Blockchain Based Zero Trust Modeling and Quantification for 5G Networks

Safwan Elmadani, Salim Hariri, PhD., and Sicong Shao, PhD.

Agility - Building Enablers for Multi Industry Sectors Collaborative Federated Open Platforms & Assets, as a Foundation for Cross-Industry End-to-End Services Innovation and Delivery Agility in 5G and Beyond

Presentation Outline

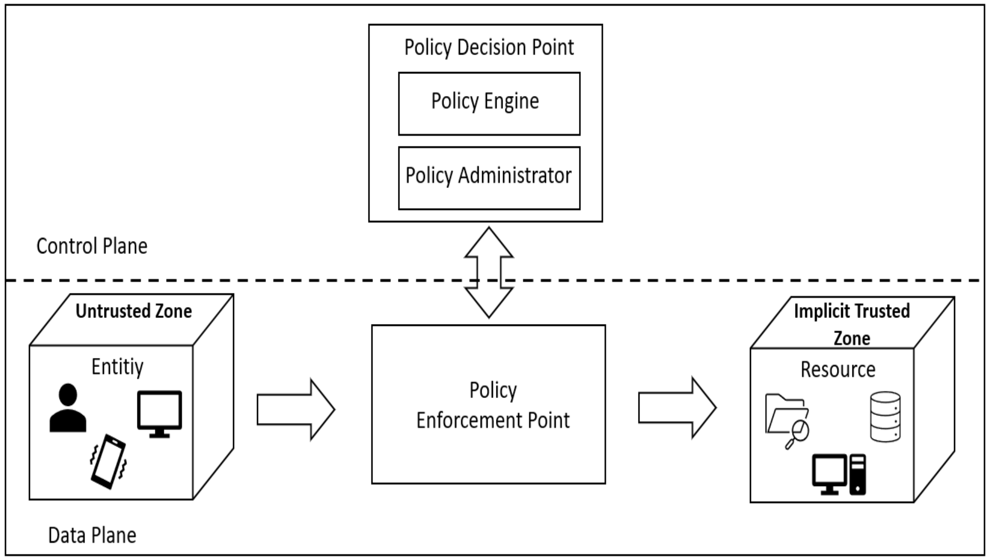
- Zero Trust Motivation and Definition
- NIST Zero Trust Architecture (ZTA)
- Trust Modeling and Evaluation
- Deploying ZTA in 5G Networks
- Implementation Tasks
- Future Work

Zero Trust Motivation

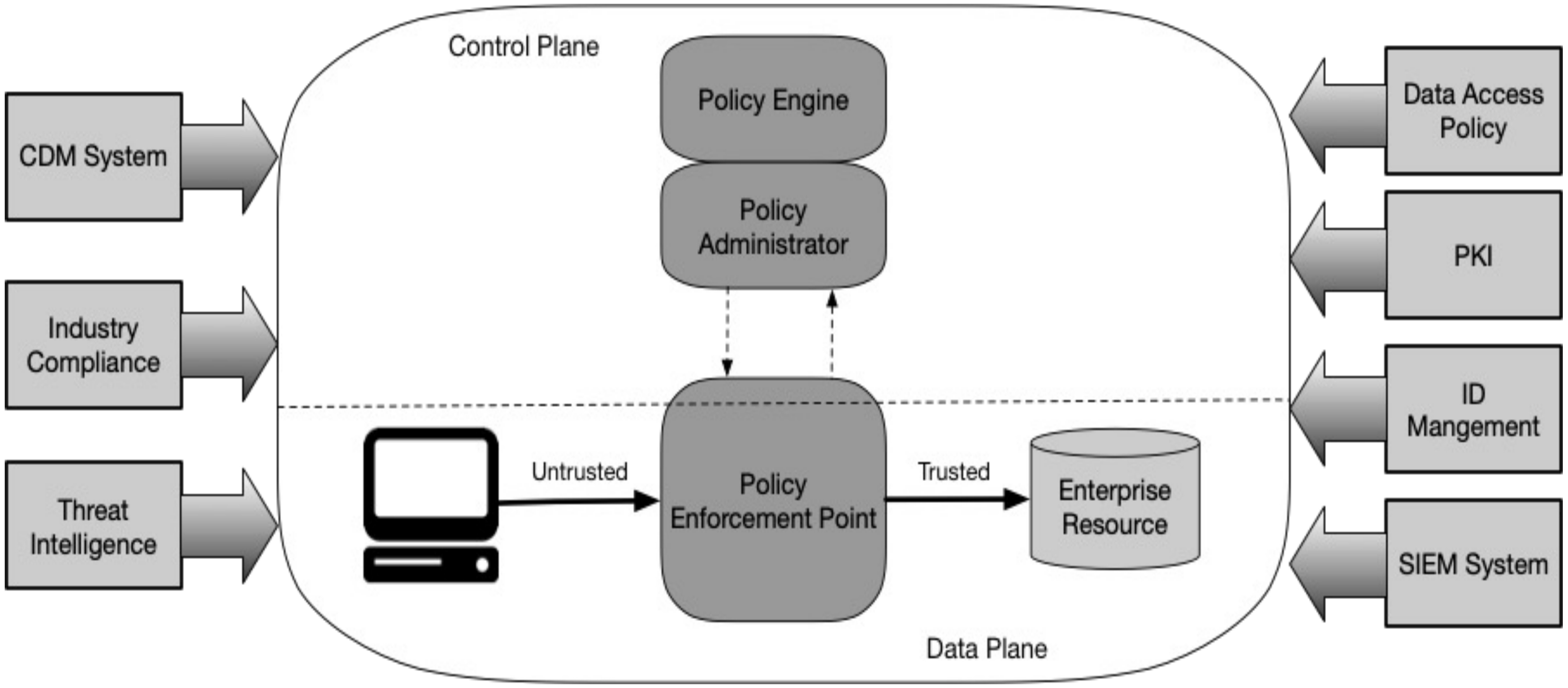
- A single enterprise may operate several internal networks, remote offices with their own local infrastructure, remote and/or mobile individuals, and cloud services.
- This complexity has outstripped traditional methods of perimeter-based network security as there is no single, easily identified perimeter for the enterprise.
- A Zero Trust Architecture (ZTA) strategy is one where there is no implicit trust granted to systems based on their physical or network location (i.e., local area networks vs. the Internet).
- ZTA focuses on protecting data and resources, not the network segments, as the location is no longer seen as the prime component to the security posture of the resource.

NIST Zero Trust Architecture (ZTA)

- Zero-trust architecture (ZTA) described in NIST SP-800-207, introduced in 2014.
- ZTA uses two modules: Policy Decision Point (PDP) and Policy Enforcement Point (PEP)
- **Policy Decision Point (PDP).** The PDP, comprising the policy engine (PE) and the policy administrator (PA). PE grant access to a resource for a given client or subject. PA responsible for issuing credentials/tokens to grant access to a resource, under decision issued by the PE.
- **The Policy Enforcement Point (PEP).** It is responsible for enabling, monitoring and terminating connections between a subject and objects, according to instructions received from the PA, which is in the control plane.



ZTA Logical Components



Cloud and Autonomic Computing Center

Autonomic Computing Lab

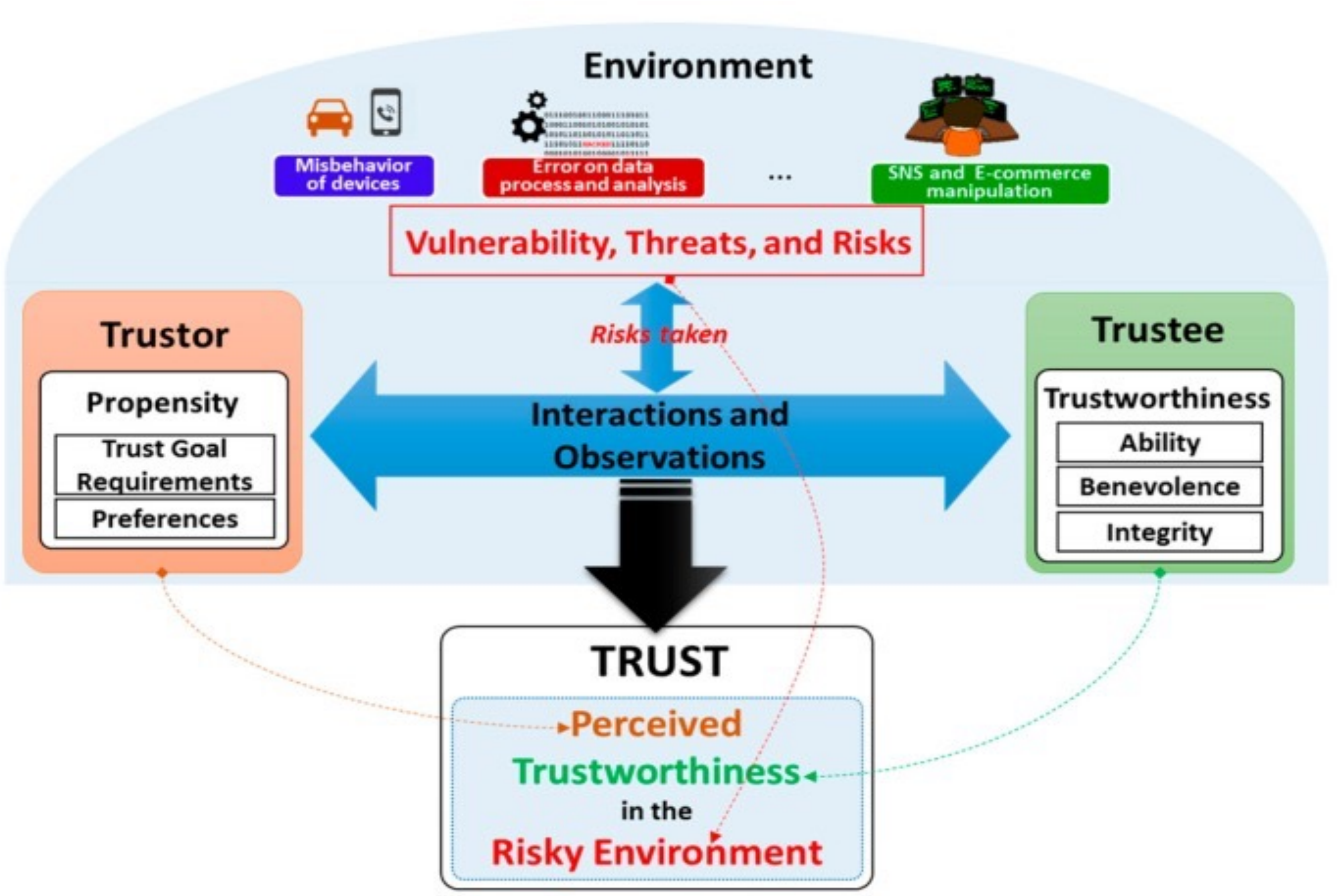
Trust Definition

- A general definition of trust has been broadly used:
  - Trust is defined as a *belief* of a *trustor* in a *trustee* that the trustee will provide or accomplish a *trust goal as trustor's expectation* within a *specific context* for a *specific period of time*.
- In computer science, trust can be defined as:
  - A system is trustworthy if it is secure and not compromised, meaning that it identifies people accessing the system and only allows authorized users
  - Data is trustworthy if the data access control ensures that data is only accessed by those authorized users even in the presence of adversary

Trust Modeling and Evaluation

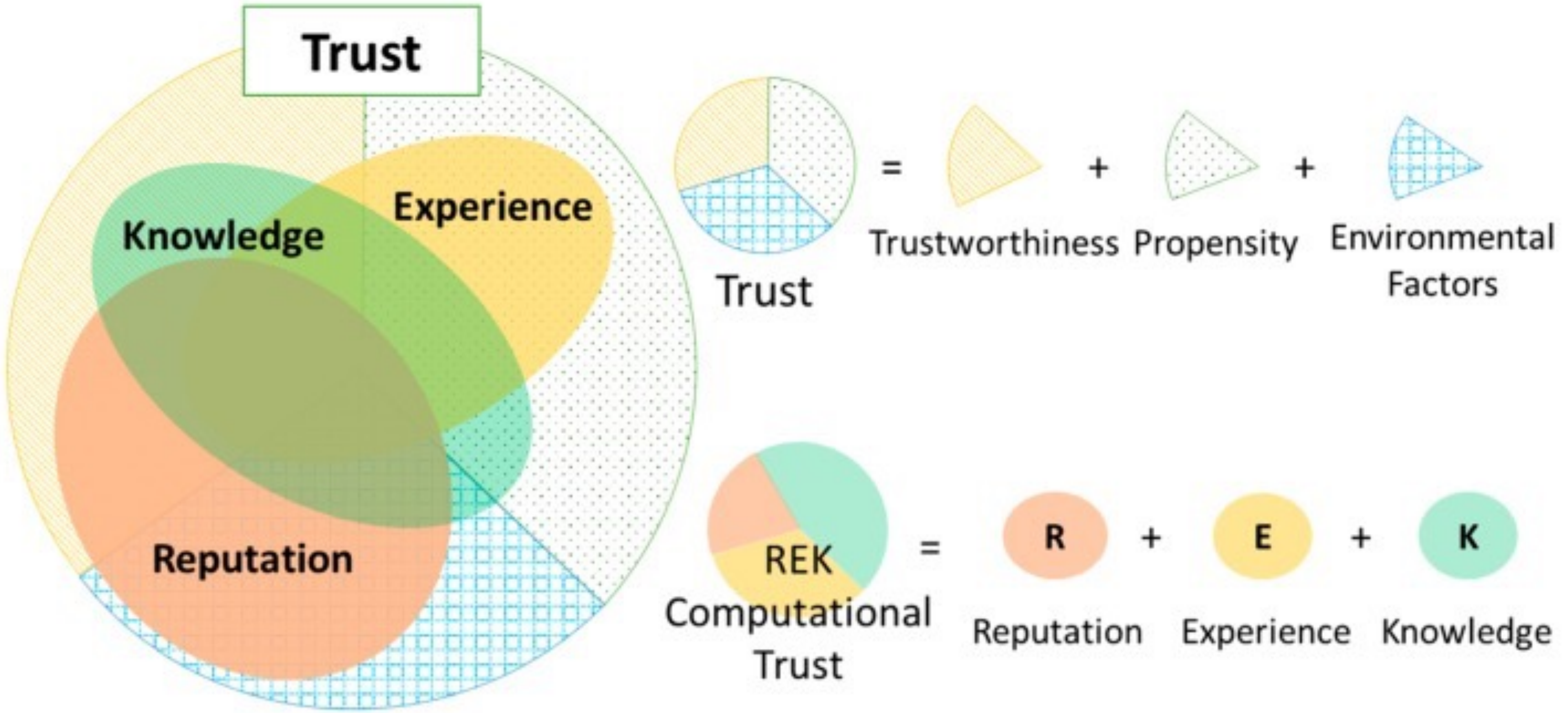
Reference: Toward a Trust Evaluation Mechanism in the Social Internet of Things, Nguyen Binh Truong, Hyunwoo Lee, Bob Askwith, Gyu Myoung Lee, Sensors (Basel) 2017 Jun; 17(6): 1346. Published online 2017, doi: 10.3390/s17061346, PMCID: PMC5492667

Trust Conceptual Model



Trust Indications (TIs)

- Instead of measuring trust using only the direct trust approach, a prospective approach is to use Trust Indicators (TIs) that are feasible, not so complicated to obtain, and cover different aspects of trust.



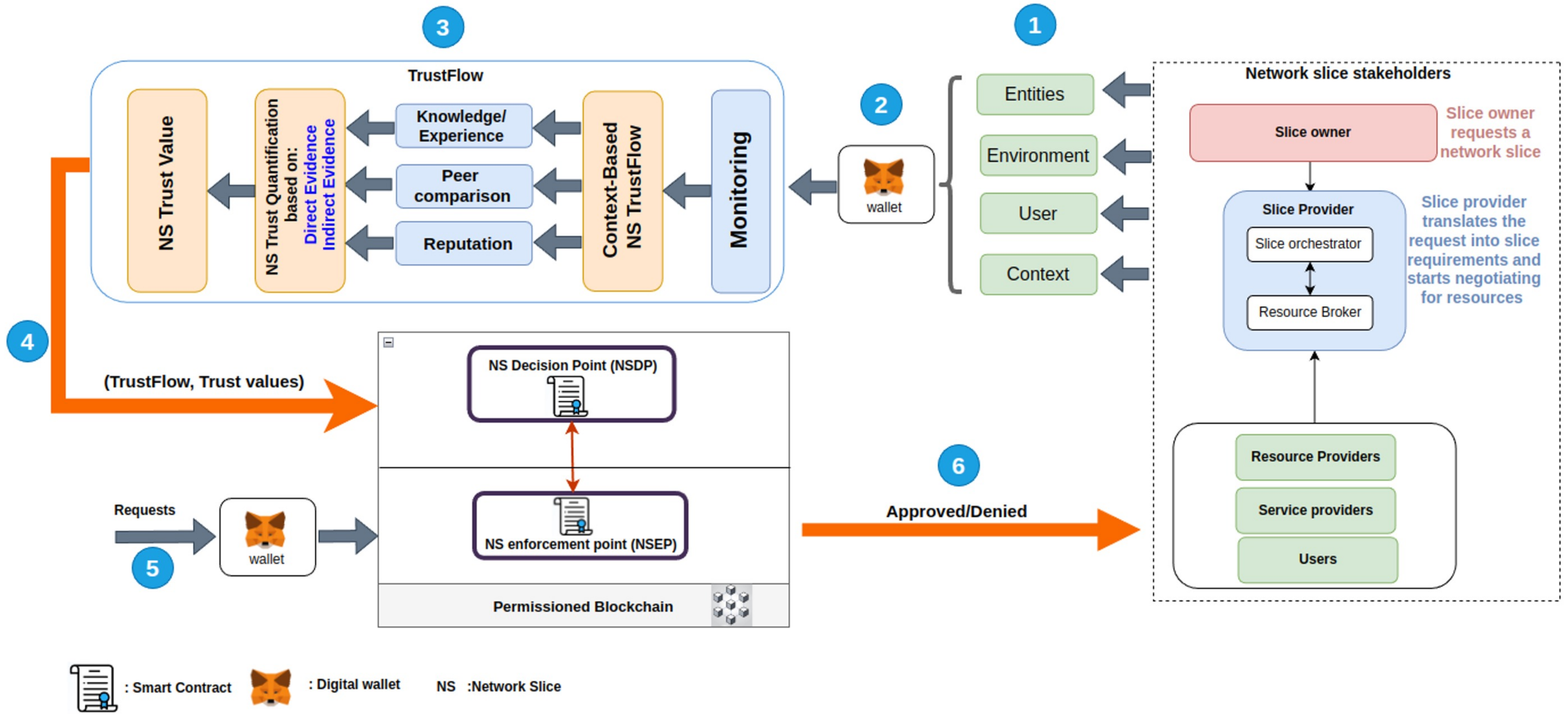
Zero-Trust Motivation in 5G Networks

- In 5G networks, a network slice is defined as a logical network created by partitioning a shared physical infrastructure.
- Each slice is customized and optimized to meet customers' needs.
- Network slicing brings unprecedented security challenges because of its dynamic and diverse structure.
- Trust in the 5G ecosystem is a cornerstone for global adaptation and tackling security and privacy risks.
- In this research, we shed light on the Zero Trust concept in 5G using distributed ledger (Blockchain).
- We will be using blockchain technology to establish trust between network slice stakeholders (i.e., slice owners, users, slice resource providers, and service providers).

Our Trust Evaluation Approach

- It will be based on two factors:
  - Trust based on Direct Evidence (Experience & Knowledge)
    - Using the measurements as Trust Indicators (TIs) to decide if observed behavior is trustworthy; by ensuring that these TIs do not violate any security policies or goals, etc.
  - Trust based on Indirect Evidence
    - reputation. This will compare the behavior of one user with all peer users that have similar role and use the same type of platform (e.g., all peer users of Mac computers).

Zero Trust Modeling and Quantification in 5G Network Slicing - Architecture

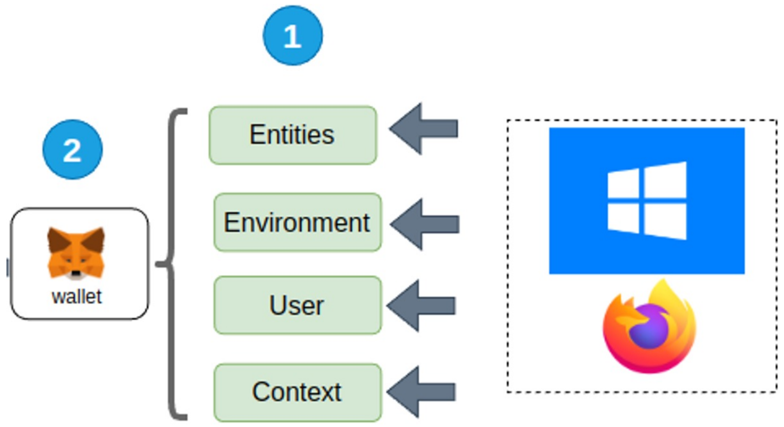


Development Plan and Implementation

- Implementing a data collection tool to gather relevant data from the network slice/system.
- Develop the TrustFlow module that processes real-time date and quantifies the trust of an entity using:
  - Deterministic-based Quantification.
  - Machine Learning-based Quantification.
- Implement the Zero Trust Architecture using blockchain technology with two smart contracts.

Implementing Data Collection Tool

- The data collection tool will be used to collect different types of information 24/7, such as:
  - System data.
  - Network data.
  - Application data.
- This data will be fed to the TrustFlow module to quantify the trust.



TrustFlow Module and Trust Quantification

1. Trust based on Direct Evidence (T_de)

   By continuously monitoring and analyzing the behavior of any component in the environment.

   a. Deterministic-based Quantification
   b. Machine Learning-based Quantification

1. Trust based on Indirect Evidence (T_ie)

   a. Peer Comparison:

      Comparing the behavior of an entity with its peers that

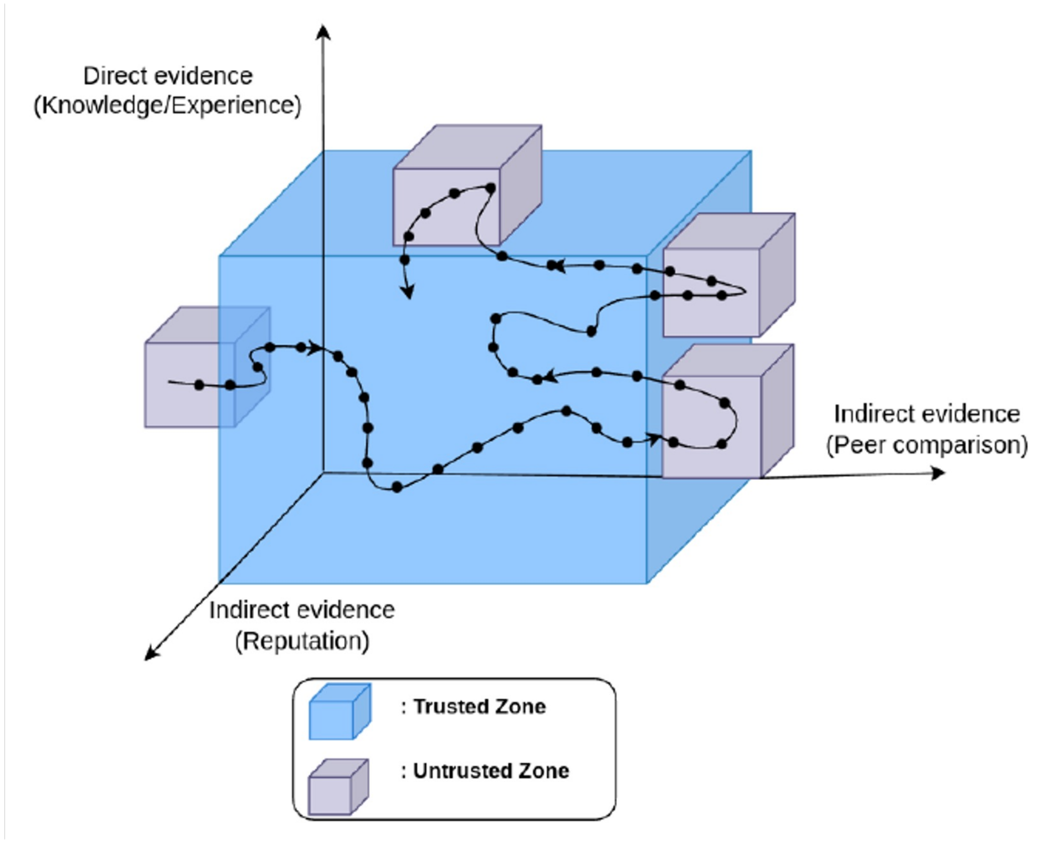
      have a similar role.

   a. Peer Reputation:

      Obtain trust from other entities that are currently interacting with

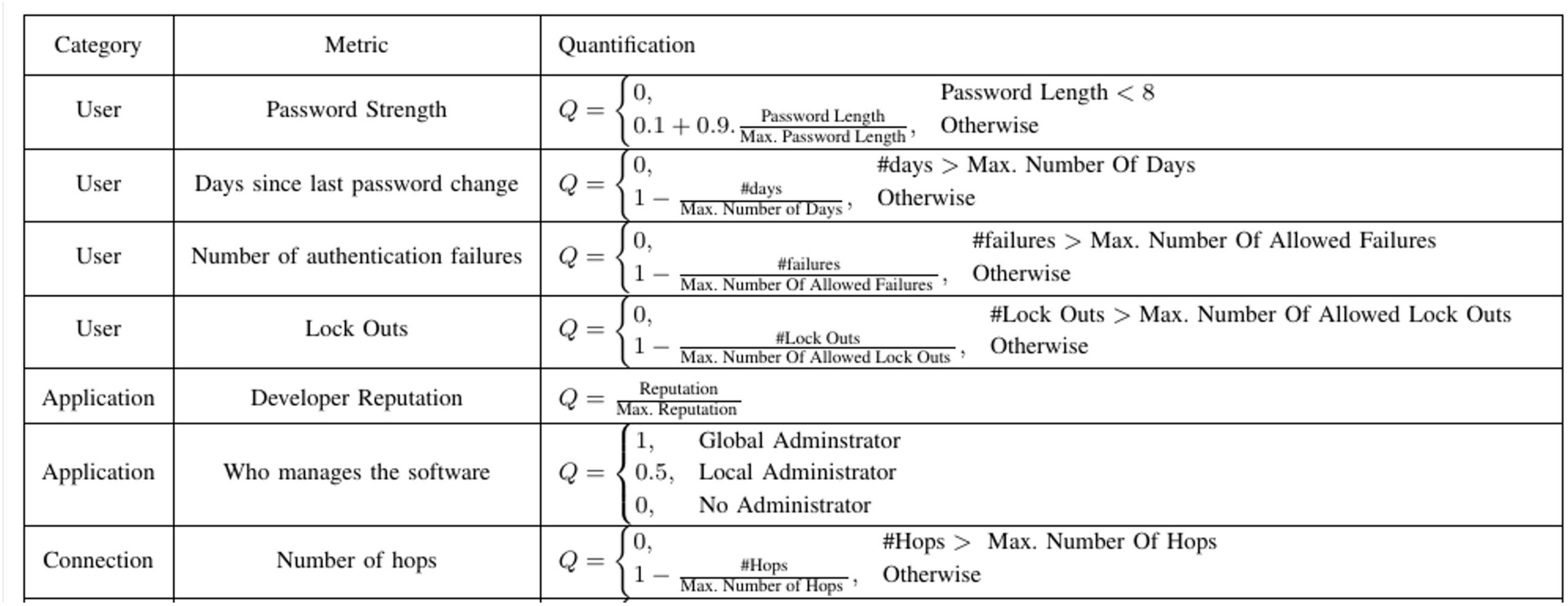
      that entity or have interacted with it in the past.

TrustFlow Module – Conceptual Model



Trust Quantification – Deterministic Approach

- We use deterministic features that can be easily measured and can be used as Trust Indicator (TI) of an entity



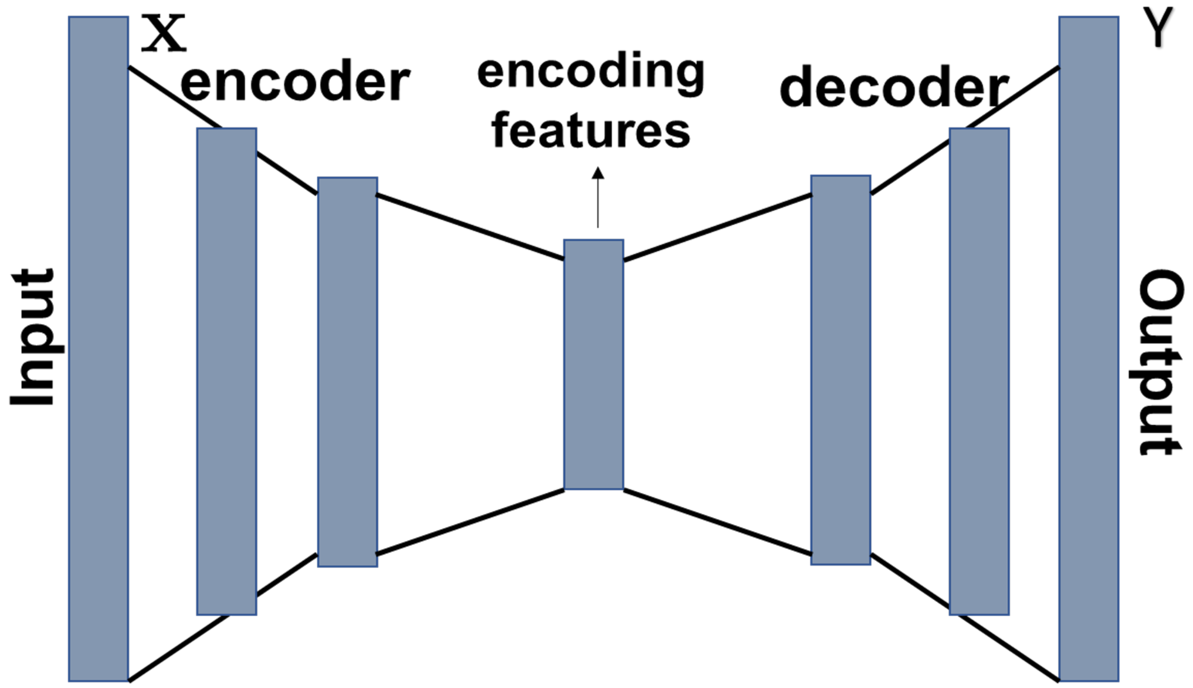
Category	Metric	Quantification
User	Password Strength	$Q = \begin{cases} 0, & \text{Password Length} < 8 \\ 0.1 + 0.9.\frac{\text{Password Length}}{\text{Max. Password Length}}, & \text{Otherwise} \end{cases}$
User	Days since last password change	$Q = \begin{cases} 0, & \text{\#days} > \text{Max. Number Of Days} \\ 1 - \frac{\text{\#days}}{\text{Max. Number of Days}}, & \text{Otherwise} \end{cases}$
User	Number of authentication failures	$Q = \begin{cases} 0, & \text{\#failures} > \text{Max. Number Of Allowed Failures} \\ 1 - \frac{\text{\#failures}}{\text{Max. Number Of Allowed Failures}}, & \text{Otherwise} \end{cases}$
User	Lock Outs	$Q = \begin{cases} 0, & \text{\#Lock Outs} > \text{Max. Number Of Allowed Lock Outs} \\ 1 - \frac{\text{\#Lock Outs}}{\text{Max. Number Of Allowed Lock Outs}}, & \text{Otherwise} \end{cases}$
Application	Developer Reputation	$Q = \frac{\text{Reputation}}{\text{Max. Reputation}}$
Application	Who manages the software	$Q = \begin{cases} 1, & \text{Global Adminstrator} \\ 0.5, & \text{Local Administrator} \\ 0, & \text{No Administrator} \end{cases}$
Connection	Number of hops	$Q = \begin{cases} 0, & \text{\#Hops} > \text{Max. Number Of Hops} \\ 1 - \frac{\text{\#Hops}}{\text{Max. Number of Hops}}, & \text{Otherwise} \end{cases}$

Trust Quantification – ML based Approach

- The gap between the observed behavior of an entity and the expected behavior can be used as a Trust Indicator
- ML-based techniques can be used to quantify TI
  - We train the autoencoder model with the normal samples. After completing training, the model is able to detect anomalies
  - An abnormal sample should have a reconstruction error that is higher than the threshold θ, while a normal sample should have a reconstruction error lower than the threshold θ.

Autoencoder-based Trust Quantification

- Autoencoder: is a class of artificial neural networks that learn to reproduce approximations to typical examples used in their training data.



Overall Trust Evaluation Strategies

There are three strategies to obtain an overall value of the trust: **Pessimistic**, **Average**, and **Optimistic** as shown in blow. In our approach, we adopt the **pessimistic** approach in evaluating the network slice trust value.

A system is only as secure as its weakest component.

Table II. Trust Evaluation Strategies

Trust Confidence	Trust Evaluation Strategy
Optimistic	$T(flow) = max\{T_{de}, T_{ie}\}$
Average	$T(flow) = avg\{T_{de}, T_{ie}\}$
Pessimistic	$T(flow) = min\{T_{de}, T_{ie}\}$

Trust Evaluation based on Direct Evidence (T_{de}):

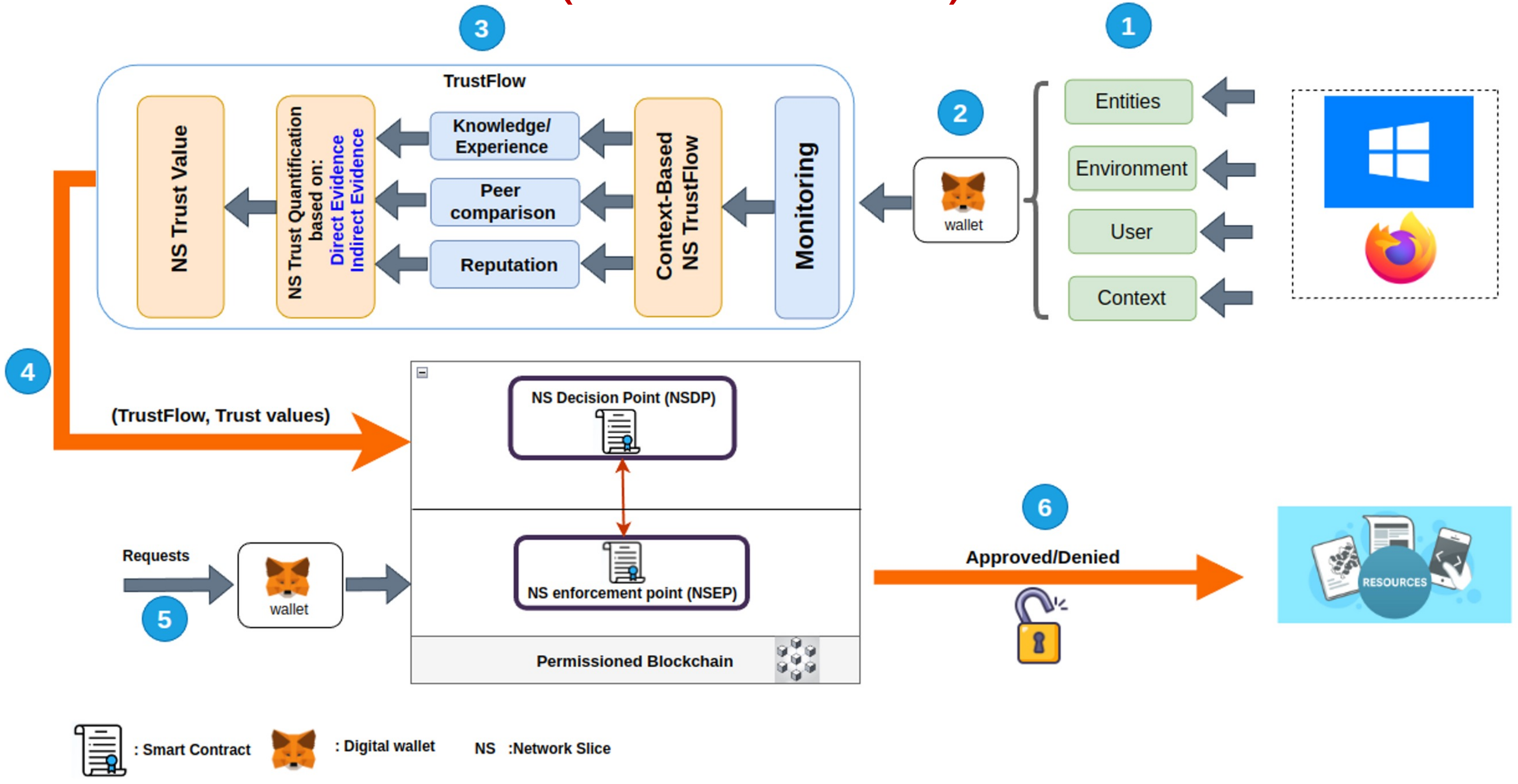
$$\begin{aligned} T_{de}(NS\_TFlow, T) = min\{ \\ T_{de}(\mathrm{NS\_TFlow}, \mathrm{User}), \\ T_{de}(\mathrm{NS\_TFlow}, \mathrm{ServiceProvider}), \\ T_{de}(\mathrm{NS\_TFlow}, \mathrm{ResourceProvider})\} \end{aligned}$$

Implementing a Zero Trust Architecture using blockchain technology with smart contracts

- The blockchain and smart contracts provide the following benefits:
  - Confidentiality, Integrity, and Availability.
    - It uses encryption to protect user privacy and transaction information.
    - It is a decentralized system.
    - The authenticity of a transaction is verified and confirmed by participants.
  - It ensures that the content of the block remains trustworthy at all times. Changes will be detected by other nodes, so it brings trust to the system environment.

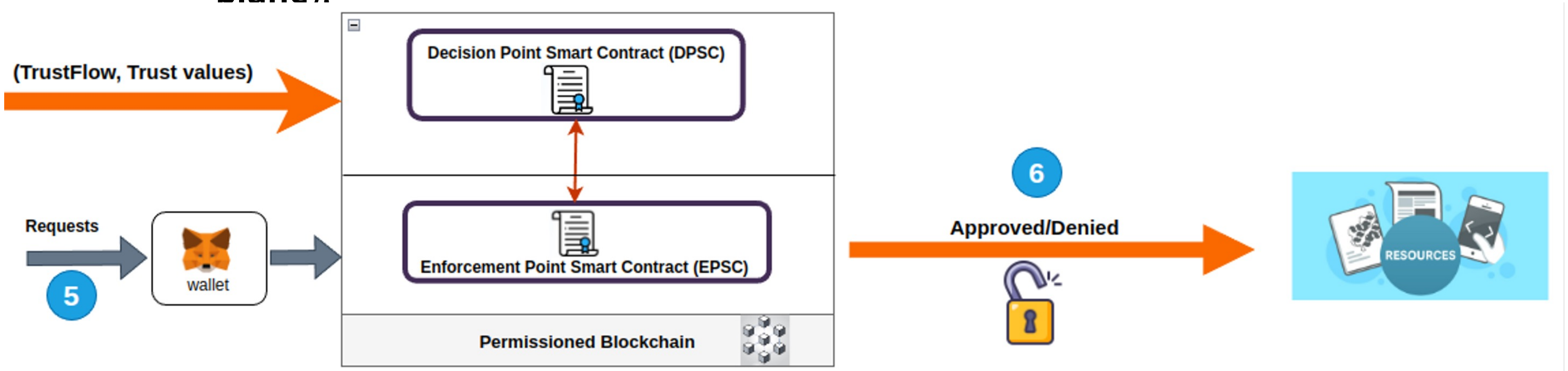
BC is not about financial servies or bitcoin its much more than that.

ZTA Implementation Using Blockchain Technology (Architecture)



ZTA Implementation Using Blockchain Technology

- Decision Point Smart Contract (DPSC):
  - This contract will have all security policies, rules, and trust levels to make a decision on whether an entity should be given access permission or not. (**control plane**).
  - Only selected admins can interact with this contract.
- Enforcement Point Smart Contract (EPSC):
  - This contract can be accessed by all entities on the network. Access requests need to be submitted to this contract.
  - EPSC communicates with DPSC to determine whether the access is allowed (**data plane**).


(TrustFlow, Trust values)
Decision Point Smart Contract (DPSC)
Requests
5
wallet
Enforcement Point Smart Contract (EPSC)
Permissioned Blockchain
6
Approved/Denied
RESOURCES

ZTA Using Blockchain Technology Implementation Tools

TrustFlow Development Tools

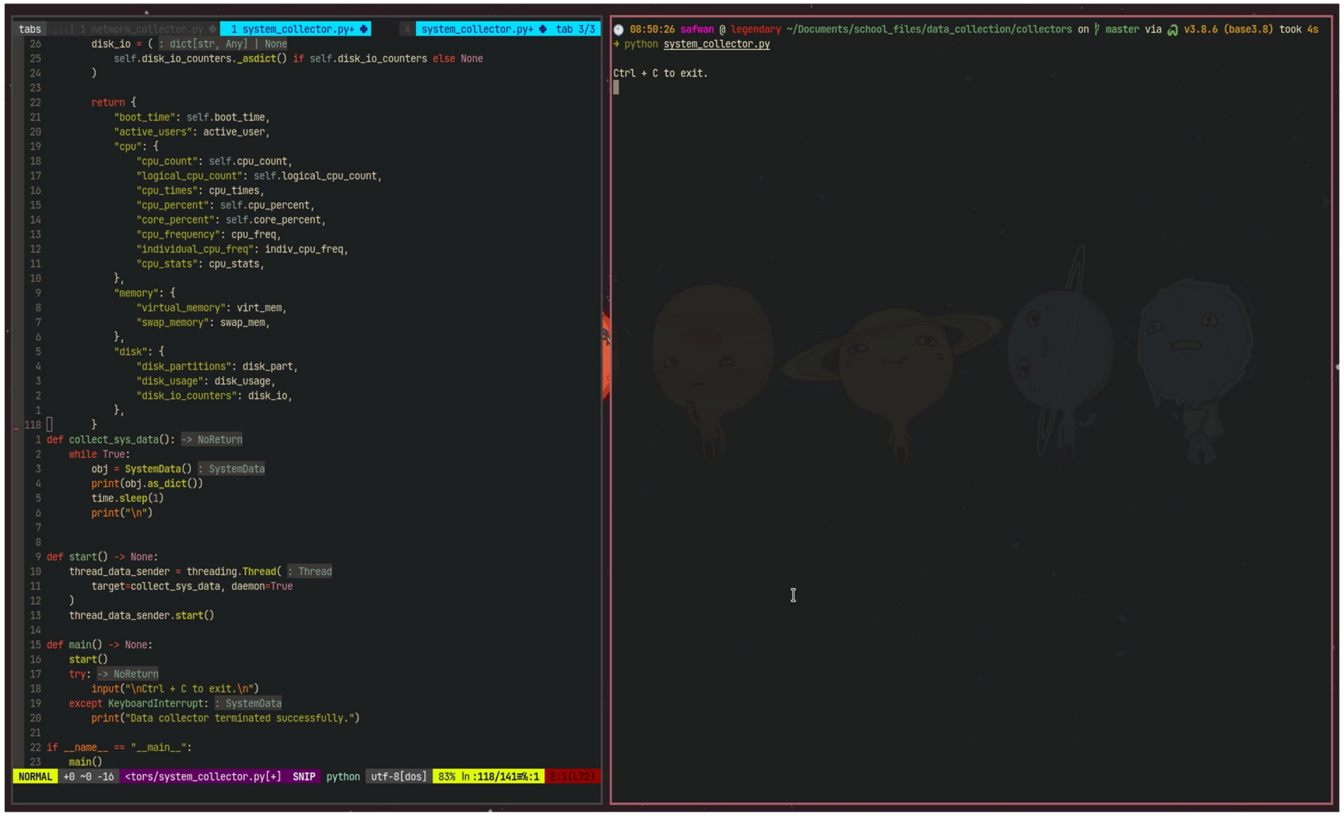
- Python, and web3 library

Blockchain Development Tools

- Solidity, javascript, and web3 library
- Truffle suite: https://trufflesuite.com/
  - Smart contract compilation
  - Interactive console for direct contract communication
  - Github: https://github.com/trufflesuite/truffle
- Ganache: https://trufflesuite.com/ganache/
  - Personal blockchain for Ethereum development.
  - Github: https://github.com/trufflesuite/ganache-ui

ZTA Implementation Using Blockchain Technology (Data Collection)

- Type of data that is being collected:
  - Application data (only this for the demo)
  - Network data
  - System data
- Data is send to Kafka server.



```
tabs  ...| 1 network_collector.py    1 system_collector.py+     X  system_collector.py+   tab 3/3
        disk_io = ( : dict[str, Any] | None
            self.disk_io_counters._asdict() if self.disk_io_counters else None
        )

        return {
            "boot_time": self.boot_time,
            "active_users": active_user,
            "cpu": {
                "cpu_count": self.cpu_count,
                "logical_cpu_count": self.logical_cpu_count,
                "cpu_times": cpu_times,
                "cpu_percent": self.cpu_percent,
                "core_percent": self.core_percent,
                "cpu_frequency": cpu_freq,
                "individual_cpu_freq": indiv_cpu_freq,
                "cpu_stats": cpu_stats,
            },
            "memory": {
                "virtual_memory": virt_mem,
                "swap_memory": swap_mem,
            },
            "disk": {
                "disk_partitions": disk_part,
                "disk_usage": disk_usage,
                "disk_io_counters": disk_io,
            },
        }
def collect_sys_data(): -> NoReturn
    while True:
        obj = SystemData() : SystemData
        print(obj.as_dict())
        time.sleep(1)
        print("\n")


def start() -> None:
    thread_data_sender = threading.Thread( : Thread
        target=collect_sys_data, daemon=True
    )
    thread_data_sender.start()

def main() -> None:
    start()
    try: -> NoReturn
        input("\nCtrl + C to exit.\n")
    except KeyboardInterrupt: : SystemData
        print("Data collector terminated successfully.")

if __name__ == "__main__":
    main()
NORMAL  +0 ~0 -16  <tors/system_collector.py[+]  SNIP  python  utf-8[dos]  83% ln :118/141≡%:1  E:1(L72)
```

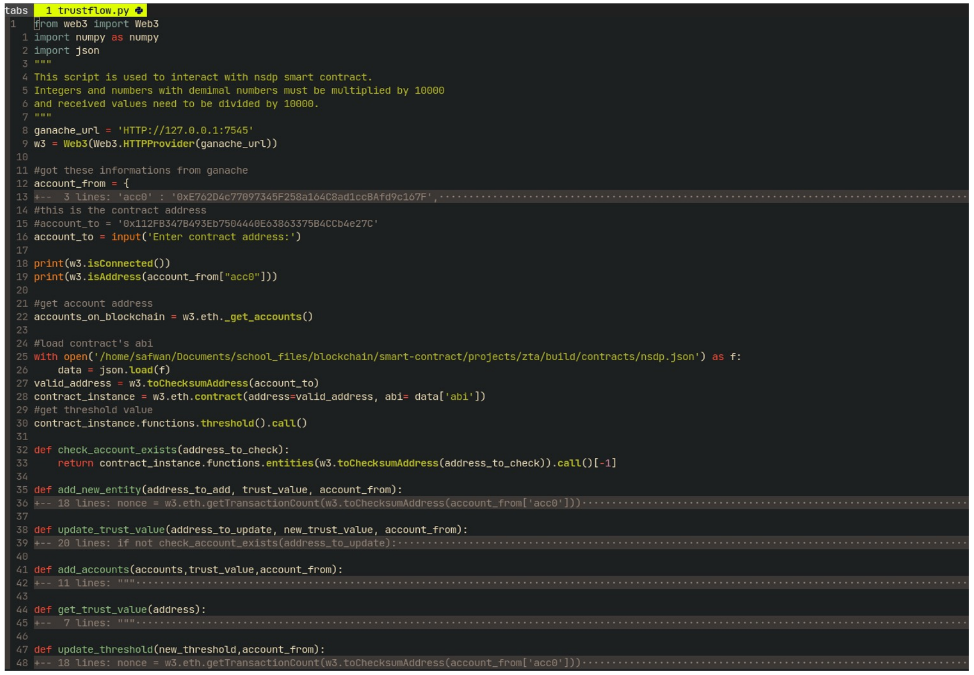
```
08:50:26 safwan @ legendary ~/Documents/school_files/data_collection/collectors on master via v3.8.6 (base3.8) took 4s
→ python system_collector.py

Ctrl + C to exit.
```

Cloud and Autonomic Computing Center

Autonomic Computing Lab

ZTA Implementation Using Blockchain Technology (Trust Flows)

- The machine learning model receives real-time data from Kafka and processes it.
- The deterministic component is getting real-time data from Kafka
- Send TI values to the smart contract to store the trust values on the blockchain.



tabs 1 trustflow.py

```python
from web3 import Web3
import numpy as numpy
import json
"""
This script is used to interact with nsdp smart contract.
Integers and numbers with demimal numbers must be multiplied by 10000
and received values need to be divided by 10000.
"""
ganache_url = 'HTTP://127.0.0.1:7545'
w3 = Web3(Web3.HTTPProvider(ganache_url))

#got these informations from ganache
account_from = {
+--  3 lines: 'acc0' : '0xE762D4c77097345F258a164C8ad1ccBAfd9c167F',·····························
#this is the contract address
#account_to = '0x112FB347B493Eb7504440E63863375B4CCb4e27C'
account_to = input('Enter contract address:')

print(w3.isConnected())
print(w3.isAddress(account_from["acc0"]))

#get account address
accounts_on_blockchain = w3.eth._get_accounts()

#load contract's abi
with open('/home/safwan/Documents/school_files/blockchain/smart-contract/projects/zta/build/contracts/nsdp.json') as f:
    data = json.load(f)
valid_address = w3.toChecksumAddress(account_to)
contract_instance = w3.eth.contract(address=valid_address, abi= data['abi'])
#get threshold value
contract_instance.functions.threshold().call()

def check_account_exists(address_to_check):
    return contract_instance.functions.entities(w3.toChecksumAddress(address_to_check)).call()[-1]

def add_new_entity(address_to_add, trust_value, account_from):
+-- 18 lines: nonce = w3.eth.getTransactionCount(w3.toChecksumAddress(account_from['acc0']))··············

def update_trust_value(address_to_update, new_trust_value, account_from):
+-- 20 lines: if not check_account_exists(address_to_update):·············································

def add_accounts(accounts,trust_value,account_from):
+-- 11 lines: """······························································································

def get_trust_value(address):
+--  7 lines: """······························································································

def update_threshold(new_threshold,account_from):
+-- 18 lines: nonce = w3.eth.getTransactionCount(w3.toChecksumAddress(account_from['acc0']))··············
```

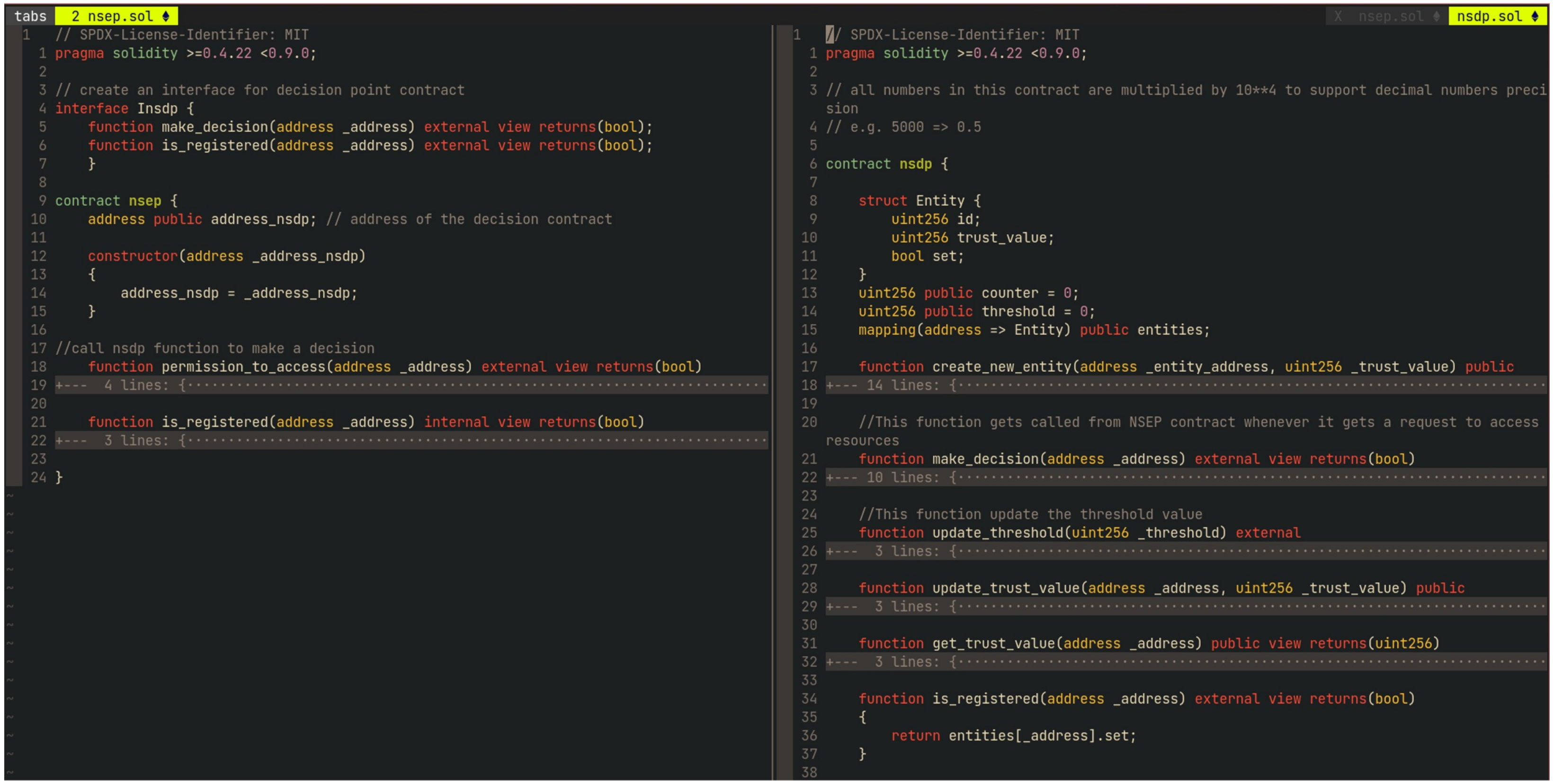
Cloud and Autonomic Computing Center

Autonomic Computing Lab

THE UNIVERSITY OF ARIZONA®

ZTA Implementation Using Blockchain Technology (Smart Contract)

- Decision Point Smart Contract (DPSC)
- Enforcement Point Smart Contract (EPSC)



tabs 2 nsep.sol

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.9.0;

// create an interface for decision point contract
interface Insdp {
    function make_decision(address _address) external view returns(bool);
    function is_registered(address _address) external view returns(bool);
    }

contract nsep {
    address public address_nsdp; // address of the decision contract

    constructor(address _address_nsdp)
    {
        address_nsdp = _address_nsdp;
    }

//call nsdp function to make a decision
    function permission_to_access(address _address) external view returns(bool)
+---  4 lines: {·························································

    function is_registered(address _address) internal view returns(bool)
+---  3 lines: {·························································

}
```

X nsep.sol   nsdp.sol

```
// SPDX-License-Identifier: MIT
pragma solidity >=0.4.22 <0.9.0;

// all numbers in this contract are multiplied by 10**4 to support decimal numbers precision
// e.g. 5000 => 0.5

contract nsdp {

    struct Entity {
        uint256 id;
        uint256 trust_value;
        bool set;
    }
    uint256 public counter = 0;
    uint256 public threshold = 0;
    mapping(address => Entity) public entities;

    function create_new_entity(address _entity_address, uint256 _trust_value) public
+--- 14 lines: {·························································

    //This function gets called from NSEP contract whenever it gets a request to access resources
    function make_decision(address _address) external view returns(bool)
+--- 10 lines: {·························································

    //This function update the threshold value
    function update_threshold(uint256 _threshold) external
+---  3 lines: {·························································

    function update_trust_value(address _address, uint256 _trust_value) public
+---  3 lines: {·························································

    function get_trust_value(address _address) public view returns(uint256)
+---  3 lines: {·························································

    function is_registered(address _address) external view returns(bool)
    {
        return entities[_address].set;
    }

```

Cloud and Autonomic Computing Center

Autonomic Computing Lab

ZTA Implementation Using Blockchain Technology (Use Case)

Use case setup:

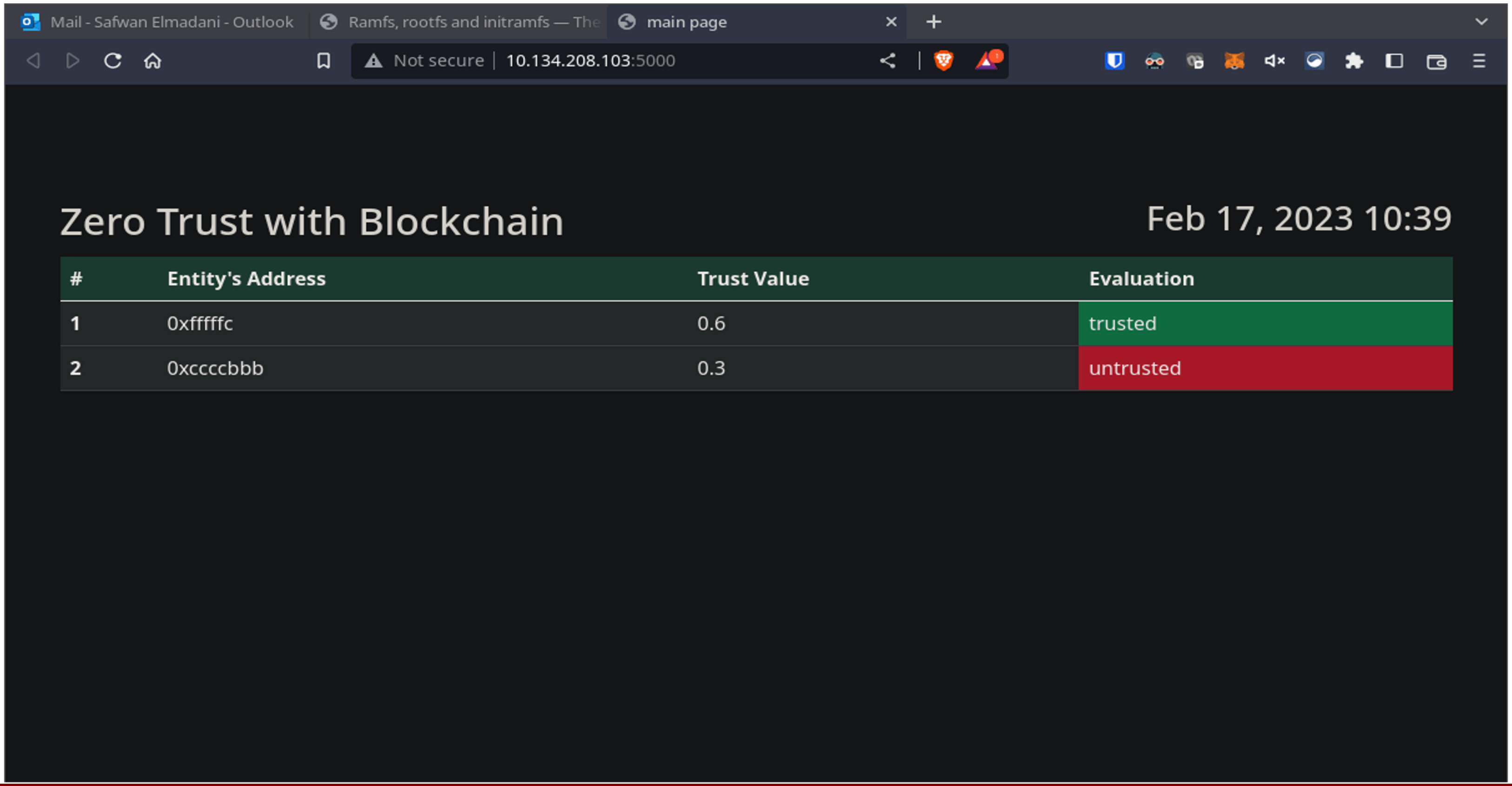
- A windows machine running Firefox browser that is being monitored by the TrusFlow to quantify its trust.
- The trust value for the browser is updated on the blockchain in real-time.
- An attack is launched on the browser that increases memory consumption very rapidly.

Results:

- The TrustFlow module was able to detect the the abnormal behavior and immediately updated the trust value of the browser on the blockchain.

ZTA Implementation Using Blockchain Technology (Use Case)

- Real-time Update of Trust Values


Mail - Safwan Elmadani - Outlook
Ramfs, rootfs and initramfs — The
main page
Not secure
10.134.208.103:5000

Zero Trust with Blockchain

Feb 17, 2023 10:39

#	Entity's Address	Trust Value	Evaluation
1	0xfffffc	0.6	trusted
2	0xccccbbb	0.3	untrusted

Future Work

- Deploy the blockchain based zero trust architecture to monitor 5G network slices and quantify their trust.
- We are currently building a 5G testbed using Openstack and Open Source MANO (OSM) to be used to experiment with and evaluate our ZTA approach
- Develop a real Agility application and demonstrate the feasibility of our ZTA implementation methodology using blockchain technology.

Backup Slides

ZTA Logical Component Functions

- **Policy Engine (PE)**. It is the ultimate decision to grant access to a resource for a given client or subject. The PE uses enterprise policy as well as input from external sources like IP Blacklists, threat intelligence services that will be used by the trust algorithm to decide to grant or deny access to the resource.
- **Policy Administrator (PA**). It is responsible for establishing the connection between a client and a resource. It would generate any authentication token or credential used by a client to access an enterprise resource. The PA communicates with the Policy Enforcement Point (PEP) when creating the connection that is done in the control plane.
- **Policy Enforcement Point (PEP)**. It is responsible for enabling, monitoring, and eventually terminating connections between a subject and an enterprise resource. This might be broken into two different components: the client (an agent) and resource side (gateway component in front of resource that controls access) or a single portal component that acts a gatekeeper for connections.

ZTA Additional Data Sources

- **Continuous Diagnostics and Mitigation (CDM) System(s)**: The system gathers information about the enterprise system's current state and applies updates to configuration and software components. The CDM provides the Policy Engine with the information about the system making an access request, such as whether it is running the appropriate patched OS and applications or whether the system has any known vulnerabilities.
- **Industry Compliance System**. This ensures the enterprise remains compliant with any regulatory regime they may fall under.
- **Threat Intelligence Feed(s):** it provides information from outside sources about newly discovered attacks or vulnerabilities. This includes DNS blacklists, discovered malware, or command and control systems that the Policy Engine will want to deny access to from enterprise systems
- **Data Access Policies:** These policies are the starting point for granting access to a resource as they provide the basic access privileges for actors and applications.
- **Enterprise Public Key Infrastructure (PKI):** This system is responsible for generating and logging certificates issued by the enterprise to resources, actors, and applications.
- **ID Management System:** This is responsible for creating, storing and managing user accounts and identity records. This has all information related to user (name, email, certificates, etc.) and other enterprise characteristics such as role, access attributes, or assigned systems.
- **Security Incident and Event Management (SIEM) system:** It aggregates system logs, network traffic, resource entitlements, and other events that provide feedback on the security posture of enterprise information systems. This data is then used to refine policies and warn of possible active attacks against enterprise systems.