**PROJECT DESCRIPTION/REQUIREMENT.**

Develop a Python Desktop application to interact with NTAG 213/214/216 and similar types of NFC Cards. Following are the deliverables:

Desktop Application:

===================

Purpose: Create a Desktop application in Python to bulk read, write NFC Tags with custom data.

The NFC Writer Device to be used is:

Action: The program shall write custom URLs into the NFC tags so that the tag can be used as interactive shortcut to make a transaction.

For example:

1. To wite a text string into the tag:

record = ndef.TextRecord("Hello sigsec readers!")

2. To put an email string into the tag:

record = ndef.TextRecord("Hello sigsec readers!"

3. Request a Payment: URL like https://pay2mate.com/registrations/merchantpay

NDEF record types

There are a number of different types of records that we can create, with different uses and applications. To write any of these to a smartcard or NFC tag, use the steps in the heading above with the record object produced below.

Text records

Text records are super easy; they contain a string of some text, a language key and an encoding. If you don’t specify anything, ndeflib will use en for the language and UTF-8 for the encoding by default.

record = ndef.TextRecord(text, language, encoding)

# eg. record = ndef.TextRecord("Hello sigsec readers!")

URI records

A lot of useful actions can be constructed using the URI record. The basic form of the record constructor is:

record = ndef.UriRecord("https://www.example.org/")

If you want to go straight to Google Maps navigation, put in an address in this format (or look here for more Google intents):

record = ndef.UriRecord("google.navigation:q=1600+Pennsylvania+Avenue,+Washington+DC")

To compose an email (even including a pre-filled subject line), use this format:

record = ndef.UriRecord("mailto:recipient@example.com?subject=Hello%20there")

To dial a phone number, use this format:

record = ndef.UriRecord("tel:+61400123456")

When such NFC Tag with the custom URL is scanned by an NFC Application, the web URL contained in the tag is opened and a Transaction processing is started(The URL we will provide will have such option.)

References are below:

References:

Here is a portal providing some links:

https://libs.garden/python/search?q=nfc

1. NFC Card Writer that will be used is :

https://www.acs.com.hk/en/driver/3/acr122u-usb-nfc-reader/

Setting up ACR122U for Bulk writing NFC Cards

2. https://blog.sigsec.net/posts/2019/04/nfc-fun-with-python.html

Reading and writing NFC tags

3. https://blog.sigsec.net/posts/2019/04/reading-and-writing-nfc-tags.html

Reading and writing NFC tags with Python using ACS-ACR122U-NFC-Reader/blob/master/NFCReader

4. https://github.com/StevenTso/ACS-ACR122U-NFC-Reader/blob/master/NFCReader.py

5. How to bulk write NFC Tags using a unix shell script

https://www.youtube.com/watch?v=c0Qsmgvj\_oo

b. Code for the shell script and the Git is below:

The bash script i wrote can be found here:

The NFC Card reader/writer used for this POC is called ACR122U-A9

The Linux type I am using is a Debian 9 with a newer kernel version 4.18.3

I got the reader to work at this kernel level: Linux Lab 4.9.0-6-amd64

The mfoc utility can be found here: https://github.com/nfc-tools/mfoc.git

The RFID i clone the card to was bought on ebay found by the search string:

"10PCS UID Changeable Sector 0 Block 0 Writable RFID Proximity Card" (the white cards)

and

"5PCS UID Tags Changeable Keyfob Compatible MCT Block 0 Direct Writable by Phone" (the blue round tags)

UPDATE 2019: A little help to installing

sudo apt-get install pcscd pcsc-tools autoconf2.64 autoconf libnfc-dev

# Test the reader:

pcsc\_scan

# Optional : libacsccid1

# sudo service pcscd restart

# Install mfoc

git clone https://github.com/nfc-tools/mfoc

autoreconf -is

./configure

make && sudo make install

Seshu Kanuri

Pay2mate Inc.