COPENHAGEN SCHOOL OF DESIGN AND TECHNOLOGY

Ransomware PoC

By Magnus Holgersen

For exam in Crypto 2019

Teacher: Lorena Ronquillo Moreno

10-02-2019

Introduction:

For my exam project i've decided on making a ransomware proof of concept.
I want to find out how hard it would be to create a script that can lock people out of their systems or specific files.

The idea behind a ransomware, is to get it on someones(target) computer and run the script, the script will encrypt files on the target computer, and have them pay you for the key/password to decrypt the files so they can be accessed once again.

The script i've made, is made in Python and loops through the folder it is placed in, it will then encrypt all files in the directory and all its subfolders. To encrypt a password is entered wich will be hashed and used as the key for the encryption, once a password have been given all files will be encrypted using that password, to decrypt files one will have to give the password used to encrypt the files, and the file name, only one file can be decrypted at a time.

For this we will need the script to be able to find the files and encrypt them with a password and decrypt them with the same password.

Design and implementation:

Language: Python

I've chosen to write my ransomware PoC in python because it is the language used in class to teach cryptography, and it contains a lot of useful and easy to use libraries.
With the vast choices in libraries it is was an easy choice as to what language i would use for the project.
I am mainly using the PyCryptodome library for all AES encryption, it is from this library i am importing the AES encryption, SHA256 and Padding. AES is the import i am using to create the encryptor, for which i need a key, this is where the SHA256 comes in, this is used to hash my password which can in turn be used as a key for the encryption.

Encryption: Advanced Encryption Standard(AES) Cipher Block Chaining (CBC)

For my script i have chosen to use AES with CBC, i am using the PyCrypto library.
To use CBC you must have an initialization vector(IV), which is takes the place of the "first" cipher text, as each block is XORed with the next, therefore we need the IV to start the encryption, because of the fixed block size the IV also needs to be 16 byte of length, therefore i am using os.urandom(16) to generate a 16 byte IV, which i can then use to start the encryption, i am saving the IV in the encrypted file so i can use it to decrypt the files again.
Furthermore for an AES CBC encryption you need your key to be 128, 192 or 256 bits long, to ensure this i use an SHA256 hash function on the password to generate a key that have an length of 256 bits.
Padding is needed if the plain text is not divitional by 16, as mentioned above, every block size have a fixed size of 16 bytes, se we need to add bytes to the plain text to ensure that the block size is 16 bytes when we start encrypting, i am using the

PyCrypto library which have a built in padding method, so before i encrypt i pad the data, likewise when i decrypt i unpad the data after i have decrypted it.

Discussion:

The script i've created is working as intended, it is placed in a folder and encrypts all the files to they cannot be accessed or used by the owner, and without knowing the password used for the encryption you cannot decrypt the files, however the passwords are not stored anywhere so it suffers from human error and complex passwords can be hard to remember.
I've tested that the encryption decryption works as intended by running the script in a folder and encrypting .txt files as well as .png, after decryption of the files i've made sure to open the files to see if they reverted to their normal state.
a ransomware script can be dangerous to play around with, as i've experienced while creating it and testing it, during my work period i've accidentally encrypted the entire project folder and had i not made a safe copy of the project it would have been lost, in conclusion it is a very powerful way to pressure people for money, but if used wrongly or placed in a wrong place, it could lock down your computer for good.

Conclusion:

From this project it can concluded that creating a basic ransomware script is fairly easy to make, but if used recklessly, you can just as easily damage the files or the entire computer system it is ran on.
For future improvements and implementations, if one was to actually use this as malware, it would need to be able to encrypt by it self and create a password for the encryption, that would in turn be stored in a database through a secure connection that could not be tracked. In this database you could also store location of the targeted computer, such as the MAC address.