Configurações de segurança de rede

O recurso de configurações de segurança de rede permite que os aplicativos personalizem as configurações de segurança de rede em um arquivo de configurações declarativo e seguro sem modificar o código do aplicativo. Essas configurações podem ser definidas para domínios específicos e para um aplicativo específico. Os principais recursos são:

* **Âncoras de confiança personalizadas:** Personalize quais autoridades de certificado (CA) são confiáveis para as conexões seguras de um aplicativo. Por exemplo, confiar em certificados autoassinados privados ou restringir o conjunto de CAs públicas nas quais o aplicativo confia.
* **Substituições somente de depuração:** Depure conexões seguras do aplicativo com segurança, sem adicionar riscos à base instalada.
* **Cancelar uso de tráfego de texto simples:** Proteja aplicativos contra o uso acidental de tráfego de texto simples.
* **Fixar certificados:** Restrinja a conexão segura de um aplicativo a certificados específicos.

Adicionar um arquivo de configurações de segurança

O recurso de configurações de segurança de rede usa um arquivo XML no qual você especifica as configurações do seu aplicativo. Inclua uma entrada no manifesto do seu aplicativo para apontar para esse arquivo. Este trecho de código de um manifesto demonstra como criar essa entrada:

<?xml version="1.0" encoding="utf-8"?>  
<manifest ... >  
    <application android:networkSecurityConfig="@xml/network\_security\_config"  
                    ... >  
        ...  
    </application>  
</manifest>

Personalizar CAs confiáveis

Um aplicativo pode querer confiar em um conjunto personalizado de CAs em vez de no padrão da plataforma. Os motivos mais comuns para isso são:

* Conectar-se a um host com uma autoridade de certificação personalizada, como uma CA autoassinada ou emitida internamente em uma empresa.
* Limitar o conjunto de CAs para apenas aquelas nas quais você confia em vez de todas as CAs pré-instaladas.
* Confiar em CAs adicionais não incluídas no sistema.

Por padrão, conexões seguras (usando protocolos como TLS e HTTPS) de todos os aplicativos confiam nas CAs pré-instaladas do sistema e os aplicativos destinados ao Android 6.0 (nível de API 23) e inferiores também confiam no repositório de CAs adicionadas pelo usuário por padrão. Um aplicativo pode personalizar as próprias conexões usando base-config (para personalização em todo o aplicativo) ou domain-config (para personalização por domínio).

Configurar uma CA personalizada

Suponhamos que você queira se conectar a um host que use um certificado SSL autoassinado ou a um host cujo certificado SSL foi emitido por uma CA não pública na qual confia, como a CA interna da sua empresa.

res/xml/network\_security\_config.xml:

<?xml version="1.0" encoding="utf-8"?>  
<network-security-config>  
    <domain-config>  
        <domain includeSubdomains="true">example.com</domain>  
        <trust-anchors>  
            <certificates src="@raw/my\_ca"/>  
        </trust-anchors>  
    </domain-config>  
</network-security-config>

Adicione o certificado da CA autoassinada ou não pública em formato PEM ou DER em res/raw/my\_ca.

Limitar o conjunto de CAs confiáveis

Um aplicativo que não queira confiar em todas as CAs nas quais o sistema confia pode especificar o próprio conjunto limitado de CAs confiáveis. Isso protege o aplicativo contra certificados fraudulentos emitidos por qualquer outra CA.

A configuração para limitar o conjunto de CAs confiáveis é semelhante a [confiar em uma CA personalizada](https://developer.android.com/training/articles/security-config#ConfigCustom) para um domínio específico, mas fornecendo várias CAs no recurso.

res/xml/network\_security\_config.xml:

<?xml version="1.0" encoding="utf-8"?>  
<network-security-config>  
    <domain-config>  
        <domain includeSubdomains="true">secure.example.com</domain>  
        <domain includeSubdomains="true">cdn.example.com</domain>  
        <trust-anchors>  
            <certificates src="@raw/trusted\_roots"/>  
        </trust-anchors>  
    </domain-config>  
</network-security-config>

Adicione as CAs confiáveis em formato PEM ou DER em res/raw/trusted\_roots. Observe que, ao usar o formato PEM, o arquivo deve conter *somente* dados PEM, sem texto adicional. Você também pode fornecer vários elementos [<certificates>](https://developer.android.com/training/articles/security-config#certificates) em vez de um.

Confiar em CAs adicionais

Um aplicativo pode querer confiar em CAs adicionais nas quais o sistema não confia. Isso pode ocorrer se o sistema ainda não incluiu a CA ou se a CA não atender aos requisitos de inclusão no sistema Android. O aplicativo pode fazer isso ao especificar várias fontes de certificados para uma configuração.

res/xml/network\_security\_config.xml:

<?xml version="1.0" encoding="utf-8"?>  
<network-security-config>  
    <base-config>  
        <trust-anchors>  
            <certificates src="@raw/extracas"/>  
            <certificates src="system"/>  
        </trust-anchors>  
    </base-config>  
</network-security-config>

Configurar CAs para depuração

Ao depurar um aplicativo conectado por HTTPS, você pode querer se conectar a um servidor de desenvolvimento local que não tenha o certificado SSL do seu servidor de produção. Para fazer isso sem modificar o código do aplicativo, você pode especificar CAs somente de depuração que sejam confiáveis *comente* quando[android:debuggable](https://developer.android.com/guide/topics/manifest/application-element.html#debug) for true, usando debug-overrides. Normalmente, IDEs e ferramentas de compilação definem esse sinalizador automaticamente para compilações de não lançamento.

Isso é mais seguro do que o código condicional normal, pois, como medida de segurança, os repositórios do aplicativo não aceitam aplicativos marcados como depuráveis.

res/xml/network\_security\_config.xml:

<?xml version="1.0" encoding="utf-8"?>  
<network-security-config>  
    <debug-overrides>  
        <trust-anchors>  
            <certificates src="@raw/debug\_cas"/>  
        </trust-anchors>  
    </debug-overrides>  
</network-security-config>

Cancelar uso de tráfego de texto simples

Aplicativos que pretendem se conectar a destinos usando apenas conexões seguras podem cancelar o uso de texto simples de suporte (usando o protocolo HTTP não criptografado em vez de HTTPS) para esses destinos. Essa opção ajuda a evitar regressões acidentais em aplicativos devido a alterações nos URLs fornecidos por fontes externas, como servidores de back-end. Consulte [NetworkSecurityPolicy.isCleartextTrafficPermitted()](https://developer.android.com/reference/android/security/NetworkSecurityPolicy.html#isCleartextTrafficPermitted()) para saber mais.

Por exemplo, um aplicativo pode querer garantir que todas as conexões com secure.example.com sejam sempre realizadas por HTTPS para proteger o tráfego confidencial de redes hostis.

res/xml/network\_security\_config.xml:

<?xml version="1.0" encoding="utf-8"?>  
<network-security-config>  
    <domain-config cleartextTrafficPermitted="false">  
        <domain includeSubdomains="true">secure.example.com</domain>  
    </domain-config>  
</network-security-config>

Fixar certificados

Normalmente, um aplicativo confia em todas as CAs pré-instaladas. Se qualquer uma dessas CAs emitir um certificado fraudulento, o aplicativo estará em risco de ataques de interceptação. Alguns aplicativos optam por limitar o conjunto de certificados que aceitam restringindo o conjunto de CAs ou fixando certificados.

A fixação de certificados é realizada ao fornecer um conjunto de certificados pelo hash da chave pública (SubjectPublicKeyInfo do certificado X.509). Uma cadeia de certificados é válida somente se contiver pelo menos uma das chaves públicas fixadas.

Observe que, ao usar a fixação de certificados, você deve sempre incluir uma chave de backup para que, se você for forçado a alternar para novas chaves ou alterar as CAs (ao fixar um certificado de CA ou um intermediário dessa CA), a conectividade do seu aplicativo não seja afetada. Caso contrário, você precisará enviar uma atualização ao aplicativo para restaurar a conectividade.

Além disso, é possível definir um tempo de expiração para as fixações, após o qual elas não sejam mais realizadas. Isso ajuda a evitar problemas de conectividade em aplicativos que não foram atualizados. No entanto, definir um tempo de expiração para fixações pode permitir que as fixações sejam ignoradas.

res/xml/network\_security\_config.xml:

<?xml version="1.0" encoding="utf-8"?>  
<network-security-config>  
    <domain-config>  
        <domain includeSubdomains="true">example.com</domain>  
        <pin-set expiration="2018-01-01">  
            <pin digest="SHA-256">7HIpactkIAq2Y49orFOOQKurWxmmSFZhBCoQYcRhJ3Y=</pin>  
            <!-- backup pin -->  
            <pin digest="SHA-256">fwza0LRMXouZHRC8Ei+4PyuldPDcf3UKgO/04cDM1oE=</pin>  
        </pin-set>  
    </domain-config>  
</network-security-config>

Comportamento de herança de configuração

Valores não definidos em uma configuração específica são herdados. Esse comportamento permite configurações mais complexas, mantendo o arquivo de configuração legível.

Se um valor não for definido em uma entrada específica, o valor da entrada mais genérica será usado. Por exemplo, valores não definidos em um domain-config são obtidos pelo domain-config pai, se aninhados, ou, caso contrário, pelo base-config. Valores não definidos no base-config usam os valores padrão da plataforma.

Por exemplo, considere um caso no qual todas as conexões para subdomínios de example.com devem usar um conjunto personalizado de CAs. Além disso, o tráfego de texto simples para esses domínios é permitido *exceto*ao se conectar com secure.example.com. Ao aninhar a configuração para secure.example.com dentro da configuração para example.com, o trust-anchors não precisa ser duplicado.

res/xml/network\_security\_config.xml:

<?xml version="1.0" encoding="utf-8"?>  
<network-security-config>  
    <domain-config>  
        <domain includeSubdomains="true">example.com</domain>  
        <trust-anchors>  
            <certificates src="@raw/my\_ca"/>  
        </trust-anchors>  
        <domain-config cleartextTrafficPermitted="false">  
            <domain includeSubdomains="true">secure.example.com</domain>  
        </domain-config>  
    </domain-config>  
</network-security-config>

Formato do arquivo de configurações

O recurso de configurações de segurança de rede usa um formato do arquivo XML. A estrutura geral desse arquivo é mostrada no seguinte exemplo de código:

<?xml version="1.0" encoding="utf-8"?>  
<network-security-config>  
    <base-config>  
        <trust-anchors>  
            <certificates src="..."/>  
            ...  
        </trust-anchors>  
    </base-config>  
  
    <domain-config>  
        <domain>android.com</domain>  
        ...  
        <trust-anchors>  
            <certificates src="..."/>  
            ...  
        </trust-anchors>  
        <pin-set>  
            <pin digest="...">...</pin>  
            ...  
        </pin-set>  
    </domain-config>  
    ...  
    <debug-overrides>  
        <trust-anchors>  
            <certificates src="..."/>  
            ...  
        </trust-anchors>  
    </debug-overrides>  
</network-security-config>

As seções a seguir descrevem a sintaxe e outros detalhes do formato do arquivo.

<network-security-config>

**pode conter:**

0 ou 1 de [<base-config>](https://developer.android.com/training/articles/security-config#base-config)  
Qualquer número de [<domain-config>](https://developer.android.com/training/articles/security-config#domain-config)  
0 ou 1 de [<debug-overrides>](https://developer.android.com/training/articles/security-config#debug-overrides)

<base-config>

**sintaxe:**

<base-config [**cleartextTrafficPermitted**](https://developer.android.com/training/articles/security-config#CleartextTrafficPermitted)=["true" | "false"]>  
    ...  
</base-config>

**pode conter:**

[<trust-anchors>](https://developer.android.com/training/articles/security-config#trust-anchors)

**descrição:**

A configuração padrão usada por todas as conexões cujo destino não é coberto por um [domain-config](https://developer.android.com/training/articles/security-config" \l "domain-config).

Qualquer valor não definido usa os valores padrão da plataforma. A configuração padrão para aplicativos destinados ao Android 7.0 (nível de API 24) e superiores é a seguinte:

<base-config cleartextTrafficPermitted="true">  
    <trust-anchors>  
        <certificates src="system" />  
    </trust-anchors>  
</base-config>

A configuração padrão para aplicativos destinados ao Android 6.0 (nível de API 23) e inferiores é a seguinte:

<base-config cleartextTrafficPermitted="true">  
    <trust-anchors>  
        <certificates src="system" />  
        <certificates src="user" />  
    </trust-anchors>  
</base-config>

<domain-config>

**sintaxe:**

<domain-config [**cleartextTrafficPermitted**](https://developer.android.com/training/articles/security-config#CleartextTrafficPermitted)=["true" | "false"]>  
    ...  
</domain-config>

**Pode conter:**

1 ou mais [<domain>](https://developer.android.com/training/articles/security-config#domain)   
0 ou 1 [<trust-anchors>](https://developer.android.com/training/articles/security-config#trust-anchors)   
0 ou 1 [<pin-set>](https://developer.android.com/training/articles/security-config#pin-set)   
Qualquer número de <domain-config> aninhados

**Descrição**

A configuração usada para conexões com destinos específicos, conforme é definido pelos elementos domain.

Observe que, se vários elementos domain-config cobrirem um destino, a configuração com a regra de domínio correspondente mais específica (mais longa) será usada.

<domain>

**sintaxe:**

<domain includeSubdomains=["true" | "false"]>example.com</domain>

**Atributos:**

includeSubdomains

Se "true", a regra de domínio corresponderá ao domínio e a todos os subdomínios, incluindo subdomínios de subdomínios. Caso contrário, a regra se aplica apenas a correspondências exatas.

**Descrição:**

<debug-overrides>

**sintaxe:**

<debug-overrides>  
    ...  
</debug-overrides>

**Pode conter:**

0 ou 1 [<trust-anchors>](https://developer.android.com/training/articles/security-config#trust-anchors)

**Descrição:**

Substituições a serem aplicadas quando [android:debuggable](https://developer.android.com/guide/topics/manifest/application-element.html" \l "debug) for "true", o que normalmente ocorre em compilações de não lançamento geradas por IDEs e ferramentas de compilação. Âncoras de confiança especificadas em debug-overrides são adicionadas a todas as demais configurações e a fixação de certificados não é realizada quando a cadeia de certificados do servidor usa uma dessas âncoras de confiança somente de depuração. Se [android:debuggable](https://developer.android.com/guide/topics/manifest/application-element.html" \l "debug) for "false", esta seção será ignorada por completo.

<trust-anchors>

**sintaxe:**

<trust-anchors>  
...  
</trust-anchors>

**Pode conter:**

Qualquer número de [<certificates>](https://developer.android.com/training/articles/security-config#certificates)

**Descrição:**

Conjunto de âncoras de confiança para conexões seguras.

<certificates>

**sintaxe:**

<certificates src=["system" | "user" | "*raw resource*"]  
              overridePins=["true" | "false"] />

**descrição:**

Conjunto de certificados X.509 para elementos trust-anchors.

**atributos:**

src

A fonte de certificados de CA. Cada certificado pode ser um dos seguintes:

* um código de recurso bruto que aponta para um arquivo que contém certificados X.509. Os certificados devem ser codificados em formato DER ou PEM. No caso de certificados PEM, o arquivo *não deve* conter dados não PEM adicionais, como comentários.
* "system" para os certificados de CA pré-instalados do sistema
* "user" para certificados de CA adicionados pelo usuário

overridePins

Especifica se as CAs dessa fonte ignoram a fixação de certificados. Se "true", a fixação não é realizada em cadeias de certificados que são assinados por uma das CAs dessa fonte. Isso pode ser útil para depurar CAs ou para testar ataques de interceptação no tráfego seguro do seu aplicativo.

O padrão é "false" a não ser que seja especificado em um elemento debug-overrides. Nesse caso, o padrão será "true".

<pin-set>

**sintaxe:**

<pin-set expiration="date">  
...  
</pin-set>

**Pode conter:**

Qualquer número de [<pin>](https://developer.android.com/training/articles/security-config#pin)

**Descrição:**

Um conjunto de fixações de chave pública. Para que uma conexão segura seja confiável, uma das chaves públicas na cadeia de confiança deve estar presente no conjunto de fixações. Consulte [<pin>](https://developer.android.com/training/articles/security-config#pin) para saber mais sobre o formato das fixações.

**Atributos:**

expiration

A data, no formato yyyy-MM-dd, na qual as fixações expiram e são desativadas. Se o atributo não for definido, as fixações não expirarão.

A expiração ajuda a evitar problemas de conectividade em aplicativos que não recebem atualizações para o conjunto de fixações, por exemplo, quando o usuário desativa as atualizações do aplicativo.

<pin>

**sintaxe:**

<pin digest=["SHA-256"]>base64 encoded digest of X.509  
    SubjectPublicKeyInfo (SPKI)</pin>

**Atributos:**

digest

O algoritmo de resumo usado para gerar a fixação. Atualmente, só há suporte para "SHA-256".