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

Jacopo Federici

February 7, 2020

Contents

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

The need for privacy is becoming more important day after day. The heavily digitalization of many aspects of our life implicitly exposes us and our data. The basic actions performed on a computer which involve an internet connection, as a majority of them, are becoming unsafe.
In the last years we have seen security solutions integration in popular applications, such as browsers or mobile apps, following the increasing 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 website to use SSL/TLS certificates and make secure the communication between client and server. It's interesting to discover the first HTTPS version has been made by Netscape Navigator browser, a pioneer browser at the beginning of internet, around 1990 and the official RFC publication has been released in May 2000 **??**. ((Why this solution has 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 **??**. In the last years, this feature is making the use of VPN very large because it solves privacy problems the VPN was not intentionally design for, such as crossing a censored network rather than use services available only in certain countries.
Currently, there are many commercial solutions that 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 geoposition he prefers, skipping firewalls and other censure methods applied by some countries, for instance China with the Great Fire Wall.
There are many security concerns about those commercial solutions, such as the ability of the company to keep logs, inspect the traffic or sell statistic data obtained by the users.
Starting from these problems, some open source projects were, lately, born. Their goal is to revolution the idea of VPN and its uses, building a rich network of peers where two nodes can connect to each other through a VPN. The project provides a platform including all the components needed to have a efficient, fast and easy to use system, from the discovering to the payment.

In this research document we analyzed the most developed project, Mysterium by Mysterium Network, as case of study. As we write, Mysterium offers a working network composed by hundreds of nodes around the world and it's the best project where to investigate malicious contexts and design vulnerabilities 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 can be implemented in the system to patch those critical aspects. A comparing between different solutions is provided with an eye on the practical prospective.

1.1 Abstract

1.2 Work structure

1.3 Outline

1.4 Goal

2 State of art

2.1 The world, now

(stato attuale, attacchi etc)

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

2.2 the need of trust and privacy (vpn, blockchain, decentralizzazione, fiducia in sistemi democratici)

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

It is natural that concepts as "consensus" and as "distributed" gain value and it's undoubted the main technology to manage such kind of situation is the blockchain in the Ethereum variant thanks to its contract mechanism.

2.3 just more, we are our trust (come funziona blockchain, perchè in contesti come dvpn serve decentralizzazione e consenso)

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

The main open source projects are Mysterium Network, Privatix, Substratum and Sentinel. We explored all of them and we discovered the four different approaches they implement to reach the goal: building a distributed network based on blockchain with a discovering system, a crypto coin payment method, a mobile and desktop application. We can safely say the most advanced project is Mysterium Network, 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 decide to use Mysterium Network as test case to implement our proof of concepts.

3.1.2 The business idea

As a business project, which aim to earn money, every project has a specific business plan but the common idea is the same: 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. Each couple of entities, consumer and provider, agree on a transaction paid in crypto coin in the face of the used service. This transaction is ruled by an Ethereum smart contract signed by both of the sides. To close the transaction a fee to the system must be paid. This fees is not the Ethereum smart contract fee, called gas, but the Mysterium source of earnings. For this reason Mysterium and other projects realized an internal market with a private coin (for Mysterium is called MYST) used to buy and sell internet bandwidth inside the network. It's possible to convert MYST to ETH like normal crypto coin exchange.
The type of market is called two-sides market where supply and demand respectively grow on the other side growth. It's the typical market for digital goods and usually only one actor tends to dominate more than half of the market.

3.1.3 Mysterium Network

Architecture Mysterium Network in an open source project. The main goal is to realize a decentralized infrastructure made up of layered VPN protocols, blockchain and smart contracts.
There are four core components composing the infrastructure:

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

Mysterium collects all the agents service proposals and presents them to the clients, register the client's intention to pay for the agent's service choosed by the user and create the VPN connection between the two points.
It is saved in the blockchain the outcome of each transaction. In this manner it's possible to rebuild the transaction history for any specific user, be it a consumer or a producer.

Substratum caratteristiche comuni BTF Attacchi di tipo...

3.2 Basic sniffing to catch content (Wireshark, tcpdump)

3.3 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`

3.3.1 liv 2: arp, vlan tagging

3.3.2 liv 3: ip, icmp

3.3.3 liv 4: tcp/udp

3.3.4 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

3.3.5 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)

3.4 Design solutions for malicious context

3.4.1 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)

3.4.2 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 intrinsicamente verificare quali modificano i dati nel network.
- Applicazione di modelli di reputazione (Eigentrust): attualmente non studiato

Valutazione di un nuovo utente della rete

Contesto Rete VPN decentralizzata attiva e con connessione multi-hop tra una generica coppia Agent-Client.

Problema 1: identificare gli Agent in posizione endnode che inviano contenuto non originale In una rete VPN decentralizzata gli Agent endnode hanno il compito di recuperare una risorsa web e restituirla al Client attraverso la rete VPN instaurata. L'Agent, che quindi ha accesso intrinseco al dato ottenuto, cifrato o meno, può modificarlo prima di inviarlo al Client. Questa operazione può avere scopi malevoli, tra cui quello del guadagno economico. In possibile scenario, in questo contesto, è quello che vede l'incremento della grandezza del pacchetto da Agent a Client al fine di maggiorare i consumi della banda stabilita nel contratto iniziale di fornitura del servizio.

Problema 2: scoraggiare abusi della rete Per scoraggiare abusi della rete si utilizza una *prova di lavoro* (Proof of Work) per attestare che il nuovo utente della rete ha speso delle risorse al fine di guadagnare reputazione nel contesto distruibuito.
La reputazione iniziale, all'ingresso della rete distribuita, ha valore zero.

Soluzione problema 1

- Dati civetta: è richiesta dal Client all'Agent, con frequenza variabile, una risorsa dal valore noto al Client per la verifica di integrità. Se il test risulta negativo si ipotizza la modifica della risorsa da parte dell'Agent.
- Dati duplicati: il Client si connette a due Agent diversi e chiede ad entrambi la risorsa desiderata, alla ricezione si comparano per la verifica di integrità. Se il test risulta negativo si ipotizza la modifica della risorsa da parte di uno dei due Agent.
  Aspetti negativi: questa soluzione non è applicabile in caso di connessioni con la risorsa che prevedono *stati*.(??)
- Hash Dato: il Client si connette ad un Agent al quale chiede una risorsa e successivamente chiede ad un altro Agent l'hash della risorsa stessa. Il Client effettua quindi un hash (stessa funzione hash) della risorsa ottenuta dall'Agent e la compara con l'hash ottenuto dal secondo Agent. Ulteriore implementazione può essere effettuata richiedendo a più agent l'hash della risorsa.
  Se il test risulta negativo si ipotizza la modifica della risorsa da parte del primo Agent.
  Aspetti negativi: questa soluzione non è applicabile in caso di connessioni con la risorsa che prevedono *stati*.
  Caso d'uso: si reputa questo metodo adatto a verificare un Agent al suo primo ingresso nella rete in quanto non è spesso adatto per l'aspetto negativo sopra descritto.

Soluzione problema 2 L'originalità della soluzione proposta è l'operazione di Proof of Work da far compiere al nuovo utente. Si propone come prova di lavoro che l'utente, prima di entrare attivamente a far parte della rete o durante la sua partecipazione, consumi della banda internet.
La Proof of Work consiste quindi nello sfruttare la sua banda di rete per ottenere una risorsa internet, sulla quale viene verificata l'integrità, come descritto nella soluzione *Dato Hash* del problema 1.
Più specificatamente, nel caso d'uso si identificano un Agent A, un Client C ed un nuovo utente X che vuole entrare nella rete e deve dimostrare *la prova di lavoro*.
La situazione iniziale prevede una connessione tra A e C. Per verificare che A fornisca dato d^A originale, C richiede a X di ottenere lo stesso dato d^X e di eseguirne la funzione hash così da ottenere $H(d^X) = h_1$. h_1 sarà restituito a C che ha già effettuato la stessa funzione hash su d^A proveniente da A ed ha ottenuto $H(d^A) = h_2$.
Il test di integrità verifica che $h_1 = h_2$.
Se il test ha successo si ha certezza che:

1. il nuovo utente X ha eseguito correttamente la proof of work
2. A ha fornito a C un dato originale, a meno che sia A che X applichino le stesse modifiche al dato prima di effettuarne la funzione hash.
   Questa vincolo è da verificare.

Considerazioni In un contesto di rete distribuita si possono ipotizzare n attori nel ruolo di X e n test di integrità che C effettua per verificare l'integrità del dato ottenuto da A. *riferimento al consenso BFT*
Il risultato dei test condiziona la reputazione di ogni attore nel ruolo di X.
Si può ipotizzare l'applicazione della soluzione m volte, tante quante sono le entità per cui effettuare la proof of work, e considerare vincoli come tempo t minimo di esecuzione della proof of work o la quantità minima di dati q che X deve ottenere su cui applicare la funzione hash.

Considerando una costo ipotetico c associato al consumo di una quantità q di dati per un'unità di tempo t si può impostare una soglia $S = c * q * t$ superata la quale la proof of work si può considerare eseguita.
L'ultima affermazione garantisce l'asimmetria della proof of work (??). (che è: hash difficile da calcolare perchè spendo risorsa di tempo e di banda e facile da verificare dal client perchè effettuo un semplice confronto di un hash)(vedere https://it.wikipedia.org/wiki/Hashcash per similitudini sulla variabile del tempo)

La reputazione di ogni entità all'interno della rete, sia Agent che Client, subisce una variazione in quanto ciascun esito delle proof of work viene registrata nella blockchain e questo consente di avere una classifica di affidabilità delle entità quasi in tempo reale.