New issue

Jump to bottom

Implement KeyStore backed fingerprint verification #897

Merged

hjubb merged 1 commit into `oxen-io:dev` from `hjubb:handle_authenticated_fingerprint` on Jun 2

hjubb commented on May 26 Collaborator

No description provided.

`feat: handle KeyStore backed fingerprint verification` `c69b49e`

hjubb requested a review from **KeeJef** 6 months ago

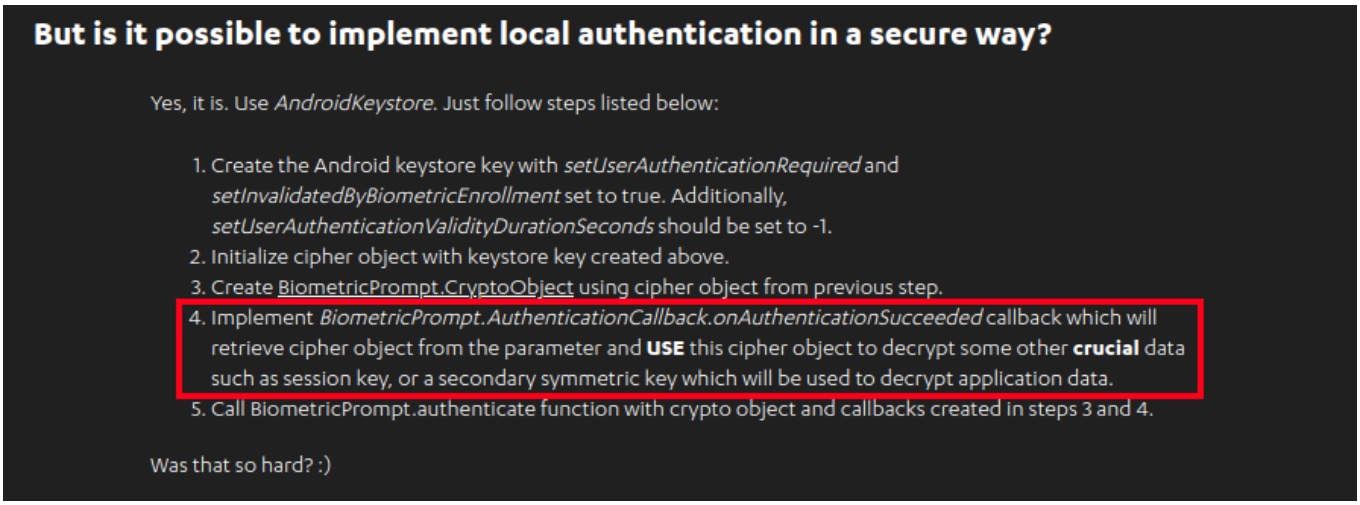
Retr02332 commented on May 30 • edited

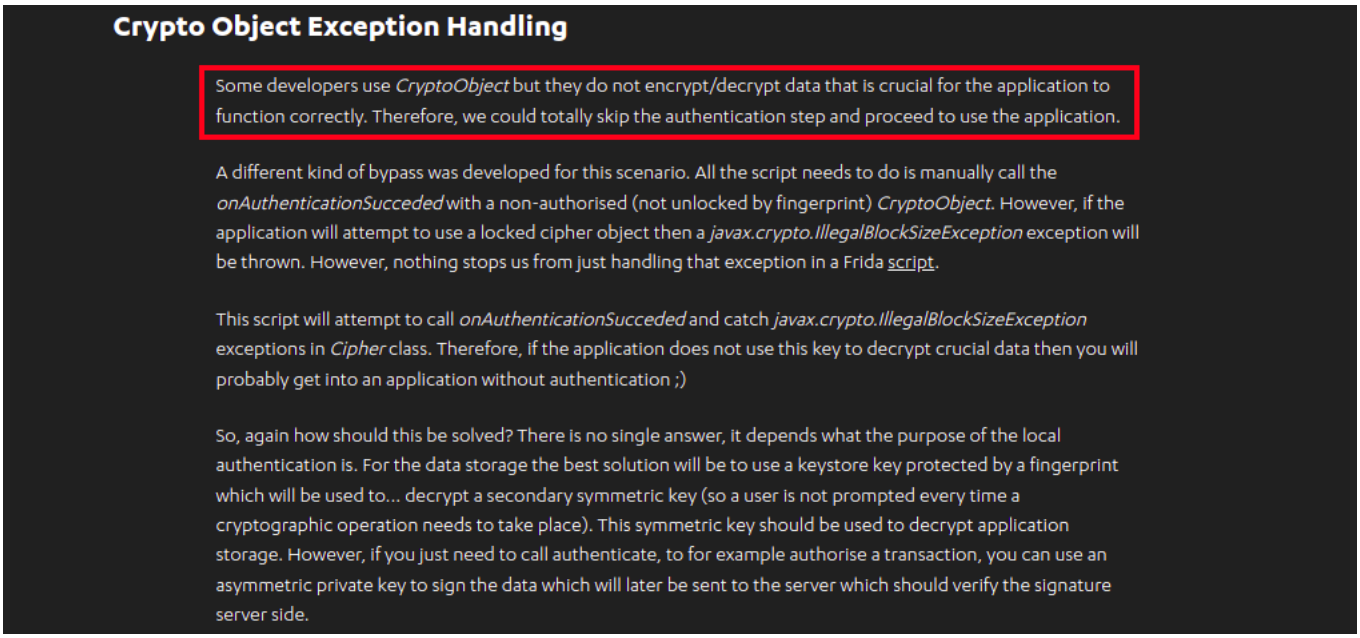
Hello Session team

I will test your solution once implemented to verify that everything works correctly.

However, doing an analysis of the code, it seems to me that you should take into account step 4, which I will show you below:

Link: https://labs.f-secure.com/blog/how-secure-is-your-android-keystore-authentication





I mention this, because it seems that the onAuthenticationSucceeded method still depends on a boolean.

If the cryptographic verification works well, it returns true. However, the correct thing would be that it will 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 do not mean the private key of the session app. 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.

In this way, an attacker could no longer simply hook into the process so that the function returns true, because now we are dealing with cryptographically sound processes, which if we do not have the corresponding key to decrypt the data, then we will not be able to enter the app.

If you want some guidance with this, I invite you to see the following project, which is an app that implements biometric authentication in a secure way: https://github.com/FSecureLABS/android-keystore-audit/tree/master/keystorecrypto-app

Finally, you can check this link where it is explained in even more detail how to solve this problem so that the fingerprint protections cannot be bypassed definitively: https://mobile-security.gitbook.io/mobile-security-testing-guide/android-testing-guide/0x05f-testing-local-authentication#biometric-library

Biometric Library

Android provides a library called Biometric which offers a compatibility version of the `BiometricPrompt` and `BiometricManager` APIs, as implemented in Android 10, with full feature support back to Android 6.0 (API 23).

You can find a reference implementation and instructions on how to show a biometric authentication dialog in the Android developer documentation.

There are two `authenticate` methods available in the `BiometricPrompt` class. One of them expects a `CryptoObject`, which adds an additional layer of security for the biometric authentication.

The authentication flow would be as follows when using CryptoObject:

- The app creates a key in the KeyStore with `setUserAuthenticationRequired` and `setInvalidatedByBiometricEnrollment` set to true. Additionally, `setUserAuthenticationValidityDurationSeconds` should be set to -1.
- This key is used to encrypt information that is authenticating the user (e.g. session information or authentication token).
- A valid set of biometrics must be presented before the key is released from the KeyStore to decrypt the data, which is validated through the `authenticate` method and the `CryptoObject`.
- This solution cannot be bypassed, even on rooted devices, as the key from the KeyStore can only be used after successful biometric authentication.

If `CryptoObject` is not used as part of the authenticate method, it can be bypassed by using Frida. See the "Dynamic Instrumentation" section for more details.

Developers can use several validation classes offered by Android to test the implementation of biometric authentication in their app.

Remember not to reuse keys and not to leave them exposed in RAM for a long time !!

Regards, **@Retr02332**

hjubb commented on May 31 Collaborator Author

@Retr02332

I think the issue presented with that approach is that it would require refactoring the application's keystore which provides the ability to decrypt the key information for the user's profile to depend on the biometric-backed key. In the event that a user entrolls a new fingerprint, or turns off biometric security on their device, this key would no longer work and lock the user out of Session, requiring them to restore their account again. Currently the refactored AuthenticationCallback uses a signature and certificate mechanism, which generates a signature from the CryptoObject parameter (unlocked after biometric authentication), then verifies that signing random data with that key matches what is expected, otherwise preventing someone from opening the application. It prevents the original approach to attack the process returning a successful authentication without requiring a major refactor or depending on a key which may be erased external to the application process for crucial data.

Does this make sense to you and sync up with your understanding?

Retr02332 commented on May 31 • edited

Hi **@hjubb**

You may perform cryptographic procedures with the cryptobject object. However, the proposed check results in a boolean:

```
fun verifySignature(data: ByteArray, signedData: ByteArray): Boolean {
    val ks = KeyStore.getInstance(ANDROID_KEYSTORE)
    ks.load(null)
    val certificate = ks.getCertificate(BIOMETRIC_ASYM_KEY_ALIAS)
    val signature = Signature.getInstance(SIGNATURE_ALGORITHM)
    signature.initVerify(certificate)
    signature.update(data)
    return signature.verify(signedData)
}
```

With this, nothing would prevent an attacker from hooking into the process so that this function always returns true. This is why in this OWASP document we can see the following statement:

Link: https://mobile-security.gitbook.io/mobile-security-testing-guide/android-testing-guide/0x05f-testing-local-authentication

Local Authentication on Android

During local authentication, an app authenticates the user against credentials stored locally on the device. In other words, the user "unlocks" the app or some inner layer of functionality by providing a valid PIN, password or biometric characteristics such as face or fingerprint, which is verified by referencing local data. Generally, this is done so that users can more conveniently resume an existing session with a remote service or as a means of step-up authentication to protect some critical function.

As stated before in chapter "Mobile App Authentication Architectures": The tester should be aware that local authentication should always be enforced at a remote endpoint or based on a cryptographic primitive. Attackers can easily bypass local authentication if no data returns from the authentication process.

In Android, there are two mechanisms supported by the Android Runtime for local authentication: the Confirm Credential flow and the Biometric Authentication flow.

It is necessary to adopt the approach I have mentioned to avoid bypassing biometric authentication, even on rooted devices. This is because if the approach I mentioned to you is adopted, an attacker no matter how much he hooks into the process, would not be able to bypass authentication, since for the user's session to be unlocked a symmetric key must first be recovered with the help of the cryptobject after a successful biometric authentication.

As you see, it is no longer a matter of reimplementing a function in memory so that it returns true forever (the bypass), but we are now facing solid cryptographic process that if we do not have the keys that will help us decrypt the information needed to continue (keys that will be obtained after a successful biometric authentication with the help of the cryptobject), then we will not be able to enter the application.

Finally, you tell me that "it would require refactoring the application keystore that provides the ability to decrypt the user's profile key information to trust the biometric-backed key". This is true, but only if the user has configured biometric fingerprint authentication as the system protection method.

Regards, **@Retr02332**

Retr02332 commented on May 31 • edited

Hi **@hjubb**

We have assigned the CVE-2022-1955 to refer to the issue. The details of the CVE will be published once the corresponding patch has been implemented and verified.

Disclosure policy: https://fluidattacks.com/advisories/policy/

Regards, **@Retr02332**

hjubb commented on May 31 Collaborator Author

Hi **@Retr02332** could you please confirm if this patch still applies to your original issue

Retr02332 commented on May 31 • edited

@hjubb The original problem was solved. That is, now a Cryptobject is used, so the validation of the fingerprint no longer depends on the result of the event handler.

However, please note the recommendations given to improve the patch.

hjubb commented on May 31 Collaborator Author

Ok thanks for that info

hjubb commented on May 31 Collaborator Author

Regarding the further recommendations, I am currently trying to understand in which context the signature verification boolean return statement - after using the crypto object which is only unlocked after biometric authentication by the KeyStore - would be an attack surface. I believe the `CryptoObject` returned via the biometric authentication callback will throw a security exception if it is used in a locked state before this check is performed, and I would assume the threat model of an unlocked device (is a rooted device required?) would have different opportunities to circumvent even the requirement of a fingerprint in the first place. Please let me know if I'm on the right track here or if I'm missing something and if you could elaborate on the requirements of the state of the device and OS (root, physically unlocked etc) to perform such a procedure.

Retr02332 commented on May 31

@hjubb

I understand. I would like to explain all these points in a practical way. So could you approve this patch that you have made and then I could test the patch in a practical way.

In case I find the bypass in the way I raised you above, then I will send you a mail as I did with this finding.

Regards.

👍 1

 **hjubb** merged commit `db92034` into `oxen-io:dev` on Jun 2

Retr02332 commented on Jun 6

Hi **@hjubb**

I just sent an email with the following subject line: **"Bypass of the biometric fingerprint authentication patch"**

As I had told you before, once I had the functional exploit I would send you a poc so you could see in detail what I was referring to. In the email are all the complementary details you need.

Regards, **@Retr02332**

👍 1

Retr02332 commented on Jun 19

Hi **@hjubb**

Any update?

---

hjubb commented on Jun 21 Collaborator Author

Hey **@Retr02332** at the moment the further points you have raised aren't practical to implement from an application perspective. It is not really practical to take into consideration an attack relying on full root control of an unlocked device which has been configured with appropriate screen locks or to completely re-architect the application in a way that would rely on such defences which would leave a user's application unusable by turning off or adding additional fingerprints on a device from the settings menu forcing them to re-sync their account. If there are further practical implementations we can look at discussing them and implementing them further but for now I think it is sufficient 👍

👍 1

Retr02332 commented on Jun 29

Hi **@hjubb**

In compliance with our disclosure policy: https://fluidattacks.com/advisories/policy/ previously shared with you. We proceed to make responsible disclosure of the finding: https://fluidattacks.com/advisories/tempest/.

Regards,
@Retr02332

👍 1

hjubb commented on Jun 29 Collaborator Author

Thanks **@Retr02332**, it might be helpful in describing the context and state of the devices, I can only see the OS level, not any indication of whether the device was rooted as you had previously mentioned.

👍 1

Retr02332 commented on Jun 30 • edited

Hi **@hjubb**

Thanks for the feedback, in a few moments I will update the advisory to include those details.

Regards.

👍 1

Reviewers

 **KeeJef**

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Development

Successfully merging this pull request may close these issues.

None yet

2 participants