Sphinx: A Novel Encryption Platform for Secure Barcodes

Background:

Millions of barcodes are interacted with on a day to day basis with no hesitation or afterthought. Barcodes have become a constant in every-day life we do not question, we only accept. But what is actually contained in a barcode? A barcode whilst it may appear complex on the surface is little more than encoded plain text. This encoding provides no security itself and is even readable by the human eye with some knowledge of barcode technologies. This itself is a massive unrecognised security consideration, especially when the scope of barcode applications is considered.

A PDF417 Barcode used
in the airline industry

Aim:

Sphinx aims to solve this problem by removing any discernible data away from the barcode itself preventing any would be attacker from devising any illicit information. This is achieved through a hybrid encryption scheme using both a client application and control server to control the encryption and decryption of data. The encryption system itself is novel in addition to asynchronous programming being used for client server connections.

In layman's terms, by removing the data from the barcode we remove the attack vector. Data is stored encrypted on our servers and cannot be decrypted without authorisation. A server much is harder to break into than a piece of text on paper!

Why Sphinx?

In ancient Greek mythology the Sphinx is said to have guarded the ancient city of Thebes, asking those who entered its riddle. Those who answered correctly were granted entry, those who failed were killed. The platform not only creates riddles in the form of encrypted text but prevents entry to unauthorised users.

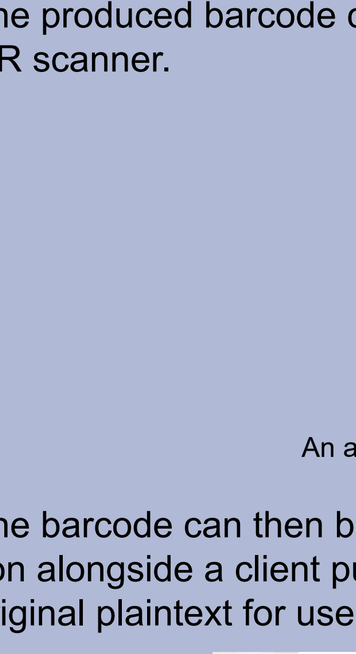
Methods:

The system itself is written in Python and is a proof of concept. The encryption system was designed with inspiration from the encryption systems found by dissecting cryptolocker malware.

To authenticate users a working login system is created which implements the bcrypt algorithm to hash and salt passwords to further prevent hash (password) cracking. Salting this hash further makes it resistant to rainbow table attacks. Bcrypt itself is an iterative algorithm more secure for password hashing (compared to SHA-512 for instance) due to the slow nature of it. Hashed passwords are checked against hashes in the database to authenticate users.

An exerpt of the users.db table
containing a hashed password.

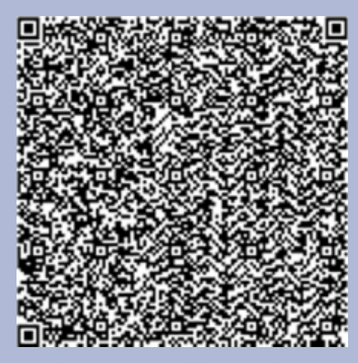
Methods:



The hybrid encryption system used
by the platform

The system utilises a hybrid encryption scheme of both symmetric encryption (AES-128) and asymmetric encryption (RSA-2048); both proven mathematically extremely secure and current industry standards.
Symmetric encryption is used for the bulk of encryption due it being less computationally intense with this job also being given to the client rather than server to reduce load.
The encryption system is designed with security in mind with no un-encrypted text being stored or sent over network.

The produced barcode contains no discernable data even read with a QR scanner.



An actual encrypted barcode produced
by the stystem

The barcode can then be scanned in and sent to the server for decryption alongside a client public key. This public key is used to encrypt the original plaintext for use for network transmission.

An example of encrypted data on the
server ready to be decrypted

The platform also implements asynchronous networking to promote scalability. Asynchronous networking has the advantage over multi-threaded systems that have memory blocked off for a single thread, handled by the python scheduler. Instead callbacks are used so that memory is allocated on the arrival of data. Using this over threading server connections can be scaled from thousands to hundreds of thousands.

```
async def tcp_echo_client(loop, passw, user,loginAttempts):
    reader, writer = await asyncio.open_connection('127.0.0.1', 1121,
                                                   loop=loop)
```

An asynchronous loop class
object used for login networking

Conclusion:

The proof of concept works well as a solution to a previously ignored problem. Further implementation such as SSL certificating could further be implemented to authenticate both clients and secure networking. The platform is currently configured for use with airlines but the overarching platform can be used in any environment where security is needed. The project itself has furthered my deep-rooted passion for cyber security and I look forward to furthering my research in cryptography and biometrics.

Nicholas Venis
Email: 15565251@students.lincoln.ac.uk

Supervised by Dr. Georgios Leontidis

UNIVERSITY OF LINCOLN