The Once and Future Onion

Paul Syverson

U.S. Naval Research Laboratory paul.syverson@nrl.navy.mil

Abstract. Onionsites are Internet sites accessed via protocols offering security protections beyond those provided by the usual protocols and infrastructure of the Internet, such as confidentiality of address lookup, and that significantly strengthen commonly offered protections; for example, their self-authenticating addresses preclude the kinds of certificate hijacks that have occurred against registered domain names. I will sketch the properties and design of onion services, including early history as well as recent developments. I will also describe integration of onionsites much more fully into conventional Internet sites in ways that promote their general widescale adoption.

1 Introduction

Prior to a decade ago, website access via encrypted and authenticated connections was relatively uncommon. Now this is recognized as fundamental to online commerce, government, and more generally to functioning in many aspects of modern life. The mechanisms for secure site access that we will discuss herein are roughly where certificates and TLS were at the turn of the century. I will describe combining and extending protections provided by such conventional mechanisms with the stronger mechanisms of Tor's onion services in ways that both further improve the security and usability that is currently provided by either alone and that promote broad adoption of more secure site access.

1.1 Predecessors to Onion Services

We introduced onion routing in the 1990s "to separate identification from routing" for networked communication [21]. Primary intended uses were for clients to connect to Internet sites with publicly discoverable network locations, such as connecting to ordinary websites, but without revealing to the infrastructure carrying the connection's traffic, who is visiting which site. At the same time we introduced onion routing we also introduced *reply onions*, which were designed to allow replies to such connections or to otherwise permit connection to sites with hidden locations [7]. One application we proposed for reply onions was private location tracking: user location was regularly uploaded to a user's server, which could then selectively provide access to the user's location information. The sensors and routing infrastructure, however, could not tell which user was sending her location to which server. Another application was a protocol to permit mobile telephony, including per-call billing, without revealing to the local cell tower what phone number is making the call or, to the account provider, where the call is being made from [22]. Ross Anderson introduced the design for a censorship-resistant Eternity Service the same year we introduced onion routing [1], which featured the location-hiding placement and retrieval of documents at redundant distributed servers. These were all designs without any implementation. The first system with at least a research implementation to permit connections to a service without revealing the service's network location was Rewebber [6], followed a few years after by Publius [30]. These were systems specifically for connecting to a web service, a primary application of Tor's onion services half a decade later.

1.2 Basic overview of Tor design and onion services

I now give a high-level description of Tor and onion service protocols that should be sufficient to understand what follows. For more detailed descriptions see the Tor design paper [5] and related documentation at the Tor website [28]. For a high-level graphical description of onion services see [25]. For a more up to date, and much more technical, description of onion services protocols see the Tor Rendezvous Specification [27].

Tor clients randomly select three of the many thousands of relays [26] comprising the current Tor network, and create a cryptographic circuit through these to connect to Internet services. Since only the first relay in the circuit sees the IP address of the client and only the last (exit) relay sees the IP address of the destination, this technique separates identification from routing.

To offer an onion service, a web (or other) server creates Tor circuits to multiple *introduction points*, Tor relays that await connection attempts from clients. A user wishing to connect to a particular onion service uses the service's onion address to look up these introduction points in a directory system. In a successful interaction, the client and onionsite then both create Tor circuits to a client-selected relay, the *rendezvous point*. The rendezvous point joins their circuits together, and they can then interact as ordinary client and server of a web connection over this rendezvous circuit.

Since the onionsite only communicates over Tor circuits it creates, this protocol hides its network location, the feature that gives it the name 'hidden service'. But, there are other important features to the onion service protocols, notably self-authentication. The onion address is the hash of the public key of the onionsite. For example, if one wished to connect to the DuckDuckGo search engine's onion service, the address is 3g2upl4pq6kufc4m.onion. If that address is linked to or entered in the address bar of Tor Browser (a browser based on Firefox ESR, designed to work with Tor, and bundled in the default Tor download), the Tor client recognizes this as an onion address and thus knows to use the above protocol rather than attempting to pass the address through a Tor circuit for DNS resolution at an exit relay. The public key hashed to produce the address corresponds to the key that signs the list of introduction points and other service descriptor information provided by the directory system. In this way, onion addresses are self-authenticating, a central point to which we will return.

2 The Alliuminated Web

Users are generally completely in the dark about how their information moves around the Internet. Though Tor does provide confidentiality of routing metadata, it also provides the user with far more routing metadata, indeed authenticated routing metadata, than she would otherwise have, and does so in a highly usable fashion. A pulldown on the Tor Browser indicates the country and IP address of the relays in the path of an active Tor circuit.

As noted, we originally called Tor onion services "hidden services" (actually "location-hidden services" in the first publication [5]). This was perhaps natural given the above history, but it was misleading terminology in at least two ways. First, given the varied and nuanced meanings of 'hidden' it is easy to insinuate a general air of exotic mystery and arcane offerings on such sites, rather than the mundane idea that network location is not revealed merely by making a site reachable. Calling these "hidden services" did not exactly dissuade those tech pundits and television drama writers who might be generally inclined to titillating and frightening stories that boost readership and ad revenue.

More important technically, it calls attention to only one sort of protection that onion services provide, hiding the network location of the service. This is an important security property, and researchers and developers continue to work on strengthening its protection. But putting just that aspect into the name makes it easy to downplay the other important protections that onion services provide. In fact, while other properties such as self-authentication remain inherent, location hiding is now a configuration option since it is not desirable for all settings. Because 'hidden services' was importantly misleading in multiple ways, we now generally refer to these simply as "onion services".

3 Evolution of Onion Services

Guards and Vanguards: One of the first design changes to occur after we introduced onion services in 2004 was to add *entry guards*. A malicious client can rapidly request many connections to an onion service, each of which will cause the onion service to use a new circuit to the rendezvous point. Setting up even a single relay and making many connections to an onion service, we were able to correlate connections we requested with ones from a server connecting into the Tor network at our relay. We were thus able to find the network address of the onion service within minutes. To counter such attacks, we introduced guard relays, a set of a very few relays that a client used persistently to connect to the Tor network [19]. Guards protect onion-service-originated circuits, but also all clients circuits. Normal clients make multiple connections to multiple sites during the course of their online activity—albeit normally at a much slower rate than just described. In that same work, we showed that a similar attack could quickly uncover a service's entry guards, and we proposed *layered guards* as a means to make such attacks on onion services even slower and more complex. Over the last decade, many have researched this area, for example exploring the performance implications of using layered guards for hidden services [12]. Design and implementation specifics are actively being settled at the time of writing. Further details can be found in a Tor Proposal [13]. (Tor Proposals are similar to IETF RFCs.)

Counters to Mining the Onion Service Directory: The first onion-service directory system, for looking up introduction points and other information given in a service descriptor, was run at the Tor directory authorities (which maintain and serve information about the relays comprising the Tor network). But this was only intended to get onion services up and running, and even the original Tor design paper mentioned running the directory on a distributed hash table (DHT) comprised of Tor relays [5]. The DHT-based onion-service directory system was deployed a few years later. Even with the dynamic distribution of a DHT, an adversary occupying any of the six positions holding at a given time the service descriptor for a given onion address could monitor when lookups of it occur, and could even deny service if it held all six positions in the DHT. We proposed a partial counter to this by encrypting both the record locator and its content using the onion address as key [20]. This was later implemented and deployed [27]. Though deployed, it was not widely used, and published research showed how adversaries could position themselves in the DHT to learn (or block) most onion addresses [3].

Even if widely used, such encryption would not resist DHT monitoring or DoS of onion addresses an adversary knew otherwise. Something that does help even in this case (and is now implemented and deployed) is for the Tor directory authorities to run a distributed random-value-commitment protocol to be used in the determination of next-round DHT assignments, thus confounding any adversary's attempt to predictably position itself within the DHT [9].

Metrics for onions: Onion-service traffic constitutes a tiny fraction of overall traffic on the Tor network, but until a few years ago we had no idea how much. This is now regularly reported on the Tor Metrics site and is roughly 1–5% of overall application traffic [17]. Likewise the number of onion addresses that exist (c. 50K at any time), are reachable, serve content, etc. was not known. These latter appear to be far fewer, on the order of 10K and 1K respectively, but good numbers are not yet readily available. Collecting such statistics without harming privacy is difficult [8]. Future work using more secure techniques, such as provided by PrivCount [11], should give us additional statistics, e.g., the per-onion-service distribution on connections to onion services during a given period. The Tor Metrics site also tracks performance, reporting on the time to download various size files over the network. Until recently, this was limited to downloads from external servers via exit circuits. With the introduction of OnionPerf [10], more complex traffic performance could be generated and monitored, and in particular, performance of onion services is now measured and reported.

Ephemeral and Personal Onions: Further complicating things, onionsites are not all ordinary web pages. As just one example, OnionShare [18] is a tool for secure and private file transfer. It creates an onion service on the source computer and places the desired file at its onion address. In default use, once the file is retrieved, the onion service and the file are deleted. Obviously such onionsites complicate our understanding of onionsite statistics. Another example of a different sort of onion service is Ricochet [23], a secure instant messaging system with no central server. Each Ricochet user has an onionsite on his computer and shares the onion address with potential communicants. Two users wishing to talk will connect to each other's onionsite. Ricochet presents the exchanged messages as a dialogue in its GUI. Onionsites can also be useful for securely operating a personal cloud service. With privacy and cost in mind, many people are operating their own cloud infrastructure to store files and calendar entries using open-source systems such as Cozy.

Facebook and increased integration with the less-secure web: Thousands of users connect to Facebook from locations that do not allow direct connections to facebook.com. And many others simply use Tor Browser for the added security it provides for general Internet activity. Indeed, in April 2016, Facebook reported over a million people accessing Facebook over Tor [16]. Now Facebook could simply encourage users to make an ordinary connection over Tor to facebook.com. But on the Tor network limited exit capacity is often a dominating factor for Tor performance. This was one of the motivations Facebook described for offering an onionsite rather than merely encouraging connections to their registered domain via Tor [15]. More recently, Facebook has begun allowing onionsite owners to offer previews of their sites to non-Tor users on pages with a link to their onionsite. Facebook also provides guidance for anyone attempting to follow such a link using a non-Tor browser, telling them how and why they might use Tor. And if the onionsite has opted to allow it, a link to the less secure (non-Tor) version of the site is also offered [24].

Facebook is the largest site by far to incorporate onion service, but is not the only significant "conventional" site to do so. A few other examples include ProPublica, a well-known news site, DuckDuckGo, a popular search engine I have already mentioned, and services and repositories of the Debian operating system. Some news sites do not, at the time of writing, offer onion addresses for accessing their content but do make use of SecureDrop, which is an onion service for sources to securely and anonymously contribute to media organizations including The Washington Post, The New Yorker, and The Globe and Mail.

A potential concern for popular mainstream sites is doppelgangers. If someone were to put up an onionsite at 3g2upk4au4ldfc4m.onion that appears to be the DuckDuckGo homepage, users might not spot that they had not reached 3g2upl4pq6kufc4m.onion. Onion addresses are self-authenticating, but by themselves offer nothing to tie themselves to known public entities. This is an example of Zooko's Triangle, which states that names can be any two of decentralized, secure, and human-meaningful, but not all three at once. One of the ways to get closer to having all three is to leverage TLS certificates. If 3g2upl4pq6kufc4m.onion is entered in the Tor Browser, the display in the URL bar shows "Duck Duck Go, Inc. (US) | https://3g2upl4pq6kufc4m.onion": for this address, DuckDuckGo has obtained a TLS certificate that includes the identification of itself as the organization holding the certificate. And that is possible because the CA/Browser Forum has authorized the issuance of extended validation (EV) Certs for onion addresses. Note that this provides an additional element of site-owner control over authentication that no certificates can: even with an accepted certificate, without the private key from which the onion address derives, an adversary cannot read or respond to traffic encoded for that address (though this does not preclude certificates for doppelgangers). One important enabling condition for allowing issuance of certs for onion addresses was the recognition of .onion as a reserved top-level domain by the IETF in 2015 [2]. RFC 7686 designated .onion as a special-use TLD: onion addresses are not be resolved by DNS as an ordinary registered domain, and they are given a standardized status.

Only EV certs are eligible for display of the organization name and lock icon together in the browser URL bar. And, onion addresses are only eligible for EV certs. This limits them to entities with enough time, money, and motivation to jump through the hoops necessary to obtain them. Smaller or less well-funded entities generally obtain domain validation (DV) certificates, which are much quicker and easier to obtain. One of the concerns that the CA/Browser Forum had concerning onion addresses, prompting the limitation to EV certs, was the 16-character names that might make them vulnerable to hash collisions. Whatever the validity of that or some other expressed cryptographic concerns, they should all be addressed by the new protocols and 56-character names [14] that are already in the Tor-alpha code release and should be in the stable release by the time this paper is published.

The motivations for Facebook to run an onion service, e.g., as cited above, do not include hiding server network location. As such, the original protocol's use of Tor circuits from the onion service to the rendezvous point and to the introduction points only adds overhead and reduces performance for both the onion service and the Tor network. Facebook thus uses single onion services. These make direct connections from the onion service to the rendezvous and introduction points and are now specified and implemented for general Tor use [4].

4 John Jacob Onionheimer Schmidt

Should the CA/Browser Forum approve issuing DV certs for onion addresses, it will further advance the integration of onion services with existing, familiar authentication mechanisms. But even if that happens, it will not permit the inclusion of organization names in the URL bar or solve other problems associated with addresses that are not generally understood or recognizable by humans.

The Onion Name System (OnioNS) attempts to respond to these concerns by creating a system for globally-unique but still human-meaningful names for onionsites [29]. This has the advantage of not being dependent on any existing naming scheme, such as existing domain registration. On the other hand, through much experience and design, existing approaches to naming have evolved effective usage and infrastructure that we can leverage. And integrating onion addresses with registered domain names has other advantages.

One way to further this integration is literally, i.e., by incorporating onion addresses as subdomains of registered domain names. Top-level onion addresses will still be important, particularly for sites without registered domain names. And this does not automatically require 'onion' to be part of the name, but the address should be self-authenticating as onion addresses are and should have adequate encoding properties to preclude confusion with subdomain names not intended to provide this property. Whether or not that will require standardization or regulation along the lines of RFC 7686 will need to wait for more details than I present herein. But, as a strawman illustration, imagine 3g2upl4pq6kufc4m.onion replaced by 3g2upl4pq6kufc4m.onion.duckduckgo.com. This would have numerous positive prospects.

First, this is not a top-level onion address as in RFC 7686. Thus non-Tor browsers can resolve and reach this address. As long as the site has content there, the browser should be able to load it. There will not be a self-authentication check or other security protections that the Tor Browser adds, nor the routing security that comes by accessing the service via Tor. Assuming no adversary shenanigans, however, nothing will break. This should make it appealing to site owners wanting to minimize overhead and duplicated effort.

Second, because the onion address is simply a subdomain of a registered domain, it can be covered by a DV cert from any certificate authority that allows wildcards or the issuing of certs for multiple subdomains. Thus, the address can be human-meaningful, self-authenticating (if appropriate checks are done), and still give users the familiar indications that the connection is secure (lock icon indicating a valid cert from a recognized CA). I will return to this below.

Third, it leverages existing human-meaningful names in a way similar to other things sites currently do. Whatever user-education component is needed to engender understanding of the security advantages, there is little or no need for an established domain to create a campaign to explain a surprising address change in the URL bar to its users.

Further, it would now be easy for a site to offer multiple subdomain onion addresses that are automatically tied to one another via their primary domain name. These could be to offer different services at different places or to different users, but it is also an easy way to do expiry or revocation without needing to interact with CRLs or possibly even keep track of user accounts. One can route multiple onion subdomains to the same page. If one wants to revoke or expire access for the users reaching the content or service via a particular onion subdomain, one can simply throw the relevant private key away. Also, one can do self-certification for some content within a certified domain, for example to do load balancing and content distribution. Finally, a site that provides a platform for its users to host individual pages or content and that has a wildcard certificate, e.g., Facebook, could allow users to set up their own onionsites on the hosting site with the user's onion key "certified" by the host's onion key. This would allow users much more direct control over authentication of and access to their content, while still providing TLS certification of the host and host "certification" of the user's onionsite. There are many details and limitations for some of these to be practical, but this should give an inkling of the potential.

5 Onions Everywhere

Subdomain onion addresses should be eligible for DV cert issuance just like any other subdomain. But to get full security advantages, issuance protocols will need to make sure that relevant checks for possession of the domain, the private TLS key, and the private onion-service key all properly validate each other. They should also be checked, e.g., to verify that it is not possible to interleave one type of expired key or proof of access with still-valid keys of another type, resulting in an extension or escalation of authorization. In short, there is some research to be done, even without getting into questions of performance.

Relatedly, a Tor-Browser connection to a subdomain onion service should provide all the security advantages of current Tor-based access to the onion service, together with the protections provided by certified TLS. (It should after the client software and onion-service directory system have been updated to handle such addresses.) And as noted above, subdomain onion services will be backwards compatible in that a browser knowing nothing about Tor will be able to reach and interact with the service. But intermediate levels of protection are also enabled by this approach. Browsers not configured to access Tor could still have plugins or modifications that check for possession of the appropriate private key associated with an onion address. Though not offering the routing protection of connecting via Tor, resistance to DNS hijack and certificate hijack is significantly improved since it would be necessary to overcome the self-authentication at the same time.

An adversary could in principle do all the relevant lookup, routing, and certificate hijacks, coupled with a phished or otherwise insinuated doppelganger onion address. Even this could be countered by building the right onion address into the HTTPS Everywhere ruleset. HTTPS Everywhere is a free and open browser extension that checks for a TLS-protected equivalent to a requested HTTP connection and then substitutes the appropriate protected connection request. The need for a ruleset is both because not every site offers an HTTPS version, and because simply adding an "S" to "HTTP" will not always take the user to the equivalent site, which depends on the configuration and policies of the site in question. The equivalent encrypted content may be at a slightly different address, and an HTTPS connection to the URL as requested may go to a different page within the domain. If one adds onion addresses to the HTTPS Everywhere ruleset for Tor Browser and other browsers configured to parse and check onion authentication, then this too would have to be overcome for such attacks to succeed.

Furthermore, with existing onion addresses, ruleset redirection would again raise user-surprise concern if a request for a given URL yields a completely different-looking and not-apparently-related address in the URL bar. With subdomain onions, the redirection is much more along the lines of existing HTTPS

Everywhere switches. User surprise should thus be comparable to the current status quo.

User-friendly onionsite set up: Let's Encrypt is a certificate authority that allows anyone to obtain a free DV cert for her site. But it is more than that. Let's Encrypt strives to make certificate issuance as quick, automatic, and transparent as possible, so that site owners have as painless an experience as possible setting up a TLS-protected version of their site. Once the above mentioned systems and protocols are in place, it would be natural for Let's Encrypt to facilitate an onion-protected version of a site just as they do now for TLS protection.

6 Conclusion

I hope the nature, history, and prospects for onion services are now well alliuminated for you. I hope also that you are enthusiastic to see subdomain onion addresses researched, specified, implemented, and deployed as sketched above. In such a future, individual, business, and government websites and services can all be set up to offer much more secure access than is now possible.

Acknowledgments

More people have helped shape the work and ideas I have described above than could be acknowledged here. Specific thanks to Richard Barnes for conversations that led to the ideas for subdomain onions, and to Matt Traudt and Ryan Wails for helpful comments on a draft of this paper.

References

1. Anderson, R.: The eternity service. In: 1st International Conference on the Theory and Applications of Cryptology (Pragocrypt '96). pp. 242–252. Czech Technical University Publishing House, Prague, Czech Republic (September/October 1996)
2. Appelbaum, J., Muffett, A.: The .onion special-use domain name. `https://tools.ietf.org/html/rfc7686` (2015)
3. Biryukov, A., Pustogarov, I., Weinmann, R.P.: Trawling for Tor hidden services: Detection, measurement, deanonymization. In: IEEE Symposium on Security and Privacy (SP), 2013 (2013)
4. Brown, T.W., Brooks, J., Johnson, A., Jansen, R., Kadianakis, G., Syverson, P., Dingledine, R.: Rendezvous single onion services, (Tor proposal 252. `https://gitweb.torproject.org/torspec.git/tree/proposals/260-rend-single-onion.txt` (2015)
5. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The Second-Generation Onion Router. In: Proceedings of the 13th USENIX Security Symposium (August 2004)
6. Goldberg, I., Wagner, D.: TAZ Servers and the Rewebber Network: Enabling Anonymous Publishing on the World Wide Web. First Monday 3(4) (April 1998)

7. Goldschlag, D.M., Reed, M.G., Syverson, P.F.: Hiding routing information. In: Anderson, R. (ed.) Information Hiding: First International Workshop. pp. 137–150. Springer-Verlag, LNCS 1174 (May 1996)
8. Goulet, D., Johnson, A., Kadianakis, G., Loesing, K.: Hidden-service statistics reported by relays. Tor Tech Report 2015-04-001, The Tor Project (April 2015)
9. Goulet, D., Kadianakis, G.: Random number generation during Tor voting, (Tor proposal 250). https://gitweb.torproject.org/torspec.git/tree/proposals/250-commit-reveal-consensus.txt (2015)
10. Jansen, R.: Onionperf. https://github.com/robgjansen/onionperf
11. Jansen, R., Johnson, A.: Safely measuring Tor. In: Proceedings of the 23rd ACM conference on Computer and Communications Security (CCS 2016) (2016)
12. Jansen, R., Tschorsch, F., Johnson, A., Scheuermann, B.: The sniper attack: Anonymously deanonymizing and disabling the Tor network. In: Proceedings of the Network and Distributed Security Symposium - NDSS '14. IEEE (Feb 2014)
13. Kadianakis, G., Perry, M.: Defending against guard discovery attacks using vanguards, (Tor proposal 247). https://gitweb.torproject.org/torspec.git/tree/proposals/247-hs-guard-discovery.txt (2015)
14. Mathewson, N.: Next-generation hidden services in Tor (Tor proposal 224). https://gitweb.torproject.org/torspec.git/tree/proposals/224-rend-spec-ng.txt
15. Muffett, A.: How to get a company or organisation to implement an onion site, i.e. a Tor hidden service. https://www.facebook.com/notes/alec-muffett/how-to-get-a-company-or-organisation-to-implement-an-onion-site-ie-a-tor-hidden-/10153762090530962 (October 2015)
16. Muffett, A.: 1 million people use Facebook over Tor. https://www.facebook.com/notes/facebook-over-tor/1-million-people-use-facebook-over-tor/865624066877648 (April 2016)
17. Onion service traffic metrics site. https://metrics.torproject.org/hidserv-rend-relayed-cells.html
18. Onionshare. https://onionshare.org/
19. Øverlier, L., Syverson, P.: Locating hidden servers. In: 2006 IEEE Symposium on Security and Privacy (S& P 2006), Proceedings. pp. 100–114. IEEE CS (May 2006)
20. Øverlier, L., Syverson, P.: Valet services: Improving hidden servers with a personal touch. In: Danezis, G., Golle, P. (eds.) Privacy Enhancing Technologies: 6th International Workshop, PETS 2006. pp. 223–244. Springer-Verlag, LNCS 4258 (June 2006)
21. Reed, M.G., Syverson, P.F., Goldschlag, D.M.: Proxies for anonymous routing. In: Twelfth Annual Computer Security Applications Conference. pp. 95–104. IEEE CS Press (1996)
22. Reed, M.G., Syverson, P.F., Goldschlag, D.M.: Protocols using anonymous connections: Mobile applications. In: Security Protocol: 5th International Workshop, Paris France, April 7-9 1997, Proceedings. pp. 13–23. Springer-Verlag, LNCS 1361 (1998)
23. Ricochet. https://ricochet.im/
24. Shackleton, W.: Improved sharing of .onion links on Facebook. https://www.facebook.com/notes/facebook-over-tor/improved-sharing-of-onion-links-on-facebook/1196217037151681/ (2017)
25. Tor: Hidden Services Protocol. https://www.torproject.org/docs/hidden-services.html.en

26. Tor network size. `https://metrics.torproject.org/networksize.html`
27. Tor Rendezvous Specification. `https://gitweb.torproject.org/torspec.git/tree/rend-spec.txt`
28. The Tor Project. `https://www.torproject.org/`
29. Victors, J., Li, M., Fu, X.: The onion name system: Tor-powered decentralized DNS for Tor onion services. Proceedings on Privacy Enhancing Technologies 2017(1), 21–41 (2017)
30. Waldmen, M., Rubin, A.D., Cranor, L.F.: Publius: A robust, tamper-evident, censorship-resistant web publishing system. In: Proceedings of the 9th USENIX Security Symposium (August 2000)