FIDO 应用标识符与类型规范 v1.0

FIDO 联盟推荐标准 2014-12-08

当前版本:

https://fidoalliance.org/specs/fido-uaf-v1.0-ps-20141208/fido-appid-and-facets-v1.0-ps-20141208.html

之前版本:

https://fidoalliance.org/specs/fido-appid-and-facets-v1.0-rd-20141008.pdf

编写者:

德克•巴尔凡茨（Dirk Balfanz），谷歌（Google, Inc.）；

布拉德•希尔（Brad Hill)，贝宝（PayPal, Inc.）；

贡献者:

罗尔夫•林德曼博士（Dr. Rolf Lindemann）， Nok Nok Labs, Inc.

达维特•巴格达萨利安（Davit Baghdasaryan）, Nok Nok Labs, Inc.

翻译者:

芦馨雨（Xinyu Lu），联想（Lenovo）

常秉俭（Nick Chang），联想（Lenovo）

摘要

FIDO 协议簇引入了新的安全概念：*应用类型*，来描述用户身份凭证的范围和支持应用隔离的可信计算基如何做出访问控制的决定，决定哪些应用和网页源可以使用哪些密钥。

本文档描述了使用应用类型概念的动机和要求，以及如何应用到FIDO协议中来。

文档状态

本章节描述了文档发布时的状态。本文档有可能会被其它文档所取代。当前FIDO联盟出版物的列表以及此技术报告的最新修订可在FIDO 联盟规范索引上找到。网址：https://www.fidoalliance.org/specifications/.

本文档由FIDO联盟作为推荐标准发布。如果您希望就此文档发表评论，请联系我们。欢迎所有评论。

本规范中某些元素的实现可能需要获得第三方知识产权的许可，包括（但不限于）专利权。FIDO 联盟及其成员，以及此规范的其他贡献者们不能，也不应该为任何识别或未能识别所有这些第三方知识产权的行为负责。

本 FIDO 联盟规范是“按原样”提供，没有任何类型的担保，包括但不限于，任何明确的或暗示的不侵权、适销性或者适合某一特定用途的担保。

本文档已经由 FIDO 联盟成员评审并签署成为推荐标准。这是一篇稳定的文档，可能会作为参考材料被其它文档引用。FIDO 联盟的作用是引起对规范的注意并促进其广泛的分发。

目录

1. 注释

类型名称、属性名称和元素名称用代码形式书写。

字符串文本包含在双引号“”内，比如“UAF-TLV”。

公式中用 “|” 来表示按字节串联操作。

本文档同时适用于 U2F 协议和 UAF 协议中，本文档中 UAF 的专有术语在 FIDO 术语表[FIDOGlossary]中有定义。

本规范中所有的图表、例子和注释是非规范性的。

1.1 关键字

本文档中的关键字：“必须”，“不得”，“要求”，“将”，“将不”，“应该”，“不应该”，“建议”，“可能”，“可选”都会按照[RFC2119]的描述来解释。

2. 概览

本节是非规范性的。

现代的网络应用通常为用户提供了多种与其交互的方式。文档引入了*应用类型*（Application Facet） 的概念来描述一个逻辑应用在不同平台上的标识。例如，应用程序 MyBank 可能有一个安卓版本，一个 iOS 版本和一个可以从浏览器访问的网页版本，这些是应用程序 MyBank 的所有类型。

与传统的用户名口令方式相比，FIDO 架构提供了更简单、更强大的鉴别能力，同时避免了其他鉴别方案的许多不足。FIDO 协议的核心是使用代表用户信任凭据的公私密钥对，来执行挑战和响应操作。

为了最小化经常遇到的关于隐私、“身份”概念的纠纷和可信第三方的必要性的问题，FIDO 的密钥是严格限定范围的，在用户和每个依赖方间动态配置，仅可选择与服务器分配的用户名相关联。与该种方式形成对比的例子，例如 TLS 中使用的传统 PKIX 客户端证书，引入了可信的第三方，在具体实施中混合了标识断言和密钥持有者加密凭证，缺少受众的监督，甚至可能在用户未知或未准许的情况下在协议握手的明文部分被发送。

出于诸多原因，FIDO 方法是更好的，但也引起了一些质疑。

- 什么样的网页源和本地应用（类型）能构成一个单一逻辑的应用，他们怎样进行可靠地识别？
- 在每个网页浏览器和应用都访问相同目标实体提供的服务时，如何避免用户重复在设备上为每个网页浏览器或应用注册新的密钥。
- 怎样做到如用户所希望的那样，在不违反应用程序安全隔离和防护恶意代码的前提下进行注册密钥的共享操作。
- 用户如何在多个具有友好用户界面的 FIDO 可信计算基的设备上进行身份凭证的漫游？

文档描述了 FIDO 通过允许应用程序声明一个贯穿了所有提供给用户的不同类型的凭证范围来解决以上问题（有足够的平台机制来进行实现）。

2.1 动机

FIDO 概念性地给注册密钥设定了一个范围，用一个三元组（用户名，认证器，依赖方）表示。但是依赖方由什么组成呢？对一个用户来说，在相同设备上，通过一个或多个浏览器或者专门的应用程序访问同一依赖方的相同服务集是很普遍的。依赖方可能需要用户执行一个复杂的过程来证明用户的身份并为其注

册一个新的 FIDO 密钥，但是用户不希望在相同的设备上每次打开浏览器或者应用程序时，都需要重复这个复杂的身份验证过程。

2.2 避免 APP 钓鱼欺骗

FIDO 提供一种友好用户界面的认证方式来允许访问注册的密钥，例如输入一个简单的 PIN 码、触摸一个设备或扫描指纹。如果用户在不同的依赖方中重复使用相同的验证输入，应该不会有安全方面的问题，特别是当验证输入是生物识别时，用户可能别无选择。

使用“应用商店”分发模式的现代操作系统经常向用户许诺“安全试用”任何应用程序。操作系统把不同应用隔离开来，所以一个应用不能读取其他应用的数据或者互相访问接口，而且需要获得用户明确的许可来存取共享的系统资源。

如果用户要下载一个恶意的游戏，这个游戏会让用户去激活他的 FIDO 认证器来“保存进度”，但实际上却解锁他的银行凭证并且接管了他的账户，FIDO 就失败了，因为网络钓鱼的风险仅从口令转移到了应用的下载。FIDO 通过保持对注册密钥所代表的高价值共享状态的良好监管，从而不违反平台“安全试用”任何应用程序的承诺。

2.3 与 OAuth 和 OAuth2 的比较

OAuth 和 OAuth2 协议是用于服务器间的安全模型，假定每个应用实例可以发布并且保护“应用的秘密”。这种方法并不适用于应用商店的安全模型。虽然在其应用服务中提供 OAuth 类型的应用安全保障来试图仅允许授权的/官方的应用程序来连接是很常见的，这类“秘密”实际上对所有能够访问应用商店的程序来说是共享的，并且这些“秘密”可以通过简单的逆向工程复原出来。

相反地，FIDO 提出类型的概念从一开始就是为了“应用商店”模型。它依赖于客户端平台的隔离特性，来保证用户在行为良好的“可信组”注册的密钥只停留在该可信组内部，即使后来用户安装了恶意软件也不受影响，并且不需要将秘密硬编码到共享包中。不管怎样，用户必须正确地选择用哪个应用和浏览器来进

行注册。如果应用要求用户在依赖方注册 FIDO 密钥用来非法或仿冒使用，应用商店策略可以删除这些应用。

2.4 非目标

应用类型 (Application Facet)的概念并不是企图通过网络辨别出调用应用的是什么服务。应用程序身份的远程鉴证是明确的非目标。

如果一个未授权的应用能够让用户提供所有用来注册新的 FIDO 密钥的信息，依赖方并不能通过 FIDO 协议或者类型的概念来判别它是未授权的，或者拒绝这个应用执行 FIDO 操作。并且如果用户选择信任这样的应用，也会导致在本文档描述机制外的密钥共享访问。

类型机制为通过可信计算基创建和读取注册密钥提供了一种保持其正确范围的方式，*可信计算基*实现了对恶意程序的隔离。用户凭证也可以在多个拥有友好用户界面的可信计算基设备间漫游，如果每个设备都正确地实现了这种机制，凭证会保持在正确的范围内。然而，在可信计算基与用户敌对的环境里安全并不能得到保证，例如在恶意代码以“root”级别权限运行的设备上。在不提供应用隔离但是使用用户权限运行所有代码的环境里（例如传统桌面操作系统），完好无损的可信计算基（包括网页浏览器）可能仅能够成功地强制规定网络源凭证的可用范围，但是不能有效的强制规定应用程序的可用范围。

3. AppID 和 FacetID 断言

当用户执行注册操作[UAFArchOverview]时，认证器会生成一个新的私钥，并且将公钥发给依赖方。作为该操作的一部分，每个密钥都会与一个 AppID 相关联。AppID 是一个 URL，作为服务器发送的协议报文的一部分，并且指出了凭证的目标。默认情况下，凭证的受众被限制为与 AppID 同源。在某些情况下，依赖方希望在一个更大的范围内使用密钥。如果 AppID URL 采用 https 方案，FIDO 客户端可能会把它作为一个 TrustedFacetList 进行解引用并处理。可信类

型列表指定了范围和受众的限制，包括多个类型，例如与 AppID 源相同 DNS 区域的其他网页源，或者 URL 指出了类似于移动应用这样的可信类型。

> **注释：**
>
> 用户可能会在同一个认证器上为一个 AppID 注册多个密钥，例如有多账户的情况。这种注册可能会使用依赖方用户名或本地昵称来方便用户区分，但是也可能不提供用户名，例如在双因子认证的情况下，与密钥相关的用户账号可能通过带外方式与 FIDO 协议的其他模块进行通信。所有共享同一 AppID 的注册信息，也同时共享相同的受众限制。

3.1 AppID 和 FacetID 断言的处理规则

3.1.1 确定请求应用程序的类型标识符（facetID）

如果是网页，FacetID 必须是触发 FIDO 操作网页的网页源[RFC6454]，可写为一个空路径的 URI。忽略默认的端口和任何路径组件。

一个 FacetID 的例子如下所示:

```
https://login.mycorp.com/
```

如果是安卓系统[ANDROID]，FacetID 必须是一个来源于 APK 签名证书[APK-Signing] 的 SHA-1 哈希值的 URI。例如：

```
android:apk-key-hash:<sha1_hash-of-apk-signing-cert>
```

SHA-1 哈希值的计算如下：

> **例 1: 计算 APK 签名证书的哈希值**
>
> #以 DER 格式，经过哈希、base64 编码、去掉‘=’的形式，输出签名证书
>
> ```
> keytool -exportcert -alias androiddebugkey -keystore \
> <path-to-apk-signing-keystore> &>2 /dev/null | openssl sha1 \
> -binary | openssl base64 | sed 's/=//g'
> ```

如果是 iOS[iOS]系统，facetID 必须是应用程序的 BundleID[BundleID]的 URI。

ios:bundle-id:<ios-bundle-id-of-app>

3.1.2 判断请求者的 FacetID 是否被 AppID 授权

1. 如果 AppID 不是 HTTPS URL，并且与请求者的 FacetID 相匹配，该操作可以继续进行而无需进行额外的处理。
2. 如果 AppID 为 null 值或为空，客户端必须将 AppID 设为请求者的 FacetID，该操作可以继续进行而无需进行额外的处理。
3. 如果请求者的 FacetID 是 https://的形式，源与 AppID 共享相同的主机（例如，一个应用程序的主页是 https://fido.example.com/myApp，将 AppID 设置为 https://fido.example.com/myAppId），该操作可以继续进行而无需进行额外的处理。如果需要的话，该算法可异步进行来获取可信类型列表。
4. 开始用 HTTP GET 方法来获取可信类型列表，地址必须用 HTTPS URL 来指定。
5. URL 必须是可以匿名解引用的，也就是说 HTTP GET 不能包含 cookies、鉴别、源或者引用报头，并且不能有 TLS 证书或者其他形式的证书。
6. 必须将响应的 MIME Content-Type（内容形式）设为 "application/fido.trusted-apps+json"。
7. 当获取可信类型列表时，应该遵守 HTTP 响应中缓存的相关 HTTP 报头。
8. 保存可信类型列表的服务器必须对所有客户端做出统一的响应。也就是说，不能根据任何证凭证材料更改响应正文的内容。包括环境授权，例如请求中提供的源 IP 地址。
9. 如果服务器返回一个 HTTP 重定向（状态码为 3xx），服务器也必须发送 HTTP 报头 FIDO-AppID-Redirect-Authorized:true，并且客户端必须在重定向前验证这个 HTTP 报头。这么做防止了未授权方在目标域中对开放式重定向器的滥用。如果此项检查通过，从第 4 步重新开始算法。
10. 可信类型列表可能含有无数项，但是客户可以截短或者拒绝处理大的响应。

11. 从 trustedFacet 数组的对象中，选择一个与协议报文 version 一致的对象。
12. Ids 中的 URL 方案必须识别出应用程序的身份（例如使用“apk:”“ios:”或者其他相似方案）或者 https：网页源[RFC6454]。
13. Ids 中使用 https://方案的实体必须只能包含方案、主机和端口组件，以及可选的尾部“/”。任何路径、查询字符串、用户名/口令，或者分块信息都必须被丢弃。
14. 所有列出来的网页源必须包括 DNS 中相同最小特定私有标签范围内的主机名，使用如下算法：

    1) 从 https://publicsuffix.org/list/effective_tld_names.dat 处（客户端可以缓存该数据）或者平台中相同功能处获取公共 DNS 后缀列表。
    2) 在重定向前，提取出原始 AppID URL 中的主机部分。
    3) 最小特定私有标签是 AppID URL 中与公共后缀左边加上附加标签后的样式相匹配的一部分。
    4) 对于可信类型列表中的每个网页源，DNS 中最小特定私有标签的计算结果与 AppID URL 的匹配必须是不区分大小写的。不匹配的项必须被丢弃。

15. 如果可信类型列表无法检索或无法通过以上规则完成解析，客户端必须终止对请求的 FIDO 操作的处理。
16. 在处理了正确 version 的 trustedFacets 项、并且移除了所有无效项之后，如果请求者的 FacetID 与 ids 列表中的一项相匹配，那么这个操作可以进行。

3.1.3 TrustedFacet 结构

AppID URL 中的 JSON 资源由一个包含唯一成员 trustedFacets 的结构组成，trustedFacets 是一个 TrustedFacets 结构的数组。

WebIDL

```
dictionary TrustedFacets {
    Version         version;
    DOMString       ids;
};
```

3.1.3.1 TrustedFacets 结构成员

version 类型为 Version

这组可信类型支持的协议版本。可以在[UAFProtocol] 中查看 version 结构的定义。

ids 类型为 DOMString 数组

一个 URL 数组，用来鉴别该 AppID 所授权的类型（facets）。

3.1.4 AppID 示例 1

".com" 是公共后缀。 "https://www.example.com/appID" 作为一个 AppID，位于该地址的资源主体包括：

例 2：

```
{
  "trustedFacets" : [{
    "version": { "major": 1, "minor" : 0 },
    "ids": [
     "https://register.example.com", // 有效，共享 "example.com" 标签
     "https://fido.example.com",      // 有效，共享 "example.com" 标签
     "http://www.example.com",        // 丢弃，不是 https:
     "http://www.example-test.com",//丢弃，"example-test.com" 不匹配
     "https://www.example.com:444"    //有效，端口不重要
    ]
  }]
}
```

在该策略中，"https://www.example.com" 和"https://register.example.com"可以访问为这个 AppID 注册的密钥，但是"https://user1.example.com" 则不可以。

3.1.5 AppID 示例 2

"hosting.example.com" 是公共后缀，在"example.com"下运行并且为多家公司提供托管云服务。"https://companyA.hosting.example.com/appID" 作为一个 AppID，位于该地址的资源主体包括:

例 3：

```
{
  "trustedFacets" : [{
    "version": { "major": 1, "minor" : 0 },
    "ids": [
          "https://register.example.com",
              // 丢弃，不共享"companyA.hosting.example.com"标签
          "https://fido.companyA.hosting.example.com",
            // 有效，共享"companyA.hosting.example.com"标签
          "https://xyz.companyA.hosting.example.com",
              // 有效，共享"companyA.hosting.example.com"标签
          "https://companyB.hosting.example.com"
              // 丢弃，"companyB.hosting.example.com"不匹配
        ]
  }]
}
```

在这个策略中，"https://fido.companyA.hosting.example.com"可以访问为这个 AppID 注册的密钥。但是"https://register.example.com"和 "https://companyB.hosting.example.com" 则不能，因为 DNS 名和 AppID 的名字之间存在一个公共后缀。

3.1.6 获取本地 Android 应用的 FacetID

本节是非规范的。

下面的代码展示了 FIDO 客户端如何获取和构建一个调用的安卓本地应用的 FacetID。

```
  private String getFacetID(Context aContext, int callingUid) {
        String packageNames[] =
aContext.getPackageManager().getPackagesForUid(callingUid);
        if (packageNames == null) {
            return null;
        }
        try {
            PackageInfo info =
aContext.getPackageManager().getPackageInfo(packageNames[0],
PackageManager.GET_SIGNATURES);
            byte[] cert = info.signatures[0].toByteArray();
            InputStream input = new ByteArrayInputStream(cert);
            CertificateFactory cf = CertificateFactory.getInstance("X509");
            X509Certificate c = (X509Certificate) cf.generateCertificate(input);
            MessageDigest md = MessageDigest.getInstance("SHA1");
            return "android:apk-key-hash:" +
                    Base64.encodeToString(md.digest(c.getEncoded()),
Base64.DEFAULT | Base64.NO_WRAP | Base64.NO_PADDING);
        }
        catch (PackageManager.NameNotFoundException e) {
            e.printStackTrace();
        }
        catch (CertificateException e) {
            e.printStackTrace();
        }
        catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
```

```
        }
        catch (CertificateEncodingException e) {
            e.printStackTrace();
        }
        return null;
    }
```

3.1.7 其他的安全性考虑

UAF 协议支持传递 FacetID 给 FIDO 服务器和在鉴别响应的计算结果中包含 FacetID。

信任了网页源类型，毫无疑问也就要信任这个实体名下所有的子域，因为网页用户代理不在这样的源之间提供屏障。所以，在 AppID 示例 1 中，虽然没有明确列出，但是 https://foobar.register.example.com 实际上仍然可以访问 AppID "https://www.example.com/appID"注册的凭证，因为它实际上可以当作 "https://register.example.com"。

这里所描述的控件的组件必须可靠的识别请求者，从而安全地执行这项机制。允许这种识别的平台进程间通信机制如果可用就应该使用。

执行上面描述的控件的组件不太可能验证 TrustedFaceList 中项目的完整性和意图。如果一个可信的类型会被恶意方破解或被作为*混淆代理人*[FIDOGlossary]，用户就有可能在恶意方的诱导下完成鉴别过程。

3.1.7.1 *可信类型标识符通配符*

本节是非规范的。

可信类型标识是不支持通配符的。这遵守了 RFC6125 [RFC6125]7.2 节的内容。

FacetID 是唯一识别特定安全主体的 URI，被信任与给定的注册凭证进行交互。通配符给主体的定义带来了模糊性，因为到底通配符是什么意思、通配符怎样

扩展、在哪里出现等在不同的应用程序和协议中并没有统一的规范。对于表明应用程序身份的方案来说，并没有明确通配符是否在任何方式中都合适。对于网页源来说，它扩展了凭证的范围，有可能包含流氓或有缺陷的主机。

总体来说，这些模糊性可能导致在客户端的实现过程中，出现可被利用的身份检查行为的差异，并且会需要过度复杂和低效率的身份检查算法。

A. 参考文献

A.1 参考规范

[FIDOGlossary]

R. Lindemann, D. Baghdasaryan, B. Hill, J. Hodges, *FIDO Technical Glossary*. FIDO Alliance Proposed Standard. URLs:

HTML: fido-glossary-v1.0-ps-20141208.html

PDF: fido-glossary-v1.0-ps-20141208.pdf

[RFC2119]

S. Bradner. *Key words for use in RFCs to Indicate Requirement Levels*. March 1997. Best Current Practice.

URL:https://tools.ietf.org/html/rfc2119

[RFC6125]

P. Saint-Andre, J. Hodges, *Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS) (RFC 6125)*, IETF, March 2011,

URL:http://www.ietf.org/rfc/rfc6125.txt

[RFC6454]

A.Barth. *The Web Origin Concept*. December 2011. Proposed Standard.

URL: https://tools.ietf.org/html/rfc6454

[UAFProtocol]

R. Lindemann, D. Baghdasaryan, E. Tiffany, D. Balfanz, B. Hill, J. Hodges, *FIDO UAF Protocol Specification v1.0*. FIDO Alliance Proposed Standard. URLs:

HTML: fido-uaf-protocol-v1.0-ps-20141208.html

PDF: fido-uaf-protocol-v1.0-ps-20141208.pdf

A.2 参考资料

[ANDROID]

The Android™ Operating System. Google, Inc., the Open Handset Alliance and the Android Open Source Project (Work in progress)

URL:http://developer.android.com/

[APK-Signing]

Signing Your Applications.The Android™ Operating System. Google, Inc., the Open Handset Alliance and the Android Open Source Project (Accessed 11-March-2014)

URL: http://developer.android.com/tools/publishing/app-signing.html

[BundleID]

"Configuring your Xcode Project for Distribution", section "About Bundle IDs",.Apple, Inc. Accessed March 11, 2014.

URL:https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/ConfiguringYourApp/ConfiguringYourApp.html

[UAFArchOverview]

S. Machani, R. Philpott, S. Srinivas, J. Kemp, J. Hodges, *FIDO UAF Architectural Overview*. FIDO Alliance Proposed Standard.

URLs: HTML: fido-uaf-overview-v1.0-ps-20141208.html

PDF: fido-uaf-overview-v1.0-ps-20141208.pdf

[iOS]

iOS Dev Center Apple, Inc. (Accessed March 11, 2014)

URL: https://developer.apple.com/devcenter/ios/index.action