When Cellular Networks Met IPv6: Security Problems of Middleboxes in IPv6 Cellular Networks

Hyunwook Hong, Hyunwoo Choi, Dongkwan Kim, Hongil Kim,
Byeongdo Hong, Jiseong Noh, and Yongdae Kim
Korea Advanced Institute of Science and Technology
{*hyunwook.h, zemisolsol, dkay, hongilk, byeongdo, jiseong.noh, yongdaek*}*@kaist.ac.kr*

***Abstract*—Recently, cellular operators have started migrating to IPv6 in response to the increasing demand for IP addresses. With the introduction of IPv6, cellular middleboxes, such as firewalls for preventing malicious traffic from the Internet and stateful NAT64 boxes for providing backward compatibility with legacy IPv4 services, have become crucial to maintain stability of cellular networks. This paper presents security problems of the currently deployed IPv6 middleboxes of five major operators. To this end, we first investigate several key features of the current IPv6 deployment that can harm the safety of a cellular network as well as its customers. These features combined with the currently deployed IPv6 middlebox allow an adversary to launch six different attacks. First, firewalls in IPv6 cellular networks fail to block incoming packets properly. Thus, an adversary could fingerprint cellular devices with scanning, and further, she could launch denial-of-service or over-billing attacks. Second, vulnerabilities in the stateful NAT64 box, a middlebox that maps an IPv6 address to an IPv4 address (and vice versa), allow an adversary to launch three different attacks: 1) *NAT overflow attack* that allows an adversary to overflow the NAT resources, 2) *NAT wiping attack* that removes active NAT mappings by exploiting the lack of TCP sequence number verification of firewalls, and 3) *NAT bricking attack* that targets services adopting IP-based blacklisting by preventing the shared external IPv4 address from accessing the service. We confirmed the feasibility of these attacks with an empirical analysis. We also propose effective countermeasures for each attack.**

1. Introduction

The increasing popularity of mobile devices such as smartphones and tablets enables a great number of users to enjoy cellular data services. This new trend inevitably increases Internet usage via mobile devices, along with demand for Internet Protocol (IP) addresses. This need has been even more pressing when we consider the upcoming Internet of Things (IoT) era. Because of this demand, Internet Service Providers (ISPs) have started to deploy IPv6 addresses, and many services are following this trend. For example, Apple announced that they would support IPv6-only network services and encouraged developers to use IPv6-based APIs [4]. Similarly, cellular operators are also beginning to adopt IPv6 addresses on their networks [15]. With the introduction of IPv6, as in IPv4 cellular networks, middleboxes are required to manage and protect their network resources effectively. Firewalls need to filter out incoming malicious packets. At least until the transition from IPv4 to IPv6 completes, stateful NAT64, a middlebox that translates an IPv6 address to an IPv4 address (and vice versa), is indispensable for backward compatibility with legacy IPv4 services.

For IPv4 cellular networks, there have been several studies revealing that middleboxes can be abused by exploiting their properties [31, 32, 40]. Wang *et al.* investigated middlebox properties such as mapping patterns on NAT boxes and filtering rules on firewalls that may have a large impact on both the performance and security of cellular customers [40]. Other works demonstrated that an adversary can inject malicious data by exploiting vulnerabilities in sequence number verification on a firewall [31, 32]. However, no prior work has considered the security issues related to IPv6 middleboxes.

In this paper, we analyze the properties of middleboxes deployed in IPv6 cellular networks. To this end, we investigate the security problems of IPv6 middleboxes in five major operators in three countries on different continents. As a result, we conclude that certain features of cellular networks combined with IPv6 middleboxes may expose end users to various attacks. One of the key features is the support of *end-to-end transparency* [9]. Since all hosts in IPv6 networks are allocated with public IP addresses, it is possible to directly send packets to end hosts (i.e. smartphones) in the cellular networks from the Internet. Other key features include that the core network of cellular networks utilizes only a /64 prefix of an IPv6 address for data transmission to end hosts, and a mobile device can change the last 64 bits of an IPv6 address (defined as the interface identifier, in short the IID) at any time. The last key feature originates from the nature of stateful NAT64. The stateful NAT64 maps N public IPv6 addresses to one public IPv4 address, an arrangement known as *N-to-1* mapping. This stateful NAT64 utilizes a relatively small number of external IPv4 addresses compared to NAT in IPv4 cellular networks. Our measurement also shows a high N value of *N-to-1* mapping. Combined with these features of IPv6 cellular networks, we discover that several properties of IPv6 middleboxes can cause serious problems.

First, we examined firewalls in IPv6 cellular networks. In

IPv4 cellular networks, a host outside the cellular network cannot scan the inside because NAT middleboxes do not allow incoming traffic not initiated from the internal network. However, when *end-to-end transparency* is applied, any external host (both IPv4 and IPv6 hosts) could scan the entire IPv6 cellular network unless cellular operators deploy additional access control using a firewall. Unfortunately, firewalls in three of the networks fail to block both scanning and unwanted traffic. Exploiting this vulnerability, an adversary could fingerprint cellular devices with scanning, and further, she could simply launch a denial-of-service (DoS) attack or an over-billing attack which gives an over-charged bill to a victim by sending unwanted data traffic. Moreover, /64 prefix-based data transmission amplifies the efficiency of these attacks, because an adversary only needs to identify the /64 prefix rather than the full IPv6 address to launch these attacks. These attacks are simple, yet critical to both users and operators. Note that operators usually consider an over-billing problem to be more serious compared to undercharging, and they are afraid of an out-of-service state, because it could cause the departure of subscribers.

Second, we investigated IPv6 middleboxes used for interconnecting IPv4 and IPv6 networks. With the features of IPv6 cellular networks, we developed three DoS attacks against the middleboxes. Free use of the IID allows a *NAT overflow attack*. A single cellular device can use at most 2^{64} IPv6 addresses by changing the IID part of an IPv6 address; thus, an adversary can create an extremely large number of mappings on stateful NAT64. Consequently, the created mappings can prevent other legitimate users from creating mappings, and further, this disrupts the availability of a service. Next, we developed *NAT wiping* and *NAT bricking attacks* by exploiting high *N* value of *N-to-1* mapping. A *NAT wiping attack* utilizes a typical NAT functionality where NAT boxes remove a NAT mapping record for a connection to recycle resources after the connection is closed. Because the firewall does not block TCP reset packets that do not fall in the TCP receive window, if an adversary identifies the target mapping, she can wipe out active NAT mappings by sending TCP reset packets. Since none of the packets can pass through the NAT boxes without its NAT mapping, victims who are using the wiped out mapping will experience a disruption of communication. A *NAT bricking attack* targets a service that adopts IP-based blacklisting to prevent malicious behavior. If an adversary prevents the shared external IPv4 address of stateful NAT64 from accessing the service, other users sharing the same external IPv4 address are denied the use of the service. We demonstrate the feasibility of these attacks with empirical experiments.

We also suggest possible mitigation for each problem. Scanning, over-billing, and traffic flooding problems can be simply mitigated by deploying a stateful firewall capable of checking each state of a flow. The *NAT overflow attack* can be alleviated, if stateful NAT64 in cellular networks utilizes only a /64 prefix for mapping creation or cellular operators utilize a full IPv6 address with limitations on the number of IIDs used concurrently. Furthermore, the *NAT wiping* and the *NAT bricking attacks* can be mitigated by lowering the *N* value, but it would not be a fundamental solution. The *NAT wiping attack* can be resolved with TCP sequence number verification, and the *NAT bricking attack* may be resolved when service providers do not block an IP address, which may be unavoidable in some cases. We discuss various cases for these solutions.

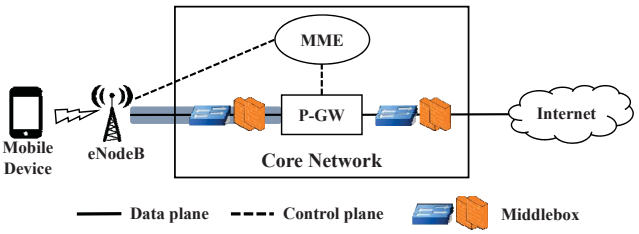


Figure 1: Data flow in LTE network

We summarize our contributions as follows:

- To the best of our knowledge, we are the first to investigate the security problems of IPv6 middleboxes in cellular networks. We analyze features only in IPv6 cellular network, and discover the critical problems of both a firewall and a NAT box deployed in IPv6 cellular networks.
- With the features only available in IPv6 cellular networks, we develop and demonstrate three DoS attacks on a stateful NAT64 middlebox as well as other attacks exploiting a defective firewall (please refer to Table 7). At the time of writing, we responsibly disclosed all these vulnerabilities to the operators.
- We propose effective and immediate countermeasures, and further devise long-term and comprehensive solutions that can mitigate these problems.

The rest of paper is organized as follows. We provide preliminaries in §2, then examine and discover several exploitable features only in IPv6 cellular networks in §3. In §4, we provide fingerprinting end hosts, over-billing, and traffic flooding attacks that exploit a defective firewall, and in §5 we introduce *NAT overflow*, *NAT wiping*, and *NAT bricking attacks* on middleboxes interconnecting IPv4 and IPv6 networks. Finally, we discuss and propose countermeasures in §6 and conclude the paper in §7.

2. Preliminaries

In this section, we describe the data flow and functions of IPv6 middleboxes in cellular networks. In addition, we introduce prior work related to the work in this paper.

2.1. Data flow in cellular network

Figure 1 shows the data flow in an LTE network. The core network first allocates an IP address to a mobile device during network attach procedures. The core network plays an important role in services such as call routing, authentication, and data charging. It includes a Mobility Management Entity (MME) and gateways such as Packet data network gateway (P-GW). MME provides control plane functions such as paging and authentication, and gateways provide connectivity between a mobile device and the data network.

The gateway is connected to the Internet, and the data traffic goes through the gateway when the mobile device uses the data service.

The IP packet flow from a mobile device to the Internet as follows. When IP packets are transmitted to eNodeB (the base-station in LTE) from a mobile device, eNodeB transfers these packets to the P-GW. Each IP packet is, then, encapsulated and tunneled between the P-GW and eNodeB. This tunnel is known as the GPRS Tunneling Protocol (GTP) tunnel. The P-GW eliminates the tunnel header and transmits the raw IP packet to the Internet. The downlink (from the Internet to the mobile device) follows the reverse direction.

Middleboxes located before or after the P-GW serve many important functions such as address translation, and dropping or modifying packets as needed. Note that other generation of networks (e.g., 2.5G/3G) also contains entities taking similar role as those in LTE networks. The following subsection details the functionalities of different middleboxes.

2.2. Functionality of IPv6 middleboxes

Network Address Translation (NAT). NAT is a mapping function, which maps one IP address into another by modifying the network address information such as the source IP address and source port number. The majority of NAT boxes are implemented on routing devices connected to the Internet with a public address assigned by an ISP, and it has a private address space [33] to map multiple private hosts to a single public IP address. When data traffic passes through NAT, the source IP address of the host is translated from a private IP address to the public IP address, and NAT records its mappings (especially using the destination IP address and port number) to forward the response traffic to a proper host. The benefits of NAT are an IP address sharing and the separation of networks. A large number of hosts can be connected to the Internet with a single IP address by using NAT, and NAT naturally protects internal networks by dropping incoming packets that do not exist on the recorded NAT mapping table. Stateful NAT64 [5] is an IPv6 transition mechanism that provides connectivity between IPv6 and IPv4 hosts by using a form of NAT. In NAT, a private IPv4 address is translated to a public IPv4 address, whereas in stateful NAT64, a public IPv6 address is translated to a public IPv4 address. In addition, stateful NAT64 maps the N number of public IPv6 addresses to one public IPv4 address to share limited IPv4 addresses. For the sake of brevity, we refer to this stateful NAT64 as "NAT64" throughout this paper.

Firewall. A Firewall is a crucial element for cellular operators. Firewalls in a cellular network detect and prevent harmful traffic from the Internet by pre-defined rules, and protect both the network infrastructure and end users from various attacks. Some firewalls enhance the performance and security by adopting rich functionalities such as TCP state tracking, TCP sequence number verification, and out-of-order packet buffering. In IPv6 cellular networks, there are two types of firewalls. One is for the traffic from the IPv6 Internet, and the other is for the IPv4 Internet. For the sake of brevity, we use "Firewallv6" to indicate the firewall between the IPv6 cellular network and the IPv6 Internet, and "Firewall64" for the firewall between the IPv6 cellular network and the IPv4 Internet.

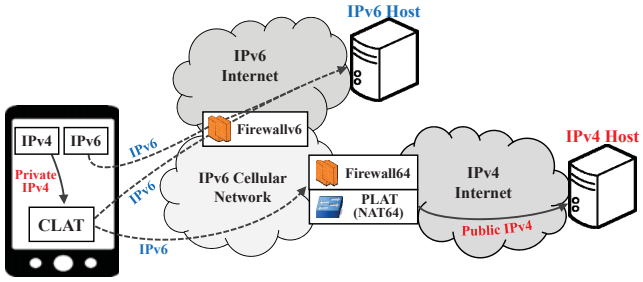


Figure 2: 464XLAT functionalities over cellular network architecture

464XLAT. Even with NAT64 and DNS64 [6], there is no guarantee that applications and services that operate on the IPv4 network would run on the IPv6 Internet. Specifically, applications that only support the IPv4 network are not able to communicate with other applications over the IPv6 network, when they use a hard-coded IPv4 address or use an IPv4-only socket to connect to another host. Accordingly, 464XLAT [27] can be used to solve this problem. 464XLAT uses two network translators, called Customer-side Translator (CLAT) [25] and Provider-side Translator (PLAT) [5]. CLAT provides a private IPv4 address for the application and algorithmically translates a private IPv4 address to the public IPv6 address, and vice versa. Because the CLAT function is applicable to a router or an end-node (e.g., mobile phone), Android smartphones may have the CLAT function inside [21]. On the other hand, PLAT translates an IPv6 address into a public IPv4 address. It translates many public IPv6 addresses to a single public IPv4 address and vice versa. Thus, PLAT has the functionality of NAT64. Figure 2 depicts the IPv6 cellular network architecture incorporating 464XLAT functionalities. These two translators enable IPv4-based hosts to communicate with other hosts over the IPv6 network. Note that some operators utilize a dual stack for the IPv6 Internet rather than using 464XLAT architecture. In this case, the operator provides a dual IP version of cellular networks: both IPv4 and IPv6 networks, where each host uses the cellular network based on their IP version. Thus, middleboxes such as Firewall64 and NAT64 are not used.

2.3. Related work

Identifying cellular middlebox properties related to security issues. Many studies on middlebox security in cellular networks have been conducted in recent years. Wang *et al.* [40] investigated the properties and policies on NAT boxes and firewalls in cellular networks. They discovered that misconfigured policies and properties on middleboxes may lead to a performance degradation, an increase in energy consumption on mobile devices, and be vulnerable to several attacks. They stated that scanning attack is possible when IP spoofing is available, flaws on

closing TCP connections give a chance to launch a battery draining attack, and TCP out-of-buffering behavior and a large sequence number of window amplifies a data injection attack. Qian *et al.* [31, 32] demonstrated that keeping the TCP sequence number state on the firewall can be exploited for a TCP sequence number inference attack. By obtaining side channel feedback such as OS packet counters from unprivileged malware installed on the victim's device, an adversary can identify the TCP sequence number of current sessions successfully. A successful inference can hijack an HTTP session and inject malicious data, and this leads to the returning of a phishing page or other malicious behavior on the victim's device. Unlike prior work dealing with IPv4-only middleboxes, we examined the properties of middleboxes in *IPv6 cellular networks*, and discovered misconfigured settings such as incorrect filtering rules and a large sequence number of windows.

Security issues of IPv6. A number of studies have been conducted on the security issues of new features in IPv6. One is about IPv6 Router Advertisement (RA). It conveys control information to hosts with the purpose of their auto-configuration. However, once an adversary sends rogue RAs, it can cause operational failure of neighboring hosts on an IPv6 link [10]. Furthermore, Yang *et al.* [42] experimented with a DoS attack against IPv6 Neighbor Discovery Protocol, DDoS attacks such as TCP, UDP, ICMP flooding, and Smurf attacks. In addition, Gont *et al.* [17] discussed an IPv6 version of the Smurf attack that misuses an IPv6 option with a multicast address. In this paper, we only focused on the features of an IPv6 cellular network, and discovered critical problems of IPv6 cellular middleboxes.

Network scanning. Bellovin *et al.* [7] introduced scanning methods of IPv4 and IPv6 by examining address-space scans for worm propagation. They observed that patterns in the address-space can improve scanning ability. Czyz *et al.* [11] studied IPv6 security policies by examining dual-stack servers and routers. They used scamper [26], an open-source scanning tool, to conduct IPv6 traceroute and ping to collect data for various protocols such as ICMP, TCP, and UDP. Nmap (Network Mapper) is a flexible tool [34] used to scan large networks. There are also many methods for network scanning, such as IRLscanner [36] and Zmap [13]. Especially, Zmap is able to scan the entire IPv4 address range in under 45 minutes. We also performed network scanning to fingerprint live end hosts in cellular networks. We discovered a number of live end hosts by using a smart strategy. The strategy is useful for scanning in a short time while minimizing side effects.

Threats to accounting and charging system. An accurate and fair data charging is an important issue for both operators and users. A number of studies warned that misconfigured middlebox policies can lead to two types of attack: an over-billing attack and a free riding attack. The timeout after a client closes the connection could expose the client to over-billing attack, and an adversary could use free charge of data by exploiting the loopholes that data communication using DNS port is not charged [29]. Furthermore, Go *et al.* showed that TCP retransmission was misused for an attack vector [16]. Because accounting system has only a view of IP layer in the middle of network, it cannot figure out the accurate state of TCP context; thus it cannot verify whether the retransmission is malicious or not. By exploiting this vulnerability, they showed that an over-billing attack and a free riding attack were possible. Peng *et al.* showed that an adversary can impose over charged bill to a victim when IP spoofing is feasible in cellular networks [30]. Moreover, the adversary can hide inflated data usage volume by adjusting a TTL value in IP header. In a LTE network, there were several studies introducing over-billing and free riding attacks by exploiting a VoLTE interface newly adopted to LTE [22, 24]. In this paper, we first show that users in IPv6 cellular network are exposed to an over-billing attack due to the absence of stateful firewall functionality in the network.

DoS attacks targeting cellular core network modules. Zhao *et al.* [43] pointed out that the security of IMS is poorly examined. They showed that an adversary can utilize the chained automatic reaction of the Resource List Server (RLS) to block all the services of IMS. Traynor *et al.* [37] analyzed the impact of cellular botnets on HLR, a central repository of user profile data. They discovered that a flooding attack on HLR can congest the authentication procedure and block all users from connecting to the core network. In LTE networks, Kim *et al.* [22] demonstrated that launching DoS attacks on IMS is possible by sending a few INVITE messages for call initiation. In our work, we develop and demonstrate three DoS attacks on stateful NAT64 middleboxes with the features only available in IPv6 cellular networks.

3. Discovering Exploitable Features of IPv6 Cellular Networks

In this section, we examine and discover several exploitable features of IPv6 cellular networks that can cause various security problems. Note that we investigate the security problems by carrying out real-world experiments on five major operators in three countries on different continents. In accordance with responsible disclosure, we anonymized each operator and labeled them as OP-I, OP-II, OP-III, OP-IV and OP-V, respectively. For all the experiments in the work, we only targeted our devices and services running on our servers; therefore, no legitimate users and services were affected by the experiments. Furthermore, we target middleboxes used for common data communications between mobile devices and the Internet; thus, middleboxes for voice data communications, such as Voice over LTE (VoLTE), are beyond the scope of this paper.

3.1. End-to-end transparency in cellular networks

The term *"transparency"* refers to the original Internet concept of a single universal logical addressing scheme, and the mechanisms by which packets may flow from source to destination essentially remain unaltered [9]. Nevertheless, the use of NAT and private addresses due to the shortage of public IPv4 addresses have spoiled the *end-to-end*

TABLE 1: Access control on firewalls (✓: Existence, ✘: Non-existence, –: No Firewall64)

Middlebox	Incoming traffic	OP-I	OP-II	OP-III	OP-IV	OP-V
Firewallv6	from IPv6 Internet	✓	✘	✘	✘	✓
Firewall64	from IPv4 Internet	-	✘	✘	✘	-

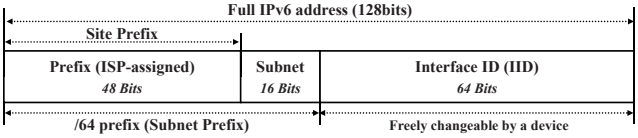


Figure 3: A /64 prefix and the interface identifier of an IPv6 address

transparency in the Internet. However, because IPv6 has an enough address space for all hosts, it can provide the *end-to-end transparency*. Recently, cellular networks have started to deploy the IPv6 architecture supporting the *end-to-end transparency*. This facilitates a direct communication between end-hosts inside and outside of the cellular network.

In contrast to the Internet, there exist many cellular-specific issues such as mobility management, power consumption, and data usage based charging in cellular networks. Therefore, cellular networks require appropriate access control (e.g. a firewall), while providing *end-to-end transparency*. Without a proper access control mechanism, cellular networks could be exposed to direct attacks from outside.

To check whether firewalls were in place to manage external traffic correctly, we first performed a reachability test to end hosts in IPv6 cellular networks by sending probe packets from the outside. For five different operators, we sent both a TCP and a UDP packet to the end-host (our phone) inside the IPv6 cellular networks and observed whether these sent packets correctly arrived at our device. Table 1 presents the results. OP-I and OP-V successfully blocked the probe packets coming from the Internet. However, OP-II, OP-III, and OP-IV allowed the probe packets to pass through. In other words, these three networks could be vulnerable to the scanning, DoS, and over-billing attacks described in §4.

3.2. IPv6 address allocation in cellular networks

As shown in Figure 3, an IPv6 address comprises a /64 prefix and the interface identifier (IID), and Figure 4 presents an IPv6 allocation procedure in LTE networks. To obtain an IPv6 address, a mobile device sends an *Attach Request*, which includes a flag indicating the use of IPv6 communication, to a MME. Next, the MME sends a *Create Session Request* to a Serving Gateway (S-GW), and the S-GW relays the messages to a P-GW to establish a default bearer. The default bearer comprises a set of network parameter that defines the best effort service, and each default bearer comes with an IP address. Then, the P-GW sends an IPv6 IID and an IPv6 prefix to the MME through a *Create Session Response*. Once the MME informs the mobile device of the *Attach Accept* that contains the IID, the mobile device

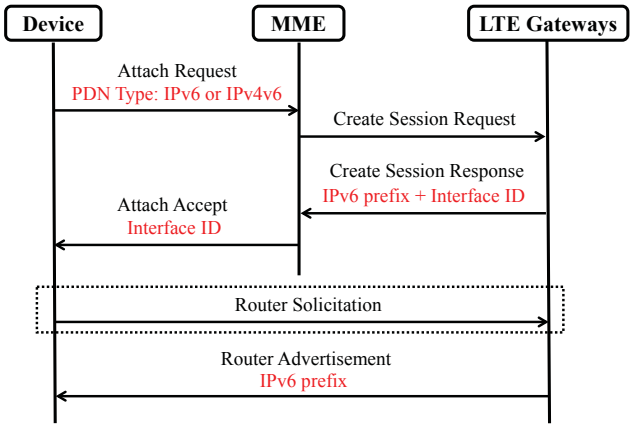


Figure 4: IPv6 address allocation procedure in LTE networks

receives an IPv6 prefix through a *Router Advertisement* from the P-GW. This means that the range of the prefix depends on the network, rather than on the mobile device. The mobile device uses the identifier from the MME or sets an address by generating an IID.

The P-GW allocates a unique /64 prefix for each default bearer, and a mobile device can change the IID [1, 2, 23] at any time. We conducted an experiment to check if a mobile device is able to use a particular data service when the IID is changed. We arbitrarily modified the IID of the source IP by using a raw socket and sent these packets to the Internet. We performed this experiment for each operator, and received the same response from all of them. When we sent a modified *TCP-SYN* packet to a web server, we received a *TCP-SYN/ACK* packet. Then, we completed the three-way handshake by sending a *TCP-ACK* packet that uses the same IID. After this, we could send data to the service through the session completed by three-way handshake.

Note that we did not need to change the IPv6 address allocated to the network interface (e.g., rmnet1). We simply sent and received the IID modified packets. This result shows that a network only checks the /64 prefix to route the data to an end host, and not the full IP address. As long as the /64 prefix is identical to the initially allocated IPv6 address, the core network forwards the data packets even when the IID is different from the allocated IPv6 address. This allows us to use 2^{64} IP addresses with only a single device (because the IID is 64-bit). This feature can lead to several problems. In §4 and §5.2, we use this feature to demonstrate several attacks.

3.3. N-to-1 mapping on NAT64

NAT64 maps *N* public IPv6 addresses to one public IPv4 address, so-called *N-to-1* mapping. This seems reasonable because 1) IPv4 address space is much smaller than IPv6 space and 2) operators have a fixed number of public IPv4 addresses. Consequently, many users would share the same public IPv4 address; if the shared IPv4 address is blocked from a service by a malicious user, all the end users sharing the same IP address would be affected. Furthermore, if *N*

TABLE 2: Identified *N* value of NAT64

NAT type	NAT64			NAT in IPv4 network		
Operators	OP-II	OP-III	OP-IV	OP-I	OP-III	OP-V
# of trial (Airplane mode)	8,000	8,000	8,000	8,000	8,000	8,000
# of success IP allocation	6,285	7,383	7,999	6,368	7,772	7,918
# of unique local IP	6,213	7,382	7,999	6,367	7,769	7,817
# of unique external IP	646	1,021	504	3,844	2,106	1,789
Calculated *N* value	**9.62**	**7.23**	**15.87**	**1.66**	**3.69**	**4.37**

becomes larger, the impact of this problem is also increased. Thus, we investigate the *N-to-1* mapping characteristics of NAT64 in each operator and compare them with NAT in IPv4 cellular networks.

Because the external IPv4 address used for NAT64 mapping is changed when the source IPv6 address of the device changes (we learned this from the mapping pattern test, which is described in §5.1), we need to repeatedly change the allocated IPv6 on the device. In other words, IP allocation and deallocation procedures have to be repeated to determine how many internal IPv6 addresses are mapped to a single external IPv4 address of NAT64. These procedures can be repeated by utilizing Airplane mode. To utilize this, we implemented a shell script that runs on an Android device. This script automatically activates and deactivates the Airplane mode of the device, logs the IPv6 address allocation history, and sends *TCP-SYN* packets to our external server on the IPv4 Internet to obtain the external IPv4 address of NAT64. We repeated this for NAT in the IPv4 network. During the experiment, we observed unexpected errors in that the network reaches an out-of-service state due to an incomplete allocation procedure. Since the failure rate varies among the operators, we only considered successful IP allocation. However, the number of successful IP allocation is sufficient to identify the *N* value of *N-to-1* mapping of NAT64 and NAT in the IPv4 network.

Table 2 lists the values we measured for each operator. For NAT64, in the case of OP-II, among the 8,000 trials of Airplane mode, we could successfully establish 6,285 allocations. After eliminating duplicated IP addresses, we could obtain 6,213 unique IPv6 addresses which are mapped to 646 unique external IPv4 addresses. Thus, the *N* value was about 9.62, which means that about 10 users, on average, use the same public IPv4 address to access the IPv4 Internet simultaneously. The results were similar for OP-III, where the *N* value was about 7.23. However, in the case of OP-IV, the *N* value was much higher at 15.87. On the other hand, NAT in IPv4 networks had relatively lower *N* than NAT64. In OP-I, *N* was the lowest as 1.66, which means that this operator used more public IPv4 addresses than the other operators. *N* of OP-III was about 3.69, which is only half of the *N* of its NAT64. In the case of OP-V, the corresponding value was 4.37.

In summary, NAT64 utilizes a smaller number of public IPv4 addresses than NAT in IPv4 networks. Therefore, more efficient attacks could be available when an adversary targets NAT64 which has a high *N* value. We discuss this kind of attacks in §5.3 and §5.4.

4. Firewall Problems in IPv6 Cellular Networks

In this section, we examine firewalls in IPv6 cellular networks and show the problems lead to three attacks: fingerprinting end hosts, over-billing, and traffic flooding attacks. These attacks exploit the fact that firewalls in IPv6 cellular networks do not have proper access control for incoming traffic from the Internet. Furthermore, the /64 prefix-based data transmission, one of the key features we discovered in IPv6 cellular networks, amplifies the efficiency of these attacks.

4.1. Fingerprinting end hosts

In IPv4 cellular networks, an external host cannot scan the cellular network, because NAT allocates a private IP address to each internal host and blocks unsolicited traffic from outside. Therefore, it is not possible for an adversary to fingerprint live end hosts in IPv4 cellular networks. However, in IPv6 cellular networks, owing to the large IPv6 address space, operators can allocate a public IPv6 address to each internal host without utilizing NAT. In this case, it would be possible for an adversary to scan the network and fingerprint the live end hosts when the IPv6 address range of the target cellular network is known. In our work, to show the possibility of fingerprinting end hosts in IPv6 cellular networks, we performed a scanning test on the five cellular networks. Fingerprinting end hosts in a cellular network requires the target service (the port number), the scanning method (such as a TCP, UDP, or ICMP scan) and its origin (the Internet or the cellular network), and the target range of IPv6 addresses to be determined in advance. We first explain how these conditions can be determined, and then show our scanning results.

Port number. As we only focus on finding end hosts instead of finding running services, we chose an arbitrary port number. In the actual experiment, we used TCP/UDP port number 80. However, it should be noted that any port number can be used instead.

Scanning method and its starting point. We conducted a response test that consists of sending three different packets – an ICMP echo request (Ping), a *TCP-SYN*, and a UDP – to our test devices to determine if the networks block probing packets. We sent these three packets from inside of each cellular network and from the Internet. Table 3 shows the results. Three of the operators (OP-II, OP-III, and OP-IV) did not block *TCP-SYN* and UDP packets originating from the Internet. Therefore, it would be possible to scan these operators from the Internet. In OP-I, although network scanning from the Internet was impossible, scanning from within the same cellular network was possible using all three scanning methods. In OP-V, regardless of the scanning method and its starting point, network scanning was found to be impossible. Note that, from an adversary's perspective, scanning from within the cellular networks is not an attractive option, because 1) the Internet has much higher bandwidth than

TABLE 3: Results of a response test (**ICMP-I**: Echo Request, **ICMP-II**: Echo Reply, **ICMP-III**: Destination Port Unreachable, **–**: No Response)

Origin	Packets	OP-I	OP-II	OP-III	OP-IV	OP-V
Cellular Networks	ICMP-I	ICMP-II	ICMP-II	–	ICMP-II	–
	TCP-SYN	TCP-RST	TCP-RST	–	TCP-RST	–
	UDP	ICMP-III	ICMP-III	–	ICMP-III	–
Internet	ICMP-I	–	–	–	ICMP-II	–
	TCP-SYN	–	TCP-RST	TCP-RST	TCP-RST	–
	UDP	–	ICMP-III	ICMP-III	ICMP-III	–

a cellular network, and 2) an adversary inside the cellular network would be charged for the attack traffic. Therefore, launching the attacks from inside the cellular network would neither be effective nor efficient in terms of both bandwidth and cost. Therefore, we determined the Internet as being a suitable starting point from which to conduct network scanning and targeting the three operators OP-II, OP-III, and OP-IV, and chose *TCP-SYN* as the scanning method.

The range of IPv6 addresses. Scanning an IPv6 network is not trivial due to the large address space. At full-rate, full-duplex 40 Gbps, over 5,000 years are known to be required to scan the entirety of a single 64-bit subnet [12]. In our work, because the purpose of scanning is to demonstrate that an adversary can find a set of live end hosts rather than to discover all the live end hosts in the network, we focused on how to identify more live end hosts in a short time. To this end, we extracted the IPv6 address range for each operator from the IP list acquired in Table 2, and investigated the allocated IPv6 addresses on a device of each operator to discover the range that is actually used.

Table 4 presents the summary of the IPv6 address range of each operator. We observed that some parts of the addresses are fixed. For example, in OP-III, only 58 bits are changed randomly during IP allocation. Specifically, all the bits in the 4th, 7th, and 8th 16 bits block are variable excepting the 3rd 16 bits block. Each of the other operators also showed their own pattern. By analyzing these patterns, we could reduce the range of IPv6 addresses for scanning purposes.

Although we narrowed the address range as in Table 4, it still requires a considerable amount of time to scan all possible IP addresses. For example, in the worst case, we would need to try 75×2^{45} IP addresses to scan the OP-II networks, which is roughly 600,000 times more than all the IPv4 addresses. In addition, because scanning would lead to a large amount of traffic, this might harm the network operator. Therefore, we designed a smart strategy, which is useful for scanning in a short time and minimizing side effects. We could identify the live end hosts during the scanning experiment by taking the advantage of the way in which operators allocate an IPv6 prefix to devices. As described in §3.2, operators allocate a unique /64 prefix for each default bearer (and to the device) and route user traffic based only on that prefix regardless of the IID. In other words, to ensure that a scanning packet (a *TCP-SYN* packet in our case) reaches an end device does not require the full IPv6 address of the end device; instead, the /64 prefix is sufficient. Thus, if we send a scanning packet of which the /64 prefix is identical to that of the device with different IID, the packet still arrives at the device. Furthermore, when the device receives this unsolicited packet, it may respond to the packet by sending an ICMP destination unreachable message. Thus, we can easily fingerprint live end hosts. In our test, we modified the /64 prefix based on Table 4 for the destination of *TCP-SYN* packets while keeping the IID fixed as 1.

As a result, only within five minutes, we could fingerprint 364, 674, and 12 live hosts in OP-II, OP-III and OP-IV, respectively. Although we could not find all end hosts in these cellular networks, these results are meaningful for an adversary. The IPv6 cellular network maps the IP addresses directly (i.e., without using NAT) and a firewall does not block scanning traffic, this approach to network scanning is useful for an adversary trying to find target hosts. Finally, network scanning could lead to further problems such as over-billing, flooding, and DDoS attacks detailed below.

4.2. Over-billing attack

Most cellular operators provide various charging plans for mobile users, by which the users are charged for their data usage according to the volume of data they download or upload. Therefore, when a device initially attempts to connect to the Internet (i.e., outbound traffic), an operator can charge the user for data usage by considering that such connections are intended by the user. However, in the opposite case (a connection is initiated by inbound traffic from the Internet, before data communication is started), it is difficult to setup a charging policy because the operator cannot identify whether the data traffic is user-intended or not. IPv4 cellular networks overcome this problem by leveraging the existing NAT solution to filter out unintended inbound traffic.

When connections are established from the device to the Internet, NAT records the mapping of the connections in the form of a 5-tuple (source IP address, destination IP address, source port number, destination port number, and protocol). This NAT mapping is later used to filter out unknown inbound traffic. If there is a mapping for inbound traffic, it is passed to the target device and charged. Note, there are some cases in which inbound traffic is permitted such as communication between two end hosts behind NAT or a real-time messaging service [41]. In these cases, operators use additional methods such as NAT traversal [8, 35] or the PUSH mechanism [3, 18] to permit only these types of traffic.

On the other hand, an IPv6 cellular network does not require NAT because IP addresses are mapped directly to the devices without the need for translation. Therefore, IPv6 operators cannot take the advantage of NAT to control inbound traffic reaching the user. If operators do not perform an additional access control check by means of a firewall, the cellular user may receive large amounts of inbound traffic from an adversary, and consequently the user may be

TABLE 4: Summary of allocated IPv6 address ranges. (**xxxx**: the operator's unique identifier, **[x-y]:** the range from x to y (in hex), **# of Fixed bits**: the number of fixed bits, **# of Possible IPs**: the number of possible IP addresses, **# of Possible /64 prefixes**: the number of possible /64 prefixes)

Operators	**Allocated IPv6 address ranges** (total 128 bits, separated by 16 bits block)	**# of Fixed bits**	**# of Possible IPs**	**# of Possible /64 prefixes**
OP-I	xxxx : xxxx : b0[00-4f] : [0-ffff] : 0 : [0-4a] : [0-ffff] : [0-ff]01	74	375×2^{44}	5×2^{20}
OP-II	xxxx : xxxx : 120[0-f] : [0-ffff] : 0 : [0-4a] : [0-ffff] : [0-ff]01 xxxx : xxxx : 190[0-f] : [0-ffff] : 0 : [0-4a] : [0-ffff] : [0-ff]01	74	75×2^{45}	2^{21}
OP-III	xxxx : xxxx : 21[0-f] : [0-ffff] : 0 : 0 : [0-ffff] : [0-ffff] xxxx : xxxx : 24[0-f] : [0-ffff] : 0 : 0 : [0-ffff] : [0-ffff] xxxx : xxxx : 30[0-f] : [0-ffff] : 0 : 0 : [0-ffff] : [0-ffff] xxxx : xxxx : 9[10-2f] : [0-ffff] : 0 : 0 : [0-ffff] : [0-ffff] xxxx : xxxx : 95[0-f] : [0-ffff] : 0 : 0 : [0-ffff] : [0-ffff]	70	3×2^{53}	3×2^{21}
OP-IV	xxxx : xxxx : 10[00-16] : [0-ffff] : 0 : [0-38] : [0-ffff] : [0-ff]01 xxxx : xxxx : 30[00-16] : [0-ffff] : 0 : [0-38] : [0-ffff] : [0-ff]01	76	1311×2^{41}	23×2^{17}
OP-V	xxxx : xxxx : 818[0-3] : [0-ffff] : 1 : [1-2] : [0-ffff] : [0-ffff] xxxx : xxxx : 81c[0-3] : [0-ffff] : 1 : [1-2] : [0-ffff] : [0-ffff] xxxx : xxxx : 898[0-3] : [0-ffff] : 1 : [1-2] : [0-ffff] : [0-ffff] xxxx : xxxx : 89c[0-3] : [0-ffff] : 1 : [1-2] : [0-ffff] : [0-ffff] xxxx : xxxx : 90a[0-3] : [0-ffff] : 1 : [1-2] : [0-ffff] : [0-ffff] xxxx : xxxx : a08[0-2] : [0-ffff] : 1 : [1-2] : [0-ffff] : [0-ffff] xxxx : xxxx : a80[0-2] : [0-ffff] : 1 : [1-2] : [0-ffff] : [0-ffff] xxxx : xxxx : a84[0-2] : [0-ffff] : 1 : [1-2] : [0-ffff] : [0-ffff] xxxx : xxxx : b00[0-2] : [0-ffff] : 1 : [1-2] : [0-ffff] : [0-ffff] xxxx : xxxx : b04[0-2] : [0-ffff] : 1 : [1-2] : [0-ffff] : [0-ffff]	69	35×2^{49}	35×2^{16}

TABLE 5: Data usage results before and after launching an over-billing attack.

Setting dst. IPv6 addr.	**Use full IPv6 addr. for data transmission**			**Use only the /64 prefix for data transmission**		
Operators	OP-II	OP-III	OP-IV	OP-II	OP-III	OP-IV
Before Test	132 MB	79.35 MB	9 MB	187.6 MB	13 MB	180 MB
After Test	**228.5 MB (+96.5)**	**177.97 MB (+98.62)**	**108 MB (+99)**	**285.5 MB (+97.9)**	**111 MB (+98)**	**280 MB (+100)**

charged for unwanted data usage. Therefore, we examined if this attack is possible in the IPv6 cellular networks. As already shown in Table 1, as OP-II, OP-III, and OP-IV have no access control for incoming traffic from the Internet, we also tested an over-billing attack on them. We sent a 1,024 byte UDP packet (including the header) 102,400 times (100 MB) from the Internet to our mobile device. Furthermore, because IPv6 operators only utilize the /64 prefix for data transmission, we also checked if they charge based only on the /64 prefix regardless of the IID. Note, to ensure more precise test results, we disabled all the background tasks of the device to prevent additional traffic from being produced. We utilized the customer service web page of each operator to check the billed data usage. Finally, we compared the data usage before and after conducting the test. Table 5 lists the results.

As shown in Table 5, OP-II, OP-III, and OP-IV charged 96.5 MB, 98.62 MB, and 99 MB, respectively, when the full IPv6 address was used for data transmission. Moreover, the same three operators also charged 97.9 MB, 98 MB, and 100 MB, respectively, when we only used the /64 prefix for data transmission. Note, the reason for the amount of data usage (e.g., +96.5 MB in OP-II) not being equal to the total size of data originally sent may be caused by different accounting policies in terms of the network header and payload [16] or packet loss. Nonetheless, these results show that an over-billing attack is possible in IPv6 networks. Moreover, this attack can be launched only with the /64 prefix of IPv6 addresses. If an adversary succeeded in discovering the /64 prefix of the IPv6 addresses utilizing Airplane mode described in §3.3 before launching a massive over-billing attack, it can amplify the efficiency of the attack, and might result in significant financial loss to both the cellular users and the operators.

4.3. Traffic-flooding attack

In IPv6 cellular networks, owing to *end-to-end transparency* and carelessly configured firewalls, any inbound traffic from the Internet can be forwarded to an end device. This renders the cellular networks vulnerable not only to scanning and over-billing attacks but also to a traffic flooding attack. Therefore, we carried out a traffic-flooding test. Note, if an adversary has sufficient resources such as a botnet, a kind of DDoS attack that targets cellular core network modules would be possible. Even though this attack is more powerful and critical, demonstrating this attack on a real operating network is infeasible; hence, we targeted our end devices.

The flooding test was conducted as follows. We first sent meaningless *TCP-SYN* packets from the Internet to our device and observed the status, while gradually increasing the bandwidth. This test was performed on the networks of three of the operators OP-II, OP-III, and OP-IV with

Samsung Galaxy S5, S6 edge plus, A5, Note 4 and LG g4 devices, all of which were connected to an LTE network. During the test, we measured traffic bandwidth currently processed by using a network monitoring app [28]. Despite the slight differences among the devices of each operator, most tested devices could not process more than 20 Mbps traffic bandwidth. When the sending bandwidth exceeded 70 Mbps, which is three times more than the capacity of the tested device, none of the tested devices on the network of each operator was able to access the Internet. Even after the test finished, some devices such as the Galaxy Note 4 and Galaxy S6 edge plus could not access the Internet. Because their respective network interfaces had fallen into the out-of-service state, this state continued unless we reset the network interface.

Since the purpose of this test is to demonstrate the feasibility of a flooding attack, we did not attempt to determine the reason for the problems presented by the device and the reason why the network interface fell into an out-of-service state.

5. Exploiting Properties of Middleboxes for Legacy IPv4 Networks

In this section, we investigate the properties of middleboxes in three operators that use the 464XLAT, and show how they can be misused by an adversary. From this, we develop three novel DoS attacks.

5.1. Properties of middleboxes for legacy IPv4 networks

We checked whether middleboxes bridging the gap between IPv4 and IPv6 networks have exploitable properties by investigating three of the operators: OP-II, OP-III, and OP-IV. We describe how we studied these properties and why they can be exploited.

Size of TCP sequence number window in Firewall64. As shown by Wang *et al.* [40], if a firewall permits a large TCP sequence number of windows, an adversary can inject malicious traffic by simply identifying the 5-tuple entry (source IP address, destination IP address, source port number, destination port number, and protocol). Therefore, we also identified the TCP sequence window size of the Firewall64 in three cellular operators. We created a TCP connection passing through a Firewall64, and sent packets using this connection. While sending the packets, we gradually increased the sequence number and checked whether these packets could pass through the Firewall64. If the Firewall64 inspects the sequence number of packets and drops packets whose sequence number does not fall into the TCP receive window, we would not be able to receive the packets at end hosts. However, in our test, we could receive all packets regardless of the sequence number, which means that none of the tested firewalls inspected the sequence number. Thus, an adversary would not have to infer the sequence number of the target connection for traffic injection. We utilized this loophole for the DoS attacks described in §5.2 and §5.3.

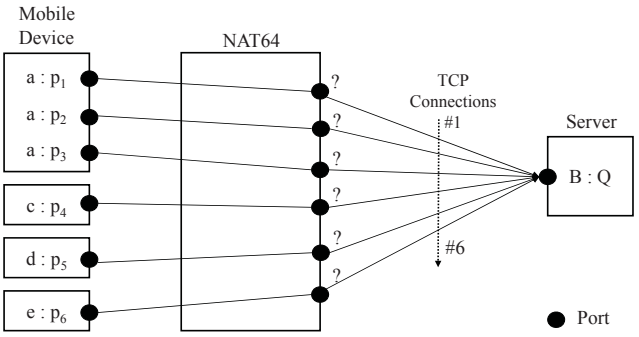


Figure 5: TCP connections established to identify NAT mapping patterns. An end device uses the local IP address a, c, d, and e with different port number (p_1 - p_6) respectively. The server uses a fixed IP address and a port number (B : Q). The pattern of mapped address and port number for each connection on NAT64 determines the NAT mapping pattern.

Mapping pattern of NAT64. When an end device inside the cellular network initially connects to an external host, NAT translates the IP address and port number of the device to its own external values, and creates a 5-tuple entry on its external mapping table. This external mapping varies depending on the policy of NAT, thus each NAT presents different mapping patterns. If NAT shows a predictable mapping pattern, an adversary would be able to identify target mappings from the pattern, which would enable her to send malicious packets using the identified mappings.

We analyzed the mapping pattern of NAT64 by repeatedly re-establishing connections and observing the variation of the external IP address and port number recorded by the NAT64. Figure 5 depicts an overview for identifying NAT mapping patterns. We first sent a *TCP-SYN* packet from an end device to an external server located on the IPv4 network so that NAT64 created a 5-tuple entry of this connection on its mapping table. Then we reset the mapping by sending a *TCP-RST* packet from the server to the NAT64. While repeating this procedure three times in each trial, we recorded the external IP address and port number of the NAT64 at the server, and checked if we could infer the mapping pattern. Note, we assumed that a user connects to a known server with the fixed destination IP and port number (B : Q), and the operating system of the end device randomly assigns a source port (p_1 - p_6) to every new connection. Moreover, we also considered the case that the source IPv6 address of the device is changed. Thus, we used different local IP addresses (a, c, d, and e) for the device.

Table 6 presents the results of the NAT mapping patterns for each cellular operator. All the external port number of NAT64 varied randomly, thus we learned that identifying a NAT mapping for a connection by only using known information is not feasible. Nevertheless, from this test, we identified that the external IPv4 address is changed when the source IPv6 address of a device is changed, and this characteristic can amplify the *NAT bricking attack* (in §5.4). Furthermore, since a large number of IPv6 addresses are mapped to a single IPv4 address as described in §3.3,

TABLE 6: Observed NAT mapping pattern (IP address : port number mappings)

#	From	To	OP-II	OP-III	OP-IV
1	a : p_1	B : Q	A_1 : F	A_2 : L	A_3 : T
2	a : p_2	B : Q	A_1 : G	A_2 : M	A_3 : U
3	a : p_3	B : Q	A_1 : H	A_2 : N	A_3 : V
4	c : p_4	B : Q	C_1 : I	C_2 : O	C_3 : W
5	d : p_5	B : Q	D_1 : J	D_2 : R	D_3 : X
6	e : p_6	B : Q	E_1 : K	E_2 : S	E_3 : Y

this *N-to-1 mapping* property can be exploited by an adversary, and the attack scenarios are described in §5.3 and §5.4.

Closing TCP connections and removing NAT mappings. When connection between a client and a server is terminated, NAT removes the mapping for the connection to recycle resources for other mappings. Although TCP manages its own state, some NAT middleboxes in cellular networks cannot receive control packets (e.g., *TCP-FIN* or *TCP-RST*) due to out of the network coverage or even the fact that some do not support connection state management. Therefore, most NAT middleboxes utilize a timeout for a TCP mapping. When the NAT no longer receives packets for a TCP flow within a given timeout, it drops the mapping entry. Similarly, NAT also utilizes a timeout for UDP, because UDP does not provide state management.

However, removing NAT mappings based on a timeout can lead to erroneous results. Once NAT removes its mapping entry (when it is timed out), the connection using the removed mapping cannot be correctly translated to an external IP address and a port number. Consequently, communication between the end host and the external server is disrupted. Recreating a mapping requires the device to send another initial packet to the external server to open a mapping. When NAT does not remove the mapping entry even after a TCP connection is normally terminated, an adversary can send malicious packets to internal devices.

Therefore, Wang *et al.* [40] pointed out that removing a NAT mapping should be conducted cautiously. They considered the flaws associated with remaining NAT mappings for TCP connections in the *CLOSED* state. In contrast, we focused on removing NAT mappings for TCP connections in the *ESTABLISHED* state, in which NAT mappings for TCP connections are removed when the connections are timed out or closed. Thus, we first investigated a timeout value for NAT64 of each operator. We measured this timeout value by gradually increasing the idle time for a TCP connection. When the communication using the connection is not available, we regard this as a timeout value. As a result, OP-III and OP-IV used 1 hour as a timeout value for TCP connections whereas OP-II used only 5 minutes. We utilized these aspects to design the *NAT overflow attack* (in §5.2).

Next, we focused on how an adversary would be able to intentionally remove the mappings used by TCP connections in the *ESTABLISHED* state. For this, we utilized a *TCP-RST* to remove the NAT mapping. We first established a TCP connection from an end device in the IPv6 cellular network to a host on the external IPv4 Internet, which passed through NAT64. Using this, we could obtain the (external) 5-tuple mapping on NAT64. Then, we sent a *TCP-RST* packet from another host on the Internet to NAT64. Note that we do not need to consider the sequence number of the *TCP-RST* packet because none of the tested Firewall64s inspect the sequence number of the packets. Furthermore, although the end device receives the *TCP-RST* packet, it does not close its connection since the sequence number of the packet is not in the TCP receive window. However, when the device tries to communicate with the external host, because the corresponding mapping is removed at the NAT64, none of the packets are transmitted to the external host. As a result, the communication is blocked and the end device sends TCP retransmission packets repeatedly for a certain period to reconnect to the external host. To verify this, we sent a data packet and checked whether the packet can pass through the NAT64 after the device received the *TCP-RST*. This experiment confirmed that all the operators removed the mapping as soon as we sent a *TCP-RST* packet, and this can be utilized for a *NAT wiping attack* (in §5.3).

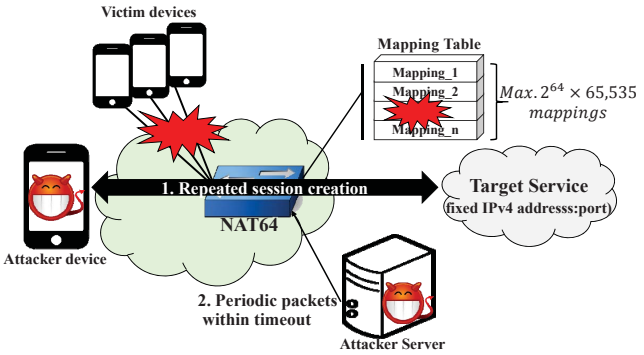


Figure 6: Overview of the *NAT overflow attack*

5.2. NAT overflow attack

In IPv4 cellular networks, because most major operators drop source IP address spoofed packets and use full IPv4 address-based routing, a device can send and receive data packets with only a single private IPv4 address allocated by NAT. Accordingly, the maximum number of external mappings for a single target service can be 65,535 (only the source port of the 5-tuple varies within the range from 1 to 65,535). However, in IPv6 cellular networks, a device can utilize 2^{64} IPv6 addresses. If the device creates mappings on NAT64 for a service using all 2^{64} of IPv6 addresses, $2^{64} \times 65,535$ mappings can be created and this can lead to a huge amount of overhead for NAT64. Based on this insight, we created a large number of mappings using only a single device and observed the response and status of NAT64. Note, we first experimented on the testbed which was identical to the corresponding commercial network.

An overview of this test is depicted in Figure 6. To reduce the overhead of the end device, we only created sessions on NAT64, rather than on the device. Thus, we first

sent a *TCP-SYN* packet using a raw socket and waited for a *TCP-SYN/ACK* packet from the target service, which uses a fixed IP address and port number. When the *TCP-SYN/ACK* packet was observed at the device, we sent a *TCP-ACK* packet to complete the TCP three-way handshake. Because we used a raw socket to send the *TCP-SYN* and *TCP-ACK* packet rather than the socket provided by the operating system, no sessions were created on the device, but NAT64 considers it to be the same as the normal TCP three-way handshake. Consequently, the mapping for this connection was created on NAT64. This mapping information was also stored at an external server in order to prevent removal of this mapping information because of the timeout. Using this information, we could periodically send a packet within the timeout period for each operator. While we sent *TCP-SYN* and *TCP-ACK* packets, we changed their source IP address (more accurately the IID of source IP address) and source port number for each connection; thus their mapping was also created on NAT64 in the same way. When the number of mapping entries to the target service exceeded 1,500, we discovered that the tested NAT64 created no further mappings for the target service and sent a *TCP-RST* packet as a response to the received *TCP-SYN* packet. In this case, none of the devices using the same NAT64 could establish any more sessions for the target service, but they are allowed to create sessions for other services using different IP address. From this result, we learned that creating an extremely large number of mappings with a single device is impossible due to the restriction of the mapping creation policy on NAT64.

However, we also found that this policy can be exploited by an adversary for a DoS attack. If an adversary deliberately creates over 1,500 sessions for a target service and keeps these sessions alive, no more sessions can be established by any other users. Based on this, we first carried out a DoS attack in the testbed which was identical to the commercial network. An adversary's device sends a *TCP-SYN* packet to the test server (we targeted an SMTP service), and after receiving a *TCP-SYN/ACK* packet from the server, it transmits a *TCP-ACK* packet. We repeated this process by changing the IID of the source IP address and source port number until the number of created sessions exceeded 1,500. Under this attack, victim devices connected to the same NAT64 could not use the targeted SMTP service. Moreover, an HTTP web service running on the same server (using the same destination IP address but different port number) could not be served as well. Therefore, we found that NAT64 prevented the number of mapping entries with the same destination IP address from exceeding 1,500 mapping entries, regardless of the port number.

To prove the attack in real environments, we tested operational cellular networks. Since an actual attack may cause severe damage to both the cellular network and users, we setup our own server and only checked whether our end device inside the cellular network was able to create more than 1,500 sessions with the same server. Note, creating 1,500 mappings imposed negligible overhead on NAT64. The analysis revealed that all the operators follow the same NAT64 policy; sessions to one destination IP address cannot exceed 1,500 regardless of port numbers. This shows that a policy designed to prevent a large number of mappings can also lead to a different kind of DoS attack.

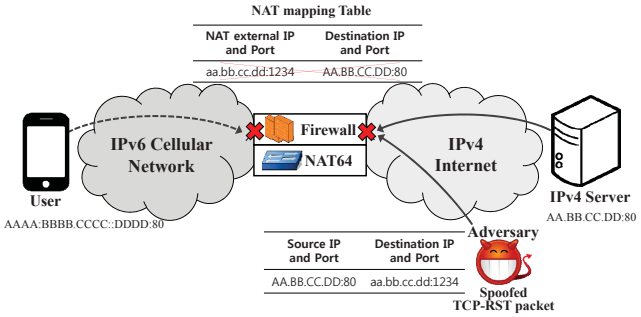


Figure 7: Overview of the *NAT wiping attack*

In summary, the proposed NAT overflow attack requires no state in the device and the server, while creating infinite number of states in the NAT box. Therefore, a DoS attack on the NAT box is always possible with or without limiting the mapping entries per server. With such a policy, we can block connections to a server. Without it, we can infinitely increase the state overhead of the NAT box. We suggest a clear solution for this problem in §6.

5.3. NAT wiping attack

As we discussed in §3.3, the *N* value of *N-to-1* mapping on NAT64 is much higher than NAT in IPv4 cellular networks. As a result, if an adversary targets an external IPv4 address of NAT64, *N* hosts sharing the same external IPv4 address can be exposed to a DoS attack on NAT middleboxes. The adversary could launch the attack by first identifying the TCP 5-tuple for the mappings used by the target service, before sending a number of malicious *TCP-RST* packets to wipe out the target mappings. Once the attack is successfully launched, the target mappings are removed from NAT64; thus, all users on that NAT64, who are mapped to the same external IP address, are denied access to the service. Therefore, a *NAT wiping attack* completely destroys the availability of the target service. Figure 7 shows an overview of the *NAT wiping attack*. To conduct the attack, we need to identify the TCP 5-tuple of the target service, as described below:

Protocol. Because the *NAT wiping attack* targets a TCP connection, we focus on the TCP protocol.

Destination IP address and port number. An adversary can easily obtain the IP address and port number of the target service, since the IP addresses and port number of popular services are already well known and some are fixed (e.g., TCP 80 for a web service). Therefore, we assume these values are already known to adversaries.

External IP address of NAT64. The adversary can easily figure out the range of external IP address of NAT64 by utilizing the method (Airplane mode) described in §3.3. Note, the adversary can target more external IP addresses of NAT64 depending on her resources (e.g., bandwidth).

External port number of NAT64. As described in §5.1, it is difficult to predict the port number only based on map-

ping patterns because it varies randomly whenever a new connection is established. However, since the port number is allocated within the range 1-65,535 (2 bytes), an adversary can use brute force to determine the port number. In other words, she can send *TCP-RST* packets to all port numbers from 1 to 65,535. Note, because the size of a *TCP-RST* packet is 40 bytes, it would take less than 0.2 seconds to knock all the ports with a 100 Mbps network (40 bytes $\times$ 8 bits/byte(s) $\times$ 65,535 / 100 Mbps = 0.1999 seconds).

After identifying the TCP 5-tuple, an adversary can launch the *NAT wiping attack*. We conducted the attack against our device in the IPv6 cellular network, which was already connected to our server on the IPv4 Internet. We reduced the side effects caused by the attack by targeting only a single external IP address of NAT64 mapped to our device. After launching the attack, the device seemed to be unaffected and was still in the *ESTABLISHED* state. However, when the device attempted to send packets to the server, none of the packets passed through NAT64, such that the device had to retransmit the data several times. Eventually, it reset the connection and tried to create a new connection. At this moment, because we performed the attack continuously, it repeatedly blocked the new connection by immediately removing the new mapping entry. Note that we performed the attack with 100 Mbps link, allowing a new mapping to be wiped out every 0.2 seconds. As a result, the device was prevented from even completing a three-way handshake.

In our test, we only targeted a single external IP address of NAT64 to reduce the side effects that may be caused by the attack. However, in the real world, it would be possible for the adversary to target the mappings of long-lived connections (e.g., PUSH [3]) using multiple external IP addresses of NAT64. A brief calculation allows us to determine that with a 100 Mbps link, the adversary would be able to block a service using 50 external IP addresses within 10 seconds. In this case, about 480 hosts (9.62 $\times$ 50) in the network of the OP-II operator or 790 hosts (15.87 $\times$ 50) in OP-IV would experience service disruption.[1] Note, sending *TCP-RST* packets every 10 seconds is adequate to disrupt the availability of the service. Since the time required for several data retransmissions and the mapping re-creation took over 10 seconds, this is sufficient to perform the *NAT wiping attack*. Of course, it may differ depending on the application using the target service, but if the adversary would be able to allocate more seconds or more bandwidth for sending *TCP-RST* packets, many more hosts would experience service disruption.

5.4. NAT bricking attack

In this section, we introduce another type of DoS attack known as a *NAT bricking attack* which also efficiently exploits the high *N* value of *N-to-1* mapping on NAT64. Services that do not require user accounts (e.g., Google web search, and YouTube video streaming), usually deny malicious users based on the source IP address. However, an adversary on the Internet can abuse this IP-based blacklisting to launch another type of DoS attack against normal users behind NAT as they share the same public IP address assigned by NAT64. This problem may also occur for IPv6-based cellular networks with a high *N* value, as described in §3.3. To demonstrate this, we perform our experiment on Google Scholar [19], which is an IPv4-based online search engine for academic literature.[2] The service allows users to cite these resources conveniently by providing a *citation importing feature* that supports several formats of bibliography managers. However, different from its own purposes, the resources are being heavily crawled by bots. To prevent such activities, Google Scholar adopted *CAPTCHA* to differentiate bots from normal users, but details of the prevention mechanism are not known. Thus, to identify the exact conditions under which crawling behavior is detected, we implemented a program that sends queries to Google Scholar. When we sent 150 queries within a minute, *CAPTCHA* was triggered, and after repeatedly sending additional queries without responding to *CAPTCHA*, our host was denied access for around two days. This is because Google Scholar might consider our host as a bot. Note that this experiment was conducted by one of our desktop PCs connected to the Internet, and sending 150 queries per minute is a sufficiently low rate as to neither disturb the network bandwidth nor Google Scholar.

This indicated that the bot prevention scheme on Google Scholar is based on the IP address; thus, an adversary behind NAT in cellular networks would be able to cause DoS for other mobile users. We named this attack the *NAT bricking attack*. We demonstrated this attack by utilizing three mobile devices that are legitimately connected to the IPv6 cellular network in OP-III; hence, should these devices have to use IPv4-based services, NAT64 would translate their IPv6 addresses to a public IPv4 address. The attack was carried out by instructing one of our devices to behave like an adversary by sending 150 query messages per minute to Google Scholar to trigger a *CAPTCHA* request. With the other devices, as victims, we used Google Scholar in the normal way. On receiving a *CAPTCHA* request, we changed the external IP address of the adversary device by turning Airplane mode on and off. Then, the device repeatedly sent query messages to trigger the *CAPTCHA* request again. During this time, we conducted a *NAT bricking attack* by turning Airplane mode on and off 1,000 times. As a result, the adversary device could trigger *CAPTCHA* for a total of 631 external IPv4 addresses from Google Scholar including one of the victim's external IP address. For validation, we simply confirmed that the *CAPTCHA* request is also informed of the victim device. The *CAPTCHA* request was also observed at the victim device when using the Google Scholar service. Of course, when the victim changed its external IP address, the *CAPTCHA* request was no longer observed. Note that we stopped sending additional queries right after receiving the CAPTCHA, to prevent the external IP address from being

1. *N* value from Table 2 is used for the calculation

2. We used DNS queries to check that the Google scholar service only uses IPv4 addresses.

denied access to the service. This experiment indicated that we could trigger the *CAPTCHA* request within a minute. Therefore, we showed that a *NAT bricking attack* is a simple yet powerful attack because only a single device is required to prohibit roughly around 6,000 users within a day ($6,542\,\text{users/day} = 631\,\text{external IP addr.}/1,000\,\text{trials} \times 7.2\,\text{users/external IP addr.} \times 1\,\text{trial/min} \times 1,440\,\text{min/day}$).

6. Countermeasures and Discussion

In this section, we discuss further consideration on the security of middleboxes in IPv6 cellular networks and propose short- and long-term mitigation strategies for the security problems.

Scanning, over-billing, and traffic flooding attacks stem from the absence of a firewall or appropriate filtering rules at the firewall in the early stage of IPv6 deployment. In IPv4-only cellular networks, NAT plays an additional role as a firewall to verify the incoming traffic based on a NAT mapping table. However, in IPv6 networks, the NAT function is no longer required except to provide backward compatibility with legacy IPv4 infrastructure. Thus, unless the operators give adequate thought to their security implications, they would suffer from security problems such as scanning, over-billing, and traffic flooding attacks. In addition, this can lead to a DDoS attack on the core network as well. If the amount of traffic into the IPv6 cellular network exceeds the capacity of core components (e.g., a gateway or NAT), the users would no longer be able to use the network. Note, operators usually consider an over-billing problem to be more serious than undercharging, and they are afraid of an out-of-service state resulting from a DDoS attack.

The security problem in IPv6 cellular networks can be addressed by implementing a stateful firewall that monitors each state of a flow and blocks unsolicited incoming traffic from the Internet. The stateful firewall would enable operators to alleviate the problems including DDoS attacks against IPv6 cellular networks. Even with the firewall blocking unsolicited incoming traffic, an adversary could still send malicious traffic from inside a cellular network such as scanning traffic; thus, operators should also consider traffic originating from inside. Moreover, by configuring their P-GWs to utilize their IPv6 address range randomly, operators could prevent an adversary from easily narrowing the range to conduct a scanning attack. Note that even after this configuration, the DDoS attack might be possible when a mobile device inside the cellular network misuses 2^{64} IIDs, thereby helping an external attacker to establish the state using a hole punching technique [14, 20]. The following solution for a *NAT overflow attack* can be used to resolve this problem.

NAT overflow attack originates from the difference between a part of IPv6 address that are actually used on a cellular network and the Internet. An IPv6 cellular network only utilizes the /64 prefix, whereas NAT64 interconnecting IPv4 Internet utilizes the full IPv6 address. Accordingly, an adversary could freely utilize the IID part of IPv6 address. In our work, we only succeeded in creating 1,500 mappings due to the restriction policy of NAT64, but it also led to another type of DoS attack. If there was no restriction, we expect an adversary to be able to impose a huge overhead on NAT64. This problem can be alleviated by using an identical part of IPv6 address in both the cellular network and the Internet. By allowing NAT64 to utilize only a /64 prefix for mapping creation or by allowing cellular operators to utilize a full IPv6 address with limitations on the number of IIDs used concurrently, cellular operators could prevent an adversary from performing a *NAT overflow attack*, because she is not able to create a large number of mappings.

NAT wiping and **NAT bricking attacks** differ from the previous two attacks because they originate from the natural characteristics of NAT64. This fate-sharing problem happens because of scarce IPv4 addresses. If everyone were to use IPv6, this problem would not occur anymore. Therefore, this problem remains as long as the IPv4 network exists. Because the effectiveness of the attacks depends on the *N* value of *N-to-1* mapping, this problem can be alleviated by lowering the *N* value, but it would not be a fundamental solution.

A *NAT wiping attack* is more efficient when an adversary can predict a NAT mapping for a connection. In this work, we performed an exhaustive search to identify the TCP 5-tuple of target mapping because of the random mapping pattern. However, several NAT middleboxes have the property of predictable NAT mapping patterns that need to support NAT traversal. In this case, the adversary can easily identify the TCP 5-tuple. Moreover, an adversary with significant resources (e.g., a bot master) can target the entire IPv4 address range of NAT64. In this case, all the users using the target NAT64 could be victims of the attack. A *NAT wiping attack* can be prevented by inspecting the TCP sequence number properly. Although TCP sequence number verification is an optional functionality, when the firewall is located at the edge of a network separated by NAT, the firewall should verify that the TCP sequence number falls in the TCP receive window of the current connection. Otherwise, the hosts in the network could be the victims of a *NAT wiping attack*. Note, scanning, over-billing, traffic flooding, and *NAT wiping attacks* can be mitigated by the deployment of a stateful firewall, but it should be kept in mind that the deployment of new firewalls in cellular networks could have a significant effect on performance, energy consumption, and security.

In terms of the *NAT bricking attack*, service providers should understand that a source IP address based blacklisting is not a good solution to protect their services. As a *NAT bricking attack* is based on the application level and caused by the defense mechanism of the service providers, they are more likely to be addressed by the service providers. In addition, the service providers can utilize various approaches to prevent attacks against their services. For instance, when malicious behavior is detected, the provider can require the user to login to their account by denying access to the service without the login session.

TABLE 7: Summary of attacks (✓: feasible, ✘: infeasible, △: feasible but inefficient, -: no NAT64)

Attacks	Features for attacks	Affected middleboxes	Operators				
			OP-I	OP-II	OP-III	OP-IV	OP-V
Scanning	No access control of firewalls and the /64 prefix based data transmission	firewall	△	✓	✓	✓	✘
Over-billing			△	✓	✓	✓	✘
Traffic Flooding			✘	✓	✓	✓	✘
NAT Overflow	Free use of the IID	NAT64	-	✓	✓	✓	-
NAT Wiping	High *N* value of *N-to-1 mapping*		-	✓	✓	✓	-
NAT Bricking			-	✓	✓	✓	-

7. Summary and Conclusion

Throughout this work, we developed six attacks that exploit the properties of IPv6 middleboxes and the features of IPv6 cellular networks. Table 7 presents a summary of these attacks. In OP-II, OP-III, and OP-IV, all the proposed attacks were feasible (marked as ✓). Additionally, since OP-I and OP-V do not use 464XLAT architecture, we could not conduct *NAT overflow*, *NAT wiping*, and *NAT bricking attacks* (marked as '-'). In OP-I, because the firewalls filtered out only incoming traffic from the Internet, and not traffic from the cellular network, scanning and over-billing attacks would still be available if the adversary launches the attacks inside the cellular networks. Because an adversary can launch this attack from the inside, these attacks against OP-I are less efficient in terms of both bandwidth and cost (marked as △). In addition, a traffic flooding attack on OP-I was infeasible because of the access controls provided by a firewall (marked as ✘). In OP-V, since firewalls are used to filter out malicious traffic properly, none of the attacks were feasible (marked as ✘).

As a conclusion, transition to IPv6 in cellular networks opens new set of vulnerabilities. This is not different from recent trends in cellular networks: VoLTE [22, 24] and CSFB [38, 39] have been shown to present serious problems as well. We hope cellular operators, middlebox developers, and even 3GPP standard writers to consider security once more before deploying a new technology.

Acknowledgment

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2017-2015-0-00403) supervised by the IITP (Institute for Information & communications Technology Promotion).

References

[1] 3GPP. General Packet Radio Service (GPRS); Service description; Stage 2. TS 23.060, 3GPP, 2008.

[2] 3GPP. Interworking between the Public Land Mobile Network (PLMN) supporting packet based services and Packet Data Networks (PDN). TS 29.061, 3GPP, 2015.

[3] Apple. Apple Push Notification Service (APNs). https://developer.apple.com/notifications/.

[4] Apple. Supporting IPv6 in iOS 9. https://developer.apple.com/news/?id=08282015a.

[5] M. Bagnulo, P. Matthews, and I. van Beijnum. Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers. RFC 6146, IETF, 2011.

[6] M. Bagnulo, A. Sullivan, P. Matthews, and I. van Beijnum. DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers. RFC 6147, IETF, 2011.

[7] S. M. Bellovin, B. Cheswick, and A. Keromytis. Worm propagation strategies in an IPv6 Internet. *LOGIN: The USENIX Magazine*, 31 (1), 2006.

[8] A. Biggadike, D. Ferullo, G. Wilson, and A. Perrig. NATBLASTER: Establishing TCP connections between hosts behind NATs. In *ACM SIGCOMM Asia Workshop*, 2005.

[9] B. Carpenter. Internet Transparency. RFC 2775, IETF, 2000.

[10] T. Chown and S. Venaas. Rogue IPv6 Router Advertisement Problem Statement. RFC 6104, IETF, 2011.

[11] J. Czyz, M. Luckie, M. Allman, and M. Bailey. Don't Forget to Lock the Back Door! A Characterization of IPv6 Network Security Policy. In *Network and Distributed System Security Symposium (NDSS)*, 2016.

[12] G. V. de Velde, T. Hain, R. Droms, B. Carpenter, and E. Klein. Local Network Protection for IPv6. RFC 4864, IETF, 2007.

[13] Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast Internet-wide Scanning and Its Security Applications. In *22nd USENIX Security Symposium*, 2013.

[14] B. Ford, P. Srisuresh, and D. Kegel. Peer-to-peer communication across network address translators. In *USENIX Annual Technical Conference, General Track*, pages 179–192, 2005.

[15] M. Fujii. Evolution of Mobile Networks and IPv6. https://www.apnic.net/events/apnic-speakers/presentations/other/files2/2014-04-25-apectel49-APNIC.pdf/, 2014.

[16] Y. Go, J. Won, D. F. Kune, E. Jeong, Y. Kim, and K. Park. Gaining Control of Cellular Traffic Accounting by Spurious TCP Retransmission. In *Network and Distributed System Security Symposium (NDSS)*, 2014.

[17] F. Gont and W. Liu. Security Implications of IPv6 options of Type 10xxxxxx. Internet-draft, IETF, 2013.

[18] Google. Google Cloud Messaging (GCM). https://developers.google.com/cloud-messaging/.

[19] Google Scholar. Google Scholar. https://scholar.google.com.

[20] M. Holdrege and P. Srisuresh. Protocol complications with the ip network address translator. RFC 3027, IETF, 2001.

[21] Internet Society. Case Study: T-Mobile US Goes IPv6-only Using 464XLAT. http://www.internetsociety.org/deploy360/resources/case-study-t-mobile-us-goes-ipv6-only-using-464xlat/.

[22] H. Kim, D. Kim, M. Kwon, H. Han, Y. Jang, D. Han, T. Kim, and Y. Kim. Breaking and Fixing VoLTE: Exploiting Hidden Data Channels and Mis-implementations. In *ACM Conference on Computer and Communications Security*, 2015.

[23] J. Korhonen, J. Soininen, B. Patil, T. Savolainen, G. Bajko, and K. Iisakkila. IPv6 in 3rd Generation Partnership Project (3GPP) Evolved Packet System (EPS). RFC 6459, IETF, 2012.

[24] C.-Y. Li, G.-H. Tu, C. Peng, Z. Yuan, Y. Li, S. Lu, and X. Wang. Insecurity of Voice Solution VoLTE in LTE Mobile Networks. In *ACM Conference on Computer and Communications Security*, 2015.

[25] X. Li, C. Bao, and F. Baker. IP/ICMP Translation Algorithm. RFC 6145, IETF, 2011.

[26] M. Luckie. Scamper: a scalable and extensible packet prober for active measurement of the internet. In *ACM SIGCOMM conference on Internet measurement*, 2010.

[27] M. Mawatari, M. Kawashima, and C. Byrne. 464XLAT: Combination of Stateful and Stateless Translation. RFC 6877, IETF, 2013.

[28] NetworkMonitorMini. Network Monitor Mini. https://play.google.com/store/apps/details?id=info.kfsoft.android.TrafficIndicator.

[29] C. Peng, C.-y. Li, G.-H. Tu, S. Lu, and L. Zhang. Mobile data charging: new attacks and countermeasures. In *ACM SIGSAC Conference on Computer and Communications Security*, 2012.

[30] C. Peng, C.-Y. Li, H. Wang, G.-H. Tu, and S. Lu. Real Threats to Your Data Bills: Security Loopholes and Defenses in Mobile Data Charging. In *ACM Conference on Computer and Communications Security*, 2014.

[31] Z. Qian and Z. M. Mao. Off-path TCP sequence number inference attack-how firewall middleboxes reduce security. In *IEEE Symposium on Security and Privacy (S&P)*. IEEE, 2012.

[32] Z. Qian, Z. M. Mao, and Y. Xie. Collaborative TCP sequence number inference attack: how to crack sequence number under a second. In *ACM Conference on Computer and Communications Security*, 2012.

[33] Y. Rekhter, R. G. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Address Allocation for Private Internets. RFC 1918, IETF, 1996.

[34] L. Rizzo. Netmap: a novel framework for fast packet I/O. In *21st USENIX Security Symposium*, 2012.

[35] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN - Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs). RFC 3489, IETF, 2003.

[36] M. Smith and D. Loguinov. Enabling highperformance internetwide measurements on windows. In *Passive and Active Measurement*. Springer, 2010.

[37] P. Traynor, M. Lin, M. Ongtang, V. Rao, T. Jaeger, P. McDaniel, and T. La Porta. On cellular botnets: measuring the impact of malicious devices on a cellular network core. In *ACM Conference on Computer and Communications Security*, 2009.

[38] G.-H. Tu, C. Peng, H. Wang, C.-Y. Li, and S. Lu. How voice calls affect data in operational LTE networks. In *international conference on Mobile computing & networking*, 2013.

[39] G.-H. Tu, Y. Li, C. Peng, C.-Y. Li, H. Wang, and S. Lu. Control-plane protocol interactions in cellular networks. In *Proceedings of the ACM conference on SIGCOMM*, 2014.

[40] Z. Wang, Z. Qian, Q. Xu, Z. Mao, and M. Zhang. An untold story of middleboxes in cellular networks. In *ACM SIGCOMM Computer Communication Review*, 2011.

[41] Whatsapp. WhatsApp Messenger. https://web.whatsapp.com/.

[42] X. Yang, T. Ma, and Y. Shi. Typical dos/ddos threats under ipv6. In *IEEE ICCGI*, 2007.

[43] B. Zhao, C. Chi, W. Gao, S. Zhu, and G. Cao. A chain reaction dos attack on 3G networks: Analysis and defenses. In *IEEE INFOCOM*, 2009.