

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

Jacopo Federici

February 19, 2020

Contents

0.1	Note	2
1	parte introduttiva	3
1.1	Abstract	3
1.2	Work structure	3
1.3	Outline	3
1.4	Goal	3
2	State of art	3
2.1	Introduction	3
2.2	The world now	4
2.3	the need for trust and privacy	5
2.4	We are our trust	5
3	My work, steps	5
3.1	Decentralized VPNs	5
3.1.1	Introduction	5
3.1.2	The business idea	6
3.2	Mysterium Network	6
3.3	Architecture	7
3.4	Basic sniffing to catch content (Wireshark, tcpdump)	12
3.5	Craftberry, a demo tool	12
3.5.1	liv 2: arp, vlan tagging	12
3.5.2	liv 3: ip, icmp	12
3.5.3	liv 4: tcp/udp	12
3.5.4	side attacks	12
3.5.5	liv 5+: http/imap/dns/ntp/etc	12
3.6	Design solutions for malicious context	12
3.6.1	comparing	12
3.6.2	comparing hashes	12
4	Design integration with reputation systems	13
4.1	something here	13
5	Conclusions	13
5.1	Tests	13
5.2	Future development	13
5.3	Considerations	13

0.1 Note

parlare di:

- docker (cosa è, comandi base per fare cosa)

- robe di networking per dev mode del progetto:
 - <https://www.digitalocean.com/community/tutorials/a-deep-dive-into-iptables-and-netfilter-architecture>
 - <http://www.naturalborncoder.com/virtualization/2014/10/17/understanding-tun-tap-interfaces/>
 - <http://linux-training.be/networking/ch14.html>

1 parte introduttiva

1.1 Abstract

1.2 Work structure

1.3 Outline

1.4 Goal

2 State of art

2.1 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, following the increase of security attacks. For instance, in 2017, the popular Chrome browser announced it would mark as non-secure any website using the HTTP protocol instead of HTTPS. This move was to encourage the website to use SSL/TLS certificates and make secure the communication between client and server. It's interesting to discover the Netscape Navigator browser has made the first HTTPS version around 1990, but the official RFC publication has been released only in May 2000 [1]. ((Why has this solution been largely used only fifteen years later?)) Similar scenario for the VPN: abbreviation for "virtual private network" it's a technical solution to extend a private network across a public network. It is created by establishing a virtual point-to-point connection through the use of dedicated circuits or with tunneling protocols over existing networks. Many of the VPN implementations encrypt the traffic between the two points. A first VPN specification proposal can be dated around 2000 [2], but only in the last years, the use of VPN is becoming massive. The data encryption feature solves privacy problems the VPN was not intentionally designed for, such as crossing a censored network rather than use services available only in certain countries. Currently, many commercial solutions allow users to have a private connection between their computer and the company servers across the world. In this way, the user can have access to the internet resources from the position around the globe

he prefers, skipping firewalls, proxies, and other tracking and censure methods applied by some companies or countries. There are many security concerns about those commercial solutions, such as the ability of the VPN provider to keep logs, inspect the traffic, or sell statistic data obtained by the users. 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 be implemented to patch those critical aspects and provides a comparison between different solutions with an eye on the practical perspective.

2.2 The world now

We still are in the middle of a huge transformation. Since the invention of the pc and the born of the internet. More and more services are becoming digital; more and more people are using those services; more and more bad guys are trying to pull down those services.

But there are differences from some decades ago. First, everything is now much faster. The internet infrastructure can grant us to move significant amounts of data within a few seconds, and access to online services is used and necessary.

Furthermore, our personal computer can perform operations only some years ago were impossible to execute on mainstream servers. This speed is finding the limits of many security solutions, such as ciphers that based and still do, their strength on the amount of time needed to reverse the encryption. Second, the architecture is changing back to a cloud-based infrastructure. Many workflows see, again, a thin client sending the execution step on a cloud server that elaborates the output to send back to the client. This architecture changes the attack surface addressing 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 paradigm, 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".

Million of users worldwide resort to security solutions to accomplish online operations of any sort. Unfortunately, the existing technologies often show their design vulnerabilities.

<https://www.geeksforgeeks.org/need-of-information-security/>
<https://www.hackerfactor.com/blog/index.php?/archives/868-Deanonymizing-Tor-Circuits.html>

2.3 the need for trust and privacy

=

<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

Anonymization: in every network systems, the anonymization is a tough challenge. Even more, if privacy is a crucial point. In some countries, the VPN is massively used to protect data from surveillance at any level, from a public unsecured wifi connection to the country ISPs. There are countries, the People's Republic of China, for instance, where censorship is present and where identity obfuscation is necessary. In these cases, the use of a VPN is remediation as long as a safe company provides it. With the adjective "safe", we mean a company that does not make detectable your traffic into his network, or even worst sniffs and crafts your traffic for malicious actions.

The decentralized VPN completely changes the paradigm, the infrastructure design. A similarity 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 the trusts all of us put in the bank on ourself, cryptocurrency participants, 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 to based on different indexes.

2.4 We are our trust

3 My work, steps

3.1 Decentralized VPNs

3.1.1 Introduction

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\\

A decentralized VPN is a network of nodes where the users can connect through a VPN to nodes without a centralized entity. 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, it connects to the network providing its availability such as consumer or provider. The consumer user chooses the provider he prefers based on different features such as the node's region, its quality, or its cost, 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, even called client and server, is established. When it correctly ends, the network closes the smart contract, and a result of the transaction is registered in the blockchain.

We have outlined the main steps from which we can deduce a decentralized VPN is more 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)

3.1.2 The business idea

As a business project, which aims to earn money, it has a specific business plan shared with similar projects: 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. Substantially, they want 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 ruled by an Ethereum smart contract signed by both of the sides and fee is required to close it. This fee is not the Ethereum fee, generally called gas, but the Mysterium one, its source of earnings. For this reason, Mysterium and other projects created its coin used to buy and sell internet bandwidth between the nodes inside the network. Its 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)

3.2 Mysterium Network

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

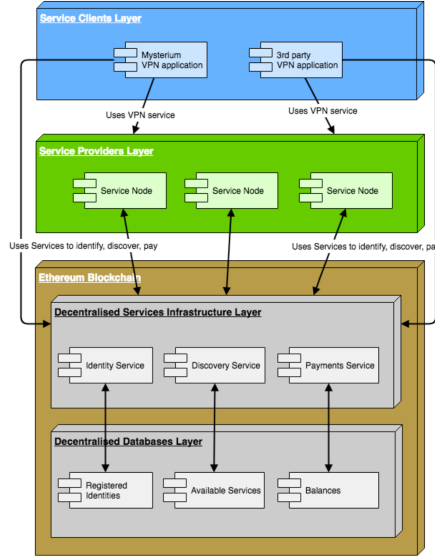


Figure 1: A boat.

writing this research document, is Mysterium Network. It is the most completed in terms of development and design. Furthermore it's 100% open source, that for this kind of project is an important quality which conditions the users adoption and market diffusion.

Therefore, we decided to use Mysterium Network as a test case to analyze the malicious context the project can face and propose our proof of concepts for solutions.

3.3 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 a service client block that uses the service provided by the service provider block. Both of them lean on the Ethereum blockchain block, which is the distributed database where are recorder all the network operations, the identities, and the service proposals.

As a general workflow description, Mysterium starts by collecting all the agents' service proposals and presents them to the clients. Then 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 realize all the functionalities just said, Mysterium has four core compo-

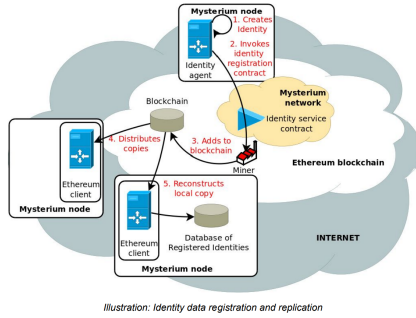


Figure 2: The Mysterium identity creation steps

nents inside its architecture:

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

The **identity service** manages the network identities that every node needs. A single node can create one 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 and takes fees the nodes must pay to the network for this operation. This amount of money has the purpose of giving value to the identity. In such a way, it is unattractive to abandon it, expensive, but not impossible.

We will discuss those aspects later, but we can surely say this is the first barrier to make prohibitive in mass identity creation.

Furthermore, the user benefits using the same identity because it is possible to rebuild its transaction history and its balance from the Ethereum blockchain, making him more predictable and thus trustworthy for service providers.

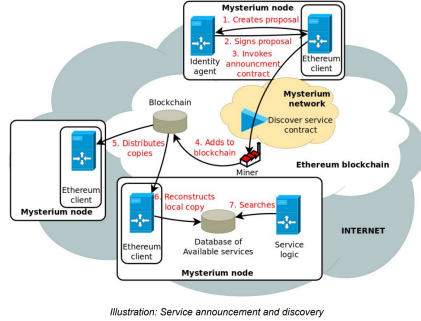


Figure 3: The Mystery service announcement steps

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 it to the blockchain, making it publicly available for everyone.

Ethereum blockchain holds all the mechanisms for the **payments handling**. Therefore everyone needs an account to operate in the Mystery Network. To advance a request, the consumer has to deposit currency in his wallet. If so, he can ask for a connection to a provider who has previously published a service proposal. Mystery calls this step a promise creation because the client effectively promises he will pay when the service ends. Positively clearing a promise is the interest of everybody, producer and consumer, since a canceled promise marks the consumer's history hitting his reputation for future connections. From what we have described seems the consumer can decide to avoid paying for the used service: technically, it is possible, but realistically there are some considerations to note.

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 its 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 with a high price, accepting the risk of having an unpaid transaction.

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 cryp-

to currency wallet, and initiates the VPN connection as a client or manages the incoming connection as a server. Mysterium does not implement its version of VPN but leans on famous VPN applications, which are OpenVPN and Wireguard.

It can be run in two modes, as a client or as a server, and is managed via command-line interface or via RESTful API exposed in the localhost, called TequilaAPI. TequilaAPI is used by the client application to manage the connection and for diagnostic operations. Mysterium developed a desktop client written using Electron, a technology that is a web application locally executed by NodeJs, a server backend.

The node is also available as Docker image ready-to-use and published on the Docker Hub. On the mobile side, Mysterium is present with its multiplatform app that works only as a client.

Project setup

We have chosen the Linux platform and OpenVPN as the VPN application, but the similar deductions can be made with different configurations.

When the Mysterium application starts, it calls the OpenVPN, which setups 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, TUN or TAP.

The TAP type is used to simulate a link layer device. That means the corresponding ISO model level it refers to is the number 2. Therefore, a TAP interface handles the transmission of data frames between two nodes on the same physical layer. The behavior is the equal to a bridge: to connect two separates networks as if they were one.

On the other hand, the TUN type is one layer up on the ISO model, working at level 3. A TUN interface handles packets, such as Internet Protocol packets. The main functions include host addressing and traffic control, for example. The behavior is, in this case, equal to a router: to connect two separates networks keeping them independents.

OPENVPN WORKING EXPLANATION:

https://subscription.packtpub.com/book/networking_and_servers/9781783553136/1/ch011v11sec12/

The VPN server application creates a virtual interface that encrypts and decrypts the traffic.

The server works as a proxy: forwards all the data that comes from the VPN tunnel to the internet and vice-versa. The client, on the other side of the tunnel, performs the same operations, but inverted.

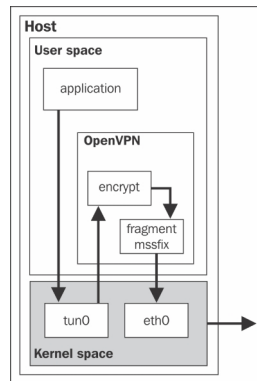


Figure 4: A boat.

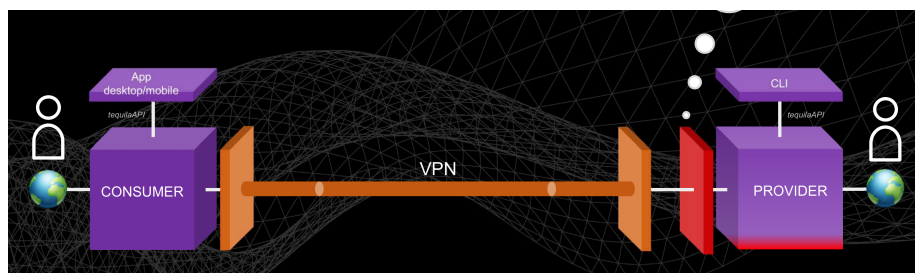


Figure 5: A boat.

3.4 Basic sniffing to catch content (Wireshark, tcpdump)

3.5 Craftberry, a demo tool

useful tips

<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/MFnVEKCNCAAJ>

<https://byt3bl33d3r.github.io/using-nfqueue-with-python-the-right-way.html>

3.5.1 liv 2: arp, vlan tagging

3.5.2 liv 3: ip, icmp

3.5.3 liv 4: tcp/udp

3.5.4 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)

- copia x volte dei pacchetti (TCP/UDP)
- gonfiamento dei pacchetti nei campi che poi verranno corretti dai check (TCP/UDP)
- modifica del payload se in chiaro (TCP/UDP)
- azione positiva sul payload (UDP), ad esempio cifratura chacha20

3.5.5 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)

3.6 Design solutions for malicious context

3.6.1 comparing

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)

3.6.2 comparing hashes

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

4 Design integration with reputation systems

4.1 something here

5 Conclusions

5.1 Tests

5.2 Future development

5.3 Considerations

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 intrinsecamente 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