November 15, 2020

Requirements Document

Cybersecurity Monitoring Client Requirements

for Research Project:
Cybersecurity Considerations for Blockchain Systems - Version 1.2

By

M. T. Midani

1 Introduction

This document provides the functional requirements and design specifications for a light-weight client (application) to be developed, coded, and tested on a selected IoT device for this project.

The objective is to assess levels of cybersecurity risks of blockchain-based systems by capturing key parameters on the selected IoT device, and log/store these parameters on the device itself, and also push them to the VizLore Hyperledger Server with cryptographic access to create permanent traces of the device signature and its operation. This information will be used by a subsequent phase of this project to create a mathematical model that will be able to detect and predict cyberattacks on IoT devices.

2 Task at Hand

The task at hand is to develop a *Cybersecurity Monitoring Client (CMC)* on a selected IoT device to monitor and log the following information:

- **Static information,** which is collected from device itself, such as manufacturer, hardware features, software features, version number, etc.,

- **Dynamic information,** which is generated during IP-based communication handshaking sequences, as defined by the respective communication protocols used by the device to connect with the network, such as security key lengths, encryption and signature algorithms, refresh periods, initialization vectors and similar cryptographic-relevant parameters.

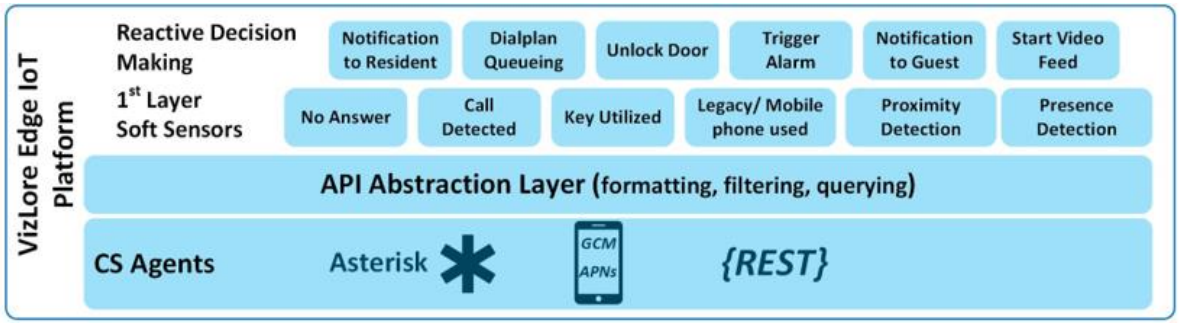
These device parameters/attributes will be logged and stored with cryptographic access. This document will define the parameters to be captured on the device itself, and on the adjacent devices interacting with the IoT in a defined local cluster.

3 About the IoT Device

The selected IoT device is a RaspberryPi-based IoT system provided by VizLore for this project. This device is used as an edge device for smart access applications, supporting REST API interface with the Hyperledger fabric server.

4 IoT Device Protocol Stack

The selected IoT controller device functional decomposition and protocol stack are shown below.



The monitoring client is to be developed and integrated smoothly in this environment.

> *Note: The controller/client is not intended to be an active blockchain node but rather a client with access to blockchain ledger.*

5 Design Requirements and Specifications

The IoT *Cybersecurity Monitoring Client (CMC)* shall acquire the following parameters:

1. Static Parameters: Upon device boot/startup sequence, the client shall make function calls to the OS to collect the following parameters:

    a) Device MAC address & serial number

    b) Manufacturer

    c) OS type and version number

    d) Memory size

    e) If the device IP address was hard-coded, then this static address should also be collected by the client. If the IP address is dynamically assigned to the device, then the IP address will be determined using DHCP, as described in the next section.

    f) Other static system parameters are to be considered

    These parameters should be stored on the device and on the remote Hyperledger Server.

2. IP layer Addressing Parameters: An IP device will typically run the DHCP protocol after a powerup sequence, and will exchanges the DHCP messages (Discover, Offer, Request, and Ack) with the local/remote Gateway. The IoT CMC shall capture the following parameters:

    a) Device dynamic IP address and subnet mask

    b) First Hop IP address and subnet mask

    c) Name and IP address of the DNS server

    d) Other IP layer parameters in the DHCP messages are to be considered

    These parameters should be stored on the device and on the remote Hyperledger Server.

3. IP Layer Error Reporting Protocol: This is the standard ICMP protocol used by IP devices to discover other IP devices on the Internet. The IoT CMC shall capture the following parameters:

    a) Echo request & Reply (ping)

    b) Destination unreachable

    c) TTL Expired (routing loop, or destination is too far)

    d) Traceroute information

    These parameters should be stored on the device and on the remote Hyperledger Server.

4. ARP Cache: On a single physical network, individual hosts are known to other devices in the network by their physical hardware address. Since higher-level protocols address destination hosts using IP addresses, a translation table between Physical MAC addresses and Network IP addresses is created by all the devices on the local network using the ARP protocol. This is known as the ARP Cache.

    The ARP Cache should be captured periodically (periodicity to be determined), and stored on the device and on the remote Hyperledger Server.

5. DNS: Each IP device has a name resolver routine, which knows the name of a local DNS server. The Resolver sends a DNS request to the DNS server to get the desired destination IP address from the destination name. The IP device keeps a DNS Cache in its memory.

    The DNS Cache should be captured periodically (periodicity to be determined), and stored on the device and on the remote Hyperledger Server.

6. TCP layer Connectivity Parameters: TCP sessions are created and released all the time between IP devices communicating over the Internet. The IoT CMC shall capture the following parameters:

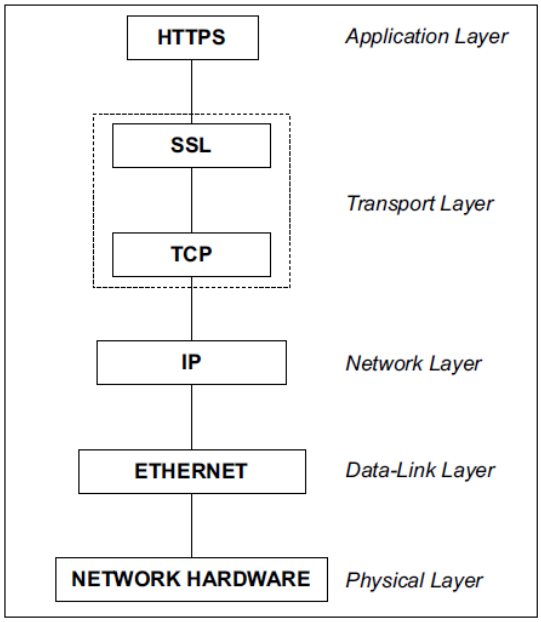
    a) Source IP address and Port

    b) Destination IP address and Port

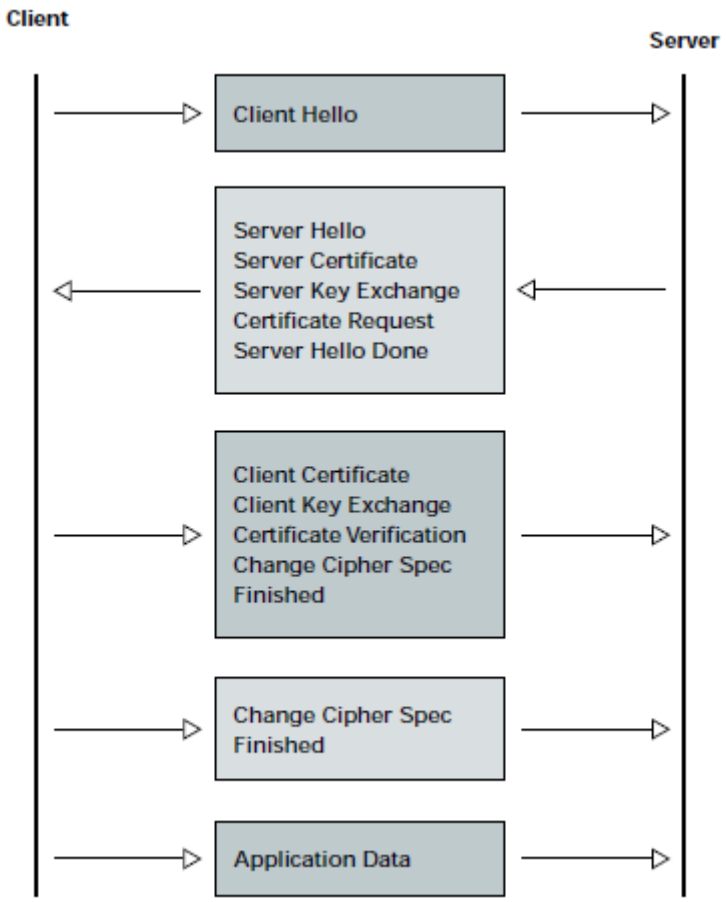
c) Will be considering other parameters…

These parameters should be stored on the device and on the remote Hyperledger Server.

7. SSL layer Connectivity Parameters. Secure Sockets Layer (SSL) was originally developed by Netscape Communications to allow secure access of a browser to a Web server. SSL is the accepted standard for Web security. The main role of SSL is to provide security for Web traffic. Security includes confidentiality, message integrity, and authentication. SSL achieves these elements of security through the use of cryptography, digital signatures, and certificates. The SSL reference model in relation to TCP/IP protocol stack is as shown below.



Every time a device attempts to establish a secure connection/session with a web server, handshaking for session initialization takes place between the client and the server to exchange and agree upon the keys to be used for cryptography, digital signatures, and certificates. The handshaking is shown below.



The key parameters in this handshaking are the following:

a) Server IP address

b) Server's digital certificate

c) Agreed upon Cryptographic algorithm (between the client and the Server)

d) Server's public Key

e) Cipher

These parameters should be stored on the IoT device and on the remote Hyperledger Server.

8. Cookies: Servers deposit cookies on the client device when the server is accessed. The cookies should be captured periodically (periodicity to be determined), and stored on the device and on the remote Hyperledger Server.

9. Browsing History: Browsers keep history of web activities on the local device. This history should be captured periodically (periodicity to be determined), and stored on the device and on the remote Hyperledger Server.

6 Hyperledger Server

The parameters listed above should be pushed to the Hyperledger fabric (Server IP address will be provided) over the REST API interface to create a permanent record of this activity.

7 Concluding Remarks

This version of the document lists the initial parameters being considered for the IoT CMC. The parameters will be used to create a mathematical model that should be able to detect and predict cyberattacks on IoT devices, using machine learning techniques.

The current list of parameters will be revised as we further investigate the capabilities of the selected IoT device. A balance is to be made between the volume of collected parameters and the value gained from a cybersecurity perspective.