

Christoph Döpmann; Sebastian Rust; Florian Tschorsch

Exploring Deployment Strategies for the Tor Network

Conference paper | Accepted manuscript (Postprint)

This version is available at <https://doi.org/10.14279/depositonce-8378>



Döpmann, Christoph; Rust, Sebastian; Tschorsch, Florian (2018). Exploring Deployment Strategies for the Tor Network. 2018 IEEE 43rd Conference on Local Computer Networks (LCN).
<https://doi.org/10.1109/lcn.2018.8638043>

Terms of Use

© © 2018 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Exploring Deployment Strategies for the Tor Network

Christoph Döpmann
Distributed Security Infrastructures
Technische Universität Berlin
christoph.doepmann@campus.tu-berlin.de

Sebastian Rust
Institut für Informatik
Humboldt-Universität zu Berlin
rustseba@informatik.hu-berlin.de

Florian Tschorsch
Distributed Security Infrastructures
Technische Universität Berlin
florian.tschorsch@tu-berlin.de

Abstract—In response to upcoming performance and security challenges of anonymity networks like Tor, it will be of crucial importance to be able to develop and deploy performance improvements and state-of-the-art countermeasures. In this paper, we therefore explore different deployment strategies and review their applicability, impact, and risks to the Tor network. In a simulation-based evaluation, which leverages historical data of Tor, we show that the deployment strategies can practically be applied to realize significant protocol changes in Tor. Our results, however, also indicate that during the transitional phase a certain degradation of anonymity is unavoidable.

Keywords—Internet security, overlay networks

I. INTRODUCTION

Anonymous communication networks (ACNs) such as Tor [1] face a series of challenges, including performance and security challenges. In order to keep up with current and future requirements, it is of decisive importance to carry out active research in this field. Great technological advances will fail to reach practical importance, though, if they cannot actually be deployed in a safe and practical manner. Yet, this is an aspect that is oftentimes neglected in research.

With ACNs in general, deployment of system changes becomes an especially difficult task due to their inherent system properties, i. e., preserving users' anonymity and network access. ACNs are oftentimes highly decentralized systems, usually with many autonomous entities. Incremental upgrades might easily result in a network split or otherwise harm the anonymity set that is essential to protect its users.

The main focus of this paper lies on the deployment of changes that affect Tor's network communication or infrastructure. Examples that may be desirable to adopt in the future include fundamental changes to connection handling [2], congestion control [3], or even the transport protocol [4], [5]. Such protocol changes can have tremendous, inherently backwards-incompatible consequences.

In this paper, we first investigate how *flag day transitions* can be applied in the context of Tor. Pointing at the deficiencies associated with this approach, we also consider coexistence-based approaches like *dual stack*, *translation* and *tunneling*, analyzing their strengths and drawbacks and relating them to real proposed network changes. We validate their applicability by carrying out a simulation-based evaluation. An extended version of this paper, with additional results, is available in [6]. Our work indicates that certain network changes possibly

cannot be deployed without some degradation of anonymity during the transitional phase.

Our contributions include: After reviewing related work (Section II), we give an overview of deployment challenges in Tor and identify existing mechanisms that can be leveraged to deploy small, compatible changes (Section III). We investigate strategies suitable for deploying more fundamental changes to the Tor protocol and juxtapose their advantages and individual risks regarding security and anonymity (Section IV). Furthermore, we evaluate the proposed approaches' impact on the network (Section V) and emphasize in our conclusion the importance of carefully choosing an appropriate deployment strategy during development (Section VI).

II. RELATED WORK

The deployment of network protocols has often enough proven to be cumbersome. As a prime example, we refer to the still ongoing transition from IPv4 to IPv6, which, to some extent, inspired the strategies proposed in this work. However, we focus on anonymous communication networks in general and the Tor network in particular, which induce additional requirements regarding security and anonymity.

In the context of Tor, [7] constitutes a concise survey of proposed extensions to the Tor protocol. It provides a useful overview of what kinds of changes we have to take into account when constructing a deployment scheme for Tor.

The challenge of deploying changes to the Tor network has barely been covered by previous research, though. [8] reports on challenges of deploying a low-latency anonymity network, but mostly deals with general challenges, including the social challenges, and not specifically to changing an *existing* system. To the best of our knowledge, [9] is the only work that covers how a specific incremental update to the Tor software was realized. In contrast, our work has a strong focus on the technical point of view of deployment strategies and strives to explore generic solutions to realize complex protocol evolution in Tor.

III. THE TOR ECOSYSTEM

We give a concise overview of Tor, including its components and the network's current ability to deploy new features.

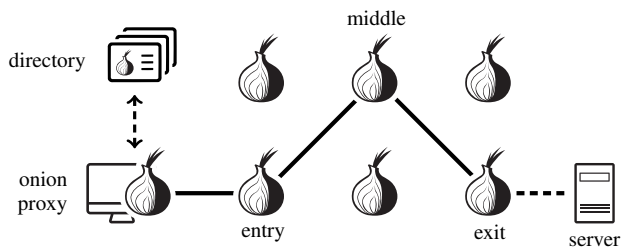


Fig. 1. The Tor network. Each onion symbol represents a relay. The list of all relays is provided by the directory. Clients typically select three relays to construct a circuit. Onion routing ensures anonymity of the transmitted data.

Onion Routing: Tor is based on the concept of onion routing [10]. The main idea consists in building a multi-hop tunnel that is used to carry the payload over a series of intermediate onion routers, casually referred to as *relays*. In Tor, such a cryptographically secured tunnel, which is called a *circuit*, typically consists of three relays as illustrated in Figure 1. A circuit is constructed by extending each hop incrementally, like a telescope, where each relay removes or adds one layer of encryption. The onion routing ensures that each relay in a circuit only knows its immediate predecessor and successor, which eventually provides anonymity. For bootstrapping communication as well as for authenticating relays, Tor relies on a *directory service*, which provides cryptographic keys and other meta information, published by the relays.

Data Transport: Data transport is realized in Tor by passing so-called *cells* along the circuit, carrying payload as well as control signals. Relays multiplex circuits over TLS-secured TCP connections. For anonymity reasons, cells generally have a fixed size. They consist of three parts: the *circuit ID*; the *command* denoting the cell type, thus defining how the cell should be handled; and the *payload*, which contains command-specific data, e. g., parameters. Tor defines numerous different cell types [11]. RELAY cells are particularly interesting because their payload is “onion encrypted”, which means that only the endpoints of a circuit are able to decipher it.

Protocol Versions in Tor: The Tor protocol [11] can be regarded as a suite of different subprotocols that describe different aspects of the Tor network. Among these, the subprotocols that define the network-level communication between relays (Link and Relay) are especially subject to improvements.

Tor provides basic mechanisms to support the evolution of these subprotocols. The subprotocol versions that are supported by each relay are published in the directory data, together with a span of minimum recommended and minimum required supported version. However, Tor relays do not generally rely on this data for normal operation. Instead, the protocol versions are engineered in a way that enables seamless interaction, agreeing on a common version when necessary. For example, at the time of writing, the Link protocol version is chosen by exchanging VERSIONS cells, but was previously determined through subtle, compatible differences in the handshake. In general, Tor offers two ways of rolling out protocol changes in a backwards-compatible

way. Firstly, new cell types and RELAY cell subcommands can easily be introduced. Unknown cell types will be ignored by legacy nodes¹. Secondly, support for specific protocol versions can be signaled through the relay descriptors in the directory.

Participants: Nodes participating in the Tor network take one of two general roles. If they provide access to the network for an end-user, they are commonly called clients. On the other hand, the relays form the backbone of the Tor network. Clients choose relays for constructing their circuits through the network, based on information from the directory.

Moreover, it is important to note that relays can take different roles, too. *Non-exit relays* purely act within the network, carrying data between other relays. In contrast, *exit relays* are the circuits’ endpoint, facing the Internet. The heterogeneity of relays and the different roles make deployments challenging.

IV. DEPLOYMENT STRATEGIES

In this section, we develop general approaches towards deploying changes to the Tor network and relate them to existing Tor research, serving as examples. Communication through Tor constitutes a *multi-hop* scenario, which denotes a major challenge. Far-reaching changes to the Tor protocol may require all relays on a circuit (or even the whole network) to support the new feature. A situation in which some relays can no longer communicate with each other is to be avoided under all circumstances as it would effectively result in a network split, heavily reducing the users’ anonymity set. In the following, we consider two main strategies, flag day and co-existence.

A. Flag Day: Building a Global Switch

The *flag day* strategy consists in incrementally rolling out a software update but keeping it disabled until coordinated, simultaneous activation. This way, any change to the network can be deployed, with no backwards compatibility required. While implementing a reliable flag day is far from trivial, it may have to be considered for large, breaking system changes.

Note, however, that a flag day transition may break *client compatibility*. In contrast to relays, clients do not publish version information. Therefore, we realize that carrying out a flag day without putting an unknown number of end users to the risk of being excluded from the Tor network, is currently infeasible. This could, however, change in the future with the advent of privacy-preserving collection of user statistics [12].

B. Co-existence: Implementing Interoperability

In contrast to the previous all-or-nothing approach, co-existence strategies allow relays of different versions or feature sets to interoperate, which is desirable as it mitigates the risk of a network split. For any circuit, choosing a co-existence scheme may be done either for the whole circuit based on global information from the directory, or locally on a hop-by-hop basis. We here present different approaches that enable backwards-incompatible protocol changes to co-exist with the legacy protocol.

¹We call relays that have not (yet) upgraded legacy nodes.

Dual Stack: For the *dual stack* strategy, we assume that upgraded nodes can still use the legacy protocol selectively. This way, a circuit can make use of the newest supported protocol version that all relays support. As a consequence, single circuits may already benefit from the newly deployed technology, while circuits with legacy relays stick to the previous behavior until upgraded. A dual stack approach can be used to deploy a wide range of complex changes to the network, at the expense of only slowly spreading use of the new feature. We will quantify this delay in our evaluation.

Moreover, this strategy might lead to weakened anonymity. An attacker may be able to deduce from local observations information about other relays in the circuit, reducing the anonymity set. For example, consider the case of replacing TCP with UDP as the underlay transport protocol. By observing which variant is used for a specific circuit, an attacker can infer a set of relays which cannot be part of the circuit.

Candidates for being deployed with a dual stack approach are for example *DefenestraTor* [3] and *BackTap* [5]. Both aim at improving Tor’s congestion control, requiring the interaction of every relay on the circuit in order to work properly.

Translation: If the relays in a circuit differ in their feature sets, *translation* between the protocol versions may be applied. Improvements that do not require the whole circuit to participate can thus partially become active earlier. If compatibility can be achieved by converting between cell types, the translation scheme may be trivial, unless non-local information is required for this translation step. For example, *TCP-over-DTLS* [4], which proposes to use a TCP connection per circuit and to encapsulate the respective TCP segments in UDP, might be deployable via translation.

Tunneling: If multiple non-consecutive relays in a circuit have upgraded, one may also *tunnel* new protocol data through legacy nodes. For instance, information could be carried end-to-end without interaction by intermediate legacy nodes.

Tunneling schemes may leverage Tor’s RELAY cells to exchange data between circuit endpoints. Embedding opaque information into other legacy cells may also be an alternative.

Tunneling could, for instance, be applied to proposals like *UDP-OR* [13] that rely on an end-to-end semantics for improving performance but can be built to tolerate single legacy relays in the circuit.

V. EVALUATION

We now aim at quantifying the impact of each of the basic conceptual ideas we have discussed, generating insight on their individual benefits and drawbacks.

In order to compare against a realistic model of relays’ upgrade behavior, we base our evaluation on historical data of software version distribution within the Tor network. Thus, we make use of data collected by the *Tor Metrics* project [14]. We consider a time span from March 2013 to January 2018, over which we track the share of relays that have upgraded to Tor version 0.2.5 or higher, which was released at the beginning of this time span. We chose to analyze the deployment of this particular Tor version because its predecessor has been

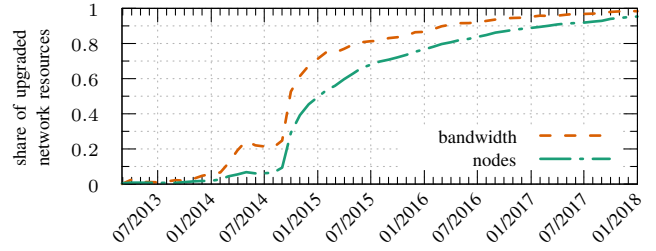


Fig. 2. Share of Tor network resources retained after a flag day, over time.

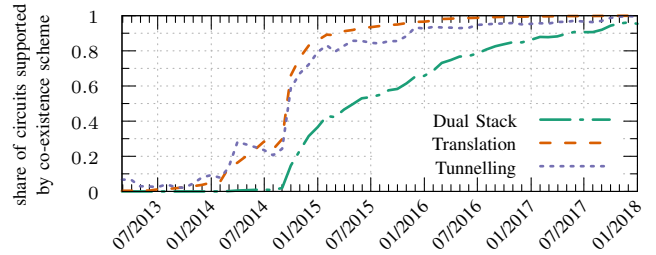


Fig. 3. Share of circuits that support the respective co-existence scheme.

exceptionally prevalent in the Tor network, which virtually constitutes a worst case scenario for deployment. Moreover, we assume the client software to be up to date. The study thus investigates to what degree a modern client can benefit from technological progress as it is rolled out in the Tor network over time.

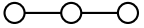
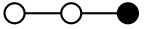


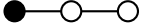
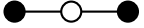


A. Flag day

For analyzing the impact of a flag day strategy, we take a general evaluation approach that allows us to quantify a flag day’s impact independently of a specific protocol change. For any given point in time, we evaluate which network relays have upgraded. These are the relays that would keep working if a flag day was carried out at that time. From this set, we calculate both the *share of relays* and the *share of bandwidth* that are retained (in comparison to the overall network).

Figure 2 shows the results of evaluating the historical data for the introduction of Tor version 0.2.5. As this version was deliberately chosen as a worst case scenario, our results show that the upgrade progress is slow over the network. After one year, less than 10% of the relays have upgraded, accounting for approximately 20% of the network bandwidth. Despite the initially slow upgrade behavior, a clear rise can be observed approximately one and a half years after version 0.2.5 was initially published. The upgrade process continues slowly, the 90% landmark of upgraded relays is reached four years after the initial release (March 2017). Also, note that there is a temporary decline in upgrade progress around April 2014. This is due to the *Heartbleed* vulnerability which required to remove a number of newer vulnerable relays from the network.

These statistics reveal that it can take a long time until a sufficient amount of relays have applied software upgrades. However, for far-reaching changes to the network, a flag day

TABLE I
APPLICABILITY OF CO-EXISTENCE SCHEMES.

Circuit configuration (entry-middle-exit)	Applicable schemes		
	Dual Stack	Translation	Tunneling
	x	x	x
	x	x	✓
	x	x	x
	x	✓	✓
	x	✓	x
	x	✓	✓
	x	✓	x
	✓	✓	✓

● upgraded ○ legacy

may still constitute a conceivable strategy. This, however, does not tackle the issue of client compatibility mentioned before.

B. Co-existence

We now evaluate to what extent deployment schemes based on co-existence enable the use of the novel protocol extension during the transition phase, at which circuits may comprise both, legacy and upgraded relays.

This evaluation requires to operate on a more fine-grained, circuit-level scale. Since the effectiveness of these strategies is highly dependent on the configuration of every single circuit, we simulate circuit selection using the *Tor Path Simulator* (TorPS) [15]. Given the historical Tor network data, it allows us to gather aggregate data on specific properties of historical circuits. In order to measure the applicability of each of the co-existence strategies for given circuit configurations, we assume the proposals we provided as examples in Section IV as mental templates and derive a compatibility matrix, shown in Table I.

Figure 3 presents our results, depicting how well each of the co-existence schemes can enable the partial activation of the protocol change during the transition phase. We note that translation and tunneling are considerably more effective than dual stack because they can apply the protocol change at least to parts of the circuit much earlier. For example, after one and a half years, already 60% of the circuits could have applied a translation or tunneling scheme. At the same time, dual stack would only be used in approximately 10% of the circuits. We conclude that incrementally deploying changes via co-existence schemes can, in fact, ease the deployment process.

VI. LESSONS LEARNED AND CONCLUSION

In this work, we explored the problem of deploying changes to the Tor network. We find that, through its directory service and the extensible cell structure, Tor can in principle be extended to feature novel functionalities. Evolving the protocol at its core networking routines is a particularly challenging task, though. When altering its behavior at the link and transport layer, special care must be taken to maintain maximum possible compatibility with legacy nodes. Otherwise, the consequences on anonymity may be severe.

We firstly evaluate the use of flag days, finding that they may be used for deploying fundamental changes, but are not currently practically applicable in Tor due to various risks and weaknesses, including network splits, Sybil attacks, and abandoning clients.

We then develop a class of co-existence schemes, which allow a new protocol version to co-exist with the legacy system. In this regard, we introduce the notions of dual stack, translation, and tunneling. We generally observe the transitional phase to be rather long in Tor. With our deployment strategies, though, a significant number of users might be able to benefit already from upgrades during that period.

However, our research also suggests that, with currently available techniques, one cannot always realize incremental deployment without partly sacrificing some of Tor’s anonymity promises during the transitional phase. In essence, this is due to the risk of unintentionally making circuits distinguishable.

With our work, we also hope to raise awareness of the issue among the research community, motivating to integrate suitable deployment strategies directly into their Tor research.

ACKNOWLEDGMENT

Christoph Döpman was supported by HU Berlin within the Excellence Initiative of the states and the federal government.

REFERENCES

- [1] R. Dingleline, N. Mathewson, and P. Syverson, “Tor: The second-generation onion router,” in *USENIX Security ’04*, 2004.
- [2] M. AlSabah and I. Goldberg, “PCTCP: per-circuit TCP-over-IPsec transport for anonymous communication overlay networks,” in *CCS ’13*, 2013.
- [3] M. AlSabah, K. Bauer, I. Goldberg, D. Grunwald, D. McCoy, S. Savage, and G. Voelker, “DefenestraTor: Throwing out windows in Tor,” in *PETS ’11*, 2011.
- [4] J. Reardon and I. Goldberg, “Improving Tor using a TCP-over-DTLS tunnel,” in *USENIX Security ’09*, 2009.
- [5] F. Tschorsch and B. Scheuermann, “Mind the gap: Towards a backpressure-based transport protocol for the Tor network,” in *NSDI ’16*, 2016.
- [6] C. Döpman, S. Rust, and F. Tschorsch, “Exploring deployment strategies for the Tor network,” Cryptology ePrint Archive, Report 2018/661, 2018, <https://eprint.iacr.org/2018/661>.
- [7] M. AlSabah and I. Goldberg, “Performance and security improvements for Tor: A survey,” *ACM Computing Surveys (CSUR)*, vol. 49, no. 2, 2016.
- [8] R. Dingleline, N. Mathewson, and P. Syverson, “Challenges in deploying low-latency anonymity,” Tech. Rep., 2005.
- [9] R. Jansen and M. Traudt, “Tor’s been KIST: A case study of transitioning Tor research to practice,” *arXiv e-print arXiv:1709.01044*, 2017. [Online]. Available: <http://arxiv.org/abs/1709.01044>
- [10] D. M. Goldschlag, M. G. Reed, and P. F. Syverson, “Hiding Routing Information,” in *IHW ’01*.
- [11] R. Dingleline and N. Mathewson, “Tor protocol specification.” [Online]. Available: <https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt>
- [12] N. Mathewson, T. Wilson-Brown, and A. Johnson, “Tor proposal 288: Privacy-preserving statistics with Privcount in Tor (Shamir version),” 2017. [Online]. Available: <https://gitweb.torproject.org/torspec.git/tree/proposals/288-privcount-with-shamir.txt>
- [13] C. Viecco, “UDP-OR: A fair onion transport design,” in *HotPETS ’08*, 2008.
- [14] The Tor Project, “Tor Metrics Portal.” [Online]. Available: <https://metrics.torproject.org/>
- [15] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. F. Syverson, “Users get routed: traffic correlation on tor by realistic adversaries,” in *CCS ’13*, 2013.