CARONTE: Detecting Location Leaks for Deanonymizing Tor Hidden Services

Srdjan Matic
Universita degli Studi di Milano
Milan, Italy
srdjan.matic@unimi.it

Platon Kotzias
IMDEA Software Institute
Madrid, Spain
platon.kotzias@imdea.org

Juan Caballero
IMDEA Software Institute
Madrid, Spain
juan.caballero@imdea.org

ABSTRACT

Anonymity networks such as Tor are a critical privacy-enabling technology. Tor's hidden services provide both client and server anonymity. They protect the location of the server hosting the service and provide encryption at every hop from a client to the hidden service. This paper presents CARONTE, a tool to automatically identify *location leaks* in hidden services, i.e., sensitive information in the content served by the hidden service or its configuration that discloses the server's IP address. Compared to prior techniques that deanonymize hidden services CARONTE implements a novel approach that does not rely on flaws on the Tor protocol and assumes an open-world, i.e., it does not require a short list of candidate servers known in advance. CARONTE visits the hidden service, extracts Internet endpoints and looks up unique strings from the hidden service's content, and examines the hidden service's certificate chain to extract candidate Internet endpoints where the hidden service could be hosted. Then, it validates those candidates by connecting to them. We apply CARONTE to 1,974 hidden services, fully recovering the IP address of 101 (5%) of them.

Categories and Subject Descriptors

C.2.0 [**Computer-communication networks**]: Security and protection; H.3.5 [**Online Information Services**]: Web-based services

General Terms

Security

Keywords

Tor hidden services; location leaks; deanonymization

1. INTRODUCTION

The increasing surveillance of communications have made anonymity networks a critical privacy-enabling technology. Tor [21] is arguably the most popular anonymity network. It provides both sender anonymity and recipient anonymity for *hidden services*. Hidden services protect the location (i.e., IP address) of the server hosting the hidden service. In addition, they further protect against network-level eavesdropping by providing encryption all the way from the client to the hidden service. This includes the communication between the last Tor relay and the hidden service, even when the application traffic is not encrypted.

Deanonymizing the location of a hidden service, i.e., recovering the IP address of the server hosting the hidden service, enables taking down the hidden service, seizing its content, and possibly identifying the owners. Prior work has proposed attacks to deanonymize Tor hidden services through flaws on the Tor protocol [17, 40] and clock-skew fingerprinting [36, 47]. Attacks on the Tor protocol are promptly fixed by the Tor project. For example, the attack by Øverlier and Syverson [40] was fixed by introducing guard nodes and the more recent attack by Biryukov et al. [17] has also been fixed [16]. Attacks leveraging clock-skew fingerprinting assume a *closed-world* where a short list of possible candidate servers is known and the fingerprinting validates which candidate is the hidden server. Deanonymization attacks have also been proposed for the equivalent of hidden services in I2P (called eepSites) [20]. These attacks also assume a closed-world, where the IP addresses of I2P peers are candidate servers for eepSites.

This paper studies the problem of *location leaks*, i.e., information in the content or configuration of a hidden service that gives away its location. Location leaks are introduced by the hidden service administrators and cannot be centrally fixed by the Tor project. Deanonymizing hidden services through location leaks does not require the attacker to be part of the anonymity network, but only to access the hidden services.

Such content and configuration leaks are a well-known problem for hidden services, but their extent is currently unknown. In fact, some notorious takedowns of hidden services by law enforcement have been linked to such leaks. For example, in July 2013, law enforcement identified the location of the Silk Road marketplace, where products such as cocaine, heroin, LSD, and counterfeit currencies were traded [19]. The FBI claimed in court that they located the server of the original Silk Road through a leak of its IP address when visiting the site [27]. The FBI story has been disputed [27, 34], but researchers still believe it is likely that the takedown was due to a leak in the server's configuration [27]. After the Silk Road takedown other similar hidden services took its place. In November 2014, an international law-enforcement operation codenamed *Onymous*, took down over 400 hidden services including the Silk Road 2.0, Cloud 9, and Hydra drug markets [28]. The deanonymization method used by law enforcement in Onymous remains unknown [42]. Unfortunately, the same location leaks that law enforcement may be using to deanonymize abusive hidden services can also be used by oppressive governments to deanonymize and censor hidden services of political activists.

CCS'15, October 12–16, 2015, Denver, Colorado, USA.

ACM 978-1-4503-3832-5/15/10..$15.00
DOI: http://dx.doi.org/10.1145/2810103.2813667.

In this paper we propose a novel approach to deanonymize hidden services using a subset of location leaks in an *open-world*, i.e., without previous knowledge of a set of candidate servers. Our approach includes two steps. First, we propose techniques to extract candidate Internet endpoints (i.e., domains and IP addresses that may correspond to the hidden server) from the content and configuration of a hidden service. Our techniques examine endpoints and unique identifiers in the content and the HTTPS certificates. This step allows moving from an open-world to a closed-world. Then we validate each candidate pair $\langle hidden\_service, Internet\_endpoint \rangle$, checking if the Internet endpoint corresponds to the Web server hosting the hidden service. Previous work that leverages leaks on a server's clock skew [36,47] or software version [20] assume a closed-world and use the leaks only for validation. In contrast, our approach leverages location leaks to obtain also the candidate servers.

We implement our approach in a tool called CARONTE, which takes as input the URL of a hidden service and tries to deanonymize it through location leaks. CARONTE could be used by law enforcement agencies to automate the currently manual process of deanonymizing abusive hidden services, and also by political activists running hidden services that risk being censored. CARONTE checks a hidden service for a subset of content and configuration errors that can lead to deanonymization. CARONTE has no false positives. If it recovers the hidden service's IP address, it proves the need to improve operational security. However, it only tests for a subset of location leaks whose detection can be automated and may fail to validate some leaks. Thus, it cannot guarantee the hidden service is free of location leaks.

To test CARONTE's effectiveness we have applied it to 1,974 live HTTP and HTTPS hidden services, of which CARONTE recovers the IP address of 101 (5%). Our results can be considered the first measurement study of location leaks in Tor hidden services. Since CARONTE only tests for some types of location leaks and may fail to validate some leaks its results are conservative, i.e., some services not deanonymized could still be vulnerable.

Our results also show that 21% of the deanonymized services are hosted on Tor relays. Hidden services on Tor relays can easily be deanonymized, even in the absence of location leaks, assuming a closed-world where relays' IP addresses are candidate locations for a hidden service. This result also highlights the importance of assuming an open-world, as 79% of hidden services CARONTE deanonymizes cannot be deanonymized under the closed-world assumption. CARONTE also identifies 9 hidden services that redirect their users to Internet sites through HTTP, negating the benefit of encryption in the last hop.

This work makes the following contributions:

- We propose a novel approach to deanonymize hidden services through location leaks. Our approach assumes an open-world, i.e., no prior knowledge on candidate servers. To move from an open-world to a closed-world we propose techniques to identify candidate servers from the content and configuration of a hidden service.
- We implement our approach into CARONTE, a tool that attempts to deanonymize the location of hidden services through location leaks.
- Using CARONTE we perform the first measurement study on the prevalence of location leaks in hidden services. CARONTE analyzes 1,974 input hidden services, recovering the IP address of 101 (5%). It also uncovers that 21% of the deanonymized hidden services are running on Tor relays.

The rest of this paper is structured as follows. Section 2 introduces hidden services, location leaks, and our approach. Section 3 details the approach and CARONTE's implementation. Section 4 presents the measurements. Section 5 discusses defenses against location leaks and Section 6 ethical considerations. Section 7 describes related work and Section 8 concludes.

2. OVERVIEW & PROBLEM DEFINITION

In this Section we first briefly describe hidden services (Section 2.1), then we detail the type of leaks that CARONTE looks for (Section 2.2), and finally we provide an overview of our approach (Section 2.3).

2.1 Hidden Services

The Tor network provides hidden services as a mechanism for users to anonymously offer services accessible by other users through Tor. Hidden services provide recipient anonymity, i.e., they hide the IP address of the server hosting the hidden service. Hidden services can offer different functionality, e.g., Web services and Bitcoin mining. This paper focuses on hidden Web services.

The creation of a hidden service picks a public key and a secret key and creates an onion identifier by doing a SHA1 hash of the public key truncated to 80 bits. The onion identifier is encoded in base32 producing a 16 character string, which is appended the suffix ".onion" to produce an *onion address*, e.g., `niazgxevgzlrbpvq.onion`[1]. Some hidden services generate large numbers of onion addresses until they find one containing a desired substring. For example, Facebook's official hidden service has onion address `facebookcorewwwi.onion`. For Web services, hidden services are advertised as *onion URLs*, where the DNS domain is replaced by an onion address, e.g., `http://niazgxevgzlrbpvq.onion/content.html`. The distribution of onion URLs happens out-of-band: there is no a central repository that lists all the hidden Web services available in a given moment and thus users can access only those for which they know an onion URL.

Rendezvous protocol. The hidden service chooses a set of 3 Tor relays as *introduction points* and creates *circuits* (i.e., encrypted tunnels) to each of them. The hidden service then produces a signed *descriptor* that lists the service's public key and its introduction points. The descriptor is published in a distributed hash table on the Tor relays using as index the onion identifier and time period.

To use a hidden service a Tor client establishes a circuit to a Tor relay randomly chosen as *rendezvous point*. Then, it retrieves the hidden service's descriptor using the onion identifier and current time. Next, it creates another circuit to one of the introduction points, and communicates the rendezvous point to the hidden service through it. The hidden service creates a circuit to the rendezvous point and communication between client and hidden service happens over their respective circuits to the rendezvous point.

With hidden services all hops between client and hidden service are encrypted, compared to visiting an Internet domain through Tor where communication between last Tor relay and the service is not encrypted, unless the application layer is encrypted (e.g., with HTTPS).

2.2 Location Leaks

Hidden services hide the location of the server hosting them, but they do not protect against administrators unintentionally leaking information in the content or configuration of their hidden services

[1]Fake onion address for illustration.

that may lead to deanonymizing them (*identity leaks*) or the IP address of the hidden server (*location leaks*). The Tor project describes the risks of content and configuration leaks [11] but it is not clear to what extent hidden services administrators are taking precautions against them and how prevalent they are.

Using a leak in the content or configuration of a hidden service to deanonymize their administrators or its location involves two steps. First, find some *candidate identity* (e.g., the owner of a phone number embedded in the hidden services content) or *candidate Internet endpoint* (e.g., an IP address or DNS domain in an error page). Then, *validate* that the candidate identity truly corresponds to the administrators or the candidate Internet endpoint to the IP address. In this work we focus exclusively on location leaks that provide candidate Internet endpoints for the IP address of the hidden service, by simply visiting the hidden service, without adding any nodes to the Tor network or compromising the hidden service. Identity leaks as well as leaks that an attacker can induce by exploiting the hidden service's code (e.g., SQL injections) are considered out of scope.

We do not focus on location leaks because they are more important, but rather because we know a way to automatically validate candidate Internet endpoints to confirm that they are hosting the hidden service. Validation of identity leaks is better suited for manual police work than for an automated tool. Of course, deanonymizing the location of a hidden service may be a first step towards deanonymizing its owner's identity, e.g., by checking server registration records or monitoring accesses to the server. Note that location leaks are a problem for any type of anonymous services. While we focus on Tor hidden services, our approach is independent of Tor and can be applied to deanonymize the equivalent of hidden services in other anonymity networks, e.g., I2P eepSites [9].

There exist many different types of location leaks that can lead to deanonymizing the hidden server's location. CARONTE automatically identifies 3 types of location leaks due to *endpoints* (i.e., IP addresses, domains) and *unique strings* (i.e., Google Analytics ID, Google AdSense ID, Bitcoin wallets, page titles) embedded in the content of the hidden service, and the *HTTPS certificate* of the hidden service. Obviously, many other types of location leaks may exist and thus CARONTE may not find all location leaks.

Validating location leaks. Our candidate selection techniques output candidate pairs ⟨ onion_address, Internet_endpoint ⟩, where the Internet endpoint is a DNS domain or IP address that may host the hidden server of the onion address. If the candidate pair contains an IP address, validation connects to the IP address through the Internet but requests the content of the hidden service, i.e., sets the HTTP Host header to the onion address. If the Web server delivers the hidden service's content, it confirms that the hidden service is hosted on the candidate IP address. Note that the Tor network is an overlay of the Internet. Thus, hidden Web services are hosted on a Web server with a public IP address. The (hidden) Web server will reply to a request from the Internet to its public IP address unless a firewall allows only connections from certain IP addresses. Even when a firewall is used, the firewall will often be configured to allow connections through Tor (i.e., from Tor relays), in which case we can run validation through Tor (or from a Tor relay).

When the candidate pair contains a DNS domain, validation works the same by first resolving the domain to an IP address. The main reason why an Internet domain may resolve to the public IP address of the hidden Web server is when the domain identifies an Internet service hosted on the same Web server as the hidden service. Thus, if the hidden service runs on its own Web server CARONTE will probably fail to validate a candidate domain, even if the candidate domain belongs to the hidden service owners. For example, CARONTE could identify from the content of the hidden service an Internet site owned by the hidden service's owners, but hosted on a separate Web server. While CARONTE cannot validate that location leak, network-level attackers could monitor connections to the Internet site in hopes that the hidden service owners relax protections when accessing it. For that reason, CARONTE still outputs the candidate domains it found even if they are not validated, so that the hidden service administrator can check them.

In summary, CARONTE produces no false positives. If it outputs a leak, the deanonymization has been validated. On the other hand, CARONTE may have false negatives due to unsupported classes of leaks and leaks it cannot validate. We discuss how to safely configure hidden services in Section 5.

Unintentional leaks. Hidden services often contain sensitive content and need to protect their location to avoid censorship or legal prosecution. However, not all hidden services are like that. Some hidden services advertise an Internet clone, i.e., an Internet site that provides the same content, and explicitly link to it, e.g., Facebook. Similarly, an Internet site may explicitly link to its hidden service clone, indicating its availability to privacy-concerned users. In Section 4 we describe 3 automatic checks that CARONTE performs to classify location leaks leading to deanonymizations as unintentional or not.

2.3 Approach Overview

Figure 1 summarizes our approach. It comprises three main steps: *exploration*, *candidate selection*, and *validation*. Exploration takes as input a set of initial onion URLs (Section 3.1), creates an extended set of onion URLs and visits them through HTTP and HTTPS to collect the content of the hidden service and its certificate chain (Section 3.2). All the information of the exploration is stored in a central database.

Candidate selection takes as input the information in the database and outputs a list of candidate pairs ⟨ onion_address, Internet_endpoint ⟩, where the Internet endpoint is a DNS domain or IP address considered a candidate to be hosting the hidden server of the onion address (Section 3.3). To select candidates, it examines the endpoints (i.e., URLs, email domains, IP addresses) and unique strings (i.e., Google Analytics ID, Google AdSense ID, Bitcoin wallets, page titles) in the onion pages collected from the hidden service, as well as its HTTPS certificate. It searches the unique strings on Internet search engines to find if they are embedded in the content of Internet sites and queries certificate stores (i.e., Sonar [6]) to find Internet sites using the same certificates as the hidden service.

Validation takes as input the candidate pairs and verifies if a candidate Internet endpoint indeed hosts the hidden service (Section 3.4). It visits the Internet endpoints to collect their content and certificates. Then, it compares pairs of HTTP responses: one from the candidate Internet endpoint and the other from the hidden service. If it finds a pair with similar content and served from a similar Web server it outputs a location leak.

3. APPROACH

3.1 Collecting Onion URLs

CARONTE takes as input a list of onion URLs corresponding to a hidden service to deanonymize. These may be provided by an administrator that wants to check its hidden service or come from external means. However, for evaluating CARONTE we need to collect onion URLs for a large number of hidden services.

Obtaining onion URLs is a challenging process because there is no centralized repository and valid onion addresses are difficult to

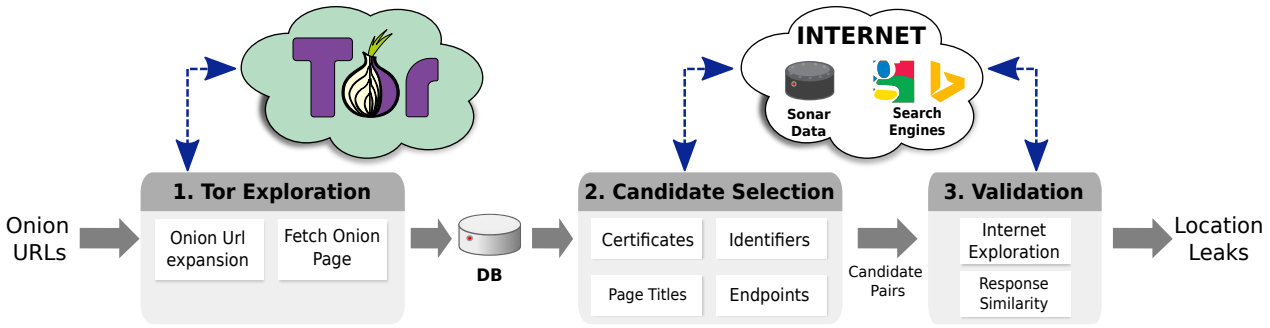

Figure 1: Approach overview.

predict being a SHA1 hash of the hidden service's key truncated to 80-bit. Owners of hidden services obtain their onion address during installation and are in charge of advertising their onion URLs. Given the nature of some hidden services, their onion URLs may not be publicly advertised, but only disclosed in private forums. In addition, many hidden services do not link to other hidden services, which limits crawling. Furthermore, onion addresses exhibit high churn, as the same hidden service may change onion addresses over time. In 2013, Biryukov et al. [17] presented a technique to list all onion addresses leveraging a flaw in Tor. However, that flaw was fixed after version 0.2.4.10-alpha [16].

As a consequence of these challenges, many sites exist that list onion URLs of hidden services with a description of their content such as "Deepweb links" [2] and "The Hidden Wiki" [4]. There are also specialized search engines for hidden services, typically accessible both from the Internet and through a hidden service [3, 10]. In addition, many onion addresses can be found by querying Web search engines such as Google and Bing.

We have developed scripts that periodically visit the above resources, scrape their links, perform searches on common terms, and feed the identified onion URLs to CARONTE. In addition, as CARONTE explores the hidden services, it adds any new onion URLs found in their content to the list, so that they are visited in the next round of exploration. Overall, CARONTE explores 15 K onion URLs on 6 K onion addresses. On February 2013, Biryukov et al. obtained a complete listing of hidden services and scanned them for open ports [16]. They found 3,741 HTTP hidden services listening on 80/tcp, while CARONTE finds 1,965. Assuming the number of HTTP hidden services has remained relatively constant since then, our sample would cover 52% of them, with popular hidden services being better represented.

3.2 Exploring Hidden Services

Given the initial list of onion URLs, CARONTE first creates an extended set of onion URLs that includes for each onion address: the root page, all resources for that onion address in the initial set, and one random resource. The random resource is added to trigger a not found error page, which may leak configuration information specific to the hidden server.

The exploration visits each onion URL in the extended set through Tor eight times: using HTTP and HTTPS, two Host header values (i.e., the onion address and a random onion address), and requesting the resource identified during the collection and a random one. The intuition behind using a random onion address is that the hidden service's Web server may also host other sites. If the hidden service is not the default site, the Web server may return the default site to the random onion address, and that site may leak some candidate Internet endpoints.

Method	Description
Endpoints	Extract IP addresses, Internet domains in email addresses, and Internet domains in URLs embedded in onion pages.
Identifiers	Use search engines to locate Internet domains embedding Google Analytics, Google AdSense, and Bitcoin wallet identifiers found in onion pages.
Titles	Use search engines to locate Internet domains with similar title as onion page.
Certificates	(1) Extract DNS domains and IP addresses from leaf certificate of hidden service. (2) Search leaf certificate of hidden service in Sonar [6] to obtain IP addresses where observed. (3) Search public key of hidden service certificate in Sonar to obtain IP addresses where observed. (4) Search Sonar for certificates on the Internet with onion addresses and the IP addresses where observed.

Table 1: Description of candidate endpoints.

For each request, the exploration collects the response, preprocesses it, and stores both request and response in a database. We limit CARONTE to download only textual and HTML responses (i.e., onion pages) to avoid storing copyrighted software and offensive multimedia content (e.g., explicit videos and pictures). This restriction means that CARONTE cannot identify leaks in other content types, e.g., EXIF data in images, although we do not expect they contain many candidate endpoints. For HTTPS, it also stores the certificate chain provided by the hidden server. The exploration is multi-process and runs from a single host; furthermore this step is rerun over time as new onion URLs are collected.

3.3 Identifying Candidate Endpoints

After the exploration finishes one round, the next step is to extract for each onion address a list of candidate Internet endpoints, i.e., DNS domains and IP addresses, which may point to the hidden server. For this, CARONTE examines the endpoints (Section 3.3.1) and unique strings (Section 3.3.2) contained in the *onion pages* collected through Tor, as well as the HTTPS certificate of the hidden server (Section 3.3.3). Table 1 summarizes the methods used to obtain candidate endpoints that are detailed in this section.

The input to this step is the information from the exploration stored in the database. This step does not require interacting with

the hidden service, but it interacts with some public Internet Web services. The output is a list of candidate pairs ⟨onion_address, Internet_endpoint⟩ where each pair indicates that the hidden service at the onion address could be hosted at the Internet endpoint. The same onion address may appear in multiple candidate pairs with different Internet endpoints.

3.3.1 Internet Endpoints in Onion Pages

Internet endpoints contained in the onion pages of a hidden service are good candidates for the hidden server. While we expect hidden service administrators to be careful about Internet URLs they link to, there exist several cases in which such leaks can happen, e.g., links or email addresses pointing to other sites of unrelated content but hosted on the same Web server, IP addresses and domains leaked in error pages, and endpoints in comments not visible and possibly forgotten.

For each onion page in the database, CARONTE applies regular expressions to extract URLs, email addresses, and IP addresses in the page. If an extracted URL contains an onion address, it is discarded since we are interested in Internet endpoints, but the onion URL is added to the database if previously unknown. If the URL contains a DNS domain, the domain is checked against the Alexa list of one million most popular domains. Alexa domains are discarded since very popular domains do not typically have a hidden service or publicly advertise it when they do (e.g., Facebook). Non-Alexa domains are added to the list of candidate Internet endpoints for the onion address of the page examined. For email addresses, if the email domain is in a list of popular email providers, it is discarded, otherwise it is a candidate.

IP addresses in the onion page are added as candidates, except when the onion page contains more than 5 IP addresses. In that case, all IP addresses in the page are discarded to prevent large directories of IP addresses unrelated to the hidden server (e.g., lists of Tor relay nodes) to be added as candidates.

3.3.2 Unique Strings in Onion Pages

A general technique to identify candidate Internet endpoints is to first extract some distinctive string from the content of the onion pages of the hidden service and then look up those strings in Internet search engines. Search engines will return Internet sites where they have observed those strings, and their DNS domains can be used as candidate Internet endpoints. This technique can be applied to any unique string that appears in the content of a hidden service. The more unique the string is, i.e., the less it appears in *unrelated* hidden and Internet services, the better the candidates produced. In any case, candidate pairs from non-unique strings are removed during validation. So far, we have added support for two classes of such unique strings: identifiers and page titles. We detail them below and leave as future work applying this technique to other classes of unique strings in the onion pages.

Identifiers. Onion pages may contain identifiers unique to the owners of the hidden service. CARONTE queries dedicated search engines to find Internet sites that also contain those identifiers. Such Internet sites likely belong to the owners of the hidden service and are good candidates for being hosted on the same Web server as the hidden server. Even when they are not, such leaks are dangerous, as those other Internet sites can be monitored by an adversary. If the owners of the hidden service connect to those other Internet sites they own (e.g., to configure them or change their content) without precautions (e.g., without a VPN) their identity could be revealed.

For each onion page, CARONTE applies regular expressions to extract Google Analytics and Google AdSense identifiers, as well as Bitcoin wallets. Google Analytics identifiers are unique to a Google Analytics account and are often embedded in pages to collect statistics. Google AdSense identifiers are less frequently included in pages but are unique for a publisher's account. Bitcoin wallets are often added to ask for donations. An onion page could add the Bitcoin wallet of some other user to encourage donations to that user's site and could even spoof the Google Analytics identifier of an unrelated account. This is not an issue, as unrelated candidates will be discarded during validation.

For each unique identifier extracted from at least one onion page, CARONTE queries dedicated identifier search engines that index Internet pages where identifiers have been observed (e.g., SameID [7]). DNS domains where the identifier has been observed are added as candidates for the onion addresses of the onion pages from where the identifier was extracted.

Titles. Page titles are often specific to the content of a page. Thus, they can also be used as distinctive strings to be looked up in Internet search engines. To reduce the number of queries to search engines, CARONTE first removes any title served by more than 10 hidden services, which removes generic titles such as those of default error pages. Each unique title is then queried on a search engine. DNS domains of the top 10 Internet sites where the title has been observed are added as candidates for the onion address of the onion page with that title.

3.3.3 HTTPS Certificates

CARONTE uses four methods to select candidate pairs from the collected leaf certificate of each HTTPS hidden service. The first method is to extract from the hidden service's certificate the Subject's Common Name (CN) and the Subject Alternative Name (SAN) extension. DNS domains and IP addresses in those fields are added as candidates for any onion address that provided that certificate. The intuition is that Web servers hosting multiple sites oftentimes reuse the same leaf certificate for all sites. While it is possible to configure separate certificate chains for each site this relies on the client sending the SNI header. Thus, hidden services' certificates may contain the domains or IP addresses of other Internet sites hosted on the same Web server.

Even if the certificate of a hidden service does not contain any Internet endpoints, a leak may happen if that certificate (or its public key) is also used by an Internet site on the same Web server. The other 3 methods leverage recently created certificate repositories that store information about certificates that have been observed in Internet traffic (passively, by crawling, or by scanning the Internet). These include Rapid7's Sonar project [6], ICSI's Certificate Notary [8], and Google's Certificate Transparency [1]. ICSI's notary does not provide the IP addresses that served a certificate and Certificate Transparency only allows querying for extended validation (EV) certificates. CARONTE uses Sonar, which contains 35 M unique certificates obtained by periodically scanning the Internet.

The second method computes the SHA1 hash of the DER format of the hidden service's certificate. Then, it uses the hash to search in Sonar whether this certificate has been served by any Internet site. If so, the IP addresses from where the certificate was served are added as candidates for the onion addresses from where CARONTE collected the certificate. Interestingly, some researchers have recently hypothesized that the Silk Road takedown could have been done by correlating the Silk Road's HTTPS certificate with an Internet HTTPS scan as we do [27].

The third method extracts the public key of the hidden service's certificate and searches in Sonar for certificates served by Internet sites that use the same public key. Public keys are left unchanged in many certificate replacements [39]. Thus, it could happen that

Sonar observed an older certificate with the same public key in the same IP address still hosting the hidden service.

The fourth method searches in Sonar for any certificate whose Subject's CN or SAN extension contains an onion address. Intuitively, a certificate that contains an onion address but is observed on the Internet likely corresponds to a Web server that hosts both hidden services and Internet sites. This is a wide search for any certificate with an onion address, even if the certificate was not seen during exploration. This method is unlikely to deanonymize a specific target hidden service, but is included for completeness of the measurement study. The onion address and each IP address that served the certificate form a candidate pair. If the onion address was previously unknown, it is queued for exploration.

3.4 Validation

Given the list of candidate pairs ⟨onion_address, Internet_endpoint⟩ validation first resolves the candidate domain endpoints to candidate IP addresses. For each candidate pair, it sends eight HTTP requests to the candidate IP address through the Internet. It fetches both the root page and a random resource using HTTP and HTTPS. In addition, it uses two different values of the Host header: the Internet endpoint (domain or IP) and the onion address. The responses and certificate chains are stored in the database. Then, CARONTE computes the distance between pairs of responses, where one response comes from the Internet endpoint and the other was received through Tor from the hidden service. We detail this distance in Section 3.4.1. If it finds a pair of similar, non-generic, responses the leak is confirmed.

If the candidate endpoint corresponds to a site on the same Web server as the hidden service, that site can serve similar content as the hidden service or a completely unrelated site. If the candidate site is similar, when providing the candidate endpoint in the Host header, the similarity with the hidden service content will manifest. If the sites are different, when provided with the onion address in the Host header the Web server will serve back the hidden service content through the Internet, and the similarity will manifest. The Web server may be configured to serve the hidden service only through Tor, which prevents validation through the Internet. However, validation can also contact the Internet candidate endpoint through Tor, which would bypass this protection. Validation could also be performed using the clock-skew fingerprinting technique from Murdoch [36]. We leave these as future work.

The rest of this section describes the HTTP response similarity metric (Section 3.4.1), classifying leaks as intentional or unintentional (Section 3.4.2), and clustering confirmed leaks on the same servers (Section 3.4.3).

3.4.1 HTTP Response Similarity

Our HTTP response similarity metric takes as input two HTTP responses (one from the hidden service, the other from the candidate Internet endpoint) and outputs one if both responses have similar content and come from a similar server, zero otherwise. When the output is one, the candidate pair is flagged as a location leak.

One challenge is that the similarity should ignore generic pages, e.g., "It works" pages and default error pages, which do not indicate a leak even if observed in both responses. Another challenge is that the similarity needs to handle dynamic content, which may make two equivalent responses be different.

CARONTE uses a simple, yet effective, blacklisting mechanism to identify generic pages. To populate the blacklist, it hashes the body of every page in the database. Hashes that were retrieved from at least 5 different DNS domains or were associated to at least 5 status codes are added to the blacklist. When computing similarity, if any of the two responses has a body whose hash is in the blacklist, the similarity is considered zero and no leak is flagged.

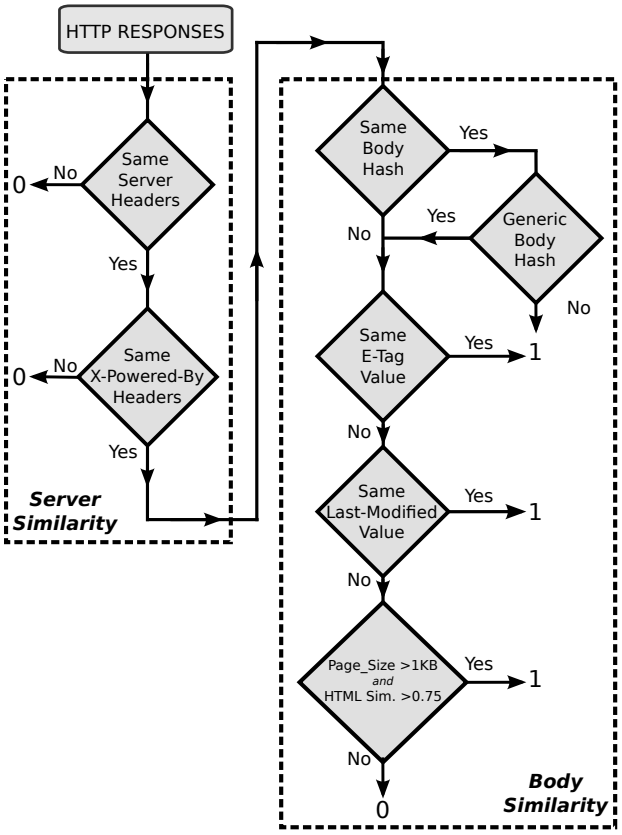

Figure 2: HTTP response similarity algorithm

The similarity metric uses 7 features. For each response, it computes the hash of the body and extracts the values of the E-Tag, Last-Modified, Content-Length, Server, and X-Powered-By headers. In addition, it computes the similarity of the two HTML documents using an off-the-shelf package [30], which compares the HTML tag structure of the documents, but not the tag contents. The HTML similarity is high even if both documents contain different dynamic content, as long as the structure of the HTML has not changed significantly.

The similarity metric computation is summarized in Figure 2. First, it determines if the content is served from similar servers. For this, it checks whether the Server and X-Powered-By header values are identical. If either the Server or X-Powered-By headers are missing in both responses it considers them identical. If those headers are not identical, it considers the responses different (i.e., outputs zero). Then, it determines that the body of both pages are similar if: they have the same hash and the hash is not in the blacklist, or if they have the same E-Tag value, or the same Last-Modified value, or if both pages are larger than 1 KB and their HTML similarity is greater than 0.75. This threshold allows for small changes in the HTML structure of the pages because the two pages may have been collected at different points in time.

3.4.2 Determining Leak Intention

Next, our approach determines if the leaks are unintentional or not. Clearly we cannot know for sure whether the leak was unintentional, but there exist some indications that the hidden service own-

	All	Live	HTTP	HTTPS
Onion URLs	31,849	4,794	4,617	177
Onion addr.	6,426	1,974	1,965	79

Table 2: Onion URLs and addresses in our collection (all), those returning content (live), and their split by protocol.

ers may not be trying to protect their service's location. CARONTE uses 3 automatic checks to classify a leak as intentional.

First, it compares the onion address and the Internet endpoint. If their longest common substring is larger or equal to 4 characters, they are considered similar. Since the onion address comes from a truncated hash, an onion address with a common substring with an Internet domain indicates brute-forcing on the onion address generation (e.g. `facebookcorewwwi.onion` and `www.facebook.com`).

Second, if the Internet site contains the onion address of the hidden service, the leak is also intentional. For example, the Internet site could advertise: "Hidden service available at `niazgxevgzlrbpvq.onion`". This check is only applied to connections where we do not spoof the Host header to prevent errors from servers that echo back the domain provided.

Third, it compares the Internet endpoint with the title of the onion page where the leak was validated. If the title of the page retrieved from the hidden service is embedding the Internet endpoint, likely the owner of the service is intentionally reminding users that the hidden service is also reachable through the Internet. This check is not applied to connections where CARONTE spoofs the Host header.

If none of these 3 conditions is satisfied, CARONTE considers the leak unintentional.

3.4.3 Clustering Leaks

A leak comprises of an onion address and an Internet endpoint (DNS domain or IP address). Multiple leaks may correspond to the same hidden service, e.g., a hidden service may change onion address over time or multiple DNS domains may match an onion address. To simplify the analysis of leaks, CARONTE clusters them based on their onion address, endpoint, and IP addresses.

First, for each leak with a domain endpoint, CARONTE resolves the domain and outputs a tuple ⟨onion_addr, endpoint, IP⟩ for each IP address the domain resolves to. A similar tuple is created for IP endpoints. The clustering uses a distance function that given two tuples returns zero if the onion address, the IP address or the effective second-level domain (ESLD) are the same, otherwise it returns one. The ESLD of a domain is the SLD unless the SLD enables third parties to obtain a subdomain, in which case it is the 3LD. For example, for `www.google.com` the ESLD is `google.com`, and for `books.amazon.co.uk` it is `amazon.co.uk`. To obtain a domain's ESLD CARONTE uses Mozilla's Public Suffix List [5].

The clustering starts with zero clusters and iterates on the list of tuples. For each tuple, if the distance is zero to any tuple already in a cluster it adds the tuple to the cluster. If the distance is zero to tuples in different clusters, it merges those clusters and adds the tuple to the merged cluster. Otherwise, it creates a new cluster for the tuple.

4. EVALUATION

This section details the evaluation of CARONTE: the datasets used (Section 4.1), the selection of candidate pairs (Section 4.2) and the validation of the candidate pairs (Section 4.3).

Type	Endpoints		Onion Addr.		Cand.
URLs	6,207	(11,207)	916	(2,301)	4,372
Domains	182	(231)	172	(269)	249
IPs	73	(22,182)	75	(89)	102
TOTAL					4,704

Table 3: Endpoints extracted from onion pages.

4.1 Datasets

Table 2 summarizes the collection and exploration of onion URLs. It shows the number of onion URLs and addresses initially collected from search engines and indices (*All*), the total number of onion URLs that returned some content (*Live*), and the split by protocol. Overall, CARONTE explored 31,849 onion URLs from 6,426 onion address. Only 31% of the onion addresses were alive, which may be due to non-HTTP(S) services, short-lived hidden services, and hidden services that change onion address over time. Only 4% of the live onion addresses offered HTTPS, which may be due to the communication with hidden services being encrypted in all hops and the difficulty of having a CA sign a certificate for an onion address. The exploration took place in 4 rounds, each round lasting three days on average.

Sonar data. The Sonar project [6] offers data from 68 Internet-wide scans performed between October 2013 and February 2015. Each scan has 3 files. The *certs* file (average size of 448MB per scan) has one row for each certificate with its SHA1 hash in DER format and the certificate in base64 encoding. Each row in the *names* file (30 MB) contains the certificate hash and the Subject's CN. The *hosts* file (2 GB) contains in each row the certificate hash and an IP address from where the certificate was collected. Overall, the Sonar data comprises 205 GB and contains 35 M distinct certificates.

4.2 Candidate Pairs

This section describes the candidate pairs extracted from the endpoints, identifiers, and certificates.

Endpoints. Table 3 summarizes the endpoints extracted from the onion pages collected during exploration. For each type of endpoint, it shows the number of distinct endpoints after filtering (before filtering in brackets), the number of onion addresses with at least one endpoint after filtering (before filtering in brackets), and the number of candidate pairs obtained from that type of endpoint. The numbers show that the filtering of non-Alexa domains and pages with more than 5 IP addresses removes 81% of the endpoints, most of those being IP addresses from Tor relay status pages. After filtering, the median contribution of each onion address is: 2 URLs, 1 domain from email addresses and 1 IP address. CARONTE produces 4,704 candidate pairs from endpoints: 93% from URLs, 5% from email domains, and 2% from IP addresses.

Identifiers. Table 4 summarizes the identifiers extracted from the onion pages collected during exploration. For each type of identifier, it shows the number of distinct IDs, the number of onion addresses and URLs containing at least one identifier of that type, and the number of candidate pairs obtained from these identifiers by searching on Internet search engines. CARONTE extracted 58 unique identifiers: 24 Google Analytics IDs, 3 Google AdSense IDs, and 31 Bitcoin wallets. Only 66 hidden services (3.3%) contained an identifier, indicating that most administrators are careful to avoid them. After querying the 58 identifiers on search engines, CARONTE found candidate Internet endpoints for 32 (64%), for a total of 192 candidate pairs.

Identifier	IDs	Onion Addr.	Onion URLs	Cand. Pairs
Google Analytics	24	33	52	146
Google AdSense	3	3	3	24
Bitcoin	31	31	36	22
TOTAL	*58*	*66*	*90*	*192*

Table 4: Identifiers extracted from the onion pages.

Method	Candidates Pairs	Onion Addr.	IPs	Dom.
Method 1	81	29	0	81
Method 2	188	30	188	0
Method 3	20	9	20	0
Method 4	97	30	37	60
TOTAL	*366*	*63*	*225*	*141*

Table 5: Candidates pairs from certificates.

Titles. After filtering titles in onion pages that appear in more than 10 hidden services CARONTE identifies 583 distinct titles. Looking up these titles on a search engine returns hits for 183 titles. From those hits, CARONTE produces 200 candidate pairs.

Certificates. Of the 79 onion addresses with port 443/tcp open, 50 provided a certificate chain. CARONTE uses 4 methods to obtain candidate pairs from HTTPS certificates, which are summarized in Table 5. The table shows the number of candidate pairs, and the number of distinct onion addresses, IP addresses, and DNS domains in those pairs. CARONTE produces a total of 366 candidate pairs from certificates. Next, we detail the results for each method.

First, CARONTE extracts DNS domains and IP addresses from the Subject's CN and SAN extension from each leaf certificate collected during exploration. Of the 50 leaf certificates, 11 (22%) contain only onion addresses, 3 (6%) contain both Internet endpoints and onion addresses, 26 (52%) contain only Internet endpoints, and 10 (20%) contain no Internet endpoint or onion address. The 3 certificates with both onion and Internet endpoints are likely doing this on purpose. However, 52% of the HTTPS hidden services are leaking endpoints in their certificate. This method produces 81 candidate pairs for 29 onion addresses (2.8 candidate endpoints per onion address).

Second, CARONTE searches the hash of the 50 leaf certificates in Sonar. It finds 30 of them, so at least 60% of the HTTPS hidden services are using the same certificate that an Internet Web server does. Of those 30, 26 were identified by the previous method. This method identifies 4 certificates that were not leaking an Internet endpoint but can still be found on Internet Web servers. Of those 4, 3 contain no Internet endpoints or onion addresses and the other one an onion address only. For each of the 30 certificates, Sonar provides one or more IP addresses from where the certificates were collected for a total of 188 candidate pairs.

Third, CARONTE searches the public keys of the 50 leaf certificates in Sonar. This method finds 9 new certificates that share their public key with one of the 50 leaf certificates. Those 9 certificates are observed being distributed from 20 IPs, of which 5 are not known from the previous methods.

Fourth, CARONTE searches Sonar for certificates that contain an onion address in their Subject CN or SAN extension. It finds 30 previously unknown certificates (i.e., not observed during exploration), each with one onion address. Some of those are additional Facebook certificates, which are excluded because they generate many useless candidate pairs. The IP addresses from where the remaining certificates were observed produce 97 candidate pairs.

Total. The left side of Table 6 summarizes the candidate pairs per method. In summary, CARONTE produced 5,462 candidate pairs: 86% from endpoints, 3% from identifiers, 4% from titles, and 7% from certificates.

4.3 Validation

Of the 5,462 candidate pairs, 303 (5.8%) successfully validated. Of those 303 pairs, 88 had an IP address as Internet endpoint and 215 a domain name. Those 303 pairs contained 100 unique onion addresses, 87 IP addresses, and 163 domains. One of the onion addresses offered two different hidden services, one through HTTP and the other through HTTPS. Thus, 101 hidden services were successfully deanonymized by CARONTE. Next, we classify leaks as intentional or not.

Intentional leaks. CARONTE automatically labels as intentional those leaks that match any of the 3 rules described in Section 3.4.2. Overall, 49% deanonymized hidden services are considered to be leaking their location intentionally. For these, the hidden service has an Internet site counterpart that serves similar content and runs on the same Web server, and there are references from the onion site to the Internet site and/or viceversa, e.g., the onion address appearing in the Internet site or the title of the onion site being similar to the domain of the Internet site. This result indicates that almost half of the hidden services CARONTE deanonymized may not be trying to protect the hidden server's location. This may be due to a selection bias in our collection towards the most advertised hidden services. One benefit to these Internet sites of having a hidden service is that users of the hidden service have their communication encrypted at all hops from the client to the hidden service. Furthermore, since the onion address is derived from the public key through a hash, the public key is self-authenticating, which helps preventing man-in-the-middle attacks.

Unintentional leaks. We manually analyze 15 of the unintentional leaks. They correspond to 4 cases. First, 5 hidden services run on a Web server with multiple virtual hosts, where the hidden service is not the default virtual host. When connecting to the hidden service providing a random domain in the Host header the Web server provides the default site, different from the hidden service, which contains the candidate Internet endpoints.

Second, 8 hidden services are deanonymized through certificate leaks, which provide a candidate Internet endpoint, even if the content of the hidden service did not provide any candidate. Of these, 4 are certificates that include the domain or IP address of the hidden server; the other 4 are certificates with an onion address observed by Sonar but not in our exploration.

Another hidden service is deanonymized because it contains an email address to a DNS domain with an Internet site also hosted in the same Web server. While the Internet site serves content unrelated to the hidden server, when querying the Internet site with the Host header being the onion address, the hidden service content is returned through the Internet.

The last deanonymization happens with two hidden services running on the same onion address through HTTP and HTTPS, respectively. One has an intentional leak to an Internet site offering the same content through both HTTP and HTTPS. When requesting the Internet site with the Host header being the onion address the two hidden services are observed, one on each port.

Location leak types. Table 6 summarizes the candidate pairs and deanonymizations per method. For the candidates it shows the total number of pairs and the distinct onion addresses in those pairs. The

Method	Candidates		Deanonymizations	
	Pairs	Onions	All	Unintentional
Endpoints	4,704	793	67	32
Identifiers	192	66	12	2
Titles	200	157	44	20
Certificates	366	63	30	18
TOTAL	5,462	841	101	51

Table 6: Summary of location leaks.

deanonymizations show the total onion addresses deanonymized by each method and how many of those are classified as unintentional. When considering all 101 deanonymized hidden services, URL endpoints are the most effective content leak being present in 67 deanonymizations, followed by titles (44), certificates (30), and identifiers (12). The identifiers correspond to Google Analytics (8) and Bitcoin wallets (4). Note that some deanonymizations contain several types of leaks. When considering only unintentional leaks, the most effective location leaks are endpoints (32), titles (20), and certificates (18). These numbers indicate that certificate leaks are best to identify unintentional leaks (60%) and identifiers worst (17%), which may indicate that administrators are already careful about them.

Clustering. Grouping the 101 hidden services by shared domains and IP addresses leaves 80 clusters. Of those, 12 contain multiple onion addresses. Three of those 12 correspond to hidden services that offer the same content through different onion addresses. The other 9 clusters with multiple onion addresses correspond to servers that are being used for hosting multiple hidden services.

Tor relay hosting. Of the 101 hidden services deanonymized, 21 are hosted on 13 Tor relays. Of those, 18 are considered intentional leaks and 3 unintentional, a smaller fraction of unintentional leaks compared with non-relay servers. This matches reports by the Tor project that several Tor relays were taken down during operation Onymous [42]. In some cases multiple hidden services are hosted at the same Tor relay, indicating that those hidden services belong to the owner of the Tor relay or that the Tor relay provides hidden service hosting services. Note that the Tor project discourages hosting hidden services on Tor relays as Tor relay runtime is known and can be correlated with the uptime of a hidden service [11]. Furthermore, CARONTE could be configured to consider the IP addresses of all Tor relays as candidates for all hidden services. We leave this as future work as our goal is to demonstrate open-world deanonymization and also to avoid impacting Tor relays.

Redirections. Fourteen hidden services redirect to an Internet site counterpart at some point of the study. Surprisingly, 9 of these hidden services redirect to the HTTP version of the Internet site. This is problematic for their users as they expect all hops of the communication to be encrypted when accessing a hidden service, but the last hop will not be. For these sites, the benefit of having a hidden service is not clear.

4.4 Performance

CARONTE currently runs from one off-the-shelf workstation. Once preprocessing finishes (e.g., parsing/indexing Sonar records) analyzing a hidden service takes close to one minute. Most time is spent in network connections. To scale CARONTE we could use multiple backend servers. Once a hidden service to be deanonymized is submitted a link could be generated where results would be available after the analysis completed.

5. DEFENSES

This section summarizes best practices that administrators of hidden services should take to eliminate location leaks. Some of these best practices are mentioned at the Tor project's how-to configure a hidden service [11].

Use a dedicated Web server. CARONTE can validate candidate domains when the same Web server is used to host both a hidden server and an Internet site. The Tor project recommends installing the hidden service in a separate Web server, but we believe this is a *must* and we recommend changing the wording to emphasize this. Administrators should specifically avoid reusing an existing Web server that already hosts an Internet site by simply creating a new virtual host. It is also a good idea to host the hidden service on a dedicated machine.

Bind the web server to localhost. Another best practice is binding the hidden service's Web server only to localhost, so that HTTP requests from Tor (running as a SOCKS proxy on localhost) are successfully answered, but HTTP requests from the Internet receive an HTTP Forbidden error response. We emphasize that this protection is required *in addition* to running the hidden server on a dedicated Web server. More specifically, if the hidden server is hosted on the same Web server as another Internet site, even if the Web server is bound to localhost, it is still possible to perform validation by contacting the Internet site over Tor, rather than through the Internet. To prevent configuration errors we recommend the Tor project to add detailed instructions on how to configure a Web server in this manner, at least for the most common open source Web servers.

A firewall can also block non-Tor connections to the hidden server. Network firewalls can block connections that do not come from a Tor relay. However, the list of Tor relays needs to be continuously updated. Host firewalls can only allow connections from localhost. Similar to binding the Web server to localhost, the hidden service should still be run on a dedicated Web server. The advantage is that Internet connections fail without generating an error message.

Another principle could be that if the server does not know its IP address it cannot leak it. Thus, the hidden service could be run in a NATed VM without a public IP address. However, other leaks (e.g., domain names in the content) could still happen.

Site auditing. Administrators should carefully audit their site's contents for leaks before making the site available as a hidden service. A general rule is that given the many possible sources of location leaks, the smaller the site, the easier the auditing and the smaller the probability of location leaks. During the audit, administrators could use CARONTE to identify validated location leaks, as well as to examine the candidate pairs CARONTE could not validate. Also, all external links should be made to point to onion addresses rather than Internet domains or IP addresses. Since not all Internet sites have a corresponding onion address, they should also audit all remaining IP addresses and Internet domains so that they do not point to the public IP address of the hidden server. They should also remove links to Internet domains or IP addresses owned by them, even when hosted in other servers, as tracking accesses to those other servers may reveal the identity of the administrators. Furthermore, identifiers such as email addresses, Google Analytics, Google AdSense, and Bitcoin addresses should be removed or at least not be reused in any other sites.

Certificates. Certificates with onion addresses in the CN or SAN fields should not include DNS domains or IP addresses, and should exclusively be used on Web servers that only run a hidden service bound to localhost. In addition, a hidden service should never reuse the certificate chain of another Internet site from the owner. Nowa-

days, the vast majority of hidden services use self-signed certificates since few CAs will issue a certificate for an onion address (only Facebook's hidden service seems to have a valid certificate chain [12]). Thus, certificate chains do not validate and will generate warnings, but man-in-the-middle attacks are limited by the fact that public keys of hidden services are self-authenticating.

Avoid Tor relays. Hidden services that want to hide their location should not be hosted on Tor relays, as this enables attackers to perform closed-world validation using the IP addresses of all Tor relays as candidates for any hidden service to be deanonymized. Also, Tor relay uptime is public and can be correlated with hidden service uptime. Similarly, running a hidden service on a machine that is not always on enables uptime correlation attacks.

6. ETHICAL CONSIDERATIONS

While there exist tools like Shadow [31] to simulate attacks against Tor, the application layer leaks studied in this paper cannot be easily simulated because they are caused by erroneous configurations by administrators of hidden services. Similar to Biryukov et al. [17] we deem experiments on the live Tor network worthwhile and necessary to enhance the scientific understanding of hidden services, as long as they do not cause degradation of the network and hidden services.

Our approach does not add any malicious Tor relay and gives us no access to the plaintext of traffic of any user. We purposefully avoid deanonymization techniques based on exploiting software vulnerabilities. However, our data is sensitive because it contains location information on some hidden services that may want to protect their location. This work has been approved by our institution's ethics review board, which has mandated that due to its sensitive nature the data must be protected with diligence, must not be disclosed to third parties, and must be deleted when the paper is accepted for publication. Furthermore, in this paper we do not disclose any deanonymized hidden service but only provide aggregate data or fake examples to illustrate important steps and findings.

During the exploration of hidden services, to prevent downloading copyrighted material and offensive content (e.g., pornography), we limit CARONTE to collecting textual and HTML content, ignoring other content such as images, videos, or documents.

We have sent a copy of this draft to the Tor project to give them a heads up on the work.

7. RELATED WORK

The first generation of Tor's hidden services is described in the original design paper [21], but has since been revised [13]. Øverlier and Syverson [40] first demonstrated techniques to deanonymize hidden services. They show how an adversary could lie about the available bandwidth of a relay it controls to increase the probability of that relay being selected for a hidden service circuit. This adversary can repeatedly connect to a hidden server until traffic correlation indicates that the hidden server built a circuit to one of the adversary's relays. Bauer et al. [15] extended the attack to general purpose circuits. As the result of these attacks, entry guard nodes were added to the Tor hidden services specification [13, 46]. Elahi et al. [23] propose improvements to reduce the guard compromise rate. Biryukov et al. [17] deanonymize hidden services in the presence of guard nodes by combining the bandwidth inflation attack with a technique to phase their relays in and out of the consensus at will without them losing their flags.

These attacks target a specific hidden service, while we show that large-scale deanonymization of many hidden services is possible through content leaks. Also, these attacks rely on vulnerabilities in the Tor specification that can be centrally addressed by the Tor project, while content leaks need to be fixed by each hidden service. Furthermore, our attacks do not require adding a malicious relay to the Tor network.

Most similar to our approach is the work by Crenshaw on deanonymizing I2P eepSites [20]. Crenshaw sets up an I2P router and uses the IP addresses of the peers known to his router as candidates for hosting the eepSites. In Tor, which is not peer-to-peer based, his approach is similar to considering the list of all Tor relays as candidates for hosting any hidden service. This closed-world approach would enable deanonymizing only 21% of hidden services that are hosted on Tor relays. Our approach in contrast shows how to move from an open-world to a closed-world by extracting candidates from the identifiers and endpoints in the content of hidden services, as well as their HTTPS certificates. In addition, his validation step relies solely on similar Server headers, which can produce a high number of false positives.

A different deanonymization approach uses clock-skew measurements when repeatedly connecting to a hidden service [36, 47]. This attack also assumes a closed-world. We propose techniques to move from a closed-world to an open-world and could also use this technique in our validation.

DoS attacks. Selective denial-of-service attacks on relays can force circuits to be re-built, increasing the probability of end-to-end compromise [18]. Øverlier and Syverson [41] introduce Valet Service nodes to improve the resilience of introduction points against DoS attacks. Jansen et al. [32] deanonymize hidden services through selective denial-of-service of relays' memory that forces the hidden service to choose guard nodes in control of the adversary. The attack requires hours or days to deanonymize a single hidden service and requires the adversary to identify the target's guards.

Forensics. A separate line of work considers the forensics problem of proving that a confiscated machine hosted a hidden service [24, 43]. Those works assume the hidden service logs the requests and place identifiable fingerprints in the log files through crafted queries.

AS-level adversaries. An attacker that is able to observe encrypted traffic from a client to the first relay and from the final relay to the destination can link the client and destination by correlating traffic patterns [44]. Prior work has studied Tor's vulnerability to adversaries that can establish themselves in that position both at the Internet exchange [38] and the AS [14, 22, 26, 33, 45] levels. Our attacks instead deanonymize hidden services.

Traffic analysis. Prior work proposes traffic analysis attacks on Tor that make probabilistic inferences about relays and clients that are part of a specific circuit. One approach is to congest a relay [25, 37]. Another approach leverages network-level characteristics such as circuit throughput and latency [29, 35]. These attacks target clients and may not fully deanonymize them, while our attacks fully deanonymize the location of hidden servers.

8. CONCLUSION

In this paper we have presented CARONTE, a tool to deanonymize hidden services through location leaks in their content and configuration. CARONTE implements a novel approach to deanonymize hidden services that does not rely on flaws on the Tor protocol and assumes an open-world, i.e., it does not assume a short list of candidate servers is known in advance. Instead, it implements novel techniques to identify candidate servers from the content and configuration of a hidden service, which enable moving from an open-world to a closed-world.

Using CARONTE we perform the first measurement study on the prevalence of location leaks in hidden services. Out of 1,974 live HTTP hidden services, CARONTE successfully deanonymizes the location of 5% of them. Of the deanonymized hidden services 21% are running on Tor relays. The remaining 79% could not be deanonymized in a close-world.

9. ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their feedback. This work was performed while Srdjan Matic was a visiting Ph.D. student at the IMDEA Software Institute.

This research was partially supported by the Regional Government of Madrid through the N-GREENS Software-CM project S2013/ICE-2731 and by the Spanish Government through the StrongSoft Grant TIN2012-39391-C04-01. All opinions, findings and conclusions, or recommendations expressed herein are those of the authors and do not necessarily reflect the views of the sponsors.

10. REFERENCES

[1] Certificate Transparency. http://www.certificate-transparency.org/.

[2] Deep web links: .onion hidden service urls list. http://deepweblinks.org/.

[3] Duckduckgo. http://3g2upl4pq6kufc4m.onion.

[4] Hidden wiki: Tor .onion urls directories. http://thehiddenwiki.org/.

[5] Public Suffix List. http://publicsuffix.org/.

[6] Rapid 7: Sonar Project. https://scans.io/study/sonar.ssl.

[7] Sameid. http://sameid.net/.

[8] The ICSI Certificate Notary. http://notary.icsi.berkeley.edu/.

[9] The Invisible Internet Project. https://geti2p.net/.

[10] Tor hidden service (.onion) search. https://ahmia.fi/search/.

[11] Tor project: Configuring hidden services for tor. https://www.torproject.org/docs/tor-hidden-service.html.en.

[12] Tor project: Facebook, hidden services, and https certs. https://blog.torproject.org/blog/facebook-hidden-services-and-https-certs.

[13] Tor project: Tor rendezvous specification. https://gitweb.torproject.org/torspec.git/tree/rend-spec.txt.

[14] M. Akhoondi, C. Yu, and H. V. Madhyastha. LASTor: A Low-Latency AS-Aware Tor Client. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2012.

[15] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker. Low-resource Routing Attacks Against Tor. In *Proceedings of the ACM Workshop on Privacy in Electronic Society*, 2007.

[16] A. Biryukov, I. Pustogarov, F. Thill, and R.-P. Weinmann. Content and Popularity Analysis of Tor Hidden Services. In *Proceedings of the First International Workshop on Big Data Analytics for Security*, June 2014.

[17] A. Biryukov, I. Pustogarov, and R.-P. Weinmann. Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2013.

[18] N. Borisov, G. Danezis, P. Mittal, and P. Tabriz. Denial of Service or Denial of Security? In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, 2007.

[19] N. Christin. Traveling the silk road: A measurement analysis of a large anonymous online marketplace. In *Proceedings of the 22nd international conference on World Wide Web*, pages 213–224. International World Wide Web Conferences Steering Committee, 2013.

[20] A. Crenshaw. Darknets and hidden servers: Identifying the true IP/network identity of I2P service hosts, 2011. Black Hat DC.

[21] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium*, 2004.

[22] M. Edman and P. Syverson. AS-awareness in Tor Path Selection. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, 2009.

[23] T. Elahi, K. Bauer, M. AlSabah, R. Dingledine, and I. Goldberg. Changing of the Guards: A Framework for Understanding and Improving Entry Guard Selection in Tor. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society*, 2012.

[24] J. A. Elices, F. Perez-Gonzalez, and C. Troncoso. Fingerprinting Tor's Hidden Service Log Files Using a Timing Channel. In *Proceedings of the IEEE International Workshop on Information Forensics and Security*, 2011.

[25] N. S. Evans, R. Dingledine, and C. Grothoff. A Practical Congestion Attack on Tor Using Long Paths. In *Proceedings of the 18th USENIX Security Symposium*, 2009.

[26] N. Feamster and R. Dingledine. Location Diversity in Anonymity Networks. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society*, 2004.

[27] R. Graham. Reading the Silk Road configuration, October 2014. http://blog.erratasec.com/2014/10/reading-silk-road-configuration.html.

[28] A. Greenberg. Wired: Global Web Crackdown Arrests 17, Seizes Hundreds Of Dark Net Domains, November 2014. http://www.wired.com/2014/11/operation-onymous-dark-web-arrests/.

[29] N. Hopper, E. Y. Vasserman, and E. Chan-TIN. How Much Anonymity Does Network Latency Leak? *ACM Transactions on Information Systems Security*, 13(2):1–28, Feb. 2010.

[30] Html::Similarity. http://search.cpan.org/~xern/HTML-Similarity-0.2.0/lib/HTML/Similarity.pm/.

[31] R. Jansen and N. Hopper. Shadow: Running Tor in a Box for Accurate and Efficient Experimentation. In *Proceedings of the 19th Annual Network and Distributed System Security Symposium*, February 2012.

[32] R. Jansen, F. Tschorsch, A. Johnson, and B. Scheuermann. The Sniper Attack: Anonymously Deanonymizing and Disabling the Tor Network. In *Proceedings of the 21st Annual Network and Distributed System Security Symposium*, 2014.

[33] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson. Users Get Routed: Traffic Correlation on Tor by Realistic Adversaries. In *Proceedings of the ACM Conference on Computer and Communications Security*, 2013.

[34] B. Krebs. Silk Road Lawyers Poke Holes in FBIâĂŹs Story, October 2014.

http://krebsonsecurity.com/2014/10/silk-road-lawyers-poke-holes-in-fbis-story/.

[35] P. Mittal, A. Khurshid, J. Juen, M. Caesar, and N. Borisov. Stealthy Traffic Analysis of Low-latency Anonymous Communication Using Throughput Fingerprinting. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, 2011.

[36] S. J. Murdoch. Hot or Not: Revealing Hidden Services by Their Clock Skew. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, 2006.

[37] S. J. Murdoch and G. Danezis. Low-Cost Traffic Analysis of Tor. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2005.

[38] S. J. Murdoch and P. Zieliński. Sampled Traffic Analysis by Internet-exchange-level Adversaries. In *Proceedings of the 7th International Conference on Privacy Enhancing Technologies*, 2007.

[39] Netcraft. Keys left unchanged in many Heartbleed replacement certificates!, April 2014. http://news.netcraft.com/archives/2014/05/09/keys-left-unchanged-in-many-heartbleed-replacement-certificates.html.

[40] L. Øverlier and P. Syverson. Locating Hidden Servers. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2006.

[41] L. Øverlier and P. Syverson. Valet Services: Improving Hidden Servers with a Personal Touch. In *Proceedings of the 6th International Conference on Privacy Enhancing Technologies*, 2006.

[42] T. Project. Thoughts and concerns about operation onymous, November 2014. https://blog.torproject.org/blog/thoughts-and-concerns-about-operation-onymous.

[43] B. Shebaro, F. Perez-Gonzalez, and J. R. Crandall. Leaving Timing-channel Fingerprints in Hidden Service Log Files. *Digital Investigations*, 7:104–113, Aug. 2010.

[44] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr. Towards an Analysis of Onion Routing Security. In *Designing Privacy Enhancing Technologies: Workshop on Design Issue in Anonymity and Unobservability*, July 2000.

[45] L. Vanbever, O. Li, J. Rexford, and P. Mittal. Anonymity on QuickSand: Using BGP to Compromise Tor. In *Proceedings of the 13th ACM Workshop on Hot Topics in Networks*, 2014.

[46] M. Wright, M. Adler, B. N. Levine, and C. Shields. Defending Anonymous Communications Against Passive Logging Attacks. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2003.

[47] S. Zander and S. J. Murdoch. An Improved Clock-skew Measurement Technique for Revealing Hidden Services. In *Proceedings of the 17th USENIX Security Symposium*, 2008.