Vault 102 Writeup

Target:

> 211d043827a65e07b45e20105fa8331d Vault102-1.1-release.apk

Exploitation:

Use Jadx to decompile apk file.

1. Analyse Binder Interface

There is Binder interface `IVault` used for interaction between `MainActivity` and `VaultService`.

```
/* renamed from: b.c.a.a */
public interface IVault extends IInterface {

    /* renamed from: b.c.a.a$a */
    public static abstract class Stub extends Binder implements IVault {

        /* renamed from: b.c.a.a$a$a */
        public static class Proxy implements IVault {
            ...
        }

        public Stub() {
            attachInterface(this, "com.sctf2020.vault102.IVault");
        }
        ...
    }
    ...
}
```

`VaultService` provides implementation of this interface by extending `IVault.Stub` class.

```
public class VaultService extends Service {
    /* renamed from: a */
    public IBinder binder = new IVault.Stub() /* renamed from:
com.sctf2020.vault102.VaultService$a */ {
        ...
    }

    public IBinder onBind(Intent intent) {
        return this.binder;
    }
    ...
}
```

Meanwhile, `MainActivity` binds to `VaultService` in `onCreate()` method and receiving `IVault` interface
in `onServiceConnected()` method.

```
public class MainActivity extends C0021e implements View.OnClickListener {
    /* renamed from: r */
    public volatile IVault vault;

    /* renamed from: s */
    public ServiceConnection connection = new ServiceConnection() /* renamed
from: com.sctf2020.vault102.MainActivity$a */ {
        public void onServiceConnected(ComponentName componentName, IBinder
iBinder) {
            vault = IVault.Stub.asInterface(iBinder);
            ...
        }

        public void onServiceDisconnected(ComponentName componentName) {...}
    }

    public void onCreate(Bundle bundle) {
        ...
        bindService(new Intent(this, VaultService.class), connection, 1);
    }
}
```

The main part is in `onClick()` method that invokes method `mo3487b()` with user's supplied text in EditText.

```
public void onClick(View view) {
    ...
    boolean b = this.vault.mo3487b(this.editText.getText().toString());
    ...
}
```

2. Analyze Service

When `MainActivity` binds to `VaultService` it first checks if application from challenge Vault 101 (`com.sctf2020.vault101`)
installed and signed by the same vendor in `mo3486a()` method. So Vault 101 need to be installed in order to run this application.

Method `mo3487b()` increments attempts counter (only a few attempts available before the restart) and then invokes native method `unlock()`.

3. Native C library

- Disassemble/decompile native library `lib/armeabi-v7a/libvault.so`, for instance.

```
int __fastcall Java_com_sctf2020_vault102_VaultService_unlock(JNIEnv *env,
jclass clazz, jstring jPassword)
{
    ...
    int ctx[16]; // [sp+0h] [bp-100h]
    char key[32]; // [sp+40h] [bp-C0h]
    char buffer[128]; // [sp+60h] [bp-A0h]

    password = ((*env)->GetStringUTFChars)(env, jPassword, 0);
    length = ((*env)->GetStringLength)(env, jPassword);
```

```
    ...
    sub_830(buffer);
    sub_85C(key, buffer);

    output = calloc(length, 1u);
    sub_B70(ctx, key, "Cooking");
    sub_C30(ctx, password, output, length);

    ((*env)->ReleaseStringUTFChars)(env, jPassword, password);
    result = sub_88C(output);
    ...
    return result;
}
```

- Function `sub_830` XORs static 128-byte array with constant `"Love"`, turns out it produces string

> Fresh& Canned Tomatoes + Jalapeno + Coriander + Garlic + Pepper + Red Onion + Green Onion + Chili + Sugar + Cumin + Lime + Salt

which looks like one of Salsa receipts!

- Function `sub_85C` shrinks 128-byte array to 32-byte array

  `.rodata:00002B09 byte_2B09 DCB 0xA, 0x1D, 0x13, 0x16, 0x24, 0x49, 0x56, 0x26, 0x2D...`

- Function `sub_B70` initialize array with some suspicious constants (`0x61707865`, `0x3320646E`, `0x79622D32`, `0x6B206574`)

```
int __fastcall sub_B70(int *ctx, _DWORD *a2, _DWORD *a3)
{
    int v4; // [sp+Ch] [bp-Ch]

    *ctx = 0x61707865;
    ctx[1] = *a2;
    ctx[2] = a2[1];
    ctx[3] = a2[2];
    ctx[4] = a2[3];
    ctx[5] = 0x3320646E;
    ctx[6] = *a3;
    ctx[7] = a3[1];
    ctx[8] = 0;
    ctx[9] = 0;
    ctx[10] = 0x79622D32;
    ctx[11] = a2[4];
    ctx[12] = a2[5];
    ctx[13] = a2[6];
    ctx[14] = a2[7];
    ctx[15] = 0x6B206574;
    return _stack_chk_guard - v4;
}
```

Google reveals that it might be Salsa20 cipher. Moreover, we have salsa receipt as a hint.

- Function `sub_C30` XORs blocks of 64-bytes of input with outcome of `sub_8B4` function.
- Function `sub_8B4` indeed contains Salsa20 core quarter-round operations.

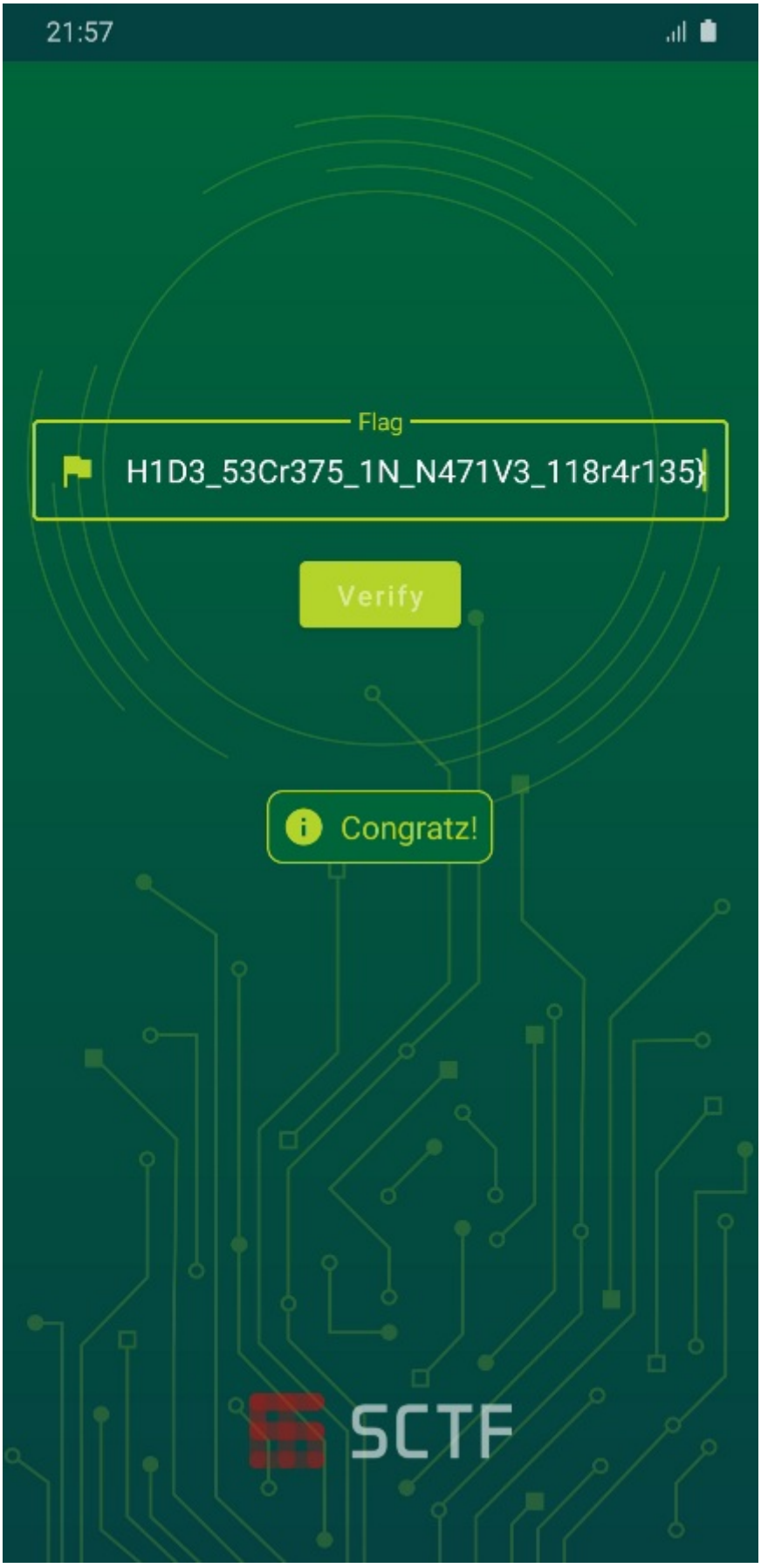
```
int __fastcall sub_8B4(int *input, char *output)
{
    int x[16]; // [sp+1Ch] [bp-4Ch]
    ...

    for ( i = 0; i < 16; ++i ) x[i] = input[i];

    for ( j = 0; j < 20; j += 2 )
    {
        x[4] ^= __ROR4__(x[0] + x[12], 25);
        x[8] ^= __ROR4__(x[0] + x[4], 23);
        x[12] ^= __ROR4__(x[4] + x[8], 19);
        x[0] ^= __ROR4__(x[8] + x[12], 14);
        ...
```

- Function `sub_88C` just compares Salsa20 ciphertext with the hardcoded one.

  `.rodata:00002B89 byte_2B89 DCB 0xE1, 0x21, 0x53, 0x50, 0xA6, 0xDC, 0x93, 0x71, 0x66 ...`

4. PROFIT!

Combine altogether in exploit script.
Flag is `SCTF{D0_N07_H1D3_53Cr375_1N_N471V3_118r4r135}`.


21:57
Flag
H1D3_53Cr375_1N_N471V3_118r4r135}
Verify
Congratz!
SCTF