Design Description Document

Version 1.0~~0~~

Cybersecurity Monitoring Client Design Description
For research project: Cybersecurity Considerations for Blockchain Systems

~~**Outline:**~~

1. **Introduction**: ~~Objectives and reference to the Requirement document~~

   The objective is to develop a light-weight monitoring client to assess levels of cybersecurity risks of blockchain-based systems by capturing key parameters on the selected IoT device. These key parameters can be categorized into static information and dynamic information. This information shall be stored locally as well as on a blockchain Hyperledger fabric ledger, through REST API calls.

   ~~1.~~

~~1.~~ **Target IoT device Description**

2.

   a) Hardware platform

      Model: 3B
      Architecture: aarmv7l
      ~~b)~~ Processor: 32-bit

   b) Operating System & Programming Environment

      OS: Raspbian GNU/Linux
      Version: 9
      ~~c)~~ Type: Debian

   ~~d)~~ Basic Functions of IoT

   c)

      The IoT device is generally used to work in a specific limited environment. In this case, the IoT device is something that can be used in a day to day life and controlled through smartphones. It can be used for trivial stuffs such as authoring and giving access to certain people, automating smart homes etc. This type of device can be used in various places such as homes, factories, hospitals, schools based on the design and specification of the IoT device.

   d) IP address

      IP Address is allocated dynamically using DHCP
      ~~e)~~

3. Cybersecurity Monitoring Client (CMC) Description

~~2.~~

   a) Position in architecture

Formatted: Font: Bold

Formatted: Line spacing: 1.5 lines

Formatted: English (United States)

Formatted: Indent: Left: 0", Line spacing: 1.5 lines, Numbered + Level: 1 + Numbering Style: 1, 2, 3, … + Start at: 1 + Alignment: Left + Aligned at: 0.25" + Indent at: 0.5"

Formatted: Indent: Left: 0.25", No bullets or numbering

Formatted: Font: Bold

Formatted: Space After: 0 pt, Line spacing: 1.5 lines

Formatted: Indent: Left: 0", Line spacing: 1.5 lines, Numbered + Level: 1 + Numbering Style: 1, 2, 3, … + Start at: 1 + Alignment: Left + Aligned at: 0.25" + Indent at: 0.5"

Formatted: Underline

Formatted: Space After: 0 pt, Line spacing: 1.5 lines

Formatted: Normal, Indent: Left: 0.5", No bullets or numbering

Formatted: Underline

Formatted: Indent: Left: 0.25", Space After: 0 pt, Line spacing: 1.5 lines, Numbered + Level: 1 + Numbering Style: a, b, c, … + Start at: 1 + Alignment: Left + Aligned at: 0.5" + Indent at: 0.75"

Formatted: Indent: Left: 0.5"

Formatted: Normal, Indent: Left: 0.5", Line spacing: 1.5 lines, No bullets or numbering

Formatted: Underline

Formatted: Space After: 0 pt, Line spacing: 1.5 lines

Formatted: Indent: Left: 0.25", Space After: 0 pt, Line spacing: 1.5 lines, Numbered + Level: 1 + Numbering Style: a, b, c, … + Start at: 1 + Alignment: Left + Aligned at: 0.5" + Indent at: 0.75"

Formatted: Indent: Left: 0.5", No bullets or

Formatted: Underline

Formatted: Indent: Left: 0.25", Space After: 0 pt, Line spacing: 1.5 lines, Numbered + Level: 1 + Numbering Style: a, b, c, … + Start at: 1 + Alignment: Left + Aligned at: 0.5" + Indent at: 0.75"

Formatted: Indent: Left: 0.5", No bullets or

Formatted: Font: Bold

Formatted: Space After: 0 pt, Line spacing: 1.5 lines

Formatted: Font: Bold

Formatted: Underline

Formatted ...

The cybersecurity monitoring client sits within the IoT device monitoring all the network communications in and out of the IoT device.

b) Functions

The CMC should be able to pre-emptively predict / detect if there is going to be a cyber attack on the IoT device, using the static and dynamic information stored on the Hyperledger.

c) Interfaces

Currently the IoT device is interfaced using CLI (Command Line Interface) through socket programming to imitate communication through REST API to transfer the static information collected during the boot sequence.

4. Design Decisions

a. Do we need to obtain "Requirement 1" on the fly?

Since these variables / parameters are the initial messages transferred between the device and the gateway to establish the connection right after bootup, it might seem necessary to capture the this data while these messages are exchanged, but these variables can also be obtained later to find parameters such as "Device IP", "Gateway IP", "DNS Nameservers" etc.

b. ARP Cache (vs) ARP Table

Although the IoT device maintains ARP cache of the devices that the device had communicated previously / is currently communicating, along with the devices on the network whose IP address has been associated with a MAC Address, it is also important to store the contents of ARP Table of the network periodically. There might be cases where the contents of ARP cache have been poisoned / different from that of the device in case the attacker has specifically targeted a certain device. This can be only detected by comparing it with the networks ARP Table. Hence it is necessary to maintain both the ARP Cache and the ARP Table.

~~3.~~5. Implementation/Coding Description

a) Function calls for getting Static Information & Internal data structure

Requirement 1. Collect static parameters of the device upon boot sequence.

- Device MAC address: Used "getInterfaces", "getMAC" packages to obtain MAC Address.
- Serial Number, Manufacturer, Hardware, Memory size: The file "/proc/cpuinfo" contains relevant information such as Serial Number, Manufacturer, Hardware etc.
- OS Info and version number: Upon reading the contents of "/etc/os-release" file, we obtain information pertaining to the operating system.
- Static IP (if assigned): Categorize the IP by the contents of "/etc/network/interfaces"

Formatted: Font: Bold

Formatted: Font: Bold, English (United Kingdom)

Formatted: Font: Bold, English (United Kingdom)

b) Function calls for getting Dynamic Information & Internal data structure

Requirement 2. Obtaining IP layer Addressing Parameters

Since parameters such as IP Address, Broadcast IP, Gateway IP and DNS Nameservers need not be captured on the go, as these can be retrieved back.

- IP Layer Parameters: Used “netifaces” package to handle IP Layer addressing.

Captured Dynamic Device IP, Broadcast IP, Gateway / First Hop IP Addresses along with their respective subnet mask, information about DNS Nameservers.

~~4.~~6. Pushing the parameters to the Hyperledger Server:

- Describe how it is done
- List the REST messages/calls

~~5.~~7. Next steps

~~6.~~ Concluding Remarks