КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 4 (24), 2024
ISSN 2663 - 4023
DOI 10.28925/2663-4023.2024.24.321340
UDC 004.72:004.77

Maksym Kotov
Master's degree in computer and information systems security,
student at the Department of Cybersecurity and Information Protection
Taras Shevchenko National University of Kyiv, Kyiv, Ukraine
ORCID ID: 0000-0003-1153-3198
maksym_kotov@ukr.net

Serhii Toliupa
Doctor of Sciences, professor, professor of Department of
Cybersecurity and Information Protection
Taras Shevchenko National University of Kyiv, Kyiv, Ukraine
ORCID ID: 0000-0002-1919-9174
tolupa@i.ua

METHODS OF BUILDING DURABLE UDP PORT MAPPINGS IN A NAT-BASED ENVIRONMENT

Abstract. Staying abreast with User Datagram Protocol (UDP) has become more crucial in modern digital networks, which are continuously expanding and becoming more intricate. Maintaining UDP mappings in a NAT-based environments, reliable and uninterrupted communication for various duties, such as expeditiously transmitting data and establishing secure connections via virtual private networks (VPNs) like WireGuard is of utmost importance. Network Address Translation (NAT) is an important part of protecting the limited number of global Internet Protocol (IP) addresses and making networks safer by hiding how private communication networks are set up on the inside. However, NAT presents a number of challenges, one of which is the dynamic assignment of port numbers, which has the potential to result in disruptions in connections. The objective of this article is to elaborate on the functioning of WireGuard, placing particular emphasis on the criticality of dependable UDP mappings in order to achieve peak performance. In addition, the paper examines VMware's Network Address Translation solution to illustrate the challenges associated with maintaining UDP mappings. In this article, an investigation is conducted into the many methods and current solutions that have been developed in order to mitigate said issues. Some of the strategies that have been implemented include the utilization of static port mapping in order to establish a reliable route through NAT, the extension of the Time to Live (TTL) for port mappings in order to reduce the number of connection disruptions, and the approach of sending empty UDP packets in order to keep active mappings. In addition, a novel solution is suggested: a protocol for managing NAT mapping that makes an effort to simplify the process of modifying the frequency of UDP probes by requiring NAT devices to disclose their TTL settings. The purpose of this protocol is to make NAT mapping easier to manage and more efficient in terms of overall network traffic.

Ключові слова: Network Address Translation (NAT); User Datagram Protocol (UDP); Virtual Private Networks (VPNs); WireGuard; VMware; persistent UDP mappings; port mapping Time to Live (TTL); static port mapping; NAT traversal techniques; NAT mapping support protocol; network reliability; network performance optimization.

INTRODUCTION

When it comes to network communications, it is hard to place enough emphasis on the significance of protocols and procedures that simplify the process of transmitting information across a variety of digital infrastructures.

An essential protocol that is utilized across a wide range of various domains is the User Datagram Protocol, which is also referred to as UDP. In spite of the fact that it does not

necessitate the additional function of ensuring delivery, it is well recognized for the ease with which it transmits data and the effectiveness with which it does so. Applications built on the UDP utilize it as a transport layer to establish reliable and highly effective transmissions. One example of such applications is WireGuard Virtual Private Networks. UDP is frequently used for two other applications such as transmission of real-time video streaming and the facilitation of online gaming [1] – [7].

Simultaneously, the widespread use of Network Address Translation in contemporary networking contexts introduces a different set of issues to the process of maintaining durable UDP mappings, which are needed for ongoing communication. In order to alleviate the lack of IPv4 addresses and increase security by masking internal IP addresses, network address translation is necessary [8] – [14].

Unfortunately, there is a mismatch between the fundamental features of Network Address Translation and the operational needs of programs that use User Datagram Protocol. Thus, this research is aimed to present an investigation of methods that are expressly designed to overcome these various limitations.

The purpose of this study is to conduct an analysis of the complexity of UDP mappings in NAT systems. Additionally, the goal is to present an overview of the operational ideas that underpin both UDP and NAT. In this research the implementation of NAT by VMware has been done as a case study in order to shed light on the practical challenges that are encountered when maintaining durable UDP mappings.

As an additional point of interest, this paper investigates a variety of proven and innovative techniques to address challenges related to persistent UDP port mappings. In addition to the strategic usage of empty UDP packets and the proposal of a unique NAT mapping support protocol, these include modifications to port mapping Time to Live and static port mapping. As a consequence of this, this article provides a comprehensive analysis of the approaches that aim to enhance the dependability and efficiency of networked applications when they are challenged with limits imposed by Network Address Translation (NAT).

WIREGUARD AS A UDP-BASED SERVICE EXAMPLE

We are going to go over each step that is necessary to establish a secure connection with the main server so that we can ensure that we have a complete grasp of how WireGuard works. The initiation of the operations begins with the creation of a virtual network interface. Considering that it employs a high level of encryption in order to carry out its functions, this interface is one of a kind. An IP address, which serves as the client's unique identification on the virtual network, is also provided to the client at the time of setup in order to protect the confidentiality of the communication that takes place between the client and the server. Furthermore, in order to establish a connection with other clients, it is necessary to possess their public keys, IP addresses, and occasionally other setup data. For the WireGuard server to successfully recognize a client trying to connect to it as a peer, only the client's public and preshared keys need to be added to the server's configuration beforehand [15] – [17].

After being activated, the WireGuard interface takes any IP packets that it receives and alters them. They are encapsulated into UDP packets, which is a method that tunnels the original data by wrapping it with additional headers of the UDP transport layer protocol, effectively transforming all the packets into datagrams. In addition to the encapsulation, the packet itself will be encrypted with the client's private key. This makes the data suitable for secure transmission across the internet to peers that have been chosen. When it comes to protecting

ISSN 2663 - 4023

one's integrity and privacy, this encapsulation and encryption layer is very crucial. It ensures that the information will be sent to the person who is supposed to receive it without being intercepted or altered in any way. Additionally, thanks to the encryption, it is not even possible for eavesdroppers to intercept the destination IP address [15].

The WireGuard operations flowchart is shown on the Fig. 1:

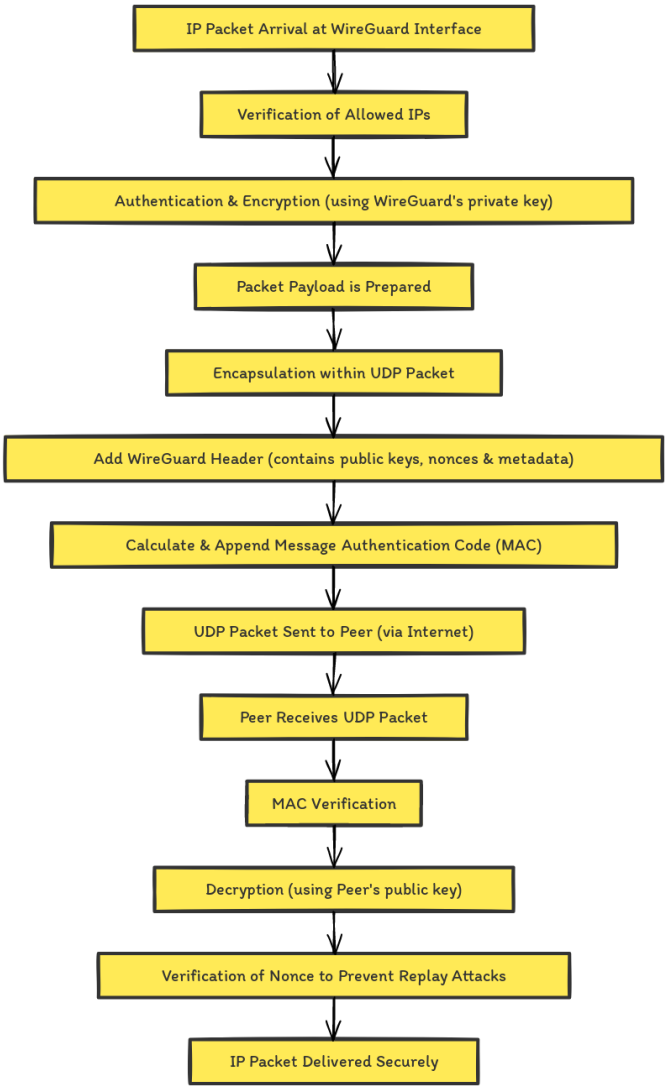

Fig. 1. WireGuard operations flowchart

The public IP address of the server that is responsible for providing the WireGuard encryption service should be known to clients in order for them to be able to connect with a WireGuard server. The WireGuard server itself is able to automatically determine the external address of each client. Due to the fact that this automated learning takes place the minute the server receives data from a client that has been correctly authenticated, it makes it possible for communication to continue in both ways without any interruptions occurring [15].

WireGuard demonstrates its flexibility by exhibiting its capacity to adjust to changes whenever abrupt interruptions occur in a network such as public IP address changes. WireGuard is able to handle this issue without any difficulty whatsoever, in contrast to traditional virtual private network systems, which are prone to encountering failure and disconnections whenever

ISSN 2663 - 4023

they are used. The server instantly notifies each client connected to the network of any changes occurring while the network is being updated, thus providing them with the ability to modify their configurations without worrying about losing their connection [15] – [17].

The WireGuard operations schema is shown on the following Fig. 2:

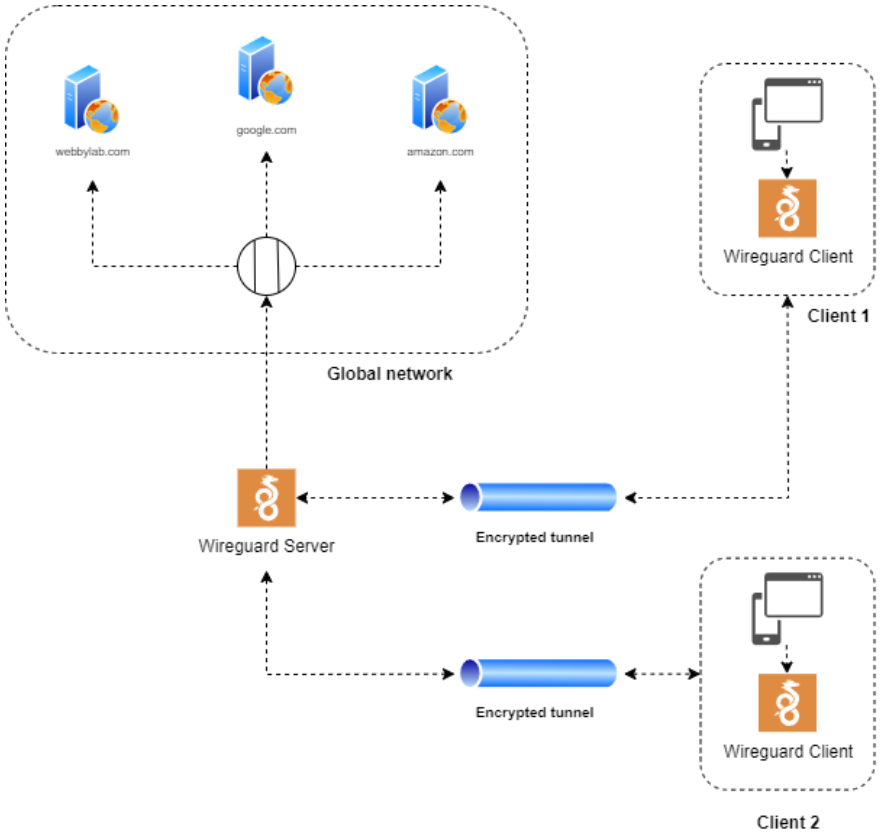

Fig. 2. WireGuard interactions

WireGuard is based on the fundamental notion of Cryptokey Routing, which also serves as the foundation for its operation. Through the utilization of this one-of-a-kind method, both the reception and transmission of packets are directed. A connection is made between the public key of a peer and each and every packet that is analyzed by WireGuard. The server performs an examination of the source field whenever an incoming packet is encrypted or processed in any other manner. If it is discovered that a packet matches an IP address that is contained within the peer's allowed IPs settings, then the packet is regarded to be valid and accepted. This dual verification technique, which is based on the legality of the packet as well as its conformity with the approved IPs, offers support for security measures [15] – [16].

WireGuard's architectural framework is distinguished by a combination of simplicity and ground-breaking innovations in the realm of security paradigms. This combination is what makes that framework so unique. There are a number of features that it embodies, some of which include, but are not limited to, robust encryption methods, a quick configuration procedure, and an efficient routing mechanism. Because of this combination, it is not only simpler to use, but it also guarantees a higher level of security, which makes it an appealing alternative for a wide range of clients to select from.

ISSN 2663 - 4023

UNDERSTANDING NAT THROUGH VMWARE'S IMPLEMENTATION

The birth of network address translation technology took place during the times when the global network began to emerge as a widespread utility. Such rapid development in the magnitude of connected clients simultaneously led to the problem of limited IP addresses. Due to the fact that there could not be more than 2^{32}, or 4,294,967,296 IPv4 addresses, engineers had to come up with a solution for the ever-increasing demand for growth. As a result, we have IPv6 that supports 2^{128} unique addresses, a number too big to write here. The new version of the protocol solved the issue for future clients and infrastructure, but the global network still consisted mostly of clients that did not support such protocols, and forcing a new version could potentially break the existing infrastructure [8] – [10].

As with every technology in the ever-evolving field of computer science, engineers had to find a temporary solution for everyone using the old version of the protocol. First and foremost, tunneling is used to hop IPv6 packets over the nodes that support only IPv4. Using such a technique, IPv6 packet gets temporarily incapsulated into an IPv4 packet. But the issue for businesses that still used IPv4 stayed with the limitations of the number of IP addresses. In addition to that, it is not desired to have every single server routed in the global network all the time. For some services, occasional access to external resources is enough. For these purposes, the NAT was created [11], [12].

The NAT operations flowchart is shown on the Fig. 3:

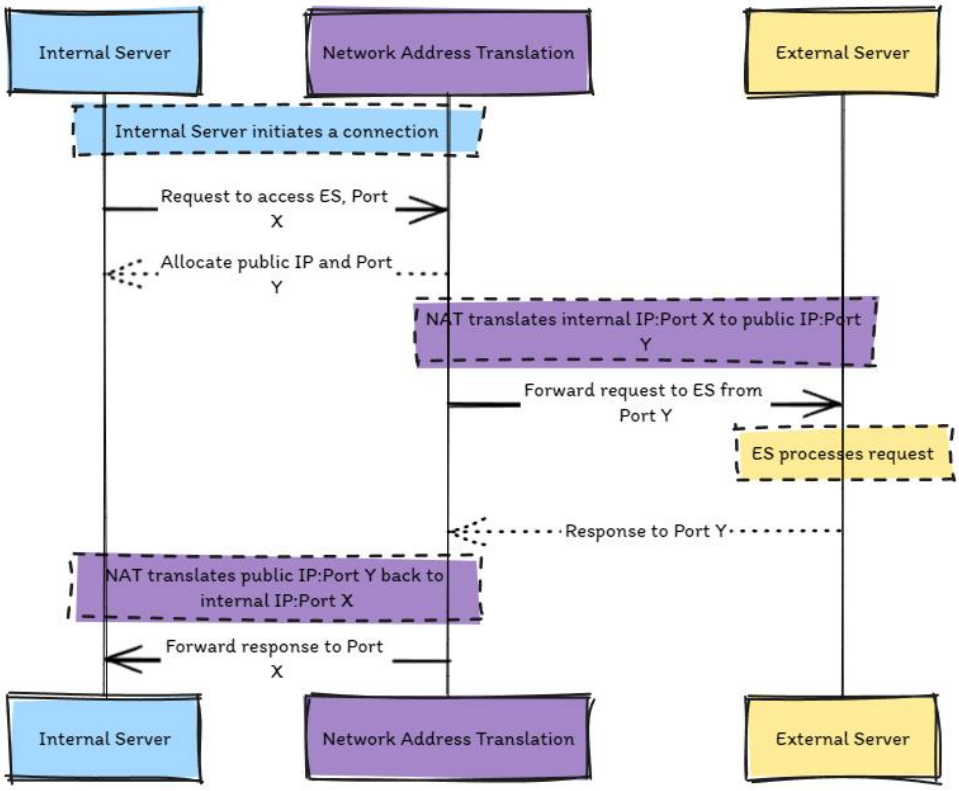

Fig. 3. NAT operations

Servers could use the private ranges of IP addresses for internal routing and for requesting external resources; the connection had to be established through a NAT. There are a few versions of NAT: some of them use a sliding window of IP addresses, and internal internal

ISSN 2663 - 4023

servers would then try to acquire a public address to access the global network; others link public IP addresses statically; but the most common version of NAT that is ubiquitous nowadays is a port mapping NAT.

When the server from the internal network tries to connect to the global network, it will send its traffic through NAT. Then NAT would associate one of its ports with this server, effectively establishing a connection. NAT will then use one of its other ports to establish the connection with the servers in the global network. In such a setup, NAT will sit exactly in the middle of communication between internal and external servers. When the response from the external server is received, NAT will use the saved combination of IP and port associated with the port that received the request from the internal server [12] – [14].

This setup, even though it provides a great way to manage the limited amount of address space and effectively provides a way for private servers to reach global resources, will still restrict external clients from accessing such servers. Thus, it could both serve as a great security solution and a limitation when the goal is to provide a service.

As virtualization was gaining momentum, VMware also provided their own implementation of the entire virtualized network, including virtualized DNS, DHCP, network segmentation, and most importantly for this paper — virtualized NAT [18].

The VMware Virtual Network schema is shown on the Fig. 4:

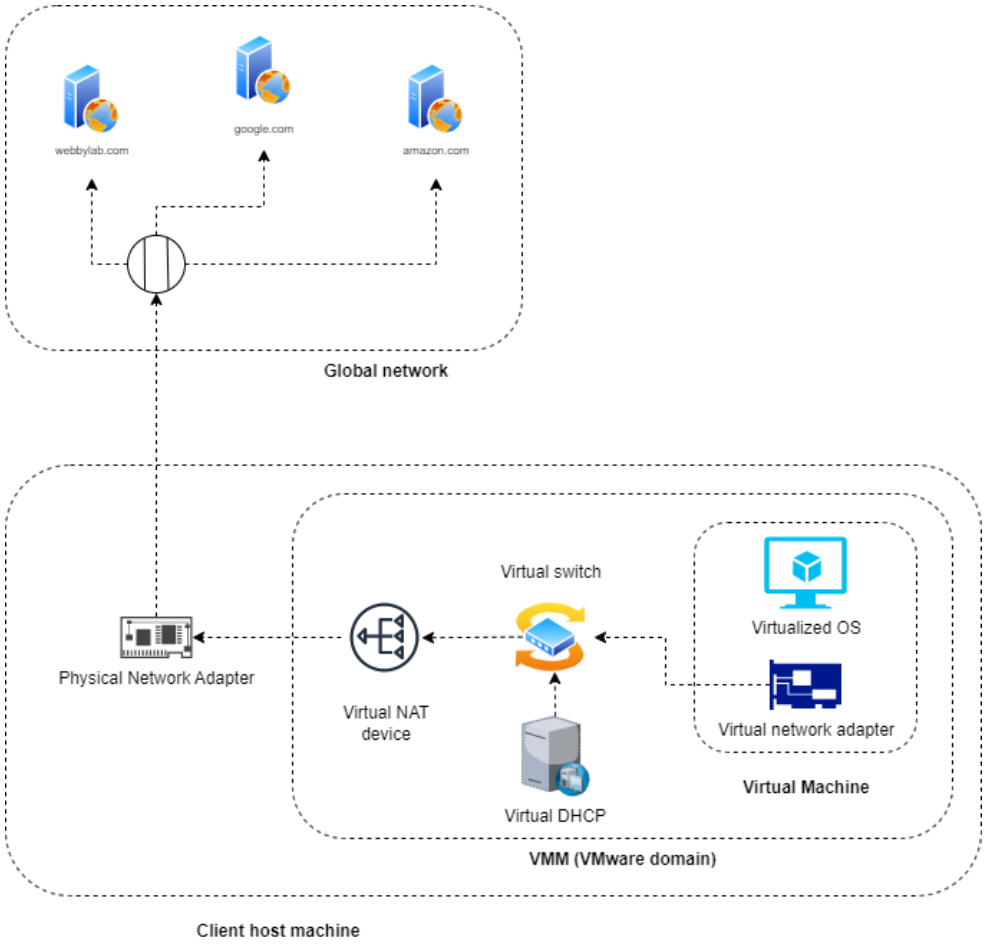

Fig. 4. VMware Virtual Network

Every Virtual Machine (VM) is connected to the internal virtualized implementation of the network by VMware. Virtualized services such as DHCP and DNS provide the basic network configuration needed to establish communication with both the host and other virtualized peers. A virtual switch facilitates such connections by providing data link layer commutation capabilities. This is a component that is capable of switching frames to the host's physical interface [18].

Even though the VM can access external resources, Network Address Translation mode will disguise all network activity as though it originated from the host Operating System (OS). Guests will be on a private subnet, with the host serving as a router and a firewall if configured. The VM will be assigned to a distinct subnet, much like most wireless home networks. For example, if the host computer's IP address is 192.168.0.104 and VM's is 192.168.121.100, the VM can access the outside network just like the host, but it is shielded from direct outside access.

EVALUETING THE IMPACT OF UNSTABLE PORT MAPPING

While using WireGuard client on host and NAT for virtual machines managed by VMware, connection drops or significantly slows down from time to time but could also be restored if the WireGuard client gets reloaded.

The assumption is that the connection will drop due to a change in the VMware NAT port. When the port changes, it's required to resend the request because the response won't get to the destination. It's important to remember that WireGuard encapsulates all its data in UDP datagrams [15] – [17]. Having said that, there is no guarantee of delivery on this layer, and such guarantees depend entirely on the encapsulated logic. In some cases, where the incapsulated layer does not provide strong reconnection capabilities, which could be the case for application layer Layer protocls that rely on stable connections, the connection will be lost entirely.

Next step is to confirm such assumption by conducting a ports mapping test with Node.js dgram module and Wireshark. At this stage, we're going to send some UDP packages from a virtualized Ubuntu Desktop distribution and listen for incoming UDP packages on a host machine. We'll also leverage Wireshark's capabilities to monitor local traffic. At this stage, we're using VMware virtual NAT implementaion.

The following is a code snippet for a server listening for UDP packages:

```
const dgram = require('dgram');
const server = dgram.createSocket('udp4');

server.on('error', (err) => {
  console.log(`server error:\n${err.stack}`);
  server.close();
});

server.on('message', (msg, rinfo) => {
  console.log(`server got: ${msg} from ${rinfo.address}:${rinfo.port}`);
});

server.on('listening', () => {
  const address = server.address();
  console.log(`server is listening ${address.address}:${address.port}`);
});

server.bind(8000);
```

ISSN 2663 - 4023

Next is a code of a client sending UDP packages:

```
const dgram = require('dgram');

const ADDRESS = '192.168.0.104';
const PORT = 8000;
const INTERVAL = 1000;
const client = dgram.createSocket('udp4');

client.send('Hello World!', 0, 12, PORT, ADDRESS, () => {
  console.log(`A message has been sent to ${ADDRESS}:${PORT}`);
});

setInterval(() => {
  client.send('Hello World!', 0, 12, PORT, ADDRESS, () => {
    console.log(`A message has been sent to ${ADDRESS}:${PORT}`);
  });
}, INTERVAL)
```

On the following figure, client is sending UDP packages with a 1-second interval:

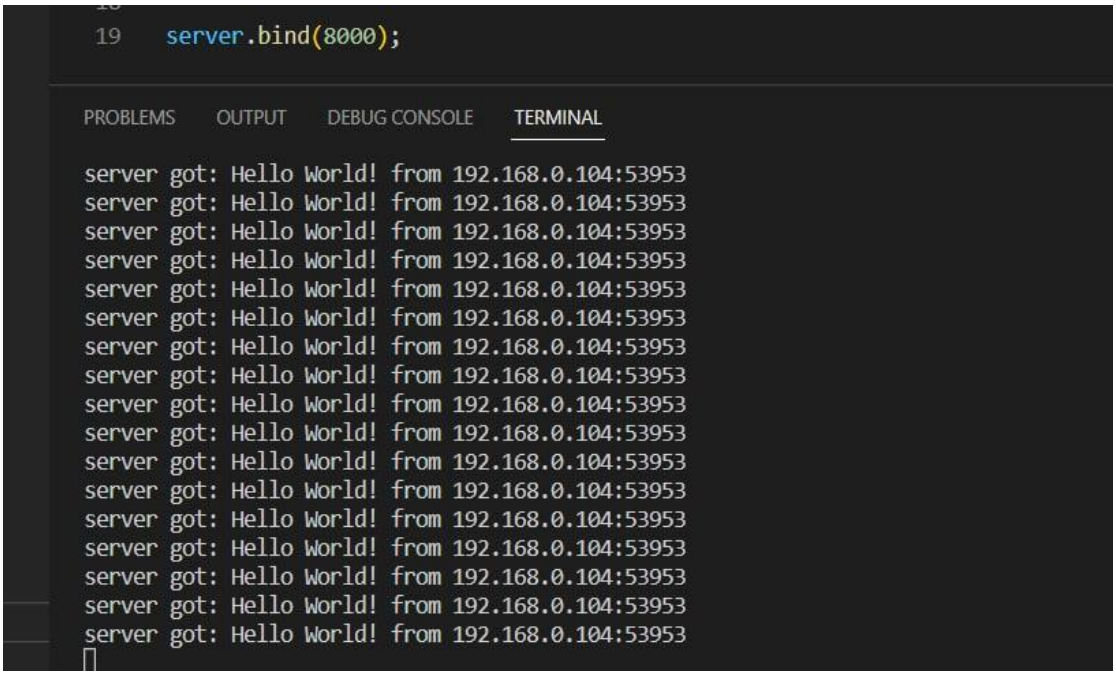

Fig. 5. Logs of receiving UDP packages through NAT with a 1-second interval

It is evident that the port mapping stays stable. NAT implementation by VMware uses one of the techniques called port mapping TTL to avoid premature connection drops. Next figure shows datagrams intercepted with the means of Wireshark:

ISSN 2663 - 4023

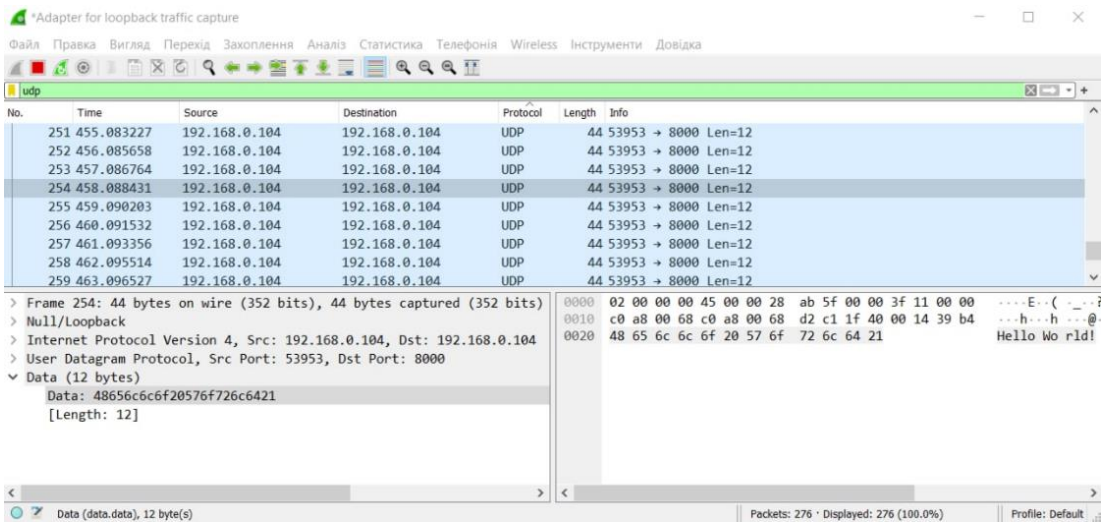

Fig. 6. Intercepting 1-second interval packages with Wireshark

In the shown example, it seems that the communication endpoints remain stable due to the mentioned technique: the source port is always the same, hence the connection is preserved.

Following is the case when a 1-minute interval is used:

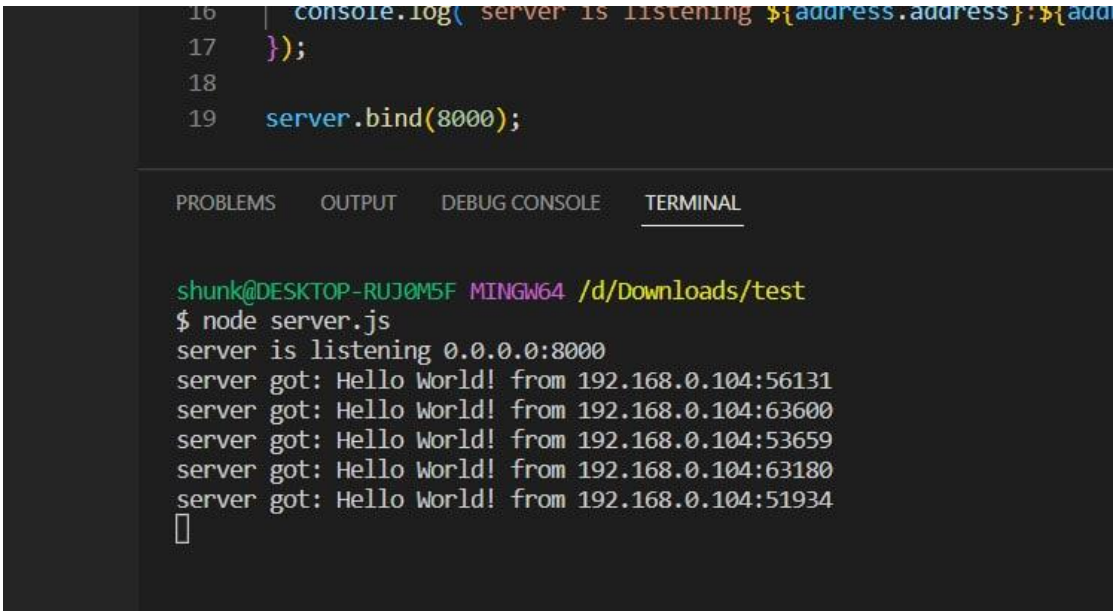

Fig. 7. Logs of receiving UDP packages through NAT with a 1-minute interval

By reading through the log messages from the server shown in the Fig. 7, it is becoming obvious that the source port changes with each subsequent call.

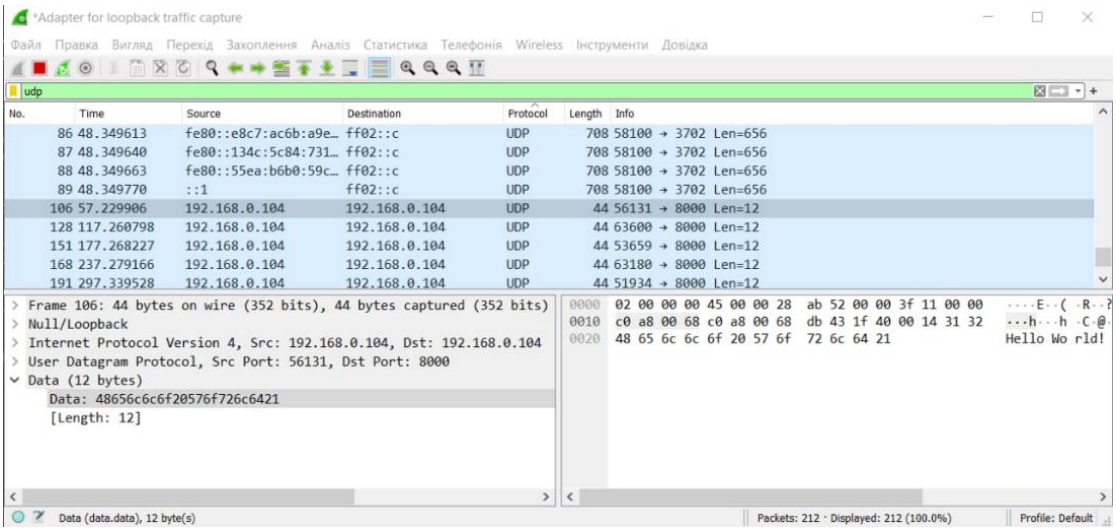

Fig. 8. Intercepting 1-minute interval packages with Wireshark

Having analyzed log messages and intercepted datagrams from Wireshark in Figs. 5–7, it is obvious that VMware's NAT implementation has port mapping Time to Live in place, and as soon as it expires, the mapping gets released. This has to be done to avoid indefinitely taken resources, but such an approach has its flows to manage: if the TTL is too low, then it's possible that the client would not be able to get replies at all in a congested network, and if the TTL is too large, then clients could abuse the limited resources of the server. Remember that each client communication through NAT takes two ports: one for incoming messages from the client and another for incoming packages from remote servers, thus effectively limiting the amount of alive mapping to 32768 in the best case. NAT can have multiple Network Interfaces but in order for them to be of any use, they need to be assigned a routable IP address.

These issues eventually come down to a negotiable balance between resource utilization on NAT and communication timing within the client-server connection it supports. In the following chapters, we are going to examine existing solutions and propose our own.

EXISTING DURABLE PORT MAPPING SOLUTIONS

In the following chapter, we will discuss existing solutions for durable port mappings. We will start with the common solution used inside provide networks called "Static Ports Mapping" and examine why such a solution is not optimal for a general case and cannot be used for most NAT systems used worldwide.

Static port mapping enables the forwarding of all incoming packages from a specified port on the NAT to the port on the Ubuntu VM behind the NAT. Such mapping is durable and persistent, the two key factors needed for specific kinds of applications like VPNs or gaming servers.

Such a solution allows duplex communication initiation. Since the port is statically mapped and available on NAT's interface that has a routable IP address assigned, the client from outside the network can initiate the communication, as opposed to the traditional usage of NAT that conceals the internal network members and architecture [19].

In VMware environments, you can configure static port mapping through the virtual network editor. The configuration of the VMware NAT static mapping is show in Fig. 9 [19]:

ISSN 2663 - 4023

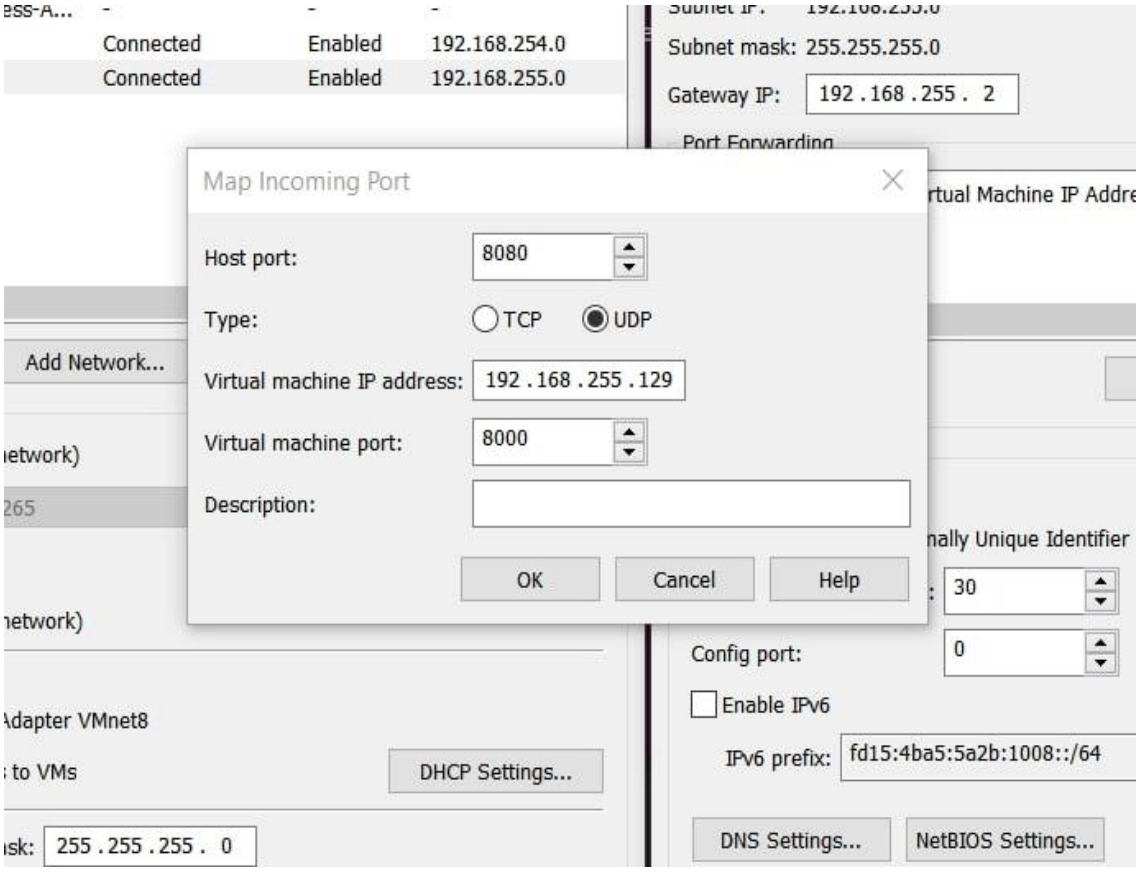

Fig. 9. VMware NAT static ports mapping

Even though it's guaranteed now that the incoming packages will be sent on the same port, this is just a partial solution since outgoing packages will still have their ports changed. Having said that, this will not suffice as a general problem solution. Furthermore, it's extremely inefficient to hold a lot of static port mappings due to the limited resources discussed in the previous sections. This approach cannot be used in an Internet Service Provider's NAT due to the sheer number of clients, elasticity, and scaling that it should provide.

It is also worth noting that even though in some cases it is beneficial to have a direct static mapping to the outside network, in such a setup, the machine from a private network and the entire private network, for that matter, become prone to security vulnerabilities since the outsiders can effectively initiate connections, data requests, and traffic flow inside the private network.

Another common technique used to mitigate port mapping drops is to increase Time to Live setting for the NAT device. TTL represents the duration for which a specific port mapping remains valid when there's no ongoing traffic. Finding a balanced value allows for both: decreasing the degree of connection drops and reclaiming resources on NAT in an efficient way [20].

VMware allows you to configure port mapping TTL through the virtual network editor. The configuration of the VMware NAT TTL is show in Fig. 10 [20]:

ISSN 2663 - 4023

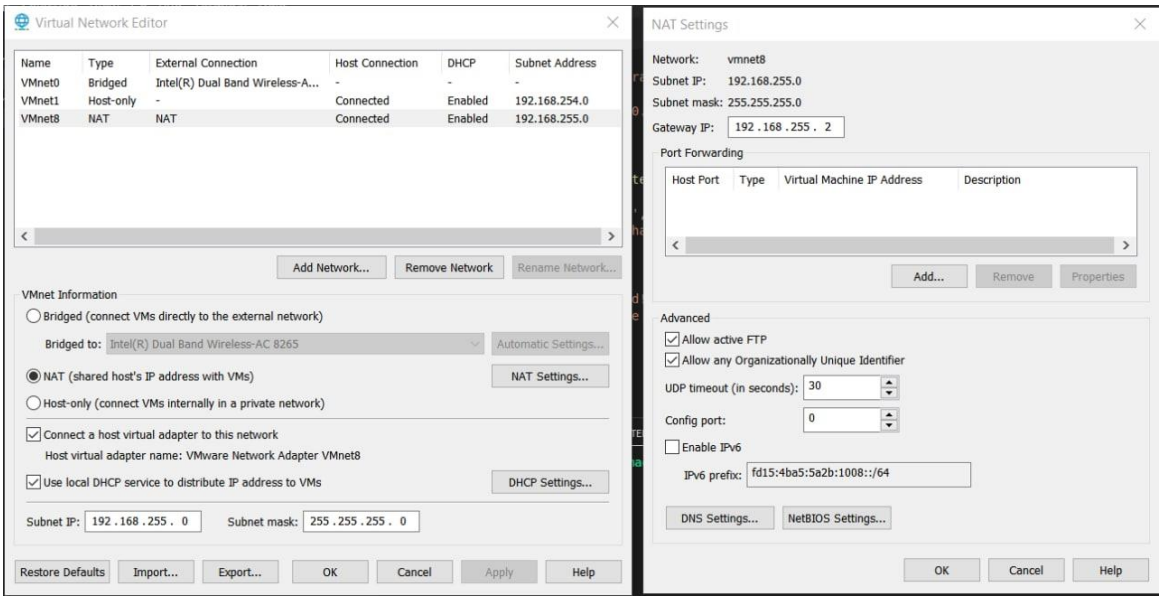

Fig. 10. VMware NAT configuration

VMware virtual NAT allows for increased TTL for port mappings, which is beneficial for maintaining connections during extended periods of inactivity [20]. Applications that require persistent connections in order to ensure safe and robust communication, such as WireGuard, benefit tremendously from this feature. Because unused mappings consume NAT table entries for extended periods of time, overly long TTLs can lead to an inefficient use of network resources. Leaving open ports of internal services that do not have security measures necessary for managing incoming requests from the outside network for extended periods of time may put the entire network at risk of being compromised. The goal here is to find a balance where the TTL is long enough to sustain essential connections but not so long that it causes a security risk.

When the configuration of the network is constantly altered, it may be beneficial to increase the TTL by a reasonable amount. It is also worth noting that a longer TTL might be more suited in cases where devices require open ports on a constant basis, such as voice streaming or VPN connection.

Setting a TTL that somewhat higher than the default will help reduce interruptions caused by reauthentication or reconnection procedures that are triggered by expired NAT port mappings. This is especially applicable to WireGuard implementations that are situated within a NAT-based environment since it incapsulates all the traffic in the UDP packages, and each NAT mapping drop will require all encapsulated logic to follow the reconnection procedures. In cases with TCP, it would require going through the process of a three-way handshake again; in cases with TLS, it would require yet again exchanging the key and verifying the certificates; and if there was any other higher-level logic that depended on such a connection, it would require calling its own connection failure procedures.

Finally, one of the most common techniques used for keeping the data streaming connections alive is sending an empty UDP package in order to artificially reset the NAT port mapping TTL [21].

The following Fig. 11 illustrates the diagram of NAT mapping refreshing process [21]:

ISSN 2663 - 4023

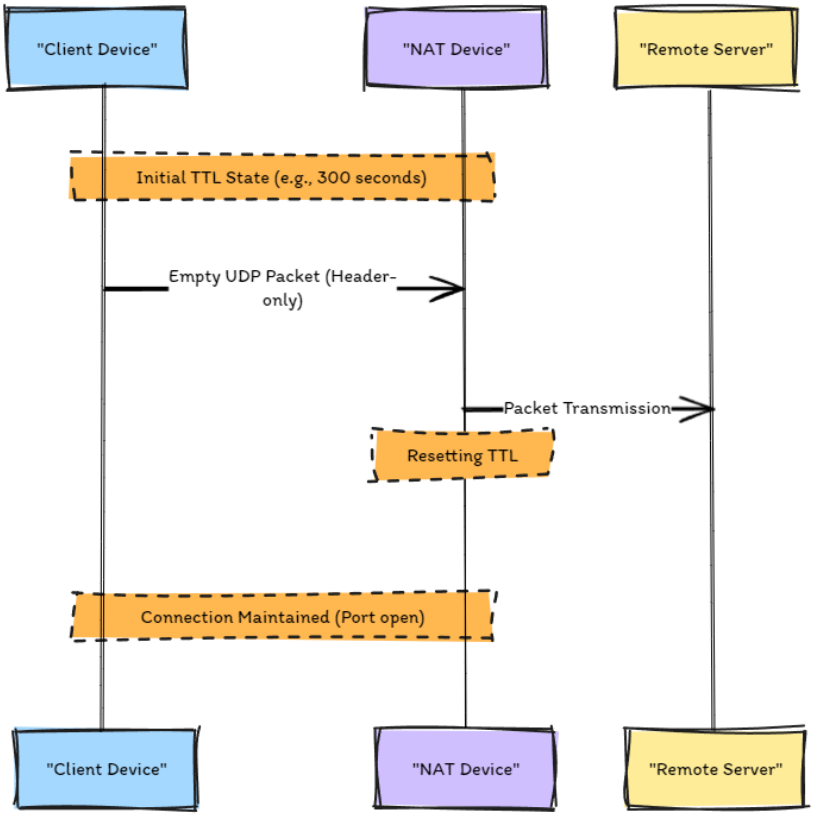

Fig. 11. NAT mapping refreshing diagram

After the TTL has elapsed, NAT devices will close ports and drop mappings from the table that were not actively used in the past. Sending an empty packet in such a case assures that the NAT device will notice activity on the port, which will then cause the TTL to get reset. This kind of datagram does not have a payload, but it still follows the protocol-defined structure perfectly and has the header information, which includes source and destination ports as well as IP addresses. The TTL associated with the port mapping is refreshed because the NAT device interprets this packet as genuine traffic when it gets it.

The following code snippet show a functioning of a simple NAT mapping refresher:

```
const dgram = require('dgram');
const client = dgram.createSocket('udp4');

function keepAlive() {
    const message = Buffer.from('');
    client.send(message, 0, message.length, PORT, HOST, (err) => {
        if (err) {
            console.error('Failed to send keep-alive packet:', err);
        } else {
            console.log('Keep-alive packet sent');
        }
    });
}

setInterval(keepAlive, 300000);
```

Having said that and seen the implementation, we can confidently say that among the advantages of such method is that the implementation of this method is simple, it does not need any complicated configuration adjustments in the network architecture, and additionally, this approach is resource-efficient due to the fact that UDP packets, particularly empty ones, are quite tiny and need just a minimum amount of processing. In situations where re-establishing connections might be expensive or technically difficult, it is very helpful to have this capability.

It is should be mentioned that long living mappings could pose a threat if malicious actors have successfully identified port mapping behind NAT and no additional security measures are in place. Also, the issue with this method is that it can generate too much network traffic if a large number of devices use it all the time and the efficacy of this method depends on the configuration of the NAT, because some of them may have mechanisms to detect and block these keep-alive strategies.

DEVELOPING A NAT MAPPING SUPPORT PROTOCOL (NMSP)

In order to permit dynamic TTL management and continue to ensure compatibility with systems that do not support this extension, the protocol combines an upgraded payload inside a conventional UDP datagram. When the UDP header is first created, it is structured in a normal manner. It includes fields such as the source and destination ports, the length of the packet (which is determined entirely by the standard UDP header and the initial payload), and a checksum that encompasses the standard datagram without enhancement. Next, the initial payload is added, which could be a simple message such as “Hello, this is a standard message”. This is followed by the enhanced section. Any device that does not recognize the improved structure will be able to handle the message normally, just like any other conventional UDP datagram, thanks to this.

A magic number, such as 0x004c652043686174, designates the beginning of the enhanced section, which comes immediately after the first payload. For complying devices, this identification serves as a signal that extra data is following, which requires particular processing. In the subsequent byte, which is known as the Flags byte, there are several indicators: the first bit ensures that a specified TTL is adhered to if it is set (mandatory), the second bit allows for TTL adjustment within a range (adjustable), and the third bit triggers an immediate response from the natural language processing (NAT) with its TTL settings if it is set (immediate response).

Within the subsequent segment, the TTL settings themselves are provided, and the length of this segment is specified (for example, 0×0014 for several 6-byte records in addition to a checksum as an example). For the client, each NAT device that the packet passes through, and the final receiver, each TTL record is appended, and the records length should be updated. This record includes the desired TTL, the maximum TTL, and the minimum TTL time.

With the use of a straightforward sum modulo 65536, a checksum is computed across all of the TTL fields in order to verify the accuracy of the data. After receiving this packet, a device that is compatible will identify the magic number and then proceed to process the TTL settings in accordance with the information.

Every transitioned target node (NATs and the recipient) should instantly send back a packet that has its own TTL values when the Immediate Response flag is set to the “on” mode. This makes it possible for dynamic feedback to be sent back and avoid situations where the recipient does not support the protocol and won’t send the eventual data records. On the other

ISSN 2663 - 4023

hand, devices that do not adhere to the standard disregard the data that has been added to the packet and proceed to handle it as if it were any other UDP datagram possible.

Having such a setup, the protocol is guaranteed to be backwards compatible, and it guarantees that legacy systems will continue to function without any interruptions. Additionally, this well-considered design makes it feasible for the protocol to be simply included into pre-existing network infrastructures, while at the same time giving expanded capabilities for network management and diagnostics across all platforms that are supported.

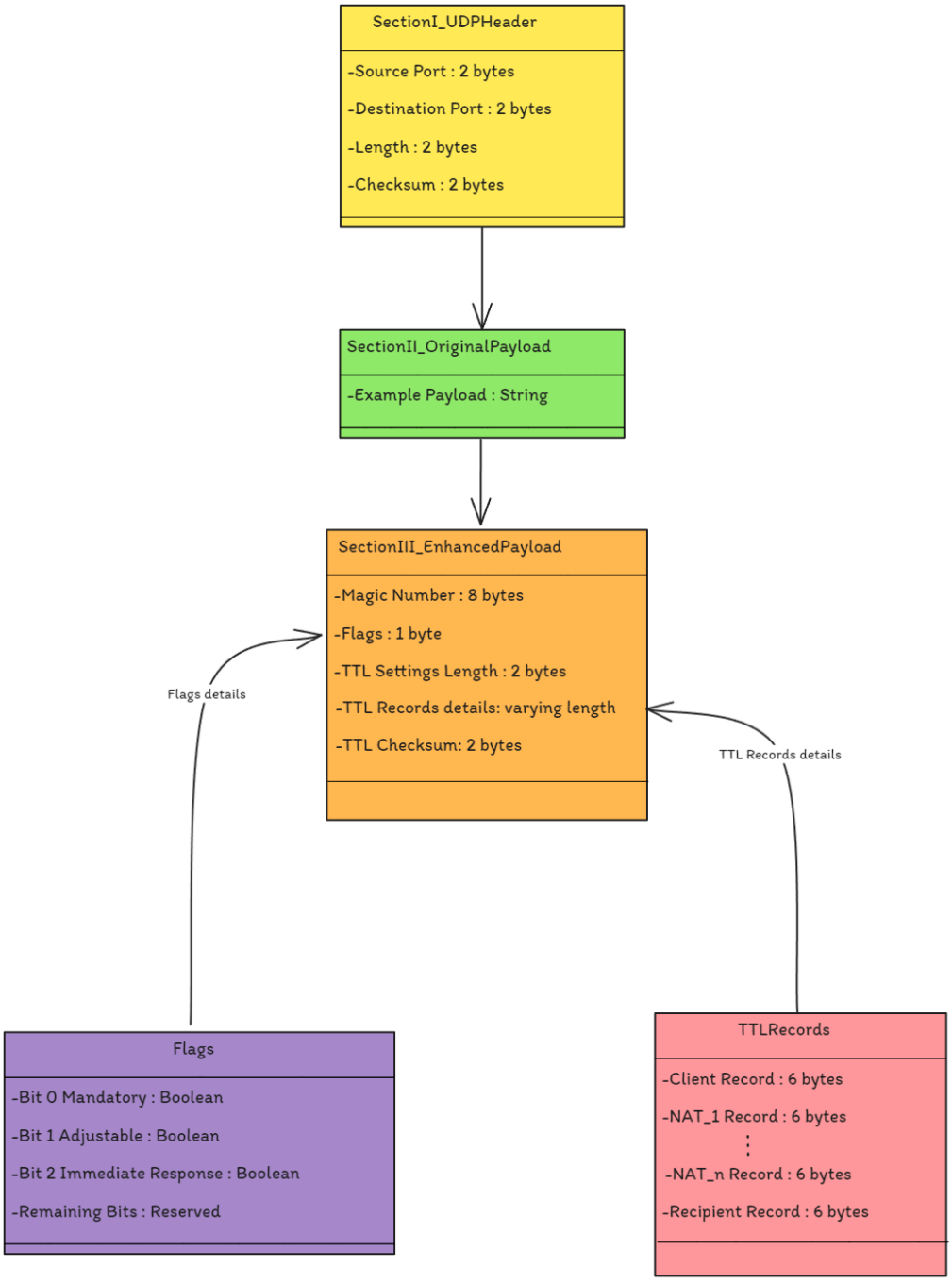

Fig. 12. Protocol's datagram structure

Section I: UDP Header

Description: standard components as defined by the UDP protocol.

Structure:

- Source Port (2 bytes)
- Destination Port (2 bytes)
- Length (2 bytes) — This field includes the length of the standard UDP header and original payload without appended enhanced data.
- Checksum (2 bytes) — Covers the entire packet.

Section II: Original Payload

Description: The application data intended for the recipient, placed immediately after the UDP header to ensure that non-compliant devices process the packet as a normal UDP datagram:

Example Payload: “Hello, this is a standard message”.

Section III: Enhanced Payload

Description: Appended after the original payload, containing the Magic Number and TTL settings.

Structure:

- Magic Number (8 bytes)
  - Description: A unique identifier that signifies the beginning of the enhanced portion of the payload.
  - Example: 0×004c652043686174
- Flags (1 byte)
  - Description: Bits set to indicate how the packet should be handled.
  - Structure:
    - Bit 0 (Mandatory): If set, the desired TTL must be adhered to.
    - Bit 1 (Adjustable): Allows NAT to adjust the TTL within a specified range.
    - Bit 2 (Immediate Response): If set, instructs the NAT to respond immediately with its TTL settings.
    - Remaining Bits: Reserved for future use.
  - Example: 0×07 (All three features are enabled).
- TTL Settings Length (2 bytes)
  - Description: Specifies the length of the TTL settings data that follows, including all records and the checksum.
  - Example: 0×0014 (for 3 records each 6 bytes and a 2-byte checksum).
- TTL Records
  - Description: Each NAT device appends a 6-byte record of its TTL settings.
  - Record Format: Desired TTL (2 bytes), Max TTL (2 bytes), Min TTL (2 bytes)
  - Examples:
    - Client Record: 0300, 0E10, 012C (Desired: 768 seconds, Max: 3600 seconds, Min: 300 seconds)
    - NAT Record: Similar format, added by each NAT through which the packet passes.
    - Recipient Record: Reflects the end recipient’s TTL preferences.
- TTL Checksum (2 bytes)
  - Description: A checksum computed over all TTL records to ensure integrity.
  - Calculation: Simple sum of TTL fields values, modulo 65536.

Packet Processing and Response Mechanism

- For Compliant Devices: Recognize the Magic Number and process the appended TTL settings. If the Immediate Response flag is set, the NAT device sends back a packet containing its TTL settings to the sender immediately.
- For Non-Compliant Devices: Treat the entire packet as a standard UDP datagram, ignoring the data after the original payload.

Example of Full Packet Structure:

```
[UDP Header] | [Original Payload: "Hello, this is a standard message."] |
[Magic Number: 0x004c652043686174] | [Flags: 0x07] | [TTL Settings
Length: 0x0008] | [Client Record: 0300, 0E10, 012C] | [TTL Checksum]
```

After having a complete round trip picture of TTLs used by all the target transitional nodes, they can adjust its intervals of sending empty UDP packages to keep the binding alive. Such enhancement allows for more efficient use of network resources and allows to find the balanced interval for reaffirming the binding in an automated way.

The protocol is constructed to be fully compatible with the existing infrastructure. It is not required for any intermediate NAT or eventual recipient to be able to handle extended payload. The target process on the receiver's end should not even pass the enhanced payload part to the target process since it will be outside of a specified UDP datagram length.

Such property is invaluable in the current enormous network environment where it absolutely should be expected that the new protocol will not be adopted any time soon by most of the networking infrastructure and peers.

CONCLUSIONS

This article critically analyzes both established and new ways, methods, and technologies to construct a reliable, efficient, and secure modern network infrastructure that supports long-lived UDP port forwarding mappings in each chapter.

In this study, the complexity of developing and maintaining UDP-based apps like video streaming platforms, live chats, and virtual private networks was assessed. We've discussed how important it is to manage UDP-based data transfers in NAT-based environments.

Through a deep investigation of the internal operations of the WireGuard VPN protocol, we have covered the fundamental principles and challenges that arise in such an architecture. With the security, efficiency, and effectiveness of the network address translation technology came the exhaustive problem of inefficient management of network traffic and abruptly dropped connections.

We conducted a thorough overview and analysis of common techniques, methods, and technologies used to mitigate such issues. Through a comprehensive study of the internal workings of VMware's Network Address Translation, we have seen how the issue arises and the most basic solutions that could potentially mitigate such a problem with the built-in functions of VMware. The practical demonstration with simple Node.js scripts, VMware's NAT, and Wireshark has shown the core principles of NATs that encompass the issues with unstable UDP data streaming.

As a result of such investigation, in this article, a new method of building reliable data streams based on UDP transport has been proposed, its applications have been described in great detail, and potential limitations have been shown, which could spark additional future discussions and research regarding this issue.

REFERENCES (TRANSLATED AND TRANSLITERATED)

1. IETF. (n.d.). *Internet Engineering Task Force.* https://www.ietf.org/rfc/rfc0768.txt
2. IBM documentation. (n.d.). *IBM in Deutschland, Österreich und der Schweiz.*
3. Advantages of UDP | disadvantages of UDP. (n.d.). *RF Wireless Vendors and Resources | RF Wireless World.* https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-UDP.html
4. Javatpoint. (n.d.). *UDP protocol | user datagram protocol - javatpoint.* https://www.javatpoint.com/udp-protocol
5. ClouDNS Blog. (n.d.). *UDP (user datagram protocol) explained in details - cloudns blog.* https://www.cloudns.net/blog/udp-user-datagram-protocol-explained-in-details/
6. Khan Academy. (n.d.). *User datagram protocol (UDP) (article) | khan academy.* https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:the-internet/xcae6f4a7ff015e7d:transporting-packets/a/user-datagram-protocol-udp
7. bunny.net. (n.d.). *What is user datagram protocol(udp)? What are its benefits?. What Is User Datagram Protocol(UDP)? What are its benefits?* https://bunny.net/academy/network/what-is-user-datagram-protocol-udp-and-how-does-it-work/
8. *Chapter 1 - an introduction to network address translation. Microsoft Learn: Build skills that open doors in your career.* (n.d.). https://learn.microsoft.com/en-us/azure/rtos/netx-duo/netx-duo-nat/chapter1
9. Hanna, K. T., & Burke, J. (2024). *What is network address translation (NAT) and how does it work?.* Networking. https://www.techtarget.com/searchnetworking/definition/Network-Address-Translation-NAT
10. CompTIA. (n.d.). *Network address translation definition | how NAT works | computer networks | comptia.* https://www.comptia.org/content/guides/what-is-network-address-translation
11. GeeksforGeeks. (n.d.). *Network address translation (NAT) - geeksforgeeks.* https://www.geeksforgeeks.org/network-address-translation-nat/
12. Fortinet. (n.d.). *What is NAT (network address translation)? How does NAT work?.* https://www.fortinet.com/lat/resources/cyberglossary/network-address-translation
13. Avi Networks. (n.d.). *What is network address translation? | avi networks.* https://avinetworks.com/glossary/network-address-translation/
14. Cisco. (n.d.). *What is network address translation (NAT)?* https://www.cisco.com/c/en/us/products/routers/network-address-translation.html#:~:text=Network%20Address%20Translation%20(NAT)%20is,sent%20to%20an%20external%20network. (date of access: 17.02.2024).
15. NDSS Symposium. (n.d.). *WireGuard: next generation kernel network tunnel - NDSS symposium.* https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/wireguard-next-generation-kernel-network-tunnel/
16. *WireGuard: fast, modern, secure VPN tunnel.* (n.d.). https://www.wireguard.com/
17. NordLayer. (n.d.). *What is WireGuard protocol? | NordLayer Learn. Network Access & Security Solutions.* https://nordlayer.com/learn/vpn/wireguard/?gad_source=1&gclid=CjwKCAjw_e2wBhAEEiwAyFFFo3O_v0y1DaH_T0pBVCSGjs6vjr_nykdMmznFsK9MiVH_5yB3CbmX4hoCeW4QAvD_BwE
18. VMware Docs Home. (n.d.). *Understanding virtual networking components.* https://docs.vmware.com/en/VMware-Workstation-Pro/17/com.vmware.ws.using.doc/GUID-8FDE7881-C31F-487F-BEF3-B2107A21D0CE.html
19. VMware Docs Home. (n.d.). *Using the virtual network editor.* https://docs.vmware.com/en/VMware-Workstation-Pro/17/com.vmware.ws.using.doc/GUID-AC956B17-30BA-45F7-9A39-DCCB96B0A713.html
20. VMware Docs Home. (n.d.). *Configuring network address translation.* https://docs.vmware.com/en/VMware-Workstation-Pro/17/com.vmware.ws.using.doc/GUID-89311E3D-CCA9-4ECC-AF5C-C52BE6A89A95.html
21. Halkes, G., Pouwelse, J. (2011). UDP NAT and Firewall Puncturing in the Wild. *NETWORKING 2011. Lecture Notes in Computer Science, Vol. 6641.* https://doi.org/10.1007/978-3-642-20798-3_1

КІБЕРБЕЗПЕКА: освіта, наука, техніка
CYBERSECURITY: EDUCATION, SCIENCE, TECHNIQUE
№ 4 (24), 2024
ISSN 2663 - 4023

Котов Максим Сергійович
магістр кібербезпеки, студент кафедри кібербезпеки та захисту інформації
Київський національний університет імені Тараса Шевченка, Київ, Україна
ORCID ID: 0000-0003-1153-3198
maksym_kotov@ukr.net

Толюпа Сергій Васильович
д.т.н., професор, професор кафедри кібербезпеки та захисту інформації
Київський національний університет імені Тараса Шевченка, Київ, Україна
ORCID ID: 0000-0002-1919-9174
tolupa@i.ua

МЕТОДИ СТВОРЕННЯ НАДІЙНИХ СПІВСТАВЛЕНЬ UDP ПОРТІВ У СЕРЕДОВИЩІ НА ОСНОВІ NAT

Анотація. Використання протоколів без встановлення з'єднань, таких як UDP (User Datagram Protocol) стає все більш важливим у сучасних цифрових мережах, які постійно розширюються та стають все більш складними. Підтримка відображень UDP у середовищах на основі NAT, надійний і безперебійний зв'язок для різних завдань, таких як оперативна передача даних і встановлення безпечних з'єднань через віртуальні приватні мережі (VPN), як-от WireGuard, є надзвичайно важливими. Трансляція мережевих адрес (NAT) є важливою частиною вирішення проблеми обмеженої кількості глобальних адрес Інтернет-протоколу (IP) і підвищення безпеки мереж, приховуючи приватні мережі. Однак трансляція мережевих адрес (NAT) створює низку проблем, однією з яких є динамічне призначення номерів портів, що потенційно може призвести до перебоїв у з'єднаннях. Метою цієї статті є докладне пояснення функціонування WireGuard, приділяючи особливу увагу важливості надійних відображень UDP для досягнення максимальної продуктивності. Крім того, у статті розглядається рішення VMware для трансляції мережевих адрес (NAT), щоб проілюструвати проблеми, пов'язані з підтримкою зіставлення UDP. У цій статті проводиться дослідження багатьох методів і поточних рішень, які були розроблені для вирішення цих проблем. Деякі з реалізованих стратегій включають використання статичного відображення портів для встановлення надійного маршруту через NAT, розширення часу життя (TTL) для відображення портів для зменшення кількості розривів з'єднання та підхід надсилання порожніх UDP-пакетів, щоб зберегти активні відображення. На додаток до цього, запропоновано нову концепцію: протокол для простого зіставлення NAT, який намагається спростити процес зміни частоти UDP-зондів, вимагаючи від пристроїв NAT розкривати свої налаштування TTL. Мета цього протоколу — зробити відображення NAT легшим і ефективнішим з точки зору загального мережевого трафіку.

Ключові слова: Network Address Translation (NAT); User Datagram Protocol (UDP); віртуальні приватні мережі (VPNs); WireGuard; VMware; стійке співставлення UDP; час життя співставлення портів (TTL); статичне співставлення портів; протокол підтримки співставлень NAT; надійність мережі; оптимізація мережевої взаємодії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. IETF. (n.d.). *Internet Engineering Task Force.* https://www.ietf.org/rfc/rfc0768.txt
2. IBM documentation. (n.d.). *IBM in Deutschland, Österreich und der Schweiz.*
3. Advantages of UDP | disadvantages of UDP. (n.d.). *RF Wireless Vendors and Resources | RF Wireless World.* https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-UDP.html
4. Javatpoint. (n.d.). *UDP protocol | user datagram protocol - javatpoint.* https://www.javatpoint.com/udp-protocol
5. ClouDNS Blog. (n.d.). *UDP (user datagram protocol) explained in details - cloudns blog.* https://www.cloudns.net/blog/udp-user-datagram-protocol-explained-in-details/

6. Khan Academy. (n.d.). *User datagram protocol (UDP) (article) | khan academy.* https://www.khanacademy.org/computing/computers-and-internet/xcae6f4a7ff015e7d:the-internet/xcae6f4a7ff015e7d:transporting-packets/a/user-datagram-protocol-udp
7. bunny.net. (n.d.). *What is user datagram protocol(udp)? What are its benefits?. What Is User Datagram Protocol(UDP)? What are its benefits?* https://bunny.net/academy/network/what-is-user-datagram-protocol-udp-and-how-does-it-work/
8. *Chapter 1 - an introduction to network address translation. Microsoft Learn: Build skills that open doors in your career.* (n.d.). https://learn.microsoft.com/en-us/azure/rtos/netx-duo/netx-duo-nat/chapter1
9. Hanna, K. T., & Burke, J. (2024). *What is network address translation (NAT) and how does it work?.* Networking. https://www.techtarget.com/searchnetworking/definition/Network-Address-Translation-NAT
10. CompTIA. (n.d.). *Network address translation definition | how NAT works | computer networks | comptia.* https://www.comptia.org/content/guides/what-is-network-address-translation
11. GeeksforGeeks. (n.d.). *Network address translation (NAT) - geeksforgeeks.* https://www.geeksforgeeks.org/network-address-translation-nat/
12. Fortinet. (n.d.). *What is NAT (network address translation)? How does NAT work?.* https://www.fortinet.com/lat/resources/cyberglossary/network-address-translation
13. Avi Networks. (n.d.). *What is network address translation? | avi networks.* https://avinetworks.com/glossary/network-address-translation/
14. Cisco. (n.d.). *What is network address translation (NAT)?* https://www.cisco.com/c/en/us/products/routers/network-address-translation.html#:~:text=Network%20Address%20Translation%20(NAT)%20is,sent%20to%20an%20external%20network. (date of access: 17.02.2024).
15. NDSS Symposium. (n.d.). *WireGuard: next generation kernel network tunnel - NDSS symposium.* https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/wireguard-next-generation-kernel-network-tunnel/
16. *WireGuard: fast, modern, secure VPN tunnel.* (n.d.). https://www.wireguard.com/
17. NordLayer. (n.d.). *What is WireGuard protocol? | NordLayer Learn. Network Access & Security Solutions.* https://nordlayer.com/learn/vpn/wireguard/?gad_source=1&gclid=CjwKCAjw_e2wBhAEEiwAyFFFo3O_v0y1DaH_T0pBVCSGjs6vjr_nykdMmznFsK9MiVH_5yB3CbmX4hoCeW4QAvD_BwE
18. VMware Docs Home. (n.d.). *Understanding virtual networking components.* https://docs.vmware.com/en/VMware-Workstation-Pro/17/com.vmware.ws.using.doc/GUID-8FDE7881-C31F-487F-BEF3-B2107A21D0CE.html
19. VMware Docs Home. (n.d.). *Using the virtual network editor.* https://docs.vmware.com/en/VMware-Workstation-Pro/17/com.vmware.ws.using.doc/GUID-AC956B17-30BA-45F7-9A39-DCCB96B0A713.html
20. VMware Docs Home. (n.d.). *Configuring network address translation.* https://docs.vmware.com/en/VMware-Workstation-Pro/17/com.vmware.ws.using.doc/GUID-89311E3D-CCA9-4ECC-AF5C-C52BE6A89A95.html
21. Halkes, G., Pouwelse, J. (2011). UDP NAT and Firewall Puncturing in the Wild. *NETWORKING 2011. Lecture Notes in Computer Science, Vol. 6641.* https://doi.org/10.1007/978-3-642-20798-3_1