1.1.1 Kodda Açık Bir Şekilde Kriptografik Anahtar Kullanılması (Use of Hardcoded Cryptographic Key) (CWE-321)

Açıklık Önem Derecesi: Orta

Açıklığın Etkisi: Hassas Bilgilere Yetkisiz Erişim, Bilgi İfşası, Sızma girişimlerinde saldırının boyutunun artması

Açıklığın Barındıran Dosyalar/Satırlar:

Proje Dosyası/Dosya Adı	Satır Numarası

Açıklığın Açıklaması:

Uygulama kaynak kodları hassas verileri şifreleme ve deşifreleme için şifreleme anahtarı kullanabilirler. Bu ihtiyaçtan dolayı gerekli olabilir. Fakat kaynak koddaki statik ve değişmez şifreleme anahtarları saldırganlarca uygulama kaynak kodlarına veya binary'lerine erişilmesiyle çalınabilir. Saldırganlar şifreleme anahtarına bir kez sahip oldular mı bu anahtar ile şifrelenmiş herhangi gizli bir veriye erişebilirler. Bu ise gizliliğin ihlali anlamına gelir. Bu durumda "Kodda Açık Bir Şekilde Kriptografik Anahtar Kullanılması (CWE-321)" açıklığı vardır denir.

Açıklığı izah eden kod bloklarına yer verilmiştir.

Java - Güvensiz Kod Bloğu:

```java
// Güvensiz Örnek

// Hardcoded Encryption Key

static final byte[] ENCRYPTION_KEY =
"SUPER_secret_ENCRYTION_key".getBytes(StandardCharsets.UTF_8);

public static String encrypt(String value) {
    try {
        SecretKeySpec skeySpec = new SecretKeySpec(ENCRYPTION_KEY, "AES");
        Cipher cipher = Cipher.getInstance(CIPHER_ALGORITHM);
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec,iv);
        byte[] encrypted = cipher.doFinal(value.getBytes());
        return Base64.encodeBase64String(encrypted);
    }
    catch (/* Exceptions */) {
        // Exception handling
    }

    return null;
}
```

Java - Güvenli Kod Bloğu:

```java
// Güvenli Örnek

// Retrieving Encryption Key from KeyStore

public static String encrypt(String plaintext, byte[] ivBytes) {
    try {
        IvParameterSpec iv = new IvParameterSpec(ivBytes);
        SecretKeySpec skeySpec = new SecretKeySpec(
                keyStore.getKey(ENCRYPTION_KEY_ALIAS,
keyStorePassword).getEncoded(),
                "AES");
        Cipher cipher = Cipher.getInstance(CIPHER);
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec, iv);
        byte[] encrypted = cipher.doFinal(plaintext.getBytes());
        return Base64.encodeBase64String(encrypted);
    }
    catch (Exception ex) {
        ex.printStackTrace();
    }
    return null;
}
```

Java güvensiz kod bloğunda kaynak koda açık bir şekilde şifreleme anahtarı gömülmüştür. Bu güvensiz kullanımdır. Java güvenli kod bloğunda ise şifreleme anahtarı harici bir anahtar depolama dosyasından çekilmek suretiyle kullanılmaktadır. Bu ise güvenli kullanımdır.

C# - Güvensiz Kod Bloğu:

```
string hardcodedKey = "EncryptionKey123";
Byte[] dataBytes = Encoding.ASCII.GetBytes(data);
Byte[] cryptoKey = Encoding.ASCII.GetBytes(hardcodedKey);

SymmetricAlgorithm symAlg = Rijndael.Create();

symAlg.Key = cryptoKey;
symAlg.Mode = CipherMode.CBC;
symAlg.Padding = PaddingMode.PKCS7;

MemoryStream ms = new MemoryStream();
CryptoStream cs = new CryptoStream(ms, symAlg.CreateEncryptor(),
CryptoStreamMode.Write);

cs.Write(dataBytes, 0, dataBytes.Length);
cs.Close();

string encodedUnsafe = Encoding.ASCII.GetString(ms.ToArray());
ms.Close();
```

C# güvensiz kod bloğunda yine kaynak kodda açık bir şekilde şifreleme anahtarı gömüldüğü örneklemesi yapılmıştır.

Kurum uygulamada Kodda Açık Bir Şekilde Kriptografik Anahtar Kullanılması (CWE-321) açıklığı tespit edilmiştir.

::::::::BULGU:::::::::

Açıklığın Önlemi:

Açıklığın giderilmesi için tavsiyeler şu şekildedir:

- Şifreleme anahtarları uygulama kaynak kodlarında yer almamalıdır.
- Şifreleme anahtarları harici bir dosyada (örn; konfigürasyon dosyasında, anahtar depolama (keystore) dosyasında) yer almalıdır.
- Şifreleme anahtarları harici dosyalara açık metin (plaintext) olarak depolanmamalıdır.

Referanslar:

1. https://cwe.mitre.org/data/definitions/321.html