Issues in Supporting Third-Partys In-Network Services in the Internet

A. Silvestro[†‡*], R. Bifulco[†], S. Sharma[†], F. Schneider[†], J. Kangasharju[⊥], X. Fu[‡]
NEC Laboratories Europe, Germany[†], University of Helsinki[⊥], University of Goettingen[‡]
Email: Alessio.Silvestro@neclab.eu[*]

***Abstract*—In-network services are important building blocks for today's network applications (e.g., CDNs, antiviruses, proxies, etc.). However, current solutions ignore to a large extent their presence forcing ISPs and service providers in implementing several workarounds which negatively impact services' security and reliability. This paper presents an early stage Ph.D. project which aims to highlight current solutions inefficiencies, identifying a few areas which need further investigation and that will be addressed in our future work.**

I. Introduction

In-network services are important building blocks for today's network applications [5]. Content Distribution Networks (CDNs) [15], antiviruses [10], privacy protecting proxies [16], performance enhancers [19], are just a few examples of such services [18]. Despite this state of affair, the Internet's architecture and protocols ignore to a large extent the presence of in-network services, forcing network and service providers in implementing workarounds [1], [8] that negatively impact services' security and reliability [7]. The issue is particularly relevant if one considers that such in-network services are actually provided by a third party. For instance, when a user connects to her bank website, she may be actually connecting to a front-end CDN, which is provided by a third party the bank employs while the user is unaware of it [3]. Recent studies on cryptographic private key sharing and certificate bundling give an idea of the extent to which this issue is hitting everyday's service provisioning practices [11]. Addressing the problem in a systematic way is required to open a new market for the provisioning of in-network services [9].

A. Related work

Previous work focused on introducing a few new properties to the Internet's architecture, such as *in-network service visibility* and *end-points control* [14], [20]. The former guarantees that an in-network service is not hidden to any of the communication's end-points. The latter requires the network to enable an end-point in deciding whether to use an in-network service or not, when connecting to another end-point. Still, the mentioned works assume that the node that provisions a given service, i.e., the middlebox, is known in advance. However, we notice that each in-network service is usually implemented using several middleboxes, typically deployed in different locations [2]. The selection of a specific middlebox for the provisioning of a given service is performed by the in-network service provider, which typically takes into account a number of variables, including system and network loads, end-point locations, local regulation constraints, etc. [12] To solve a similar issue, services deployed at end-points employ solutions such as DNS redirection [4] and IP anycast [6].

B. Gaps

We argue that those solutions are not applicable in the context of in-network services and point out two significant differences to support our position. First, for end-point's services there is an assumption that the user will contact only one of such services in the context of a single network communication. In practice, his means that there will be, e.g., the need to perform just one DNS name resolution per each end-to-end network communication. In contrast, in presence of intermediaries there could be an arbitrary number of in-network services, introduced by different parties, e.g., content and service providers, enterprises or even regular end-users. Therefore, the mechanisms used for the handling of a single service, e.g., DNS redirection, may not be suitable anymore. In fact, they may not provide properties such as in-network service visibility, or they could introduce important overheads on critical performance metrics such as the connection establishment time [21]. Second, the mentioned mechanisms optimize the serving node selection using metrics such as the expected delay from the client (IP anycast) [2], sometimes taking into account more complex variables such as load level of the back-end infrastructure (DNS redirection) [13]. In presence of multiple in-network services, the current mechanisms may provide not optimized serving nodes (i.e., middleboxes) selection. In particular, at each middlebox, a new "next-hop selection" would happen independently from the selections performed at the other middleboxes. In fact, each in-network service would perform its selection independently from the other services and would optimize for the local optimum, which may be very far from the global one. The remainder of this paper will present a strawman solution to the problem and highlight its issues, justifying the need for performing additional research in such direction. To this end, we conclude presenting open issues and our planned future work.

II. A strawman solution

To describe a concrete case, we use mcTLS for establishing an end-to-end communication through a number of middleboxes. In short, mcTLS requires a client to build a list of middleboxes' IP addresses before establishing the communication with a server. Then, the client establishes a TCP connection with the first middlebox in the list and sends

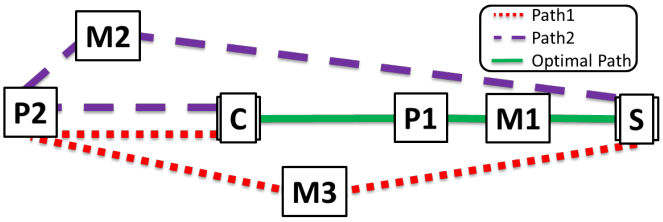

Fig. 1: Example scenario. The user C connects to the bank S using two in-network services. A parental control service implemented by the middleboxes P_1 and P_2. A CDN service implemented by the middleboxes M_1, M_2 and M_3.

a *ClientHello*. Such message contains, among other things, the list of middleboxes' IP addresses that should be used in the communication. In turn, each middlebox will establish a connection to the next middlebox in the list, until the server closes the chain. Notice that mcTLS does not define (i) how the client obtains the middleboxes' IP addresses, and (ii) how the server includes its in-network services' middleboxes in the list. In fact, mcTLS assumes DNS is used for that (cfr Sec.6.1 of [14]).

Assume now a scenario in which a user connects to her bank's web site (Fig. 1). The user subscribed to a parental control in-network service, which is implemented by a proxy running in, e.g., a cloud datacenter. In particular, the parental control service uses two proxies deployed at different locations, i.e., P_1 and P_2. In this scenario, the two proxies are at about the same distance from the user. However, P_1 is also close to the bank, instead P_2 and the bank are far from each other. The bank subscribed to a CDN service. The CDN has a number of middleboxes distributed in the network: M_3 is the closest to the user, while M_1 and M_2 are the closest to P_1 and P_2, respectively.

Our strawman solution uses DNS to discover the IP addresses of the middleboxes that should be used when the user establishes a connection with the bank. In such a case, the user introduces the parental control service by configuring a proxy on her client. The proxy is specified with a domain name such as *parental-ctr.com*. Likewise, the bank introduces the CDN service by associating the bank's domain name, e.g., *bank.com*, with the CDN service domain, e.g., *cdn.com*. The DNS mapping for parental-ctr.com and cdn.com to IP addresses is dynamic, since the serving middlebox is selected to be the closest (in terms of delay) to the requesting client. Thus, when the user wants to establish a connection to the bank, she will first resolve the bank's domain name with a DNS query. Since the bank is using the CDN service, the name will be finally resolved to an IP address of a CDN's middlebox that is close the user's location, e.g., M_3. Then, the user will perform a second DNS query to resolve the parental control proxy's domain name, obtaining the IP address of, e.g., P_2. In this case, the user will first establish a connection to P_2, which in turn establishes a connection to M_3 (Fig. 1, Path1). Unfortunately, M_3 is a CDN's middlebox far from P_2, yielding sub-optimal performance. A more efficient solution would require a modification to mcTLS to include a list of domain names in the ClientHello message, instead of using IP addresses. With this modification each middlebox would be required to perform a DNS query. Thus, once P_2 receives the ClientHello message from the user, it will in turn perform a DNS resolution. This time, P_2 will then connect to the closer M_2 (Fig. 1, Path2). However, also in this case the solution is suboptimal, since P_2 is the farthest of the proxies from the bank. In this case, using P_1 and M_1 as middleboxes would have been the optimal solution (Fig. 1, Optimal Path). Furthermore, in both cases, the need to perform a DNS query for each in-network service may negatively impact the connection establishment time, which is a critical metric of today's network services [17].

III. Future work

In light of the detailed examples, in this section we identify a few areas that require further investigation and that will be addressed in our future work. Our main observation is that the problem is related to the uncoordinated decisions performed by different parties. For instance, in the previous example both the parental control and CDN services run their own service's location selection processes, while a coordinated placement of the middleboxes may provide better performance [19]. Unfortunately, while in a single-party case decisions are easier to coordinate [12], this is not the case when multiple parties are involved. First, different parties may not be aware of each other or may not be willing to exchange information among them. Here, notice that it is not necessarily the case that a party who contributes to the construction of an optimized solution would also benefit from such a solution. Second, assuming that parties agree on collaborating for building an optimal solution, it is unclear what is the minimal amount of information they should share. Third, once there is enough information to compute an optimal solution, such task should be performed efficiently, to cope with the stringent performance requirements of modern network applications. Here, notice this is a difficult problem in which both information dissemination and collection, as well as solution computation, should be quick enough to not impact communications' performance. Finally, a system that solves the previous challenges should be ideally deployable incrementally in the current network architecture.

Acknowledgment

This research work has been partly funded by the EU in the context of the "FP7 ITN CleanSky" project (Grant Agreement: PITN-GA-2013-607584).

References

[1] C. Boulton, J. Rosenberg, G. Camarillo, and F. Audet. NAT Traversal Practices for Client-Server SIP. RFC 6314 (Informational), July 2011.

[2] Matt Calder, Ashley Flavel, Ethan Katz-Bassett, Ratul Mahajan, and Jitendra Padhye. Analyzing the performance of an anycast cdn. In *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*, IMC '15, pages 531–537, New York, NY, USA, 2015. ACM.

[3] Thomas Callahan, Mark Allman, and Michael Rabinovich. On modern dns behavior and properties. *SIGCOMM Comput. Commun. Rev.*, 43(3):7–15, July 2013.

[4] Fangfei Chen, Ramesh K. Sitaraman, and Marcelo Torres. End-user mapping: Next generation request routing for content delivery. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 167–181, New York, NY, USA, 2015. ACM.

[5] Yingying Chen, Sourabh Jain, Vijay Kumar Adhikari, and Zhi-Li Zhang. Characterizing roles of front-end servers in end-to-end performance of dynamic content distribution. In *Proceedings of the 2011 ACM SIGCOMM Conference on Internet Measurement Conference*, IMC '11, pages 559–568, New York, NY, USA, 2011. ACM.

[6] Danilo Cicalese, Jordan Auge, Diana Joumblatt, Tim ur Friedman, and Dario Rossi. Characterizing ipv4 anycast adoption and deployment. In *ACM CoNEXT*, Heidelberg, DE, 12/2015 2015. ACM, ACM.

[7] Xavier de Carné de Carnavalet and Mohammad Mannan. Killed by proxy: Analyzing client-end tls interception software. In *Network and Distributed System Security Symposium (NDSS 2016), San Diego, CA, USA*, 2016.

[8] S. Loreto et al. Explicit trusted proxy in http/2.0. http://goo.gl/BUxQ22, 2014.

[9] Benjamin Frank, Ingmar Poese, Yin Lin, Georgios Smaragdakis, Anja Feldmann, Bruce Maggs, Jannis Rake, Steve Uhlig, and Rick Weber. Pushing CDN-ISP collaboration to the limit. *SIGCOMM Comput. Commun. Rev.*, 43(3):34–44, July 2013.

[10] Avast Software Inc. Avast 2016: Https scanning in web shield - faqs. https://www.avast.com/it-it/faq.php?article=AVKB190, 2016.

[11] J. Liang, J. Jiang, H. Duan, K. Li, T. Wan, and J. Wu. When https meets cdn: A case of authentication in delegated service. In *2014 IEEE Symposium on Security and Privacy*, pages 67–82, May 2014.

[12] Hongqiang Harry Liu, Raajay Viswanathan, Matt Calder, Aditya Akella, Ratul Mahajan, Jitendra Padhye, and Ming Zhang. Efficiently delivering online services over integrated infrastructure. In *Proceedings of the 13th Usenix Conference on Networked Systems Design and Implementation*, NSDI'16, pages 77–90, Berkeley, CA, USA, 2016. USENIX Association.

[13] Bruce M. Maggs and Ramesh K. Sitaraman. Algorithmic nuggets in content delivery. *SIGCOMM Comput. Commun. Rev.*, 45(3):52–66, July 2015.

[14] David Naylor, Kyle Schomp, Matteo Varvello, Ilias Leontiadis, Jeremy Blackburn, Diego R. López, Konstantina Papagiannaki, Pablo Rodriguez Rodriguez, and Peter Steenkiste. Multi-context tls (mcTLS): Enabling secure in-network functionality in tls. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 199–212, New York, NY, USA, 2015. ACM.

[15] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun. The akamai network: A platform for high-performance internet applications. *SIGOPS Oper. Syst. Rev.*, 44(3):2–19, August 2010.

[16] Fotios Papaodyssefs, Costas Iordanou, Jeremy Blackburn, Nikolaos Laoutaris, and Konstantina Papagiannaki. Web identity translator: Behavioral advertising and identity privacy with wit. In *Proceedings of the 14th ACM Workshop on Hot Topics in Networks*, HotNets-XIV, pages 3:1–3:7, New York, NY, USA, 2015. ACM.

[17] Sivasankar Radhakrishnan, Yuchung Cheng, Jerry Chu, Arvind Jain, and Barath Raghavan. Tcp fast open. In *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies*, CoNEXT '11, pages 21:1–21:12, New York, NY, USA, 2011. ACM.

[18] Justine Sherry, Shaddi Hasan, Colin Scott, Arvind Krishnamurthy, Sylvia Ratnasamy, and Vyas Sekar. Making middleboxes someone else's problem: Network processing as a cloud service. *SIGCOMM Comput. Commun. Rev.*, 42(4):13–24, August 2012.

[19] Giuseppe Siracusano, Roberto Bifulco, Simon Kuenzer, Stefano Salsano, Nicola Blefari Melazzi, and Felipe Huici. On the fly tcp acceleration with miniproxy. In *Proceedings of the 2016 Workshop on Hot Topics in Middleboxes and Network Function Virtualization*, HotMIddlebox '16, pages 44–49, New York, NY, USA, 2016. ACM.

[20] Michael Walfish, Jeremy Stribling, Maxwell Krohn, Hari Balakrishnan, Robert Morris, and Scott Shenker. Middleboxes no longer considered harmful. In *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6*, OSDI'04, pages 15–15, Berkeley, CA, USA, 2004. USENIX Association.

[21] Wenxuan Zhou, Qingxi Li, Matthew Caesar, and P. Brighten Godfrey. Asap: A low-latency transport layer. In *Proceedings of the Seventh COnference on Emerging Networking EXperiments and Technologies*, CoNEXT '11, pages 20:1–20:12, New York, NY, USA, 2011. ACM.