Anonybox

A deep journey into Anonybox's Technology

Morad Abdelrasheed Mokhtar Ali Gill

6/28/2020

Comprehensive Technology made by a 15 years old teenager to Establish Anonybox.

First: JSNCP.

Also known as "Java Secure Network Communication Protocol", is a protocol by which data can be transferred between two endpoints. The features of this Protocol are as following:

1- A workaround for writeUTF(). You can send Data Bigger than 64KB using Buffer Technology. DATA_BUFFER_CONST is responsible for the Data Size per packet. Actual Packet Size is DATA_BUFFER_CONST + 6 bytes "A link by which the read() function knows that the next packet is appended to the existing one".
2- Uses AES-256. It uses AES-256 with CBC mode and PKCS5 Padding with IV customization.
3- IV Customization. Can be used to exchange IVs between client and server "will be discussed later".

This JSNCP is a very important component for Anonybox. As it's used to receive and/or send packets.
Note for non-java-programmers: writeUTF() is the function we use to pass data through connection. read() function is the function we use to read data from the stream.

IV Exchanging? , What is it?

Anonybox uses IV Exchanging to Increase Security Level, The process goes like this:
1- Client Requests a random IV from the server
2- Server generates the random IV and sets the Default IV to the generated IV
4- Server sends the IV to the Client under AES-256 Encryption
5- Client receives the IV and sets the Default IV to the generated IV
7- Client sends a TEST Message to the Server, if the two IVs are identical, Server will reply with a confirmation message

What about the Encryption Key?

Anonybox doesn't use Diffie-Hellman Key Exchanging Algorithm at the moment*. Although it's required by the client to know the Encryption Key. The process goes like this:
1- Client Connects to the server
2- Client Sends to the server a TEST message with All-Zeroed IV "till the client receives the new IV"
3- if the two Encryption Keys are identical, Server will reply with s confirmation message

* = Will be added in the future

Encryption Key isn't the same !

Yes, the Input Encryption Key isn't the actual one. Instead, it takes first 16 bytes of the Encoded-SHA1-Key and set it to work!

Second: Anonybox API.

I "individually" prefer to call it "Anonybox Utilities" as it considers as a bunch of tools rather than a complete API.

JSNCP is included in the API, along with other utilities "functions" to grease the wheels and simplify the code.

It includes:
1- JSNCP
2- Read Files "A shortcut for like 4 lines lol"
3- Generate Random Strings of length 20 chars.
4- Generate Random IV of a size of 16 bytes
5- Decrypt/Encrypt Data

About the API's Constants.

Till 28/6/2020, Two Constants Exist:
1- EMPTY_IV Which contains All-Zeroed IV
2- DATA_BUFFER_CONST Which is amount of data "in bytes" that is included in one packet "Return to Page 2 to the features of JSNCP". It's a required value by the JSNCP.

Future Plans.

1- Convert Read Files Function to Secured Read Files Function **
2- Make a secure Write on Files Function **
3- Add Diffie-Hellman Key Exchanging Algorithm
4- Improve GUI

** = Using AES-256 CBC PKCS5 Encryption with All-Zeroed IV and Encryption File "will be generated out of another Encryption key the user will enter on first time only and the user must save it for later for further usage except he deleted all files and he started fresh again"

Security?

We added a Filter that filters all special characters that can be used in many vulnerabilities such as:
1- RCE
2- Buffer Overflow

HashMapStore.

It's a feature that takes users' and admins' DB files and translates them to A HashMap.

EDelimiter is the Element Delimiter. Or we can say one Key and Value
KVDelimiter is the Key-Value Delimiter. Or we can say it's the delimiter in every element that splits the Key and Value so HashMapStore can identify them separately.

WriteData Function.

It inserts data in DB files.
If there're existing queries, a delimiter will be prefixed to the input text. Otherwise, the text will remain as it is.

WriteFile Function.

Write on file... That's it!

Why I did it? I'm so lazy and if I wrote ~5 lines each time I write on a file the code would be huge.

ReadFile Function.

Reads a file… That's it!

Why I did it? I'm so ducking lazy and also I have to replace the zero-content with 'NOTHING'.

Why? Well, When a Mailbox is empty the ReadFile() will return null, which won't send the message and/or not displayed correctly.

III – Anonybox

As we can see, Anonybox depends on many components such as JSNCP and Anonybox API and HashMapStore. Also, a firewall will be implemented. The firewall rules are as following:
1- Block >64KB Packets
2- Filter all special characters from any input and/or data that it sent through port 1111 "Official port for Anonybox" except [.] and [@] and [-] and [,] and [(] and [)] and [!] and [?] and the space character and \n and [:]
3- Limit processing speed to a specific speed like 150 p/s "packets per second"

Commands

user [user] -> enter the username
pass [pass] -> Enter the Password
IVGENERATE -> Generate an IV
TEST -> Test connection " I/O "
For user:
mail -> view mailbox
compose [content],[dest] -> Send [content] to [dest]
logout -> Logout
change password [oldpass],[newpass] -> Change the password from [oldpass] to [newpass]
change username [olduser],[newuser] -> Change the username from [olduser] to [newuser]
For Admin:
create user [user],[pass] -> Create a user of credentials [user] and [pass]
delete user [user],[pass] -> Delete a user of credentials [user] and [pass]

Anonybox in numbers

63 is the number of Anonybox API usages all over the application.
238 is the number of lines in Client side
15 is the number of lines in Firewall code
202 is the number of lines in Anonybox API
472 is the number of lines in Server side
174 is the number of lines in Starter page
1101 is the number of lines of code to make this project