QUIC Attacks: Implementation and analysis

Yuchen Wang

（Department of Computer Science and Engineering　Supervisor：Shan Chen）

[**ABSTRACT**]: The QUIC protocol is receiving increasing attention from major technology companies since the establishment of standardization by IETF, and more and more QUIC implementations are being developed. As the core protocol of HTTP/3, the security of QUIC will continue to be of concern. In this regard, this report will start from the version negotiation mechanism of QUIC to introduce an request forgery attack based on this mechanism. We analyze the controllable bits of the version negotiation messages and show that this scheme can be utilized to impersonate other UDP-based protocols, e.g., DNS requests.

[**Key words**]: QUIC; Version negotiation; Request forgery

[摘要]：QUIC 协议自从被 IETF 标准化以来，受到了来自主要技术公司的越来越多的关注，越来越多的 QUIC 实现正在被开发。作为 HTTP/3 的核心协议，QUIC 的安全性将继续受到关注。在这方面，本报告将从 QUIC 的版本协商机制开始，介绍基于该机制的请求伪造攻击。我们分析了版本协商消息中可控的比特，并展示了该方案可以被利用来模拟其他基于 UDP 的协议，例如 DNS 请求。

[关键词]： QUIC 协议 ； 版本协商 ； 请求伪造

Contents

1. Background

QUIC (Quick UDP Internet Connections) is a novel transport layer protocol designed to offer faster, more reliable, and more secure network connections. Initially developed by Google, QUIC serves as an alternative to TCP and TLS for transmitting data over the Internet. It is built on top of the UDP protocol, enabling it to bypass the TCP connection establishment process, thereby reducing connection latency and enabling faster connection recovery during network switches.

The development journey of QUIC has been marked by significant milestones. Google first introduced QUIC in 2013 as an experimental protocol to address latency issues. Over time, it underwent several iterations and refinements, with the IETF (Internet Engineering Task Force) officially adopting it as an Internet standard in 2020, known as QUIC Version 1[1-2].

Various tech giants have actively researched and contributed to the advancement of QUIC. Google, being the original developer, has been a major driving force behind its development. Additionally, companies like Cloudflare, Facebook (now Meta Platforms), and Microsoft have also been actively involved in QUIC research and implementation.

As mentioned above, QUIC will keep connection persistent during network switches, allowing the peer UDP port and IP address to be changed. This mechanism is called Connection Migration in QUIC. In this situation, a server may send UDP packets to an unknown host. As a consequence, QUIC is certainly vulnerable to address spoofing and request forgery. The specification also acknowledges the vulnerabilities and provides first security considerations.

In this report, we will take a detailed look at version negotiation request forgery attack. To this end, we focus on request forgery attacks initiated by a QUIC client (the attacker). In this situation, request forgery induces a QUIC server (the victim) sending packets under the control of the attacker. Therefore, the attacker can use the server’s position in the network to gain higher privileges.

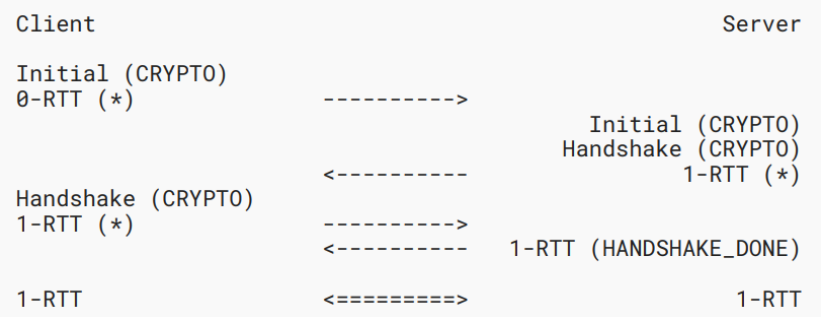


Figure 1 Simplified QUIC handshake[1]

2. Request Forgery Attack in QUIC

2.1 QUIC Basics

QUIC connection migration allows for adapting to network changes during the connection process without interrupting the service, thereby providing a better user experience and reliability. To support this mechanism and identify connections, connection ID (CID) is designed. Connection ID is a unique identifier used in QUIC to identify connections. During the establishment of a QUIC connection, both parties exchange Connection IDs to initiate the connection. The byte length of CID can be variable but should be less than 20 bytes normally. Its length can be up to 255 bytes in order to allow adjustments in the definition of future QUIC versions.

The QUIC handshake(Fig.1) combines the transport layer handshake and the TLS cryptographic handshake. The initial packets resemble the 3-way handshake of TCP, while the TLS parameters are carried in CRYPTO frames. All packets in the handshake use the long header format containing a Source CID (SCID) as well as a Destination ID (DCID) with their corresponding lengths. If a server receives an initial packet of unknown version, it will response a version negotiation packet, providing a list of its supported versions. The version field of version negotiation packets must always be 0x00000000[1].

2.2 Threat Model

Request forgery attacks occur when an attacker is able to trigger a host (victim) to send one or more “unintended” network requests to another host (target). An attacker can utilize the request forgery attack for two goals: First, utilizing the high authority of the victim. For example, an attacker may gain internal network access of the server. Second, utilizing the high bandwidth available from the server to the target.

For the attacks, we assume that an attacker has the ability to partially or even fully control the content of QUIC packets sent to the victim, besides IP address and port spoofing. We restrict the attacker to modifications of messages that are still recognized by the victim as valid QUIC packets to show that the vulnerabilities stem primarily from the protocol design. The target does not required to be able to understand QUIC packets but at least one UDP port is available to receive incoming datagrams. While the target might not be directly reachable from the attacker, the victim must be able to reach which follows the threat model showed in Fig.2.

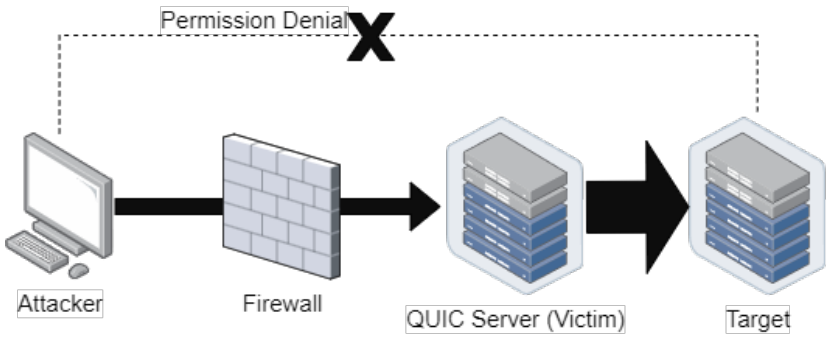


Figure 2 Threat model of request forgery, through which an attacker can obtain high privileges or bandwith.

3. Attack Modality

3.1 Version Negotiation Request Forgery

Version Negotiation Request Forgery (VNRF) capitalizes on a flaw within the QUIC handshake process. Typically, upon receiving an initial packet from a client with an unknown version, a QUIC server responds with a version negotiation packet. However, VNRF involves a malicious client sending a fabricated version identifier. This action prompts the server to initiate the version negotiation process. Simultaneously altering the source IP and port of the UDP datagram, the malicious client directs the version negotiation packet toward the target destination.

3.1.1 Controllable Bits of Version Negotiation Packet

Version negotiation packets have a simple structure, as shown in Fig.3. The first bit is always set to 1, indicating a long header packet. The next seven bits are unused and can be set to any value by the server. The following 32-bit version identifier should be all zeros indicating a version negotiation packet. The last 32 bits of the packet contain the identifiers for supported versions, which are decided by the server. Importantly, these bits are beyond the attacker's control.

The only controllable bits are DCID, SCID and their corresponding lengths. The length of connection ID is 8 bits, representing a maximum CID length of 255 bytes, which is 2040 bits. Besides, according to the QUIC stander, the server must mirror the DCID and SCID fields of the client packets when responding a version negotiation packet. Thus, there are always totally 512 bytes of the packet controlled by the attacker.

3.1.2 VNRF Proof of Concept

In this section, we will design a datagram that is a valid QUIC version negotiation packet as well as a valid DNS request. This datagram could be sent by the server (the victim) and

```
Version Negotiation Packet {
  Header Form (1) = 1,
  Unused (7),
  Version (32) = 0,
  Destination Connection ID Length (8),
  Destination Connection ID (0..2040),
  Source Connection ID Length (8),
  Source Connection ID (0..2040),
  Supported Version (32) ...,
}
```

Figure 3 Version Negotiation Packet Format[1]

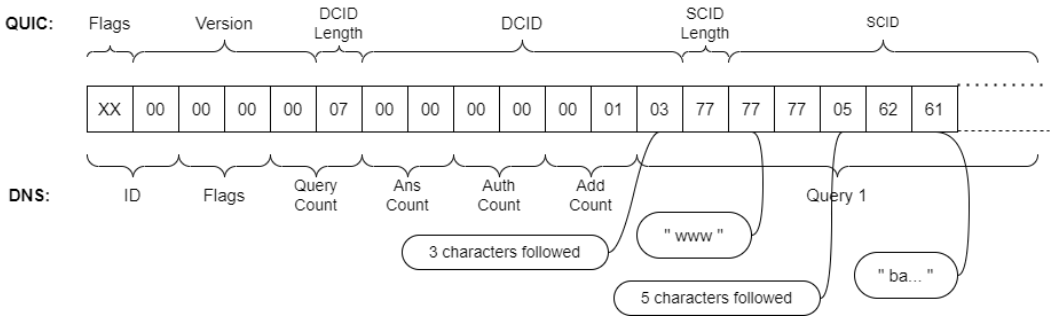


Figure 4 Translation of payload under QUIC and DNS.

trigger a valid DNS response sent by a DNS server. Here, DNS is just an example, which is a widely used UDP-based protocol. There are several UDP-based protocols in the networks while some of them may be impersonated through VNRF.

We attempt to create a DNS request for the domain www.baidu.com. In Fig.4, the initial bytes of the custom packet are displayed, along with the QUIC interpretation (above) and the DNS interpretation (below). The first byte of the QUIC packet is set to one, indicating a long header, followed by seven bits of arbitrary value. This byte, combined with the first zero-byte of the version field, serves as the query ID. The subsequent two zero-bytes of the version number represent the DNS flags specified in RFC 1035[3]. These flags indicate a standard query without truncation and recursion desired, which is a valid setting for DNS queries. The last byte of the version field represents the first byte of the number of host queries within the DNS request. The second byte of this number is determined by the DCID length of the QUIC version negotiation packet.

To determine the DCID length, we select a fixed value of seven. This choice minimizes the number of required hostnames in the query while still allowing us to skip the remaining bytes related to the answer number (Ans), authority records (Auth), and additional records number (Add). Typically, the Ans and Auth bytes are set to zero in a normal query and cannot be utilized to expand the payload. While the number of additional records is usually zero in a standard DNS query, we set the two Add bytes to 0x0001 to handle the remaining version number identifiers of the version negotiation packet that follow the CIDs.

The final byte of the DCID indicates the label count in the DNS query, representing

the number of characters before the first dot of the domain name to be queried. As for the SCID length, it corresponds to the first byte of the hostname www.baidu.com. Each domain level is denoted by a length octet, and the top-level domain is concluded by a zero-byte. With the SCID length designated as w (0x77), there remain 119 bytes of SCID available for the remaining payload. This capacity is sufficient to accommodate the entire query for www.baidu.com.

To ensure a total of seven queries, six additional queries to the root domain were appended. The hostnames used for padding are arbitrary but must conform to the DNS specification. The root domain was chosen as it occupies minimal payload space. In the Add section query entry, the domain root (0x00) was designated, and both the type and class were set to zero. The length of the Add entry was adjusted to encompass the remaining SCID payload along with the length of the version identifier array in the version negotiation packet. The number of version identifiers advertised by the server remains static and can be determined by initiating a version negotiation without a spoofed address. Given that version identifiers are consistently 4-byte values and the Add entry length is measured in bytes, the length of the array is multiplied by four. The remaining payload space between the beginning of the Add entry and the version identifiers was filled with random bytes. This encoding method ensures that the additional record remains nonsensical to a DNS server. Nevertheless, the entire packet is accounted for, and the forged request remains a valid DNS request.

Fig.6a shows the packet capture of a VNRF-based protocol impersonation, utilizing the payload described previously. To verify the legitimacy of the forged packet, the entire QUIC traffic (left) is decoded as DNS (right) in Wireshark. For the execution of a genuine DNS query, the spoofed address was configured to the Google DNS server 8.8.8.8:53. Consequently, Fig.6aa illustrates the packet types, Fig.6ab displays the first header byte and version, and Fig.6ac exhibits the CIDs in the payload. Although the initial packet appears as a malformed DNS packet, the right pane confirms that the version negotiation packet is correctly interpreted as a valid DNS request to www.baidu.com (Fig.6bd), resulting in a valid DNS response containing the IP addresses (refer to Fig.6ce).

This proof of concept highlights the potential of VNRF to impersonate various protocols. Crafting payloads for other protocols than DNS may require significant effort and creativity due to protocol-specific constraints. However, with persistence and careful manipulation, VNRF can be a potent attack method against a variety of protocols in utilizing the specification of the impersonated protocol.

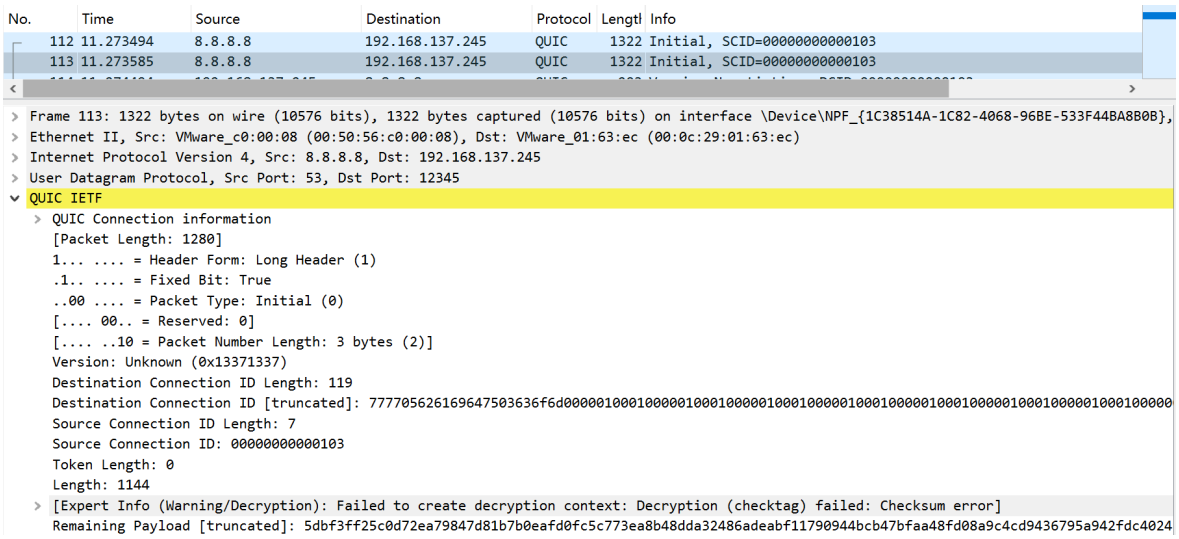
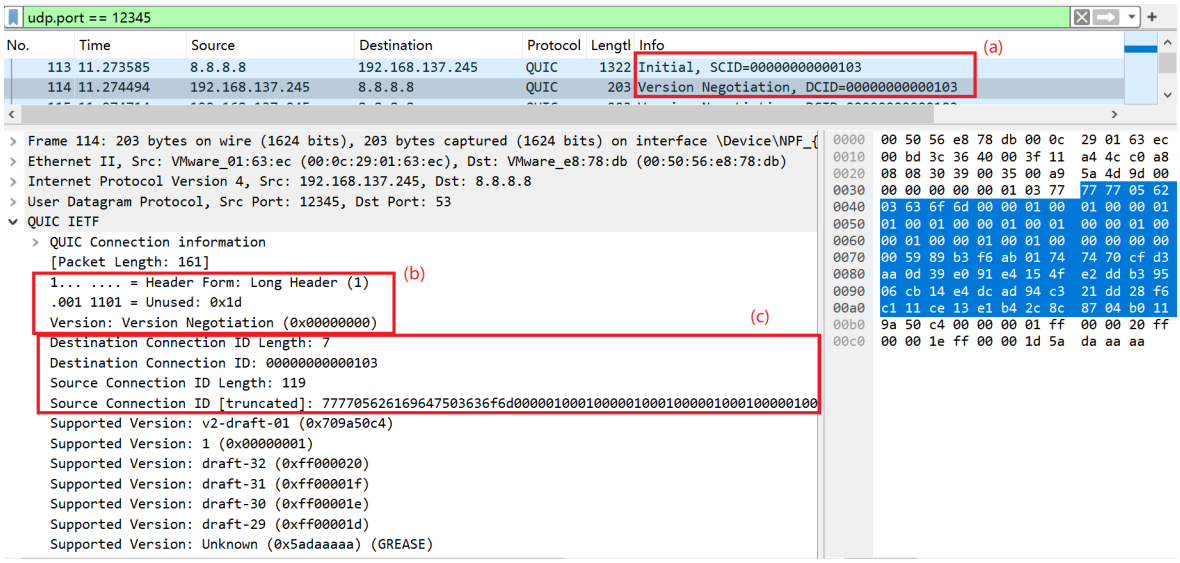


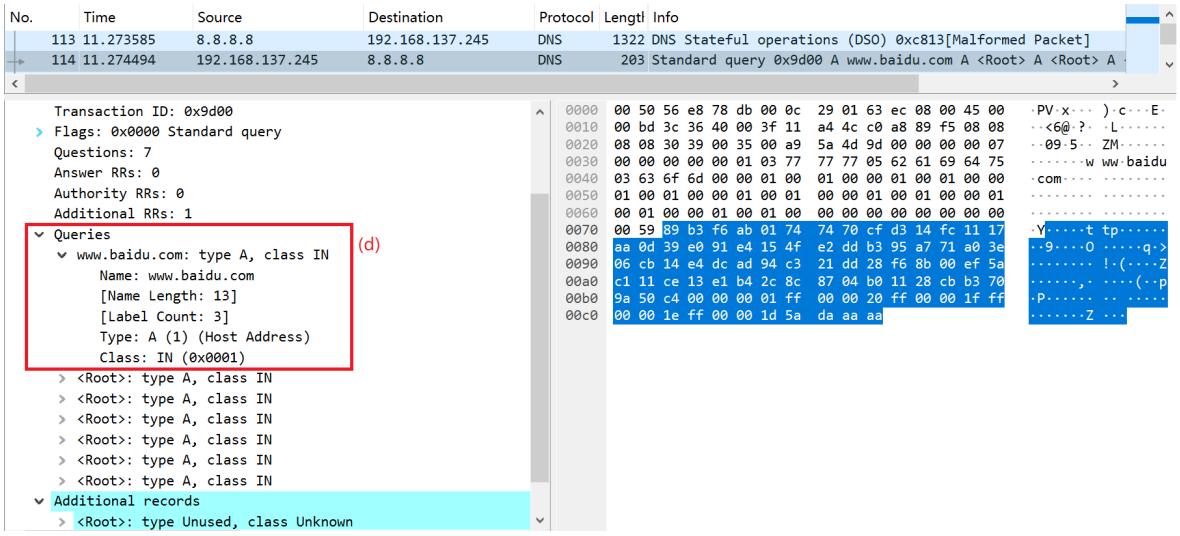
Figure 5 Wireshark client initial packets capture.(Decoded as QUIC)

3.1.3 VNRF Drawback

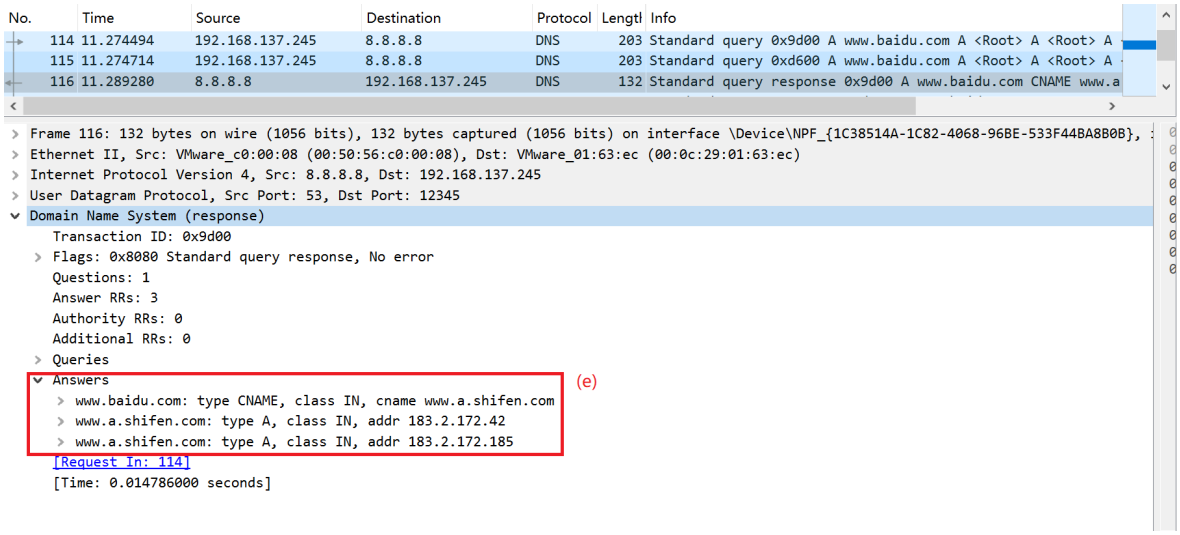
The proof of concept discussed above is run on two virtual machine with Ubuntu 20.04 LTS system. However, while running the same code on two Tencent Cloud lighthouse (a light-weight cloud server), it did not work. According to our conjecture, the reason is that the packets sent from port 53 will be blocked by firewalls. Normally a DNS server will never send packets proactively to any host.



a) Wireshark version negotiation packets capture.(Decoded as QUIC)



b) Wireshark version negotiation packets capture.(Decoded as DNS)



c) Wireshark DNS response packets capture.(Decoded as DNS)

4. Conclusion

In this report, we investigated the controllable bits of QUIC version negotiation packets and demonstrated a practical DNS request that masquerades as a QUIC packet, thereby being forwarded to a DNS server by a QUIC server. This exploit highlights the potential for request forgery attacks facilitated by version negotiation in QUIC, potentially granting attackers elevated access privileges.

Given QUIC's central role as the underlying protocol for HTTP/3, it's imperative to conduct additional security research before its widespread adoption across networks. The demonstrated VNRF attack underscores the importance of thoroughly assessing and addressing potential vulnerabilities within QUIC implementations.

Reference

[1] IYENGAR J, THOMSON M. QUIC: A UDP-Based Multiplexed and Secure Transport[Z]. RFC 9000. 2021.

[2] THOMSON M, TURNER S. Using TLS to Secure QUIC[Z]. RFC 9001. 2021.

[3] Domain names - implementation and specification[Z]. RFC 1035. 1987.