Security analysis of AF inference local trained model branch

19.2.0 · ProVerif+ Agent report

Documents	5G System; Application Function Artificial Intelligence/Machine Learning (AI/ML) Services; Stage 3 (protocol definition)
Properties examined	11: 10 hold, 0 fail, 1 without an answer
Findings	None

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 Naf_Inference service is the service-based interface of an Application Function acting as a VFL server. An NF service consumer, here the NWDAF, subscribes to inference event notifications, may modify the subscription and later unsubscribes, and the AF reports inference events back to it. In this configuration the AF has a trained model of its own and selects no NWDAF VFL client, so the results it reports come only from its local trained model, associated with the VFL correlation ID it determines.

The consumer holds its OAuth2 access token for the single service scope naf-inference, the VFL correlation IDs and the analytics subscriptions it created; the AF holds the local trained model and the Individual Inference Subscription resources with their notifUri and notifCorreId. When the exchange ends the consumer is sure that the subscription it holds is the one it created and that the results it received were computed for that subscription, and the AF is sure that every creation, modification or cancellation it performed came from the NF that owns the subscription and holds a valid token.

The parties:

- **NF service consumer (NWDAF)**: requests inference notifications from the AF. Holds: OAuth2 access token for scope naf-inference, notifCorreId, notifUri, inferAnaSubs.
- **AF (VFL server, trusted)**: produces inference results without any NWDAF VFL client. Holds: local trained ML model, VFL correlation IDs, subscriptions.

Figure 1 shows the exchange as the documents specify it.

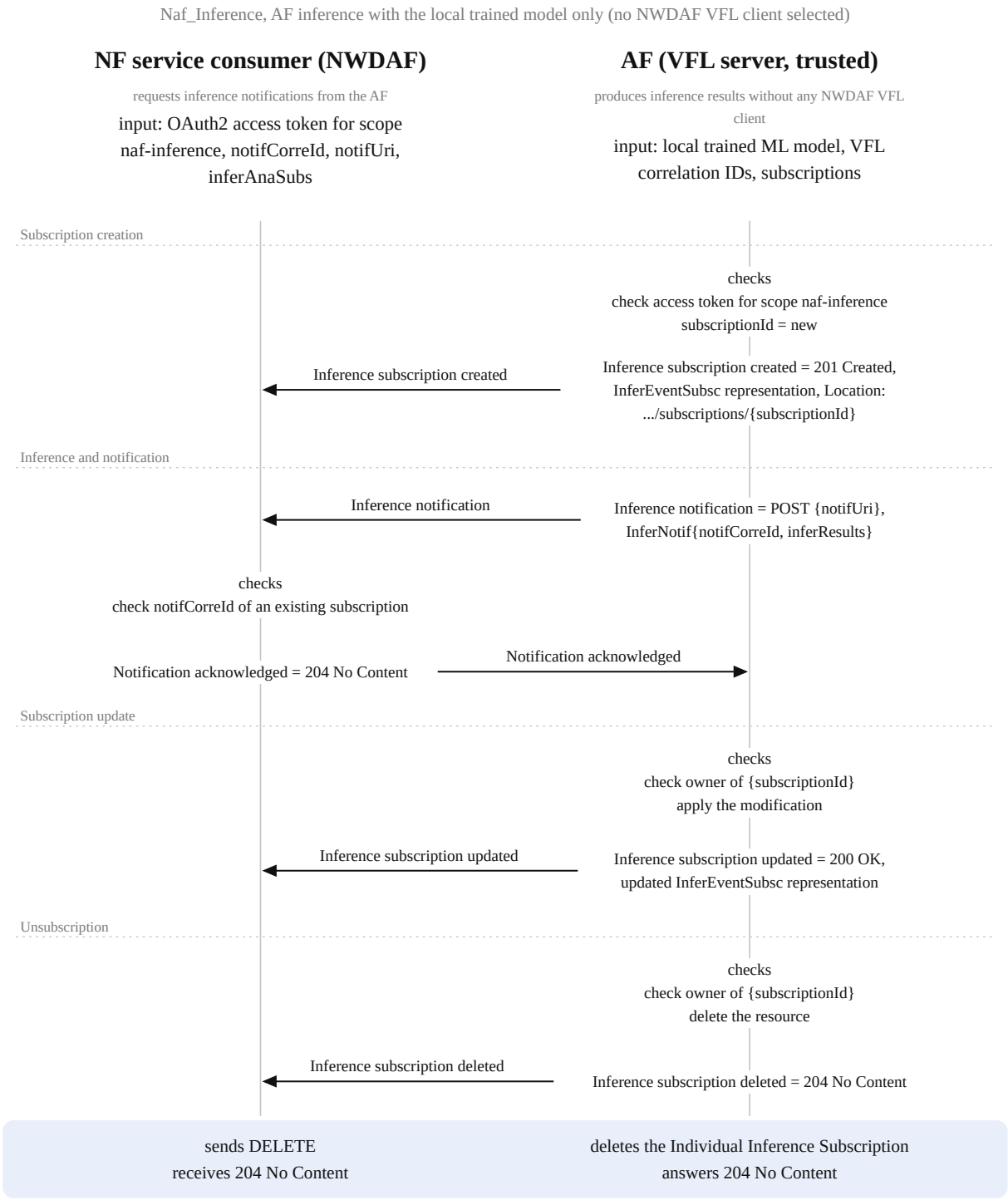


Figure 1. The protocol as specified.

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

No finding. The analysis found nothing that the documents do not already state or set aside. The table in section 3 has the result for every property.

No finding: within the trusted-AF branch of Naf_Inference with the local trained model, every target is answered and no attack on the protocol was found. The AF binds each subscription to the consumer that created it and to that consumer's access token, so an NF holding its own token cannot create, modify or cancel another NF's subscription, and the control shows that without the ownership check it could. In the unrolled lifecycle a deleted subscription is neither notified nor modified again, with a control for each. Results of the local model are computed for the subscription's correlation ID and delivered sealed under the AF/consumer session, and the consumer accepts only the notification of its own correlation id and callback endpoint. Two answers are weaker than a proof: the confidentiality of the results could not be settled at the trace level, and the freshness of results after an update rests on the AF because InferNotif carries no version. The analysis reproduces the limits the documents state, one service scope for the whole API with no resource-level scope and the bearer-token property that a leaked token makes its holder that NF.

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	**Responses bound to the consumer's own request.** Every timeNFThe consumer accepts a successful response to the creation/update/cancellation of a subscription, and there is a matching request for a subscription initiated by the consumer itself, andsubscriptionId、notifUri、notifCorreIdConsistent with the value in the request; Every timeAFA single notification exists that was previously created or updated by the consumer, andnotifUriandnotifCorreIdMatches the subscription.	no compromise	Holds	medium	no
A08	**Notification bound to the local-model inference.** Every timeAFGenerate and return/notify a local model reasoning, which is carriedVFLcorrelationID_Other OrganiserIDConsistent, and notification to the subscribernotifUriand carry the subscribernotifCorreId。	no compromise	Holds	medium	no
Z01	**Valid token required for each operation.** Whenever a protected operation (subscription to create, update, delete, or notification is accepted) is submitted, the submitting party holds a valid OAuth2 token issued by NRF; Requests without valid tokens do not have protected side effects.	no compromise	Holds	medium	yes
Z03	**Token holder is the subscription creator.** Whenever a token is used to update or delete a subscription of subscriptionId, the holder of the token is the creator of the subscription (or an authorized entity) and the audience of the token is AF; The token cannot be used for other subscriptions or other operations.	no compromise	Holds	medium	yes
Z04	**No notifications after unsubscribing.** Whenever a subscription is successfully deleted (orAFHere.termCause) and thereafterAFNo longer send a new notification to the Subscription, and for thatsubscriptionIdThe update is no longer effective.	no compromise	Holds within a bound (the changing value unrolled to 4 steps)	medium	no

ID	Property	Compromise assumed	Result	Value	In the documents
S05	**Old identifiers rejected after unsubscription.** Following effective subscription to delete, use oldsubscriptionIdOldnotifUriOr old.notifCorreIdrequest cannot trigger a new notification or modify a subscription.	no compromise	Holds within a bound (the changing value unrolled to 4 steps)	low	no
K02	**Target identities and results kept from otherNFs.** Target identity in subscription requests and notifications (supis, gpsis,intGroupIds、exterGroupIds) and the result of the reasoning is only for the consumers who initiated the subscriptions and for the consumers who signed the subscriptions.AFVisible, OtherNFWith external assailants, they cannot be obtained.	no compromise	Reachable only	low	unclear
K01	**Token andTLSkeys kept from the attacker.** The attackers cannot get itNFConsumer'sOAuth2Access to tokens and no access to consumers andAFBetweenTLSThe key to the session.	no compromise	Holds	low	yes
A08.9	**Result carries matching subscription and correlation ids.** Every timeAFReturn or notify a local model reasoning result, which is carriedsubscriptionIdandvflCorrIdis consistent with the value established when the subscription is created/updated.	no compromise	Holds	medium	no
Z03.10	**No cross-subscription update or cancel.** A NF with its own valid OAuth2 token cannot update or delete another NF-created subscription.	no compromise	Holds	medium	yes
U01	**No stale results after an update.** After the update of the subscription became effective,AFReturns or notifies the reasoning that is not based on the subscription configuration before it is updated; The old results produced before the update cannot be associated with the updated configuration.	no compromise	Holds	medium	unclear

The questions asked, and the analyst's answers:

Question	Answer	Rests on
Q1: Local result and subscriptionID、VFLcorrelationIDTie	The binding holds in the model: a notified result carries the correlation data of the subscription it was computed for, and the consumer's acceptance is bound to its own notifCorreId and notifUri; the one gap is that the subscriptionId itself is not carried in the notification, so the consumer correlates by notifCorreId alone.	A08, A08.9, A03
Q2: LegalNFCross-subscribe to update/release permission boundaries	No: with its own valid token an NF can only act as itself at the AF, and every update or cancellation is checked against the owner stored with the subscription, so another NF's subscription is out of reach; what remains is the bearer-token limit, a leaked token makes its holder that NF, which the documents already say.	Z01, Z03, Z03.10
Q3: Relevance of the previous results to the updated configuration	Only partly: results notified after an update are bound to the subscription by its correlation data, which the update does not change, so the consumer cannot tell a result produced before the update from one produced after it; the source defines no version or sequence number in InferNotif, and the model's AF always recomputes, so this rests on the AF behaving correctly rather than on a check the consumer makes.	U01, A03, A08

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:

- The AF is taken as trusted, as this branch of the API presupposes; the untrusted-AF configuration through the NEF, indirect communication through an SCP and inter-PLMN SEPP are out of scope.
- TLS is modelled as an authenticated channel between the AF and one NF with a session key per endpoint; the model does not prove the computational security of TLS or OAuth2, and a broken primitive or a leaked session key is not modelled.
- The HTTP layer (status codes, error bodies, redirection, feature negotiation) is not modelled; a refusal is modelled as the absence of the protected effect, not as a particular status code.
- Subscription identifiers are fresh values generated by the AF; the source does not state how they are generated or whether they can be predicted.
- The adversary cases are scripted: the malicious NF sends a fixed set of hostile requests with its own credentials, and no process holds a leaked token, so those rows say what the checks do rather than what a token thief with a known identifier could do.
- The subscription lifecycle is unrolled to four steps in bounded.pv; a monotone table cannot model a deletion in model.pv, and the position of the notification relative to the update is a modelling choice because the source leaves it free.
- The inference results are modelled as a private function of the VFL correlation ID; the quality of the machine-learning model and the correctness of the analytics are beyond a symbolic prover.
- Availability, latency and the cost of a request are not modelled, and the individual identifiers of the data model (supis, gpsis, intGroupIds, exterGroupIds) are represented only by the protected request body.

What the host records:

- No answer for: K02.
- The answers of Z04, S05 rest on an operation the model simplifies (Appendix D).
- Set by the analyst and without a proof or an attack (Appendix D): K02 under `network`; K02 under `rogue_nf`; K02 under `stolen_token`.
- Proved without a control query (Appendix C): A03: a03_resp, A03: a03_notif, A03: a03_acc, A08: a08, A08: a03_notif, Z01: a03_resp, Z01: z01_upd, Z01: z01_del, Z03: z03_upd, Z03: z03_del, K01: k01_tok, K01: k01_key, A08.9: a089, Z03.10: z03_upd, U01: a089. 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: Responses bound to the consumer's own request

Every timeNFThe consumer accepts a successful response to the creation/update/cancellation of a subscription, and there is a matching request for a subscription initiated by the consumer itself, andsubscriptionId、 notifUri、 notifCorreIdConsistent with the value in the request; Every timeAFA single notification exists that was previously created or updated by the consumer, andnotifUriandnotifCorreIdMatches the subscription.

The AF creates a subscription only for a request the consumer itself sent, and every notification it sends is bound to that subscription's notifUri and notifCorreId. The consumer accepts a notification only if the AF sent it for its own notifUri and carries its own correlation id, so results cannot be moved from one subscription to another. No attack was found against either party.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The binding is what makes the notification usable for analytics without mixing subscriptions; the proof holds with the transport and ordering caveats above, and the documents state the procedure but not this property.
- Stated in the documents: no
- Rests on: TLS between the AF and each NF is an authenticated channel; the reply and callback channels are created by the consumer and travel inside it.; The notification is modelled as a sealed record under the AF/consumer session key, checked with the consumer's notifCorreId.; The position of the notification relative to the update is a modelling choice; the source leaves it free.
- Open: Whether a notification can be replayed to the consumer, which TLS is assumed to prevent and the model therefore does not show.; Which party's check fails first when a notification is replayed inside one session; the prover gave no control for this property.
- Next step: Model the notification on a channel the adversary writes, with the AF's session authentication as the only sender check, to see whether a replayed InferNotif is accepted twice.

A08: Notification bound to the local-model inference

Every timeAFGenerate and return/notify a local model reasoning, which is carriedVFLcorrelationID_Other OrganiserIDConsistent, and notification to the subscribernotifUriand carry the subscribernotifCorreId。

In this branch the AF computes the result from its local trained model for the correlation ID it determined for the subscription and notifies that result to the subscription's notifUri under its notifCorreId. The result and the subscription are therefore never mixed, and no party other than the consumer learns it from the exchange.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: It is the central guarantee of the no-VFL-client branch, and it holds with the caveats of an internal correlation ID and a sealed transport.
- Stated in the documents: no
- Rests on: The result is modelled as a private function of the correlation ID; the model says nothing about the quality of the analytics.; The channel between the AF and the consumer carries the notification sealed under the

session key.

- Open: The VFL correlation ID is internal to the AF and is not carried in InferNotif, so its correspondence is established by construction rather than by a field the consumer checks.
- Next step: Check whether a later release carries the correlation ID in the result, which would let the consumer verify the association itself.

Z01: Valid token required for each operation

Whenever a protected operation (subscription to create, update, delete, or notification is accepted) is submitted, the submitting party holds a valid OAuth2 token issued by NRF for that operation; Requests without valid tokens do not have protected side effects.

Every protected operation the AF performed in the model was requested by the identity that holds the access token for scope naf-inference: a request under another NF's name with its own token creates, modifies and deletes nothing. The single service scope covers the whole API, so the scope itself distinguishes no operation or resource; the identity check does.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: 6.4.9 states that access is authorized with an OAuth2 token, so the requirement is known; the analysis shows the identity check is what stops an NF from acting under another name, with the audience and expiry parts open.
- Stated in the documents: yes. Where the documents say it: s0272, s0273, s0275
- Rests on: The token is modelled by a value private to each NF, checked for equality with the token of the identity named in the request.; The AF is trusted to perform the check; a dishonest AF voids it by definition.
- Open: The audience and expiry claims of the token are not modelled; the model checks that the caller holds the token of the identity it names.; No control was recorded for the creation claim, so nothing shows that this property as written can fail in the model.
- Next step: Model the token as an assertion signed by the NRF with subject, audience and expiry, and ask the same queries against a token of another audience.

Z03: Token holder is the subscription creator

Whenever a token is used to update or delete a subscription of subscriptionId, the holder of the token is the founder of the subscription (or an authorized entity) and its audience is AF; The token cannot be used for other subscriptions or other operations.

An update or a cancellation is applied only when the requester is the NF that created the subscription, so an NF with its own valid token cannot touch another NF's subscription. The protection rests on the AF's ownership check, not on the scope, and a holder of another NF's token is that NF at the AF, which the documents state as the bearer-token limit.

- Result: Holds; answers what is asked: partly; usable with caveats
- Value medium: It isolates that the single service scope protects nothing by itself and that the ownership check carries the property; the stolen-token case is stated in the documents and only partly modelled.
- Stated in the documents: yes. Where the documents say it: s0275
- Rests on: Subscription identifiers are fresh values the AF generates; the source does not say whether they can be predicted.; The leaked-token attacker is not represented by a process, so its row is a statement about the check and not an attack the prover ran.

- Open: The audience of the token and the claim that a token cannot be used for another operation are not modelled.; Under a stolen token the model has no process holding it, so that row states what the check does rather than an attack the prover ran.
- Next step: Add a process that holds a leaked token and a published subscriptionId, and ask whether it can redirect the notifUri of that subscription; that is what settles the bearer-token exposure.

Z04: No notifications after unsubscribing

Every time a subscription is successfully deleted (orAFHere.termCause) and thereafterAFNo longer send a new notification to the Subscription, and for thatsubscriptionIdThe update is no longer effective.

After a successful deletion the AF refuses to notify for that subscription: the chain keeps the new state and the notification attempt is refused. The proof depends on that state being replaced, which the control shows: without the replacement the notification is sent.

- Result: Holds within a bound (the changing value unrolled to 4 steps); answers what is asked: yes; usable with caveats
- Value medium: It is the guarantee that unsubscribing stops the reporting, shown mechanically with a control, on a model that stops at four steps.
- Stated in the documents: no
- Rests on: The lifecycle is unrolled to four steps: create, delete, notification attempt, update attempt.; The state is the position of the chain because a monotone table cannot model a deletion.
- Open: Deletions after more than four steps, and two subscriptions of the same consumer, are outside the unrolled model.
- Next step: Extend the unrolled chain and add a second subscription of the same consumer to see whether a deletion of one leaves a notification path for the other.

S05: Old identifiers rejected after unsubscription

Subscription after effective, use oldsubscriptionIdOldnotifUriOr old.notifCorreIdrequest cannot trigger a new notification or modify a subscription.

A request that still names the identifiers of a deleted subscription no longer modifies it and produces no notification: the update attempt after the deletion is refused, and the control without the state replacement accepts it. What the consumer keeps of the old correlation id is outside the model.

- Result: Holds within a bound (the changing value unrolled to 4 steps); answers what is asked: yes; usable with caveats
- Value low: It follows from the deletion state once the state is replaced; the value lies in the control that shows the state replacement is what carries it.
- Stated in the documents: no
- Rests on: The unrolled chain compares the state before the update and before the notification.; A monotone table in the main model cannot show the removal, so this answer comes only from the unrolled model.
- Open: A request that names a deleted subscription while a new subscription of the same consumer exists is not modelled.
- Next step: Unroll two subscriptions of one consumer and ask the same questions across the deletion of one of them.

K02: Target identities and results kept from other NFs

Subscription requests and targeted identities in notifications (supis, gpsis,intGroupIds、exterGroupIds) and the result of the reasoning is only for the consumers who initiated the subscriptions and for the consumers who signed the subscriptions.AFVisible, OtherNFWith external assailants, they cannot be obtained.

The situation the target speaks about is reachable: the AF computes a local-model result and the subscribing consumer receives and accepts it. The confidentiality of that result could not be settled: ProVerif verified that a notified result is known to the attacker only if that consumer released it, but it could not build a trace for the premise, so the recorded answer is reachability and not a proof of secrecy. Results are sealed under the AF/consumer session key, so no other NF reads them as long as that key stays with the endpoints.

- Result: Reachable only; answers what is asked: partly; not usable
- Value low: The prover did not answer the confidentiality question at the trace level, and the reachability that is recorded says only that the exchange happens.
- Stated in the documents: unclear. Source text it rests on: s0237, s0246
- Rests on: The result travels sealed under the AF/consumer session key; the consumer that releases a result is a separate identity in the model.; The confidentiality query was inconclusive on its premise, so this target has no proof-level answer.
- Open: The confidentiality query was verified, but the prover found no trace for its premise, so the answer rests on an over-approximation.; The target identities (supis, gpsis, intGroupIds, exterGroupIds) are carried as the protected request body and are not individual values in the model.
- Next step: Rebuild the notification and the consumer with explicit per-session state in a tool that models sessions concretely, and ask the secrecy question there; that is the case for Tamarin with an agent.

K01: Token and TLS keys kept from the attacker

The attackers were unable to obtain a NF consumer OAuth2 access card or a key to a TLS conversation between the consumer and AF.

Neither the access token nor the TLS session key of a consumer becomes known to the external attacker or to another NF: they are private values in the model and no message carries them. This is the assumption on which the other results rest rather than a result about the protocol.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: It restates the credential and transport assumption that the whole analysis needs, which the documents already require of the transport.
- Stated in the documents: yes. Where the documents say it: s0272, s0273
- Rests on: Tokens and session keys are private names per NF; the model does not prove the computational strength of the primitives.
- Open: Storage of the token and of the session key inside the NF is not modelled.
- Next step: None for this model; the answer is the transport and credential assumption on which the other results rest.

A08.9: Result carries matching subscription and correlation ids

Every timeAFReturn or notify a local model reasoning result, which is carriedsubscriptionIdandvflCorrIdis consistent with the value established when the subscription is created/updated.

A result the consumer accepts carries the notifCorreId and notifUri of a subscription of that consumer, so the result is attached to the right resource. The subscriptionId itself is not in InferNotif, so the correlation is by notifCorreId and by the callback endpoint.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: It is the binding the consumer relies on, and it holds with the caveat that the identifier the model binds by is the correlation id, not the subscription URI.
- Stated in the documents: no
- Rests on: The notification carries only notifCorreId and the results, as the data model defines; the subscriptionId is not in it.
- Open: Immediate reporting, where the results come in the POST response instead of a notification, is not modelled.
- Next step: Model reportInfo with immediate reporting and ask the same binding question about the results in the 201 response.

Z03.10: No cross-subscription update or cancel

One holder of its own valid OAuth2 token NF cannot update or delete another Subscription created by NF.

An NF holding only its own token did not modify or cancel another NF's subscription in the model; every update and cancellation is checked against the owner stored with the subscription, and the control shows that without that check an NF can cancel a subscription another NF created. The answer rests on identifiers the attacker cannot name: the model gives each subscription a fresh identifier, so what is established is the ownership check, not the secrecy of the identifier.

- Result: Holds; answers what is asked: partly; usable with caveats
- Value medium: It shows what the single service scope leaves to the AF's own authorization, with a control, and rests on a fresh-identifier assumption the documents do not make.
- Stated in the documents: yes. Where the documents say it: s0275, s0055, s0074
- Rests on: Subscription identifiers are fresh, unguessable values; the source does not specify their generation.; The malicious NF is a scripted process, so only the hostile requests written into the model were tried.
- Open: Cross-subscription attacks need the victim's subscriptionId, which is a fresh value in the model; the source does not say how it is generated.; The stolen-token case is not represented by a process.
- Next step: Give a thief process the leaked token together with a published subscription URI and ask the same ownership queries; that is the case in which the single scope becomes visible.

U01: No stale results after an update

After the update of the subscription became effective,AFReturns or notifies the reasoning that is not based on the subscription configuration before it is updated; The old results produced before the update cannot be associated with the updated configuration.

A result the consumer accepts after its update is still bound to the subscription by its notifCorreId and notifUri, which the update does not change. Whether the result was computed before or after the update cannot be told by the consumer: the data model carries no version, so this part rests on the AF recomputing and on nothing the consumer can check.

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

- Value medium: The binding part holds, and the freshness part identifies a gap in the data model that the documents do not state; the prover could not be asked the freshness question without a version field.
- Stated in the documents: unclear. Source text it rests on: s0237, s0246, s0061
- Rests on: The AF is modelled as recomputing after the update; the source does not require it and leaves in-flight results unspecified.; The notification carries only notifCorreId and the results.
- Open: InferNotif carries no version or sequence number, so the consumer has no check that tells a result produced before an update from one produced after it.; The model's AF always recomputes after the update, so a stale result delivered after an update is not part of the model.
- Next step: Add a version or reportInfo revision to InferNotif and ask the query on a model in which the AF may also deliver a result it computed before the update; that is what would settle the freshness part.

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_resp			6 proved
A08	proved		a08			4 proved
Z01	proved		a03_resp			6 proved
Z03	proved		z03_upd	6 of 8		8 proved
Z04	bounded	the changing value unrolled to 4 steps	z04_notif	checked		2 bounded
S05	bounded	the changing value unrolled to 4 steps	z04_upd	checked		2 bounded
K02	reachable		run			6 reachable
K01	proved		k01_tok			4 proved
A08.9	proved		a089			2 proved
Z03.10	proved		z03_upd	6 of 7		7 proved
U01	proved		a089			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	Whenever the AF creates an Individual Inference Subscription for consumer id with the subscription data (notifCorreId, notifUri), that consumer itself sent the creation request with those values, so the subscriptionId and the correlation fields the consumer later sees match its own request. Attacker beyond the network: none.	`network`	proved	a03_resp	none
A03	Whenever the AF sends a notification for a subscription, that subscription was created by that consumer with the same notifUri and the same notifCorreId that the notification carries. Attacker beyond the network: none.	`network`	proved	a03_notif	none
A03	The consumer accepts a notification only if the AF sent that notification for its notifUri with its own notifCorreId, so a notification is not attributed to the wrong subscription. Attacker beyond the network: none.	`network`	proved	a03_acc	none
A03	Whenever the AF creates a subscription for consumer id, that consumer itself sent the creation request with those values, even though another NF holds a valid token of its own. Attacker beyond the network: its own OAuth2 token for scope naf-inference and its own TLS identity towards the AF.	`rogue_nf`	proved	a03_resp	none
A03	Every notification the AF sends belongs to the subscription of that consumer with matching notifUri and notifCorreId, also when another NF with its own valid token consumes the API. Attacker beyond the network: its own OAuth2 token for scope naf-inference and its own TLS identity towards the AF.	`rogue_nf`	proved	a03_notif	none

target	claim	attacker	answer	query	control
A03	The consumer accepts only a notification the AF sent for its notifUri and its notifCorreId, so a malicious NF consumer cannot get a notification of its own making accepted by another consumer. Attacker beyond the network: its own OAuth2 token for scope naf-inference and its own TLS identity towards the AF.	`rogue_nf`	proved	a03_acc	none
A08	Every result the AF notifies is the result it computed from its local trained model for that subscription's determined VFL correlation ID, so the result and the subscription are not mixed. Attacker beyond the network: none.	`network`	proved	a08	none
A08	The notification carrying a local-model result goes to the notifUri of the subscription and carries that subscription's notifCorreId. Attacker beyond the network: none.	`network`	proved	a03_notif	none
A08	Every result the AF notifies is the one computed for that subscription's correlation ID, also when another NF with its own valid token consumes the API. Attacker beyond the network: its own OAuth2 token for scope naf-inference and its own TLS identity.	`rogue_nf`	proved	a08	none
A08	The result notification goes to the notifUri of the subscription with its notifCorreId, even with another NF consuming the API. Attacker beyond the network: its own OAuth2 token for scope naf-inference and its own TLS identity.	`rogue_nf`	proved	a03_notif	none
Z01	A subscription is created at the AF only for a request that the identity named in it sent, so a request without a valid access token produces no subscription. Attacker beyond the network: none.	`network`	proved	a03_resp	none
Z01	An update of an Individual Inference Subscription is carried out only for a request the subscribing identity itself sent with its token. Attacker beyond the network: none.	`network`	proved	z01_upd	none
Z01	A deletion of an Individual Inference Subscription is carried out only for a request the subscribing identity itself sent with its token. Attacker beyond the network: none.	`network`	proved	z01_del	none
Z01	A subscription is created at the AF only for a request from the identity it names, so an NF that presents its own token under another NF's identity creates no subscription. Attacker beyond the network: its own OAuth2 token for scope naf-inference and its own TLS identity.	`rogue_nf`	proved	a03_resp	none
Z01	An update is carried out only for a request from the subscribing identity, so an NF using its own token under another NF's identity changes nothing. Attacker beyond the network: its own OAuth2 token for scope naf-inference and its own TLS identity.	`rogue_nf`	proved	z01_upd	none
Z01	A deletion is carried out only for a request from the subscribing identity, so an NF using its own token under another NF's identity deletes nothing. Attacker beyond the network: its own OAuth2 token for scope naf-inference and its own TLS identity.	`rogue_nf`	proved	z01_del	none
Z03	An update is performed only on behalf of the identity the token was issued to, so a holder of another NF's stolen token is treated as that NF; the check that the update is by the subscription's creator still holds for the identity in the token. Attacker beyond the network: a leaked OAuth2 access token of a legitimate NF for scope naf-inference.	`stolen_token`	proved	z03_upd	none

target	claim	attacker	answer	query	control
Z03	A deletion is performed only on behalf of the identity the token was issued to; a stolen token makes its holder the same NF at the AF, which is the documented bearer-token limit, not a break of the ownership check. Attacker beyond the network: a leaked OAuth2 access token of a legitimate NF for scope naf-inference.	`stolen_token`	proved	z03_del	none
Z03	An update of a subscription is performed only on behalf of the consumer that created that subscription. Attacker beyond the network: none.	`network`	proved	z03_upd	checked
Z03	A deletion of a subscription is performed only on behalf of the consumer that created that subscription. Attacker beyond the network: none.	`network`	proved	z03_del	checked
Z03	An NF holding only its own valid token cannot get an update performed on another NF's subscription. Attacker beyond the network: its own OAuth2 token for scope naf-inference and its own TLS identity.	`rogue_nf`	proved	z03_upd	checked
Z03	An NF holding only its own valid token cannot get another NF's subscription deleted. Attacker beyond the network: its own OAuth2 token for scope naf-inference and its own TLS identity.	`rogue_nf`	proved	z03_del	checked
Z03	An update is performed only on behalf of the identity the token was issued to, so a holder of another NF's stolen token is treated as that NF; the ownership check itself still holds for the identity in the token. Attacker beyond the network: a leaked OAuth2 access token of a legitimate NF for scope naf-inference.	`stolen_token`	proved	z03_upd	checked
Z03	A deletion is performed only on behalf of the identity the token was issued to; a stolen token makes its holder that NF at the AF, which is the bearer-token limit the documents state. Attacker beyond the network: a leaked OAuth2 access token of a legitimate NF for scope naf-inference.	`stolen_token`	proved	z03_del	checked
Z04	After the successful deletion of its subscription (204 No Content) the AF sends no further notification for that subscription: on the unrolled lifecycle the notification attempt after the deletion is refused, and a notification is only sent when the state the AF keeps still says the subscription exists. Attacker beyond the network: none.	`network`	bounded (the changing value unrolled to 4 steps)	z04_notif	checked
Z04	After the successful deletion of a subscription the AF sends no further notification for it, also when another NF with its own valid credentials consumes the API; the deletion is the owner's and the state the AF keeps decides. Attacker beyond the network: its own OAuth2 token for scope naf-inference and its own TLS identity.	`rogue_nf`	bounded (the changing value unrolled to 4 steps)	z04_notif	checked
S05	After the successful deletion, a request naming the old subscriptionId (with its notifCorreId and notifUri) no longer modifies the subscription and creates no notification: on the unrolled lifecycle the update after the deletion is refused, while it is accepted when the state still says the subscription exists. Attacker beyond the network: none.	`network`	bounded (the changing value unrolled to 4 steps)	z04_upd	checked
S05	After the deletion of a subscription, requests naming its old identifiers no longer modify it, also when another NF with its own valid credentials is present. Attacker beyond the network: its own OAuth2 token for scope naf-inference and its own TLS identity.	`rogue_nf`	bounded (the changing value unrolled to 4 steps)	z04_upd	checked

target	claim	attacker	answer	query	control
K02	The situation the target speaks about is reachable: the AF computes a result from its local trained model for the subscription and the subscribing consumer receives and accepts it; the confidentiality of that result is a further claim, on which the prover gave no trace-level answer. Attacker beyond the network: none.	`network`	reachable	run	
K02	The situation the target speaks about is reachable with another NF consuming the API: a local-model result reaches the subscribing consumer and is accepted. Attacker beyond the network: its own OAuth2 token for scope naf-inference and its own TLS identity.	`rogue_nf`	reachable	run	
K02	The situation the target speaks about is reachable: a local-model result reaches the subscribing consumer and is accepted, while another party holds a leaked token. Attacker beyond the network: a leaked OAuth2 access token of a legitimate NF for scope naf-inference.	`stolen_token`	reachable	run	
K02	The situation the target speaks about is reachable: the AF computes a result from its local trained model for the subscription and the subscribing consumer receives and accepts it. The confidentiality of that result is a further claim on which the prover gave no trace for its premise. Attacker beyond the network: none.	`network`	reachable	reach_notif	
K02	The situation the target speaks about is reachable while another NF with its own credentials consumes the API: a local-model result reaches the subscribing consumer and is accepted. Attacker beyond the network: its own OAuth2 token for scope naf-inference and its own TLS identity.	`rogue_nf`	reachable	reach_notif	
K02	The situation the target speaks about is reachable while another party holds a leaked access token: a local-model result reaches the subscribing consumer and is accepted. Attacker beyond the network: a leaked OAuth2 access token of a legitimate NF for scope naf-inference.	`stolen_token`	reachable	reach_notif	
K01	The external attacker does not obtain the consumer's OAuth2 access token. Attacker beyond the network: none.	`network`	proved	k01_tok	none
K01	The external attacker does not obtain the key of the consumer's TLS session with the AF. Attacker beyond the network: none.	`network`	proved	k01_key	none
K01	Another NF holding its own valid credentials does not obtain the consumer's access token. Attacker beyond the network: its own OAuth2 token for scope naf-inference and its own TLS identity.	`rogue_nf`	proved	k01_tok	none
K01	Another NF holding its own valid credentials does not obtain the key of the consumer's TLS session with the AF. Attacker beyond the network: its own OAuth2 token for scope naf-inference and its own TLS identity.	`rogue_nf`	proved	k01_key	none
A08.9	A notification the consumer accepts carries the correlation id and the notifUri of the subscription that this consumer created, so the result is bound to its subscription. Attacker beyond the network: none.	`network`	proved	a089	none
A08.9	A notification the consumer accepts carries the correlation id and notifUri of the subscription that this consumer created, also when a malicious NF consumer is present. Attacker beyond the network: its own OAuth2 token for scope naf-inference and its own TLS identity.	`rogue_nf`	proved	a089	none

target	claim	attacker	answer	query	control
Z03.10	An update is performed only on behalf of the identity the token was issued to, so a leaked token makes its holder that NF at the AF; the ownership check itself still holds for the identity in the token. Attacker beyond the network: a leaked OAuth2 access token of a legitimate NF for scope naf-inference.	`stolen_token`	proved	z03_upd	none
Z03.10	An update of a subscription is performed only on behalf of the consumer that created it, so no other party can modify another NF's subscription. Attacker beyond the network: none.	`network`	proved	z03_upd	checked
Z03.10	A cancellation of a subscription is performed only on behalf of the consumer that created it, so no other party can cancel another NF's subscription. Attacker beyond the network: none.	`network`	proved	z03_del	checked
Z03.10	An NF holding only its own valid token cannot update another NF's subscription. Attacker beyond the network: its own OAuth2 token for scope naf-inference and its own TLS identity.	`rogue_nf`	proved	z03_upd	checked
Z03.10	An NF holding only its own valid token cannot cancel another NF's subscription. Attacker beyond the network: its own OAuth2 token for scope naf-inference and its own TLS identity.	`rogue_nf`	proved	z03_del	checked
Z03.10	A cancellation is performed only on behalf of the identity the token was issued to; a leaked token makes its holder that NF at the AF, which is the bearer-token limit the documents state. Attacker beyond the network: a leaked OAuth2 access token of a legitimate NF for scope naf-inference.	`stolen_token`	proved	z03_del	checked
Z03.10	An update is performed only on behalf of the identity the token was issued to, so a leaked token makes its holder that NF at the AF; the ownership check itself still holds. Attacker beyond the network: a leaked OAuth2 access token of a legitimate NF for scope naf-inference.	`stolen_token`	proved	z03_upd	checked
U01	A result the consumer accepts is bound to the subscription it created: the notification carries that subscription's notifCorreId and goes to its notifUri, which the update leaves unchanged, so an old result cannot be labelled with a different configuration. Attacker beyond the network: none.	`network`	proved	a089	none
U01	A result the consumer accepts is bound to the subscription it created with the same notifCorreId and notifUri after the update, so another NF cannot attach a result to that subscription. Attacker beyond the network: its own OAuth2 token for scope naf-inference and its own TLS identity.	`rogue_nf`	proved	a089	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-20261115T000111, 2026-11-15. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 65, prover 10 s, wall 17 min, cost 0.145 USD.

ProVerif: 186 proofs run, 182 decided, 4 without an answer, 6 s, 58 answers recorded.

What was analysed

Protocol	AF inference local trained model branch
Version	19.2.0
Analysed	Select5.5.2.2.2Unselect explicitly allowedNWDAFVFLClits, onlyAFBranches of locally trained models producing results; Trustable.AFandNFConsumer, one subscription/update/notification/cancellation. Count as many missions as multiple centsAPIdifferent configurations.
Not analysed	Symbolic models do not support the calculation of algorithms as safe or as resistant to quantum security; The trajectory of the attack is established only under clear assumptions and is not directly equivalent to a normative loophole. Each nature must be subject to accessibility/proportional controls. FixedRel-19。 Distinguishing between an outside cyber attacker and an ill-intentioned and legitimate person holding his own evidence.NF； TLSAuthenticate encryption modeling by endpoint, validNFCan not open messageNFIt's...TLSSession.OAuth2Subject/udience/scope/expiry andTLS(a) The identity is bound in accordance with the attached baseline; Service scope does not automatically equal ownership of resources. Single-modules cannot simulate real deletions, and the life cycle of a limited state must be clearly defined. Targets are specified.APISubprocess, not complete 5G secure.
Source text	318 numbered passages (173955 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	5.5	Naf_Inference Service
main	5.5.1	Service Description
main	5.5.1.1	Overview
main	5.5.1.2	Service Architecture
main	5.5.1.3	Network Functions
main	5.5.1.3.1	Application Function (AF)
main	5.5.1.3.2	NF Service Consumers
main	5.5.2	Service Operations
main	5.5.2.1	Introduction
main	5.5.2.2	Naf_Inference_Subscribe
main	5.5.2.2.1	General
main	5.5.2.2.2	Inference Subscription Creation
main	5.5.2.2.3	Inference Subscription Update

Document	Section	Title
main	5.5.2.3	Naf_Inference_Unsubscribe
main	5.5.2.3.1	General
main	5.5.2.3.2	Inference Subscription Deletion
main	5.5.2.4	Naf_Inference_Notify
main	5.5.2.4.1	General
main	5.5.2.4.2	Inference Notification
main	6.4	Naf_Inference Service API
main	6.4.1	Introduction
main	6.4.2	Usage of HTTP
main	6.4.2.1	General
main	6.4.2.2	HTTP standard headers
main	6.4.2.2.1	General
main	6.4.2.2.2	Content type
main	6.4.2.3	HTTP custom headers
main	6.4.3	Resources
main	6.4.3.1	Overview
main	6.4.3.2	Resource: Inference Subscriptions
main	6.4.3.2.1	Description
main	6.4.3.2.2	Resource Definition
main	6.4.3.2.3	Resource Standard Methods
main	6.4.3.2.3.1	POST
main	6.4.3.2.4	Resource Custom Operations
main	6.4.3.3	Resource: Individual Inference Subscription
main	6.4.3.3.1	Description
main	6.4.3.3.2	Resource Definition
main	6.4.3.3.3	Resource Standard Methods
main	6.4.3.3.3.1	PUT
main	6.4.3.3.3.2	PATCH
main	6.4.3.3.3.3	DELETE
main	6.4.3.2.4_2	Resource Custom Operations
main	6.4.4	Custom Operations without associated resources
main	6.4.5	Notifications
main	6.4.5.1	General
main	6.4.5.5	Inference notification
main	6.4.5.5.1	Description
main	6.4.5.5.2	Target URI

Document	Section	Title
main	6.4.5.5.3	Standard Methods
main	6.4.5.5.3.1	POST
main	6.4.6	Data Model
main	6.4.6.1	General
main	6.4.6.2	Structured data types
main	6.4.6.2.1	Introduction
main	6.4.6.2.2	Type InferEventSubsc
main	6.4.6.2.3	Type InferEventSubscPatch
main	6.4.6.2.4	Type InferAnaSub
main	6.4.6.2.5	Type InferNotif
main	6.4.6.2.6	Type InferResult
main	6.4.6.3	Simple data types and enumerations
main	6.4.6.3.1	Introduction
main	6.4.6.3.2	Simple data types
main	6.4.7	Error Handling
main	6.4.7.1	General
main	6.4.7.2	Protocol Errors
main	6.4.7.3	Application Errors
main	6.4.8	Feature negotiation
main	6.4.9	Security
main	6.4.10	HTTP redirection

The choice of the program as the source step listed it in the documents:5.5.2.2.2Unselect explicitly allowedNWDAFVFLClits, onlyAFBranches of locally trained models producing results; Trustable.AFandNFConsumer, one subscription/update/notification/cancellation. Count as many missions as multiple centsAPIdifferent configurations.

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	Responses bound to the consumer's own request	The NF service consumer (NWDAF) is assured that every successful response it accepts to subscription creation/update/deletion matches a request it made itself, with the same subscriptionId, notifUri and notifCorreId; the AF is assured that every notification it sends belongs to a subscription that consumer created or updated, with matching notifUri and notifCorreId.	The creation and update procedures pair an HTTP request from the NF service consumer with a response carrying the created/updated Individual Inference Subscription resource and a Location URI (s0040, s0049, s0055, s0061); notification is sent to the notifUri received from the consumer with the notifCorreId of that subscription (s0086); the data structures define both fields as mandatory (s0237, s0246).	s0004, s0021, s0040, s0049, s0086, s0237, s0246
A08	Notification bound to the local-model inference	The NF service consumer is assured that a result produced by the AF from its local trained ML model in the no-VFL-client branch is computed for that subscription (its determined VFL correlation ID and inferAnaSubs) and is delivered to that subscription's notifUri under that subscription's notifCorreId.	s0048 allows the VFL server, when no NWDAF VFL client is selected, to generate the results from its local trained ML model associated with the determined VFL correlation ID; s0086 requires the notification to go to the notifUri of the corresponding subscription and to carry the InferNotif structure with notifCorreId (s0246).	s0048, s0086, s0246
Z01	Valid token required for each operation	The AF is assured that every protected operation on the Naf_Inference API (subscription creation, update, deletion) is requested by an NF consumer holding an OAuth2 access token valid for that request, and that a request without such a token has no protected side effect.	6.4.9 states that access to the Naf_Inference API may be authorized by OAuth2 client credentials with the NRF as authorization server and that an NF service consumer shall obtain a token before consuming the service (s0272, s0273); the POST table lists 403 Forbidden and the general SBI error codes apply (s0143).	main:7257, main:7259, s0272, s0273, s0143
Z03	Token holder is the subscription creator	The AF is assured that a token used to update or delete a given subscriptionId belongs to the NF consumer that created that subscription, that the token's audience is this AF, and that the token cannot be used for another subscription or another operation.	The API defines a single service-level scope and no resource- or operation-level scope (s0275); the update and deletion procedures are directed at the Individual Inference Subscription resource by its URI (s0055, s0074) and 403 Forbidden is a defined response for the resource methods (s0143); the task scope notes that a service scope does not by itself mean ownership of a resource.	main:7263, s0275, s0055, s0074, s0143
Z04	No notifications after unsubscribing	The NF service consumer is assured that after its unsubscription succeeds (204 No Content) the AF sends no further notifications for that subscription, and that an update against that subscriptionId no longer takes effect.	Naf_Inference_Unsubscribe deletes the Individual Inference Subscription resource (s0067, s0074) and success is 204 No Content (s0075); Naf_Inference_Notify reports events for subscriptions that consumer has subscribed to (s0079, s0086); InferResult also allows the AF to request termination with termCause (s0249).	s0067, s0074, s0075, s0079, s0086, s0249
S05	Old identifiers rejected after unsubscription	The AF is assured, and the consumer can rely, that requests still naming the old subscriptionId, notifUri or notifCorreId of a deleted subscription neither create notifications nor modify the subscription.	Deletion removes the Individual Inference Subscription resource matching the subscriptionId (s0075, s0191); notifications are issued only for a previously subscribed consumer on the notifUri and notifCorreId of the corresponding subscription (s0086).	s0075, s0191, s0086
K02	Target identities and results kept from other NFs	The NF service consumer is assured that the target identities it sends in inferAnaSubs (supis, gpsis, intGroupIds, exterGroupIds) and the inference results it receives are kept from other NF consumers and from the external attacker.	The InferEventSubsc and InferNotif structures carry the target identities and the results (s0237, s0246, s0216); the API is a service-based interface protected as described in the security clause (s0272, s0275) and the task scope requires that a legitimate NF cannot read another NF's TLS session.	s0237, s0246, s0216, s0272

ID	Name	About	What the documents say	Where
K01	Token and TLS keys kept from the attacker	The NF service consumer is assured that neither its OAuth2 access token nor the TLS keys of its session with the AF become known to another NF or to the external attacker.	6.4.9 makes the token the authorization credential for the API (s0272-s0274); the API is reached over HTTP with HTTP standard headers and TLS as described in TS 29.500 (s0103, s0109).	main:7257, main:7259, s0272, s0273, s0103
A08.9	Result carries matching subscription and correlation ids	The NF service consumer is assured that every result the AF returns or notifies carries the subscriptionId and the correlation id determined for that subscription at creation or update.	The creation and update procedures return the representation of the Individual Inference Subscription resource identified by its URI (s0049, s0061); the notification carries the notifCorreId of the corresponding subscription (s0086, s0246) and results are those of the subscription's analytics events (s0248).	s0048, s0049, s0061, s0086, s0246
Z03.10	No cross-subscription update or cancel	The AF is assured, and each NF service consumer can rely, that an NF holding only its own valid token cannot update or delete a subscription another NF created.	There is one service-level scope and no per-resource scope (s0275); update and deletion address the Individual Inference Subscription resource by URI (s0055, s0074) and 403 Forbidden is defined for the resource methods (s0143); the task scope states that a service scope does not by itself mean ownership of a resource.	main:7263, s0275, s0055, s0074, s0143
U01	No stale results after an update	The NF service consumer is assured that after an update of its subscription takes effect, the results the AF returns or notifies are not based on the pre-update subscription configuration, and that a result produced before the update cannot be taken as one for the updated configuration.	Upon success the AF updates the targeted Individual Inference Subscription resource accordingly (s0061), and the notification delivers results for the subscribed analytics events and reporting information of that subscription (s0086, s0237, s0249).	s0055, s0058, s0061, s0086, s0237

The analyst added no property: the drafted list covers what the documents state.

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: 20. Not taken: 378. 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	Preface and pointer to the OAuth threat model: no requirement and no attack on this protocol.
P003	rfc6749:2790-2795	K01	
P004 to P006	rfc6749:2804-2823	Outside the scope	Client passwords of native and user-agent clients and redirection URIs: grant types this NF-to-NF client-credentials API does not use.
P007	rfc6749:2827-2829	Z03, Z03.10, `rogue_nf`, `stolen_token`	
P008 to P010	rfc6749:2831-2850	Outside the scope	Requirements on the authorization server and on resource owner interaction; neither is part of the Naf_Inference subflow.
P011	rfc6749:2859-2865	K01, K02, `network`	

Paragraphs	Where	Taken up as	Why not
P012	rfc6749:2867-2868	Outside the scope	Implicit grant type; the Naf_Inference API uses the client credentials grant (6.4.9).
P013	rfc6749:2870-2872	Z01	
P014	rfc6749:2874-2877	Z03	
P015	rfc6749:2879-2881	Z03, Z03.10, `stolen_token`	
P016 to P021	rfc6749:2885-2917	Outside the scope	Refresh tokens: the client credentials grant used here has none.
P022 to P029	rfc6749:2921-2980	Outside the scope	Authorization code grant and redirection URI manipulation: grant type and user-agent flow not used by this API.
P030 to P033	rfc6749:2984-3003	Outside the scope	Resource owner password credentials grant: not used by this API.
P034 to P035	rfc6749:3013-3019	K01, K02	
P036	rfc6749:3023-3028	K01	
P037 to P039	rfc6749:3032-3041	No goal stated	Guessing of client passwords at the authorization server: this API carries no client password and token issuance is outside the selected subflow.
P040 to P048	rfc6749:3045-3110	Outside the scope	Phishing and cross-site request forgery: user-agent flows, not an NF-to-NF API.
P049 to P050	rfc6749:3119-3138	Outside the scope	Clickjacking: user-agent flow.
P051 to P052	rfc6749:3142-3150	Outside the scope	Code injection in the authorization server's own inputs: not this API.
P053 to P054	rfc6749:3154-3167	Outside the scope	Open redirectors in the authorization endpoint: not this API.
P055 to P062	rfc6749:3170-3210	Outside the scope	Implicit grant misuse: grant type not used by this API.
P063	ts-33-501:2491-2491	Outside the scope	Bidding down of AS/NAS security algorithms: RAN level, not the Naf_Inference subflow.
P064 to P072	ts-33-501:2495-2511	Outside the scope	Authentication and authorization of the UE in the 5G core: the task scope is the Naf_Inference API subflow, not the whole 5G security architecture.
P073 to P074	ts-33-501:2515-2517	Outside the scope	Keys of the 5G core and NG-RAN: outside the API subflow.
P075 to P078	ts-33-501:2523-2529	Outside the scope	General user data protection on the air interface.
P079 to P089	ts-33-501:2533-2553	Outside the scope	User and signalling data confidentiality of the RAN; the API payload protection is asked by K01 and K02 under the TLS assumption of the task scope.

Paragraphs	Where	Taken up as	Why not
P090 to P103	ts-33-501:2557-2583	Outside the scope	User and signalling data integrity of the RAN.
P104 to P110	ts-33-501:2587-2599	Outside the scope	Storage of subscription credentials in the UE and UICC.
P111 to P125	ts-33-501:2603-2631	Outside the scope	Subscriber privacy (SUCI, SUPI concealment) on the air interface and in the core; the target identities of this API are carried inside the TLS-protected subscription request (K02).
P126 to P178	ts-33-501:2637-2761	Outside the scope	gNB, ng-eNB, F1 and E1 requirements: radio access, part of the system this task leaves out.
P179 to P196	ts-33-501:2767-2805	Outside the scope	AMF NAS-signalling confidentiality, integrity and subscriber privacy: not the API subflow.
P197 to P213	ts-33-501:2809-2853	Outside the scope	SEAF, UDM, AUSF and UDR requirements: authentication and subscription data of the UE.
P214	ts-33-501:2859-2859	`rogue_nf`	
P215	ts-33-501:2865-2865	Outside the scope	Replay protection for NF discovery and registration; no replay requirement is stated for the inference subscription API.
P216 to P217	ts-33-501:2867-2869	Outside the scope	NRF authorization of discovery and topology hiding: discovery, not this API.
P218	ts-33-501:2871-2871	A03, Z01, `network`	
P219	ts-33-501:2873-2873	Z01, `network`	
P220	ts-33-501:2875-2875	A03, Z01	
P221 to P227	ts-33-501:2879-2891	Z01	
P228 to P234	ts-33-501:2895-2907	Outside the scope	NEF security requirements for external exposure: the untrusted-AF configuration of this API, a different configuration from the trusted-AF branch modelled here.
P235 to P241	ts-33-501:2911-2923	Outside the scope	Service Communication Proxy: indirect communication, left out by the task scope.
P242 to P254	ts-33-501:2929-2953	Outside the scope	Inter-PLMN communication and SEPP: outside the single-operator API subflow.

Paragraphs	Where	Taken up as	Why not
P255 to P303	ts-33-501:2957-3055	Outside the scope	SEPP and roaming intermediary requirements: outside the subflow.
P304 to P312	ts-33-501:3059-3075	Outside the scope	Protection of attributes across N32: roaming interface.
P313 to P314	ts-33-501:3079-3081	Outside the scope	IPUPS requirements: border protection, outside the subflow.
P315 to P318	ts-33-501:3085-3091	Outside the scope	Originating network identifier header for roaming: outside the subflow.
P319 to P327	ts-33-501:3097-3115	Outside the scope	Signalling monitoring configuration and streaming: outside the subflow.
P328 to P337	ts-33-501:3121-3141	Outside the scope	Security visibility and configurability requirements on operators: deployment, not the API.
P338 to P369	ts-33-501:3149-3215	Outside the scope	NAS ciphering and integrity algorithm identifiers and selection: RAN/NAS, and the strength of the API's TLS suite is not askable of the prover.
P370 to P398	ts-33-501:3219-11901	Outside the scope	5G-RG, NSSAAF, N3, EAP-TLS, IAB, MBS and NEF annex requirements: parts of the system this task leaves out.

The parties and the protected values, one by one

The host listed the 23 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: 3. Without: 20. 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 Consumers of services	party	`rogue_nf`	
S02	AF	party	Everything lost by definition	TheAFproduces the local-model results, sees every token and is the party that enforces the authorization and ownership checks; a dishonestAFvoids all of them by definition, and this branch's scope fixes theAF♪ As trusted ♪AF), the untrusted variant being theNEFconfiguration.
S03	NRF	party	Everything lost by definition	A hostile authorization server can mint a valid token for any NF identity, so every authorization property asked here fails by definition; the documents ask nothing under a compromised NRF (33.501 5.9.2.2 has NRF and NFs mutually authenticated).
S04	Other NF	party	`rogue_nf`	
S05	External assailants	party	Nothing gained	The external attacker is the attacker who has the network and nothing else; that attacker is present for every target by default and holds nothing beyond the network, so no further attacker follows from it.

ID	Party or value	Kind	Attacker	Why none
S06	subscriptionId	value	Nothing gained	The subscriptionId is a resource identifier given to the subscriber inside its TLS session; knowing one gives no ability, because the AF authorizes an update or a deletion by the caller's token and by the stored owner of that resource, which is what Z03 and Z03.10 ask and what the ownership queries show.
S07	notifUri	value	Nothing gained	Accepting a notification does not depend on knowing the notifUri: the notification body is sealed and authenticated under the AF/consumer session key and must carry the consumer's own notifCorreId (A03), so a party that only knows the URI gains nothing.
S08	notifCorreId	value	Nothing gained	The notifCorreId is checked together with the seal of the notification under the AF/consumer session key, so knowing a correlation id alone does not let a party produce an acceptable notification (A03).
S09	Subscriptions	value	Nothing gained	The subscription content says only what another consumer asked for; it gives no ability to modify a subscription or to receive its results, both of which need the owner's token (Z03, Z03.10).
S10	vflCorrId	value	Nothing gained	The VFL correlation ID is determined by the AF and used to select the local model; knowing it does not yield the result, which comes out of the AF's private model (A08).
S11	Locally trained ML model references	value	Nothing gained	A reference to the AF's local trained model is a name, not a secret; the model is used only inside the AF.
S12	The reasoning.	value	Nothing gained	Holding one inference result gives no further ability: notifications are sealed to the subscription holder and bound to its own correlation id, so a result cannot be turned into an accepted notification (A03, A08).
S13	OAuth2 Access tokens	value	`stolen_token`	
S14	naf-inference scope	value	Nothing gained	The service scope is a public label defined for the whole API; it authorizes nothing by itself, the token does (Z01).
S15	Subscription resources	value	Nothing gained	The subscription resource lives at the AF; holding a copy of its representation or its URI gives no ability to change it or to receive its notifications without the owner's token.
S16	Subscription Status	value	Nothing gained	The subscription state is enforced at the AF by the existence and owner check on every update and deletion; knowing the state adds no ability. The state after a successful deletion is answered on the unrolled model because a monotone table cannot remove a row.
S17	Announcements	value	Nothing gained	A notification is a sealed blob under the AF/consumer session key; holding one gives no ability beyond the result it carries, and a replay carries the same result and the same correlation id.
S18	supis	value	Nothing gained	The target identities (supis, gpsis, intGroupIds, exterGroupIds) say whom an analytics subscription is about; holding them gives no ability to subscribe, cancel or receive results, which is asked by K02.
S19	gpsis	value	Nothing gained	The target identities (supis, gpsis, intGroupIds, exterGroupIds) say whom an analytics subscription is about; holding them gives no ability to subscribe, cancel or receive results, which is asked by K02.
S20	intGroupIds	value	Nothing gained	The target identities (supis, gpsis, intGroupIds, exterGroupIds) say whom an analytics subscription is about; holding them gives no ability to subscribe, cancel or receive results, which is asked by K02.
S21	exterGroupIds	value	Nothing gained	The target identities (supis, gpsis, intGroupIds, exterGroupIds) say whom an analytics subscription is about; holding them gives no ability to subscribe, cancel or receive results, which is asked by K02.

ID	Party or value	Kind	Attacker	Why none
S22	TLS Session Key	value	Outside the scope	The task scope states that the symbolic model does not prove computational security: an attacker holding a TLS session key is a broken-primitive attacker the analysis leaves out.
S23	Subscription Configuration	value	Nothing gained	The subscription configuration is held by the AF and by its owner; knowing it gives no ability to change it, which needs the owner's token, and no ability to obtain results of another subscription.

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		
`rogue_nf`	A legitimate NF service consumer's own credentials: an OAuth2 access token issued by the NRF for the single scope "naf-inference" (6.4.9), its own TLS identity towards the AF, and its own API root/URI space; it cannot read the TLS sessions of other NFs (task scope).	6.4.9 says access may be authorized with OAuth2 client credentials, one scope for the whole service (s0272, s0275); the task scope separates the external network attacker from a malicious legitimate NF holding its own credentials.	main:7257, main:7263, s0272, s0275
`stolen_token`	An OAuth2 access token that was issued to a legitimate NF consumer for scope "naf-inference" and has leaked to it; it can present that token in service requests over its own connection.	6.4.9 makes the OAuth2 token the credential for accessing Naf_Inference (s0272-s0274); RFC 6749 10.3 (line 2879-2881) states that the specification gives the resource server no method to ensure that a token presented to it was issued to that presenter, and requires access tokens to be kept confidential in transit and storage (line 2859-2865).	main:7257, rfc6749:2879, s0272, s0273, s0274

ID	`network`	`rogue_nf`	`stolen_token`
A03	Holds	Holds	
A08	Holds	Holds	
Z01	Holds	Holds	
Z03	Holds	Holds	Holds
Z04	Holds	Holds	
S05	Holds	Holds	
K02	No answer	No answer	No answer
K01	Holds	Holds	
A08.9	Holds	Holds	
Z03.10	Holds	Holds	Holds
U01	Holds	Holds	

Notes on the prover and the model

- Z04: answered in ProVerif, and the analyst names changing_state for this target
- S05: 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
whether the subscriptionId still denotes an existing Individual Inference Subscription, and who owns it	the AF (VFL server), for each Individual Inference Subscription resource	a subscription is created by a successful POST (201 Created, s0049), replaced by a successful PUT/PATCH (s0061) and ended by a successful DELETE (s0075)	before every PUT/PATCH or DELETE naming the subscriptionId, and before a notification is sent for the subscription (s0086); in model.pv the check is the existence and owner lookup on the stored subscription	s0049, s0055, s0061, s0074, s0075, s0086
the notifCorreId and notifUri that bind a notification to a subscription	the NF service consumer (notifCorreId and the callback endpoint behind notifUri) and the AF (notifUri and notifCorreId stored for the subscription)	the creation request (s0040, s0237) and every update request (s0055, s0058)	on every notification the consumer receives: the notifCorreId carried in InferNotif must be the one of a subscription of that consumer, and the notification must open under the AF/consumer session key	s0086, s0237, s0246

pattern	result	target	shown by ProVerif	steps
repeat_one_time_action	nothing found			The chain in bounded.pv refuses the second operation after the deletion; model.pv's monotone table would accept a repeated DELETE because a row cannot be removed, which is a limit of the tool, not of the protocol.
replay	nothing found			InferNotif carries no sequence number or timestamp (s0246) and the consumer compares only notifCorreId (s0086), so an identical notification replayed by the network would be accepted twice. The task scope models TLS as authenticated encryption between the endpoints, which prevents record replay, so this stays inside the TLS assumption and breaks no target: no query is claimed for it.
desynchronisation	nothing found			The consumer and the AF agree on the subscription through the notifCorreId, which the update does not change, and through the notifUri the AF stores and uses; no pair of values the parties keep can be made to differ by a message the source allows, because every update and deletion is checked against the stored owner before it is applied.

pattern	result	target	shown by ProVerif	steps
update_before_check	nothing found			The AF answers an update or a deletion only after the existence and owner check on the stored subscription; z03_upd and z03_del show that no update or deletion is performed except by the creator.
no_rollback_on_failure	nothing found			A refused notification attempt and a refused update after the deletion leave the AF's state as it was in bounded.pv: the refusals emit AfRefusedNotif and AfRefusedUpd and do not move the chain back to the live state.
race_between_sessions	nothing found			Each request is bound to its session by the reply channel the consumer creates for it, and the AF checks the stored owner of the subscriptionId it is given; two consumers acting at once cannot change each other's subscription, as z03_upd and z03_del show for a malicious NF.
distinguishable_failure	nothing found			The source leaves the error code of a refused update or deletion to clause 6.4.7 (403 or 404), and both are answers to the caller only; the model does not model the error body, so nothing can be concluded about a probe that would tell an existing subscription from a non-existing one. The subscriptionId is a fresh value the AF generates and gives only to the subscriber, so such a probe would need the identifier first.
wrap_or_window	nothing found			No counter, sequence number or acceptance window is kept: the only value that changes is whether the subscription exists, and its two values are handled by the chain of bounded.pv, so nothing can wrap around.

What the analyst named for targets:

step	targets	features	reason
51	Z04, S05	changing_state	Z04 and S05 rest on the value the AF keeps and changes for each Individual Inference Subscription: whether the subscription still exists. In model.pv the subscription table is monotone (a row cannot be removed), so the effect of a successful deletion on later notifications and on later updates naming the old subscriptionId is unrolled in bounded.pv to four steps (create, delete, notification attempt, update attempt), with the state compared before the notification or the update is accepted.

The analyst's closing note

Eleven targets answered on the trusted-AF, local-model branch of Naf_Inference (TS 29.572 v19.2.0), no finding: the protocol gives what the targets ask within the assumptions analysed.

Established by proof on model.pv (any number of sessions): the AF creates a subscription only for the consumer's own request and notifies only for that subscription's notifUri and notifCorreId (A03); the notified result is the one computed for that subscription's correlation ID (A08) and the consumer accepts only a notification of its own correlation id, sealed under the AF/consumer session (A08.9); every creation, update and cancellation is requested by the identity holding the naf-inference token (Z01); an update or cancellation is applied only on behalf of the subscription's creator, with a falsified control (ctl_z03) showing the check is what carries it (Z03, Z03.10); the token and the TLS key stay with the consumer (K01); post-update results stay bound to the subscription (U01, binding part).

Established on bounded.pv, the subscription lifecycle unrolled to four steps, each with a falsified control: after a successful deletion the AF neither notifies nor applies an update for that subscription (Z04, S05).

Weaker answers, stated as such: the confidentiality of the inference results (K02) is recorded only at the level of reachability, because ProVerif verified the implication but found no trace for its premise; the freshness of results after an update cannot be asked of this data model at all, since InferNotif carries no version or sequence number, so U01's freshness claim rests on the AF recomputing.

Blocks and limits: the malicious-NF and token-thief cases are scripted processes rather than free adversary computation, subscription identifiers are fresh values the source does not specify, TLS and OAuth2 are taken as sound (a broken primitive or a leaked session key is out of scope), and a monotone table forced the deletion properties into the bounded model. Six proofs (A03, A08, A08.9, Z01, K01, U01) have no control query. The K02 confidentiality question and the per-session state around the notification are a case for Tamarin with an agent.