安全 OTA 升级系统——设计文档

陈晨 519021910176
邹涛旭 519021910075
孟详贺 519021910190
王思皓 519021910160
伍杨子涵 519021910019

2022 年 6 月 17 日

目录

1 uptane

本项目参考 uptane 实现了一个 OTA 升级云端服务器。uptane 是一种开放且安全的软件更新框架设计，用于地面车辆的安全更新框架，包含云端与主 ECU 端和次 ECU 端三个部分组成，是用以保护通过无线传输到 ECU 的软件的 OTA 更新标准。本项目中，我们的云端平台部署在阿里云上，对外开放 web 服务，旨在为使用者提供用户体验良好，操作简便，可随时访问的 OTA 更新服务。云端平台采用前端 + 后端 flask 的框架搭建，OTA 升级云端代码也集成在后端的 python 文件中。

下面介绍 uptane 主要功能与关键 API：

1. 初始化软件存储库（image repository）
2. 初始化指导库（Director）
3. 初始化时间签名服务（Timeserver）
4. 创建主次 ECU 节点

1.1 软件存储库

其中，软件存储库借助 tuf（The Update Framework）库进行实现，其用于存储所有版本的软件/固件，以及基本元数据，包括 OEM 的公私钥，密钥所用算法，签名算法，存储软件对应的文件名，文件大小，文件哈希等等。这些元数据主要用于 OTA 升级过程中的签名认证和文件完整性检查。tuf 提供的存储库类，包含 4 个基本角色：root，target，snapshot，timestamp。root 相当于证书颁发机构，用于签名和管理其他角色的公钥；target 应对所存储的文件元数据进行签名；timestamp 负责签名元数据，由于其签名有效期短，可用于指示存储库中是否有新的软件或元数据。snapshot 应生成 target 元数据文件的元数据，其包含目标元数据文件名，和版本号。

1.2 指导库

指导库为特定车辆生成更新元数据，用以指导不同车辆的各个 ECU 应当安装哪些软件（从软件存储库中获取并进行验证）。同时指导库还会接收来自不同车辆的发送的车辆清单，和各个车辆的 ECU 清单，用以标识 ECU 上运行的软件信息。对每一辆汽车使用 vin 进行唯一标识，对每个 ECU 使用序列号进行唯一标识，同时记录 ECU 所在车辆的标识符，ECU 密钥以及 ECU 级别。指导库同样使用 tuf 提供的 repo 类来实现，但与软件存储库的具体使用存在区别。指导库会为每个车辆（以 vin 进行标识）创建一个 repo，但每个 repo 的 root，target，snapshot，timestamp 的公私钥均相同，且仅为指导库所有，同时以字典的形式存储每个车辆的主 ECU 和 ECU 列表。在用户为某个车辆的指定 ECU 配置对应的软件更新策略后，指导库会将该软件作为 target 添加到对应车辆的 repo 中，以此来指导不同车辆不同 ECU 的软件安装。

同时，指导库呈现给 primary 即主 ECU 以下 XMLRPC 协议接口：

- register_ecu_serial(ecu_serial, ec_public_key, vin,is_primary=False)，表示注册一个 ECU 序列号
- submit_vehicle_manifest(in，ecu_serial，signed_ecu_manifest)，主 ECU 通过此接口传输车辆清单

指导库呈现给呈现给网站如下 XMLRPC 协议接口：

- add_new_vehicle(vin)，增加一个新的车辆
- add_target_to_director(target_filepath, filepath_in_repo, vin, ecu_serial)，将目标车辆分配给指导库中
- write_director_repo()，将新添加的目标添加到实时存储库
- get_last_vehicle_manifest(vin)，获取最近车辆清单历史记录
- get_last_ecu_manifest(ecu_serial)，获取最近 ECU 清单历史记录
- register_ecu_serial(ecu_serial, ecu_key, vin, is_primary=False)，注册一个新的 ECU 序列号

1.3 时间签名服务

时间签名服务负责导入时间签名服务的私钥，如果没有，则需手动生成。之后提供 XMLRPC 服务，为车端 OTA 提供时间签名接口。该签名接口函数接收一个参数随机数 nonces，并使用 timeserver 的私钥对该随机数进行签名，返回签名结果给车端

1.4 车端 ECU 节点

对于主次 ECU 节点，uptane 通过 clean_slate() 设置一个从未更新过的节点，同时主节点也开始通过 XMLRPC 机制监听相应命令。另外，主 ECU 通过 update_cycle() 完成如下工作：

- 从 Director 和 Image 存储库中获取并验证车辆的所有签名元数据
- 获取 Director 指示此车辆安装的所有图像，不包括与图像存储库中的相应图像不完全匹配的图像。从存储库中获取的任何与经过验证的元数据不匹配的图像都将被丢弃。
- 查询 Timeserver 以获取有关当前时间的签名证明，其中包括辅助节点发送的任何 nonce，以便辅助节点可以相信返回的时间至少与其发送的 nonce 一样新
- 生成车辆版本清单，其中包含一些车辆元数据和从辅助节点接收的所有 ECU 版本清单，描述当前安装的图像、每个 ECU 可用的最近时间以及辅助节点观察到的任何攻击的报告（也可以直接调用：generate_signed_vehicle_manifest()‘）
- 将该车辆版本清单发送给主管（也可以直接调用：submit_vehicle_manifest_to_director()‘）

2 前端设计

本项目开发了基于 django 的前端界面，用户可以通过浏览器访问我们的 web 服务，在网页端上进行进行 OTA 升级设置，除了基本的登录注册界面，我们提供了三个界面：镜像配置、升级界面、API 接口。分别进行镜像上传与查看，OTA 模拟升级流程和 API 使用介绍。[1]

[1]由于进度原因，设计的功能并未完全实现

整体上，我们采用前后端分离的设计思想，前后端服务器在逻辑上相互分离，前端服务器利用 HTTP 协议向后端请求响应的信息，如镜像列表，并提供可视化操作界面。

在镜像配置界面，用户上传需要更新的镜像（应用软件或固件），该文件会通过 django 提供的文件上传接口上传到服务器指定目录下，便于后续 OTA 更新策略的配置。同时在用户上传完成后，会调用后端 flask 接口，对用户上传的镜像名称，文件名称以及上传时间进行记录，并显示在当前镜像配置界面中，以便用户查看服务器已有的镜像文件，以及上传时间。

在升级界面，用户可进行具体的 OTA 升级配置。可设置本次更新的名称，需要更新的车辆标识符 vin，以及对应的 ecu 标识，具体的更新镜像（这里的更新镜像与镜像配置中显示的镜像列表一致，用户可在上传好最新的更新软件之后再在此界面对该软件下发更新措施），通过终端模拟车端 ECU 完成具体的一次模拟升级，用户可以在此验证镜像传输是否安全可靠。

3 后端设计

后端设计由三个模块组成：软件存储库（image repository），指导库（Director），和时间签名服务（Timeserver）。它们分别提供存储 image、元数据，自动化指示相应 ECU 安装的 image 和时间签名的功能。

在我们的项目中，软件存储库实现过程如下，首先创建 tuf 存储库，再从本地导入公钥和私钥文件，如果没有则需重新生成（这里只提供了 root 角色的密钥导入代码，其他角色导入过程相同）。再添加初始版本的软件到存储库中，并调用 repo.write() 生成相应的元数据并进行签名，同时我们还需开放 HTTP 服务，供车端对元数据文件和更新文件进行下载。此时，我们已经创建了软件存储库，并导入了公私钥，添加了初始更新软件，同时向车端 OTA 提供 HTTP 端口以供下载，完成了软件存储库的初始化过程。

在我们的项目中，指导库具体实现过程如下，首先导入指导库的公钥和私钥文件，如果没有则需重新生成(这里同样只提供了 root 角色的密钥导入代码)；同时将初始的车辆添加到车辆清单中，并为其创建 repo，包括导入 root，target，snapshot，timestamp 的公私钥。之后为每个车辆 repo 添加初始的软件，并调用 repo.write() 生成相应的元数据进行签名。同时 Director 也需开放 HTTP 服务，供车端对需更新软件列表及其详细信息进行下载，除此之外 Director 还提供了 XMLRPC 服务，用以提供给车端进行注册车辆清单和 ECU 列表。此时，我们为每个车辆创建了一个 repo，用以存储每辆车对应 ecu 应当更新的软件信息，并为每个 ecu 添加了初始更新软件，同时为车端 OTA 提供了 HTTP 端口便于现在更新软件列表，以及 XMLRPC 远程调用服务，进行车辆清单和 ecu 的注册，完成了指导库的初始化过程。时间签名服务相对较简单，其来自车端接收时间签名的请求，该请求包括随机数 nonces，用以 ECU 验证请求结果是否为实时的，避免攻击者对历史消息进行重放。

在该项目中，时间签名服务实现如下：首先需导入时间签名服务的私钥，如果没有，则需手动生成。之后提供 XMLRPC 服务，为车端 OTA 提供时间签名接口。该签名接口函数接收一个参数随机数 nonces，并使用 timeserver 的私钥对该随机数进行签名，返回签名结果给车端 OTA 验证。