Security analysis of MFAF configuration lifecycle

This report in another language: 中文.

19.7.0 · ProVerif+ Agent report

Documents	5G System; Messaging Framework Adaptor Services; Stage 3 (protocol definition)
Properties examined	13: 7 hold, 4 fail (5 attacks), 2 without an answer
Findings	1

Contents

1 Protocol and scope
2 Findings
3 Properties examined
4 Limits of this analysis
Appendix A: Reading of each property
Appendix C: Claims and prover answers
Appendix D: How the analysis was run

1 Protocol and scope

The Nmfaf_3daDataManagement service lets an NF service consumer configure the Messaging Framework Adaptor Function: the DCCF or the NWDAF creates an Individual MFAF Configuration with an HTTP POST, changes it with a PUT and removes it with a DELETE (s0006, s0007, s0030, s0070). A configuration is a MessageConfiguration carrying the notificationURI of the endpoint that is to receive mapping data or analytics and, for mapping or analytics, a correId (s0039, s0040, s0193); the MFAF assigns the transaction reference id, adds MFAF notification information when the request has none, and stores the configuration (s0049, s0050, s0052). The consumer holds its NF instance identity and an access token with the single service scope nmfaf-3dadatamanagement from the NRF; the MFAF holds the token key of the NRF and the configuration store (s0229, s0230, s0232).

Each request travels on the consumer's own mutually authenticated TLS connection to the MFAF. The MFAF checks the token, creates, updates or removes the configuration of the transRefId, and answers 201 Created with the new transRefId, 200 OK or 204 No Content (s0053, s0063, s0076). At the end the consumer is sure that the values it sent are the values stored and that nobody else can change them; the second part does not hold, as the results show.

The parties:

- **NF service consumer (DCCF/NWDAF)**: creates, updates and removes one configuration. Holds: NF instance id, client credentials held with the NRF.
- **NRF**: authorization server, issues the access token. Holds: client credentials of the NF, and the scopes it may issue.
- **MFAF**: authorizes the request and keeps the configurations. Holds: token key of the NRF, configuration store.

Figure 1 shows the exchange as the documents specify it.

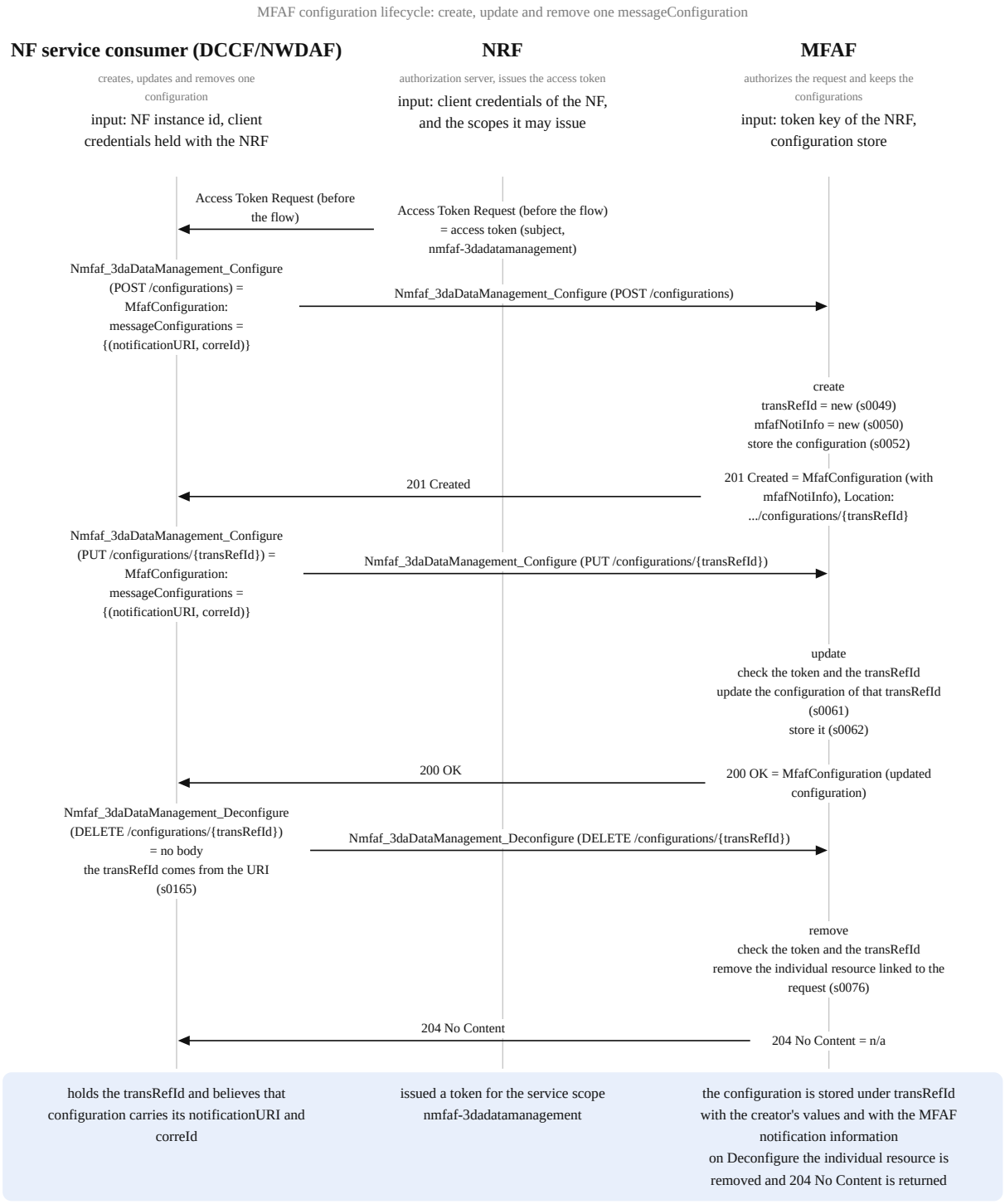


Figure 1. The protocol as specified.

The scope of the analysis is in Appendix D: what was analysed, what was left out, and the sections of the documents it is built from.

2 Findings

The analysis has 1 finding: what follows is not stated in the documents. For 1 of them it is unclear whether the documents state it.

The lifecycle correspondence and the token properties hold: every configuration operation the MFAF commits matches a real consumer run, carries a token the NRF issued for that identity, keeps the creator's content intact while the transRefId is unknown, and a removed transRefId is never updated again. The one finding is that the MFAF never checks who created the configuration it is asked to change, so any NF with a service-scope token that knows the transRefId replaces or removes another consumer's configuration, and the service-wide token is the only authorization the API defines. Two properties about repeated requests have no answer from ProVerif, because a message on a channel stays readable to every process that knows the channel, so injectivity of request to acceptance cannot be settled in this model. The results rest on the limits: TLS as an authenticated encrypted channel, no token expiry, and a store that is only flagged removed. Of the 13 targets, 9 are proved or bounded for the attacker they name, 4 are attacks and 2 have no answer.

The attack figures use the columns of Figure 1. Red is what the attacker does. Yellow marks a message that a party sends although the specified exchange does not have it at that point, and the outcome that the attack breaks.

2.1 Any NF with a service-scope token can replace or remove another NF's configuration whose transRefId it knows

Kind	Weakness of the protocol as specified
New	Unclear: the documents state it in part, or could not be searched (rfc6749:2879)
Value	medium
Properties	Z03, Z03.9, A08, A08.8 (section 3)
Source text	s0232, s0060, s0061, s0076, s0117, s0133, s0144
Shown by	ProVerif attack queries `z03_u_ctl`, `z03_d_ctl`; Figures 2, 3, 4 and 5

What happens. A second NF service consumer opens its own TLS connection to the MFAF and sends a PUT, or a DELETE, naming the transRefId of a configuration created by another consumer, with its own valid access token. The MFAF checks that the token subject is the identity of the connection, finds the configuration and applies the request: the stored notificationURI and correId become the second consumer's, or the configuration is removed. The model finds this with an update and with a removal.

What the attacker needs. A valid access token with the nmfaf-3dadatamanagement scope for its own NF instance, and the transRefId of the target configuration. The transRefId is the resource URI variable and travels in the Location header; the source assigns it no confidentiality.

What the attacker gains. Mapping data or analytics collected under that correId would be delivered to the second consumer's notificationURI, and the first consumer's configuration can be removed without its knowledge. The first consumer never learns that its configuration was replaced; its own later DELETE then removes the other consumer's configuration.

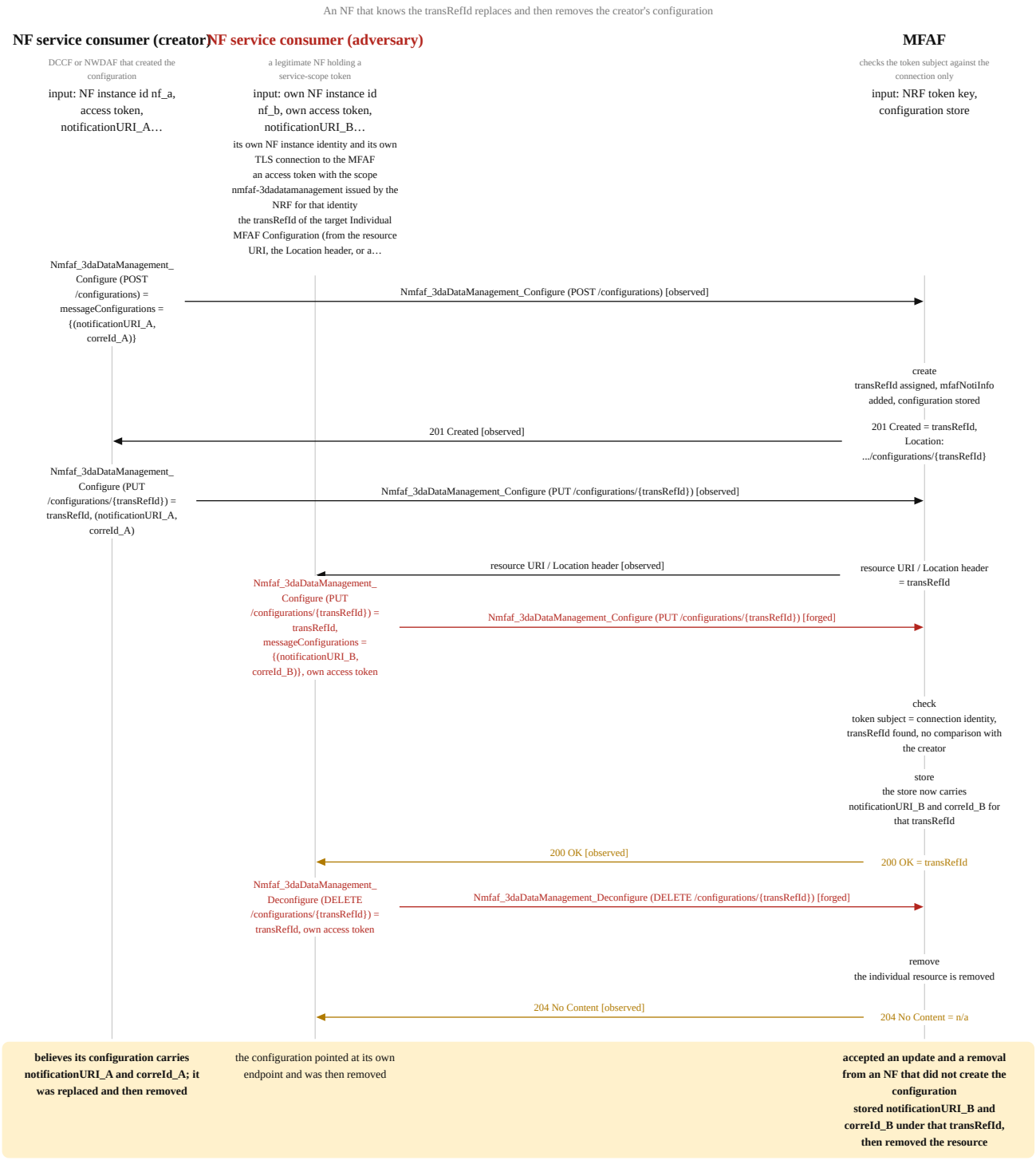


Figure 2. An NF that knows the transRefId replaces and then removes the creator's configuration. Property Z03; attack queries z03_d_ctl, z03_u_ctl.

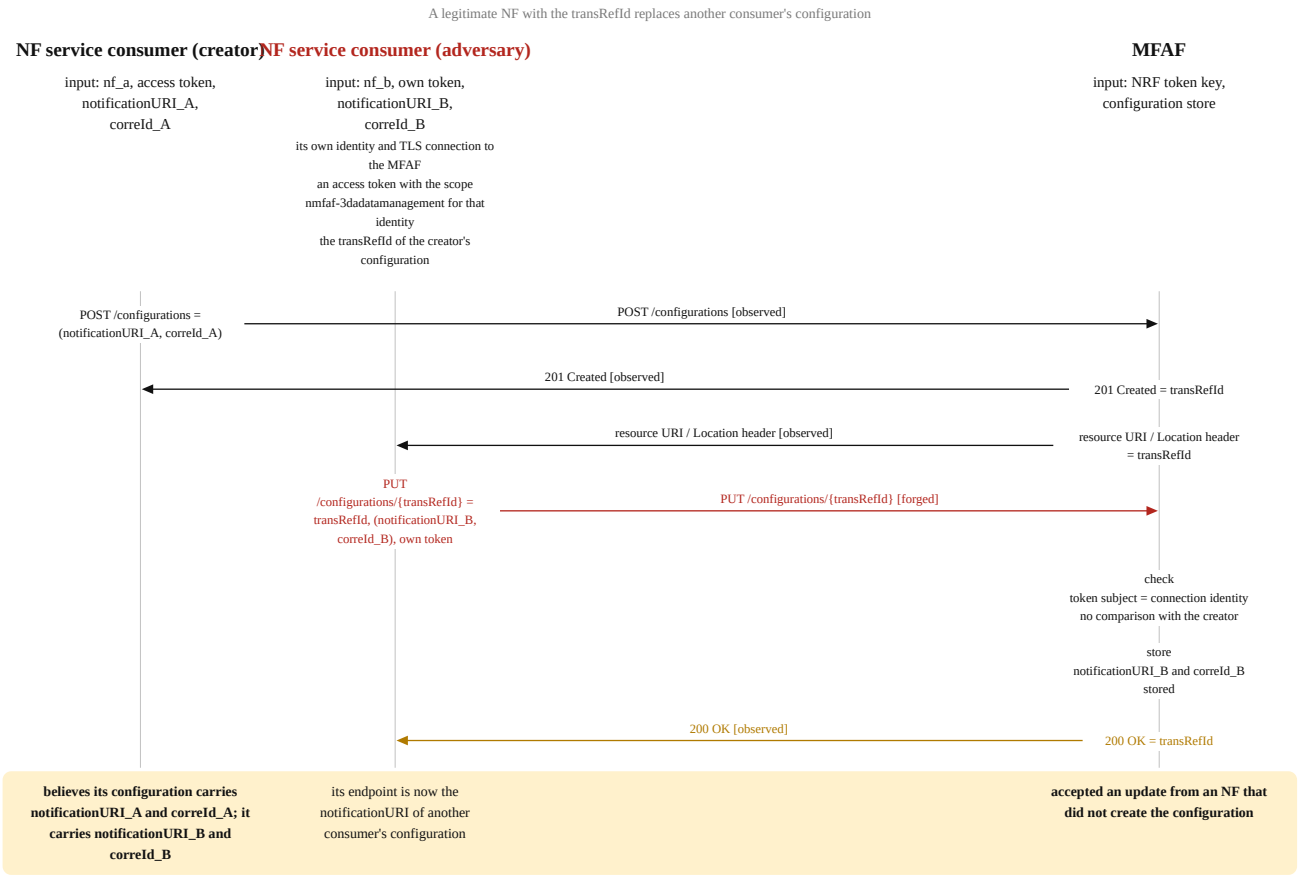


Figure 3. A legitimate NF with the transRefId replaces another consumer's configuration. Property Z03.9; attack query z03_u_ctl.

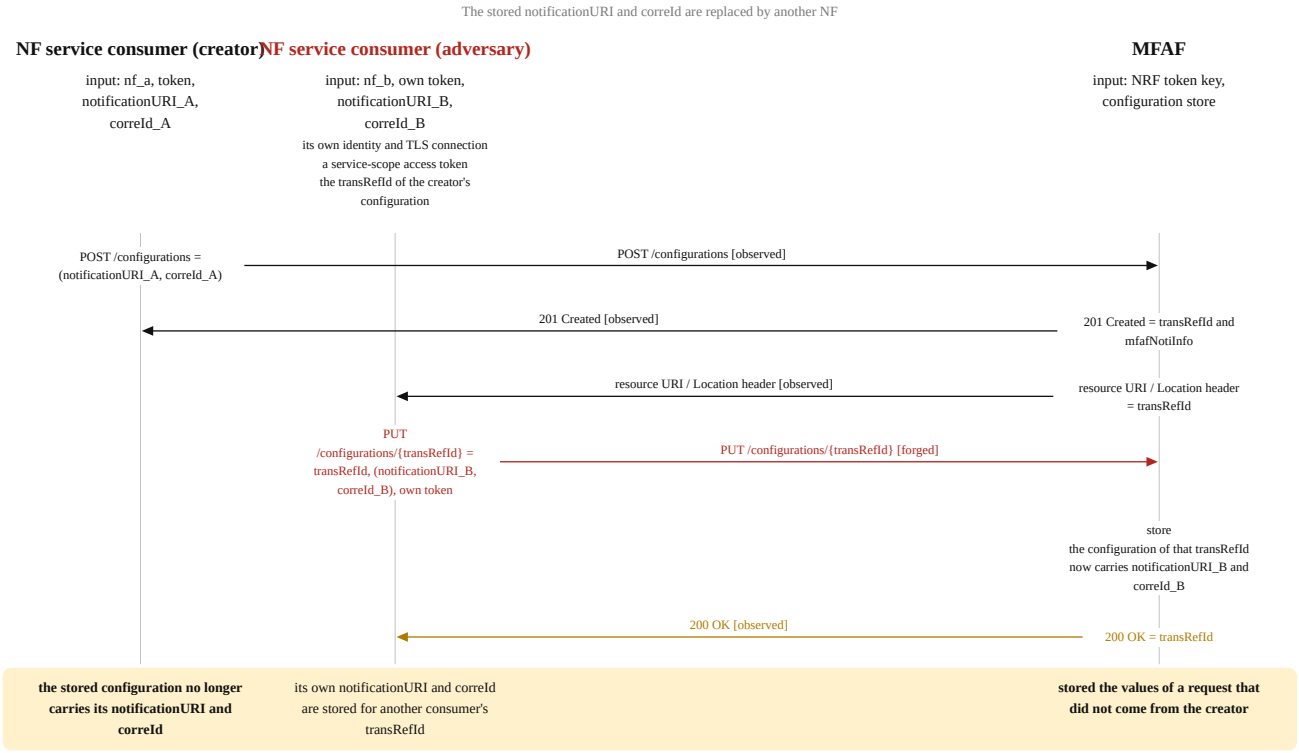


Figure 4. The stored notificationURI and correId are replaced by another NF. Property A08; attack query z03_u_ctl.

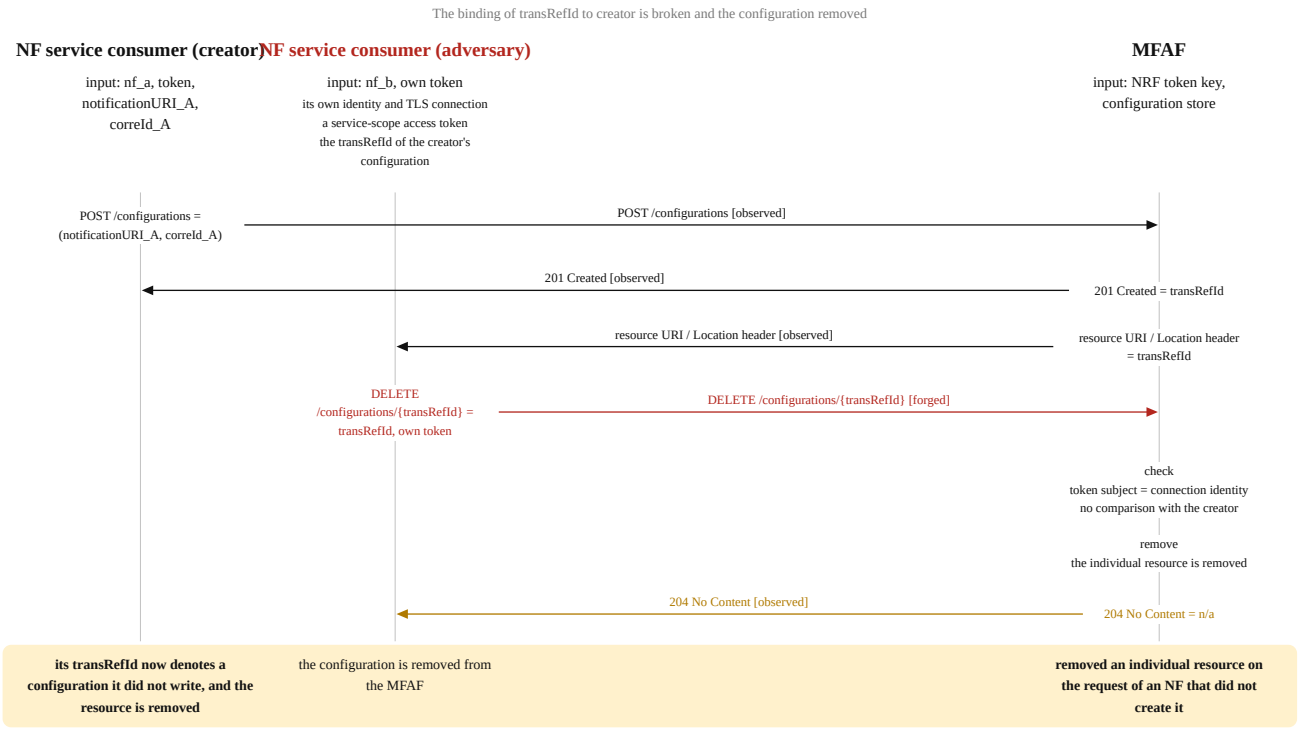


Figure 5. The binding of transRefId to creator is broken and the configuration removed. Property A08.8; attack query z03_d_ctl.

Proposed change. The MFAF shall store, with each Individual MFAF Configuration, the NF instance identifier of the consumer that created it, and shall reject a PUT or a DELETE with 403 Forbidden when the subject of the access token differs from that identifier. Per-resource authorization also requires more than the single service-wide scope of clause 5.1.9.

Check of the change with ProVerif. Not checked: This environment has no check_fix tool, so the change was not run on the model. It is a check on a value the source already has at creation time plus one comparison, it does not touch the messages, and it removes the attack.

No figure was given for these attacks of a finding: A08.8: z03_u_ctl, A08: z03_d_ctl, Z03.9: z03_d_ctl.

3 Properties examined

Each row is one property put to ProVerif. The result is ProVerif's answer on the analyst's model of the protocol; value and whether the documents already state the result are the analyst's judgement, with the reasons in Appendix A.

ID	Property	Compromise assumed	Result	Value	In the documents
A03	**Accepted operation matches a consumer run.** Whenever the MFAF accepts a configuration create/update/delete request and stores or removes a configuration accordingly, there is a real consumer run whose identity, operation type, transRefId, notificationURI and correId match the values the MFAF accepted locally.	no compromise	Holds	low	no

ID	Property	Compromise assumed	Result	Value	In the documents
A04	**Distinct runs for accepted operations.** Every time the MFAF accepts a create request and assigns a transRefId, it corresponds to a distinct consumer create run; an update/delete acceptance for the same transRefId corresponds to a distinct consumer run.	no compromise	No answer	low	no
A08	**Stored configuration equals request.** For every configuration the MFAF stores, the notificationURI and correId equal the values in that consumer's request; if the request does not provide mfafNotiInfo, the MFAF generates it locally and binds it to that configuration.	no compromise	Fails (1 attack)	medium	no
Z01	**Operation carries a valid token.** Every time the MFAF submits a configuration create/update/delete operation, there is an access token issued by the NRF that is valid for that consumer and carries the nmfaf-3dadatamanagement scope.	no compromise	Holds	low	no
Z03	**Only the creator changes a configuration.** For an update/delete operation submitted with a consumer's token, the target transRefId must belong to a configuration that consumer created earlier; the token cannot be used on another consumer's configuration.	no compromise	Fails (2 attacks)	medium	unclear
S05	**Removed configuration cannot be updated.** After the MFAF accepts a delete and removes a transRefId, that transRefId can no longer be used for an update, and that configuration no longer produces notifications.	no compromise	Holds within a bound (the changing value unrolled to 4 steps)	low	no
R03	**One operation one side effect.** The same logical create operation submits a configuration create side effect at most once at the MFAF; the same logical delete operation removes at most once.	no compromise	No answer	low	no
A08.8	**transRefId bound to creator and content.** For a configuration the MFAF stores, the binding of the transRefId to the creator identity, the notificationURI and the correId matches the consumer's request.	no compromise	Fails (1 attack)	medium	unclear

ID	Property	Compromise assumed	Result	Value	In the documents
Z03.9	**Other NF cannot replace configuration.** For an update/delete operation submitted with a consumer's token, the target transRefId must belong to a configuration that consumer created earlier; another legitimate NF cannot replace or remove that configuration.	no compromise	Fails (1 attack)	medium	unclear
A03.10	**Confirmation follows the state change.** The MFAF returns update success (200/204) if and only if the configuration for the corresponding transRefId is actually modified; it returns delete success (204) if and only if the corresponding resource is actually removed.	no compromise	Holds	low	no
X01	**Configuration content and token confidential in transit.** 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.	A valid OAuth2 access token with the nmfaf-3dadatamanagement service scope for its own NF instance and a TLS session to the MFAF under its own identity, and the transRefId of another consumer's Individual MFAF Configuration, obtained after that configuration was created (from an intermediary that handled the resource URI, from operational logs, or as a resource identifier the source keeps no secret).	Holds	low	no
X02	**Strong secrecy of the notification endpoint.** An attacker cannot tell which of two notificationURI values a consumer configured: nothing the attacker sees on the SBI depends on which of the two the configuration carries.	no compromise	Holds	none	no
X03	**Update or delete only for an existing configuration.** The MFAF accepts an update or a removal only for a transRefId that a create request really produced: a request carrying a transRefId that no configuration of the MFAF carries is rejected, and no confirmation is returned for it.	A valid OAuth2 access token with the nmfaf-3dadatamanagement service scope for its own NF instance and a TLS session to the MFAF under its own identity, and the transRefId of another consumer's Individual MFAF Configuration, obtained after that configuration was created (from an intermediary that handled the resource URI, from operational logs, or as a resource identifier the source keeps no secret).	Holds	low	no

Goals the documents state that the drafted list of properties did not have, added by the analyst (Appendix D): X01, X03.

Properties of the catalogue that the documents do not speak of, added by the analyst (Appendix D): X02 (K03).

The questions asked, and the analyst's answers:

Question	Answer	Rests on
Q1: Binding of the configuration transaction reference to the creator/notificationURI/correId	The transRefId is bound to the creator and to the notificationURI and correId of the creating request, and nothing changes those values while the transRefId is unknown; but the MFAF records no creator and checks none, so another NF that knows the transRefId rebinds the configuration to itself and can then remove it.	A08.8, A08, Z03
Q2: Whether another legitimate NF can replace/remove an existing configuration	Yes. With its own service-scope token and the transRefId of the configuration, another legitimate NF has its PUT and DELETE accepted, and the first consumer does not notice; the API defines no ownership check and no resource-level scope.	Z03.9, Z03, A08
Q3: Whether update confirmation and deregistration behaviour are tied to the exact state	For a configuration that exists, a 200 OK or 204 No Content follows the change and a 204 for a DELETE follows the removal; the source says nothing about what is answered for a transRefId that does not exist or was removed, and the model refuses such requests without any answer.	A03.10, S05, X03

4 Limits of this analysis

A result here is a result on a model of the protocol: it says what the specified messages and checks give or fail to give, with cryptography taken as ideal, and nothing about an implementation.

What this analysis does not cover, as the analyst states it:

- TLS is modelled as an authenticated encrypted channel between the NF instance and the MFAF, keyed by the NF identity: record-layer replay protection, connection handling and certificate validation are assumed, not modelled.
- The access token is a table entry of the NRF with its subject and the service scope; token format, expiry, audience validation beyond the single MFAF, revocation and token stealing are not modelled.
- The configuration store only grows: removal is a flag, and the content stored by an update is not modelled as replacing the earlier content, so the order of two writes cannot be expressed.
- Notifications are not modelled, so the part of S05 about a removed configuration producing no notification is outside the model.
- Injectivity of request to acceptance (targets A04 and R03) has no answer: a message on a channel stays readable to every process that knows the channel, so a repeated request can be accepted twice in the model. A stateful model such as Tamarin+ Agent would settle it.
- mfafTransfer, the notification endpoints of the DataAnaCollect feature, redirects through an SCP, error responses (no application errors are defined, s0223) and feature negotiation are not modelled.
- The NRF is modelled only as the entity that issued the tokens; its procedures and the discovery that precedes the call (s0231) are out of scope.

What the host records:

- No answer for: A04, R03.
- The answers of S05, X03 rest on an operation the model simplifies (Appendix D).
- The prover gave no answer (Appendix C): A04: a04_c, R03: r03_c. The property was asked and the prover neither proved it nor found an attack, so the property is shown without an answer.
- Proved without a control query (Appendix 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. A control is a variant of the property that must fail; without one, nothing shows that the property as written can fail in this model.

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

Appendix A: Reading of each property

The analyst's own reading of the prover's answers, not an answer of the prover: what the answers mean for the protocol, what the reading rests on, whether it can be used as it stands, and what it is worth.

A03: Accepted operation matches a consumer run

Whenever the MFAF accepts a configuration create/update/delete request and stores or removes a configuration accordingly, there is a real consumer run whose identity, operation type, transRefId, notificationURI and correId match the values the MFAF accepted locally.

Every configuration operation the MFAF commits corresponds to a request a real consumer run sent, with the same identity, transRefId and configured values. Nothing on the network can make the MFAF act on values no consumer sent.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: A base correspondence that the transport and token requirements imply; it confirms the model is not obviously wrong rather than showing anything new.
- Stated in the documents: no. Source text it rests on: s0030, s0048, s0052, s0061, s0076
- Rests on: TLS is an authenticated encrypted channel per NF identity, so a request cannot be forged or altered on the wire.; The token is checked by subject only; audience and expiry are not modelled.
- Next step: Nothing further; the identity part of the claim is answered by Z03 and A08.8, where it fails for another NF.

A04: Distinct runs for accepted operations

Every time the MFAF accepts a create request and assigns a transRefId, it corresponds to a distinct consumer create run; an update/delete acceptance for the same transRefId corresponds to a distinct consumer run.

The claim that each accepted create stands for a different consumer run is not answered: no trace was found and none was excluded. On the source, a replayed create request would create a second configuration, because the MFAF assigns a new transRefId per request and no idempotency key is defined.

- Result: No answer; answers what is asked: undetermined; not usable
- Value low: No answer was obtained; it points at the one question about this API that a stateful model would have to settle.
- Stated in the documents: no. Source text it rests on: s0049, s0053, s0075
- Rests on: The transport is modelled without a record layer, so repeated requests cannot be excluded.; The source says nothing about retries or idempotency.
- Open: Whether one create request can produce two configurations is not settled: the model's channels are not consumptive, so injectivity stays inconclusive.
- Next step: A stateful prover such as Tamarin+ Agent, with a linear request store, or a clarification of the source on retries.

A08: Stored configuration equals request

In every configuration the MFAF stores, the notificationURI and correId equal the values in that consumer's request; if the request does not provide mfafNotiInfo, the MFAF generates it locally and binds it to that configuration.

The MFAF stores the notificationURI and correId of the consumer's request, and the MFAF notification information it determines itself is bound to one configuration. It is not the MFAF that changes these values: another NF with the transRefId and its own token replaces them, and the creator's configuration then points where the other NF chose.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: The integrity of the stored values holds against the network and fails against another NF, with the exact step where the replacement happens.
- Stated in the documents: no. Source text it rests on: s0039, s0040, s0050, s0052, s0193
- Rests on: The store records what was accepted; replacing content in place is modelled only through the accepting event.; Notifications are not modelled, so the effect on delivery is argued, not shown.
- Open: The content a later update leaves behind is not modelled as replacing the earlier content, so only the values of the accepted request are shown.
- Next step: A model that also delivers notifications to the notificationURI would show the delivered data going to the wrong endpoint.

Z01: Operation carries a valid token

Every time the MFAF submits a configuration create/update/delete operation, there is an access token issued by the NRF that is valid for that consumer and carries the nmfaf-3dadatamanagement scope.

No configuration operation is accepted without a token the NRF issued for the identity that presents it, and the service scope is the only one the API defines. The token therefore does not say which resource the consumer may act on, which is why the ownership property fails.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: It confirms the authorization the source requires is enforced; its interest is what it does not cover, namely the resource.
- Stated in the documents: no. Source text it rests on: s0229, s0230, s0232
- Rests on: Tokens are table entries with subject and scope; expiry, revocation and audience are not modelled.; The TLS connection identity is taken to be the NF instance identity.
- Next step: Nothing further for this claim.

Z03: Only the creator changes a configuration

An update/delete operation submitted with a consumer's token must target a transRefId that belongs to a configuration that consumer created earlier; the token cannot be used on another consumer's configuration.

Against the network alone another party cannot change a configuration: an accepted update by a different identity happens only once the transRefId is disclosed. Against another legitimate NF that knows the transRefId, the update is accepted and the configuration is replaced, because the MFAF compares the token with the connection identity and never with the creator of the resource.

- Result: Fails (2 attacks); answers what is asked: yes; usable
- Value medium: A protection the API appears to intend does not hold for a legitimate NF, and the exact condition it needs is stated; the source does not state it.

- Stated in the documents: unclear. Source text it rests on: s0232, s0060, s0061, s0076
- Rests on: The transRefId is unguessable in the model and its disclosure is a separate step.; The malicious NF is a legitimate instance with its own token and TLS credentials.
- Open: The attack needs the transRefId; the model provides it after the run, and the source neither publishes nor protects it.
- Next step: Ask the specification editors whether an ownership check is intended elsewhere in the baseline, and whether the transRefId is meant to be secret.

S05: Removed configuration cannot be updated

After the MFAF accepts the deletion and removes a transRefId, that transRefId can no longer be used for an update, and the configuration no longer produces notifications.

The unrolled lifecycle shows that once the MFAF has accepted the DELETE, a stale PUT of the same transRefId is refused and no confirmation is produced, so the removal does not resurrect the configuration. This holds for four unrolled steps of one configuration.

- Result: Holds within a bound (the changing value unrolled to 4 steps); answers what is asked: yes; usable with caveats
- Value low: A property the source states indirectly by removing the resource; the model confirms it for the unrolled lifecycle without finding anything new.
- Stated in the documents: no. Source text it rests on: s0076, s0007, s0023
- Rests on: Only four steps of one configuration are unrolled; the refusal rests on the removal flag, modelled as an insert-only fact.; Notification delivery is not modelled.
- Open: The claim that a removed configuration produces no notifications is outside the model.
- Next step: A model with notification delivery would settle the second half of the claim.

R03: One operation one side effect

The same logical create operation submits at most one configuration create side effect to the MFAF; the same logical delete operation removes at most once.

Whether a repeated create request produces a second configuration, or a repeated DELETE a second removal, is not answered. The model shows no trace of two acceptances from one request but does not exclude one. On the source, nothing forbids a repeated POST from creating a second configuration, since no idempotency key or retry rule is defined.

- Result: No answer; answers what is asked: undetermined; not usable
- Value low: No answer was obtained; the useful outcome is that the source defines no retry or idempotency rule for POST.
- Stated in the documents: no. Source text it rests on: s0048, s0049, s0052, s0076
- Rests on: No record-layer model, so replay is neither allowed nor excluded.; The source defines no idempotency key.
- Open: At most one side effect per logical operation is not settled by ProVerif here.
- Next step: Tamarin+ Agent with a linear request log, or a statement in the source on retries.

A08.8: transRefId bound to creator and content

In the configuration the MFAF stores, the binding of the transRefId to the creator's identity, the notificationURI and the correId matches the consumer's request.

At creation the transRefId is bound to the creating consumer and to the values it sent, and while the transRefId is unknown no other NF can change that binding. With the transRefId known, the second NF replaces the values and then removes the configuration, without notice to the creator.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: It localises the weakness: the binding exists in the creator's and the MFAF's knowledge, not in the stored record.
- Stated in the documents: unclear. Source text it rests on: s0049, s0138, s0193
- Rests on: The absence of a creator field is read from the data type definition, which lists no such attribute.; The transRefId disclosure is a separate step in the model.
- Open: Whether the MFAF keeps the creator in its record is not stated; the model assumes it does not, because no such attribute exists.
- Next step: An editor's clarification of whether the MFAF is expected to keep the creator with the configuration.

Z03.9: Other NF cannot replace configuration

An update/delete operation submitted with a consumer's token must target a transRefId that belongs to a configuration that consumer created earlier; another legitimate NF cannot replace or remove that configuration.

A malicious legitimate NF cannot replace or remove a configuration it did not create while it does not know the transRefId. Once it knows the transRefId, both the update and the removal are accepted, so the only protection is the identifier itself.

- Result: Fails (1 attack); answers what is asked: yes; usable
- Value medium: It answers whether an insider NF can act on another consumer's configuration, with the exact precondition.
- Stated in the documents: unclear. Source text it rests on: s0232, s0060, s0076
- Rests on: The disclosure of the transRefId is a separate step, standing for an intermediary that handled the URI or for operational logs.; Only one other NF identity is modelled.
- Open: The attacker needs the transRefId; without it the property holds.
- Next step: Clarify how the transRefId is protected in deployment, for example whether it is treated as a bearer capability.

A03.10: Confirmation follows the state change

The MFAF returns update success (200/204) if and only if the configuration for the corresponding transRefId is actually modified; it returns delete success (204) if and only if the corresponding resource is actually removed.

A 200 OK or 204 No Content is produced only after the MFAF has applied the update, and a 204 for a DELETE only after the resource was removed. The acknowledgement therefore means what the consumer reads into it, for the requests the MFAF accepts.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: Confirms the reading of the acknowledgements the source specifies, without finding anything new.
- Stated in the documents: no. Source text it rests on: s0063, s0064, s0065, s0076, s0223

- Rests on: The confirmation follows the state change in the same process; atomicity with the state change is structural in the model.; No application errors are modelled (s0223).
- Open: The source defines no answer for a transRefId that does not exist, so the failure side of the equivalence is not modelled.
- Next step: A specification statement on the error answers for unknown or removed transRefIds would complete the equivalence.

X01: Configuration content and token confidential in transit

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.

The configuration content and the access token stay inside the consumer's own TLS session: neither the network nor another NF obtains the notificationURI, the correId or the token of another consumer, and nothing the attacker sees tells him which of two notificationURI values was configured.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: It follows from the transport assumption; its value is to show that the confidentiality of the content is not what fails here.
- Stated in the documents: no. Source text it rests on: s0193, s0230
- Rests on: This is the model of TLS: an encrypted channel keyed by the NF identity. It is an assumption of the analysis, not a result about TLS.
- Next step: Nothing further within this scope.

X02: Strong secrecy of the notification endpoint

An attacker cannot tell which of two notificationURI values a consumer configured: nothing the attacker sees on the SBI depends on which of the two the configuration carries.

An attacker cannot tell which of two notificationURI values a consumer configured: the value is only ever used inside the consumer's encrypted session, so strong secrecy of the configured endpoint holds against the network.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value none: A property nobody asks of this API; it follows from the transport assumption alone.
- Stated in the documents: no. Source text it rests on: s0193
- Rests on: The same TLS assumption as X01.; Two candidate values in one configuration; the claim speaks of the runs that are written.
- Next step: Nothing further.

X03: Update or delete only for an existing configuration

The MFAF accepts an update or a removal only for a transRefId that a create request really produced: a request carrying a transRefId that no configuration of the MFAF carries is rejected, and no confirmation is returned for it.

An update or a removal is accepted only for a transRefId that a create request really produced, whoever sends the request, so a consumer cannot make the MFAF act on a resource identifier it invented. Such requests are refused and confirmed to nobody.

- Result: Holds; answers what is asked: yes; usable with caveats

- Value low: A validation requirement of TS 33-501 that the API satisfies structurally; it shows the identifier space is enforced.
- Stated in the documents: no. Source text it rests on: s0144, s0059, s0075
- Rests on: The store of existing transRefIds is inserted at creation.; The failure answer is not modelled because the source defines none.
- Open: The response to a request for an unknown transRefId is not defined, so the model's refusal answers nothing.
- Next step: Nothing further.

Appendix C: Claims and prover answers

The prover's record for each target: the level reached, the query that reached it, whether a control was checked, and the attacks with the analyst's verdict on each.

target	level	bound	lemma	control	attacks	claims
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

Claim by claim, in the analyst's wording of each claim; the answer to each is the prover's:

target	claim	attacker	answer	query	control
A03	Every operation the MFAF accepts (create, update, delete) corresponds to a real consumer run of that identity with the same transRefId, notificationURI and correId.	`network`	proved	a03_c	none
A03	The update and the removal the MFAF accepts are the consumer's own request with the same transRefId and values.	`network`	proved	a03_u	none
A03	The removal the MFAF performs is the consumer's own delete request with the same transRefId.	`network`	proved	a03_d	none
A03	Every accepted operation corresponds to a real consumer run of the identity the MFAF authenticates, with the same transRefId, notificationURI and correId.	`nf_with_transrefid`	proved	a03_c	none
A04	Every accepted create corresponds to a different consumer create run, so a recorded create request is not counted twice.	`network`	undecided	a04_c	

target	claim	attacker	answer	query	control
A08	The notificationURI and correId the MFAF stores at creation are those of the creating consumer's request.	`network`	proved	a03_c	none
A08	The mfafNotiInfo the MFAF determines is bound to a single configuration: two configurations never carry the same MFAF notification information.	`network`	proved	a08_m	none
A08	The configuration keeps the notificationURI and correId of the consumer that created it; no other NF can change them.	`nf_with_transrefid`	attack (holds_in_source)	z03_u_ctl (the control of z03_u)	
Z01	Every operation the MFAF commits on a configuration comes with an access token that the NRF issued for that same NF identity and the service scope.	`network`	proved	z01_c	none
Z01	Every update and every removal the MFAF commits comes with a token the NRF issued for that NF identity and the service scope.	`network`	proved	z01_u	none
Z01	Every operation the MFAF commits is accompanied by a token issued for the NF identity it is presented with; the attacker's own token cannot be used under another identity.	`nf_with_transrefid`	proved	z01_c	none
Z03	Another NF cannot replace the configuration of the consumer that created it: an accepted update of a transRefId by a different identity happens only if that transRefId has been disclosed, and the network attacker holds nothing of a party.	`network`	proved	z03_u	checked
Z03	Another NF cannot remove the configuration of the consumer that created it: an accepted removal by a different identity happens only if that transRefId has been disclosed.	`network`	proved	z03_d	checked
Z03	Another NF cannot replace the configuration of its creator, even when it holds a valid service-scope token and knows the transRefId.	`nf_with_transrefid`	attack (holds_in_source)	z03_u_ctl (the control of z03_u)	
Z03	Another NF cannot remove the configuration of its creator, even when it holds a valid service-scope token and knows the transRefId.	`nf_with_transrefid`	attack (holds_in_source)	z03_d_ctl (the control of z03_d)	
S05	After the MFAF has removed a transRefId, a stale update of that transRefId is not accepted; the unrolled lifecycle of one configuration (create, update, remove, stale update) refuses it.	`network`	bounded (the changing value unrolled to 4 steps)	s05	checked
S05	After the MFAF has removed a transRefId, neither the creator nor another NF can obtain an update of it.	`nf_with_transrefid`	bounded (the changing value unrolled to 4 steps)	s05	checked
R03	One logical create request produces at most one configuration, and one logical delete request at most one removal.	`network`	undecided	r03_c	
A08.8	At creation the MFAF binds the transRefId to the creating consumer and to that request's notificationURI and correId, and later changes by another NF do not occur unless the transRefId is disclosed.	`network`	proved	z03_u	checked

target	claim	attacker	answer	query	control
A08.8	The binding between transRefId, creating consumer and configured values cannot be broken by another NF.	`nf_with_transrefid`	attack (holds_in_source)	z03_d_ctl (the control of z03_d)	
Z03.9	A malicious legitimate NF cannot replace a configuration it did not create: over the network alone, an update accepted for another NF's transRefId requires that transRefId to have been disclosed.	`network`	proved	z03_u	checked
Z03.9	A malicious legitimate NF cannot replace or remove a configuration it did not create, even holding its own token.	`nf_with_transrefid`	attack (holds_in_source)	z03_u_ctl (the control of z03_u)	
A03.10	An update confirmation is sent only after the configuration was changed, so a 200 OK means the update was applied.	`network`	proved	a0310_u	none
A03.10	A delete acknowledgement is sent only after the individual resource was removed, so a 204 means the removal was applied.	`network`	proved	a0310_d	none
A03.10	A 200 OK or 204 No Content is returned only for an update that the MFAF really applied, whoever sent it.	`nf_with_transrefid`	proved	a0310_u	none
X01	The notificationURI of the configuration stays inside the consumer's TLS session: the network attacker does not learn it.	`network`	proved	x01_nu	none
X01	The correId of the configuration stays inside the consumer's TLS session: the network attacker does not learn it.	`network`	proved	x01_ci	none
X01	The access token of a consumer is not obtained by another NF: it travels only inside that consumer's own TLS session.	`nf_with_transrefid`	proved	x01_tk	none
X02	Nothing the attacker sees tells him which of two notificationURI values a consumer configured.	`network`	proved	x02	none
X03	An update is accepted only for a transRefId a create request really produced, and a confirmation is sent only then.	`network`	proved	x03_u	none
X03	An update or a removal is accepted only for a transRefId a create request really produced, so a nameless transRefId is rejected.	`nf_with_transrefid`	proved	x03_d	none

Appendix D: How the analysis was run

An LLM analyst builds the model of the protocol from the documents and puts each property to ProVerif; every result in section 3 is an answer of the prover.

ProVerif+ Agent session session-20261114T231509, 2026-11-14. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 43, prover 4 s, wall 22 min, cost 0.132 USD.

ProVerif: 65 proofs run, 60 decided, 5 without an answer, 4 s, 29 answers recorded.

What was analysed

Protocol	MFAF configuration lifecycle
Version	19.7.0
Analysed	One DCCF/NWDAF and the MFAF; creating, updating and deleting a single messageConfiguration with a fixed notificationURI and correId, excluding mfafTransfer and the actual forwarding of multi-source data.
Not analysed	The symbolic model does not prove the computational or post-quantum security of the algorithms; an attack trace holds only under explicit assumptions and is not directly the same as a specification vulnerability. Every property must have a reachability/positive control. Rel-19 is fixed. A distinction is made between an external network attacker and a malicious legitimate NF that holds its own credentials; TLS authenticated encryption is modelled per endpoint, and a legitimate NF cannot read another NF's TLS session. OAuth2 subject/audience/scope/expiry and TLS identity binding follow the attached baseline; a service scope does not automatically equal resource ownership. A monotonic table cannot model real deletion, and the finite-state lifecycle must be explicitly delimited. The goal is the specified API sub-flow, not complete 5G security. The native Word tables in the previous delivery had cross-clause alignment errors; this time the same scope was re-parsed from the unchanged manuscript and independently verified. The old model is an artifact awaiting review; its old mismatched sources must not be treated as the specification or used to directly claim that the original conclusions are false.
Source text	260 numbered passages (114238 bytes) from the sections listed below. A reference such as s0001 in this report is one of these passages

The sections of the documents the analysis is built from:

Document	Section	Title
main	4.2	Nmfaf_3daDataManagement Service
main	4.2.1	Service Description
main	4.2.1.1	Overview
main	4.2.1.2	Service Architecture
main	4.2.1.3	Network Functions
main	4.2.1.3.1	Messaging Framework Adaptor Function (MFAF)
main	4.2.1.3.2	NF Service Consumers
main	4.2.2	Service Operations
main	4.2.2.1	Introduction
main	4.2.2.2	Nmfaf_3daDataManagement_Configure service operation
main	4.2.2.2.1	General

Document	Section	Title
main	4.2.2.2.2	Initial configuration of the MFAF
main	4.2.2.2.3	Update the configuration of the MFAF
main	4.2.2.3	Nmfaf_3daDataManagement_Deconfigure service operation
main	4.2.2.3.1	General
main	4.2.2.3.2	Remove MFAF configuration
main	5.1	Nmfaf_3daDataManagement Service API
main	5.1.1	Introduction
main	5.1.2	Usage of HTTP
main	5.1.2.1	General
main	5.1.2.2	HTTP standard headers
main	5.1.2.2.1	General
main	5.1.2.2.2	Content type
main	5.1.2.3	HTTP custom headers
main	5.1.3	Resources
main	5.1.3.1	Overview
main	5.1.3.2	Resource: MFAF Configurations
main	5.1.3.2.1	Description
main	5.1.3.2.2	Resource Definition
main	5.1.3.2.3	Resource Standard Methods
main	5.1.3.2.3.1	POST
main	5.1.3.2.4	Resource Custom Operations
main	5.1.3.3	Resource: Individual MFAF Configuration
main	5.1.3.3.1	Description
main	5.1.3.3.2	Resource Definition
main	5.1.3.3.3	Resource Standard Methods
main	5.1.3.3.3.1	PUT
main	5.1.3.3.3.2	DELETE
main	5.1.4	Custom Operations without associated resources
main	5.1.5	Notifications
main	5.1.6	Data Model
main	5.1.6.1	General
main	5.1.6.2	Structured data types
main	5.1.6.2.1	Introduction
main	5.1.6.2.2	Type: MfafConfiguration
main	5.1.6.2.3	Type: MessageConfiguration
main	5.1.6.2.4	Type: MfafNotiInfo

Document	Section	Title
main	5.1.6.2.5	Type: MfafTransferInfo
main	5.1.6.2.6	Type: TargetMfafNotiInfo
main	5.1.6.3	Simple data types and enumerations
main	5.1.6.3.1	Introduction
main	5.1.6.3.2	Simple data types
main	5.1.6.4	Data types describing alternative data types or combinations of data types
main	5.1.6.5	Binary data
main	5.1.7	Error Handling
main	5.1.7.1	General
main	5.1.7.2	Protocol Errors
main	5.1.7.3	Application Errors
main	5.1.8	Feature negotiation
main	5.1.9	Security

The purpose of the protocol as the source step located it in the documents: a DCCF/NWDAF and MFAF; creating, updating and deleting a single messageConfiguration, with fixed notificationURI and correId, excluding mfafTransfer and the actual multi-source data forwarding.

The properties against the documents

Before a query was written, the analyst set the drafted list of properties against what the documents themselves say the protocol is for. The name, what a property is about and what the documents say are the analyst's reading; the places named exist in the documents.

ID	Name	About	What the documents say	Where
A03	Accepted operation matches a consumer run	The MFAF, against the NF service consumer (DCCF or NWDAF): what the MFAF accepts and stores or removes as an operation on one configuration is what a real consumer run sent for that operation, with the same identity, transRefId, notificationURI and correId.	The MFAF acts on the request of an NF service consumer: create (s0030, s0048, s0052), update (s0061, s0062), remove (s0076); the source names no authentication of the request beyond TLS and the OAuth2 token (s0229, s0230).	s0030, s0048, s0052, s0061, s0062, s0076, s0229, s0230
A04	Distinct runs for accepted operations	The MFAF: each accepted create assigns a new transaction reference id (s0049) and stands for a different consumer run; update and delete acceptances on one transRefId stand for different consumer runs.	s0049 assigns one transaction reference id per create; the source says nothing about retries or idempotency keys, so the run identity has to be fixed in the model (the target's own source_limits).	s0049, s0053, s0075
A08	Stored configuration equals request	The consumer's notificationURI and correId in the configuration the MFAF stores, and the MFAF-generated mfafNotiInfo bound to that configuration.	The request shall carry notificationURI (s0039) and, for mapping or data collection, correId (s0040); the MFAF determines mfafNotiInfo when the request has none (s0050) and stores the configuration (s0052); both attributes are mandatory in MessageConfiguration (s0193).	s0039, s0040, s0050, s0052, s0193

ID	Name	About	What the documents say	Where
Z01	Operation carries a valid token	The MFAF's assurance that an operation on a configuration comes from an NF service consumer that the NRF authorized for the service scope nmfaf-3dadatamanagement.	Access may be authorized with OAuth2 Client Credentials, NRF as authorization server (s0229), the consumer obtains a token before consuming the service (s0230), and the API has a single service scope (s0232); RFC 6749 requires that unauthorized parties cannot generate or modify valid tokens (rfc6749:2870).	rfc6749:2870, s0229, s0230, s0232
Z03	Only the creator changes a configuration	The consumer that created a configuration, against any other NF: its configuration is not replaced or removed by another NF holding its own service-scope token; the MFAF's authorization of a PUT or DELETE against the individual resource.	The API defines one scope for the entire service and no scope at resource or operation level (s0232); the clauses that say what the MFAF does on PUT and DELETE list no check that the requester created the resource (s0060, s0061, s0076); RFC 6749 states that it gives the resource server no method to ensure that a token presented to it was issued to that client (rfc6749:2879).	rfc6749:2879, s0232, s0060, s0061, s0076
S05	Removed configuration cannot be updated	The MFAF's store after Deconfigure, against the consumer that removed the configuration and against the others: the removed Individual MFAF Configuration neither accepts an update nor produces notifications.	The MFAF shall acknowledge the DELETE with 204 and remove the individual resource linked to the delete request (s0076); the service allows consumers to remove configurations from the MFAF (s0007, s0023, s0070).	s0007, s0023, s0070, s0076
R03	One operation one side effect	The MFAF's store: one logical create request stores at most one configuration, one logical delete request removes at most one resource.	One create request leads to the creation of a new configuration and one assigned transaction reference id (s0048, s0049, s0052); the source defines no idempotency key and says nothing of retries (the target's own source_limits).	s0048, s0049, s0052, s0076
A08.8	transRefId bound to creator and content	The MFAF's stored binding between the transRefId and the creating consumer identity, notificationURI and correId, which the consumer relies on when it later updates or removes the configuration.	The MFAF assigns a transaction reference id (s0049), the Individual MFAF Configuration is the configuration created in the MFAF associated with that id (s0138), and MessageConfiguration carries notificationURI and correId (s0193); the source does not say whether the creator identity is part of the stored record.	s0049, s0138, s0193
Z03.9	Other NF cannot replace configuration	The consumer that created a configuration, against a malicious legitimate NF that holds its own token: the other NF cannot replace or remove that configuration.	One service-wide scope and no resource or operation scopes (s0232); no clause requires the MFAF to check the requester's relation to the resource (s0060, s0076).	s0232, s0060, s0076
A03.10	Confirmation follows the state change	The consumer: a 200 OK or 204 No Content for a PUT means the configuration of that transRefId was modified, and a 204 for a DELETE means the resource was removed.	The MFAF answers 200 OK or 204 No Content after it updated the resource (s0063, s0064, s0065) and 204 after it removed the individual resource (s0076); no application errors are defined for the service (s0223).	s0063, s0064, s0065, s0076, s0223
X01	Configuration content and token confidential in transit	The configuration content and the OAuth2 access token of an NF service consumer in transit on the SBI towards the MFAF.	Catalogue entry K02 (payload and end-to-end confidentiality) and K01 (credential confidentiality); RFC 6749 requires access token credentials to be kept confidential in transit and transmitted only over TLS (rfc6749:2859, rfc6749:3013).	rfc6749:2859, s0193, s0230

ID	Name	About	What the documents say	Where
X02	Strong secrecy of the notification endpoint	The notificationURI of a configuration, as seen by an attacker on the network.	Catalogue entry K03 (strong secrecy and indistinguishability); the source says nothing of notificationURI secrecy, so this is asked because the protocol has a value that could be hidden.	
X03	Update or delete only for an existing configuration	The MFAF's store against a consumer, legitimate or not, that names a transRefId the MFAF never assigned.	TS 33-501 requires each NF to validate all incoming messages and to reject messages that are not valid according to the protocol specification and network state (ts-33-501:2875); the Individual MFAF Configuration resource is identified by a transRefId that a create request produced (s0144, s0117, s0059, s0075).	ts-33-501:2875, s0144, s0059, s0075

Added from the documents by the analyst: X01, X03.

The security statements of the documents, paragraph by paragraph

The host listed the 398 paragraphs of the sections in which the documents say what the protocol protects and against what, and the analyst said of each what it made of it: a property, an attacker, or why neither. Taken up: 12. Not taken: 386. Which paragraphs are listed follows from the layout of the standard; where each went is the analyst's judgement.

Paragraphs	Where	Taken up as	Why not
P001 to P002	rfc6749:2778-2786	No goal stated	Introductory remarks on the OAuth2 security model that refer to the threat model document; they state no property of this API flow.
P003	rfc6749:2790-2795	Z01	
P004 to P006	rfc6749:2804-2823	Outside the scope	About client credentials for native and user-agent based clients; the consumers of this API are NFs using the client credentials grant.
P007	rfc6749:2827-2829	Z03, A03	
P008	rfc6749:2831-2838	Z01	
P009 to P010	rfc6749:2840-2850	Outside the scope	About resource owner authentication and repeated authorization requests at the authorization endpoint; the client credentials grant has no resource owner.
P011	rfc6749:2859-2865	X01	
P012	rfc6749:2867-2868	Outside the scope	About the implicit grant type, which is not used for this service.
P013	rfc6749:2870-2872	Z01	
P014	rfc6749:2874-2877	Z03	
P015	rfc6749:2879-2881	Z03, Z03.9	
P016 to P021	rfc6749:2885-2917	Outside the scope	Refresh tokens; the client credentials grant used here issues no refresh token.

Paragraphs	Where	Taken up as	Why not
P022 to P025	rfc6749:2921-2943	Outside the scope	Authorization codes; not used by this grant type.
P026 to P029	rfc6749:2947-2980	Outside the scope	Redirection URI manipulation at the authorization endpoint; no redirection URI is used for this service.
P030 to P033	rfc6749:2984-3003	Outside the scope	Resource owner password credentials; not used by this grant type.
P034 to P035	rfc6749:3013-3019	X01	
P036	rfc6749:3023-3028	A03	
P037 to P039	rfc6749:3032-3041	Outside the scope	Guessing of client credentials at the authorization server; this flow carries no client secret and the authorization server is not modelled.
P040 to P042	rfc6749:3045-3066	Outside the scope	Phishing of resource owners; the client credentials grant has no resource owner and no redirect.
P043 to P048	rfc6749:3070-3110	Outside the scope	Cross-site request forgery in browser based clients; NFs are not browsers.
P049 to P050	rfc6749:3119-3138	Outside the scope	Clickjacking; no user agent takes part in this flow.
P051 to P052	rfc6749:3142-3150	Outside the scope	Code injection into clients; no code is received in this flow.
P053 to P054	rfc6749:3154-3167	Outside the scope	Open redirectors at the authorization endpoint; not used here.
P055 to P062	rfc6749:3170-3210	Outside the scope	Misuse of access tokens in the implicit grant to impersonate a resource owner; there is no resource owner and no implicit grant here.
P063	ts-33-501:2491-2491	Outside the scope	Bidding down attacks between UE and network; no security feature negotiation of that kind is in this API flow.
P064 to P213	ts-33-501:2495-2853	Outside the scope	Requirements on subscriber authentication, RAN, UE, gNB and subscriber data protection; the analysed flow is an SBI API of the MFAF and none of these parties takes part in it.
P214	ts-33-501:2859-2859	No goal stated	States the trust-zone assumption and refers to the following sub-clauses; it states no property of its own for this API.
P215 to P217	ts-33-501:2865-2869	Outside the scope	Confidentiality, integrity and replay protection of service registration and discovery, authorization of discovery requests, and topology hiding; discovery and registration are not part of the modelled flow.
P218	ts-33-501:2871-2871	A03, Z01	
P219	ts-33-501:2873-2873	Outside the scope	Note that there is no mutual authentication with indirect communication via SCP or SEPP; indirect communication is left out of the scope.
P220	ts-33-501:2875-2875	A03, S05, X03	
P221 to P227	ts-33-501:2879-2891	Outside the scope	Security requirements on the NRF platform (registration data protection, authorization of registrations); the NRF appears here only as the token issuer and is not modelled.
P228 to P234	ts-33-501:2895-2907	Outside the scope	NEF security requirements; the NEF takes no part in this flow.

Paragraphs	Where	Taken up as	Why not
P235 to P241	ts-33-501:2911-2923	Outside the scope	Requirements on the SCP; indirect communication through an SCP is left out of the scope.
P242 to P254	ts-33-501:2929-2953	Outside the scope	General inter-PLMN and SEPP requirements; no inter-PLMN communication is modelled.
P255 to P292	ts-33-501:2957-3031	Outside the scope	SEPP requirements for roaming; the SEPP takes no part in this flow.
P293 to P303	ts-33-501:3035-3055	Outside the scope	Messages generated by roaming intermediaries; no roaming intermediary is modelled.
P304 to P312	ts-33-501:3059-3075	Outside the scope	Protection of specific attributes across trust boundaries; the flow stays inside one trust domain and carries no attributes of that kind.
P313 to P314	ts-33-501:3079-3081	Outside the scope	IPUPS functionality; not part of this flow.
P315 to P318	ts-33-501:3085-3091	Outside the scope	The 3gpp-Sbi-Originating-Network-Id header and PLMN handling; the modelled flow has no inter-PLMN messaging and no property of this API is stated.
P319 to P327	ts-33-501:3097-3115	Outside the scope	Configuration and streaming of signalling monitoring data; not part of this flow.
P328 to P337	ts-33-501:3121-3141	Outside the scope	Security visibility and configurability; deployment and management requirements of the operator, not properties of this API flow.
P338 to P369	ts-33-501:3149-3215	Outside the scope	Requirements on ciphering and integrity algorithm identifiers and their selection; the strength of the primitives is outside a symbolic model and outside this API.
P370 to P375	ts-33-501:3219-3231	Outside the scope	Requirements on 5G-RG and NSSAAF; neither takes part in this flow.
P376 to P380	ts-33-501:6540-6548	Outside the scope	Security requirements on N3; N3 is a user-plane interface, not part of this API.
P381 to P398	ts-33-501:9656-11901	Outside the scope	Annexes on EAP TLS, IAB, MBSF and MBSTF; none of these parties or procedures takes part in this flow.

The catalogue of properties, entry by entry

The properties were drafted from a catalogue of security and privacy properties. The draft left out 35 of its entries, and the analyst went through each of them: examined here as a property, or left out with the reason. Examined: 13. Left out: 22. That the documents do not speak of a property is not a reason the analyst may give.

Entry	Property of the catalogue	Examined as	Why not
K01	Key/credential confidentiality	X01	
K02	Payload and end-to-end confidentiality	X01	
K03	Strong secrecy/indistinguishability of secret values	X02	
K04	Key purpose, direction and session separation	Not applicable	The flow establishes no keys of its own and has no key hierarchy or directions; transport keys belong to the TLS baseline.
D01	Message integrity and data origin authenticity	A03, A08	
A01	Peer aliveness	A03	
A02	Weak agreement authentication	A03	

Entry	Property of the catalogue	Examined as	Why not
A05	Implicit key authentication	Not applicable	No key is established in this flow, so there is no implicit key authentication.
A06	Explicit key confirmation	Not applicable	The flow confirms a configuration operation, not the possession of a key.
A07	Identity–credential–key binding	Z01, A08.8	
R01	Authentication/challenge freshness	A04	
R02	Message-level replay resistance	A04, R03	
R04	Message acceptance order/window security	Not applicable	The API defines no sequence numbers, counters or acceptance windows; a PUT and a DELETE are ordered only by their effect on the store, which S05 covers.
R05	Downgrade resistance/security policy preservation	Not applicable	The feature negotiation of clause 5.1.8 (suppFeat) selects optional features, not a security level or version, so no downgrade claim applies.
C01	Forward secrecy (FS)	Not applicable	The flow establishes no session key of its own.
C02	Key-compromise impersonation resistance (KCI)	Not applicable	No long-term key is defined inside this flow, so key-compromise impersonation does not apply.
C03	Known-key security/compromise isolation	Not applicable	No key hierarchy or multi-session key derivation exists inside this flow.
C04	Resilience to ephemeral secret/partial state disclosure	Not applicable	The flow keeps no ephemeral secret; the only transient value is the transRefId, whose disclosure is the attacker nf_with_transrefid.
C05	Post-compromise security (PCS)	Not applicable	No ratchet or key renewal inside this flow.
C06	Enhanced authentication after registration credential compromise	Not applicable	No PAKE or registration record is used.
C07	Offline dictionary attack resistance	Not applicable	No password or low-entropy secret is used in this flow.
C08	Goal preservation under failure of the specified primitive	Not applicable	The source makes no claim of hybrid or post-quantum composition; algorithm strength is outside the symbolic model.
Z02	No expansion of delegated authority	Outside the scope	No delegation chain is modelled: delegated discovery through an SCP, which is the only delegation of authorization in this service, is left out of the scope.
Z04	Revocation and expiry take effect	S05	
Z05	Cross-tenant/cross-service security isolation	Z03, Z03.9	
S01	Cryptographic nonce uniqueness	Not applicable	The model constructs no AEAD nonce; the confidentiality of the transport is assumed from the TLS baseline.
S02	Retransmission/key reinstallation security	R03	
S03	Rollback resistance and state monotonicity	S05	
S04	Cross-phase security context consistency	Not applicable	The configuration lifecycle is the only phase; there is no registration, link establishment or recovery stage that could be mixed up.
P01	Identity anonymity	Not applicable	The service identifies NF instances by their instance identifiers throughout; no anonymity is claimed or possible.
P02	Session unlinkability	Not applicable	Configurations are bound to identified NF instances, so sessions are linkable by design.

Entry	Property of the catalogue	Examined as	Why not
P03	User/credential existence hiding	Not applicable	The source defines no lookup by a user or credential whose existence could be enumerated; it defines no application errors at all (s0223).
V01	Conditional progress/error recovery	The prover cannot be asked	Conditional progress and recovery are liveness properties; ProVerif answers reachability and equivalence, not that a run can always complete.
V02	No unintended deadlock	The prover cannot be asked	That a run never gets stuck cannot be asked of ProVerif.
V03	Limited resource abuse/amplification resistance	The prover cannot be asked	Resource consumption and amplification are beyond a count of events, which is all a symbolic model can say.

The parties and the protected values, one by one

The host listed the 14 parties and protected values that the drafted properties name, and the analyst said of each which attacker controls the party or holds the value, or why no such attacker was set. With an attacker: 5. Without: 9. The list follows from the properties; what was made of each entry is the analyst's judgement.

ID	Party or value	Kind	Attacker	Why none
S01	NF service consumer	party	`nf_with_transrefid`	
S02	MFAF	party	Everything lost by definition	The MFAF holds the whole store, the transRefIds and the configured content, and it is the party that makes every check in this flow; with it every property asked here is lost by definition and the documents ask none of them under a compromised MFAF.
S03	NRF	party	Everything lost by definition	A compromised NRF can issue a token for any subject and any audience, so every authorization property of this flow is lost by definition and the documents ask none of them under a compromised authorization server.
S04	Malicious legitimate NF	party	`nf_with_transrefid`	
S05	Configuration record	value	`nf_with_transrefid`	
S06	transRefId transaction reference	value	`nf_with_transrefid`	
S07	transRefId	value	`nf_with_transrefid`	
S08	notificationURI	value	Nothing gained	Knowledge of where notifications are to be delivered gives no capability in this flow: acting on the configuration requires the transRefId and a token, and the confidentiality of the value is asked as its own goal.
S09	correId	value	Nothing gained	The correlation identifier is not used to authorize anything; acting on the configuration needs the transRefId and a token.
S10	mfafNotiInfo	value	Nothing gained	The MFAF notification information identifies notifications of that configuration; it authorizes no operation.
S11	OAuth2 access token	value	Nothing gained	In this model a token alone does not let the attacker act: the request must also come from the NF's TLS session, whose key is derived from the NF identity and is not available to another party. A stolen token with the NF's platform is the attacker nf_with_transrefid.
S12	nmfaf-3dadatamanagement scope	value	Nothing gained	The scope name is public: the source states it in the clear (s0232).

ID	Party or value	Kind	Attacker	Why none
S13	Access token	value	Nothing gained	Same as S11: a bearer token without the consumer's transport credentials cannot be used in the modelled flow.
S14	Creator identity	value	Nothing gained	NF instance identifiers are public identities in the service based architecture; the creator identity is what A03 and Z03 speak about, not a secret.

The properties under each attacker

The attackers are the analyst's, set before a query was written; each answer was recorded under one of them. A cell is the prover's answer for the property under that attacker; "No answer" is an attacker set for the property under which the prover left a claim undecided or nothing was recorded; an empty cell is an attacker the analyst did not set for the property.

Attacker	Has	Basis	Where
`network`	the network only: it reads, drops, changes and sends messages, and holds no secret of a party		
`nf_with_transrefid`	A valid OAuth2 access token with the nmfaf-3dadatamanagement service scope for its own NF instance and a TLS session to the MFAF under its own identity, and the transRefId of another consumer's Individual MFAF Configuration, obtained after that configuration was created (from an intermediary that handled the resource URI, from operational logs, or as a resource identifier the source keeps no secret).	The task distinguishes the external network attacker from a malicious legitimate NF holding its own credentials. The source defines one service-wide scope and no resource or operation level scopes (s0232), requires no check that the requester of a PUT or DELETE created the configuration (s0060, s0076), and assigns no confidentiality to the transRefId, which is the resource URI variable (s0117, s0144) carried in the Location header (s0133); RFC 6749 says OAuth2 gives the resource server no method to ensure a token was issued to the presenting client (rfc6749:2879).	rfc6749:2879, s0117, s0133, s0144, s0232, s0060, s0076

ID	`network`	`nf_with_transrefid`
A03	Holds	Holds
A04	No answer	
A08	Holds	Fails
Z01	Holds	Holds
Z03	Holds	Fails
S05	Holds	Holds
R03	No answer	
A08.8	Holds	Fails
Z03.9	Holds	Fails
A03.10	Holds	Holds
X01	Holds	Holds
X02	Holds	
X03	Holds	Holds

Source and targets

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. The list of targets was approved as drafted, beforehand, with --approve-targets.

Decisions the host took by its documented default (task `decisions: auto`), for you to check:

- approve_package: approved the source package protocol-40801477fa231ee0bb0a793ce80c

This step as it reports itself: 1 requests (1 failed), 0.000 USD, 64865 tokens in and 4012 out, 188 s.

Notes on the prover and the model

- S05: answered in ProVerif, and the analyst names changing_state for this target
- X03: answered in ProVerif, and the analyst names changing_state for this target

A ProVerif model that simplifies algebra (xor, dh_algebra, other_algebra) gives the attacker less than the real operation: a proof there is a proof against a weaker attacker, an attack there is an attack. A ProVerif model that leaves out a check on a value that changes (changing_state) lets parties accept more: a proof there holds, an attack there may be nothing but the missing check. Tamarin+ Agent models these as they are.

A bound in steps is an answer on bounded.pv: the analyst's second ProVerif model, in which the party that keeps a value that changes is unrolled to that many steps with its check in place. That model does less than the protocol does, so the property holds for those steps and no further, and an attack found there is an attack. The number of steps is the analyst's statement.

State review by the analyst

This part is the analyst's own reading of the source, not an answer of the prover. "yes" under shown by ProVerif means that ProVerif found a trace with these steps in the unrolled model, so the steps are possible; that they break a target is said only by an attack in the table of answers, and what else they cost is the analyst's reading. "nothing found" says which patterns were looked at and proves nothing.

value	kept by	changes when	checked when	refs
The Individual MFAF Configuration record of one transRefId: its content (notificationURI, correId), the identity of the NF service consumer that created it and the flag that it has been removed by Deconfigure.	The MFAF (s0052, s0062: it stores the configuration and stores it again after an update; s0076: it removes the individual resource).	A POST creates it and assigns the transRefId (s0048, s0049); a PUT replaces its content with the content of the update request (s0061, s0062); a DELETE flags it removed (s0076).	A PUT and a DELETE must name an existing transRefId (s0144, s0059, s0075); after Deconfigure the resource no longer exists (s0076). The clauses list no check of the creator identity and no check of the token subject against the resource (s0060, s0061, s0076).	s0052, s0062, s0076, s0048, s0049, s0061, s0059, s0075, s0144, s0223
The transRefId that the consumer keeps after the 201 Created response (Location header) and quotes in every later PUT and DELETE.	The NF service consumer that created the configuration (s0053, s0133), and the MFAF in its store (s0049, s0144).	It is assigned once by the MFAF at creation and never changes; a new create assigns a new one (s0049).	The MFAF matches it against its store on every PUT and DELETE (s0059, s0075); the consumer compares the transRefId it received with the one it quotes.	s0049, s0053, s0133, s0059, s0075, s0144

pattern	result	target	shown by ProVerif	steps
replay	nothing found			A recorded request cannot be replayed into a new connection: the connection is authenticated and encrypted under a key derived from the NF identity, and every request carries a fresh per-run value that the consumer and the MFAF compare. ProVerif cannot confirm the negative: a message on a channel stays available to every process that knows the channel, so an injectivity query on the create, update or delete acceptance stays inconclusive (targets A04, R03).
repeat_one_time_action	nothing found			The removal is a flag that only ever gets set: once Deconfigure has been accepted, a second DELETE for that transRefId finds the flag and is refused, and the last unrolled state of the MFAF refuses a stale PUT (bounded.pv, query s05 on the never reached event UpdateAfterRemoval).
desynchronisation	candidate		no: not tried with ProVerif	nf_a creates a configuration and receives its transRefId (s0048, s0049, s0053). / nf_a sends the PUT with its notificationURI and correId; the MFAF stores that content for the transRefId (s0061, s0062). / The transRefId becomes known to another legitimate NF: it is the resource URI variable (s0117, s0144) and it travelled in the Location header (s0133). / The other NF sends a PUT for that transRefId with its own token; the MFAF checks only the token subject against the TLS identity, finds the configuration and stores the other NF's content (s0060, s0061, s0076). / nf_a's later DELETE removes the resource that now carries the other NF's content (s0076); nf_a never learns that its configuration was replaced.
update_before_check	nothing found			In the unrolled MFAF the token check, the existence of the transRefId and the removed flag are all checked before the content is changed and before any confirmation is produced (bounded.pv, macros M2 and M4); the same order is in model.pv, where the updated event follows the store lookup.
no_rollback_on_failure	nothing found			A request that is refused leaves the record as it was: only an accepted DELETE sets the removed flag and only an accepted PUT produces the updated event and confirmation, so an abandoned or refused run changes nothing.
race_between_sessions	nothing found			Two connections may carry a PUT for the same transRefId; both are accepted and the later one sets the stored content, which is ordinary HTTP replace semantics. The model cannot order two writes to a table, so this was looked at on the source only.
distinguishable_failure	nothing found			The source defines no application errors for this service (s0223) and no failure answer at all, so nothing tells an attacker why a request was refused; the model therefore has no failure answers to compare. This is a finding outside the targets: the source does not say what the MFAF answers to a PUT or DELETE for a transRefId it does not know.
wrap_or_window	nothing found			The API has no counter, sequence number or acceptance window; the only linear value is the transRefId, which is assigned once and never reused because the MFAF picks it fresh.
other	nothing found			Looked at whether the MFAF keeps any record of the creating consumer: no attribute of MessageConfiguration carries the creator (s0193) and no read clause requires the check, which is what Z03 and A08.8 ask about and where the attack was found.

What the analyst named for targets:

step	targets	features	reason
24	S05, X03	changing_state	S05 and X03 rest on the lifecycle state of one configuration that the MFAF keeps and changes: created, updated, flagged removed by Deconfigure. model.pv records it in a table that only grows (so "removed" is a flag, not a deletion) and cannot express an ordering between the removal and a later update; bounded.pv unrolls the MFAF through created, updated, removed and one further state, with the consumer issuing POST, PUT, DELETE and a stale PUT.
24	A03, A04, A08, Z01, Z03, Z03.9, A08.8, A03.10, R03, X01, X02	plain_primitives, many_sessions	The only cryptography is a symmetric key derived from the NF identity for the endpoint-to-endpoint connection (the TLS layer, modelled as an authenticated encrypted channel), a token table standing for the NRF-issued access token, and fresh names for the transRefId and the MFAF notification information; no algebraic laws are used anywhere. The agreement, authorization and disclosure properties are asked for any number of sessions and connections.

Falsified and not submitted

The prover falsified these properties and they are in no recorded answer.

- `a04_c` (model.pv, on an earlier version of the model, not run since): 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, on an earlier version of the model, not run since): 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, on an earlier version of the model, not run since): s: bitstring, r: bitstring, tr: bitstring; inj-event(ConfigRemoved(s,r,tr)) ==> inj-event(DeleteSent(s,r,tr))
- `r03_c` (model.pv, on an earlier version of the model, not run since): 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))

The analyst's closing note

The MFAF configuration lifecycle (POST / PUT / DELETE of one Individual MFAF Configuration) was modelled in ProVerif with TLS as an authenticated encrypted per-endpoint channel, the NRF-issued service-scope token and the configuration store, plus bounded.pv unrolling the lifecycle to four steps and a state review.

Established: the lifecycle correspondence holds (A03), every committed operation carries a token the NRF issued for that identity (Z01), acknowledgements follow the state change (A03.10), an update or removal is accepted only for an existing transRefId (X03), removal is final (S05, bounded to 4 steps with a control), the configuration content and token stay inside the consumer's session (X01, X02), creation-side integrity and the mfafNotiInfo binding hold (A08).

Finding (medium value, novelty unclear): the MFAF never checks that the requester of a PUT or DELETE created the configuration. Any NF with a service-scope token that knows the transRefId replaces another consumer's notificationURI and correId (attacks for Z03, Z03.9, A08, A08.8, each with a proved network-only counterpart and its falsified control) and can then remove the configuration, without the creator noticing. Recommendation: store the creator NF instance id with the configuration and reject a PUT or DELETE whose token subject differs (not machine-checked: no check_fix tool here).

No answer: A04 and R03 (a repeated request must not produce a second side effect) are inconclusive on the final model - ProVerif's channels are not consumptive, so the same request can be accepted by two handler copies; Tamarin+ Agent with a linear request store would settle them.

Outside the model: notification delivery (the second half of S05), mfafTransfer, redirects, error responses, token expiry and revocation, and the SCP path. The desynchronisation candidate in the state review is recorded as not confirmed by a query of bounded.pv.