Electronic Communications of the EASST
Volume 080 (2021)

Conference on Networked Systems 2021
(NetSys 2021)

Count Me If You Can: Enumerating QUIC Servers Behind Load Balancers

Kashyap Thimmaraju and Björn Scheuermann

5 pages

Guest Editors: Andreas Blenk, Mathias Fischer, Stefan Fischer, Horst Hellbrueck, Oliver Hohlfeld, Andreas Kassler, Koojana Kuladinithi, Winfried Lamersdorf, Olaf Landsiedel, Andreas Timm-Giel, Alexey Vinel
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Count Me If You Can: Enumerating QUIC Servers Behind Load Balancers

Kashyap Thimmaraju and Björn Scheuermann

Department of Computer Science, Humboldt Universität zu Berlin

Abstract: QUIC is a new transport protocol over UDP which recently became an IETF RFC. Our security analysis of the Connection ID mechanism in QUIC reveals that the protocol is underspecified. This allows an attacker to count the number of server instances behind a middlebox, e.g., a load balancer. We found 4/15 (~25%) implementations vulnerable to our enumeration attack. We then concretely describe how an attacker can count the number of instances behind a load balancer that either uses Round Robin or Hashing.

Keywords: QUIC, Security Analysis, Connection ID, Enumeration

1 Introduction

It appears that the next major step in the evolution and wide-spread adoption of a new transport protocol since TCP is QUIC [LRW+17]. QUIC was recently standardized by the IETF [IT21] and is the underlying transport protocol for HTTP/3. There are several stake-holders involved in the standardization: Mozilla, Facebook, Google, CloudFlare, Apple and others.

As we approach the standardization of the protocol (scheduled for 2021), we witness an increase in the number of publicly accessible QUIC servers [QUI20, RPDH18]: Rüth et al. measured an increase in roughly 1 million IPv4 addresses using QUIC in July 2019.

Despite the many benefits QUIC brings to Internet communication and security, e.g., no head-of-line blocking, multiplexing, authenticated and encryption with associated data, and connection migration, it also introduces new security and privacy concerns. In 2019, Sy et al. [SBFF19] pointed out that QUIC servers can identify a user across multiple connections using two features of the protocol and in 2020, Reen et al. [RR20] uncovered critical vulnerabilities in 4 QUIC implementations in addition to describing deep packet inspection elusion strategies.

Motivated by potentially new security and privacy issues introduced by QUIC, in this paper we make the following *contributions*: 1) We conduct a security analysis of the Connection ID (CID) mechanism of the QUIC protocol in 15 different implementations; 2) We uncover unspecified behaviour in the protocol that fundamentally deals with uniquely identifying the source and destination within a QUIC connection using the CID. 3) We describe an attack scenario (see Fig. 1) that exploits the unspecified behaviour and sketch an algorithm against round robin and hash-based load balancers that enables a malicious client to count the number of server instances behind the load balancer.

Paper organization: In the following section we describe the necessary background on QUIC Connection IDs. In Section 3 we describe our threat model followed by our security analysis and evaluation. Next, we describe our enumeration attack and sketch an attack algorithm in Section 4, and then conclude in Section 5.

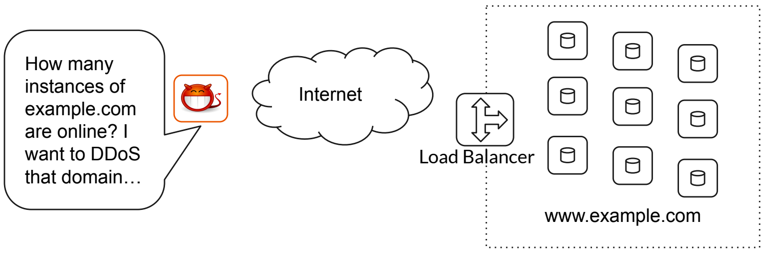
How many instances of example.com are online? I want to DDoS that domain…
Internet
Load Balancer
www.example.com

Figure 1: A high level illustration of the enumeration attack scenario described in this paper. Knowing the number of instances helps to estimate the load required to launch a DDOS attack.

2 QUIC Connection IDs

When a client connects to a server for the first time, in addition to using the IP address and ports, the client generates a (pseudo-random) Connection ID (CID) for itself (source CID) as well as for the server (destination CID) it wishes to connect to. "The primary function of a connection ID is to ensure that changes in addressing at lower protocol layers (UDP, IP) don't cause packets for a QUIC connection to be delivered to the wrong endpoint" [IT21] . These two CIDs are inserted into the QUIC (Long) header in plain-text along with the length of each CID and sent to the server as an *Initial packet*. Upon receiving the Initial packet, the server can either generate a new CID for itself or use the CID that the client generated for the server. Regardless of what it chooses, it then places the CIDs in its response to the client. Once the QUIC handshake is complete, each end-point (i.e., client or server) then only includes the destination CID in the (Short) header. The CID has a maximum length of 20 bytes and follows a variable length encoding scheme.

3 Security Analysis of the Connection ID

We begin by outlining the threat model we adopt. Next, we analyze the QUIC CID mechanism described in the protocol draft [IT21] followed by our evaluation and results of 15 publicly available (closed source and open source) implementations.

Attacker Model. We assume the attacker can actively establish connections with one or more QUIC servers routable via the Internet. This means that she can send any type of QUIC packet which could also be malicious or non-conformant to the protocol. On the server side, we assume that there is typically a load balancing system in place for scaling and availability reasons. For example, multiple IP addresses could resolve the same domain name in addition to using an L3/L4 load balancer to distribute the incoming requests across multiple instances (e.g., virtual machines, containers or bare-metal) of the QUIC server. We assume the load balancers do not use the CID to route connections. Finally, we assume that there are other benign clients attempting to connect to the server.

Analysis. Having set the context for the attacker and server instances, we now describe our security analysis of the CID aspects of the current draft (version 30) of the QUIC protocol. The draft specifies and recommends several important points on secure usage of the CID, e.g.,

i) the same CID MUST NOT be used multiple times over the same connection to prevent an eavesdropper from correlating the end-points; ii) end-points should associate a sequence number with each CID to detect reuse within the same connection; iii) CIDs should be (pseudo) randomly generated as they are also used for packet security and; iv) CIDs can be very long (max. 20 bytes). The draft, however, does not specify how servers should handle the case when successive incoming connections to a server use the same destination CID. Indeed, it can be seen that based on the assumption that the CIDs are (pseudo) randomly generated with a maximum length of 20 bytes, the probability of such a collision is very low. However, based on our threat model, this is not the case. Hence, we posit the following: *If the server (implementation) does not permit the use of the same destination CID across successive connections (for a specific timeout), then an attacker can detect if she has reached the same server instance*. Taking it a step further, she could enumerate the number of server instances behind a domain name, e.g., to prepare for a distributed denial of service (DDoS) attack. Next, we evaluate 15 of the latest implementations to verify if such a scenario is feasible.

Evaluation. The primary goal of our evaluation is to identify implementations, if any, that prevent two subsequent QUIC connections from using the same destination CID. To conduct this experiment we adopted the following methodology. First, we created a client that uses deterministic CIDs for the source and destination CIDs (we modified LiteSpeed Technologie's open-source QUIC client). Second, we used 15 implementations from those listed on the QUIC Working Group's GitHub page[1]: either manually built Docker containers or the public test server[2]. The servers tested are listed in the results. To conduct the experiment, we had the client open and hold a QUIC connection with source CID 1 and destination CID 2 for a maximum of 10s with the server, and then close it. We then idle for 10s and repeat the steps a second time. We saved the debug logs of the client, packet traces and session keys. We then repeated the process for each of the 15 implementations. Finally, to obtain the results of our test, we searched the debug log of the client or the packet trace to confirm whether the second QUIC connection completed a successful handshake or not. The absence of a successful handshake indicates a vulnerable implementation.

Results. We found 4 out of 15 implementations to be vulnerable to our attack scenario, namely, Apache Traffic Server (ATS), Chromium, LiteSpeed and NGTCP2. Akamai, Aioquic, F5, Neqo, Nginx, Picoquic, Proxygen, Quant, Quiche, Quicly and Quinn were not vulnerable.

4 Enumerating Server Instances Behind Load Balancers

In this attack, a malicious client can count the number of online server instances behind a middlebox, e.g., a load balancer that distributes incoming requests across multiple instances. By counting the number of online instances, the attacker can estimate the load required to bring down the targeted service. Furthermore, such information is typically meant to remain private to the service provider. In this paper we focus on two load balancing algorithms: Round Robin (RR) and Hashing. We designed our enumeration algorithm such that it can be used for both algorithms so that the attacker does not need apriori information about which algorithm is used.

[1] https://github.com/quicwg/base-drafts/wiki/Implementations
[2] The evaluation was conducted in October 2020.

Enumeration Algorithm. We repeatedly issue connection requests with a sequentially increasing source port, the CIDs however remain the same for each request. We then count the number C of successful connections established. If we do not receive a response from the server, we have reached an instance that was already counted. If RR is being used, then it could be that other clients' requests interleaved ours, hence, resulting in our request reaching a previously seen instance again instead of an uncounted one. If hashing is being used, then it could be that our 4-tuple values (source IP and port, and destination IP and port) hashed to the same value, and hence the same instance. Therefore, we continue to issue further requests until we do not establish any new connection with the server after a threshold of $max\_requests$ attempts. After which C will be the number of server instances. The advantage of this algorithm is its simplicity. The disadvantage is that the algorithm might not terminate if old CIDs become enabled while the algorithm is still running. This requires investigating how the timeouts behave across the implementations which we consider as future work.

5 Conclusion

In this paper we analyzed the Connection ID mechanism of the current draft (version-30) of the QUIC protocol and discovered the following: If a server does not permit the use of the same destination CID across new connections an attacker can use such behaviour to enumerate the number of server instances behind a load balancer. Our evaluation of 15 implementations revealed that ~25% of implementations are vulnerable to our enumeration attack. Finally, we sketched an enumeration algorithm that can be used against RR and hashing load balancers which can be leveraged by an attacker to extract such private information can be used to estimate the load necessary to launch a DDOS attack against the server. We view this work as motivation for the research community to rethink network security when end points use a secure transport protocol such as QUIC. Defense mechanisms and systems against attacks that exploit the QUIC protocol are necessary if QUIC is to be deployed in production networks.

Acknowledgements: The authors thank the anonymous reviewers for their valuable feedback on improving the quality of this paper. The authors also acknowledge the QUIC working group for their timely response and feedback on the findings reported in this paper.

Bibliography

[IT21] J. Iyengar, M. Thomson. QUIC: A UDP-Based Multiplexed and Secure Transport. *Internet Engineering Task Force*, 2021.

[LRW+17] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar et al. The quic transport protocol: Design and internet-scale deployment. In *Proc. ACM SIGCOMM*. Pp. 183–196. 2017.

[QUI20] QUIC in the wild. https://quic.comsys.rwth-aachen.de/stats.html, 2020. Accessed: 15-10-2020.

[RPDH18] J. Rüth, I. Poese, C. Dietzel, O. Hohlfeld. A First Look at QUIC in the Wild. In *Proc. PAM*. Pp. 255–268. Springer International Publishing, 2018.

[RR20] C. Rossow, G. S. Reen. DPIFuzz: A Differential Fuzzing Framework to Detect DPI Elusion Strategies for QUIC. In *Proc. ACSAC*. 2020.

[SBFF19] E. Sy, C. Burkert, H. Federrath, M. Fischer. A QUIC Look at Web Tracking. In *Proc. PETS*. Volume 2019(3), pp. 255–266. Sciendo, 2019.