Ócetic
Your Connection to ICT Research

Improving Openstack Tacker's Accountability Using Blockchain Technology

Yassine Jebbar

Centre d'Excellence en Technologies de l'Information et de la Communication

July 6, 2017

Outline

Your Connection to ICT Research

Cloud Computing & Virtualization

cetic
Your Connection to ICT Research

- Computing paradigm that relies heavily on outsourcing of (especially) physical resources.

- Abstraction of software from hardware

- Several advantages in combining Cloud Computing and Virtualization (Performance, Scalability, High Availability, Pay per use,...etc)
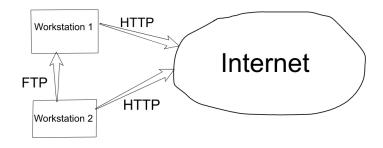
Software Defined Networks

cetic
Your Connection to ICT Research

- Traditional networks are rigid and offer little flexibility.

▶ Make networks more "programmable".

▶ Abstract networking technology/protocols (control plane) from networking hardware (forwarding plane).

Example

Network Functions Virtualization

- Various networking hardware provided by different vendors (vendor lock-in issue).

- Specific hardware for each networking function (Firewall, NAT, DNS,...etc)

► Provide common hadrware platform for the different networking functions, and make it manageable from computing devices (servers, laptops, mobile,...etc)

SDN & NFV Upgrades

▶ Central Management

▶ Direct Programmabiliy

▶ Flexibility

▶ Vendors-Neutral

Service Function Chaining

▶ Improved flexibility in networking functions.

▶ Instead of building a rigid networking system, instances of simple networking functions (NAT, DNS, DPI,...etc) are composed and "chained" together as required to form more complex service functions.

▶ SFC offers low complexity, high maintenance ability and improved adaptability.
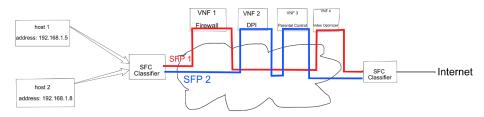
SFC Main Components

▶ **Service Function (SF)** A virtual instance of a basic networking function (Firewall, DPI, NAT, DNS,...etc)

▶ **Service Function Forwarder (SFF)** Logical component that forwards the incoming packets to SFs according to the information carried in the SFC encapsulation.

▶ **Service Function Path (SFP)** The actual path of a packet chosen by the SFF after taking into account the network's constraints (network load, availabilty of SF instances,...etc)

▶ **SFC Metadata** Exchanged context information among the different SFC components.

SFC Components (Example)

- ▶ if source ip == 192.168.1.5 then forward following SFP 1

- ▶ if source ip == 192.168.1.8 then forward following SFP 2

The Blockchain Technology

▶ Introduced to the IT field as the technology behind the Bitcoin cryptoccurency in 2009 by Satoshi Nakamoto.

▶ Blockchain is basically a massive distributed database (i.e: distributed ledger) where transactions between different peers are being recorded.

▶ Blockchain use cases are being expanded to other fields beside cryptocurrencies (Health care, copyrights,...etc)

Why the Blockchain is differrent

cetic
Your Connection to ICT Research

- ▶ Blockchain solves the "double spending" issue in digital transactions.

- ▶ Spares the need for central "trust" authority in transactions.

- ▶ Complex cryptographic computations and a huge network (7692 full node on average in July 2017).

Problem Space Definition

▶ How secure is the SFC architecture ?

▶ How can the Blockchain make a system more secure ?

▶ Is there any use case where the Blockchain can improve SFC's security ?

Problem Space Definition: The 3 A's

Ⓞcetic

Your Connection to ICT Research

- Systems' security is generally evaluated following 3 criteria:

  ▸ Authentication

  ▸ Authorization

  ▸ Accountability

Openstack Tacker

Ccetic
Your Connection to ICT Research

- ▶ New component (v 0.7 currently) in the IaaS provider Openstack.

- ▶ Tacker is released as the platform's main component for managing VNFs instances in a virtualized environment.

- ▶ Tacker relies on Openstack's other components (Keystone, Heat,...etc) to orchestrate and provide the required infrastructure for the VNFs (see NFV MANO framework).

Problem Space Considerations

- Tacker still facing major issues in its setup as well as providing its use cases.

- The availabe NFV platforms still don't adhere completely to the NFV MANO framework.

  - Problem space adaptation considering the available tools and Blockchain's strength: Improving Openstack Tacker's accountability.

Logs Immutabilization

Ocetic
Your Connection to ICT Research

- ▶ To make Tacker more "auditable", we ensure the log files authenticity in terms of content.

- ▶ We rely on Blockchain's strength in keeping records to store a proof about the logs' content.

- ▶ Log files must be secured before storing their proof of authenticity in the Blockchain.

Logs Immutabilization Mechanisms (1) cetic

Your Connection to ICT Research

- ▶ We use logs immutabilization mechanisms used in a previous research work (Distributed immutabilization of secure logs (Jordi Cucurull and Jordi Puiggal))

- ▶ The secure logs generated will have 2 types of entries: Regular entries and Checkpoint entries.

- ▶ This approach allows us to detect where the alteration of content has occured (see equations).

Logs Immutabilization Mechanisms (2)

Your Connection to ICT Research

- Regular entries are generated as the following:

$$L_i = (LogInfo_i, h_i) \quad \text{where} \quad h_i = HMAC(K_j, (h_{i-1}|LogInfo_i)) \quad (1)$$

- Meanwhile, a Checkpoint entry is generated after an arbitrary number of lines (50 lines in our implementation), in accordance with the equation:

$$Chk_j = L_i = (LogInfo_i, K_{j-1}, E(P_{enc}, K_j), Sig_j, h_{i-1}, h_i) \quad \text{where} \quad (2)$$

$$h_i = HMAC(K_b, (h_{i-1}|K_{j-1}|LogInfo_i))$$

$$Sig_j = S(S_{sig}, (h_{i-1}|K_{j-1}|E(P_{enc}, K_j)|h_i|LogInfo_i)$$

Benchmarking of Blockchain-based Solutions (1)

- **Blockstack**
  - ▶ Naming system built on Bitcoin's Blockchain.
  - ▶ Allows tracking key-value pairs in form of (unique) names and their associated data.
  - ▶ Relies on Blockchain "as a communication channel for announcing state changes".
  - ▶ Uses a dedicated data plane for storing the name-data pairs records.

- **Proof of Existence**
  - ▶ An online service for storing and timestamping the existence of a document inside the Blockchain.
  - ▶ Keeps a hash of the file in the chain, with a referrence also to the time in which the document was stored.
  - ▶ Offers the advantage of validating the documents existence even if the online service is down (built on Bitcoin's Blockchain).
  - ▶ Uses the OP_RETURN field to pass information in Bitcoin's transactions.

Benchmarking of Blockchain-based Solutions (2) cetic
Your Connection to ICT Research

- **Hyperledger**
  - ▶ Smart contract blockchain platform.
  - ▶ A set of distributed ledger solutions based on different platforms and programming languages.
  - ▶ Based on the expectation that there will be many blockchain networks, with each network ledger serving a different goal.
  - ▶ Offers various services (Blockchain Explorer, Fabric,...etc).

- **Namecoin**
  - ▶ Another Blockchain-based solution for namespaces storage.
  - ▶ Hard fork of Bitcoin's Blockchain.
  - ▶ Allows secure storage of key value pairs (same value can exist within different namespaces though).

Bitcoin's OP_RETURN
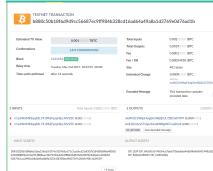
- ▶ Bitcoin's increasing popularity ==> Use cases outside finance ==> Need to send data within transactions.

- ▶ Developers tried different ways to work around this issue (adding values in outputs, manipulating Bitcoin's scripts,...etc).

- ▶ Large members of the Bitcoin's community did not like the bits of data added in transactions, especially as it affected (negatively) the mining process.

- ▶ An opcode (operation code that pushes data) called OP_RETURN was consequently standarized by Bitcoin and made available to developers in order to include up to 80 bytes of data in transactions.

OP_RETURN Transaction (Example) (1)

OP_RETURN Transaction (Example) (2)

Hex to ASCII text converter

Hex to ASCII text converter.

Enter 2 digits hex numbers with any prefix / postfix / delimiter and press the *Convert* button (e.g. FF 43 5A 7F):

48656c6c6f2c20746573746e657421

⟳ Convert   ✖ Reset   ↩ Swap

Hello, testnet!
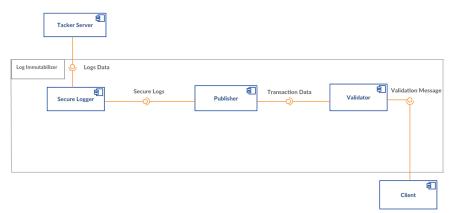
6a0f48656c6c6f2c20746573746e657421
Hello, testnet!

Analysis

Ccetic
Your Connection to ICT Research

- ▶ In order to implement the logs immutablizer, we rely on a component-oriented system.

- ▶ Beside the components that will secure Tacker's log files (see previous equations), additionnal components are needed to store the authenticity proofs in the Blockchain.

- ▶ A final process of Checkpoints validation needs also to be taken into account.
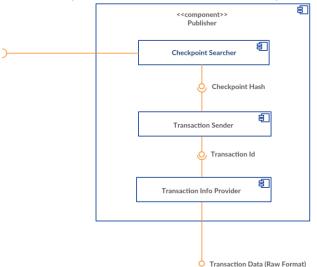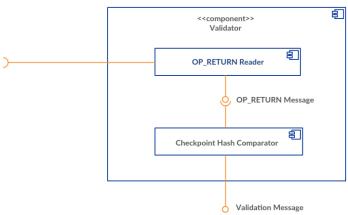
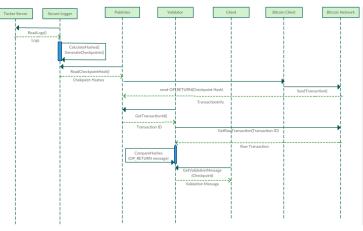# Conception: Components Diagram

Conception: The Publisher Component

Conception: The Validator Component

Conception: The Sequence Diagram

Technologies (1)

- **OP_RETURN python library**
  - Interacts directly with Bitcoin's Blockchain.
  - Include the message taken as arguemnt through CLI command in the OP_RETURN opcode.
  - Allows 3 operations: send, store and retrieve.

- **Bitcoin Core**
  - Open source project that maintains and publishes the Bitcoin client software.
  - Besides validating transactions "locally" (160 Go need to be downloaded though !!), Bitcoin Core acts also as a Bitcoin Wallet.
  - Unlike Bitcoin's online clients, Bitcoin Core offers improved security.

Technologies (2)

Ocetic
Your Connection to ICT Research

- **RabbitMQ Messaging**
  - ▶ A message broker typically used for building integration between applications (or components) using messages.
  - ▶ Relies on the producer/consumer communication via messaging queues.
  - ▶ Supports local communication as well as remote one.

Components' Implementation: The Secure Logger

- ▶ The component acts on text log files as the source of the incoming data.

- ▶ A sample log text file generated by Tacker server was stored, then secured as per the previous mechanisms.

- ▶ A Checkpoint entry is generated after evert 50 lines.

- ▶ We used the HMAC python module to calculate the hashes.

The Secure Log File

```
cal/lib/python2.7/dist-packages/oslo_config/cfg.py:2770▯▯[00m, this line hash is: 30b5429ad1e28f11e51cd22de4de42a7b7b
cfg.py:2771▯[00m, this line hash is: 7e189811cbdc595587c2cccca54dd0be5df6471fa
om (pid=18738) log_opt_values /usr/local/lib/python2.7/dist-packages/oslo_config/cfg.py:2772▯▯[00m, this line hash is
es/oslo_config/cfg.py:2774▯▯[00m, this line hash is: c2654b229f4ddb816df3716feda020af50273f59
cal/lib/python2.7/dist-packages/oslo_config/cfg.py:2775▯▯[00m, this line hash is: 826180eb1b81387e7da1066554dfee5948a
/cfg.py:2784▯▯[00m, this line hash is: a82a3168da851e8f63620f67d5eb074df6cb3a7a
g/cfg.py:2784▯▯[00m, this line hash is: c9ae9856a252790170754a7caec9375864da9e04
g/cfg.py:2784▯▯[00m, this line hash is: 1b6cd1b150c389eebcbfe7d741256f9380de7b0e
.py:2784▯▯[00m, this line hash is: 648db3d0e547e5efe7ba8c1666fc905711d55aa1
lo_config/cfg.py:2784▯▯[00m, this line hash is: 0c57fc3f3ef821edf7f8d69d90e4e4fd848150bf
g.py:2784▯▯[00m, this line hash is: a7f022ea674d702ac1ba9515f9742c5a361a5903
nfig/cfg.py:2784▯▯[00m, this line hash is: 26257e24c3277d5d3be8c1d38c72fd9e7ff4e7ff
/cfg.py:2784▯▯[00m, this line hash is: b2fb2de720aff4b063a4db938f0c4d2b69d04635
fig/cfg.py:2784▯▯[00m, this line hash is: 10b5600835f5bce4dd8b90ea53e0c7898153cfa6
/cfg.py:2784▯▯[00m, this line hash is: 7fefa67d15afee7111ac33b9a97a2e067fb339c1
fg.py:2784▯▯[00m, this line hash is: adcc9a05bafbbdea78dc22d9f276bb6559ec0221
thon2.7/dist-packages/oslo_config/cfg.py:2784▯▯[00m, this line hash is: fcdbf45d00a93b9792f25db5c7388b370abbf53c
ig/cfg.py:2784▯▯[00m, this line hash is: 8c4b844ffebe5f5030cee891db7cb3730782e791
```

Components' Implementation: The Hash Reader

- ▶ Reads the Checkpoint entries from the secure log file.

- ▶ Extract the checkpoints' hashes and store them in a separate file.

- ▶ The Checkpoint hashes file is provided for the Publisher component.

The Checkpoints' Hashes

Components' Implementation: The Publisher

- Issues an OP_RETURN Bitcoin transaction containing the Checkpoint hash.

- Due to Bitcoin Core's scalability issues (see report), an OP_RETURN transaction is sent once per hour.

- Component implemented in Bash (rather than python) due to the OP_RETURN library's inconsistency.

- After each transaction, the transaction id is stored and sent to the next component.

Bitcoin's Transactions

✓ 2017/6/19 14:58	Sent to	
✓ 2017/6/19 13:58	Sent to	
✓ 2017/6/19 13:54	Sent to	
✓ 2017/6/19 13:19	Sent to	
✓ 2017/6/19 12:58	Sent to	

cetic
Your Connection to ICT Research

Components' Implementation: The Validator

▶ Contains 4 subcomponents that communicate using RabbitMQ.

▶ JSON objects are passed through to send the recent transaction id as well as the transaction's hexadecimal format.

▶ The OP_RETURN message is identified using the prefix '6a20'.

▶ After reading the hash from the transaction, a comparison is made with the stored Checkpoint hashes, and a validation message is sent eventually to the client component.

Checkpoint Validation Message

Design Decisions

Ocetic
Your Connection to ICT Research

- A Checkpoint was issued after every 50 lines (arbitrary choice), given the log file contained circa 500 lines.

- Bitcoin Core was used in the testnet mode, so that tBTCs could be obtained for free.

- Each transaction was sent with a 0.1 tBTC fee, which limited the transaction validation time in almost 50 minutes.

Perspective & Future Work

▶ Packaging of the log immutabilzation application (Docker, Kubernetes,...etc).

▶ Working on the feasibility of the remaining A's (Keystone for Authentication? , Permissioned Blockchains for Authorization?).

▶ More tests on SFC platforms (Port Chaining in Neutron?)

Thank You For Your Attention