VPN Decentralizzate

Jacopo Federici

January 28, 2020

Abstract

Work structure

Outline

Goal

Contents

0.1 Note

parlare di:

- docker (cosa è, comandi base per fare cosa)
- robe di networking per dev mode del progetto:
  - openvpn
  - http://www.naturalborncoder.com/virtualization/2014/10/17/understanding-tun-tap-interfaces/

Some text A virtual private network (VPN) allows 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. The data passed inside the tunnel is usually encrypted and this quality has been on the ground for its widely diffusion.

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 country he prefers, skipping firewalls and other censure methods applied by some countries, such as 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 by these problems, some open source projects have, lately, been 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 others through a VPN. The Network provides technical solutions to avoid data sniffing or crafting, ensure identity anonymisation and secure payments.

Before to deep into the technical aspects why such a projects haven't been developed before, it's important to understand the scenario and the needs of the actors. We use an example to explain it clearly.
Alice wants to freely surf on social networks but her country blocks the major of them. Technically speaking, what she needs is to exit into internet with an IP address coming from some countries where those service are not blocked, exactly what a VPN service provides, and she is in favour to pay for it. The exit point, in our example, is Bob who agrees to let Alice connect to him to exit into internet, and he wants to be paid for it.
The Networks presented before create the ideal network platform where a consumer, Alice, and a producer, Bob, can meet each other, deal with a contract and rate the opposite quality at the contract closing.
Generalizing, we can assume that bigger is the network and better are all of those aspects we are striving to get, such as anonymisation, anti-sniffing and anti-crafting solutions and so on.
In this context another actor stands watching everything happening. He has a black hat, glasses and a cup of coffee close to him. His name is Mallory and, as we'll describe later, he's trying to find ways to use the system in a manner the system has not been designed. In other words, he's hacking the system. At least, tryin' to.

1 State of art

1.1 The world, now (stato attuale, attacchi etc)

https://www.geeksforgeeks.org/need-of-information-security/

1.2 the needing of trust and privacy (vpn, blockchain, decentralizzazione, fiducia in sistemi democratici) dell'arte)

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.

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

2 My work, steps

2.1 Basic sniffing to catch content (Wireshark, tcpdump)

2.2 Craftberry, a demo tool (snippets of code here)

2.2.1 liv 2: arp, vlan tagging

2.2.2 liv 3: ip, icmp

2.2.3 liv 4: tcp/udp

- 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

2.2.4 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)

2.3 Design solutions for malicious context

2.3.1 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)

2.3.2 comparing hashes

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

3 Design integration with reputation systems

3.1 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

At the time we are coding, the main projects are Mysterium Network, Privatix, Substratum and Sentinel. We explored all of them and we discovered the four different approaches they have to reach the goal of 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 Project to implement our proof of concepts.
All of the projects have in common the idea behind the business model based on the users' abilities to sell their internet or to use a VPN service, depending on their needs and, accordingly, on their role into the network.

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. It's the baseline for all kind of world-first service to build on top of it.
There are four core components:

1. **Ethereum** 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.

Nel contesto che si vuole descrivere si identificano due entità: un Client A che usa il servizio ed un Agent B che lo fornisce.
Mysterium utilizza un Service Provider, basato su blackchain ethereum, che registra le intenzioni del Client A di pagare l'Agent B per il servizio che B offre ad A.

Una promessa è una quadrupla contenente i seguenti elementi:

- P è firmato da A
- Una promessa P può essere aggiornata in caso di prosecuzione del servizio
- B può bloccare l'aggiornamento della promessa P
- P è valida se è firmata da A e se non è stata aggiornata dopo essere stata bloccata
- L'attributo SN incrementa ad ogni modifica
- Per identificare uno stato non valido basta verificare il SN se è maggiore del SN nel momento del blocco
- La chiusura di un contratto P, con P valido, termina con il pagamento della somma MYST da A a B
- Ogni transazione ethereum per il pagamento di MYST, generata a seguito della chiusura di un contratto, comporta il pagamento di fees e gas
- Si crea un equilibrio tra trasferimento di poco denaro ma frequente e trasferimento di molto denaro ma poco frequente
  - Il primo comporta un totale di fees maggiore rispetto al secondo ma corrisponde ad una più alta garanzia di pagamento rispetto al secondo (se un client A non paga, non paga poco denaro, invece che tanto) [merita di essere analizzato meglio]
- Le promesse possono essere condivise con più Agents B senza perdita di valore
  - La condivisione consente di evitare situazioni in cui un Client A con pochi denari instauri molteplici promesse (false) con diversi Agents B: [DA CAPIRE MEGLIO]

- Gli Agents B possono decidere il grado di rischio a cui esporsi (ovvero al rischio che i Clients non paghino)

La promessa non garantisce, di per sé, il pagamento: infatti se A non ha abbastanza MYST nell'account per il pagamento, la sua promessa non viene mantenuta e la sua identità, in successive promesse, viene compromessa. Si utilizza quindi un registro, chiamato Identity Registry (IR) che tiene traccia delle identità [capire meglio che dati dell'identità vengono usati] che usano il servizio, attraverso il quale scoraggiare l'emissione di promesse non mantenibili.

Privatix Privatix utilizza due tipi diversi di smart contract con due funzionalità diverse.

- PTC: Privatix Token Contract è usato per scambiare token, upgrade o nuovi contratti di servizio
- PSC: Privatix Service Contract è usato per immagazzinare balance (depositare e ritirare) e per la gestione del channel...

I coin PRIX possono essere acquistati e venduti sono con PTC. I servizi Privatix, invece, possono essere pagati solo usando PSC. L'idea quindi è quella di usare lo smart contract PSC per le operazioni interne e lo smart contract PTC per le operazioni con l'esterno, con il fine ultimo di aumentare la sicurezza.
Attacchi Sybil: Attualmente non esistono tecnologie per mitigare l'attacco che non prevedano l'introduzione di qualche svantaggio: così dicono loro. Separano l'attacco in due categorie, in base all'entità malevola.
Agents Malevoli: A tutti gli agenti è richiesto di registrare il servizio offerto nella blockchain Ethereum e di depositare una somma di denaro che è possibile ritirare in seguito così da diminuire la presenza di Smart Operation (SO) fasulle. Il deposito è proporzionale alla quantità di servizio offerto. Se l'Agent malevolo posiziona SO fasulle, gli sarà richiesto di posizionare la stessa quantità di denaro che il Client ha posizionato quando ha accettato l'offerta dell'Agent. Le SO hanno associata una età e quindi gli Agents devono mantenere la loro SO viva, altrimenti i Clients la considereranno irrilevante. Ad ogni ri-notifica degli Agents, essi devono bloccare (posizionare) nuovamente la stessa quantità di denaro.
Quando un Client crea uno state channel con un Agent e non riceve il servizio accordato, allora chiuderà lo state channel con lo stato Uncooperative. In questo caso l'Agent sarà notificato da un evento blockchain così da intaccare la sua reputazione e forzare l'Agent malevolo a creare una nuova identità.
Se l'Agent malevolo opera sia come Client che come Agent per incrementare il numero operazioni Cooperative, si troverà di fronte al pagamento di fees ed ogni volta che opera come Client diminuirà la quantità di servizio a disposizione come Agent.
Client Malevoli: Un Client può creare uno state channel e non inviare la prova di possesso della quantità di denaro necessaria a pagare il servizio. Ma al Client è richiesto di depositare quella cifra che sarà bloccata per tutto il periodo del channel. Per ottenere il denaro indietro è necessario chiudere il channel, ma

la transazione consuma Ethere e quindi costa. Quindi se un Client vuole danneggiare un Agent le sue possibilità sono ridotte e costose, inoltre sarà segnata Uncooperative, a suo sfavore, una chiusura di un channel.
Sia i Client che gli Agent ottengono la chiusura del channel in modo Cooperative o Uncooperative e la chiusura viene registrata in blockchain con associati gli indirizzi di entrambi. Il sistema blockchain consente di analizzare le chiusure Cooperative e Uncooperative di tutti così da verificare la reputazione a cascata.
Quindi: Fondamentalmente non hanno trovato un modo per eliminare al 100% attacchi Sybil ma cercano, con i sistemi sopra descritti, di limitare al massimo le possibilità di azione e la loro efficacia.

Sentinel

- Identity Chain: Anonymous User ID (AUID) creati e memorizzati in una blockchain ad hoc.
- Service Chain: Rete sicura su cui si appoggiamo le operazioni di gestione e sono usati i token $SENT/$SST per le transazioni ($SENT sono usati esternamente e sono usati comprare $SST che sono usati internamente, rapporto 1:1)
- Transaction Chain: gestisce i pagamenti ed i relativi processi di invio delle transazioni al Sentinel Transaction Pool.

L'unico punto di accesso ai servizi di Sentinel è l'AUID.
Il sistema è basato sulla reputazione tale per cui più è alta e maggiore è l'accesso ai servizi. C'è un meccanismo di guadagno di denaro in funzione della reputazione.
Il consenso distribuito riconosce e mitiga velocemente cattivi attori.
La soluzione adottata per mitigare il problema di attori cattivi sul nodo di uscita è la stessa adottata dalle reti come TOR, ovvero l'uso di un tecniche di routing e ritrasmissione dei pacchetti per assicurare che sia la sorgente che la destinazione non siano rilevati.
Si sta attualmente sviluppando la rete di ritrasmissione nella quale i partecipanti posso decidere il ruolo (perché?). **Idee di Sentinel** Al contrario delle altre soluzioni loro vogliono creare una piattaforma su cui appoggiare una serie di prodotti e servizi oltre alla dVPN: dChat, dVoIP, dFiles, dCDN, dDNS, dCompute, chiamate dAPPs. Hanno intenzione di realizzare un internet box hardware da posizionare tra l'accesso ad internet (gateway) ed il router.
Attualmente Sentinel si appoggia a Cosmos, piattaforma che consente la creazione di una blockchain decentralizzata di criptovaluta. Cosmos è costruito su Tendermin che fornisce il livello networking e consensus di una blockchain (quindi Cosmos è l'application di cryptocurrencies)
Quindi, per quanto riguarda Sentinel, il problema del consenso BFT è risolto dalla piattaforma su cui si appoggia.

Tendermint Tendermint è un software per la replicazione in modo sicuro e consistente di una applicazione su una moltitudine di macchine. Con in modo sicuro si intende che Tendermint funziona anche se 1/3 delle macchine fallisce arbitrariamente. https://tendermint.com/docs/introduction/what-is-tendermint.html

Substratum caratteristiche comuni BTF Attacchi di tipo...

4 Conclusions

4.1 Tests

4.2 Future development

4.3 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.