Poster: Improving the Performance and Security of Tor's Onion Services

Arushi Arora
Purdue University
arora105@purdue.edu

Sai Raj Karra
Purdue University
karra0@purdue.edu

Dave Levin
University of Maryland
dml@cs.umd.edu

Christina Garman
Purdue University
clg@cs.purdue.edu

***Abstract*—Tor is one of the most widely used anonymous communication networks today. A popular feature of Tor is its onion services, anonymous network services that can only be accessed via the Tor network. This enables users to both host and access such services anonymously, protecting onion services from censorship and take-down. According to Tor Metrics, over 150,000 onion services collectively serve traffic at a rate of nearly 4 Gbps, with applications ranging from news services to chat to whistleblowing. Unfortunately, onion services also suffer from a variety of performance and security concerns. Latency can be extremely high, and many services face denial of service and deanonymization attacks due to the content and types of services that they host. In this work we seek to help address these concerns *without making any changes to Tor*, thus making our improvements immediately useful and deployable. To do this, we leverage a recent advance in programmable anonymity networks, which allows us to deploy user-written *functions* on willing Tor relays. We use this architecture to design the first Content Delivery Network (CDN) for onion services, `CenTor`, aiming to enhance its usability and security. We aim to implement, deploy, and evaluate `CenTor` on the Tor network, demonstrating how it can extend and improve the capabilities, performance, and defenses of onion services.**

I. Introduction

Tor [2] is the most popular and widely used anonymous communication network today. This low-latency circuit-based, application-level overlay network system, is based on the concept of onion routing [3]. Onion services (formerly called hidden services), on the other hand, allow servers to anonymously *host* content that can be accessed using Tor. These services have supported freedom of speech, expression, and information to its clients living in oppressive regimes and have therefore experienced a sharp increase in their number and amount of traffic relayed. Interactive communication on Tor, which accounts for over 90% of connections in the Tor network, incurs latencies over $5x$ greater than on the direct Internet path [6]. The delay factor further increases when a client attempts to reach an onion service, as then a Tor circuit is utilized on both the client and server-side, which often discourages many Tor users from adopting and utilizing onion services, both as a service operator and a client. In addition, the Tor network, which uses a set of volunteer relays, is also prone to denial of service (DoS) attacks. For example, [4] estimated that congestion attacks against all Tor onion routers could increase the median client download time by 47%. Due to the asymmetrical architecture of the Tor protocol, these DoS attacks can further result in the unavailability of an onion service under attack. Furthermore, the anonymity of an onion service's clients makes it harder to detect malicious ones, preventing the deployment of typical defenses that one might employ on the non-anonymous Internet. Onion services also face many threats of deanonymization [5]. We postulate that not only would performance improvements in onion services enrich the user experience, but they would also encourage people to use and own onion services, and further grow the Tor community, thereby enlarging the anonymity set, making the aforementioned attacks much more difficult. Unfortunately, no prior work has managed to extend this to onion services. In general, extending such architectures to onion services persists to be a hard problem due to their anonymous nature. In other words, any modifications proposed for the onion services should not interfere with a service's and its respective clients' anonymity. Moreover, existing implementations of such an architecture require modifications to the Tor code.

In this work, we present the first CDN for onion services `CenTor`, aiming to enhance the usability and security of Tor's onion services, benefiting both an onion service's clients as well as the operators. In addition, we aim to do this without *any* changes to the underlying Tor protocol or architecture. We, therefore, seek to answer the following questions: *What performance and security improvements can an onion service client securely obtain today, without any modifications to Tor?; How can onion services operate with a reduced risk of DoS and deanonymization attacks?* We answer these questions by leveraging the power of `Bento`, a recently introduced architecture for achieving the features of programmable networks in Tor [7]. Tor relays can opt into acting as `Bento` servers, allowing users to safely and securely run *functions* (executable user-written code in a high-level language) on them. We design `CenTor` for Tor onion services, providing enhanced resilience, security, and reduced latency to clients. We implement `CenTor` as a sophisticated function for `Bento` which an onion service operator can install and run on the `Bento` servers, without changing the Tor architecture. We also aim to design, implement and evaluate the client-side performance improvements of a set of functions for onion services and further *compose* them with `CenTor`. We, therefore, build over Tor without "touching" the Tor source code. In addition to security and performance benefits, we also provide users of onion services with new flexibility to tailor their anonymity, bandwidth overhead, and latency preferences. While the notion of the anonymity trilemma [1] states that anonymous communication protocols can only achieve two out of the three properties (strong anonymity, low bandwidth overhead, and low latency overhead), the techniques we present in this paper allow individual users and/or onion service providers a new degree of flexibility to trade-off in this space themselves.

II. Overview: CenTor

We briefly introduce CenTor through a motivating example, discuss how it can benefit onion services, and provide an overview of our anonymity analysis.

Motivating Example. An onion service operator, Alice, wishes to scale and deploy her onion service on CenTor. She also wants to enrich her service's user experience by reducing the latency experienced by her clients and, therefore, hopes to grow the community. Additionally, Alice fears adversaries who aim to deanonymize her onion service or DDoS it.

Choosing Replicas. Alice first identifies Bento servers for every *shadow* (a geographical region), to employ them as a replica for her onion service. She does so by accessing a database of Bento servers showing the availability of these relays per *shadow*.

Deployment. Next, Alice sends the content of her service to the selected node(s) that will act as replicas. After receiving the required contents, the node(s) deploys the onion service in a non-anonymous fashion (which, therefore, reduces the number of hops) and return an alias *.onion* address to the operator. The replica nodes send the *.onion* address to Alice who can then publicize them along with their respective shadow.

Accessing the Replica. Bob, a user who wishes to access Alice's onion service (without knowing who or where it is), witnesses a list of Alice's onion service's replicas and visits the one which is 'local' to him, or in other words geographically lies in the same *shadow* as him. Bob can do so by running the *shadow*-specific client script to connect to Alice's onion service. This script ensures that Bob browses Alice's onion service through a (client-side) Tor circuit which consists of relays that are in the same *shadow* as decided by him.

Composing Functions. Alice can further improve the performance of her onion service by composing CenTor with the LoadBalance function (described in [7]), which can further strengthen DDoS resilience. Moreover, LoadBalance will allow Alice's onion service to have multiple replicas (these share the same hostname and private key) within the same *shadow* which can scale up and down based on client traffic.

Why CenTor? CenTor aims to reduce the geographical distance between the client and the onion service it wishes to access. Moreover, we argue that since revealing the location of the replica does not deanonymize the onion service, its replicas can act as *non-anonymous* (enabling direct connections in the server-side onion service protocol). This approach implies a reduction in the number of hops since the replica-side Tor circuit is no more required, thereby improving client-side performance and further reducing the latency. We also argue that an onion service, after setting up CenTor, can go offline. Now, consider an adversary who tries to deanonymize this onion service. This adversary would now be attacking a replica, instead of the actual onion service itself. Even if the location of this replica under attack is revealed, no harm to the onion service or its operator is caused, since the replicas operate independently of it. Moreover, if the onion service operator chooses its replicas to be *non-anonymous*, this, therefore, changes the way the client interacts with the onion service. Since, there is no replica-side circuit, the probability of carrying out a successful circuit fingerprinting attack is reduced. CenTor enables an onion service operator to have replicas which can be *composed* with the LoadBalance function, thereby distributing the load. Moreover, if the onion service or one of its replicas detects malicious traffic or a DoS threat, it can simply shut down the replica and can spin up new ones. (Detecting malicious traffic can be, and in practice often is, achieved with a programmable middlebox function, as well.) In addition, Bento allows a client to use a combination of multiple *functions* for best results.

TABLE I. Description of Anonymity Function Parameters.

Symbol	Description
l	Client's Autonomous System (AS) [9].
D_l	Client density in location l.
$CD(S)$	Total client density in set of location(s) in shadow S. It is equal to $\Sigma_{l \in S} D_l$.
$RD(S)$	Total Tor relay density in shadow S.
$XD(S)$	Total Tor Exit relay density in shadow S.
$ED(S)$	Total Tor Guard relay density in shadow S.
$EL(S)$	Entropy of distribution of clients across locations in a *shadow*. It is calculated as $-\Sigma_{l \in S}[D_l/Density(S) * \log_2 D_l/Density(S)]$.
$EC(S)$	Entropy of distribution of clients across countries c in a *shadow*. It is calculated as $-\Sigma_{c \in S}[D_c/Density(S) * \log_2 D_c/Density(S)]$.

Performance versus Anonymity. The proposed CenTor architecture operates by accomplishing a subset selection from the total available set of Tor relays, which act as a pool for the circuit nodes for the client. This idea of selecting a *shadow* by the client possesses a performance versus anonymity trade-off. To concretely evaluate the anonymity (and anonymity trade-offs) provided by CenTor, we introduce an anonymity function α_{CenTor} (based on [8]). We further compute $\alpha_{CenTor}(S)$, which is the anonymity score of a specific *shadow* S. This score helps a client decide how much anonymity she is willing to sacrifice for performance gains, allowing her to select and utilize a *shadow* size (to access an onion service replica) which is directly proportional to α_{CenTor}. We define $\alpha_{CenTor}(S) = (CD(S), EL(S), EC(S), RD(S), XD(S), ED(S))$ and describe these parameters in Table I. Also, we suggest clients be mindful of how much anonymity they are compromising while using CenTor.

References

[1] Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate. Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency-choose two. In *IEEE Symposium on Security and Privacy*, 2018.

[2] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, 2004.

[3] David Goldschlag, Michael Reed, and Paul Syverson. Onion routing. *Communications of the ACM*, 42(2):39–41, 1999.

[4] Rob Jansen, Tavish Vaidya, and Micah Sherr. Point break: a study of bandwidth denial-of-service attacks against tor. In *USENIX Security Symposium*, 2019.

[5] Ishan Karunanayake, Nadeem Ahmed, Robert Malaney, Rafiqul Islam, and Sanjay Jha. Anonymity with tor: A survey on tor attacks. *arXiv preprint arXiv:2009.13018*, 2020.

[6] Damon McCoy, Kevin Bauer, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Shining light in dark places: Understanding the tor network. In *Privacy Enhancing Technologies Symposium (PETS)*, 2008.

[7] Michael Reininger, Arushi Arora, Stephen Herwig, Nicholas Francino, Jayson Hurst, Christina Garman, and Dave Levin. Bento: Safely bringing network function virtualization to tor. In *ACM SIGCOMM*, 2021.

[8] Florentin Rochet, Ryan Wails, Aaron Johnson, Prateek Mittal, and Olivier Pereira. Claps: Client-location-aware path selection in tor. In *ACM Conference on Computer and Communications Security (CCS)*, 2020.

[9] Yixin Sun, Anne Edmundson, Nick Feamster, Mung Chiang, and Prateek Mittal. Counter-raptor: Safeguarding tor against active routing attacks. In *IEEE Symposium on Security and Privacy*, 2017.

Poster: Improving the Performance and Security of Tor's Onion Services

Arushi Arora
Purdue University
arora105@purdue.edu

Sai Raj Karra
Purdue University
karra0@purdue.edu

Dave Levin
University of Maryland
dml@cs.umd.edu

Christina Garman
Purdue Univxrsity
clg@purdue.edu

Tor's Onion Services

Tor is one of the most widely used anonymous communication networks today. Onion services, a popular feature of Tor, can only be accessed via the Tor network, enabling users to host & access such services anonymously, protecting them from censorship.

Challenges

Onion services also suffer from a variety of performance and security concerns. Latency can be extremely high, and many services face denial of service and deanonymization attacks due to the content and types of services that they host.

Contributions

We seek to help address these concerns without making any changes to Tor, thus making our improvements immediately useful and deployable. We leverage a recent advance in programmable anonymity networks (`Bento`), which allows us to deploy user-written *functions* on willing Tor relays. In this work, we present the first CDN for onion services `CenTor`, aiming to enhance the usability and security of Tor's onion services, benefiting an onion service's clients as well as its operators. In addition, we also provide users of onion services with new flexibility to tailor their anonymity, bandwidth overhead, and latency preferences.

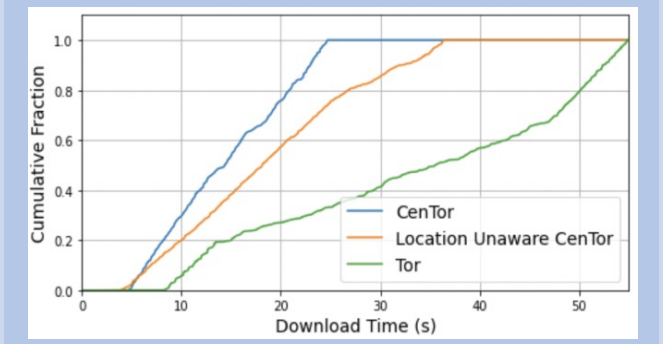

Fig 1: Performance comparison: `CenTor`, location unaware `CenTor` & standard Tor.

CenTor: A Motivating Example

Alice, wishes to scale and deploy her onion service on `CenTor`.

Choosing Replicas. Alice first identifies `Bento` servers for every *shadow* (a geographical region), to employ them as a replica for her onion service.

Deployment. Alice sends the content of her service to the selected node(s) that will act as replicas. After receiving the required contents, the node(s) deploys the onion service in a non-anonymous fashion (reducing the number of hops) and return an alias *.onion* address to the operator. The replica nodes send the *.onion* address to Alice who can then publicize them along with their *shadow*.

Accessing the Replica. Bob, a user who wishes to access Alice's onion service (without knowing who or where it is), witnesses a list of Alice's onion service's replicas and visits the one which is *local* to him (geographically lies in the same *shadow* as him). Bob can do so by running the *shadow*-specific client script to connect to Alice's onion service. This script ensures that Bob browses Alice's onion service through a (client-side) Tor circuit which consists of relays that are in the same *shadow* as decided by him. Alice can further improve the performance of her onion service by *composing* `CenTor` with other functions.

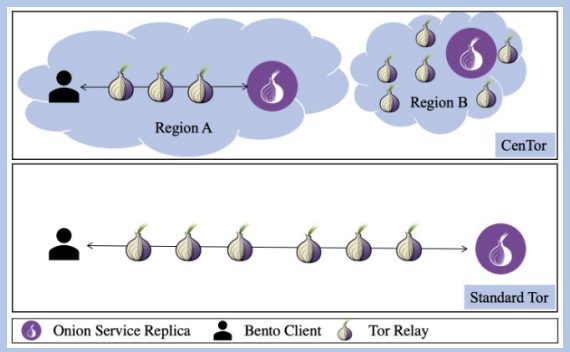

Fig 2: Communication using a `CenTor` versus standard Tor.

Performance versus Anonymity

The proposed `CenTor` architecture operates by accomplishing a subset selection from the total available set of Tor relays, which act as a pool for the circuit nodes for the client. This idea of selecting a *shadow* by the client possesses a performance versus anonymity trade-off. To concretely evaluate the anonymity (and anonymity trade-offs) provided by `CenTor`, we introduce an anonymity function $\alpha_{\texttt{CenTor}}$. We further compute $\alpha_{\texttt{CenTor}}(\texttt{S})$, which is the anonymity score of a specific shadow `S`. We define $\alpha_{\texttt{CenTor}}(\texttt{S})=(\texttt{CD(S)},\texttt{EL(S)},\texttt{EC(S)},\texttt{RD(S)},\texttt{XD(S)},\texttt{ED(S)})$ and describe these parameters in Table 1. From the values computed in table 2, we can say that $\alpha_{\texttt{CenTor}}(\texttt{Eurasia})$ provides better anonymity than $\alpha_{\texttt{CenTor}}(\texttt{APAC})$.

Table 1: Description of Anonymity Function Parameters.

Symbol	Description
l	Client's Autonomous System (AS) .
D_l	Client density in location l.
$CD(S)$	Total client density in set of location(s) in shadow S. It is equal to $\Sigma_{l \in S} D_l$.
$RD(S)$	Total Tor relay density in shadow S.
$XD(S)$	Total Tor Exit relay density in shadow S.
$ED(S)$	Total Tor Guard relay density in shadow S.
$EL(S)$	Entropy of distribution of clients across locations in a *shadow*. It is calculated as $-\Sigma_{l \in S}[D_l/Density(S) * \log_2 D_l/Density(S)]$.
$EC(S)$	Entropy of distribution of clients across countries c in a *shadow*. It is calculated as $-\Sigma_{c \in S}[D_c/Density(S) * \log_2 D_c/Density(S)]$.

Table 2: Approximate $\alpha_{\texttt{CenTor}}$ scores for various shadows.

Shadow	*EL(S)*	*EC(S)*	*CD(S)*	*ED(S)*	*XD(S)*	*RD(S)*
Africa	0.665	0.794	0.013	0.000	0.000	0.002
Africa & Europe	0.694	0.630	0.502	0.737	0.682	0.673
Western Europe	0.511	0.607	0.168	0.517	0.576	0.464
Central & Eastern Europe	0.526	0.527	0.129	0.337	0.231	0.318
Eurasia	0.839	0.710	0.601	0.742	0.695	0.713
Europe	0.649	0.602	0.690	0.737	0.682	0.670
APAC	0.645	0.548	0.264	0.032	0.026	0.084
America	0.530	0.324	0.295	0.055	0.020	0.059
America & Western Europe	0.547	0.517	0.464	0.573	0.597	0.523