# Dark Mark (DarkMark.apk)

21PC01 - Abarna S
21PC12 - Gokul D
21PC24 - S R Ashuwanthh


This enchanted application holds six hidden flags, each protected by a different layer of cybersecurity and reverse engineering challenges.

To begin your journey, tap on the logo to enter the game realm.
There, you'll find mystical cards, each representing a unique challenge that tests your wit, courage, and technical skill as the true flag seeker.

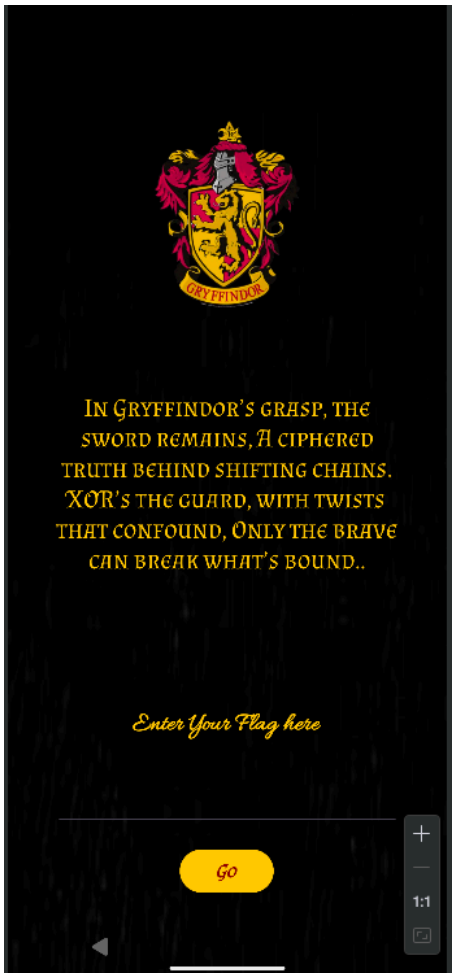
# Challenge Screen

## Gryffindor Challenge

This flag is hidden in the app using XOR encryption and a custom verification function. The participant must reverse engineer the APK using JADX to understand the decryption logic.

**Solution Steps:**

1. Decompile the APK using JADX.
2. Locate the `checkFlag()` function.
3. Identify the encrypted flag and the XOR key.
4. Reconstruct the XOR decryption logic.
5. Input the correct plaintext to match the encrypted result and validate.




**Flag:**
SWORD{F0rg3d_1n_F1r3_Pur1f1ed_By_Br4v3ry}

```
36      private fun checkFlag(input: String): Boolean {
37          val hocrux = intArrayOf(
38              0x44, 0x40, 0x58, 0x45, 0x53, 0x6c, 0x51, 0x27, 0x65, 0x70, 0x24, 0x73, 0x48, 0x26, 0x79,
39              0x48, 0x51, 0x26, 0x65, 0x24, 0x48, 0x47, 0x62, 0x65, 0x26, 0x71, 0x26, 0x72, 0x73, 0x48,
40              0x55, 0x6e, 0x48, 0x55, 0x65, 0x23, 0x61, 0x24, 0x65, 0x6e, 0x6a
41          )
42          val hat = 0x17
43          val decryptedFlag = hocrux.map { (it xor hat).toChar() }.joinToString( separator: "")
44          return input == decryptedFlag
45      }
46  }
```

## Hufflepuff Challenge

This challenge involves AES encryption (CBC mode). The user must provide a base64 cipher text, and the app encrypts a hidden string to compare it.


**Solution Steps:**

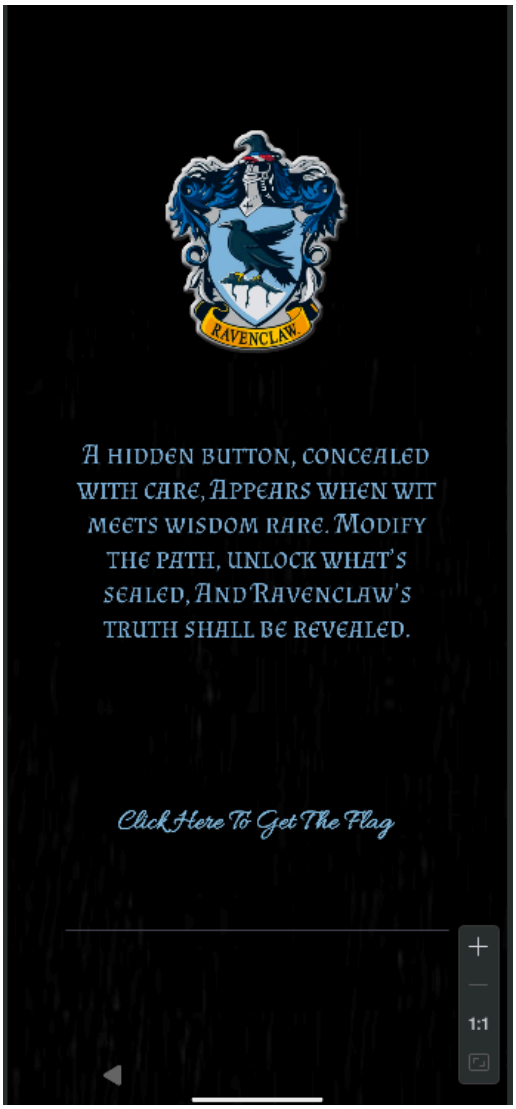
1. Inspect the app resources and layout for hints.
2. Locate the `checkHufflepuffFlag()` function in Kotlin using JADX.
3. AES key is embedded as a resource string (`R.string.cup`) and IV is hardcoded.
4. Use the known key and IV to AES encrypt the known string and produce a cipher text.
5. Submit the correct ciphertext to get validated.

**PlainText:** LoyaltyAndPatience
**Key:** HuflepuffLoyalt
**IV:** BadgerPaws4Puffs
**Flag:**
pAD08N4DvG749/KLWrEc1onuqeLzyTsl1RbNuAZksY4=

```
    private fun encryptHufflepuffFlag(cup: String): String {
        val key = "HufflepuffLoyalt"  // 16-byte AES key
        val iv = "BadgerPaws4Puffs"  // 16-byte IV for AES

        return try {
            // Convert key and IV to byte arrays
            val keyBytes = key.toByteArray(Charsets.UTF_8).copyOf( newSize: 16)
            val ivBytes = iv.toByteArray(Charsets.UTF_8)

            // Initialize AES Cipher
            val secretKey = SecretKeySpec(keyBytes,  algorithm: "AES")
            val ivSpec = IvParameterSpec(ivBytes)
            val cipher = Cipher.getInstance( transformation: "AES/CBC/PKCS5Padding")

            // Encrypt the flag
            cipher.init(Cipher.ENCRYPT_MODE, secretKey, ivSpec)
            val encryptedBytes = cipher.doFinal(cup.toByteArray(Charsets.UTF_8))

            // Return Base64 encoded encrypted flag
            Base64.encodeToString(encryptedBytes, Base64.DEFAULT).trim()
        } catch (e: Exception) {
            e.printStackTrace()
            ""
        }
    }
}
```

## Ravenclaw Challenge

A hidden function derives the flag, but it's never called. Participants must decompile the APK using apktool, modify smali or Kotlin to call the function, recompile the APK, and run it.



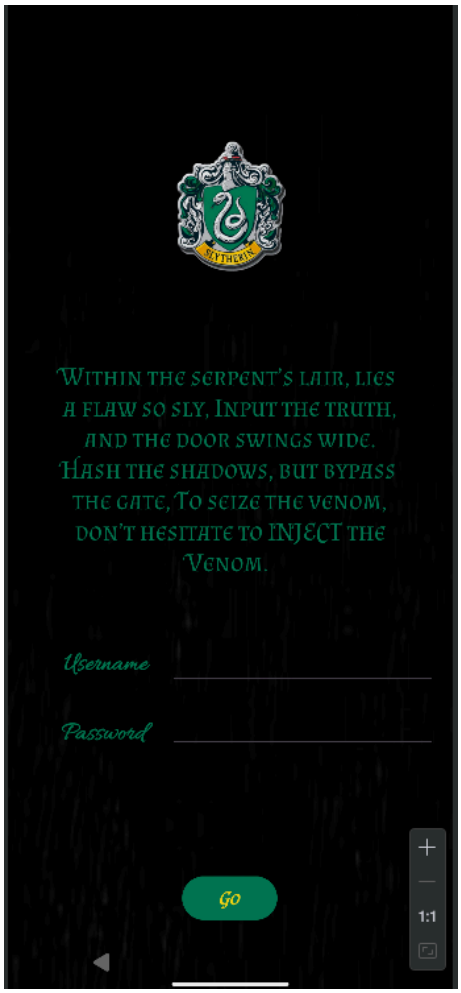

### Solution Steps:

1. Use apktool to decompile the APK.
2. Find the unused function like `hiddenWisdom()`.
3. Patch the source or smali to trigger the function (e.g., on button click).
4. Recompile and resign the APK.
5. Run to retrieve the flag.

```
private fun hiddenWisdom(): String {
    val keyPart1 = "R4v3n"  // First part of the flag
    val keyPart2 = generateKey()  // Dynamically derived part

    val xorResult = xorStrings(keyPart1, keyPart2)
    val flag = "KN0W_TH3_TRU7H_" + xorResult

    runOnUiThread {
        val h3 = findViewById<Button>(R.id.button) as Button
        h3.visibility = View.VISIBLE
        h3.setOnClickListener {
            Toast.makeText( context: this,  text: "Success", Toast.LENGTH_LONG).show()
            val intent = Intent( packageContext: this, MainActivity2::class.java)
            startActivity(intent)
        }
    }
    return flag
}
private fun generateKey(): String {
    val time = System.currentTimeMillis().toString()
    return xorStrings( str1: "cl4w", time.take( n: 4))  // Mixes constant with dynamic value
}
private fun xorStrings(str1: String, str2: String): String {
    val maxLen = Math.max(str1.length, str2.length)
    val xorResult = StringBuilder()

    for (i in 0 ≤  until  < maxLen) {
        val char1 = str1[i % str1.length].code
        val char2 = str2[i % str2.length].code
        xorResult.append((char1 xor char2).toChar())
    }
    return xorResult.toString()
}
```

## Slytherin Challenge

This challenge simulates a classic SQL injection. The user must input a crafted string to bypass the login authentication and access the flag.

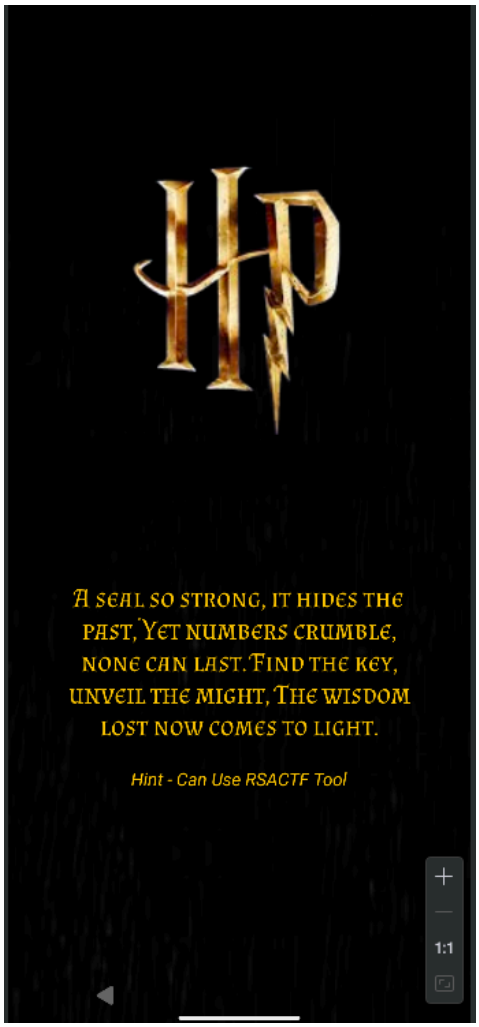



**Solution Steps:**

1. Analyze the login page logic.
2. Find the username from the Vault.kt or form App Inspection.
3. Guess a basic SQL injection payload like:
   `' OR '1'='1`
4. Input it in the password field to bypass authentication.
5. Successful login reveals the Slytherin flag.

```
override fun onCreate(db: SQLiteDatabase) {
    val createTableQuery = """
        CREATE TABLE $TABLE_USERS (
            $COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT,
            $COLUMN_USERNAME TEXT UNIQUE,
            $COLUMN_PASSWORD TEXT
        )
    """
    db.execSQL(createTableQuery)
    insertUser(db, username: "Draco", hashPassword( password: "PureBlood123"))
    insertUser(db, username: "Snape", hashPassword( password: "HalfBloodPrince"))
}
```

## Final Challenge


**Part 1:**

- Open the `AndroidManifest.xml` file.
- Look for a `<meta-data>` tag with the name `"hidden-data"`.
- The value of this field is a **Base64-encoded string**.
- Decode it. This gives you the **first part** of the final flag.
- Enter the decoded string in the challenge activity to unlock the next part.

**Flag:** H3ir_0f_5lyth3in

```
    <meta-data
        android:name="preloaded_fonts"
        android:resource="@array/preloaded_fonts" />
    <meta-data
        android:name="hidden_key"
        android:value="SDNpcl8wZl81bHl0aDNpbg==" />
</application>
```

**Part 2:**

- Once the first part is accepted, you're presented with:
  - An **RSA public key (n, e)**
  - An **RSA-encrypted message (ciphertext)**
    These are embedded in the app's source code (check for unused or obfuscated Kotlin functions).
- Use tools like RsaCtfTool to factor the modulus and retrieve the **private key**.
- Decrypt the ciphertext to obtain the **second part** of the flag.

**Flag:** AncientRunes{Br0k3n_S3als_Reve4l_S3cr3ts}

```
private fun decryptFlag(): String {
    try {
        val encryptedFlag = " q/qWs8WeXHlXd1nLh/wzRLbk99HxT4RIL8E1+bH+2Edd3A1Y5RjvotUQu4uF745spL
        val keyFactory = KeyFactory.getInstance( algorithm: "RSA")
        val publicKeyPEM = """
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEArLrHz23C7RfO8rx/gnaI
2CQBYnoPXLbY9wNwCVwrNYvYDp67ZiHykzjXiESmLqLLpRBSatXlHZIjRX04DG6A
Wg5+0iR2zSDSxcEFP3dDhe1Mw644Y9CWLkkiCCHexeYXH7XWAlOg4EA7yQOT3Tzv
7K9PRHJ1XETAcjvrspTWG6i2zJELBQWNmp8U3xEjJJVKLcKYux1yLu+hgLYottdQ
XnNQtFob5SNXOtYCDepsXjuB7D31g9+mNEm0ygmh2SvZUUt5ZZUdkhE+FOqXjyUh
i14Wr83HhY1EDJReCDdUPO7QbUQZ2dKwvKcYAG5I1vGwhF0KmnWLYSsg4LiGm1sZ
CQIDAQAB
-----END PUBLIC KEY-----

        val keySpec = X509EncodedKeySpec(Base64.decode(publicKeyPEM, Base64.DEFAULT))
        val publicKey = keyFactory.generatePublic(keySpec)

        val cipher = Cipher.getInstance( transformation: "RSA/ECB/OAEPWithSHA-256AndMGF1Padding")
        cipher.init(Cipher.DECRYPT_MODE, publicKey)

        val decryptedBytes = cipher.doFinal(Base64.decode(encryptedFlag, Base64.DEFAULT))
        return String(decryptedBytes)
    } catch (e: Exception) {
        return "The secret remains hidden..."
    }
}
```