POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Authorizing Access to Edge Resources using 5G Device Authentication

Laurea Magistrale in Computer science and Engineering - Ingegneria Informatica

Author: Luca Giacometti

Advisor: Prof. Giacomo Verticale

Academic year: 2022-2023

1. Abstract

The authentication and authorization are fundamental aspects for the identification of a user who desires to get access to a network or a service because answering to those would be addressing to the questions "who someone is" and "what someone can do".
In recent times, the 5G networks are spreading everywhere presenting different protocols and new entities compered to the previous network generations, evoking new challenges. In addition, the Edge Computing paradigm is presenting unmatched security challenges, allowing authentication processes to trust on users which are located near or inside the network.
This thesis aims to describe a Customer Edge Switch (CES) designed to be inserted at the frontier of a 5G network by leveraging the Mobile Edge Computing paradigm. This device can authenticate and authorize users to access services with a network-layer policy-based protocol. This CES component is built to authenticate users with a P4 program and an orchestrator written in Python as data and control plane with a policy server. The environment was set up with virtual machines linked to a BMv2 switch. Some performance analysis have been done to test the efficiency of this protocol.

2. Introduction

With the advent of the Mobile Edge Computing (MEC) paradigm, some new methods to deploy resources near the data and end-users have been raised up. One possible actuator for the MEC paradigm is the 5G network.
The outside resources can not rely on the security mechanisms and protocols of the 5G networks because that would mean to increase the number of malicious actors for the edge node.
Therefore, this project is intended to introduce an Access Control node, following the previous 5G MEC scenario, that implements a protocol divided in two phases: a policy-based control to retrieve if a user is authenticated for the requested service and a network-layer protocol to authorize a user for that same service.
The architecture of this device was designed by following the SDN paradigm, exploiting the possibility to split data plane from control plane and implementing the previous authorization protocol. A performance analysis has been done to test our protocol.
Our scenario present a mobile phone user which requires to access a service located in an external network via a 5G MEC environment.
The 3GPP[1] consortium released the 5G NR world standard ([1]) describing how the packet

[1] The 3rd Generation Partnership Project

traffic has to flow inside a 5G network. Specifically for our case, a packet that transit inside a 5G network has to be sent to the UPF node at the border of network and then forwarded to the external environment.
Here, the packet is checked by our CES component in order to control if that packet traffic is authenticated and authorized or not. In case of a positive result, the packets are forwarded towards the service server.

3. Architecture

The architecture of this project was designed using these main entities, illustrated in Figure 1:

- 5G network: the relevant nodes are the AUthentication Server Function (AUSF) node which negotiates the master key with the control plane and stores it in the UDM node, the User Data Management (UDM) node which records the users profile authentication data and the User Plane Function (UPF) node which is the last node of the network where the packet traffics flow through.
- CES component: it is composed by three elements. A data plane composed by a switch which screens the internal network from malicious packets, a control plane which querying the policy server and implements the security model described in Section 4 and a policy server which caches the enabled services and who is authenticated for them.
- Service server: the entity which employs our CES component is in charge to build this server. The services have to be instantiated following the Function as a Service (FaaS) paradigm.

4. Security model

Our project is based on the Zero Trust Security (ZTS) principle ([2]). Therefore, every entity is not automatically trusted, even if it is inside the internal network perimeter. In addition, we are supposing that the external server takes advantage of the FaaS paradigm to instantiate the services ([3]).
Our assumption on the 5G environment is that all the nodes considered in our scenario can be trusted and that they act in a right way. The designed CES component cannot be compromised; i.e., it cannot be manipulated redirecting the packets in a way to satisfy the attacker's will. The policy server can dialogue only with the AUSF node in the 5G network and the service server which is also assumed as a trusted entity. Moreover, the cryptography between client and server is considered ideal.
For what concern the capabilities of the attackers, we are assuming that they have an unlimited power computation.
The insider could be a user which had compromised one or more services, resulting authenticated and authorized for a service for which they are not, with the intention to spread the assault to other machines present in the system.
The external attacker could be an outsider entity which wants to retrieve a packet to break a user's privacy or to redirect that packet in a malicious way becoming an intermediate actor between client and server or even to capture some informations regarding the methodology to forge an authorization tag.
Regarding the typologies of an attack, we had taken into account multiple variations, exploiting this paper [5].
The first type of attack considered is the policy violation attack, which is applied to bypass some policies declared by the authority of that network. To defend our CES component and the users, we had implemented a packet traffic filtering procedure during the key exchange phase of our protocol. To verify if a user can gain access to the requested service, the control plane retrieves from the policy server the list of policies and checks if that previous user is authenticated for the requested service.
Another type of offensive is the subversion attack, which has the goal to compromise a machine inside the network and then to increase the attacker's control within the network. In our case, our countermeasures are the supposed usage of the FaaS paradigm inside the external server to instantiate the services, avoiding the spreading of an attacker, and the presence of the authorized tag inside the first packet of the TCP three-way handshake to admit the sender to communicate with the service server.
The last type of attack concerned is the spying and tampering attack, that consists in the possibility of an attacker to retrieve a packet from the traffic to read it or to block and modify it

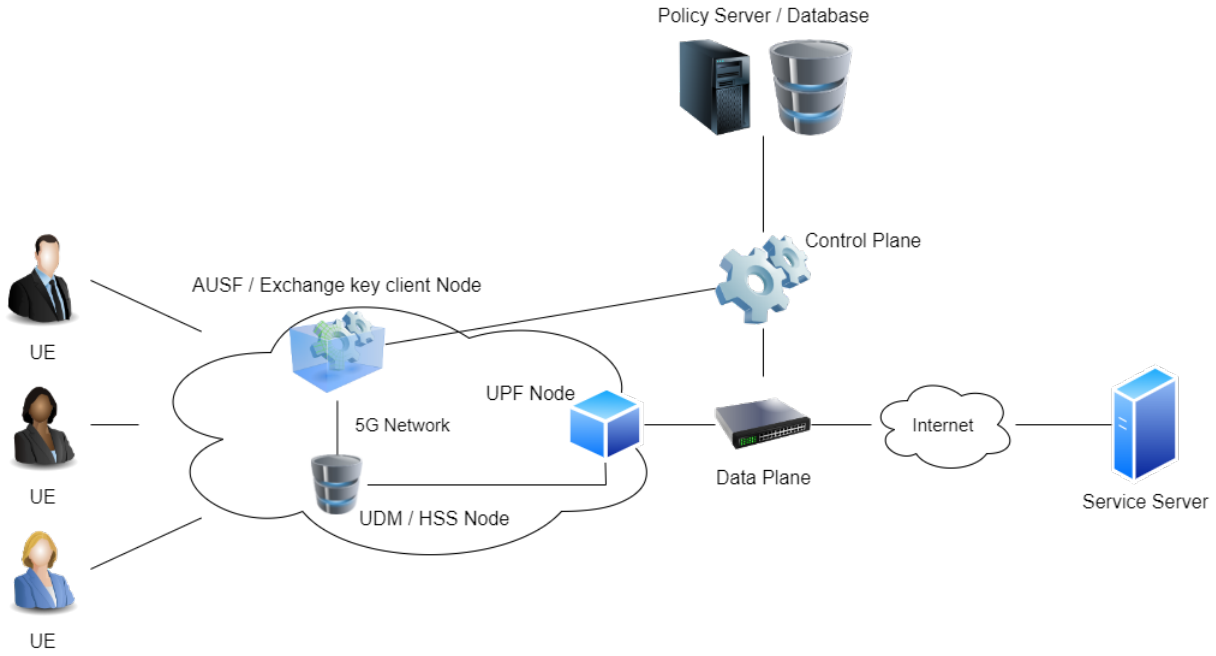

Figure 1: The reference architecture of this project

with the purpose of redirecting the manipulated packet to the original addressee entity.
One of these kind of attack is the reply attack. In our case, the attacker needs to retrieve the authorization tag from a packet but would be useless because the control plane removes from the HMAC table in the data plane that tag after a matching.
Our system is not completely protected against an hijacking attack. Nonetheless, a possible trade-off is represented by the request of an authorization tag after a small time window. With this approach, the attacker cannot infer a hypothetical tag to supply, so, the system could be under attack only during a chosen time window. On the other hand, we did not take into account the DoS attack, which aims to crash the server making every user unable to reach it, due to the absence of incentives. Moreover, to employ a Dos attack, the computational power required cannot be provided by anyone and to target our CES component the attacker has to be located in a geographical position near it.

5. Protocol description

This authorization protocol was built considering two phases and an out-of-band discovery phase: the discovery phase consists in how the credential of the service server are sent to the policy server and to the user's device. The second phase is the key exchange phase where the trade for the key is set between the AUSF node and the control plane. The last phase is the main one and it is regarded the authorization of a packet traffic by the data plane and it is represented in Figure 2.
The discovery phase is assumed already done and involves the user's device, our policy server and the service registry. In someway, the service registry has to send the name of the service and the entrypoint of the service server, for which that user is authenticated, to the policy server and the user's device.
The second phase of our protocol is the key exchange phase. We had chosen the Diffie-Hellman key agreement algorithm due to it is one of the most used and secure.
After having received the first TCP SYN open connection from a user, the AUSF node of 5G network interchanges the parameters to calculate the master key of that user with the control plane. Then, the control plane checks inside the policy server if that user is authenticated for the demanded service. If the user is authenticated, the control plane computes the master key and sends back the parameters to the AUSF node. In this way, this node can calculate the user's master key and store it inside the UDM node.
In the authorization phase we have assumed that

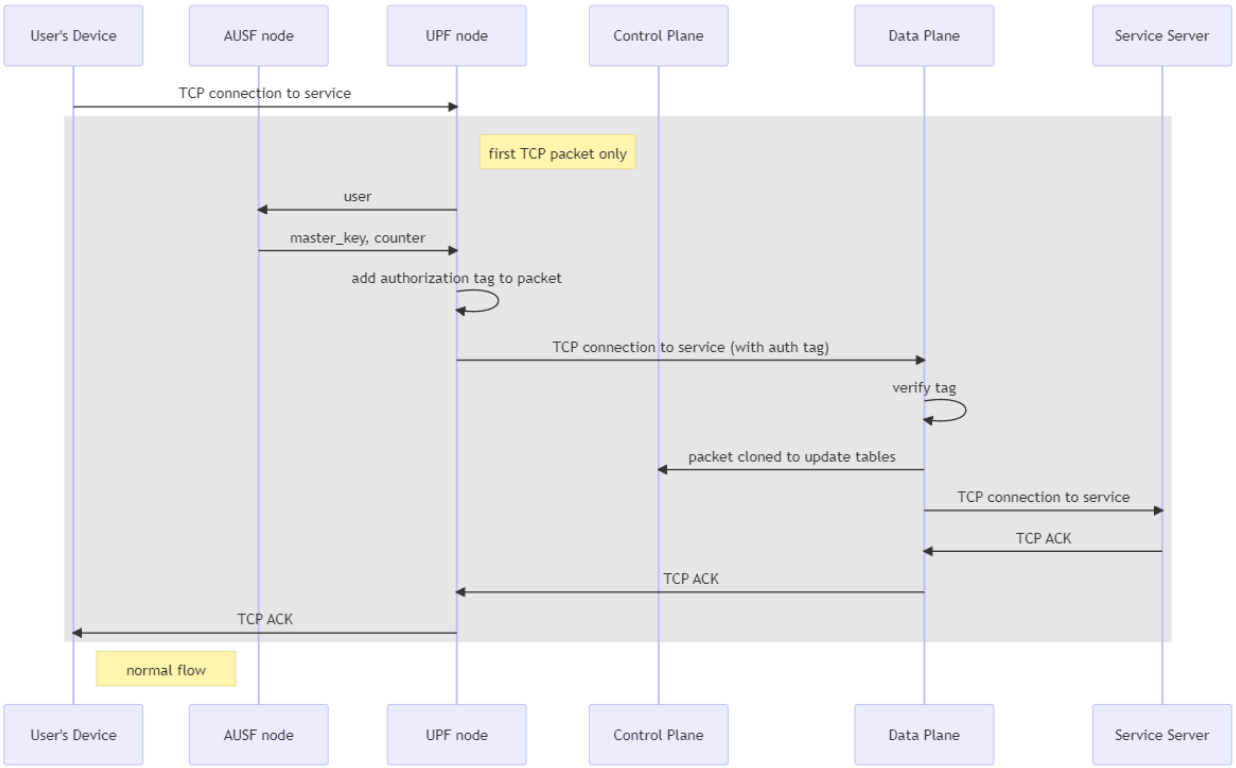

Figure 2: The authorization phase of our protocol in case of a packet with a correct authorization tag

the MAC addresses of the UPF node and service server are known by the control plane. After having captured a first TCP SYN open connection packet from a user, the UPF node queries the AUSF node in order to retrieve the master key and the counter of that user. Subsequently, this intermediate node needs to inject inside this packet the authorization tag. This tag is obtained computing the Shake-128 hash function exploiting the counter and the master key to create a HMAC; then the packet is sent and it is checked to verify if that packet traffic can be authorized or not. If so, the packet is forwarded to the service, the SYN ACK emitted by the service server is submitted to the UPF node and then the packet traffic flow is considered authorized.
The policy server was a component preexisted in the project, we have only adapted it. It is a ".yaml" file for making easier the access to it. It is read by the controller during the key exchange phase to get if a user is authenticated or not for the requested service.

5.1. Data plane

The data plane was designed with only one switch but the architecture can be extended taping to SDN paradigm with the introduction of a load balancing methodology to decrease the load of packets on each switch.
It has been thought as two phases packet checker. The first phase is the already-authorized phase where the data plane checks if the packer received is regarded to a packet traffic previously authorized in some earlier controls. There is a dedicated register where all the packet traffics previously passed towards the data plane are catalogued with an hash indexing which considers IPs, ports and protocol used (TCP or UDP).
The principal phase of our data plane is the handling packet phase. Here, it checks the previous result in order to admit directly to the forward table the packet and after a match, it emits that packet to the service. If the previous result is negative, the data plane controls if the authorization tag is present, that is made thanks to the NSH header ([4]) which can store this metadata remaining under the IP header, and in case of a positive comparison with an entry that packet traffic is signed as authorized. Then, the packet is cloned for the control plane and it is forwarded to the service server. On the contrary, if the packet has an IP header and no NSH header, it is sent to the control plane

because it could be an ARP/ICMP packet.

5.2. Control plane

Our control plane is structured as a single controller which orchestrates the data plane. It is constituted by three parts: DH key exchanger, detector policy modification, packet manager.
The first thread has the role to exchange the key with the AUSF node inside the 5G network. It contains a socket TCP server to get connections in loop. After intercepting a request, this thread reads inside the policy file to retrieve if that user is authenticated for the requested service. If there is a match, it starts to compute the master key for that user, exploiting the SHA256 hash algorithm and the user's attributes. Then, it sends back to the UPF node the parameters to calculate the master key.
There is also a thread that detects a policy modification, which employs a notify adapter to signal if some write-policy event is raised. Then, this controller thread begins to check all the new policies in order to update its stored policies.
The core procedure of our control plane is the packet manager thread which has the job to capture the packets incoming from the selection of the data plane and apply on them an additional filter to analyse only the packets directed to the service server or an ARP/ICMP packets.
If the packet is an ARP packet, the MAC address is read from the data link layer and associated with the IP from which the packet came from.
In case the packet has a transport layer, this thread needs to update the temporary rules inserted during the key exchange phase due to the natting operations of the 5G network.
The Algorithm 1 describes the behaviour of the last thread.

Algorithm 1 Packet management procedure

```
while True do
    retrieve a packet from data plane
    classify the layers of packet
    if packet protocol is TCP/UDP and it contains the access point of service server then
        update the corresponding rules inside the data plane tables
    else if packet protocol is ICMP then
        discard the packet
    else if packet protocol is ARP then
        retrieve MAC address
        associate it with the IP of packet
    end if
end while
```

6. Performance analysis

An interesting property of this component is the table size for having an optimal edge slack to update the HMAC table present in the data plane for a reasonable number of users.
To derive it, we had utilized the *queue M/M/1* from the queuing theory. Then, we had estimated an inequality which balances the total hashes already consumed plus the incoming first requests to the actual table capacity (represented by the Inequality 1) putting as constraint that the control plane cannot be set in a overload condition (outlined by the Inequality 2).

$$wN + \frac{Nw\lambda_i T_{service}}{N - w\lambda_i T_{service}} \leq c \tag{1}$$

$$\frac{w\lambda_i}{N} < \frac{1}{T_{service}} \implies N > w\lambda_i T_{service} \tag{2}$$

The results has conducted us to retrieve that a reasonable number of users is between 2500 and 2600 with an hash-consumed trigger number that can be defined between 340 and 360.
To test our authorization protocol we had compared a switch which implements our protocol to another switch with preexisting rules without any security checks.
The machine used for the tests had incorporated as CPU an Intel Xeon Silver 4114 with a frequency of 2.2 GHz, 10 cores per socket, 2 threads per core and a RAM of 64 GB and it was shared between multiple users.
In this main machine we have three virtual machines instantiated by Vagrant using a Vagrantfile where inside there were these entities:

- Client part: a python script emulating the UPF node in a 5G network in the first virtual machine.
- Data plane: a BMv2 switch running our protocol in a second VM.
- Control plane: an orchestrator script written in Python which run on the same VM of the switch.
- Policy server: a .yaml file queried by the controller to retrieve the policies.

- Server part: a socket server launched on a third VM which gets the packet traffic authorized.

We had extracted the time samples from the interface of the virtual machine where our Client part was hosted using the timestamps produced from the informations dumped by the tcpdump tool.
We have chosen to use the Scapy library of Python to simulate a TCP connection inside the user space because it would be simple to forge new packets and it was easy-accessible. In fact, our protocol requires that a NSH header has to be injected before the IP header in the network layer and it cannot be done with a classical TCP socket.
As reported in Figure 3, the first RTT of the TCP three-way handshake of a connection has spent an average time of 6.5 ms with the usage of our protocol, introducing an average delay time of 7% more than the case without our protocol. This is the case since all the authorization checks are done inside the data plane and the temporary rules were inserted in it immediately before the end of the key exchange phase.

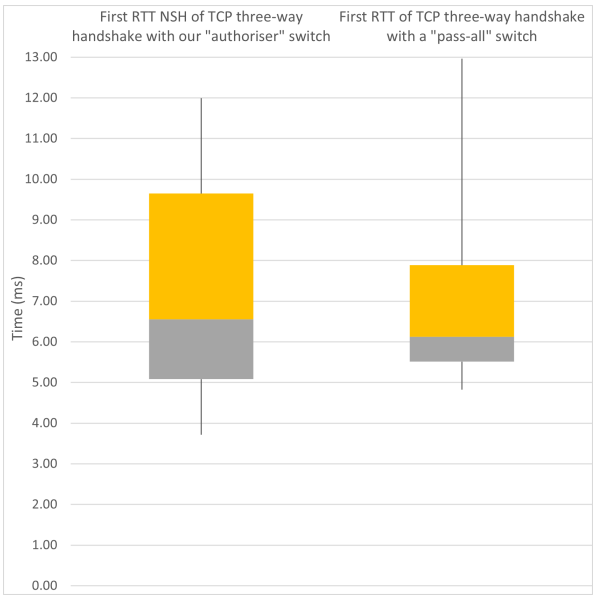

Figure 3: The comparison between a switch implementing our authorization protocol and a switch without it.

7. Conclusions

The Authentication and Authorization protocols can be developed with a plethora of different methods at various level, in particular in the network layer improving the security of an environment. This is enabled also thanks to the SDN paradigm and the separation between data and control plane.
Our project aims at designing a SDN component which can authenticate and authorize a user following a network-layer policy-based protocol to provide the access to services via a reproduced 5G network within a MEC scenario.
Considering the FaaS cloud services and the 5G networks, our component has to be enough reactive to adjust the policies and to update the table entities in order to guarantee the security of users without slowing down the system.
After the comparison of the performance between the packet traffic with a switch implementing our protocol and a "pass-all" switch, it can be inferred that our security mechanism did not add a marked delay, making this protocol enforceable.

References

[1] Erik Dahlman, Stefan Parkvall, and Johan Skold. *5G NR The Next Generation Wireless Access Technology*. Academic Press, 2020.

[2] Jason Garbis and Jerry W. Chapman. *What Is Zero Trust?*, pages 7–18. Apress, Berkeley, CA, 2021.

[3] Theo Lynn, Pierangelo Rosati, Arnaud Lejeune, and Vincent Emeakaroha. A preliminary review of enterprise serverless cloud computing (function-as-a-service) platforms. In *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, pages 162–169, 2017.

[4] Paul Quinn, Uri Elzur, and Carlos Pignataro. Network Service Header (NSH). RFC 8300, January 2018.

[5] Pamela Zave and Jennifer Rexford. Patterns and interactions in network security. *ACM Computing Surveys (CSUR)*, 53(6):1–37, 2020.