MFAF 配置生命周期的安全分析

本报告的其他语言版本：English。

19.7.0 · ProVerif+ Agent 报告

文稿	5G System; Messaging Framework Adaptor Services; Stage 3 （协议定义）
检查的性质	13 条：7 条成立，4 条不成立（5 个攻击），2 条没有答案
发现	1

目录

1 协议和范围
2 发现
3 检查过的性质
4 本次分析的局限
附录 A：逐条性质的解读
附录 C：主张和证明器的答案
附录 D：分析是怎么跑的

1 协议和范围

Nmfaf_3daDataManagement 服务允许 NF 服务消费者配置 Messaging Framework Adaptor Function：DCCF 或 NWDAF 通过 HTTP POST 创建 Individual MFAF Configuration，通过 PUT 修改它，通过 DELETE 删除它（s0006、s0007、s0030、s0070）。配置是一个 MessageConfiguration，携带用于接收映射数据或分析数据的端点的 notificationURI，并且在映射或分析场景下携带一个 correId（s0039、s0040、s0193）；MFAF 分配事务引用 id，当请求中没有 MFAF 通知信息时补充该信息，并存储配置（s0049、s0050、s0052）。消费者持有其 NF 实例标识以及从 NRF 获取的、带有单一服务范围 nmfaf-3dadatamanagement 的访问令牌；MFAF 持有 NRF 的令牌密钥和配置存储（s0229、s0230、s0232）。

每个请求都经由消费者自己的、经过双向认证的 TLS 连接到达 MFAF。MFAF 校验令牌，创建、更新或删除 transRefId 对应的配置，并返回 201 Created 及新的 transRefId、200 OK 或 204 No Content（s0053、s0063、s0076）。最终，消费者确信自己发送的值就是被存储的值，并且没有其他人能修改它们；第二部分并不成立，结果说明了这一点。

参与方：

- **NF 服务消费者（DCCF/NWDAF）**：创建、更新和删除一个配置。持有：NF 实例 id、与 NRF 之间的客户端凭据。
- **NRF**：授权服务器，签发访问令牌。持有：NF 的客户端凭据，以及它可以签发的范围。
- **MFAF**：对请求进行授权并保存配置。持有：NRF 的令牌密钥、配置存储。

图 1 展示了文稿所规定的交互流程。

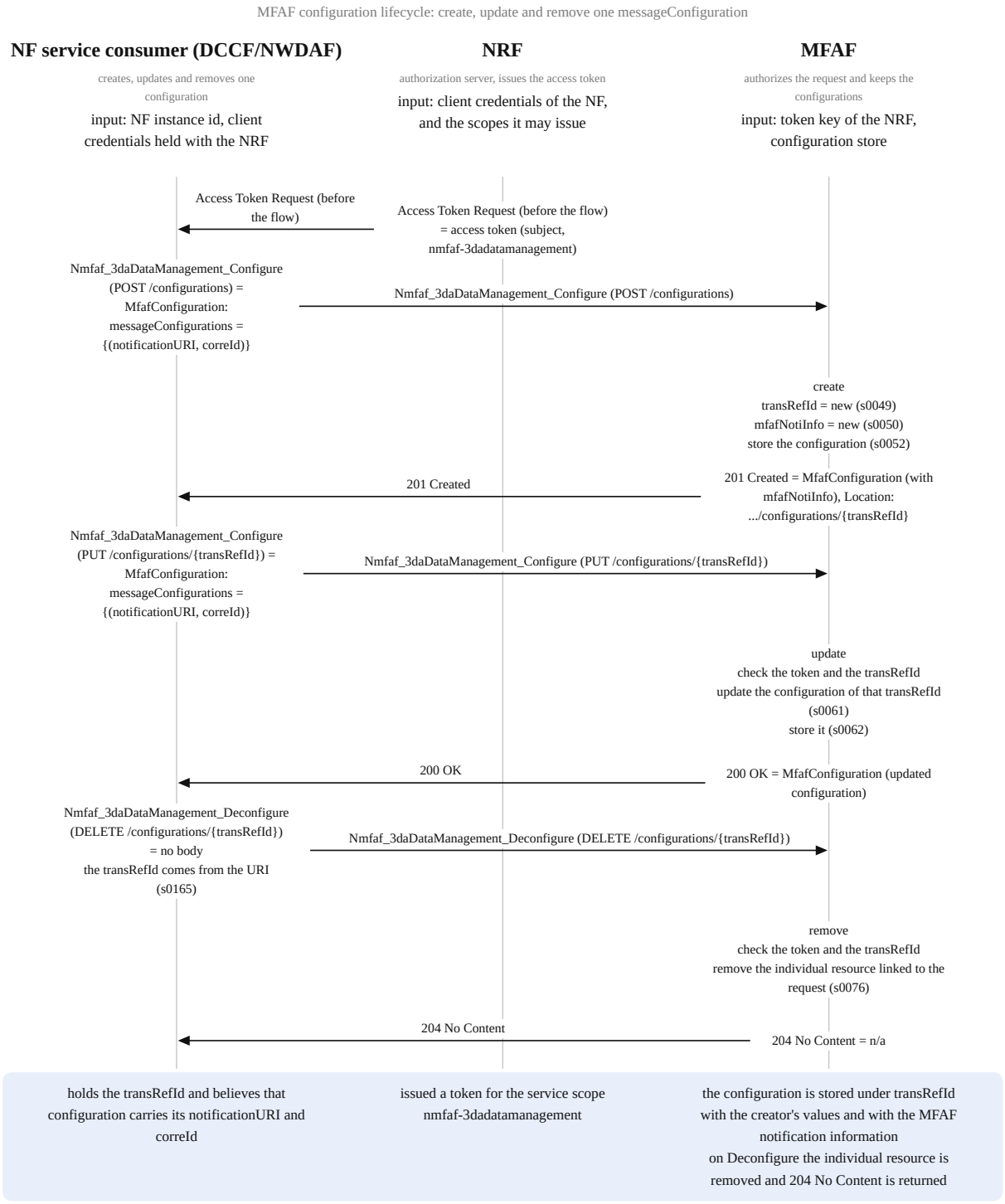


图 1. 文稿所规定的协议。

分析范围见附录 D：分析了什么、省略了什么，以及它所依据的文稿章节。

2 发现

分析共有 1 项发现：以下内容在文稿中没有陈述。其中 1 项无法确定文稿是否已陈述。

生命周期对应关系和令牌性质成立：MFAF 提交的每一次配置操作都对应一次真实的消费者运行，都携带 NRF 为该身份签发的令牌，在 transRefId 未知时保持创建者的内容不变，并且被删除的 transRefId 不会再被更新。唯一的发现是：MFAF 从不检查被要求修改的配置是由谁创建的，因此任何持有服务范围令牌、知道

transRefId 的 NF 都能替换或删除另一个消费者的配置，而服务级令牌是 API 定义中唯一的授权手段。关于重复请求的两条性质 ProVerif 没有给出答案，因为信道上的消息对每个知道该信道的进程始终可读，所以请求到接受的单射性无法在该模型中判定。这些结果依赖于以下限制：TLS 作为经过认证的加密信道、令牌不过期，以及存储仅被标记为已删除。在 13 个目标中，9 个对其所指明的攻击者已证明或有界，4 个是攻击，2 个没有答案。

攻击图沿用图 1 的列。红色表示攻击者的行为。黄色标记的是：某一方发送了某条消息，而规定的交互流程在该位置并没有该消息；以及攻击所破坏的结果。

2.1 任何持有服务范围令牌的 NF 都能替换或删除另一个 NF 的配置，只要它知道该配置的 transRefId

类型	协议规范本身的弱点
是否新发现	不明确：文稿只写了一部分，或没能检索 (rfc6749:2879)
价值	中
涉及的性质	Z03, Z03.9, A08, A08.8 （见第 3 节）
原文	s0232, s0060, s0061, s0076, s0117, s0133, s0144
依据	ProVerif attack queries `z03_u_ctl`, `z03_d_ctl`; Figures 2, 3, 4 and 5

发生了什么。 第二个 NF 服务消费者打开自己的 TLS 连接到达 MFAF，携带自己有效的访问令牌，发送一个 PUT 或 DELETE，指名另一个消费者所创建配置的 transRefId。MFAF 校验令牌主体就是该连接的身份，找到该配置并执行请求：存储的 notificationURI 和 correId 变成第二个消费者的，或者该配置被删除。模型通过一次更新和一次删除都发现了这一点。

攻击者需要什么。 一个有效的、带有 nmfaf-3dadatamanagement 范围且对应其自身 NF 实例的访问令牌，以及目标配置的 transRefId。transRefId 是资源 URI 变量，出现在 Location 头中；原文没有赋予它任何机密性。

攻击者能得到什么。 在该 correId 下收集的映射数据或分析数据将被投递到第二个消费者的 notificationURI，而第一个消费者的配置可以在其不知情的情况下被删除。第一个消费者永远不会得知自己的配置已被替换；它自己之后的 DELETE 接着会删除另一个消费者的配置。

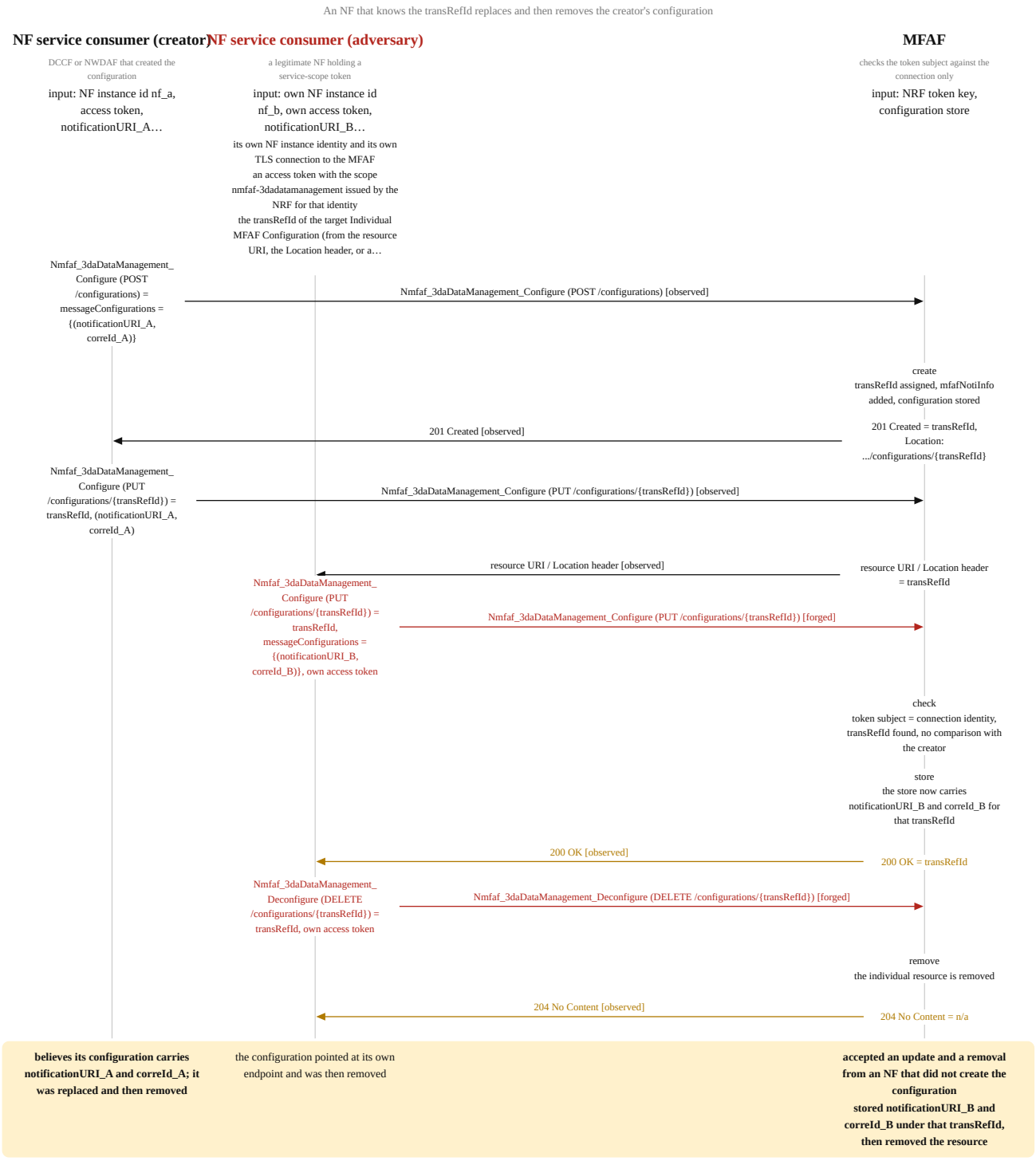


图 2. 一个知道 transRefId 的 NF 替换并随后删除创建者的配置。性质 Z03；攻击查询 z03_d_ctl、z03_u_ctl。

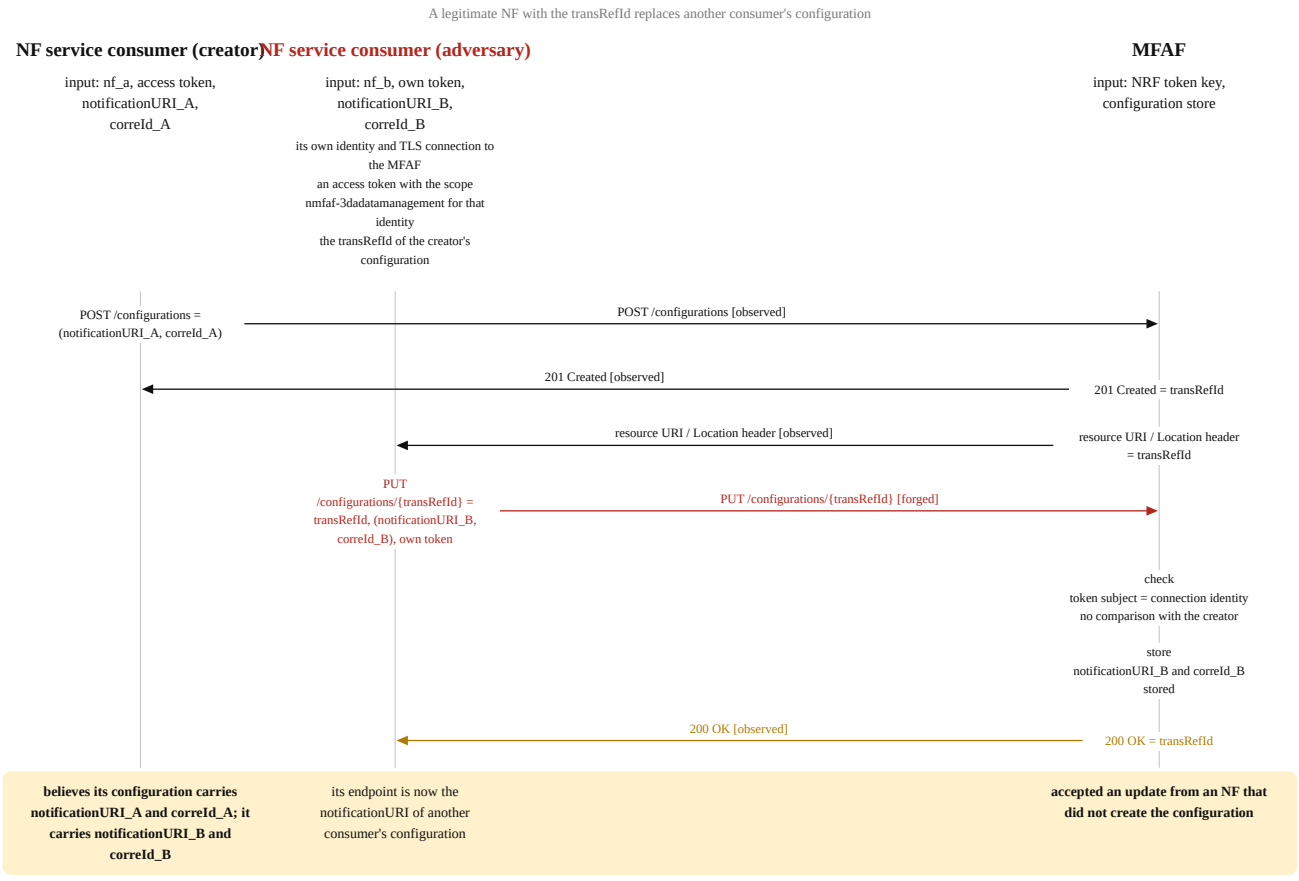


图 3. 一个持有 transRefId 的合法 NF 替换了另一个消费者的配置。性质 Z03.9；攻击查询 z03_u_ctl。

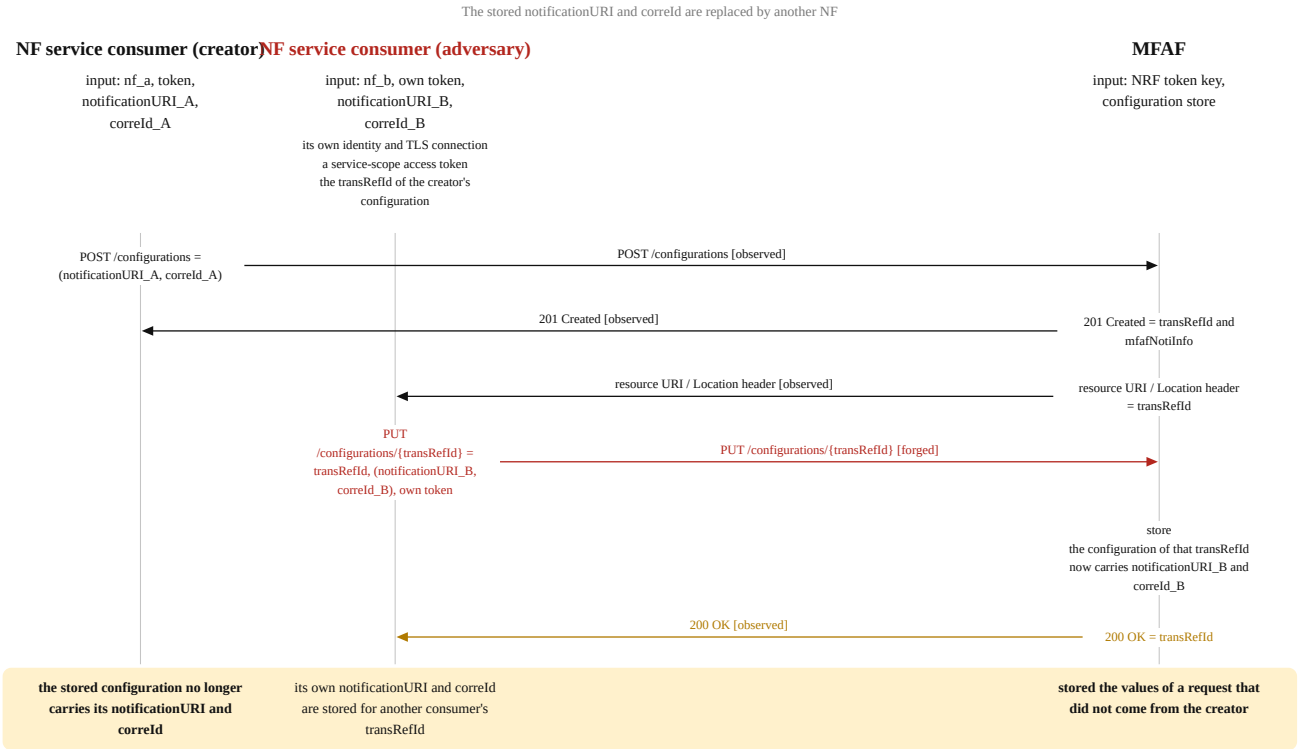


图 4. 存储的 notificationURI 和 correId 被另一个 NF 替换。性质 A08；攻击查询 z03_u_ctl。

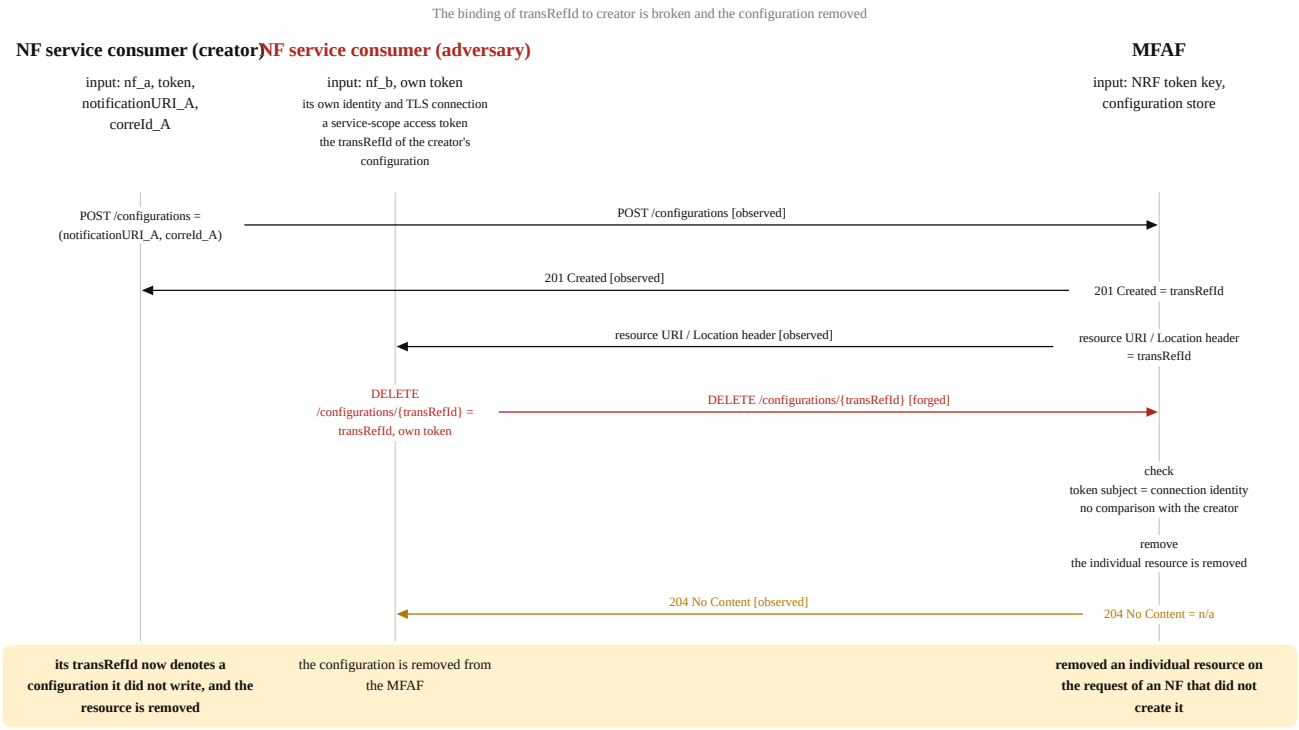


图 5. transRefId 与创建者之间的绑定被破坏，配置被删除。性质 A08.8；攻击查询 z03_d_ctl。

拟议修改。 MFAF 应当为每个 Individual MFAF Configuration 存储创建它的消费者的 NF 实例标识符，并且当访问令牌的主体与该标识符不一致时，应当以 403 Forbidden 拒绝 PUT 或 DELETE。按资源授权还需要超出第 5.1.9 条中单一服务级范围的内容。

用 ProVerif 复核。 未检查：该环境没有 check_fix 工具，因此该修改未在模型上运行。它只是对原文在创建时已持有的某个值做一次检查再加一次比较，不涉及消息，并且消除了该攻击。

对于这些发现相关的攻击，没有给出图：A08.8：z03_u_ctl、A08：z03_d_ctl、Z03.9：z03_d_ctl。

3 检查过的性质

每一行是一条提交给 ProVerif 的性质。结果是 ProVerif 对分析者所建协议模型的回答；价值和文稿是否已陈述该结果是分析者的判断，理由见附录 A。

ID	性质	假定的泄露	结果	价值	文稿是否已写
A03	**被接受操作匹配消费者运行。** 每当 MFAF 接受一次配置创建/更新/删除请求并据此存储或移除配置，就存在一个真实消费者运行，其身份、操作类型、transRefId、notificationURI 与 correId 与 MFAF 本地接受的值一致。	无	成立	低	否
A04	**被接受操作具有不同运行。** 每次 MFAF 接受创建请求并分配 transRefId，都对应不同的消费者创建运行；同一 transRefId 的更新/删除接受对应不同的消费者运行。	无	无答案	低	否

ID	性质	假定的泄露	结果	价值	文稿是否已写
A08	**存储配置等于请求。** MFAF 存储的每个配置中，notificationURI 与 correId 等于该消费者请求中的值；mfafNotiInfo 若请求未提供则由 MFAF 本地生成并绑定到该配置。	无	不成立（1 个攻击）	中	否
Z01	**操作携带有效令牌。** 每次 MFAF 提交配置创建/更新/删除操作，都存在一个由 NRF 签发的、对该消费者有效且含 nmfaf-3dadatamanagement scope 的访问令牌。	无	成立	低	否
Z03	**仅创建者可修改配置。** 使用某消费者令牌提交的更新/删除操作，其目标 transRefId 必须属于该消费者此前创建的配置；令牌不能用于其他消费者的配置。	无	不成立（2 个攻击）	中	不确定
S05	**已移除配置不可更新。** MFAF 接受删除并移除某 transRefId 后，该 transRefId 不再能用于更新，且该配置不再产生通知。	无	在界内成立（变化值展开到 4 步）	低	否
R03	**一操作一副作用。** 同一逻辑创建操作在 MFAF 至多提交一次配置创建副作用；同一逻辑删除操作至多移除一次。	无	无答案	低	否
A08.8	**transRefId 绑定到创建者与内容。** MFAF 存储的配置中，transRefId 与创建者身份、notificationURI、correId 的绑定关系与消费者请求一致。	无	不成立（1 个攻击）	中	不确定
Z03.9	**其他NF不可替换配置。** 使用某消费者令牌提交的更新/删除操作，其目标 transRefId 必须属于该消费者此前创建的配置；其他合法 NF 不能替换或移除该配置。	无	不成立（1 个攻击）	中	不确定
A03.10	**确认跟随状态变更。** MFAF 返回更新成功（200/204）当且仅当对应 transRefId 的配置被实际修改；返回删除成功（204）当且仅当对应资源被实际移除。	无	成立	低	否
X01	**配置内容与令牌在传输中保密。** 配置的 notificationURI 与 correId，以及消费者的访问令牌，只被发送它们的消费者和 MFAF 读取：网络攻击者和其他 NF 服务消费者都无法从基于服务的接口获取它们。	一个有效的 OAuth2 访问令牌，带有针对其自身 NF 实例的 nmfaf-3dadatamanagement 服务 scope，以及一个以其自身身份到 MFAF 的 TLS 会话，以及另一消费者的 Individual MFAF Configuration 的 transRefId，该 transRefId 是在该配置创建后获得的（来自处理过资源 URI 的中间方、来自运营日志，或作为源文不保密的一种资源标识符）。	成立	低	否
X02	**通知端点的强保密性。** 攻击者无法分辨消费者配置了两个 notificationURI 值中的哪一个：攻击者在 SBI 上看到的任何内容都不取决于配置携带的是两者中的哪一个。	无	成立	无	否

ID	性质	假定的泄露	结果	价值	文稿是否已写
X03	**仅对现有配置执行更新或删除。** MFAF 仅对确实由创建请求产生的 transRefId 接受更新或移除：携带 MFAF 的任何配置都不具有的 transRefId 的请求会被拒绝，且不会为其返回任何确认。	一个有效的 OAuth2 访问令牌，带有针对其自身 NF 实例的 nmfaf-3dadatamanagement 服务 scope，以及一个以其自身身份到 MFAF 的 TLS 会话，以及另一消费者的 Individual MFAF Configuration 的 transRefId，该 transRefId 是在该配置创建后获得的（来自处理过资源 URI 的中间方、来自运营日志，或作为源文不保密的一种资源标识符）。	成立	低	否

文稿写明而起草的清单里没有、由分析者补充的目标（见附录 D）：X01, X03.

文稿没有提到、由分析者按性质目录补充的性质（见附录 D）：X02 (K03).

The questions asked, and the analyst's answers:

问题	回答	依据
Q1: 配置交易引用与创建者/notificationURI/correId 的绑定	transRefId 绑定到创建者以及创建请求的 notificationURI 和 correId，并且在 transRefId 未知期间没有东西会改变这些值；但 MFAF 不记录创建者也不做任何检查，因此知道 transRefId 的另一个 NF 会把配置重新绑定到自己名下，然后就能将其移除。	A08.8, A08, Z03
Q2: 其他合法 NF 是否能替换/移除已有配置	是。凭借自己的服务 scope 令牌和该配置的 transRefId，另一个合法 NF 的 PUT 和 DELETE 会被接受，而第一个消费者不会察觉；该 API 既未定义所有权检查，也未定义资源级 scope。	Z03.9, Z03, A08
Q3: 更新确认及注销行为与准确状态关联	对于存在的配置，200 OK 或 204 No Content 跟随变更，DELETE 的 204 跟随移除；源文对不存在的或已被移除的 transRefId 会返回什么没有任何说明，而模型在没有任何答案的情况下拒绝此类请求。	A03.10, S05, X03

4 本次分析的局限

这里的结果是协议模型上的结果：它说明规定的消息和检查以理想化的密码学为前提能给出或不能给出什么，而不涉及任何实现。

正如分析者所述，本次分析不覆盖以下内容：

- TLS 被建模为 NF 实例与 MFAF 之间以 NF 身份为密钥的经过认证的加密信道：记录层重放保护、连接处理和证书验证都只是假设，并未建模。
- 访问令牌是 NRF 中的一条表项，含其主体和服务 scope；令牌格式、有效期、超出单一 MFAF 的受众验证、吊销和令牌窃取均未建模。
- 配置存储只增不减：移除只是一个标志，而更新所存储的内容未被建模为替换先前的内容，因此两次写入的顺序无法表达。
- 通知未建模，因此 S05 中关于已移除配置不产生通知的部分不在模型范围内。
- 请求到接受的单射性（目标 A04 和 R03）没有答案：信道上的消息对每个知道该信道的进程都保持可读，因此重复的请求在模型中可能被接受两次。像 Tamarin+ Agent 这样的有状态模型可以解决这个问题。
- mfafTransfer、DataAnaCollect 特性的通知端点、经 SCP 的重定向、错误响应（未定义任何应用错误，s0223）以及特性协商均未建模。
- NRF 仅被建模为签发令牌的实体；其规程以及调用之前的发现（s0231）不在范围内。

宿主记录的内容：

- 无答案：A04, R03。

- S05、X03 的答案依赖于模型简化的一种操作（附录 D）。
- 证明器没有给出结论的查询（附录 C）：A04: a04_c, R03: r03_c。这条性质问过了，证明器既没有证明它也没有找到攻击，所以结果写“无答案”。
- 没带对照查询的证明（附录 C）：A03: a03_c, A03: a03_u, A03: a03_d, A08: a03_c, A08: a08_m, Z01: z01_c, Z01: z01_u, A03.10: a0310_u, A03.10: a0310_d, X01: x01_nu, X01: x01_ci, X01: x01_tk, X02: x02, X03: x03_u, X03: x03_d。对照查询是这条性质的一个必然不成立的变体；没有它，就没有东西表明这条性质照现在的写法在模型里有可能不成立。

What stays open for each property, and what would settle it, is in Appendix A.

附录 A：逐条性质的解读

这是分析者本人对证明器答案的解读，而非证明器的答案：这些答案对协议意味着什么、该解读依据什么、能否按原样使用，以及它有多大价值。

A03: 被接受操作匹配消费者运行

每当 MFAF 接受一次配置创建/更新/删除请求并据此存储或移除配置，就存在一个真实消费者运行，其身份、操作类型、transRefId、notificationURI 与 correId 与 MFAF 本地接受的值一致。

MFAF 提交的每一次配置操作都对应一个真实消费者运行发出的请求，其身份、transRefId 和配置值均相同。网络上没有任何东西能让 MFAF 依据消费者未曾发送的值行事。

- 结果：成立；是否回答了所问：是；可带保留条件使用
- 价值低：这是传输和令牌要求所蕴含的基础对应关系；它确认模型没有明显错误，而不是显示出任何新东西。
- 文稿是否写明：否。所依据的源文：s0030, s0048, s0052, s0061, s0076
- 依据：TLS 是按 NF 身份认证的加密信道，因此请求无法在线上被伪造或篡改。；令牌仅按主体检查；受众和有效期未建模。
- 下一步：无需进一步处理；该主张中身份部分已由 Z03 和 A08.8 回答，在那里它对另一个 NF 不成立。

A04: 被接受操作具有不同运行

每次 MFAF 接受创建请求并分配 transRefId，都对应不同的消费者创建运行；同一 transRefId 的更新/删除接受对应不同的消费者运行。

关于每次被接受的创建都代表不同消费者运行这一主张没有得到回答：没有找到任何轨迹，也没有排除任何轨迹。就源文而言，重放的创建请求会创建第二个配置，因为 MFAF 为每个请求分配新的 transRefId，且未定义幂等键。

- 结果：无答案；是否回答了所问：不明确；不可用
- 价值低：未获得答案；它指出了关于此 API 的一个问题，而有状态模型必须解决这个问题。
- 文稿是否写明：否。所依据的源文：s0049, s0053, s0075
- 依据：传输建模时没有记录层，因此无法排除重复请求。；源文对重试或幂等性没有任何说明。
- 未决：一个创建请求能否产生两个配置尚未确定：模型的信道不是消耗性的，因此单射性仍然无法得出结论。
- 下一步：使用有状态证明器，例如 Tamarin+ Agent，配合线性请求存储，或由源文澄清重试问题。

A08: 存储配置等于请求

MFAF 存储的每个配置中，notificationURI 与 correId 等于该消费者请求中的值；mfafNotiInfo 若请求未提供则由 MFAF 本地生成并绑定到该配置。

MFAF 会存储消费者请求中的 notificationURI 和 correId，而它自行确定的 MFAF 通知信息绑定到某个配置。改变这些值的并不是 MFAF：另一个带有 transRefId 和自身令牌的 NF 会替换它们，随后创建者的配置就指向了那个 NF 所选择的位置。

- 结果：不成立（1 个攻击）；是否回答了所问：部分；可谨慎使用

- 价值中：存储值的完整性对网络成立，但对另一个 NF 不成立，并给出了替换发生的具体步骤。
- 文稿中是否陈述：否。依据的原文单元：s0039、s0040、s0050、s0052、s0193
- 依据：存储只记录被接受的内容；就地替换内容仅通过接受事件来建模。；通知未建模，因此对投递的影响是论证的，不是展示的。
- 未决：后续更新留下的内容没有被建模为替换先前内容，因此只展示了被接受请求的值。
- 下一步：一个同时向 notificationURI 投递通知的模型将展示投递的数据流向了错误的端点。

Z01: 操作携带有效令牌

每次 MFAF 提交配置创建/更新/删除操作，都存在一个由 NRF 签发的、对该消费者有效且含 nmfaf-3dadatamanagement scope 的访问令牌。

没有 NRF 为出示令牌的身份签发的令牌，任何配置操作都不会被接受，而该服务 scope 是 API 定义的唯一一个。因此令牌并不说明消费者可以对哪个资源进行操作，这正是所有权性质不成立的原因。

- 结果：成立；是否回答了所问：是；可谨慎使用
- 价值低：它确认了原文所要求的授权得到了执行；其意义在于它未覆盖的内容，即资源。
- 文稿中是否陈述：否。依据的原文单元：s0229、s0230、s0232
- 依据：令牌是带有主体和 scope 的表项；过期、撤销和受众未建模。；TLS 连接身份被视为 NF 实例身份。
- 下一步：对该主张无需进一步工作。

Z03: 仅创建者可修改配置

使用某消费者令牌提交的更新/删除操作，其目标 transRefId 必须属于该消费者此前创建的配置；令牌不能用于其他消费者的配置。

仅面对网络时，另一方无法改变配置：由不同身份发起的更新被接受，只在 transRefId 被披露之后才会发生。面对知道 transRefId 的另一个合法 NF，更新会被接受且配置被替换，因为 MFAF 将令牌与连接身份比较，而从不与资源的创建者比较。

- 结果：不成立（2 个攻击）；是否回答了所问：是；可用
- 价值中：API 似乎有意提供的保护对合法 NF 不成立，并且其所需的确切条件已给出；原文并未陈述该条件。
- 文稿中是否陈述：不明确。依据的原文单元：s0232、s0060、s0061、s0076
- 依据：transRefId 在模型中不可猜测，其披露是单独的一步。；恶意 NF 是一个拥有自身令牌和 TLS 凭据的合法实例。
- 未决：攻击需要 transRefId；模型在运行后提供它，而原文既不发布也不保护它。
- 下一步：询问规范编辑者，基线中是否在其他地方有意设置所有权检查，以及 transRefId 是否应当保密。

S05: 已移除配置不可更新

MFAF 接受删除并移除某 transRefId 后，该 transRefId 不再能用于更新，且该配置不再产生通知。

展开的生命周期表明，一旦 MFAF 接受了 DELETE，对同一 transRefId 的过期 PUT 会被拒绝且不产生确认，因此移除不会使配置复活。这在一个配置展开的四步内成立。

- 结果：在界内成立（变化值展开到 4 步）；是否回答了所问：是；可谨慎使用

- 价值低：原文通过移除资源间接陈述了这一性质；模型在展开的生命周期内确认了它，未发现新内容。
- 文稿中是否陈述：否。依据的原文单元：s0076、s0007、s0023
- 依据：只展开了一个配置的四步；拒绝依赖于移除标志，该标志被建模为仅插入的事实。；通知投递未建模。
- 未决：已移除配置不产生通知这一主张超出模型范围。
- 下一步：一个包含通知投递的模型将确定该主张的后半部分。

R03: 一操作一副作用

同一逻辑创建操作在 MFAF 至多提交一次配置创建副作用；同一逻辑删除操作至多移除一次。

重复的创建请求是否产生第二个配置，或重复的 DELETE 是否产生第二次移除，未得到回答。模型未显示一个请求产生两次接受的迹，但也未排除。就原文而言，没有任何规定禁止重复的 POST 创建第二个配置，因为没有定义幂等键或重试规则。

- 结果：无答案；是否回答了所问：未确定；不可用
- 价值低：未得到答案；有用的结果是原文没有为 POST 定义重试或幂等规则。
- 文稿中是否陈述：否。依据的原文单元：s0048、s0049、s0052、s0076
- 依据：没有记录层模型，因此重放既未被允许也未被排除。；原文未定义幂等键。
- 未决：每个逻辑操作至多一个副作用在此处未由 ProVerif 确定。
- 下一步：使用带线性请求日志的 Tamarin+ Agent，或原文中关于重试的陈述。

A08.8: transRefId 绑定到创建者与内容

MFAF 存储的配置中，transRefId 与创建者身份、notificationURI、correId 的绑定关系与消费者请求一致。

创建时，transRefId 绑定到创建它的消费者及其发送的值，且在 transRefId 未知时，其他 NF 无法改变该绑定。当 transRefId 已知时，第二个 NF 替换这些值，然后移除配置，而不通知创建者。

- 结果：不成立（1 个攻击）；是否回答了所问：部分；可谨慎使用
- 价值中：它定位了弱点：绑定存在于创建者和 MFAF 的知识中，而不在存储的记录中。
- 文稿中是否陈述：不明确。依据的原文单元：s0049、s0138、s0193
- 依据：创建者字段的缺失是从数据类型定义中读出的，该定义未列出此类属性。；transRefId 的披露在模型中是单独的一步。
- 未决：MFAF 是否在其记录中保留创建者未被陈述；模型假设不保留，因为不存在此类属性。
- 下一步：由编辑者澄清 MFAF 是否应随配置保留创建者。

Z03.9: 其他NF不可替换配置

使用某消费者令牌提交的更新/删除操作，其目标 transRefId 必须属于该消费者此前创建的配置；其他合法 NF 不能替换或移除该配置。

恶意的合法 NF 在不知道 transRefId 时，无法替换或移除它未创建的配置。一旦它知道 transRefId，更新和移除都会被接受，因此唯一的保护就是标识符本身。

- 结果：不成立（1 个攻击）；是否回答了所问：是；可用
- 价值中：它回答了内部 NF 能否对另一消费者的配置进行操作，并给出了确切前提。
- 文稿中是否陈述：不明确。依据的原文单元：s0232、s0060、s0076

- 依据：transRefId 的披露是单独的一步，代表处理过该 URI 的中间方或运行日志。；只建模了一个其他 NF 身份。
- 未决：攻击者需要 transRefId；没有它该性质成立。
- 下一步：澄清 transRefId 在部署中如何受到保护，例如是否将其视为持有者能力。

A03.10: 确认跟随状态变更

MFAF 返回更新成功（200/204）当且仅当对应 transRefId 的配置被实际修改；返回删除成功（204）当且仅当对应资源被实际移除。

只有在 MFAF 应用了更新之后才会产生 200 OK 或 204 No Content，DELETE 的 204 只有在资源被移除之后才会产生。因此，对于 MFAF 接受的请求，确认的含义与消费者的理解一致。

- 结果：成立；是否回答了所问：是；可谨慎使用
- 价值低：确认了原文所规定确认的解读，未发现新内容。
- 文稿中是否陈述：否。依据的原文单元：s0063、s0064、s0065、s0076、s0223
- 依据：确认在同一进程中跟随状态变更；与状态变更的原子性在模型中是结构性的。；未建模应用错误（s0223）。
- 未决：原文未定义对不存在的 transRefId 的回答，因此等价关系的失败侧未建模。
- 下一步：规范中关于未知或已移除 transRefId 的错误回答的陈述将补全该等价关系。

X01: 配置内容与令牌在传输中保密

The notificationURI and the correId of a configuration, and the access token of the consumer, are read only by the consumer that sent them and by the MFAF: neither the network attacker nor another NF service consumer obtains them from the service-based interface.

配置内容和访问令牌始终留在消费者自己的 TLS 会话内：网络或另一个 NF 都无法获得 notificationURI、correId 或另一个消费者的令牌，攻击者看到的任何内容都不能告诉他两个 notificationURI 值中配置的是哪一个。

- 结果：Holds；回答了所问：是；可带保留使用
- 价值低：这由传输假设推出；其价值在于表明内容保密性并不是这里的失败点。
- 文稿中已说明：否。所依据的原文：s0193, s0230
- 依据：这是 TLS 的模型：由 NF 身份作为密钥的加密通道。这是分析的假设，不是关于 TLS 的结果。
- 下一步：在此范围内无进一步工作。

X02: 通知端点的强保密性

攻击者无法判断消费者配置了两个 notificationURI 值中的哪一个：攻击者在 SBI 上看到的任何内容都不取决于配置携带的是哪一个。

攻击者无法判断消费者配置了两个 notificationURI 值中的哪一个：该值只在消费者的加密会话内使用，因此所配置端点的强保密性对网络成立。

- 结果：Holds；回答了所问：是；可带保留使用
- 价值无：这不是该 API 被要求的性质；它仅由传输假设推出。
- 文稿中已说明：否。所依据的原文：s0193
- 依据：与 X01 相同的 TLS 假设。；同一配置中有两个候选值；该主张说的是已写出的那些运行。

- 下一步：无进一步工作。

X03: 仅对现有配置执行更新或删除

MFAF 只接受针对 create 请求真正产生的 transRefId 的更新或删除：携带 MFAF 的任何配置都不存在的 transRefId 的请求会被拒绝，且不会为其返回任何确认。

更新或删除只对 create 请求真正产生的 transRefId 被接受，无论请求由谁发送，因此消费者无法让 MFAF 对其编造的资源标识符采取行动。此类请求会被拒绝，且不会向任何人确认。

- 结果：Holds；回答了所问：是；可带保留使用
- 价值低：这是 TS 33-501 的一项验证要求，API 在结构上满足它；它表明标识符空间得到了强制执行。
- 文稿中已说明：否。所依据的原文：s0144, s0059, s0075
- 依据：现有 transRefId 的存储在创建时插入。；失败应答未被建模，因为原文未定义任何失败应答。
- 未决：对未知 transRefId 的请求的响应未定义，因此模型中的拒绝不回答任何问题。
- 下一步：无进一步工作。

附录 C：主张和证明器的答案

证明器针对每个目标的记录：达到的级别、达到该级别的查询、是否检查了对照查询，以及各攻击及分析者对每个攻击的判定。

目标	级别	界	查询	对照	攻击	主张
A03	proved		a03_c			4 proved
A04	none					1 undecided
A08	proved		a03_c		z03_u_ctl (holds_in_source)	2 proved, 1 attack
Z01	proved		z01_c			3 proved
Z03	proved		z03_u	checked	z03_u_ctl (holds_in_source); z03_d_ctl (holds_in_source)	2 proved, 2 attack
S05	bounded	the changing value unrolled to 4 steps	s05	checked		2 bounded
R03	none					1 undecided
A08.8	proved		z03_u	checked	z03_d_ctl (holds_in_source)	1 proved, 1 attack
Z03.9	proved		z03_u	checked	z03_u_ctl (holds_in_source)	1 proved, 1 attack
A03.10	proved		a0310_u			3 proved
X01	proved		x01_nu			3 proved
X02	proved		x02			1 proved
X03	proved		x03_u			2 proved

逐条主张，按分析者对每条主张的措辞；每条的答案是证明器给出的：

目标	主张	攻击者	回答	查询	对照
A03	MFAF 接受的每个操作（创建、更新、删除）都对应同一身份的一次真实消费者运行，且 transRefId、notificationURI 和 correId 相同。	`network`	proved	a03_c	none
A03	MFAF 接受的更新和删除是消费者自身的请求，且 transRefId 和取值相同。	`network`	proved	a03_u	none
A03	MFAF 执行的删除是消费者自身的删除请求，且 transRefId 相同。	`network`	proved	a03_d	none
A03	每个被接受的操作都对应 MFAF 所认证身份的一次真实消费者运行，且 transRefId、notificationURI 和 correId 相同。	`nf_with_transrefid`	proved	a03_c	none
A04	每个被接受的创建都对应一次不同的消费者创建运行，因此一次被记录的创建请求不会被重复计数。	`network`	undecided	a04_c	
A08	MFAF 在创建时存储的 notificationURI 和 correId 就是发起创建的消费者请求中的值。	`network`	proved	a03_c	none

目标	主张	攻击者	回答	查询	对照
A08	MFAF 确定的 mfafNotiInfo 绑定到某一个配置：两个配置永远不会携带相同的 MFAF 通知信息。	network	proved	a08_m	none
A08	配置保留创建它的消费者的 notificationURI 和 correId；其他任何 NF 都不能更改它们。	nf_with_transrefid	attack (holds_in_source)	z03_u_ctl (z03_u 的对照)	
Z01	MFAF 对某个配置提交的每个操作都附带一个访问令牌，该令牌由 NRF 为同一 NF 身份和服务范围签发。	network	proved	z01_c	none
Z01	MFAF 提交的每次更新和每次删除都附带一个令牌，该令牌由 NRF 为该 NF 身份和服务范围签发。	network	proved	z01_u	none
Z01	MFAF 提交的每个操作都附带一个令牌，该令牌是为出示该令牌时所声称的 NF 身份签发的；攻击者自己的令牌不能在其他身份下使用。	nf_with_transrefid	proved	z01_c	none
Z03	另一个 NF 不能替换创建该配置的消费者的配置：不同身份对某个 transRefId 的更新被接受，仅当该 transRefId 已经被泄露，而网络攻击者不持有任何一方的信息。	network	proved	z03_u	checked
Z03	另一个 NF 不能删除创建该配置的消费者的配置：不同身份执行的删除被接受，仅当该 transRefId 已经被泄露。	network	proved	z03_d	checked
Z03	另一个 NF 不能替换其创建者的配置，即使它持有有效的服务范围令牌并知道 transRefId。	nf_with_transrefid	attack (holds_in_source)	z03_u_ctl (z03_u 的对照)	
Z03	另一个 NF 不能删除其创建者的配置，即使它持有有效的服务范围令牌并知道 transRefId。	nf_with_transrefid	attack (holds_in_source)	z03_d_ctl (z03_d 的对照)	
S05	MFAF 删除某个 transRefId 之后，对该 transRefId 的过期更新不会被接受；单个配置展开的生命周期（创建、更新、删除、过期更新）会拒绝它。	network	bounded (将变化的值展开到 4 步)	s05	checked
S05	MFAF 删除某个 transRefId 之后，无论是创建者还是另一个 NF 都无法获得对它的更新。	nf_with_transrefid	bounded (将变化的值展开到 4 步)	s05	checked
R03	一个逻辑创建请求最多产生一个配置，一个逻辑删除请求最多产生一次删除。	network	undecided	r03_c	
A08.8	在创建时，MFAF 将 transRefId 绑定到创建它的消费者，以及该请求的 notificationURI 和 correId；此后其他 NF 的更改不会发生，除非 transRefId 已被泄露。	network	proved	z03_u	checked
A08.8	transRefId、创建者消费者和配置值之间的绑定不能被另一个 NF 破坏。	nf_with_transrefid	attack (holds_in_source)	z03_d_ctl (z03_d 的对照)	
Z03.9	恶意合法 NF 不能替换不是它创建的配置：仅考虑网络，为另一个 NF 的 transRefId 接受的更新要求该 transRefId 已经被泄露。	network	proved	z03_u	checked
Z03.9	恶意合法 NF 不能替换或删除不是它创建的配置，即使持有自己的令牌。	nf_with_transrefid	attack (holds_in_source)	z03_u_ctl (z03_u 的对照)	

目标	主张	攻击者	回答	查询	对照
A03.10	只有在配置被更改之后才会发送更新确认，因此 200 OK 表示更新已生效。	network	proved	a0310_u	none
A03.10	只有在单个资源被删除之后才会发送删除确认，因此 204 表示删除已生效。	network	proved	a0310_d	none
A03.10	只有对 MFAF 确实已应用的更新才会返回 200 OK 或 204 No Content，无论发送者是谁。	nf_with_transrefid	proved	a0310_u	none
X01	配置的 notificationURI 始终留在消费者的 TLS 会话内：网络攻击者无法获知它。	network	proved	x01_nu	none
X01	配置的 correId 始终留在消费者的 TLS 会话内：网络攻击者无法获知它。	network	proved	x01_ci	none
X01	消费者的访问令牌不会被另一个 NF 获取：它只在消费者自己的 TLS 会话内传输。	nf_with_transrefid	proved	x01_tk	none
X02	攻击者所看到的任何信息都无法告诉他消费者配置了两个 notificationURI 值中的哪一个。	network	proved	x02	none
X03	只有创建请求确实产生的 transRefId 才能被接受更新，并且只有此时才发送确认。	network	proved	x03_u	none
X03	只有创建请求确实产生的 transRefId 才能被接受更新或删除，因此没有来源的 transRefId 会被拒绝。	nf_with_transrefid	proved	x03_d	none

附录 D：分析是怎么跑的

一名 LLM 分析者根据文稿构建协议模型，并把每个性质交给 ProVerif；第 3 节中的每个结果都是证明器的回答。

ProVerif+ Agent 会话 session-20261114T231509，2026-11-14。状态：已完成。模型：deepseek-v4-1-flash-261017。步数：43，证明器 4 秒，实际耗时 22 分钟，费用 0.132 USD。

ProVerif：运行了 65 次证明，60 次已判定，5 次无答案，4 秒，记录了 29 个回答。

分析了什么

协议	MFAF configuration lifecycle
版本	19.7.0
分析范围	一个 DCCF/NWDAF 与 MFAF；创建、更新、删除单个 messageConfiguration，固定 notificationURI 与 correId，排除 mfafTransfer 和实际多源数据转发。
不在范围内	符号模型不证明算法的计算安全或抗量子安全；攻击轨迹只在明确假设下成立，不直接等同规范漏洞。每个性质必须有可达性/正例控制。固定 Rel-19。分别区分外部网络攻击者与持有自身凭据的恶意合法 NF；TLS 认证加密按端点建模，合法 NF 不能读取其他 NF 的 TLS 会话。OAuth2 subject/audience/scope/expiry 及 TLS 身份绑定依照所附基线；服务 scope 不自动等于资源所有权。单调表不能模拟真实删除，有限状态生命周期必须明确界定。目标是指定 API 子流程，不是完整 5G 安全。前次交付的 native Word 表格有跨条款定位错误；本次从不变的原稿重新解析并独立验证相同范围。旧模型是待复核工件，不能将旧的错配来源当作规范或直接据此宣称原结论为假。
原文	260 个编号原文单元（114238 字节），来自下面列出的章节。本报告中的 s0001 这类引用就是其中一个原文单元

分析所依据的文稿章节：

文稿	章节	标题
main	4.2	Nmfaf_3daDataManagement 服务
main	4.2.1	服务描述
main	4.2.1.1	概述
main	4.2.1.2	服务架构
main	4.2.1.3	网络功能
main	4.2.1.3.1	消息框架适配功能（MFAF）
main	4.2.1.3.2	NF 服务消费者
main	4.2.2	服务操作
main	4.2.2.1	引言
main	4.2.2.2	Nmfaf_3daDataManagement_Configure 服务操作
main	4.2.2.2.1	概述
main	4.2.2.2.2	MFAF 的初始配置
main	4.2.2.2.3	更新 MFAF 的配置
main	4.2.2.3	Nmfaf_3daDataManagement_Deconfigure 服务操作

文稿	章节	标题
main	4.2.2.3.1	概述
main	4.2.2.3.2	删除 MFAF 配置
main	5.1	Nmfaf_3daDataManagement 服务 API
main	5.1.1	引言
main	5.1.2	HTTP 的使用
main	5.1.2.1	概述
main	5.1.2.2	HTTP 标准头部
main	5.1.2.2.1	概述
main	5.1.2.2.2	内容类型
main	5.1.2.3	HTTP 自定义头部
main	5.1.3	资源
main	5.1.3.1	概述
main	5.1.3.2	资源：MFAF 配置
main	5.1.3.2.1	描述
main	5.1.3.2.2	资源定义
main	5.1.3.2.3	资源标准方法
main	5.1.3.2.3.1	POST
main	5.1.3.2.4	资源自定义操作
main	5.1.3.3	资源：单个 MFAF 配置
main	5.1.3.3.1	描述
main	5.1.3.3.2	资源定义
main	5.1.3.3.3	资源标准方法
main	5.1.3.3.3.1	PUT
main	5.1.3.3.3.2	DELETE
main	5.1.4	无关联资源的自定义操作
main	5.1.5	通知
main	5.1.6	数据模型
main	5.1.6.1	概述
main	5.1.6.2	结构化数据类型
main	5.1.6.2.1	引言
main	5.1.6.2.2	类型：MfafConfiguration
main	5.1.6.2.3	类型：MessageConfiguration
main	5.1.6.2.4	类型：MfafNotiInfo
main	5.1.6.2.5	类型：MfafTransferInfo
main	5.1.6.2.6	类型：TargetMfafNotiInfo
main	5.1.6.3	简单数据类型和枚举

文稿	章节	标题
main	5.1.6.3.1	引言
main	5.1.6.3.2	简单数据类型
main	5.1.6.4	描述备选数据类型或数据类型组合的数据类型
main	5.1.6.5	二进制数据
main	5.1.7	错误处理
main	5.1.7.1	概述
main	5.1.7.2	协议错误
main	5.1.7.3	应用错误
main	5.1.8	特性协商
main	5.1.9	安全

协议的目的，即原文步骤在文稿中所定位的内容：一个 DCCF/NWDAF 与 MFAF；创建、更新、删除单个 messageConfiguration，固定 notificationURI 与 correId，排除 mfafTransfer 和实际多源数据转发。

对照文稿定性质

在编写查询之前，分析者已将草拟的性质清单与文稿本身所述协议的目的进行了对照。名称、某条性质所针对的内容以及文稿所述内容均属分析者的解读；所提及的位置在文稿中确实存在。

ID	简称	对象	文稿怎么说	出处
A03	被接受操作匹配消费者运行	MFAF，针对 NF 服务消费者（DCCF 或 NWDAF）：MFAF 针对某次操作所接受并存储或移除的内容，与真实消费者运行针对该操作所发送的内容一致，且身份、transRefId、notificationURI 和 correId 都相同。	MFAF 按 NF 服务消费者的请求执行动作：创建（s0030、s0048、s0052）、更新（s0061、s0062）、移除（s0076）；原文对请求的认证除 TLS 和 OAuth2 令牌外没有其他说明（s0229、s0230）。	s0030, s0048, s0052, s0061, s0062, s0076, s0229, s0230
A04	被接受操作具有不同运行	MFAF：每次被接受的创建都分配一个新的交易参考标识（s0049），并代表一次不同的消费者运行；针对同一 transRefId 的更新和删除接受也代表不同的消费者运行。	s0049 规定每次创建分配一个交易参考标识；原文没有提及重试或幂等键，因此运行身份必须在模型中固定（目标自身的 source_limits）。	s0049, s0053, s0075
A08	存储配置等于请求	MFAF 存储的配置中消费者的 notificationURI 和 correId，以及绑定到该配置的、由 MFAF 生成的 mfafNotiInfo。	请求应携带 notificationURI（s0039），对于映射或数据收集还应携带 correId（s0040）；当请求中没有 mfafNotiInfo 时由 MFAF 确定（s0050），并存储该配置（s0052）；这两个属性在 MessageConfiguration 中都是必选的（s0193）。	s0039, s0040, s0050, s0052, s0193
Z01	操作携带有效令牌	MFAF 对以下事项的保证：对配置的操作来自经 NRF 针对服务范围 nmfaf-3dadatamanagement 授权的 NF 服务消费者。	访问可用 OAuth2 客户端凭据授权，NRF 作为授权服务器（s0229）；消费者在消费服务之前获取令牌（s0230）；该 API 只有一个服务范围（s0232）；RFC 6749 要求未授权方不能生成或修改有效令牌（rfc6749:2870）。	rfc6749:2870, s0229, s0230, s0232

ID	简称	对象	文稿怎么说	出处
Z03	仅创建者可修改配置	创建了某配置的消费者，针对任何其他 NF：其配置不会被另一个持有自身服务范围令牌的 NF 替换或移除；即 MFAF 对针对单个资源的 PUT 或 DELETE 的授权。	该 API 为整个服务定义一个范围，在资源或操作级别没有范围（s0232）；规定 MFAF 在 PUT 和 DELETE 上做什么的条款没有列出任何对请求者是否创建了该资源的检查（s0060、s0061、s0076）；RFC 6749 指出，它没有为资源服务器提供任何方法来确保呈现给它的令牌是签发给该客户端的（rfc6749:2879）。	rfc6749:2879, s0232, s0060, s0061, s0076
S05	已移除配置不可更新	Deconfigure 之后 MFAF 的存储，针对移除该配置的消费者以及其他消费者：被移除的 Individual MFAF Configuration 既不接受更新，也不产生通知。	MFAF 应以 204 确认 DELETE，并移除与删除请求关联的单个资源（s0076）；该服务允许消费者从 MFAF 移除配置（s0007、s0023、s0070）。	s0007, s0023, s0070, s0076
R03	一操作一副作用	MFAF 的存储：一个逻辑创建请求最多存储一个配置，一个逻辑删除请求最多移除一个资源。	一个创建请求导致创建一个新配置并分配一个交易参考标识（s0048、s0049、s0052）；原文没有定义幂等键，也没有提及重试（目标自身的 source_limits）。	s0048, s0049, s0052, s0076
A08.8	transRefId 绑定到创建者与内容	MFAF 存储的 transRefId 与创建者消费者身份、notificationURI 和 correId 之间的绑定，消费者在之后更新或移除该配置时依赖这一绑定。	MFAF 分配一个交易参考标识（s0049）；Individual MFAF Configuration 是在 MFAF 中创建的、与该标识关联的配置（s0138）；MessageConfiguration 携带 notificationURI 和 correId（s0193）；原文没有说明创建者身份是否属于存储记录的一部分。	s0049, s0138, s0193
Z03.9	其他NF不可替换配置	创建了某配置的消费者，针对一个持有自身令牌的恶意合法 NF：该其他 NF 不能替换或移除该配置。	一个服务级范围，没有资源或操作范围（s0232）；没有条款要求 MFAF 检查请求者与该资源的关系（s0060、s0076）。	s0232, s0060, s0076
A03.10	确认跟随状态变更	消费者：对 PUT 返回 200 OK 或 204 No Content 意味着该 transRefId 的配置已被修改；对 DELETE 返回 204 意味着该资源已被移除。	MFAF 在更新资源后返回 200 OK 或 204 No Content（s0063、s0064、s0065），在移除单个资源后返回 204（s0076）；该服务没有定义任何应用错误（s0223）。	s0063, s0064, s0065, s0076, s0223
X01	配置内容与令牌在传输中保密	NF 服务消费者的配置内容和 OAuth2 访问令牌在 SBI 上向 MFAF 传输途中的保密性。	目录条目 K02（载荷与端到端保密性）和 K01（凭据保密性）；RFC 6749 要求访问令牌凭据在传输中保持保密，并且只能通过 TLS 传输（rfc6749:2859、rfc6749:3013）。	rfc6749:2859, s0193, s0230
X02	通知端点的强保密性	配置的 notificationURI，在网络上的攻击者看来。	目录条目 K03（强保密性与不可区分性）；原文没有提及 notificationURI 的保密性，之所以提出这一点，是因为协议中存在一个可以被隐藏的值。	
X03	仅对现有配置执行更新或删除	MFAF 的存储，针对一个消费者（无论是否合法），该消费者指定了 MFAF 从未分配过的 transRefId。	TS 33-501 要求每个 NF 校验所有传入消息，并拒绝不符合协议规范和网络状态的消息（ts-33-501:2875）；Individual MFAF Configuration 资源由一个创建请求产生的 transRefId 标识（s0144、s0117、s0059、s0075）。	ts-33-501:2875, s0144, s0059, s0075

分析者依据文稿补充的性质：X01, X03.

文稿里的安全表述，逐段去向

宿主把文稿里说明协议保护什么、防什么的章节逐段列出，共 398 段；分析者对每一段说明了去向：定为性质、定为攻击者条件，或者为什么都不选。选了 12 段，没选 386 段。列哪些段落由标准的章节结构决定；每一段的去向是分析者的判断。

段落	出处	定为	不选的原因
P001 至 P002	rfc6749:2778-2786	没有陈述目标	这是对 OAuth2 安全模型的介绍性说明，指向威胁模型文稿；其中没有陈述此 API 流程的任何性质。
P003	rfc6749:2790-2795	Z01	
P004 至 P006	rfc6749:2804-2823	不在范围内	关于原生客户端和基于用户代理的客户端的客户端凭据；此 API 的消费者是使用客户端凭据授予的 NF。
P007	rfc6749:2827-2829	Z03, A03	
P008	rfc6749:2831-2838	Z01	
P009 至 P010	rfc6749:2840-2850	不在范围内	关于资源所有者认证以及在授权端点重复发起授权请求；客户端凭据授予没有资源所有者。
P011	rfc6749:2859-2865	X01	
P012	rfc6749:2867-2868	不在范围内	关于隐式授予类型，此服务不使用该类型。
P013	rfc6749:2870-2872	Z01	
P014	rfc6749:2874-2877	Z03	
P015	rfc6749:2879-2881	Z03, Z03.9	
P016 至 P021	rfc6749:2885-2917	不在范围内	刷新令牌；此处使用的客户端凭据授予不签发刷新令牌。
P022 至 P025	rfc6749:2921-2943	不在范围内	授权码；此授予类型不使用授权码。
P026 至 P029	rfc6749:2947-2980	不在范围内	在授权端点操纵重定向 URI；此服务不使用重定向 URI。
P030 至 P033	rfc6749:2984-3003	不在范围内	资源所有者口令凭据；此授予类型不使用该凭据。
P034 至 P035	rfc6749:3013-3019	X01	
P036	rfc6749:3023-3028	A03	
P037 至 P039	rfc6749:3032-3041	不在范围内	在授权服务器猜测客户端凭据；此流程不携带客户端密钥，且授权服务器未建模。
P040 至 P042	rfc6749:3045-3066	不在范围内	对资源所有者的钓鱼；客户端凭据授予没有资源所有者，也没有重定向。

段落	出处	定为	不选的原因
P043 至 P048	rfc6749:3070-3110	不在范围内	基于浏览器的客户端中的跨站请求伪造；NF 不是浏览器。
P049 至 P050	rfc6749:3119-3138	不在范围内	点击劫持；此流程中没有用户代理参与。
P051 至 P052	rfc6749:3142-3150	不在范围内	向客户端注入代码；此流程不接收代码。
P053 至 P054	rfc6749:3154-3167	不在范围内	授权端点上的开放重定向器；此处不使用。
P055 至 P062	rfc6749:3170-3210	不在范围内	在隐式授予中滥用访问令牌以冒充资源所有者；此处没有资源所有者，也没有隐式授予。
P063	ts-33-501:2491-2491	不在范围内	UE 与网络之间的降级攻击；此 API 流程中没有此类安全特性协商。
P064 至 P213	ts-33-501:2495-2853	不在范围内	关于用户认证、RAN、UE、gNB 和用户数据保护的要求；所分析的流程是 MFAF 的 SBI API，这些参与方均不参与其中。
P214	ts-33-501:2859-2859	没有陈述目标	陈述信任域假设并引用后续子条款；其中没有为此 API 陈述自身的任何性质。
P215 至 P217	ts-33-501:2865-2869	不在范围内	服务注册与发现的机密性、完整性和重放保护，发现请求的授权，以及拓扑隐藏；发现和注册不属于所建模的流程。
P218	ts-33-501:2871-2871	A03, Z01	
P219	ts-33-501:2873-2873	不在范围内	指出通过 SCP 或 SEPP 进行间接通信时不存在双向认证；间接通信不在范围内。
P220	ts-33-501:2875-2875	A03, S05, X03	
P221 至 P227	ts-33-501:2879-2891	不在范围内	关于 NRF 平台的安全要求（注册数据保护、注册授权）；NRF 在此仅作为令牌签发者出现，未建模。
P228 至 P234	ts-33-501:2895-2907	不在范围内	NEF 安全要求；NEF 不参与此流程。
P235 至 P241	ts-33-501:2911-2923	不在范围内	关于 SCP 的要求；通过 SCP 的间接通信不在范围内。
P242 至 P254	ts-33-501:2929-2953	不在范围内	通用的 PLMN 间和 SEPP 要求；未建模任何 PLMN 间通信。
P255 至 P292	ts-33-501:2957-3031	不在范围内	漫游场景下的 SEPP 要求；SEPP 不参与此流程。
P293 至 P303	ts-33-501:3035-3055	不在范围内	由漫游中间节点生成的消息；未建模任何漫游中间节点。
P304 至 P312	ts-33-501:3059-3075	不在范围内	跨信任域保护特定属性；此流程停留在单一信任域内，且不携带此类属性。
P313 至 P314	ts-33-501:3079-3081	不在范围内	IPUPS 功能；不属于此流程。
P315 至 P318	ts-33-501:3085-3091	不在范围内	3gpp-Sbi-Originating-Network-Id 头和 PLMN 处理；所建模的流程没有 PLMN 间消息，也没有陈述此 API 的任何性质。
P319 至 P327	ts-33-501:3097-3115	不在范围内	信令监控数据的配置与流式传输；不属于此流程。
P328 至 P337	ts-33-501:3121-3141	不在范围内	安全可见性与可配置性；属于运营商的部署和管理要求，不是此 API 流程的性质。

段落	出处	定为	不选的原因
P338 至 P369	ts-33-501:3149-3215	不在范围内	关于加密和完整性算法标识符及其选择的要求；原语的强度超出符号模型的范围，也超出此 API 的范围。
P370 至 P375	ts-33-501:3219-3231	不在范围内	关于 5G-RG 和 NSSAAF 的要求；二者均不参与此流程。
P376 至 P380	ts-33-501:6540-6548	不在范围内	关于 N3 的安全要求；N3 是用户面接口，不属于此 API。
P381 至 P398	ts-33-501:9656-11901	不在范围内	关于 EAP TLS、IAB、MBSF 和 MBSTF 的附录；这些参与方或过程均不参与此流程。

性质目录，逐条去向

性质是从一份安全和隐私性质目录起草的。起草时有 35 条没选，分析者把它们逐条过了一遍：作为性质在这里检查，或者写明原因不检查。检查了 13 条，没检查 22 条。“文稿没有提到”不能作为不检查的原因。

条目	目录里的性质	作为哪条性质检查	不检查的原因
K01	密钥／凭据保密性	X01	
K02	载荷与端到端保密性	X01	
K03	强保密／秘密值不可区分性	X02	
K04	密钥用途、方向与会话分离	不适用	该流程不建立自己的密钥，也没有密钥层次或方向；传输密钥属于 TLS 基线。
D01	消息完整性与数据来源真实性	A03, A08	
A01	对端曾参与性（Aliveness）	A03	
A02	弱一致性认证	A03	
A05	隐式密钥认证	不适用	该流程不建立任何密钥，因此不存在隐式密钥认证。
A06	显式密钥确认	不适用	该流程确认的是一次配置操作，而不是对密钥的持有。
A07	身份—凭据—密钥绑定	Z01, A08.8	
R01	认证／挑战新鲜性	A04	
R02	消息级抗重放	A04, R03	
R04	消息接受顺序／窗口安全	不适用	该 API 未定义序列号、计数器或接受窗口；PUT 和 DELETE 只按其对该存储的效果定序，这一点由 S05 覆盖。
R05	抗降级／安全策略保持	不适用	第 5.1.8 条的特性协商（suppFeat）选择的是可选特性，而不是安全级别或版本，因此不适用任何降级主张。
C01	前向保密性（FS）	不适用	该流程不建立自己的会话密钥。
C02	抗密钥泄露冒充（KCI）	不适用	该流程内部未定义长期密钥，因此密钥泄露冒充不适用。
C03	已知密钥安全／泄露隔离	不适用	该流程内部不存在密钥层次或多会话密钥派生。
C04	临时秘密／部分状态泄露韧性	不适用	该流程不保存临时秘密；唯一的瞬态值是 transRefId，其泄露对应攻击者 nf_with_transrefid。
C05	泄露后安全恢复（PCS）	不适用	该流程内部没有棘轮或密钥更新。
C06	注册凭据泄露后的增强认证	不适用	未使用 PAKE 或注册记录。
C07	抗离线字典攻击	不适用	该流程不使用口令或低熵秘密。
C08	指定原语失效下的目标保持	不适用	原文未主张混合或后量子组合；算法强度不在符号模型范围内。

条目	目录里的性质	作为哪条性质检查	不检查的原因
Z02	委托权限不扩张	不在范围内	未对委托链建模：通过 SCP 进行的委托发现是该服务中唯一的授权委托，它被排除在范围之外。
Z04	撤销与过期生效	S05	
Z05	跨租户／跨服务安全隔离	Z03, Z03.9	
S01	加密 nonce 唯一性	不适用	该模型不构造 AEAD nonce；传输的保密性假定来自 TLS 基线。
S02	重传／密钥重安装安全	R03	
S03	抗回滚与状态单调性	S05	
S04	跨阶段安全上下文一致性	不适用	配置生命周期是唯一的阶段；不存在可能被混淆的注册、链路建立或恢复阶段。
P01	身份匿名性	不适用	该服务始终通过实例标识符识别 NF 实例；未主张匿名性，也不可能实现。
P02	会话不可关联性	不适用	配置绑定到已识别的 NF 实例，因此会话按设计就是可关联的。
P03	用户／凭据存在性隐藏	不适用	原文未定义按用户或凭据进行的查询，其存在性可能被枚举；它根本没有定义应用错误（s0223）。
V01	条件性进展／错误恢复	证明器问不了	条件性进展和恢复是活性性质；ProVerif 回答的是可达性和等价性，而不是运行总能完成。
V02	无非预期死锁	证明器问不了	运行永远不会卡住这一点无法向 ProVerif 提问。
V03	受限资源滥用／抗放大	证明器问不了	资源消耗和放大超出了事件计数的范围，而事件计数是符号模型所能表达的全部。

参与方和受保护的值，逐项是否设为攻击者条件

宿主把起草的性质里点到的参与方和受保护的值逐项列出，共 14 项；分析者对每一项说明了哪个攻击者条件控制这个参与方或持有这个值，或者为什么不设。设了攻击者条件的 5 项，没设的 9 项。清单来自性质本身；每一项怎么处理是分析者的判断。

ID	参与方或值	类别	攻击者条件	不设的原因
S01	NF 服务消费者	参与方	`nf_with_transrefid`	
S02	MFAF	参与方	按定义全部失守	MFAF 持有整个存储、transRefId 和已配置的内容，并且是该流程中执行所有检查的一方；一旦它失守，这里所问的每条性质都按定义丧失，而文稿在 MFAF 失守的前提下不问其中任何一条。
S03	NRF	参与方	按定义全部失守	失守的 NRF 可以为任意主体和任意受众签发令牌，因此该流程的每条授权性质都按定义丧失，而文稿在授权服务器失守的前提下不问其中任何一条。
S04	恶意合法 NF	参与方	`nf_with_transrefid`	
S05	配置记录	值	`nf_with_transrefid`	
S06	transRefId 交易引用	值	`nf_with_transrefid`	
S07	transRefId	值	`nf_with_transrefid`	
S08	notificationURI	值	攻击者得不到更多	知道通知投递到哪里在该流程中不带来任何能力：要对配置采取行动需要 transRefId 和令牌，而该值的保密性本身作为独立目标被提问。
S09	correId	值	攻击者得不到更多	关联标识符不用于授权任何操作；要对配置采取行动需要 transRefId 和令牌。

ID	参与方或值	类别	攻击者条件	不设的原因
S10	mfafNotiInfo	值	攻击者得不到更多	MFAF 通知信息标识该配置的通知；它不授权任何操作。
S11	OAuth2 访问令牌	值	攻击者得不到更多	在此模型中，仅凭令牌无法让攻击者行动：请求还必须来自该 NF 的 TLS 会话，其密钥由 NF 身份派生，其他方无法获得。连同该 NF 的平台一起窃取的令牌就是攻击者 nf_with_transrefid。
S12	nmfaf-3dadatamanagement scope	值	攻击者得不到更多	该 scope 名称是公开的：原文以明文给出（s0232）。
S13	访问令牌	值	攻击者得不到更多	同 S11：在没有消费者传输凭据的情况下，承载令牌无法在建模的流程中使用。
S14	创建者身份	值	攻击者得不到更多	NF 实例标识符在服务化架构中是公开身份；创建者身份是 A03 和 Z03 所讨论的内容，而不是秘密。

每条性质在各攻击者条件下的答案

攻击者由分析者设定，且在编写查询之前就已设定；每个答案都记录在其中一个攻击者之下。单元格是证明器在该攻击者下对该性质的答案；“无答案”是分析者为该性质设定的、证明器未给出结论或未记录任何内容的攻击者集合；空单元格是分析者未为该性质设定的攻击者。

攻击者	持有什么	依据	出处
`network`	只有网络：它读取、丢弃、修改和发送消息，不持有任何一方的秘密		
`nf_with_transrefid`	一个有效的 OAuth2 访问令牌，带有 nmfaf-3dadatamanagement 服务 scope，用于其自己的 NF 实例；一个以自身身份连接 MFAF 的 TLS 会话；以及另一个消费者的 Individual MFAF Configuration 的 transRefId，该值在该配置创建之后获得（来自处理过资源 URI 的中间方、来自运维日志，或作为原文不视为秘密的资源标识符）。	任务区分了外部网络攻击者与持有自身凭据的恶意合法 NF。原文定义了一个服务级 scope，没有资源级或操作级 scope（s0232），不要求检查 PUT 或 DELETE 的请求者是否创建了该配置（s0060, s0076），也不对 transRefId 赋予保密性，它是资源 URI 变量（s0117, s0144），携带在 Location 头中（s0133）；RFC 6749 指出 OAuth2 没有给资源服务器任何方法来确保令牌是签发给出示它的客户端的（rfc6749:2879）。	rfc6749:2879, s0117, s0133, s0144, s0232, s0060, s0076

ID	`network`	`nf_with_transrefid`
A03	成立	成立
A04	无答案	
A08	成立	不成立
Z01	成立	成立
Z03	成立	不成立
S05	成立	成立
R03	无答案	
A08.8	成立	不成立
Z03.9	成立	不成立
A03.10	成立	成立
X01	成立	成立

ID	network	nf_with_transrefid
X02	成立	
X03	成立	成立

原文和目标

Source: 260 units, 114238 bytes, frozen in source-delivery-01.json. Targets: 7 approved (A03, A04, A08, Z01, Z03, S05, R03); the questions asked: 3. 目标清单已按起草版本事先批准，使用 --approve-targets。

宿主按其文档化的默认值（任务 `decisions: auto`）做出的决定，供你核查：

- approve_package：批准源文包 protocol-40801477fa231ee0bb0a793ce80c

本步骤自报的情况：1 次请求（1 次失败），0.000 USD，输入 64865 个 token，输出 4012 个，188 秒。

关于证明器和模型的说明

- S05：在 ProVerif 中作答，分析者为该目标指定 changing_state
- X03：在 ProVerif 中作答，分析者为该目标指定 changing_state

简化代数的 ProVerif 模型（xor、dh_algebra、other_algebra）给攻击者的能力少于真实操作：在那里得到的证明是针对更弱攻击者的证明，在那里得到的攻击就是攻击。省略对变化值检查的 ProVerif 模型（changing_state）让参与方接受更多：在那里得到的证明成立，在那里得到的攻击可能只是缺失的检查。Tamarin+ Agent 按其原样对这些建模。

基于步数的界是 bounded.pv 上的一个答案：这是分析者的第二个 ProVerif 模型，其中保存着一个会变化的值的参与方被展开到该步数，并保留其检查。该模型所做的少于协议所做的，因此该性质对这些步成立，不能推得更远；在那里找到的攻击就是攻击。步数由分析者给出。

分析者对状态的审查

这部分是分析者自己对原文的解读，不是证明器的答案。在“由 ProVerif 展示”一列下的“是”表示 ProVerif 在展开模型中找到了带这些步的迹，因此这些步是可能的；它们是否破坏某个目标，只能由答案表中的一次攻击来表明，而它们还要付出什么代价，则属于分析者的解读。“未发现”只说明查看了哪些模式，不证明任何事情。

值	保存者	变化时机	检查时机	引用
某个 transRefId 的 Individual MFAF Configuration 记录：其内容（notificationURI、correId）、创建它的 NF 服务消费者的身份，以及它已被 Deconfigure 移除的标志。	MFAF（s0052、s0062：它存储该配置，并在更新后再次存储；s0076：它移除该单独资源）。	POST 创建它并分配 transRefId（s0048、s0049）；PUT 用更新请求的内容替换其内容（s0061、s0062）；DELETE 将其标记为已移除（s0076）。	PUT 和 DELETE 必须指名一个已存在的 transRefId（s0144、s0059、s0075）；Deconfigure 之后该资源不再存在（s0076）。这些条款没有列出对创建者身份的任何检查，也没有列出对令牌主体与资源之间关系的任何检查（s0060、s0061、s0076）。	s0052、s0062、s0076、s0048、s0049、s0061、s0059、s0075、s0144、s0223

值	保存者	变化时机	检查时机	引用
消费者在收到 201 Created 响应后保存（Location 头）并在之后每次 PUT 和 DELETE 中引用的 transRefId。	创建该配置的 NF 服务消费者（s0053、s0133），以及 MFAF 在其存储中的保存（s0049、s0144）。	它在创建时由 MFAF 分配一次，此后永不改变；新的创建会分配一个新的 transRefId（s0049）。	MFAF 在每次 PUT 和 DELETE 时将其与自己的存储进行匹配（s0059、s0075）；消费者将自己收到的 transRefId 与自己所引用的那个进行比较。	s0049、s0053、s0133、s0059、s0075、s0144

模式	结果	目标	由 ProVerif 展示	步数
replay	未发现			被记录的请求无法重放到新连接中：连接在由 NF 身份派生的密钥下进行认证和加密，并且每个请求都携带一个每次运行都新鲜的值，由消费者和 MFAF 进行比较。ProVerif 无法确认这一否定结论：信道上的消息对所有知道该信道的进程始终可用，因此对 create、update 或 delete 接受性的单射性查询仍然没有结论（目标 A04、R03）。
repeat_one_time_action	未发现			移除是一个只会被置位的标志：一旦 Deconfigure 被接受，针对该 transRefId 的第二次 DELETE 会看到该标志并被拒绝，而 MFAF 最后一个展开状态会拒绝过期的 PUT（bounded.pv，对从未到达的事件 UpdateAfterRemoval 的查询 s05）。
desynchronisation	候选		否：未用 ProVerif 尝试	nf_a 创建一个配置并收到其 transRefId（s0048、s0049、s0053）。/ nf_a 发送带有其 notificationURI 和 correId 的 PUT；MFAF 为该 transRefId 存储该内容（s0061、s0062）。/ 该 transRefId 为另一个合法 NF 所知：它就是资源 URI 变量（s0117、s0144），并且它曾通过 Location 头传递（s0133）。/ 另一个 NF 用其自己的令牌为该 transRefId 发送 PUT；MFAF 只根据 TLS 身份检查令牌主体，找到该配置并存储另一个 NF 的内容（s0060、s0061、s0076）。/ nf_a 之后的 DELETE 移除了现在携带另一个 NF 内容的资源（s0076）；nf_a 始终不知道其配置已被替换。
update_before_check	未发现			在展开的 MFAF 中，令牌检查、transRefId 的存在性和已移除标志都在内容被更改之前、在任何确认产生之前完成（bounded.pv，宏 M2 和 M4）；同样的顺序也在 model.pv 中，其中 updated 事件跟在存储查找之后。
no_rollback_on_failure	未发现			被拒绝的请求使记录保持原样：只有被接受的 DELETE 才置位已移除标志，只有被接受的 PUT 才产生 updated 事件和确认，因此被放弃或被拒绝的一次运行不改变任何东西。
race_between_sessions	未发现			两个连接可以携带针对同一 transRefId 的 PUT；两者都被接受，后到的那个设置存储的内容，这是普通的 HTTP 替换语义。模型无法对表上的两次写入进行排序，因此这仅在原文上进行了查看。
distinguishable_failure	未发现			原文没有为该服务定义任何应用错误（s0223），也根本没有失败答案，因此没有什么能告诉攻击者请求为何被拒绝；模型因此没有可供比较的失败答案。这是目标之外的一项发现：原文没有说明 MFAF 对针对其不认识的 transRefId 的 PUT 或 DELETE 会回答什么。
wrap_or_window	未发现			该 API 没有计数器、序列号或接受窗口；唯一的线性值是 transRefId，它只分配一次且永不复用，因为 MFAF 会新选一个。
other	未发现			查看了 MFAF 是否保留关于创建消费者的任何记录：MessageConfiguration 没有任何属性携带创建者（s0193），也没有任何读取条款要求该检查，而这正是 Z03 和 A08.8 所问的，也是发现攻击的地方。

分析者为目标命名的内容：

步骤	目标	特征	原因
24	S05, X03	changing_state	S05 和 X03 建立在一个配置的生命周期状态之上，该状态由 MFAF 保存并更改：已创建、已更新、被 Deconfigure 标记为已移除。model.pv 将其记录在一个只增不减的表中（因此“removed”是一个标志，而非删除），并且无法表达移除与之后更新之间的顺序；bounded.pv 将 MFAF 展开经过 created、updated、removed 以及再一个状态，由消费者发出 POST、PUT、DELETE 和一个过期的 PUT。
24	A03, A04, A08, Z01, Z03, Z03.9, A08.8, A03.10, R03, X01, X02	plain_primitives, many_sessions	唯一的密码学是从 NF 身份派生的对称密钥，用于端点对端点连接（TLS 层，建模为经过认证的加密信道）、代表 NRF 签发访问令牌的令牌表，以及为 transRefId 和 MFAF 通知信息使用的新鲜名；任何地方都没有使用代数定律。一致性、授权和披露性质针对任意数量的会话和连接进行询问。

被证伪且未提交

证明器证伪了这些性质，它们不包含在任何已记录的答案中。

- `a04_c`（model.pv，在模型的较早版本上，此后未运行）：s: bitstring, r: bitstring, tr: bitstring, u: bitstring, v: bitstring, m: bitstring; inj-event(ConfigCreated(s,r,tr,u,v,m)) ==> inj-event(PostSent(s,r,u,v))
- `a04_u`（model.pv，在模型的较早版本上，此后未运行）：s: bitstring, r: bitstring, tr: bitstring, u: bitstring, v: bitstring; inj-event(ConfigUpdated(s,r,tr,u,v)) ==> inj-event(PutSent(s,r,tr,u,v))
- `a04_d`（model.pv，在模型的较早版本上，此后未运行）：s: bitstring, r: bitstring, tr: bitstring; inj-event(ConfigRemoved(s,r,tr)) ==> inj-event(DeleteSent(s,r,tr))
- `r03_c`（model.pv，在模型的较早版本上，此后未运行）：s: bitstring, r: bitstring, tr: bitstring, u: bitstring, v: bitstring, m: bitstring; inj-event(ConfigCreated(s,r,tr,u,v,m)) ==> inj-event(PostSent(s,r,u,v))

分析者的结束说明

MFAF 配置生命周期（对一个 Individual MFAF Configuration 的 POST / PUT / DELETE）在 ProVerif 中进行了建模，其中 TLS 作为按端点认证的加密信道、NRF 签发的服务范围令牌和配置存储，外加 bounded.pv 将生命周期展开到四步，以及一次状态审查。

已确立：生命周期对应成立（A03），每个已提交操作都携带 NRF 为该身份签发的令牌（Z01），确认跟随状态变化（A03.10），更新或移除仅对已存在的 transRefId 被接受（X03），移除是最终的（S05，以对照查询为界展开到 4 步），配置内容和令牌保持在消费者的会话内（X01、X02），创建侧完整性和 mfafNotiInfo 绑定成立（A08）。

发现（价值中，新颖性不明确）：MFAF 从不检查 PUT 或 DELETE 的请求者是否创建了该配置。任何持有服务范围令牌且知道 transRefId 的 NF 都能替换另一个消费者的 notificationURI 和 correId（针对 Z03、Z03.9、A08、A08.8 的攻击，每项都有已证实的仅网络对照及其被证伪的对照查询），随后还能删除该配置，而创建者不会察觉。建议：将创建者的 NF 实例 id 与配置一同存储，并拒绝令牌主体不一致的 PUT 或 DELETE（未经过机器检查：此处没有 check_fix 工具）。

无答案：A04 和 R03（重复请求不得产生第二次副作用）在最终模型上无定论——ProVerif 的通道不是消耗性的，因此同一个请求可能被两个处理程序副本接受；Tamarin+ Agent 配合线性请求存储可以解决它们。

模型之外：通知投递（S05 的后半部分）、mfafTransfer、重定向、错误响应、令牌过期与撤销，以及 SCP 路径。状态审查中的去同步候选被记录为未经 bounded.pv 的查询确认。