Malicious context and workaround analysis of decentralized VPN: the case of Mysterium Network

Jacopo Federici

March 3, 2020

Contents

Abstract

0.1 Work structure

0.2 Outline

0.3 Goal

1 Introduction

The need for privacy is becoming more critical day after day. The massive digitalization of many aspects of our life implicitly exposes us and our data and even the simplest operations which use an internet connection are becoming unsafe.
In the last years, we have seen security solutions integrated into popular applications, such as browsers or mobile apps, to solve the increase of security attacks. In 2017, for example, the popular Google Chrome browser announced it would mark as non-secure any website using the HTTP protocol instead of HTTPS. The HTTPS protocol is the standard HTTP protocol that uses SSL/TLS certificates, an additional security layer to make secure the communication between clients and servers. Before that date, and in some cases, the connections were without any encrypting algorithm. But, the original proposal of HTTPS protocol was made around 1990 by the Netscape Navigator browser.
The Chrome browser initiative is part of a global trend to increase the security level, often achieved encrypting the communication: realtime messages applications, internet protocols like the DNS, industrial systems, and many others.
Even if internet security has done steps forwards, there still are many situations where the individual is unsafe on the internet.
A solution can be the VPN: abbreviation for "virtual private network" it's a technical solution to extend a private network across a public network. It creates a virtual point-to-point connection through the use of a tunneling protocol over the existing network, and the tunneled traffic is encrypted. A first VPN specification proposal is around 2000 [2], but only in the last years, the use of VPN became massive. The VPN data encryption feature is solving privacy problems that are becoming relevant. Those problems of different nature are exiting a censored network, using services available or blocked in certain countries, avoiding tracking methods and other similar scenarios.

Currently, there are commercial solutions that allow users to have a private connection to the company's servers across the world, but there are many security concerns about those solutions. The company can keep the connection logs, inspect the traffic, or sell statistical data.
Some open-source projects were, lately, born to solve these problems. Their goal is to revolution the idea of VPN and its uses, building a rich network of peers where two nodes can connect through a VPN. The project provides a platform that includes all the components needed to have an efficient, fast, and easy-to-use system: from the discovery to the payment.
In this research document, we analyzed the most developed project, Mysterium Network, as a case of study. As we write, Mysterium offers a working network composed of hundreds of nodes around the world, making it the best project to investigate malicious contexts and design vulnerabilities on, owns by the decentralized architecture, more than the Mysterium project itself.
We even discover how secondary aspects are pivotal to guide attacks to the system.
The second step focuses on the solutions that can patch those critical aspects and provides a comparison of them with an eye on the practical perspective.

1.1 The world now

The world now is different from some decades ago.
First, everything is now much faster. The internet infrastructure allows us to move significant amounts of data within a few seconds. Furthermore, our personal computer can perform operations that were impossible to execute some years ago on mainstream servers. This speed is jeopardizing many security solutions, such as ciphers that based, and still base, their strength on the practical difficulties to solve the mathematical problems at the foundation of cipher methods.
Second, the architecture is moving back to a cloud-based infrastructure. Many workflows see, again, a thin client sending the execution step to a cloud server that elaborates the output that is then sent back to the client. This architecture changes the surface of the attack targeting the cloud servers as the principal attack destinations.
Third, some technologies are mature enough to provide an excellent alternative to traditional models — for instance, the cryptocurrency model

against the bank model. Here, the changes regard the paradigma, and the reasons behind these new approaches are much more than technological.
The security noun is changing the meaning inside the people's minds, from "how thick is my bank caveau" to "how long my bank requires my new online account password"

1.2 the need for trust and privacy

https://www.geeksforgeeks.org/need-of-information-security/
https://www.hackerfactor.com/blog/index.php?/archives/868-Deanonymizing-Tor-Circui
https://www.howtogeek.com/133680/htg-explains-what-is-a-vpn/
https://www.forbes.com/sites/tjmccue/2019/06/20/why-use-a-vpn/#3f08d7105859
https://www.investopedia.com/terms/b/blockchain.asp
https://www.howtogeek.com/350322/what-is-ethereum-and-what-are-smart-contracts/

https://en.wikipedia.org/wiki/Proof_of_work
https://en.wikipedia.org/wiki/Proof_of_stake

Technology is becoming an essential tool to exercise the people's rights. There are sides of the world where these rights are not accessible, and people fight to obtain them. For example, in the People's Republic of China, the government built up a firewall to censure the internet. It is called the Great Firewall Of China, abbreviated in GFW, and it is a combination of laws and technologies to control the internal domestic internet, so-called censorship. It blocks social networks, search engines, and many other services that declined the government's request to make clear and detectable the traffic.
In this context, aspects such as the anonymization, become not only relevant but crucial to be preserved, and they are required for communication systems.
A solution to obtain identity obfuscation is the VPN. It is massively used to protect data from surveillance at any level, from a public unsecured wifi connection to the country Internet Service Providers.
VPNs are provided by companies that often do not act genuinely, making the identity or the traffic detectable or even worsts sniffing and crafting the data for malicious actions.
The decentralized VPN completely changes the paradigma, the design of the infrastructure.

An analogy can be made with the traditional bank model: as we know, the bank is a centralized entity trusted by definition. What if the bank suddenly becomes untrusted? Cryptocurrencies spread among ourselves, the cryptocurrency owners, and network participants the trusts we put in the bank, using a blockchain as a technical solution. A similar design is used with decentralized VPN: there is no central entity to control the system, but a significant number of peers, the network, we can choose to connect.

1.3 Decentralized systems

1.3.1 dVPNs

A decentralized VPN is a network of nodes where the users can connect through a VPN to nodes without a central entity. It allows the user to create a private tunnel with another user, making encrypted all the traffic between the two points. The decentralization system, in this context, includes several steps to make the final connection happening, rather than a simple VPN connection between two peers.
First of all, the user creates an identity into the system, linked to the payment wallet, using a client application. Then, he connects to the network providing his availability to be a consumer or a provider.
The consumer user chooses the provider he prefers based on different features such as the region, quality, or cost of the node, and asks for a new connection to that server. In this step, the blockchain, based on Ethereum, allows creating a smart contract between the two participants with a predetermined amount of money. Using commercial VPN applications, such as OpenVPN or Wireguard, the connection between consumer and provider, otherwise called client and server, is established. When the connection correctly ends, the network closes the smart contract, that registers the result of the transaction in the blockchain.
We outlined the main steps from which we can think of a decentralized VPN more as a platform solution rather than a simple VPN application. Indeed, other projects suggested the VPN is one of the features a decentralized system so composed can have. (EXPLAIN BETTER THE IDEA)

1.3.2 The business idea

As a business project, which aims to earn money, the project has a specific business plan: provide a system network where users can join as a service provider, selling their internet bandwidth, or as a service consumer, buying other's internet bandwidth. For every cryptocurrency transaction, the network applies a fee that wants, substantially, to create a market to manage.
Each couple of entities, a consumer and a provider, agree on a transaction paid using cryptocurrency in the face of the used service. This transaction is regulated by an Ethereum smart contract signed by both sides and requires a fee to be closed. This fee is not the Ethereum fee, generally called gas, but the precisely Mysterium source of earnings. For this reason, Mysterium and other projects created their coins used to buy and sell internet bandwidth between the nodes inside the network. The coin name is MYST for Mysterium Network and, as a regular cryptocurrency, it is possible to convert MYST to ETH and vice versa.
Although a node can decide to be a consumer or a provider at any moment, the two entities are different, and the type of market is called a two-sided market where bid and ask respectively grow on the other side growth.
(EXPLAIN BETTER)
(BID AND ASK GRAPHIC DEPENDING ON THE FEES)

```
concetti introduttivi
- blockchain in linea di massima\\
- ethereum in linea di massima e descrizione degli smart contracts\\
- proof of work, proof of stake\\
- reputazione in una net (Eigentrust)\\
- - cercare applicazione pratica di Eigentrust\\
- descrizione di BFT\\
```

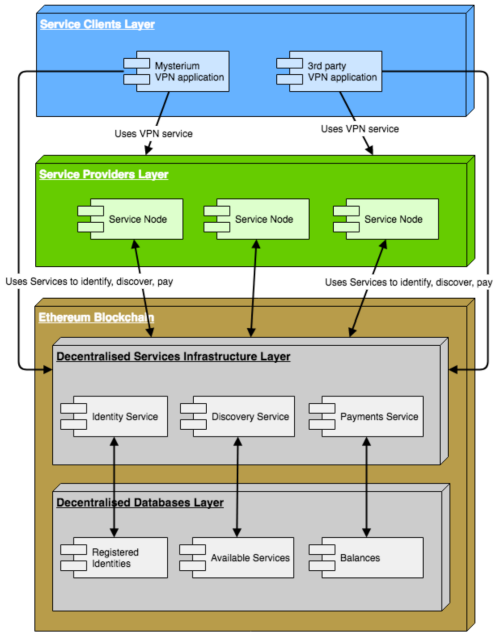
2 Mysterium Network

The main open-source projects which aim to create a decentralized VPN are Mysterium Network, Privatix, Substratum, and Sentinel. We explored all of them, and we can safely say the most advanced project is Mysterium Network, at the time we are writing this research document.

It is the most completed in terms of development and design. Furthermore, it's 100% open source, that for this kind of project is a requirement which conditions the users' adoption and the market diffusion.
Therefore, we decided to use the Mysterium Network as a test case to analyze the malicious context the project is exposed to and propose our proof of concepts as solutions.

2.1 Architecture

Mysterium Network is an open-source project. Its main goal is to build a decentralized infrastructure made up of layered VPN protocols, blockchain, and smart contracts.
As described in figure X, we can distinguish between a service client block and a service provider block. Both of them lean on the Ethereum blockchain block, which is a distributed database where all the network operations, the identities, and the service proposals are recorder.



As a general workflow description, Mysterium starts by collecting all the agents' service proposals and presents them to the clients. Then it registers the client's intention to pay for the agent's service chosen by the user and creates the VPN connection between the two points. The outcome of each transaction is saved in the blockchain.

To implement all the functionalities just said, Mysterium has four core components inside its architecture:

- **Ethereum Blockchain** allows running decentralized code with smart contracts, enabling reliable services and payment handling.
- **Identity service and database of registered identities** ensure the proper identity acknowledgment between client and service provider.
- **Discovery service and database of available services** provide means to announce the availability of VPN services and to pick the most suitable VPN service.
- **Payment service and database of balances** allow secure promise-based micropayments for services.

2.1.1 Identity Service

The **identity service** manages the network identities that every node needs. A single node can create one identity or more, and each of them consists of a public and a private key. The last 20 bytes of the hash function, applied on the public key, is the unique id used to identify the node into the network. To publish a new identity so that other network members can reach it, Mysterium uses the smart contract mechanism. The private and public keys are used to validate the signature for the communication between the nodes.
The blockchain mining process appends the public key and the unique id to the Ethereum blockchain after the node pays a fee to the network. This amount of money has the purpose of giving value to the identity. In such a way, to abandon the identity to create a new one is unattractive, because it is expensive, but not impossible.
We will discuss those aspects later, but we can surely say this is the first barrier to block in mass identity creation.
Furthermore, the user benefits in using the same identity because it is possible to rebuild his transaction history and his balance from the Ethereum blockchain, making the user more predictable and thus trustworthy for service providers.

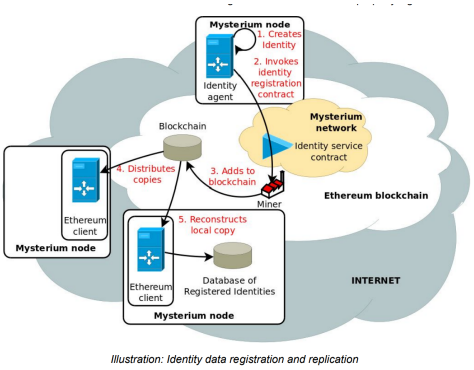


Figure 1: The Mysterium identity creation steps

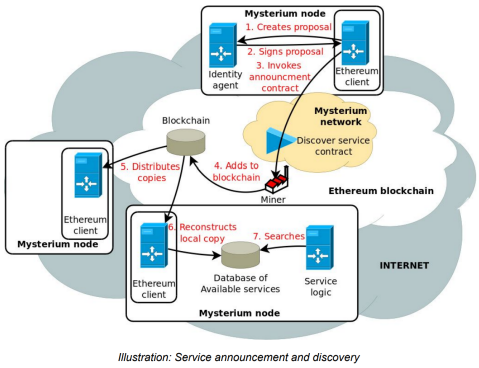


Figure 2: The Mysterium service announcement steps

2.1.2 Service Discovery

The **service discovery** provides the list of service proposals available in the network. The service posts its service offer characterized by some service parameters: those parameters can be the VPN application the provider supports, the provider's connection IP address and port, the provider's server location, or the bandwidth per session. Then, the miners append the proposal to the blockchain, making it publicly available for everyone.

2.1.3 Payments Handling

Ethereum blockchain holds all the mechanisms for the **payments handling**. Therefore everyone needs an account to operate in the Mysterium Network. To advance a request, the consumer has to deposit currency in his wallet. Then, he can ask for a connection to a provider who has previously published a service proposal. Mysterium describes this step as a promise creation because the client effectively promises he will pay when the service ends. Positively clearing a promise is the everybody's best interest, the producer and the consumer, since a canceled promise marks the consumer's history hitting his reputation for future connections.
From what we have described so far, it seems the consumer can decide to avoid paying for the used service: technically, it is possible, but realistically there are some considerations to be made.
First of all, as previously described, identity creation takes money to be completed, and for any new connection, the user's balance has to be positive. In any case, the blockchain records any connection result so that everybody can rebuild the user's history at any moment. Therefore, it is possible to deduct the user's reputation based on its history.
Using this reputation, the provider can decide to block his service to all users that have a low reputation caused by an unpaid precedent. On the other hand, other providers allow new users' connections with a low reputation but at a high price, accepting the risk of having an unpaid transaction.

2.2 Components

The core application is called **node**. It's written in GoLang, a new program language made by Google that is having a high adoption in the last years. The node creates a local webserver to connect to the network, manages the cryptocurrency wallet, and initiates the VPN connection as a client or manages the incoming connection as a server. Mysterium leans on famous VPN applications, which are OpenVPN and Wireguard.
The node can be run in two modes, as a client or as a server, and it is managed via command-line interface or RESTful API, called TequilaAPI, exposed in the localhost.
The main functionalities of TequilaAPI concern:

- the authentication,

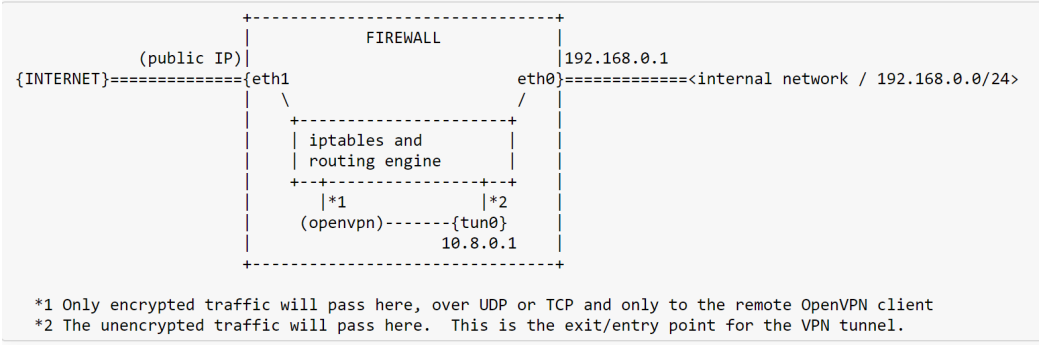
- the connection,
- the identities,
- the location,
- the proposals,
- the service operations in provider mode
- other minor features

TequilaAPI is used by the client's application to manage the connection and for diagnostic operations.
Mysterium developed a desktop client written using Electron, which uses the TequilaAPIs.
The node is also available as a ready-to-use Docker image and published on the Docker Hub.
On the mobile side, Mysterium is present with its multiplatform app that works only as a client.
(MORE ON MOBILE)

2.3 setup

We have chosen the Linux platform and OpenVPN as the VPN application, but similar considerations can be made with different configurations.
When the Mysterium application starts, it calls OpenVPN, which sets up its network environment. OpenVPN creates a virtual network device used to encrypt and decrypt the traffic. These devices reside in the kernel and are entirely virtual, which means there is no hardware on their back.
They can be of two types, TAP or TUN.
The TAP type is used to simulate a link layer device, which means the corresponding ISO model level it refers to is number two. Therefore, a TAP interface handles the transmission of the data frames between two nodes on the same physical layer. The behavior is equal to a bridge device: to connect two separates networks as if they were one.
On the other hand, the TUN type is, in the ISO model, one layer above, working at level three. A TUN interface handles packets, such as Internet Protocol packets. The main functions include port forwarding, host addressing, and traffic control, for example. The behavior is, in this case,

equal to a router device: to connect two separates networks allowing their communication but keeping them independent.
Usually, the TUN type is the most used, since the TAP type brings some limitations such as it cannot be used with Android or iOS devices, causes higher overhead on the VPN tunnel, and scales poorly.
The following schema describes an example of a scenario where the network has two ethernet cards, one public and one internal.
The virtual TUN interface created by OpenVPN is tun0, the public interface is eth0, and the internal interface is eth1.



When a packet arrives at the “eth1” interface, OpenVPN picks it up, decrypts, and sends the data to the tun0 interface. The “iptables” and the routing engine filter and masquerade the packet and send it to eth0, where it becomes available to the final receiver.
In the other direction, when the packet arrives at “eth0” interface, “iptables” and the routing engine apply their rules to filter and masquerade, and then they forward it to “tun0” interface. Thus, OpenVPN picks it up, encrypts the data, and sends it to the public “eth1” interface.
In our scenario, the network configuration is slightly different and does not include an internal interface network. As the schema below describes, OpenVPN sends the traffic coming from “tun0” to the public interface eth0 on a specific port. The client application listens on that port, and the OpenVPN client decrypts the received data.

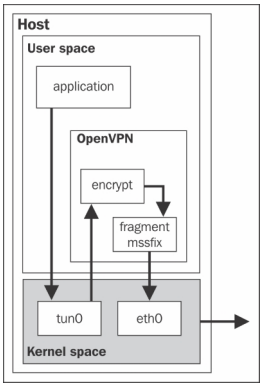


IMAGE TO EDIT AS BEFORE DESCRIBED

2.3.1 iptables

The "iptables" is a framework to manage the network decisions. It is based on the "Netfilter" kernel component, which exposes a set of hooks that are triggered at specific points of the network stack. Every packet that enters the networking system will trigger these hooks as it proceeds through the stack so all the programs that register with these hooks can interact with the traffic.
Five different hooks correspond to different points of the stack:

- `NF_IP_PRE_ROUTING`
- `NF_IP_LOCAL_IN`
- `NF_IP_FORWARD`
- `NF_IP_LOCAL_OUT`
- `NF_IP_POST_ROUTING`

IMAGE OF THE POINTS IN THE FLOW WHERE THE HOOKS ARE
Iptables is mainly used as a firewall which organizes its rule in tables and chains. The tables classify the rules according to the type of decision that regards the packet.
The available tables are the filter table, the NAT table, the Mangle table, the Raw table, and the Security table.
The chains correspond to the available hooks, therefore iptbales has the **prerouting**, **input**, **forward**, **output**, and **postrouting** chains.

The following table explains the triggering logic and execution precedence.

Chains **Tables**	**PREROUTING**	**INPUT**	**FORWARD**	**OUTPUT**	**POSTROUTING**
ruoting decision				X	
raw	X			X	
connection tracking enabled	X			X	
mangle	X	X	X	X	X
DNAT (NAT)	X			X	
routing decision	X			X	
filter		X	X	X	
security		X	X	X	
SNAT (NAT)		X			X

The packet triggers the hooks from the first row going down to the bottom of the table. It enters the chain depending on the routing decisions that are made based on its direction, incoming or outgoing, and whether the filtering rule accepts the packet.
The chain traversal order is the following:

- Incoming packets destined for the local system: PREROUTING - INPUT
- Incoming packets destined to another host: PREROUTING - FORWARD - POSTROUTING
- Locally generated packets: OUTPUT - POSTROUTING

Every chain is a list of rules. Every packet is checked against each rule and, if the matching component checks, iptables executes an action.
The rule can match the packet depending on the protocol type, the source or destination address, or packet, among other criteria.
The action called target, which is triggered if the rule matches, can be of terminating or non-terminating type.
The terminating type target causes the packet to exit the chain after a verdict that can be to allow or to drop the packet. The non-terminating

type target causes the packet to continue its evaluation in the chain. There can be any number of non-terminating targets before a terminating one.
A very used non-terminating target is the jump action: a rule can decide to jump a packet into another chain, and when that chain finishes the rules check, the execution comes back to the previous chain. This way is useful to organize the rules in chains based on their purposes.
LITTLE IMAGE OF CHAIN JUMP ACTION
With "iptables", it is possible to build firewalls and control systems to manage the computer network efficiently.

2.4 Design vulnerabilities

Mysterium is a complex system that integrates components of different types. The distinct nature of these components and their conjunction opens to new vulnerabilities scenarios where the focus is on how the elements dialog each other.
In the following chapter, we provide detailed considerations on the Mysterium Network system concerning its components and their integration. In order to do this, we had to assumed some components behaviours and their impact on the system because, Mysterium, at this time, does not implement them yet. Mysterium project is under development, which plans to have a fully decentralized network in 2021.
We classify the several aspects that, on a project like Mysterium, must be considered to evaluate its design vulnerabilities based on:

- the position in the network: starting, middle or endpoint point;
- the step of the complete flow: the service proposals, the promise issuing, the transaction closing, and other steps;
- the operations on the passing data: sniffing, filtering or crafting the data;
- side channels attack and other particular scenarios.

2.4.1 The position in the network

In the simplest configuration, the network includes a starting point that is the service consumer node and an endpoint, which is the service provider

node. The VPN tunnel connects the two nodes and encrypts their traffic. The service node, as previously described, is the consumer's exit point to the internet.
We consider secure the traffic passing through the VPN tunnel because it is encrypted, not secure otherwise, that is from the provider to the internet.
SIMPLE NETWORK CONFIGURATION WITH AN EXIT NODE
The exit nodes connect two areas with two different security levels: for this reason, it is sensitive by default and without specific precautions can be the weak link of the system.
The Tor network has a similar problem. Tor stands for The Onion Router, and it is a network of peers where the user's encrypted traffic is sent into, and it bounces multiple times before getting an exit node from where, finally, it reaches the external resource.
The traffic is encrypted for each bouncing, except the last one, where it must be in clear to connect to the internet resource. The end node's role is to decouple the encrypted network from clear internet, and this makes it so valuable. If an exit node is malicious, it can potentially understand all the data passing through it and going to the internet resource.
A Sweden researcher, Egerstad [5], in his research, intercepted thousands of diplomatic emails setting up malicious exit nodes around the world and sniffing the connection between them and the clear internet.
To discover malicious nodes in the Tor network, another researcher built up a website with a login and a system to count the login attempts. Then, he used multiple Tor paths with different exit nodes to access the page. The system counted more login attempts that the researcher have done: this means the exit nodes sniffed the connections, caught the username and password, and tried to login by themselves.
In both cases, the use of HTTP instead of HTTPS, or of other unencrypted protocols, has been necessary, but the research aim was to demonstrate the exit nodes' critical role because there still are many protocols and applications that do not use encryption for their traffic.
The conclusion is that Tor's end nodes are the weak link of the system when the users use it as an end-to-end encryption network, rather than an anonymization network, as well as Mysterium Network end nodes.
For this reason, many of the conceptual vulnerabilities held by the exit node of Tor, can be applied to a Mysterium Network exit node. But we want to point out a difference between Tor and Mysterium networks: the type of target users of the two projects.

The Tor's users use it to accomplish specific online activities where the identity obfuscation is mandatory. For the rest of the actions, Tor is not efficient or does not work at all.
The Mysterium Network users, instead, use it for a broader range of online activities, and we can assume a continuous connection to the VPN network with almost no impact on the user's system. In this direction, the implementation of the mobile app is crucial: smartphones are becoming the first device from where people access the internet and the capability to have a device connected all the time with the VPN network provides a security level not yet implemented for consumer uses.
This scenario, however, introduces move vulnerabilities simply because the network is used for more activities, and many protocols and applications still use unencrypted communication.
Figure X describes the network configuration of a Mysterium end node. The provider manages all the traffic from the VPN tunnel, both incoming and outgoing, and because it has not the VPN encryption, he can potentially perform all the manipulation he wants. We will discuss further this scenario analyzing the technical solutions and providing a practical example to exploit in chapter X.
If we ask someone to look at a photo for us and tell us to describe its content, he must, of course, understand the meaning, can change a little detail, or even tells someone else the content of the picture.
An improved configuration implies one or more middle points between the consumer and the provider. Those points are called hops, and they act similarly to the provider node since they forward the traffic.
Mysterium did not implement those hops in the network yet. Therefore we make a hypothesis on its possible implementation.
As well as for the provider nodes, we can suppose there is a distributed list of the forwarder nodes. When the consumer chooses the provider, and they agree to issue a payment promise, the network picks a node from that list to place it as the hop. The middle node receives a new VPN connection from the consumer and creates a new VPN connection to the provider. He technically builds and manages two different VPN tunnels: it takes encrypted data from one side, decrypts them, and encrypts again to send on the other one.
The scenario is similar to the end node because the middle node holds unencrypted data in the forwarding action.
The following picture well describes the situation with even more than one

hop. We would have a segmented connection, called multihop connection, between the tips, where all the hops can perform unwanted operations on the traffic.
IMAGE OF MULTIHOP CONNECTION.
Mandatory would be an additional encryption layer end to end that the consumer and provider have to manage. This second encryption solves the multihop data exposing: the nodes manage encrypted data because they are inside the VPN tunnel. The cons are that it increases the computational load.
The vulnerabilities, in this scenario, are restricted to all the operations that concern the traffic shape, with no abilities to understand it. These actions hit the quality of service, a class of attacks that aim to lower or even block the service.
Many traffic shapes operations can be performed on a connection: the simplest and the fastest is on the traffic speed, and it does not need to understand the content at all.
A demonstration of how a middle node can slow down the data speed that it forwards is using a Linux package called `tc` [4], which stands for Traffic Control and that is created and used for service quality analysis.
With the following command, we instruct the network manager to apply a delay of 500 milliseconds for every packet sent to the interface `eth0`.

```
1   sudo tc qdisc add dev eth0 root netem delay 500ms
```

Here is what each option means:

- qdisc: modify the scheduler (aka queuing discipline)
- add: add a new rule
- dev eth0: rules will be applied on device eth0
- root: modify the outbound traffic scheduler (aka known as the egress qdisc)
- netem: use the network emulator to emulate a WAN property
- delay: the network property that is modified
- 500ms: introduce delay of 500 ms

With the following command, we delete the delay rule.

```
1  sudo tc qdisc del dev eth0 root netem
```

In this simple way, the traffic speed between the consumer and the provider has a poor quality because one or more middle points are performing this slowing down the operation. From the consumer side, the poor connection is a quality agreement breach, and it affects the service provider's reputation. Since the provider can detect the slow speed, he can suppose it is a packet path issue and can propose to the network to change one or more middle nodes.

2.4.2 The flow steps

The first and easiest idea to implement to use the system maliciously has already been presented before: the attacker is the consumer, he creates an account in the Mysterium Network, has a positive balance to the cryptocurrency wallet, and look for a service proposal. They agree on the service cost and issue a payment promise. If the user does not respect the promise... TODO

2.4.3 Operations on the passing data

In this section we analyze the different operation an end node can perform, based on the fact it manages traffic outside the VPN tunnel.
The traffic outside the tunnel is not encrypted by the VPN, but this does not necessarily mean it is not encrypted by itself. Some protocols encrypt data by design. Others do not encrypt or have different structure levels where the lowest have no encryption at all.
Based on this consideration, the operations that a node can potentially do on the passing data can be grouped in sniffing, filtering, and crafting.
The first step to manipulate the data is to catch it. To do so, we start the Mysterium Node on our test server and then establish the connection with a Mysterium client from another computer. We used `tcpdump`, a powerful tool to dump the traffic on a network. For the example purpose, we get an online resource hosted on a different server and available using the HTTP protocol, so that the dumped traffic is in clear.
The command is the following:

```
1  todo
```

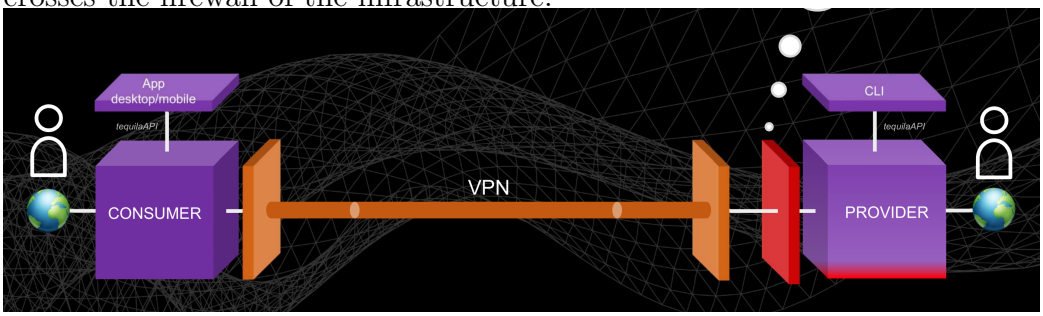
where the parameters are:....
The output is:

```
1   todo
```

2.5 Basic sniffing to catch content (Wireshark, tcpdump)

2.5.1 Side channels attack and particular scenarios

In this section, we discuss the possible scenarios that do not directly regard attacks on specific components, but more on how these harmless operations in Mysterium Network lead to vulnerabilities.
In Mysterium Network, the billing method is per the amount of transferred data or per time. Considering the first method, if the exit node is able to higher the volume of the traffic to the consumer, it means the exit node earns money without providing an equal service. A well-designed attack can perform the traffic shaping hiding the increase of traffic volume with network inefficiency.
In the "tcp/udp" subsection in the Craftberry tool description, we implemented an attack that uses this technic to make multiple copies of the packets using tcp and udp protocols with no content understanding. A more refined attack would be modifying the packet specifically for the protocol, forcing the error in the communication so that the protocol corrects it. In this document, we did not focus on those particular cases which are deeply protocol dependent, but the modification aims to increase the transmitted data because the bill is calculated on that.
VLAN tag attack when the client is into an enterprise network. The attack crosses the firewall of the infrastructure.



3 Craftberry

3.1 The idea

Craftberry is a proof of concept tool to demonstrate how a node can sniff, filter and craft the traffic on the exit node.
It is an application which runs via the command-line interface, that catches the incoming or outgoing packets. The packet catching is in line with the packet flow, and it executes different actions accurately implemented to cover many attack cases.
The software is entirely written in C and C++.
We propose part of the output provided by the application when the help command is used:

```
1   -a: Use the specified action
2   -i: Use the specified interface, tun0 by default. Can be
        interface name (e.g. eth0) or interface IPv4 address
3   -t: Use the specified timeout in seconds, if not defined
        it runs until some external signals stop the execution
        (e.g. ctrl+c)
4   -l: Write all the crafted and generated traffic into a
        pcapng file having name passed by parameter or, if the
        parameter is equal to 'default', the name is 'out_<
        epoch_ms>.pcapng'
5   -d: Direction filtering by and perform the crafting {IN,
        OUT}, default = IN
6   -v: Shows verbose debug application logs
7   -h: Displays this help message and exits
```

The parameter set by `-a` defines the action Craftberry has to perform when it catches a packet. We divide the actions into two categories: attack or defense. The attack actions are the most interesting for this research document, but we also implemented a possible solution that protects the passing traffic encrypting the packet content.
The attacks differ on which ISO-OSI layer the packet is crafted.
IMAGE OF THE ISO OSI LAYERS MODEL
The attack actions are:

- Layer 2: VLanDoubleTagging
- Layer 3: IP, icmp

- Layer 4: tcp/udp
- Layer 5 and above: HTTP, DNS, NTP

The defense action is for Layer 4. It encrypts the payload of the packets with `ChaCha20` algorithm, a Salsa20 modification published in 2008. It is a particularly efficient stream cipher.

3.1.1 VLan double tagging

use the double tagging attack method on cisco routers

3.1.2 IP

3.1.3 icmp

3.1.4 tcp/udp

The context of the attack supposes that the client and the server have an end to end encryption connection, thus supplementary to the VPN encryption. The exit node cannot then perform any content understanding, and only actions on the entire packet can be done, as described in the Design vulnerabilities chapter.
The attack accomplishes a simple copy of the packet: when the provider receives packets from the internet, it sends `n` times a copy of copied packet to the consumer.
The implementation is as simple as significant: with this method, with no content understanding, the passing data are more than the necessary, and the service consumer is affected.
We have implemented a demonstrative example of this attack using the protocols tcp and udp.

3.1.5 nfqueue extension

NFQUEUE is a kernel and user mode module for managing network packets in iptables. It allows writing `Netfilter` target modules in userspace. This module is an application that is called by `nfqueue` hooks. In the program, the hooks are bonded to a function that is called when the iptables match a rule.

The function gets the queue details and the packet payload, craft the packet, and then send a verdict to the iptables module.
FLOW OF THE PACKET (CATCHING, CRAFTING AND THEN VERDICTING IT)
It uses the PcapPlusPlus library to parse and craft the packet.

- http://linux-training.be/networking/ch14.html

http://unixwiz.net/techtips/gnu-c-attributes.html
https://gcc.gnu.org/onlinedocs/gcc-4.0.2/gcc/Type-Attributes.html
https://groups.google.com/forum/#!msg/pcapplusplus-support/e7rN93LfTSg/MFnVEKCNCAA
https://byt3bl33d3r.github.io/using-nfqueue-with-python-the-right-way.html

3.2 Description

3.2.1 liv 2: arp, vlan tagging

3.2.2 liv 3: ip, icmp

3.2.3 liv 4: tcp/udp

3.2.4 side attacks

- auditin/collezione dei dati sniffati, inviati ad un server esterno per collezionamento, statistiche, vendita a terzi. - trojan all'interno di craftberry, per botnet. (approfondire aspetto che se il sw di attacco diventa popolare, può essere lui stesso veicolo di malware, quindi attacco l' attaccante)

3.2.5 liv 5+: http/imap/dns/ntp/etc

analisi e modifica dei contenuti a livello del singolo protocollo (ad esempio sostituzione di tutte le immagini con una immagine di default)

3.3 Tests

3.4 Design solutions for malicious context

3.4.1 comparing data

invia la richiesta a due nodi e le confronta (essenziale possibilità di connessione a due nodi in contemporanea: due o più container docker

mappati su porte diverse con uno script che invia la richiesta a più container e confronta le risposte)

3.4.2 comparing hashes

come precedente ma con la gestione degli hash invece che di tutto il dato

3.5 Design integration with reputation systems

4 Conclusions

4.1 Future development

4.2 Considerations

The use of Mysterium Network like an end-to-end encryption tool: https://nakedsecurity.sophos.com/2015/06/25/can-you-trust-tors-exit-nodes/ Here there is a helpful consideration of how critical is an exit point for TOR.

problemi individuati

1. valutazione della reputazione dell'agent da parte del client
2. valutazione della reputazione dell'agent da parte della Net

soluzioni proposta Proposte per la valutazione della reputazione dell'agent da parte del client
Le seguenti proposte sono metodi di verifica del corretto comportamento ed hanno come scopo finale la valutazione della reputazione degli agent

- Dati civetta: modifiche al client che con scadenze richiede una o più risorse dal valore noto e verifica che siano integre. Le scadenze possono essere regolari/random/all'inizio frequenti/dipendenti dalla reputazione dell'agent. È necessario avere delle risorse distribuite e disponibili: potrebbero essere i nodi stessi della rete (altri agent).
- Dati duplicati: modifiche al client che implementa la possibilità di connessione con più agent. Dopo la richiesta il client compara i dati ottenuti dalle due fonti
  - Hash *: (soluzione aggiuntiva) I nodi sono generalmente in posizioni migliori dei client in termini di velocità. L'idea è quella di far generare un hash dei dati che il client richiede ad un altro nodo fuori dal servizio primario, così da comparare gli hash e non tutto il dato. Inoltre, se la richiesta la faccio a molti più nodi posso intrinsicamente verificare quali modificano i dati nel network.
- Applicazione di modelli di reputazione (Eigentrust): attualmente non studiato

References

[1] https://tools.ietf.org/html/rfc2818

[2] https://tools.ietf.org/html/rfc2764

[3] manually managing references, `https://en.wikibooks.org/wiki/LaTeX/Manually_Managing_References` 2016

[4] traffic control
https://jvns.ca/blog/2017/04/01/slow-down-your-internet-with-tc/

[5] exit node TOR network https://www.wired.com/2007/09/rogue-nodes-turn-tor-anonymizer-into-eavesdroppers-paradise/?currentPage=all