EMAIL VERIFICATION SERVICE USING BLOCKCHAIN

Arun Varghese

Follow this and additional works at: https://www.tdcommons.org/dpubs_series

Recommended Citation
Varghese, Arun, "EMAIL VERIFICATION SERVICE USING BLOCKCHAIN", Technical Disclosure Commons, (September 10, 2019)
https://www.tdcommons.org/dpubs_series/2468

This work is licensed under a Creative Commons Attribution 4.0 License.
This Article is brought to you for free and open access by Technical Disclosure Commons. It has been accepted for inclusion in Defensive Publications Series by an authorized administrator of Technical Disclosure Commons.
EMAIL VERIFICATION SERVICE USING BLOCKCHAIN

AUTHOR:

Arun Varghese

ABSTRACT

Current email security solutions depend on various attributes to reduce the chances that a given email (mail) is likely to be a threat. However, current solutions make it relatively easy to target corporate organizations with a Business Email Compromise (BEC) attack. A BEC attack is a non-malicious mail which defrauds key people in organizations into performing, for example, wire transfers meant for the suppliers or partners abroad.

The U.S. Federal Bureau of Investigation (FBI) has been tracking BEC, also known as email fraud and email account compromise (EAC), domestically and globally since October 2013. The recent trends related to fraudulent wire transfers and unauthorized disclosures of employee data are alarming:

- Total identified global exposed losses now exceed $12.5 billion (up from $5.3 billion in December 2016).
- More than 30,000 victim complaints were submitted between June 2016 and May 2018 via the recently launched Internet Crime Complaint Center (IC3) compliant form.
- BEC scams targeting the real estate sector rose more than 1,100% between 2015 and 2017.
- Wage and tax documentation BEC scams extend the threat beyond wire transfers and continue to grow. The US Internal Revenue Service (IRS) indicated it received approximately 900 reports of Form W-2 scams in 2017 (compared to just over 100 reports in 2016).

The problem is that there is no absolute way to understand if a mail was sent from a particular sender to a group of recipients.
DETAILED DESCRIPTION

Current email security focus on classifying emails (mail) as malicious or spam. Conventional security applications/services tend to focus on the reputation of the Internet Protocol (IP) address, domain, or Uniform Resource Locator (URL) contained in the mail and determine whether any of these parameters are known to be linked to a malicious/phishing attack. A problem with such arrangements is that a BEC attack almost always comes from a newly registered domain and is targeted to a specific group of people in an organization. As such, by the time the mail is understood, the attack has already taken place.

There are Domain Name System (DNS) based features such as Sender Policy Framework (SPF), DomainKeys Identified Mail (DKIM), and Domain-based Message Authentication, Reporting and Conformance (DMARC) that can verify the mail authenticity, but these also each have limitations. Limitations that need to be considered when dealing with SPF, DKIM, and/or DMARC include:

- DKIM alone does not grant, in any way, that the sender server is allowed to send outgoing mail for the specific domain.
- SPF is powerless with messages forged in shared hosting scenarios, since all of the mail will appear as the same coming IP address.
- DMARC is still in early stage and has not been widely utilized.
- DMARC can (and will) break mail flow if the system is not set up with both SPF and DKIM before changing DMARC policy to anything above “none.”

As a result of the shortcomings in conventional arrangements, presented herein is a blockchain based ledger service which marks down an outgoing mail. A public key can be generated for each mail address via, for example, elliptic curve multiplication. Additionally, a compressed bitcoin addresses may be generated.

More specifically, a bitcoin address of all the employees sending mail out will be computed, stored in the server, and retrieved at the time of sending a mail (which is long 34 string which is public key). Depending on the number of recipients, the transaction has to be signed by the senders private key (64 string) to authorize the transaction. Moreover, each piece of transaction data contains the hash value of the previous transaction, plus the
data of the current transaction. A group of transactions are added into a block and the hash
of the block is computed, where the blocks are timestamped and the hash value is added
with a nonce (a random integer between 0 and 4,294,967,296) and calculated in such a way
that the hash begins with a dynamic number of zeros bits. Once this is stored in the
blockchain, a transaction identifier (ID) is generated, which is sent along with the mail
header. If the mail header has this particular transaction ID, the system queries the
blockchain server with the transaction ID and, if the transaction ID is found in the server,
the mail sender is verified.

The advantages of this approach include:

1. No separate setup is required (since it is a decentralized service).
2. Since this is a blockchain based service, it is difficult to hack the data (100%
safe regards to privacy).
3. Decentralized since esa will act as the miners.

Conventional techniques do not offer blockchain based verification as a service.
With the proposed solution, email details (e.g., sender, recipients, timestamp, etc.) are
hashed and saved from any organization using email security gateways. As such, there is
no privacy issues. Any organization receiving the mail can check it for authenticity via the
blockchain service, thus the mail cannot be spoofed.

FIG. 1, below, is a flowchart illustrating the email verification service using
blockchain from the sender side. In FIG. 1:

1. Email gateways subscribe for the blockchain service.
2. Once a mail is sent, the details of the mail (e.g., sender, recipients, hash of the
mail, etc.) will be sent to a blockchain service.
3. Group of blockchain server nodes completes a challenge with the given details
and data is written into block once a node completes the challenge.
4. Once the challenge is completed, and block is written, the transaction ID is
sent back to the email gateway as the response.
5. The transaction ID is added as a header field into the mail and the mail is sent
to the intended recipients.
FIG. 2, below, is a flowchart illustrating the email verification service using blockchain from the receiver side. In FIG. 2:

1. Email gateway which has the service subscribed receives the mail (if not subscribed, then the mail goes through the normal flow).

2. The blockchain transaction ID is parsed from the header fields and is queried back to the blockchain server API service, which returns the stored details. These details, such as list of mail IDs, sender and hash value, can be cross checked with the receiving mail.

3. If the details match, the system can be sure that the mail is not spoofed.

4. Since there is no infrastructure dependency on the customer side, the customer business proceeds as usual for the administrators, but has an extra layer of security for the mail.

FIG. 3A, below, illustrates a transaction lookup for a non-authenticated client (machines other than provisioned devices), while FIG. 3B, below, illustrates a transaction lookup for an authenticated client (assuring privacy in plain sight).

In the techniques presented herein, the storing of the details in blockchain is performed by a group of deployed server nodes which obtain the details and compete among the nodes to complete the challenge (e.g., similar to how blockchain server nodes operate and get rewarded with bitcoins for completing a block). Email verification service using blockchain provides privacy on the data stored, while still protecting organizations from business email compromises.
**Transaction Look Up [Non Authenticated]**

View information about a email transaction

![Image](image_url)

**Summary**
- Size: 404 (bytes)
- Weight: 1616
- Received Time: 2019-06-13 15:50:13
- Visualize: [View Tree Chart](#)

**Inputs and Outputs**
- Total Input: 0.00810546 BTC
- Total Output: 0.00775434 BTC
- Fees: 0.00035112 BTC
- Fee per byte: 86.911 sat/B
- Fee per weight unit: 21.728 sat/WU
- Estimated BTC Transacted: 0 BTC
- Scripts: [Show scripts & coinbase](#)

**FIG. 3A**

---

**Transaction Look Up [Authenticated over SSO]**

![Image](image_url)

**Summary**
- Size: 404 (bytes)
- Weight: 1616
- Received Time: 2019-06-13 15:50:13
- Visualize: [View Tree Chart](#)

**Inputs and Outputs**
- Total Input: 0.00810546 BTC
- Total Output: 0.00775434 BTC
- Fees: 0.00035112 BTC
- Fee per byte: 86.911 sat/B
- Fee per weight unit: 21.728 sat/WU
- Estimated BTC Transacted: 0 BTC
- Scripts: [Show scripts & coinbase](#)

**FIG. 3B**