Trustless 2 Factor Authentication for OpenSSH server leveraging Blockchain and Ethereum Smart Contracts

Varun Amrutiya (2017H1120244P), Siddhant Jhamb (2017H1120243P), Pranjal Priyadarshi(2017H1120242P)

Motivation

- 2FA is widely used in Banking, Email and with web service with high security.
- 2FA Provides an additional layer of security along with passwords.
- OpenSSH server is mostly widely used SSH server remote login utility with great features.
- Managing a large server infrastructure is very challenging even with efficient key distribution design which comes with hefty cost of third-party products.
- Here we present a new solution for server authentication using Ethereum Blockchain

Features

- Decentralized design
- Trustless smart contracts
- Peer-to-Peer connectivity (no single point of failure)
- Uses the blockchain provided application module rather than programming out-of the box solution.
- No need to integrate with auto-generated mailing system or SMS gateways.
- Completely free.
- Free from data storage responsibility (thnx to blockchain...)

Major Implementation Components

- Ethereum Smart Contract (For application Logic)
- DApp (Decentralized App) (UI for interfacing with blockchain)
- Metamask Plugin (Browser plugin based credential store)
- Pluggable Authentication Module (Extensible linux framework for extending newer authentication schemes).

Tools Used

OS: Ubuntu

Remix-IDE for contract testing and deployment.

Ganache: A ethereum blockchain for testing and development.

MetaMask: A browser plugin for key storage.

Python-Flask: A simple web server to serve the DApp

Use Cases

The main users of this toolset are,

1) Staff : A person who would be incharge of maintaining server infrastructure.
2) User : A person who would use the system.

Staff Functionalities

- Register System: To register name and ethereum address of the system.
- Register User: To register name and ethereum address of the servers.
- Current Request: To view and approve requests made by users for access of the system.
- System Status: To see a complete list of registered user, registered system and also see list of revoked system and unlock them if required. This provides a birds-eye view of whole infrastructure.

User Functionalities

- Request System Access: To get a permanent access of any system as part of this.
- My Systems: To show all the systems allocated to the user by the staff.
- Generate OTP: To generate an random 6 digit pin and store it SHA3 form in the blockchain for later use.

Workflow (User/Staff)

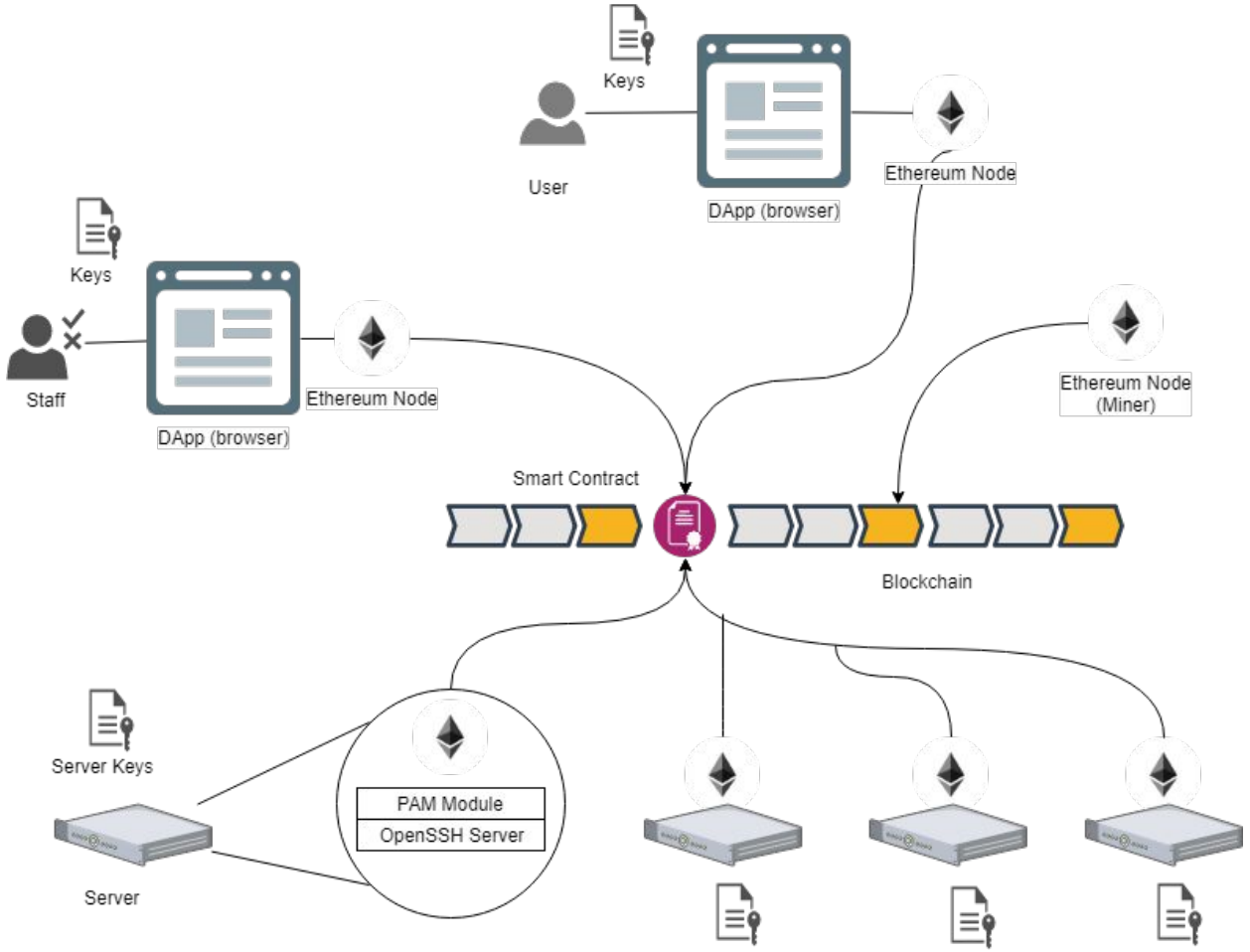
1. Staff register an system which is part of this infrastructure
2. Staff also register the set of users who can communicate with the smart contract
3. User sees all the system available, and places the request for system which he wishes to use.
4. Staff sees all the requests made by the user and accepts them.
5. User can now generate an OTP and save it in blockchain for the approved system.

(User would be required to configure the metamask plugin before hand with his private keys and Ethereum endpoints IP address and Port.)

Workflow (PAM)

1. User initiates and SSH connection to the system which he wishes to use.
2. Initially the SSH will authenticate the user using normal username and password as a part of normal ssh authentication.
3. PAM module will check in the blockchain whether the system is revoked or not. If revoked it won't allow any access to the user.
4. If the System is not revoked it will provide an prompt to the user to enter his OTP for this access.It will retrieve all stored token for particular system at this point for future use.
5. If OTP is correct it will be deleted from the blockchain so that it cannot be reused.
6. If OTP is incorrect appropriate message will be show and on 3 incorrect OTP entries the system would be marked as revoked in the blockchain and won't allow any further ssh access until staff de-revokes it.

(PAM module would pre-configured with contract address and System ethereum address at compile time.)


Keys
User
DApp (browser)
Ethereum Node
Keys
Staff
DApp (browser)
Ethereum Node
Ethereum Node
(Miner)
Smart Contract
Blockchain
Server Keys
PAM Module
OpenSSH Server
Server

Thanks