collected, stored and processed. We refer to the aforementioned works and the related literature for a more in-depth analysis and examples of adopted solutions for such issues.

Besides the challenges in common with large-scale network measurement systems, **censorship detection systems face more specific ones, related to the adversarial environment** created by censors. These can be motivated in hiding the existence of censorship or prevent access to information functional to circumvent it, and thus may want to interfere with or impede censorship detection. As a consequence, censorship detection tools can be targeted so that information about the tool or access to tool online repositories is blocked. The different traffic flows involved in active censorship detection can also be blocked, impaired or tampered with. These include measurement-setup and results-reporting communications between probes and a management server / collector service. A host, recognized by the censor as a probe of a censorship detection system, could be intentionally handled differently from a common host, e.g., could be allowed to perform testing enough to conceal the existence of the censoring system. This way the probe would falsely report the absence of censorship. For crowdsourced systems there is the additional possibility that the censor itself runs a number of *rogue* probes in order to directly pollute the reports with false data. Authentication-based and reputation-based approaches in this case would contrast with the need to preserve user privacy. Special care is to be payed also to the security aspects involved in making users install and run software on their personal or work terminals. Making the non-technical-savvy user aware of the potential risks involved in such activities may prove not trivial. Crowdsourced systems also suffer by the necessity of enlisting and retaining a voluntary user-base, the bigger the better, thus too high a barrier to user enrollment is to be avoided.

Most of these issues are still open, although promising research can be found in the field of *participatory sensing systems*, in which the ubiquity of mobile personal devices capable of high amount of data collection and transmission is exploited to implement highly distributed data-based applications (see for example [26] for a survey on privacy-preserving approaches, and [27] for a proposed solution based on reputation and pseudonymity).

Finally, **operational aspects are in need of more attention** from the research in censorship detection. Both the aspects common to wide-scale network measurement systems and the ones specific to censorship detection have been addressed only at basic levels—if at all—in the surveyed literature, despite being evidently important. Even the simple question *"How many different probes are necessary to evaluate censorship of a given target?"* has not been formally answered nor investigated. We argue that the reason behind this lack of technical maturity from the operational point of view is due to the high emphasis on the detection results more than on the tools employed to get them, because of the practical, ethical and cultural consequences of such results and their limited availability in technical literature. We foresee and suggest that future implementations of the censorship detection systems will progressively address these issues explicitly and with approaches more systematic and scientifically sound, possibly leveraging and reinterpreting the lessons learned from established wide-scale network measurement systems and other related study fields also for the objective of Internet Censorship detection.

ACKNOWLEDGEMENTS

We are grateful to the Editor and the anonymous reviewers, whose comments helped us improving the quality of the paper. We also thank the organizers and attendees of the *Citizen Lab Summer Institute* events in 2013 and 2014, whose discussions necessarily contributed to our research.

This work is partially funded by the MIUR projects: PLATINO ($PON01\_01007$), SMART HEALTH ($PON04a2\_C$), SIRIO ($PON01\_02425$), and a Google Faculty Award 2013 for the UBICA research project.

REFERENCES

[1] Race to the bottom: Corporate complacency in Chinese Internet censorship. Technical report, The Human Rights Watch, August 2006. [Online; accessed June-2014].

[2] Nicholas Aase, Jedidiah R Crandall, Alvaro Diaz, Jeffrey Knockel, Jorge Ocana Molinero, Jared Saia, Dan Wallach, and Tao Zhu. Whiskey, Weed, and Wukan on the World Wide Web: On measuring censors' resources and motivations. In *FOCI'12: Second USENIX Workshop on Free and Open Communications on the Internet*, 2012.

[3] Chaabane Abdelberi, Terence Chen, Mathieu Cunche, Emiliano Decristofaro, Arik Friedman, Mohamed Ali Kaafar, et al. Censorship in the wild: Analyzing Internet

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

filtering in Syria. In *Internet Measurement Conference (IMC)*, 2014.

[4] Giuseppe Aceto, Nick Feamster, and Antonio Pescapè. User-side approach for censorship detection: home-router and client-based platform. In *Connaught Summer Institute on Monitoring Internet Openness and Rights*. University of Toronto, July 2013. URL http://wpage.unina.it/giuseppe.aceto/pub/aceto2013userside_poster.pdf.

[5] Giuseppe Aceto, Alessio Botta, Antonio Pescapè, Nick Feamster, Tahir Ahmad, and Saad Qaisar. Monitoring Internet Censorship with UBICA. In *Seventh International Workshop on Traffic Monitoring and Analysis (TMA'15) Barcelona, Spain*, April 2015.

[6] Bernhard Ager, Wolfgang Mühlbauer, Georgios Smaragdakis, and Steve Uhlig. Comparing DNS resolvers in the wild. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 15–21. ACM, 2010.

[7] The Alexa web information company. The top 500 sites on the web. http://www.alexa.com/topsites.

[8] Alkasir. Alkasir for mapping and circumventing cyber-censorship. https://alkasir.com, 2014. [Online; accessed 20-January-2014].

[9] Collin Anderson. Dimming the Internet: Detecting throttling as a mechanism of censorship in iran, June 2013. URL http://arxiv.org/abs/1306.4361.

[10] anonymous. The collateral damage of Internet censorship by DNS injection. *ACM SIGCOMM Computer Communication Review*, 42(3), 2012.

[11] Antirez. New TCP scan method. posted on the bugtraq mailing list on Dec 18th, 1998. http://seclists.org/bugtraq/1998/Dec/79.

[12] Demetris Antoniades, Evangelos P Markatos, and Constantine Dovrolis. MOR: monitoring and measurements through the onion router. In *Passive and Active Measurement*, pages 131–140. Springer, 2010.

[13] Simurgh Aryan, Homa Aryan, and J Alex Halderman. Internet Censorship in Iran: A first look. In *3rd Workshop on Free and Open Communications on the Internet*. USENIX, 2013.

[14] D. Atkins and R. Austein. Threat Analysis of the Domain Name System (DNS). RFC 3833 (Informational), August 2004. URL http://www.ietf.org/rfc/rfc3833.txt.

[15] Hitesh Ballani, Paul Francis, and Xinyang Zhang. A study of prefix hijacking and interception in the Internet. *ACM SIGCOMM Computer Communication Review*, 37 (4):265–276, 2007.

[16] D.E. Bambauer. Censorship v3.1. *Internet Computing, IEEE*, 17(3):26–33, May 2013. ISSN 1089-7801. doi: 10.1109/MIC.2013.23.

[17] Sam Burnett. Encore code repository. https://github.com/sburnett/encore.

[18] Laetitia Campher and Christiaan Bezuidenhout. An evaluation of existing measures aimed at restricting the use of the Internet as an avenue to initiate sexual activities with adolescents. *Child Abuse Research in South Africa*, 11(1):43–56, 2010.

[19] Richard Carlson. Developing the Web100-based Network Diagnostic Tool (NDT). In *Passive and Active Measurement Workshop (PAM)*. Citeseer, 2003.

[20] Rui Castro, Mark Coates, Gang Liang, Robert Nowak, and Bin Yu. Network tomography: Recent developments. *Statistical Science*, 19(3):499–517, Aug 2004.

[21] Rick Cattell. Scalable sql and nosql data stores. *ACM SIGMOD Record*, 39(4):12–27, 2011.

[22] MIT chapter of the Free Culture Foundation. Youtomb project source code, . http://youtomb.mit.edu:8661/trunk/.

[23] MIT chapter of the Free Culture Foundation. Youtomb project website, . http://youtomb.mit.edu/.

[24] China Digital Times. News website in chinese and english. http://chinadigitaltimes.net.

[25] Daegon Cho, Soodong Kim, and A. Acquisti. Empirical analysis of online anonymity and user behaviors: the impact of real name policy. In *System Science (HICSS), 2012 45th Hawaii International Conference on*, pages 3041–3050, Jan 2012. doi: 10.1109/HICSS.2012.241.

[26] Delphine Christin, Andreas Reinhardt, Salil S. Kanhere, and Matthias Hollick. A survey on privacy in mobile participatory sensing applications. *Journal of Systems and Software*, 84(11):1928 – 1946, 2011. ISSN 0164-1212. doi: http://dx.doi.org/10.1016/j.jss.2011.06.073. Mobile Applications: Status and Trends.

[27] Delphine Christin, Christian Rokopf, Matthias Hollick, Leonardo A. Martucci, and Salil S. Kanhere. Incognisense: An anonymity-preserving reputation framework for participatory sensing applications. *Pervasive and Mobile Computing*, 9(3):353 – 371, 2013. ISSN 1574-1192. Special Issue: Selected Papers from the 2012 IEEE International Conference on Pervasive Computing and Communications (PerCom 2012).

[28] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. Planetlab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12, 2003.

[29] Claudio Guarnieri. Detekt. https://resistsurveillance.org.

[30] Richard Clayton. Failures in a hybrid content blocking system. In *Privacy Enhancing Technologies*, pages 78–92. Springer, 2006.

[31] Richard Clayton, StevenJ Murdoch, and RobertN Watson. Ignoring the Great Firewall of China. In George Danezis and Philippe Golle, editors, *Privacy Enhancing Technologies*, volume 4258 of *Lecture Notes in Computer Science*, chapter 2, pages 20–35. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-68790-0. doi: 10.1007/11957454_2. URL

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

http://link.springer.com/chapter/10.1007/11957454_2.

[32] M. Cotton and L. Vegoda. Special Use IPv4 Addresses. RFC 5735 (Best Current Practice), January 2010. URL http://www.ietf.org/rfc/rfc5735.txt. Obsoleted by RFC 6890, updated by RFC 6598.

[33] Jedidiah R Crandall, Daniel Zinn, Michael Byrd, Earl T Barr, and Rich East. Conceptdoppler: a weather tracker for Internet censorship. In *ACM Conference on Computer and Communications Security*, pages 352–365, 2007.

[34] Masashi Crete-Nishihata, Ronald Deibert, and Adam Senft. Not by technical means alone: The multidisciplinary challenge of studying information controls. *Internet Computing, IEEE*, 17(3):34–41, 2013.

[35] Steve Crocker, David Dagon, Dan Kaminsky, DKH Danny McPherson, and Paul Vixie. Security and other technical concerns raised by the DNS filtering requirements in the PROTECT IP Bill. Technical report, 2011. http://shinkuro.com/PROTECT%20IP%20Technical%20Whitepaper%20Final.pdf.

[36] Cube. Time series data collection and analysis - code repository. http://square.github.io/cube/.

[37] David Dagon, Manos Antonakakis, Paul Vixie, Tatuya Jinmei, and Wenke Lee. Increased DNS forgery resistance through 0x20-bit encoding: security via leet queries. In *Proceedings of the 15th ACM conference on Computer and communications security*, pages 211–222. ACM, 2008.

[38] David Dagon, Niels Provos, Christopher P Lee, and Wenke Lee. Corrupted dns resolution paths: The rise of a malicious resolution authority. In *Network and Distributed System Security Symposium*, 2008.

[39] Alberto Dainotti, Claudio Squarcella, Emile Aben, Kimberly C Claffy, Marco Chiesa, Michele Russo, and Antonio Pescapè. Analysis of country-wide Internet outages caused by censorship. In *Proceedings of the 2011 ACM SIGCOMM Internet measurement conference*, pages 1–18. ACM, 2011.

[40] Alberto Dainotti, Antonio Pescape, and Kimberly C Claffy. Issues and future directions in traffic classification. *Network, IEEE*, 26(1):35–40, 2012.

[41] Jakub Dalek, Bennett Haselton, Helmi Noman, Adam Senft, Masashi Crete-Nishihata, Phillipa Gill, and Ronald J. Deibert. A Method for Identifying and Confirming the Use of URL Filtering Products for Censorship. In *Internet Measurement Conference*, Barcelona, Spain, 2013. ACM. URL http://conferences.sigcomm.org/imc/2013/papers/imc112s-dalekA.pdf.

[42] Walter de Donato, Alessio Botta, and Antonio Pescapè. HoBBIT: a platform for monitoring broadband performance from the user network. In *Sixth International Workshop on Traffic Monitoring and Analysis (TMA) London, UK*, April 2014.

[43] Ronald Deibert. *Access controlled: The shaping of power, rights, and rule in cyberspace*. MIT Press, 2010.

[44] Dave Deitrich. Bogons and bogon filtering. Las Vegas, NV, USA, January 2005. North American Network Operator's Group (NANOG).

[45] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008. URL http://www.ietf.org/rfc/rfc5246.txt. Updated by RFCs 5746, 5878, 6176.

[46] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004.

[47] Marcel Dischinger, Massimiliano Marcon, Saikat Guha, Krishna P Gummadi, Ratul Mahajan, and Stefan Saroiu. Glasnost: Enabling end users to detect traffic differentiation. In *Networked Systems Design and Implementation (NSDI)*, San Jose, CA, USA, April 2010. USENIX.

[48] Maximillian Dornseif. Government mandated blocking of foreign web content, 2003. URL http://arxiv.org/abs/cs/0404005.

[49] Constantine Dovrolis, Krishna Gummadi, Aleksandar Kuzmanovic, and Sascha D Meinrath. Measurement lab: Overview and an invitation to the research community. *ACM SIGCOMM Computer Communication Review*, 40(3):53–56, 2010.

[50] T. Elahi and I. Goldberg. CORDON: A taxonomy of Internet censorship resistance strategies. Technical report, Technical Report CACR 2012-33, University of Waterloo, 2012.

[51] Roya Ensafi, Jeffrey Knockel, Geoffrey Alexander, and Jedidiah R. Crandall. Detecting Intentional Packet Drops on the Internet via TCP/IP Side Channels. In *Passive and Active Measurement Conference*. Springer, 2014.

[52] Shadi Esnaashari, Ian Welch, and Brenda Chawner. WCMT: Web Censorship Monitoring Tool. In *Telecommunication Networks and Applications Conference (ATNAC), 2013 Australasian*, pages 183–188. IEEE, 2013.

[53] Alan Eustace. Better search in mainland China. *Google Inside Search - the official Google Search blog*, (May 31), 2012.

[54] Guangchao C. Feng and Steve Z. Guo. Tracing the route of China's Internet censorship: An empirical study. *Telematics and Informatics*, (0), 2012.

[55] Emilio Ferrara, Pasquale De Meo, Giacomo Fiumara, and Robert Baumgartner. Web data extraction, applications and techniques: A survey. *CoRR*, abs/1207.0246, 2012.

[56] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard), June 1999. URL http://www.ietf.org/rfc/rfc2616.txt. Updated by RFCs 2817, 5785, 6266, 6585.

[57] Arturo Filastò and Jacob Appelbaum. OONI: Open

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

Observatory of Network Interference. In *Proc. 2nd USENIX Workshop on Free and Open Communications on the Internet (FOCI 2012)*, August 2012.

[58] Fragroute. Tool website. http://www.monkey.org/~dugsong/fragroute/.

[59] Free Haven Project. Selected papers in anonymity. http://freehaven.net/anonbib/topic.html.

[60] King-wa Fu and Michael Chau. Reality check for the Chinese microblog space: A random sampling approach. *PLoS ONE*, 8(3):e58356, 03 2013. doi: 10.1371/journal.pone.0058356. URL http://dx.doi.org/10.1371%2Fjournal.pone.0058356.

[61] King-wa Fu, Chung-hong Chan, and Michael Chau. Assessing censorship on microblogs in China: Discriminatory keyword analysis and the real-name registration policy. *IEEE Internet Computing*, 17(3):42–50, 2013. ISSN 1089-7801. doi: http://doi.ieeecomputersociety.org/10.1109/MIC.2013.28.

[62] U.S.A Georgia Institute of Technology, GA. Encore main web page. https://encore.noise.gatech.edu/.

[63] Phillipa Gill, Masashi Crete-Nishihata, Jakub Dalek, Sharon Goldberg, Adam Senft, and Greg Wiseman. Characterizing censorship of web content worldwide. 2013.

[64] F. Gont and S. Bellovin. Defending against Sequence Number Attacks. RFC 6528 (Proposed Standard), February 2012. URL http://www.ietf.org/rfc/rfc6528.txt.

[65] Google Inc. The Certificate Transparency project. http://www.certificate-transparency.org.

[66] Greatfire.org. Online censorship in China - platform main page. https://en.greatfire.org.

[67] Greatfire.org News Blog. Google bows down to Chinese government on censorship. (January 4), 2013. https://en.greatfire.org/blog/2013/jan/google-bows-down-chinese-government-censorship.

[68] Justin Grimmer and Gary King. General purpose computer-assisted clustering and conceptualization. *Proceedings of the National Academy of Sciences*, 108(7):2643–2650, 2011.

[69] Mark Handley, Vern Paxson, and Christian Kreibich. Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics. In *USENIX Security Symposium*, pages 115–131, 2001.

[70] Herdict. Platform web interface. http://www.herdict.org. [Online; accessed 8-February-2014].

[71] David K Herold. Through the looking glass: twenty years of research into the chinese Internet. 2013. URL http://ssrn.com/abstract=2259045.

[72] Rahul Hiran, Niklas Carlsson, and Phillipa Gill. Characterizing large-scale routing anomalies: A case study of the China telecom incident. In *Passive and Active Measurement*, pages 229–238. Springer, 2013.

[73] Ralph Holz, Lothar Braun, Nils Kammenhuber, and Georg Carle. The SSL landscape: A thorough analysis of the x.509 PKI using active and passive measurements. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, pages 427–444, New York, NY, USA, 2011. ACM.

[74] 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*, pages 217–234. Springer, 2012.

[75] Markus C Huebscher and Julie A McCann. A survey of autonomic computing – degrees, models, and applications. *ACM Computing Surveys (CSUR)*, 40(3):7, 2008.

[76] IANA. Special-Use IPv4 Addresses. RFC 3330 (Informational), September 2002. URL http://www.ietf.org/rfc/rfc3330.txt. Obsoleted by RFC 5735.

[77] Google Inc. Transparency report – removal requests. http://www.google.com/transparencyreport/removals/government, .

[78] Google Inc. Transparency report – traffic, . http://www.google.com/transparencyreport/traffic.

[79] Open Net Initiative. Project website. https://opennet.net.

[80] Internet2. Network diagnostic tool project website. http://software.internet2.edu/ndt.

[81] Van Jacobson. Traceroute. ftp://ftp.ee.lbl.gov/traceroute.tar.gz, 1999.

[82] Journalism and Hong Kong University media studies center. Weiboscope data feed, . http://weiboscope.jmsc.hku.hk/datazip/.

[83] Journalism and Hong Kong University media studies center. Weiboscope platform main page, . http://weiboscope.jmsc.hku.hk/.

[84] Ethan Katz-Bassett, Harsha V Madhyastha, John P John, and Arvind Krishnamurthy. Studying black holes in the Internet with Hubble. In *Proceedings of the 5th USENIX conference on Networked Systems Design and Implementation*, pages 247–262. USENIX Association, 2008.

[85] Ken Keys, Young Hyun, Matthew Luckie, and Kim Claffy. Internet-scale ipv4 alias resolution with midar. *IEEE/ACM Trans. Netw.*, 21(2):383–399, April 2013. ISSN 1063-6692.

[86] Sheharbano Khattak, Mobin Javed, Philip D. Anderson, and Vern Paxson. Towards illuminating a censorship monitor's model to facilitate evasion. In *Presented as part of the 3rd USENIX Workshop on Free and Open Communications on the Internet*, Berkeley, CA, 2013. USENIX.

[87] Gary King, Jennifer Pan, and Margaret E. Roberts. How censorship in China allows government criticism but silences collective expression. *American Political Science Review*, 107:1–18, 2013.

[88] Virginia Klema and Alan J Laub. The singular value decomposition: Its computation and some applications.

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*, http://dx.doi.org/10.1016/j.comnet.2015.03.008

Automatic Control, IEEE Transactions on, 25(2):164–176, 1980.

[89] Jeffrey Knockel. SpookyScan source code. http://cs.unm.edu/~jeffk/spookyscan.

[90] Jeffrey Knockel, Roya Ensafi, and Jedidiah Crandall. SpookyScan web interface. http://spookyscan.cs.unm.edu/scans/censorship.

[91] Jeffrey Knockel, Jedidiah R Crandall, and Jared Saia. Three researchers, five conjectures: An empirical analysis of TOM-Skype censorship and surveillance. In *Proc. 1st USENIX Workshop on Free and Open Communications on the Internet (FOCI 2011)*, San Francisco, CA, USA, August 2011.

[92] Stefan Köpsell, Rolf Wendolsky, and Hannes Federrath. Revocable anonymity. In *Emerging Trends in Information and Communication Security*, pages 206–220. Springer, 2006.

[93] Jesse Kornblum. Identifying almost identical files using context triggered piecewise hashing. *Digital investigation*, 3:91–97, 2006.

[94] Christian Kreibich, Nicholas Weaver, Boris Nechaev, and Vern Paxson. Netalyzr: illuminating the edge network. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 246–259. ACM, 2010. doi: 10.1145/1879141.1879173. URL http://dx.doi.org/10.1145/1879141.1879173.

[95] Jim Kurose. Open issues and challenges in providing quality of service guarantees in high-speed networks. *ACM SIGCOMM Computer Communication Review*, 23 (1):6–15, 1993.

[96] ARC90 labs. Readability code repository, 2009. https://code.google.com/p/arc90labs-readability/downloads/list.

[97] Thomas K Landauer, Peter W Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284, 1998.

[98] Ben Laurie. Certificate transparency. *Queue*, 12(8): 10:10–10:19, August 2014. ISSN 1542-7730. doi: 10.1145/2668152.2668154. URL http://doi.acm.org/10.1145/2668152.2668154.

[99] Christopher S Leberknight, Mung Chiang, and Felix Ming Fai Wong. A taxonomy of censors and anti-censors: Part I – impacts of Internet Censorship. *International Journal of E-Politics (IJEP)*, 3(2):52–64, 2012.

[100] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones. SOCKS Protocol Version 5. RFC 1928 (Proposed Standard), March 1996. URL http://www.ietf.org/rfc/rfc1928.txt.

[101] John Leyden. Inside 'Operation Black Tulip': DigiNotar hack analysed. http://www.theregister.co.uk/2011/09/06/diginotar_audit_damning_fail/, 2011. [Online; accessed 21-February-2014].

[102] Bingdong Li, Esra Erdin, Mehmet Hadi Gunes, George Bebis, and Todd Shipley. An overview of anonymity technology usage. *Computer Communications*, 36(12): 1269–1283, 2013.

[103] Wilson Lian, Eric Rescorla, Hovav Shacham, and Stefan Savage. Measuring the practical impact of dnssec deployment. In *Proceedings of USENIX Security*, 2013.

[104] Matthew Luckie, Young Hyun, and Bradley Huffaker. Traceroute probe method and forward IP path inference. In *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, pages 311–324. ACM, 2008.

[105] Pietro Marchetta and Antonio Pescapè. DRAGO: Detecting, quantifying and locating hidden routers in traceroute IP paths. In *INFOCOM, 2013 Proceedings IEEE*, pages 3237–3242. IEEE, 2013.

[106] Pietro Marchetta, Pascal Merindol, Benoît Donnet, Antonio Pescapè, and Jean-Jacques Pansiot. Topology discovery at the router level: A new hybrid tool targeting isp networks. *IEEE Journal on Selected Areas in Communications*, 29(9):1776–1787, Oct 2011.

[107] Morgan Marquis-Boire. Iranian anti-censorship software 'Simurgh' circulated with malicious backdoor. Citizen Lab, May 2012. https://citizenlab.org/wp-content/uploads/2012/08/04-2012-iranianantcensorship.pdf.

[108] P. V. Mockapetris. Domain names - implementation and specification. RFC 1035 (Standard), November 1987. URL http://www.ietf.org/rfc/rfc1035.txt.

[109] P.V. Mockapetris. Domain names - concepts and facilities. RFC 1034 (INTERNET STANDARD), November 1987. URL http://www.ietf.org/rfc/rfc1034.txt. Updated by RFCs 1101, 1183, 1348, 1876, 1982, 2065, 2181, 2308, 2535, 4033, 4034, 4035, 4343, 4035, 4592, 5936.

[110] Milton L. Mueller. China and global Internet governance. *Access contested: security, identity, and resistance in Asian cyberspace, ed. Ronald J. Deibert, John Palfrey, Rafal Rohozinski, and Jonathan Zittrain*, pages 177–194, 2012.

[111] Jon Mumm. Localize or fail. open tok Blog, 2011. http://www.tokbox.com/blog/localize-or-fail/.

[112] Steven J Murdoch and Ross Anderson. Tools and technology of Internet filtering. *Access Denied: The Practice and Policy of Global Internet Filtering, ed. Ronald J. Deibert, John Palfrey, Rafal Rohozinski, and Jonathan Zittrain*, pages 57–72, 2008.

[113] Zubair Nabi. Samizdat code. https://github.com/ZubairNabi/Samizdat.

[114] Zubair Nabi. The anatomy of web censorship in pakistan. In *Presented as part of the 3rd USENIX Workshop on Free and Open Communications on the Internet*, Berkeley, CA, 2013. USENIX. URL https://www.usenix.org/anatomy-web-censorship-pakistan.

[115] The GNU Netcat project. Official homepage. http://netcat.sourceforge.net/.

[116] Jong Chun Park and Jedidiah R Crandall. Empirical study of a national-scale distributed intrusion detection system: Backbone-level filtering of html responses in

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*, http://dx.doi.org/10.1016/j.comnet.2015.03.008

China. In *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*, pages 315–326. IEEE, 2010.

[117] Mukaddim Pathan and Rajkumar Buyya. A taxonomy of CDNs. In *Content Delivery Networks*, pages 33–77. Springer, 2008.

[118] AutoProxy plugin for Firefox browser. Project website. https://autoproxy.org.

[119] Becker Polverini and William Pottenger. Using clustering to detect chinese censorware. In *Proceedings of the Seventh Annual Workshop on Cyber Security and Information Intelligence Research*, page 30. ACM, 2011.

[120] The Tor Project. The tor project, a Free Software overlay network providing private TCP connections to the Internet. https://www.torproject.org, . [Online; accessed 8-February-2014].

[121] The Tor Project. Tor metrics portal, . https://metrics.torproject.org.

[122] ProPakistani Pakistani Telecom and IT news. List of blocked urls. May 2010. http://propakistani.pk/wp-content/uploads/2010/05/blocked.html.

[123] Thomas H Ptacek and Timothy N Newsham. Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical report, DTIC Document, 1998.

[124] Lin Quan, John Heidemann, and Yuri Pradkin. Trinocular: understanding Internet reliability through adaptive probing. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pages 255–266. ACM, 2013.

[125] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Address Allocation for Private Internets. RFC 1918 (Best Current Practice), February 1996. URL http://www.ietf.org/rfc/rfc1918.txt. Updated by RFC 6761.

[126] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol 4 (BGP-4). RFC 4271 (Draft Standard), January 2006. URL http://www.ietf.org/rfc/rfc4271.txt. Updated by RFCs 6286, 6608, 6793.

[127] E. Rescorla and N. Modadugu. Datagram Transport Layer Security. RFC 4347 (Proposed Standard), April 2006. URL http://www.ietf.org/rfc/rfc4347.txt. Obsoleted by RFC 6347, updated by RFC 5746.

[128] F Baldi Risso, M Morandi, O Baldini, A Monclus, and P Lightweight. Payload-based traffic classification: An experimental evaluation. in proceeding of communications, 2008. icc'08. In *IEEE International Conference*, 2008.

[129] R. Rivest. The MD5 Message-Digest Algorithm. RFC 1321 (Informational), April 1992. URL http://www.ietf.org/rfc/rfc1321.txt. Updated by RFC 6151.

[130] David Robinson, Harlan Yu, and Anne An. Collateral Freedom: A snapshot of Chinese users circumventing censorship. Technical report, The Open Internet Tools Project, May 2013. [Online; accessed on 8 February 2014].

[131] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. Detecting and defending against third-party tracking on the web. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 12–12. USENIX Association, 2012.

[132] Keith W Ross and James F Kurose. *Computer Networking: A Top-Down Approach Featuring the Internet*. Addison-Wesley Longman Publishing Co., Inc., 1999.

[133] Antonio Ruiz-Martínez. A survey on solutions and main free tools for privacy enhancing web communications. *J. Network and Computer Applications*, 35(5):1473–1492, 2012.

[134] Scapy. Project website. http://www.secdev.org/projects/scapy.

[135] Jens Schomburg. Anonymity techniques - usability tests of major anonymity networks. In *Proceedings of PET-CON 2009.1*, pages 49–58, March 2009.

[136] A. Sfakianakis, E. Athanasopoulos, and S. Ioannidis. Censmon: A web censorship monitor. In *USENIX FOCI*, August 2011. URL https://www.usenix.org/legacy/events/foci11/tech/final_files/Sfakianakis.pdf.

[137] Umesh Shankar and Vern Paxson. Active mapping: Resisting nids evasion without altering traffic. In *Security and Privacy, 2003. Proceedings. 2003 Symposium on*, pages 44–61. IEEE, 2003.

[138] Yuval Shavitt and Eran Shir. DIMES: Let the Internet measure itself. *ACM SIGCOMM Computer Communication Review*, 35(5):71–74, 2005.

[139] Christopher Soghoian and Sid Stamm. Certified lies: Detecting and defeating government interception attacks against SSL (short paper). In *Financial Cryptography and Data Security*, pages 250–259. Springer, March 2011.

[140] Srikanth Sundaresan, Walter De Donato, Nick Feamster, Renata Teixeira, Sam Crawford, and Antonio Pescapè. Measuring home broadband performance. *Communications of the ACM*, 55(11):100–109, 2012.

[141] The Crossbear Team. Crossbear. https://github.com/crossbear/Crossbear.

[142] Telecomix. Project webpage. http://telecomix.org.

[143] The Berkman Center for Internet & Society. Chilling effects. https://www.chillingeffects.org.

[144] Adam D Thierer. Parental controls & online child protection: A survey of tools & methods. *Social Science Research Network*, July 2009.

[145] Alkasir tool. Policy for urls submission. https://alkasir.com/policy.

[146] The Tor Project. OONI: Open Observatory of Network Interference. https://ooni.torproject.org. [Online; accessed 16-December-2013].

[147] John-Paul Verkamp and Minaxi Gupta. Inferring Mechanics of Web Censorship Around the World. In *Free*

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

and Open Communications on the Internet, Bellevue, WA, USA, 2012. USENIX Association.

[148] Nart Villeneuve. Breaching trust: An analysis of surveillance and security practices on China's tom-skype platform. Technical report, Information Warfare Monitor / ONI Asia, October 2008.

[149] Paul Vixie. Refusing REFUSED. http://www.circleid.com/posts/20120111_refusing_refused_for_sopa_pipa/, 2012. [Online; accessed 20-January-2014].

[150] Claire Voeux and Julien Pain. Going online in cuba. Technical report, Reporters Without Borders, October 2006.

[151] Nicholas Weaver, Robin Sommer, and Vern Paxson. Detecting forged TCP Reset packets. In *Network and Distributed System Security (NDSS) Symposium*. Internet Society, 2009.

[152] Nicholas Weaver, Christian Kreibich, Boris Nechaev, and Vern Paxson. Implications of netalyzr's DNS measurements. In *Proceedings of the First Workshop on Securing and Trusting Internet Names (SATIN), Teddington, United Kingdom*, 2011.

[153] Nicholas Weaver, Christian Kreibich, and Vern Paxson. Redirecting dns for ads and profit. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI), San Francisco, CA, USA (August 2011)*, 2011.

[154] J. Weil, V. Kuarsingh, C. Donley, C. Liljenstolpe, and M. Azinger. IANA-Reserved IPv4 Prefix for Shared Address Space. RFC 6598 (Best Current Practice), April 2012. URL http://www.ietf.org/rfc/rfc6598.txt.

[155] Philipp Winter. Selected papers in censorship. http://www.cs.kau.se/philwint/censorbib/.

[156] Philipp Winter. Towards a censorship analyser for Tor. In *Presented as part of the 3rd USENIX Workshop on Free and Open Communications on the Internet*. USENIX, 2013.

[157] Philipp Winter and Stefan Lindskog. How the Great Firewall of China is blocking Tor. In *Proc. 2nd USENIX Workshop on Free and Open Communications on the Internet (FOCI 2012)*, August 2012.

[158] Philipp Winter, Tobias Pulls, and Juergen Fuss. Scramblesuit: A polymorphic network protocol to circumvent censorship. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2013)*. ACM, November 2013.

[159] Xueyang Xu, Z. Morley Mao, and J. Alex Halderman. Internet censorship in China: Where does the filtering occur? In *Passive and Active Measurement*, pages 133–142. Springer, 2011.

[160] Tao Zhu, David Phipps, Adam Pridgen, Jedidiah R. Crandall, and Dan S. Wallach. The velocity of censorship: High-Fidelity detection of microblog post deletions, March 2013. URL http://arxiv.org/abs/1303.0597.

Giuseppe Aceto. Giuseppe Aceto is a postdoc at the Department of Electric Engineering and Information Technologies of University of Napoli Federico II (Italy), where he received his M.S. Laurea degree in Telecommunication Engineering in 2008, defending a thesis about a unified platform for available bandwidth estimation in heterogeneous networks, and his Ph.D. degree in Telecommunication Engineering in 2014, defending a thesis on a platform for the detection of Internet Censorship.

His research interests are focused on networking, more specifically on network measurements and traffic analysis.

Giuseppe Aceto is coauthor of papers on international journals (ACM Performance Evaluation Review, Elsevier Journal of Network and Computer Applications) and international conferences (IEEE International Workshop on Measurements & Networking, IEEE INFOCOM 2010, IEEE Symposium on Computer and Communications).

Giuseppe Aceto is co-author of a patent on a traffic classification method and system. In 2010 he was awarded the best local paper award at IEEE ISCC 2010.

Antonio Pescapé. Antonio Pescapé [SM '09] received the M.S. Laurea Degree in Computer Engineering and the Ph.D. in Computer Engineering and Systems at University of Napoli Federico II, Napoli (Italy).

He is currently an Associate Professor at the Department of Electrical Engineering and Information Technology of the University of Napoli Federico II (Italy) where he teaches courses in Computer Networks, Computer Architectures, Programming, and Multimedia and he has also supervised and graduated more than 100 among BS, MS, and PhD students.

His research interests are in the networking field with focus on Internet Monitoring, Measurements and Management and on Network Security.

Antonio Pescapé has co-authored over 140 journal (IEEE ACM Transaction on Networking, Communications of the ACM, IEEE Communications Magazine, JSAC, IEEE Wireless Communications Magazine, IEEE Networks, etc.) and conference (SIGCOMM, Infocom, Conext, IMC, PAM, Globecom, ICC, etc.) publications and he is co-author of a patent. He has served and serves as workshops and conferences Chair (including IEEE ICC (NGN symposium)) and on more than 90

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008

technical program committees of IEEE and ACM conferences.

He serves as Editorial Board Member of Journal of Network and Computer Applications and has served as Editorial Board Member of IEEE Survey and Tutorials (2008-2011) and was guest editor for the special issue of Computer Networks on "Traffic classification and its applications to modern networks" and for the special issue of Journal of Future Generation Computer Systems on "Internet of Things and Cloud Services". For his research activities he has received several awards, comprising a Google Faculty Award, several best paper awards and two IRTF (Internet Research Task Force) ANRP (Applied Networking Research Prize).

Antonio Pescapé has served and serves as independent reviewer/evaluator of research and implementation projects and project proposals co-funded by the EU Commission, Sweden government, several Italian local governments, Italian Ministry for University and Research (MIUR) and Italian Ministry of Economic Development (MISE). Antonio Pescapé is a Senior Member of the IEEE.

Please cite this article as: *G. Aceto, A. Pescapè, Internet Censorship Detection: a Survey, Computer Networks (2015)*,
http://dx.doi.org/10.1016/j.comnet.2015.03.008