Atualizar seu provedor de segurança para proteger-se contra explorações de SSL

O Android depende de um [Provider](https://developer.android.com/reference/java/security/Provider.html) de segurança para oferecer comunicações de rede seguras. No entanto, de tempos em tempos, são encontradas vulnerabilidades no provedor de segurança padrão. Para proteger-se contra essas vulnerabilidades, o [Google Play Services](https://developer.android.com/google/play-services/) oferece uma maneira de atualizar automaticamente o provedor de segurança de um dispositivo, proporcionando uma proteção contra explorações conhecidas. Ao chamar os métodos do Google Play Services, sei aplicativo pode garantir que ele seja executado em um dispositivo que tenha as últimas atualizações para protegê-lo contra explorações conhecidas.

Por exemplo, foi descoberta uma vulnerabilidade no OpenSSL ([CVE-2014-0224](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-0224)) que pode deixar aplicativos abertos a um ataque de interceptação que descriptografa tráfego seguro sem o conhecimento de qualquer um dos lados. Com o Google Play Services versão 5.0, uma correção é disponibilizada, mas os aplicativos devem garantir que ela seja instalada. Ao usar os métodos do Google Play Services, seu aplicativo pode garantir que ele seja executado em um dispositivo protegido contra esse ataque.

**Atenção:**Atualizar o [**Provider**](https://developer.android.com/reference/java/security/Provider.html) de segurança de um dispositivo *não* atualiza o [**android.net.SSLCertificateSocketFactory**](https://developer.android.com/reference/android/net/SSLCertificateSocketFactory.html). Em vez de usar essa classe, incentivamos os desenvolvedores de aplicativos a usar métodos de alto nível para interagir com criptografia. A maioria dos aplicativos usam APIs como a [**HttpsURLConnection**](https://developer.android.com/reference/javax/net/ssl/HttpsURLConnection.html) sem precisar definir um [**TrustManager**](https://developer.android.com/reference/javax/net/ssl/TrustManager.html) personalizado ou criar um [**SSLCertificateSocketFactory**](https://developer.android.com/reference/android/net/SSLCertificateSocketFactory.html).

Aplicação de patch no provedor de segurança com ProviderInstaller

Para atualizar o provedor de segurança de um dispositivo, use a classe [ProviderInstaller](https://developer.android.com/reference/com/google/android/gms/security/ProviderInstaller.html). Você pode verificar se o provedor de segurança está atualizado (e, se necessário, atualizá-lo) chamando o método [installIfNeeded()](https://developer.android.com/reference/com/google/android/gms/security/ProviderInstaller.html##installIfNeeded(android.content.Context)) (ou [installIfNeededAsync()](https://developer.android.com/reference/com/google/android/gms/security/ProviderInstaller.html#installIfNeededAsync(android.content.Context,%20com.google.android.gms.security.ProviderInstaller.ProviderInstallListener))) dessa classe.

Quando você chama [installIfNeeded()](https://developer.android.com/reference/com/google/android/gms/security/ProviderInstaller.html##installIfNeeded(android.content.Context)), o [ProviderInstaller](https://developer.android.com/reference/com/google/android/gms/security/ProviderInstaller.html) faz o seguinte:

* Se o [Provider](https://developer.android.com/reference/java/security/Provider.html) do dispositivo for atualizado com êxito (ou se ele já estiver atualizado), o método será retornado normalmente.
* Se a biblioteca do Google Play Services do dispositivo estiver desatualizada, o método acionará [GooglePlayServicesRepairableException](https://developer.android.com/reference/com/google/android/gms/common/GooglePlayServicesRepairableException.html). O aplicativo pode capturar essa exceção e mostrar uma caixa de diálogo apropriada ao usuário para atualizar o Google Play Services.
* Se um erro não recuperável ocorrer, o método acionará [GooglePlayServicesNotAvailableException](https://developer.android.com/reference/com/google/android/gms/common/GooglePlayServicesNotAvailableException.html)para indicar que não é possível atualizar o [Provider](https://developer.android.com/reference/java/security/Provider.html). Em seguida, o aplicativo pode capturar a exceção e escolher uma ação apropriada, como exibir o [diagrama padrão de fluxo de correção](https://developer.android.com/reference/com/google/android/gms/common/SupportErrorDialogFragment.html).

O método [installIfNeededAsync()](https://developer.android.com/reference/com/google/android/gms/security/ProviderInstaller.html#installIfNeededAsync(android.content.Context,%20com.google.android.gms.security.ProviderInstaller.ProviderInstallListener)) se comporta de forma semelhante, mas, em vez de acionar exceções, ele chama o método de retorno de chamada apropriado para indicar um sucesso ou uma falha.

Se [installIfNeeded()](https://developer.android.com/reference/com/google/android/gms/security/ProviderInstaller.html##installIfNeeded(android.content.Context)) precisar instalar um novo [Provider](https://developer.android.com/reference/java/security/Provider.html), esse processo pode durar de 30-50 ms (em dispositivos mais recentes) a 350 ms (em dispositivos mais antigos). Se o provedor de segurança já estiver atualizado, a duração do método é mínima. Para evitar impactos na experiência do usuário:

* Chame [installIfNeeded()](https://developer.android.com/reference/com/google/android/gms/security/ProviderInstaller.html##installIfNeeded(android.content.Context)) de encadeamentos de rede em segundo plano imediatamente quando os encadeamentos forem carregados, em vez de esperar que o encadeamento tente usar a rede. (Não há problema em chamar o método várias vezes, pois ele é retornado imediatamente se o provedor de segurança não precisar ser atualizado.)
* Se a experiência do usuário for afetada pelo bloqueio de threads (por exemplo, se a chamada for originada de uma atividade no thread da IU), chame a versão assíncrona do método, [installIfNeededAsync()](https://developer.android.com/reference/com/google/android/gms/security/ProviderInstaller.html#installIfNeededAsync(android.content.Context,%20com.google.android.gms.security.ProviderInstaller.ProviderInstallListener)). (Naturalmente, se você fizer isso, será preciso esperar a conclusão da operação antes de tentar realizar comunicações seguras. O [ProviderInstaller](https://developer.android.com/reference/com/google/android/gms/security/ProviderInstaller.html) chama o método [onProviderInstalled()](https://developer.android.com/reference/com/google/android/gms/security/ProviderInstaller.ProviderInstallListener.html#onProviderInstalled()) do seu ouvinte para indicar o sucesso.)

**Aviso:** Se o [**ProviderInstaller**](https://developer.android.com/reference/com/google/android/gms/security/ProviderInstaller.html) não conseguir instalar um [**Provider**](https://developer.android.com/reference/java/security/Provider.html) atualizado, o provedor de segurança do seu dispositivo poderá estar vulnerável a explorações conhecidas. Seu aplicativo se comportará como se todas as comunicações HTTP estivessem descriptografadas.

Quando o [Provider](https://developer.android.com/reference/java/security/Provider.html) for atualizado, todas as chamadas para APIs de segurança (incluindo APIs SSL) serão roteadas por ele. (No entanto, isso não se aplica ao [android.net.SSLCertificateSocketFactory](https://developer.android.com/reference/android/net/SSLCertificateSocketFactory.html), que permanece vulnerável a explorações como a [CVE-2014-0224](http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-0224).)

Aplicação de patch síncrona

A maneira mais simples de aplicar um patch no provedor de segurança é chamar o método síncrono [installIfNeeded()](https://developer.android.com/reference/com/google/android/gms/security/ProviderInstaller.html##installIfNeeded(android.content.Context)). Essa ação é apropriada se a experiência do usuário não for afetada pelo bloqueio de threads enquanto a operação não é concluída.

Por exemplo, veja aqui uma implementação de um [adaptador de sincronização](https://developer.android.com/training/sync-adapters) que atualiza o provedor de segurança. Como um adaptador de sincronização é executado em segundo plano, não tem problema se o encadeamento for bloqueado enquanto a atualização do provedor de segurança é realizada. O adaptador de sincronização chama [installIfNeeded()](https://developer.android.com/reference/com/google/android/gms/security/ProviderInstaller.html##installIfNeeded(android.content.Context)) para atualizar o provedor de segurança. Se o método for retornado normalmente, o adaptador de sincronização saberá que o provedor de segurança está atualizado. Se o método acionar uma exceção, o adaptador de sincronização poderá realizar a ação apropriada (como solicitar que o usuário atualize o Google Play Services).

/\*\*  
 \* Sample sync adapter using {@link ProviderInstaller}.  
 \*/  
public class SyncAdapter extends AbstractThreadedSyncAdapter {  
  
  ...  
  
  // This is called each time a sync is attempted; this is okay, since the  
  // overhead is negligible if the security provider is up-to-date.  
  @Override  
  public void onPerformSync(Account account, Bundle extras, String authority,  
      ContentProviderClient provider, SyncResult syncResult) {  
    try {  
      ProviderInstaller.installIfNeeded(getContext());  
    } catch (GooglePlayServicesRepairableException e) {  
  
      // Indicates that Google Play services is out of date, disabled, etc.  
  
      // Prompt the user to install/update/enable Google Play services.  
      GooglePlayServicesUtil.showErrorNotification(  
          e.getConnectionStatusCode(), getContext());  
  
      // Notify the SyncManager that a soft error occurred.  
      syncResult.stats.numIOExceptions++;  
      return;  
  
    } catch (GooglePlayServicesNotAvailableException e) {  
      // Indicates a non-recoverable error; the ProviderInstaller is not able  
      // to install an up-to-date Provider.  
  
      // Notify the SyncManager that a hard error occurred.  
      syncResult.stats.numAuthExceptions++;  
      return;  
    }  
  
    // If this is reached, you know that the provider was already up-to-date,  
    // or was successfully updated.  
  }  
}

Aplicação de patch assíncrona

A atualização do provedor de segurança pode levar até 350 milissegundos (em dispositivos mais antigos). Se você estiver realizando a atualização em um encadeamento que afete diretamente a experiência do usuário, como o encadeamento de IU, não é recomendável realizar uma chamada síncrona para atualizar o provedor, pois isso poderá fazer com que o aplicativo ou o dispositivo congele até que a operação seja concluída. Em vez disso, use o método assíncrono, [installIfNeededAsync()](https://developer.android.com/reference/com/google/android/gms/security/ProviderInstaller.html#installIfNeededAsync(android.content.Context,%20com.google.android.gms.security.ProviderInstaller.ProviderInstallListener)). Esse método indica seu sucesso ou falha chamando callbacks.

Por exemplo, apresentamos aqui um código que atualiza o provedor de segurança em uma atividade no encadeamento da IU. A atividade chama [installIfNeededAsync()](https://developer.android.com/reference/com/google/android/gms/security/ProviderInstaller.html#installIfNeededAsync(android.content.Context,%20com.google.android.gms.security.ProviderInstaller.ProviderInstallListener)) para atualizar o provedor e se designa como ouvinte para receber notificações de sucesso ou falha. Se o provedor de segurança estiver atualizado ou se for atualizado com êxito, o método [onProviderInstalled()](https://developer.android.com/reference/com/google/android/gms/security/ProviderInstaller.ProviderInstallListener.html#onProviderInstalled()) da atividade será chamado e a atividade saberá que as comunicações estão seguras. Se não for possível atualizar o provedor, o método [onProviderInstallFailed()](https://developer.android.com/reference/com/google/android/gms/security/ProviderInstaller.ProviderInstallListener.html#onProviderInstallFailed(int,%20android.content.Intent)) da atividade será chamado e a atividade poderá realizar a ação apropriada (como solicitar que o usuário atualize o Google Play Services).

/\*\*  
 \* Sample activity using {@link ProviderInstaller}.  
 \*/  
public class MainActivity extends Activity  
    implements ProviderInstaller.ProviderInstallListener {  
  
  private static final int ERROR\_DIALOG\_REQUEST\_CODE = 1;  
  
  private boolean mRetryProviderInstall;  
  
  //Update the security provider when the activity is created.  
  @Override  
  protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    ProviderInstaller.installIfNeededAsync(this, this);  
  }  
  
  /\*\*  
   \* This method is only called if the provider is successfully updated  
   \* (or is already up-to-date).  
   \*/  
  @Override  
  protected void onProviderInstalled() {  
    // Provider is up-to-date, app can make secure network calls.  
  }  
  
  /\*\*  
   \* This method is called if updating fails; the error code indicates  
   \* whether the error is recoverable.  
   \*/  
  @Override  
  protected void onProviderInstallFailed(int errorCode, Intent recoveryIntent) {  
    if (GooglePlayServicesUtil.isUserRecoverableError(errorCode)) {  
      // Recoverable error. Show a dialog prompting the user to  
      // install/update/enable Google Play services.  
      GooglePlayServicesUtil.showErrorDialogFragment(  
          errorCode,  
          this,  
          ERROR\_DIALOG\_REQUEST\_CODE,  
          new DialogInterface.OnCancelListener() {  
            @Override  
            public void onCancel(DialogInterface dialog) {  
              // The user chose not to take the recovery action  
              onProviderInstallerNotAvailable();  
            }  
          });  
    } else {  
      // Google Play services is not available.  
      onProviderInstallerNotAvailable();  
    }  
  }  
  
  @Override  
  protected void onActivityResult(int requestCode, int resultCode,  
      Intent data) {  
    super.onActivityResult(requestCode, resultCode, data);  
    if (requestCode == ERROR\_DIALOG\_REQUEST\_CODE) {  
      // Adding a fragment via GooglePlayServicesUtil.showErrorDialogFragment  
      // before the instance state is restored throws an error. So instead,  
      // set a flag here, which will cause the fragment to delay until  
      // onPostResume.  
      mRetryProviderInstall = true;  
    }  
  }  
  
  /\*\*  
   \* On resume, check to see if we flagged that we need to reinstall the  
   \* provider.  
   \*/  
  @Override  
  protected void onPostResume() {  
    super.onPostResult();  
    if (mRetryProviderInstall) {  
      // We can now safely retry installation.  
      ProviderInstall.installIfNeededAsync(this, this);  
    }  
    mRetryProviderInstall = false;  
  }  
  
  private void onProviderInstallerNotAvailable() {  
    // This is reached if the provider cannot be updated for some reason.  
    // App should consider all HTTP communication to be vulnerable, and take  
    // appropriate action.  
  }  
}