Security analysis of AF VFL client inference API

This report in another language: 中文.

19.2.0 · ProVerif+ Agent report

Documents	5G System; Application Function Artificial Intelligence/Machine Learning (AI/ML) Services; Stage 3 (protocol definition)
Properties examined	15: 12 hold, 3 fail (5 attacks)
Findings	2

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_VFLInference service exposes a trusted Application Function acting as a VFL client to the Network Data Analytics Function, or to the Network Exposure Function when the AF is untrusted; the AF holds a pre-trained model and produces VFL inference results for the analytics events a consumer asks for. The consumer, the AF and the Network Repository Function as OAuth2 authorization server take part. The consumer obtains an access token for the AF and its naf-vflinference scope, subscribes with a notifUri, a notifCorrId and a description of the analytics event, and receives 201 Created with the URI of the new subscription; it may then update the subscription and unsubscribe with it. The AF notifies each subscribed inference event to the notifUri of the subscription with the notifCorrId and the vflInferResults, and the consumer stores the notification and answers 204. When the exchange ends the consumer holds results that belong to its own subscription, from the AF instance it selected, and the AF holds a subscription whose owner it authenticated with TLS and authorised with the token. Nothing in the exchange ties a subscription to the consumer that owns it beyond the token's scope, and nothing lets either side recognise a repeated request.

The parties:

- **NWDAF**: NF service consumer, VFL server. Holds: NF identity, OAuth2 client credentials.
- **AF**: trusted AF, VFL client. Holds: NF instance identity, pre-trained model.
- **NRF**: OAuth2 authorization server. Holds: token signing key, NF profiles.

Figure 1 shows the exchange as the documents specify it.

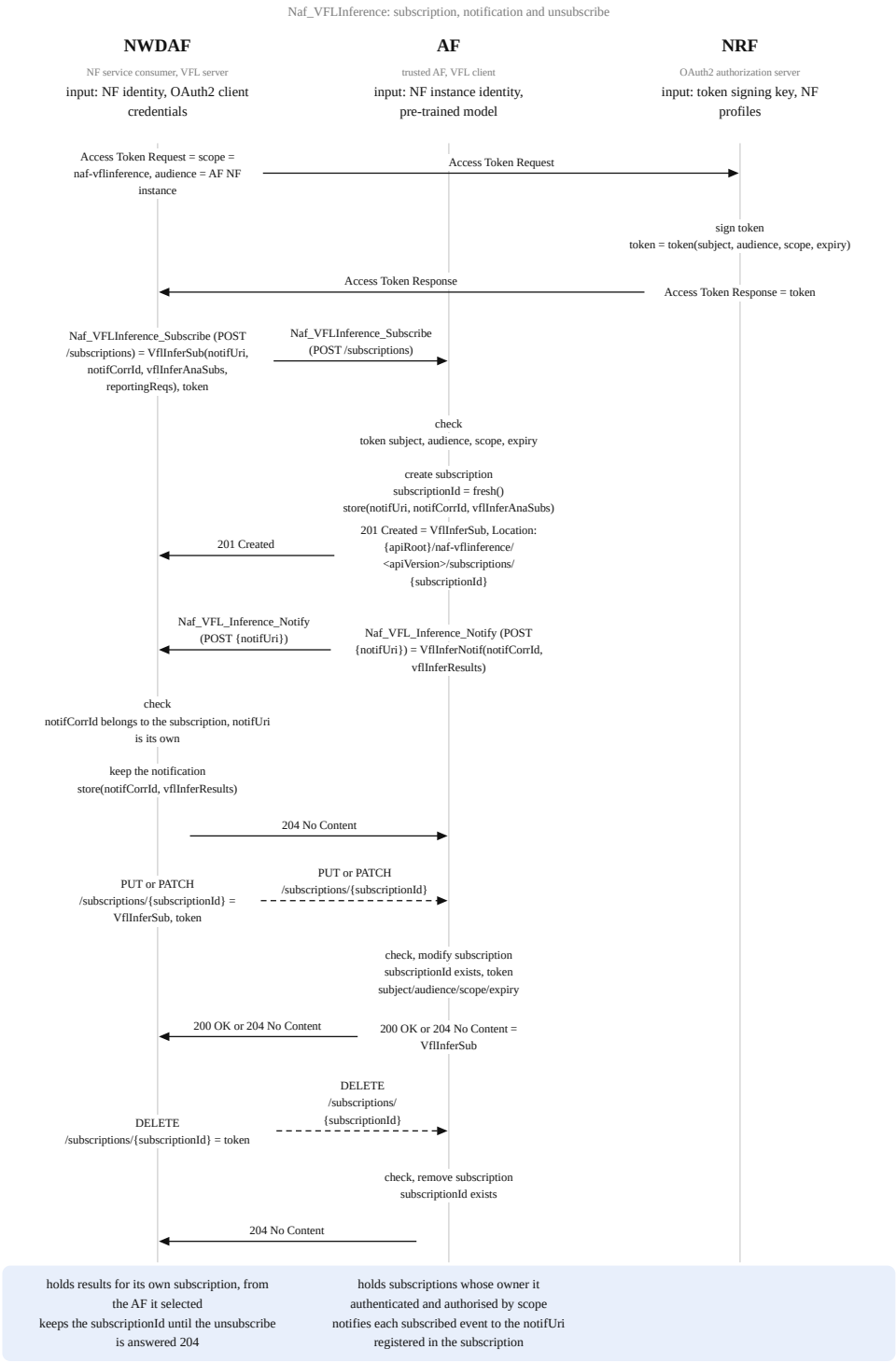


Figure 1. The protocol as specified.

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 2 findings: what follows is not stated in the documents. For 2 of them it is unclear whether the documents state them.

The notification path is sound: a consumer accepts results only from the AF instance it selected, for its own subscription, and the notifCorrId and notifUri of every notification come from one stored subscription; token confidentiality, its subject and audience binding, and the confidentiality of the user identifiers hold against the network. The resource path is not: the API has one service-wide scope and requires no check that the requester owns a subscriptionId, so any NF with a valid token that knows one subscriptionId can replace or delete another consumer's subscription, and a replaced notifUri sends the victim's inference results to the attacker. There is no duplicate detection anywhere: replayed notifications are accepted and stored twice, a replayed subscribe creates a second subscription, and a replayed DELETE is answered 204 again; with the subscription state kept, the unsubscribe is applied once and no notification is accepted after it. Two results rest on a bound (one subscription unrolled to three steps) and all of them rest on TLS endpoint authentication and on a symmetric OAuth2 token, so the guarantees are as strong as those assumptions, and the trusted AF itself is assumed honest.

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 the service scope can replace or delete another consumer's subscription

Kind	Weakness of the protocol as specified
New	Unclear: the documents state it in part, or could not be searched
Value	high
Properties	A08 (section 3)
Shown by	ProVerif attack queries `a08_upd_own`, `a08_del_own`; Figure 2

What happens. The specification defines one scope for the whole service and no scope at resource or operation level, and it requires no check that the requester of PUT, PATCH or DELETE owns the subscriptionId. Any NF that holds a naf-vflinference token and knows a subscriptionId can therefore replace or delete another consumer's subscription. The AF in the model checks the token subject, audience, scope and expiry and still accepts the operation, because nothing in the source ties the subscriptionId to its owner.

What the attacker needs. The attacker is a legitimate NF with its own credentials and a token for the naf-vflinference scope. It must know the victim's subscriptionId, which travels in the resource URI of the requests and in the Location header, so an intermediate node or a log gives it.

What the attacker gains. The attacker replaces the registered notifUri and the subscribed analysis of another consumer's subscription, so the AF sends that consumer's vflInferResults to the attacker's callback, or it deletes the subscription and the consumer stops receiving results while it still believes it is subscribed.

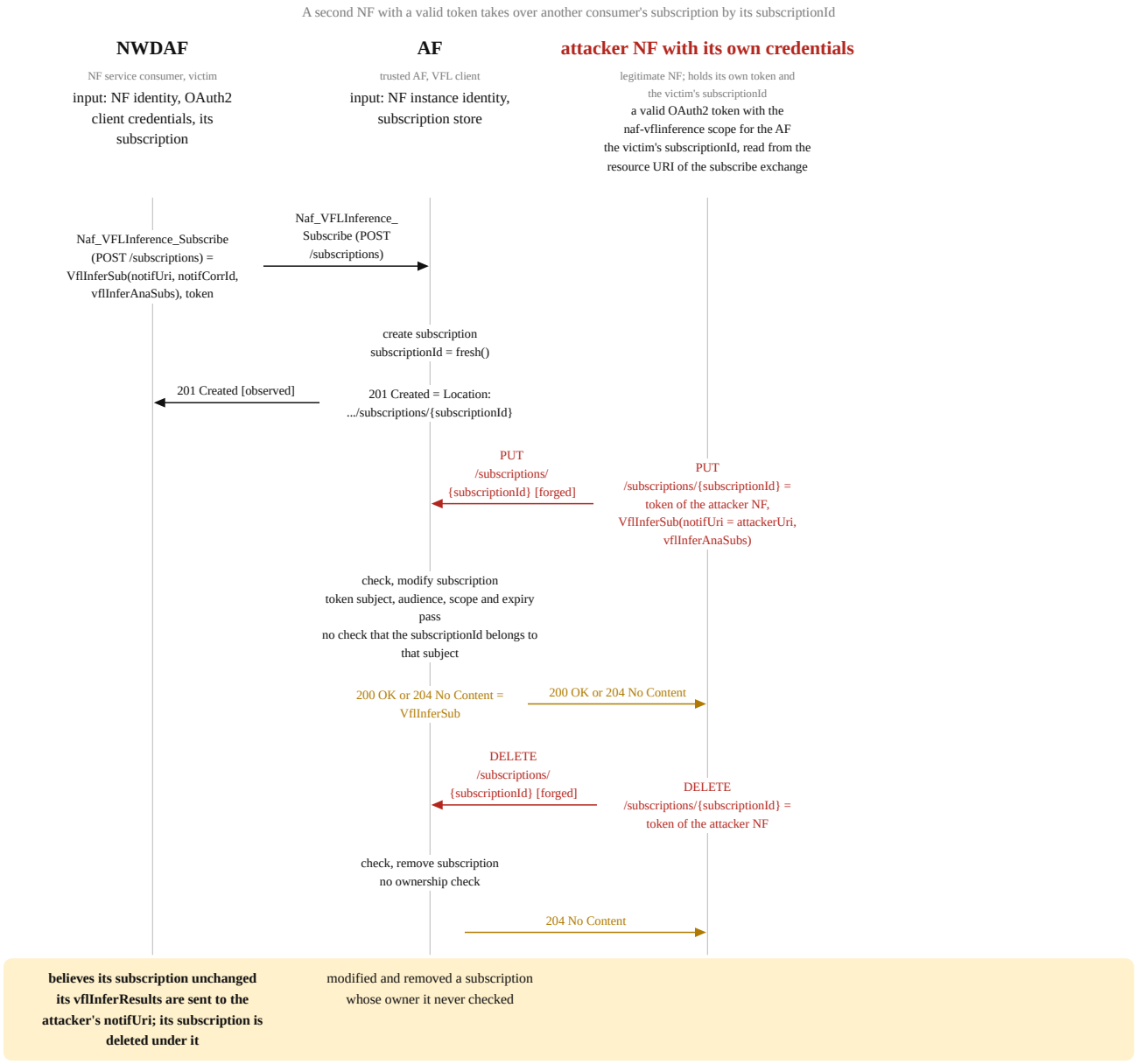


Figure 2. A second NF with a valid token takes over another consumer's subscription by its subscriptionId. Property A08; attack queries a08_del_own, a08_upd_own.

Proposed change. Require the AF to check, on PUT, PATCH and DELETE, that the subscriptionId belongs to the NF instance that is the subject of the presented access token, and answer 403 otherwise; keep the notifCorrId and notifUri used for notifications taken only from the stored subscription.

Check of the change with ProVerif. The change was put into a copy of the model: On PUT, PATCH and DELETE the AF shall check that the subscriptionId belongs to the NF instance that is the subject of the presented access token (the stored subscription must name the requester as its consumer) and answer 403 (or 404) otherwise. On the changed model ProVerif proves `a08_upd_own`, `a08_del_own`: the attack is no longer possible there. The normal run of the protocol is still found. The other 16 recorded answers are unchanged. This is a first check on the model: it shows that the change removes this attack there, not that it introduces no other problem. Changed model: `records/fixes/0044-fix-owner.pv`.

2.2 Replayed notifications and replayed subscribe and delete requests are accepted again

Kind	Weakness of the protocol as specified
New	Unclear: the documents state it in part, or could not be searched
Value	medium
Properties	R02, R03 (section 3)
Shown by	ProVerif attack queries `r02_once`, `r03_once_sub`, `r03_once_del`; Figures 3 and 4

What happens. A captured notification POST can be delivered again and the consumer accepts and stores it a second time; a captured subscribe POST creates a second subscription resource; a captured DELETE is accepted twice. Neither the messages nor the specified behaviour of the receiving party carry a value that tells a repeat from a new request, and the consumer has no state that would refuse a notification it has already consumed.

What the attacker needs. Only the network: the attacker keeps and re-sends a message it has already seen, and every message it re-sends is one the AF or the consumer accepted before.

What the attacker gains. A duplicated notification is processed and stored twice, and a duplicated subscribe leaves a second subscription resource with its own subscriptionId, so the consumer is charged twice for the same analytics event and results are delivered twice; a duplicated delete gets a second 204 for a resource that no longer exists.

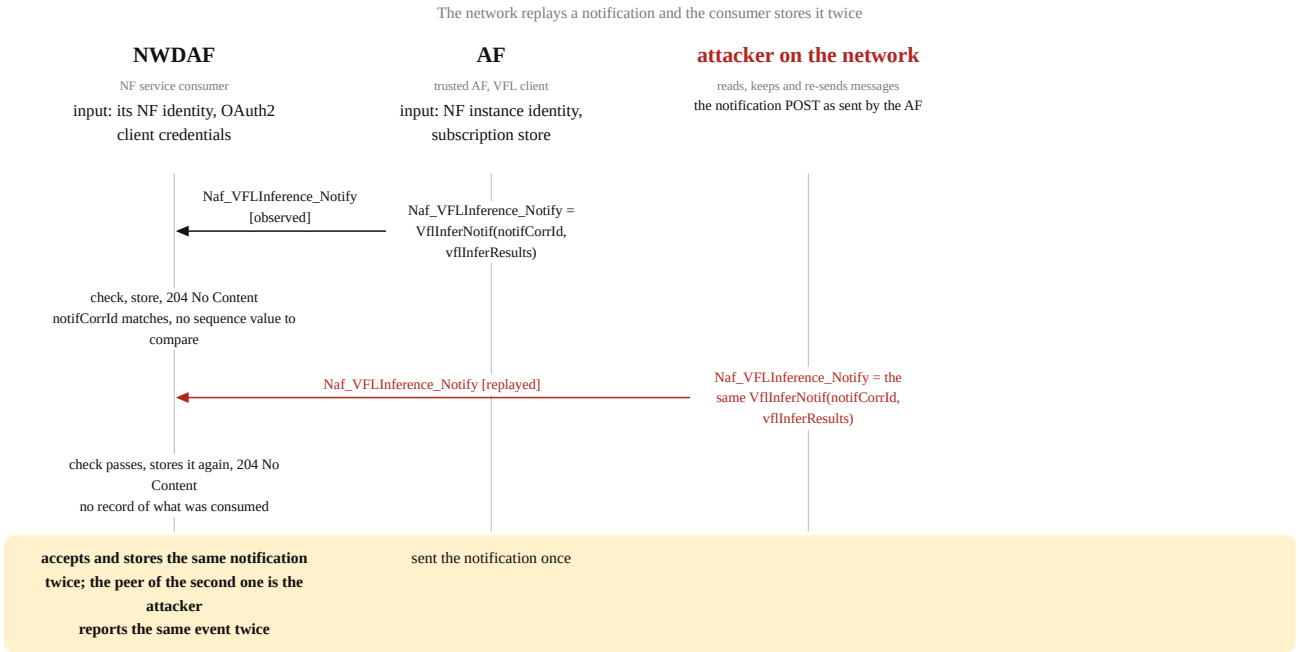


Figure 3. The network replays a notification and the consumer stores it twice. Property R02; attack query r02_once.

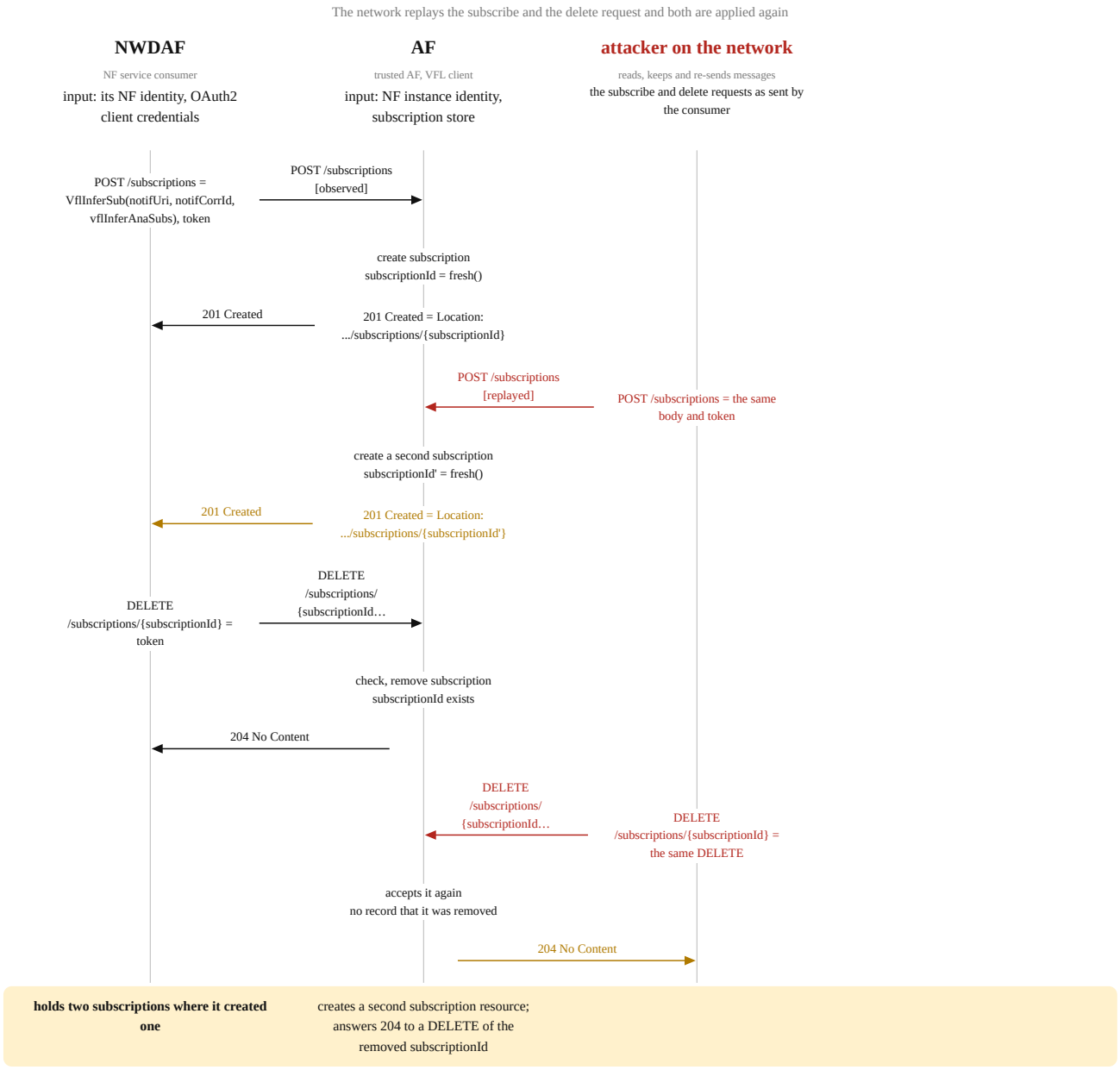


Figure 4. The network replays the subscribe and the delete request and both are applied again. Property R03; attack queries r03_once_del, r03_once_sub.

Proposed change. Give the notification POST a value with which the consumer recognises a repeat, for example a notification counter per subscription in VflInferNotif that the consumer compares with the last value it accepted; state that the AF refuses a subscribe request whose notifCorrId it already holds a subscription for, and that it answers 404 to a DELETE for a subscriptionId it has removed.

Check of the change with ProVerif. Not checked: The change adds a value that the AF increments per notification and the consumer compares, so a model of it is a value that changes across a run; ProVerif can only hold such a value in the unrolled chain of bounded.pv, which covers one subscription and three steps, and the check would then be shown only for the first notifications of that one subscription — the same bound the two bounded results already carry. The source defines no such field, so the counter, its scope (per subscription or per consumer) and the refusal rule would all be invented by the recommendation, and a model of an invented rule does not check the source.

No figure was given for these attacks of a finding: R02: r03_once_del, R02: r03_once_sub, R03: r02_once.

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
A02	**Request and notification authenticity.** Whenever the AF accepts a subscribe/update/unsubscribe request from a consumer C, or the consumer accepts a notification from the AF, it claims that the peer really did run the corresponding operation as itself.	no compromise	Holds	low	no
A03	**Notification matches a matching run.** Whenever the consumer accepts a notification, there is a matching AF run whose notifCorrId, notifUri and subscription content agree; whenever the AF accepts a subscribe/update request, there is a matching consumer run.	no compromise	Holds	medium	no
A08	**Operations bound to own subscriptionId.** Every update/partial update/delete of an Individual VFL Inference Subscription acts on the subscriptionId created by the requester itself, and every notification is sent to the notifUri registered for that subscription and carries the corresponding notifCorrId.	no compromise	Fails (2 attacks)	high	unclear
Z01	**Valid scoped token required.** Every subscription creation, update, partial update, delete or notification submission must have a valid token issued by the authorization server that covers the naf-vflinference scope.	no compromise	Holds	medium	no
Z03	**Token bound to subject and audience.** A token issued for one consumer and one AF audience cannot be used to perform operations on another consumer's subscription, nor on another AF instance.	no compromise	Holds	medium	yes
Z04	**Deleted subscription stops notifications.** After a subscription is deleted, the AF no longer sends notifications for that subscriptionId; after a subscription is updated, notifications are sent with the updated content.	no compromise	Holds within a bound (the changing value unrolled to 3 steps)	medium	no
S05	**No notification after unsubscribe.** After the unsubscribe takes effect, the old combination of subscriptionId and notifUri can no longer trigger a new notification.	no compromise	Holds within a bound (the changing value unrolled to 3 steps)	medium	no
S04	**Notification fields from same subscription.** The notifCorrId carried by each notification and the notifUri it is sent to both come from values registered for the same subscription at its creation or update.	no compromise	Holds	low	no
R02	**Notification accepted at most once.** The same logical notification instance (the same notifCorrId and the same vflInferResults) is accepted by the consumer at most once.	no compromise	Fails (1 attack)	medium	unclear
R03	**Subscribe and unsubscribe applied once.** The same logical subscription creation request produces at most one subscription resource; the same logical unsubscribe request deletes at most once.	no compromise	Fails (2 attacks)	medium	unclear
K02	**Results kept from other NFs.** vflInferResults and the user identifier attributes in the subscription are visible only to the subscription's consumer and the AF; other NFs or external attackers cannot obtain them.	no compromise	Holds	medium	yes

ID	Property	Compromise assumed	Result	Value	In the documents
K01	**Access token kept from other NFs.** The OAuth2 access token is visible only to the authorization server that issued it, the consumer that holds it and the AF that validates it; other entities cannot derive it.	no compromise	Holds	medium	yes
U01	**Results match the subscribed request.** The inference results the consumer accepts correspond to the input samples and model version requested by this subscription.	no compromise	Holds	low	no
U02	**Notification from selected AF.** The notification the consumer accepts comes from the selected AF and belongs to the consumer's own subscription.	no compromise	Holds	medium	no
Q3	**Subscription state after cancel and update.** The finite-state behaviour after cancelling and updating a subscription, and the handling of error responses.	no compromise	Holds within a bound (the changing value unrolled to 3 steps)	medium	no

The questions asked, and the analyst's answers:

Question	Answer	Rests on
Q1: Correlation of input samples/model version with inference results	The results a consumer accepts are exactly the ones the AF produced for a subscription with that notifCorrId, notifUri and analysis description; the source carries no field of the result that names a sample or a model version, so that part of the question cannot be answered from these units and is left to the VflInferResult structure of TS 29.520.	U01, A03
Q2: Authentication of the notification as coming from the selected AF, and cross-session isolation	A notification is accepted only if it was produced by the AF instance the consumer subscribed at, for a subscription the consumer created itself with the same notifCorrId, notifUri and content, so one session cannot be confused with another; a second AF instance cannot notify a consumer that did not select it.	U02, A02, A03
Q3: Finite-state behaviour of cancel/update and handling of error responses	With the subscription state kept, a subscription is modified by PUT and removed by DELETE, a repeated DELETE is refused, and after the 204 of the unsubscribe no notification is accepted for that subscription; the failure paths themselves are not modelled, and the source allows an error answer without a body, so nothing is concluded about what the error responses reveal.	Q3, Z04, S05, R03

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 a per-endpoint key: the model proves nothing about the TLS handshake, certificates or cipher suites, and every message is assumed to reach the endpoint named in it.
- OAuth2 is modelled with a token signing key shared between the NRF and the AF; the token format, introspection and revocation of TS 29.510 and TS 33.501 are not modelled.
- Time is frozen at one instant, so the expiry check of a token is not tested: an expired token cannot be distinguished from a valid one in this model.
- PATCH is not modelled separately; it follows the same acceptance path as PUT, and only bounded.pv lets a PUT change the stored subscription.

- model.pv keeps subscriptions in a monotone table, so a removal cannot be observed there; the ordering properties of a subscription's life are answered on bounded.pv, which unrolls one subscription to three steps.
- The user identifiers (supis, gpsis, group identifiers) are one opaque value per AF; the structure of VflInferSub and VflInferResult in TS 29.520, and the samples and the numerical algorithm, are outside the model.
- The notification is one message over TLS to the notifUri; the HTTP transaction, redirection by an SCP and any store-and-forward in the AF are not modelled.
- An attacker that controls the AF or reads what it stores is assumed away: the task fixes the trusted AF, whose honesty is a precondition of every result here.

What the host records:

- The answers of Z04, S05, R02, R03, Q3 rest on an operation the model simplifies (Appendix D).
- Proved without a control query (Appendix C): A02: a02_req, A02: a02_notif, A03: a03_bind, A08: s04_notif_uri, Z01: z01_sub, Z01: z01_upd, Z01: z01_del, Z03: z01_sub, Z04: b_notif_created, S04: s04_notif_uri, U01: a02_notif, U02: a03_bind, Q3: b_notif_created. 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.

A02: Request and notification authenticity

Whenever the AF accepts a subscribe/update/unsubscribe request from a consumer C, or the consumer accepts a notification from the AF, it claims that the peer really did run the corresponding operation as itself.

The AF accepts a subscription only from the NF instance that the TLS session and the token name, and the consumer accepts a notification only from the AF instance it selected, for the subscription it created. So each side knows whom it is talking to, as long as TLS authenticates the endpoints as the model assumes.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: It confirms what the SBI baseline already requires and what no reader doubts; its worth is that the other results rest on it.
- Stated in the documents: no
- Rests on: TLS endpoint authentication is assumed, not proved: the model gives each endpoint a private key that its peer can also derive, and proves nothing about the handshake.; The token is a term signed by the NRF with a key shared with the AF; token introspection and revocation are outside the model.

A03: Notification matches a matching run

Whenever the consumer accepts a notification, there is a matching AF run whose notifCorrId, notifUri and subscription content agree; whenever the AF accepts a subscribe/update request, there is a matching consumer run.

A notification the consumer accepts always corresponds to a subscription that the AF created for that consumer, with the same notifCorrId, notifUri and analysis description. The correspondence holds because the AF takes both values from the stored subscription and the consumer compares them with its own.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: It is the binding the whole notification path relies on, proved rather than assumed; it stands or falls with the model's TLS abstraction.
- Stated in the documents: no
- Rests on: The AF takes notifCorrId and notifUri from the stored subscription; a PUT that changes them is applied in bounded.pv only.; TLS endpoint authentication is assumed.

A08: Operations bound to own subscriptionId

Every update/partial update/delete of an Individual VFL Inference Subscription acts on the subscriptionId created by the requester itself, and every notification is sent to the notifUri registered in that subscription and carries the corresponding notifCorrId.

The notification side is sound: the AF notifies the notifUri registered in the subscription with its notifCorrId. The resource side is not: the AF checks only that the requester holds a token with the service-wide scope, so any NF that knows a subscriptionId can replace or delete another consumer's subscription, and with the replacement its results are sent elsewhere.

- Result: Fails (2 attacks); answers what is asked: partly; usable with caveats
- Value high: It exposes a gap in the resource model of a Rel-19 API: the token scope authorises the service, not the resource, and one subscriptionId is enough to take over another consumer's subscription. The documents do not say this is safe.
- Stated in the documents: unclear. Source text it rests on: s0044, s0082, s0190, s0218, s0291
- Rests on: The AF performs no ownership check, as the source specifies no such check; the only attestation is that s0291 defines a single scope for the whole service.; The attacker has its own credentials and knows one subscriptionId; the model does not decide whether learning a subscriptionId is possible in a deployment.
- Open: Whether a subscriptionId is confidential to the consumer: the source never says, and this decides whether the attack is reachable in a deployment.

Z01: Valid scoped token required

Every subscription creation, update, partial update, delete or notification submission must have a valid token issued by the authorisation server and covering the naf-vflinference scope.

Every subscribe, update and delete the AF accepts is backed by a token the NRF issued for that consumer, for that AF instance, with the naf-vflinference scope and the current expiry. The check is as strong as the equality of the token term with what the NRF signs, which the model gives the AF.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: It shows that the OAuth2 access control of the API is sound against the network once the AF validates the token, which the source makes optional (may be authorised), so the result holds only where OAuth2 is deployed.
- Stated in the documents: no
- Rests on: The NRF signs tokens with a key that is a private name shared with the AF; the model has one instant, so expiry is compared only as a value.; The audience travels inside the protected token request; a token request tampered with by an intermediate node is not considered.
- Open: The expiry of the token is frozen at one instant, so an operation with a lapsed token is not tested.
- Next step: Model token expiry with two token variants that the AF compares against the current instant, to test the expiry check.

Z03: Token bound to subject and audience

A token issued for one consumer and one AF audience cannot be used to perform operations on another consumer's subscription, nor can it be used at another AF instance.

A token is accepted only at the AF instance and for the consumer it was issued for, because the AF compares the token with the exact term the NRF signs for its own identity and the requesting consumer. What OAuth2 cannot give is the link between the client that presents a token and the client it was issued to, which the model supplies through the TLS session.

- Result: Holds; answers what is asked: partly; usable with caveats
- Value medium: The audience and subject binding is proved for this API, and the limitation that makes it necessary is stated by RFC 6749, so the result is a known boundary shown mechanically.
- Stated in the documents: yes. Where the documents say it: rfc6749:2879, s0288, s0289, s0291
- Rests on: Subject and audience binding rests on the TLS session that identifies the consumer, not on the token alone, as RFC 6749 notes that it gives the resource server no method to make that check.; The model has one NRF key and one instant.

- Open: The part of the claim about another consumer's subscription is answered by the attack on the operations, not by the token: see A08.

Z04: Deleted subscription stops notifications

After a subscription is deleted, the AF no longer sends notifications for that subscriptionId; after a subscription is updated, notifications are sent according to the updated content.

With the subscription state kept, the AF notifies only for a subscription it holds, its notifications after a PUT carry the modified analysis description, and after the 204 of the DELETE the consumer accepts nothing more for that subscriptionId. An unsubscribe that is answered therefore stops the notifications, in the three steps the model unrolls.

- Result: Holds within a bound (the changing value unrolled to 3 steps); answers what is asked: yes; usable with caveats
- Value medium: It answers the ordering question the source leaves open with a checked model, and the control shows the property depends on the state being kept.
- Stated in the documents: no. Source text it rests on: s0083, s0084, s0085
- Rests on: bounded.pv unrolls one subscription to three steps: one subscribe, one PUT and one DELETE; it does not cover a second subscription, a PATCH or a failure answer.; The consumer is assumed to drop its local subscription when it receives the 204; the source says only that the AF removes the resource.
- Open: Whether the AF may notify for a subscription it removed after the 204 but before the removal is complete: the unrolled model has no such window.

S05: No notification after unsubscribe

After the unsubscribe takes effect, the old subscriptionId and notifUri combination can no longer trigger new notifications.

After the unsubscribe is answered, the old subscriptionId and notifUri pair produces no notification the consumer accepts, once the AF holds the subscription as state and the consumer drops it on the 204. Without that state the same notification is accepted again, as the control run shows.

- Result: Holds within a bound (the changing value unrolled to 3 steps); answers what is asked: partly; usable with caveats
- Value medium: It separates the unsubscribe of the source from the state a consumer needs, which the source does not specify.
- Stated in the documents: no. Source text it rests on: s0083, s0084, s0085
- Rests on: The result is bounded: three unrolled steps and one subscription.; An attacker that re-sends the notification to a consumer which keeps no state would still be accepted; that is the replay result, and it is not a distinction of this target.
- Open: Whether an AF that removes the subscription but keeps a retry or a cache would notify once more; the source says nothing about it.

S04: Notification fields from same subscription

The notifCorrId carried by each notification and the notifUri it is sent to both come from the values registered in the same subscription at its creation or update.

Every notification the AF sends carries the notifCorrId of one subscription and goes to the notifUri registered in that same subscription, so a consumer that checks both cannot be handed a notification of another subscription. The binding is structural: both values are read from the same stored row.

- Result: Holds; answers what is asked: yes; usable
- Value low: It confirms a structural property of the resource definition rather than a security decision of the protocol.
- Stated in the documents: no
- Rests on: The subscription is stored as one row and the notification is built from it; the model has no cache and no retry queue.; A PUT that changes the notifUri is applied only in bounded.pv.
- Open: Where the AF takes notifUri after a PATCH: PATCH is not modelled separately.

R02: Notification accepted at most once

The same logical notification instance (the same notifCorrId and the same vflInferResults) is accepted by the consumer at most once.

The claim fails: the notification carries no value that tells a repeat from a new event, and the consumer keeps no record of what it consumed, so the network replays a notification and the consumer accepts and stores it twice. This is a state the source does not define, not a missing model check.

- Result: Fails (1 attack); answers what is asked: yes; usable
- Value medium: It shows mechanically that a one-time or periodic notification can be made to arrive twice, and the exact absence of a sequence value is what an editor would have to fix.
- Stated in the documents: unclear. Source text it rests on: s0095, s0096, s0097, s0098
- Rests on: The consumer accepts a notification whose notifCorrId and notifUri match its subscription, as s0096 and the 204 of s0098 imply, and nothing more.; The attack needs only the network.
- Open: Whether a consumer implementation is required to detect a repeat; the source does not say that it is not.

R03: Subscribe and unsubscribe applied once

The same logical subscription creation request produces at most one subscription resource; the same logical unsubscribe request deletes at most once.

Both claims fail without state: a replayed subscribe POST creates a second subscription resource with its own subscriptionId, and a replayed DELETE is accepted and answered 204 again. With the subscription kept as state the AF accepts the removal once, which is what bounded.pv shows.

- Result: Fails (2 attacks); answers what is asked: partly; usable with caveats
- Value medium: It gives the exact repeats the API admits and shows what a subscription state would buy, which is what an editor needs for an idempotency rule.
- Stated in the documents: unclear. Source text it rests on: s0043, s0044, s0084, s0085
- Rests on: bounded.pv unrolls one subscription; the repeated subscribe is shown on model.pv, where subscriptions are never removed.; The AF in the model never rejects a repeat, which matches a source that defines no idempotency key.
- Open: What the AF answers to a DELETE for a subscriptionId it has removed: the source does not say, and the model does not answer.

K02: Results kept from other NFs

vflInferResults and the user identifier attributes in the subscription are visible only to the consumer of the subscription and the AF; other NFs or external attackers cannot obtain them.

Neither the network nor another NF learns the user identifiers or the inference results during the exchange; the results travel only inside the consumer's TLS session and are computed from a user identifier the AF never sends in the clear. The compromise run shows the property is not vacuous: with the AF revealing what it holds, the identifiers become known.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: Confidentiality of user identifiers and results against the network is the promise the API is built on; the documents state it at the level of TLS and OAuth2 principles rather than for these fields, so the mechanical result adds the exact conditions.
- Stated in the documents: yes. Where the documents say it: rfc6749:2859, s0045, s0097, s0115, s0262
- Rests on: TLS confidentiality is assumed for the whole exchange; no attacker is given the key of another endpoint.; The user identifiers are one opaque value per AF, not the list of supis or group identifiers of the subscription.
- Open: A malicious NF that obtains the consumer's TLS key, which the model does not give it.

K01: Access token kept from other NFs

The OAuth2 access token is visible only to the authorization server that issued it, the consumer that holds it and the AF that validates it; other entities cannot derive it.

The access token is known only to the NRF, the consumer and the AF it is addressed to; a network attacker and other NFs never see it. The control shows the property depends on the token travelling inside TLS: with the signing key revealed, the attacker builds a token and the property fails.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: It is the confidentiality RFC 6749 requires, proved for this API against the network, with the caveat that the token scheme is a symmetric one.
- Stated in the documents: yes. Where the documents say it: rfc6749:2859, rfc6749:2870, s0288, s0289
- Rests on: Tokens are signed with a symmetric key shared by the NRF and the AF, so the AF sees them in the clear; a public-key or reference-token scheme is not modelled.; The compromise attacker reveals the signing key, not a consumer's stored token.
- Open: A token that leaks from the consumer's storage, which the model does not model as an attacker.

U01: Results match the subscribed request

The inference results the consumer accepts correspond to the input samples and model version requested by this subscription.

The results a consumer accepts are exactly those the AF computed for a subscription with that correlation id, notifUri and analysis description, so a consumer cannot be handed the results of another subscription of the same AF. Whether a result is tied to the samples and model version the consumer asked for cannot be asked of this source, which carries no such field.

- Result: Holds; answers what is asked: partly; usable with caveats
- Value low: It answers the part of the question the selected units can carry and names precisely what is missing for the rest.

- Stated in the documents: no. Source text it rests on: s0040, s0097, s0262
- Rests on: The AF computes the result from the stored analysis description; the sample and the model version are one opaque model value.; TLS endpoint authentication is assumed.
- Open: Whether a result names the samples and the model version it was computed from: the field structure is in TS 29.520, outside the selected units.

U02: Notification from selected AF

The notifications the consumer accepts come from the selected AF and belong to the consumer's own subscription.

The consumer accepts a notification only from the AF instance it subscribed at and only for a subscription it created; a second AF instance that was given the same request cannot hand the consumer a notification. Session isolation follows from the per-endpoint TLS key and from the AF identity carried in the notification.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: It answers the practical worry of a consumer about which AF is speaking to it and isolates sessions; the caveat is that the AF identity is modelled rather than carried in a specified field.
- Stated in the documents: no. Source text it rests on: s0029, s0036, s0095
- Rests on: The notification carries the AF identity in the model; the source defines no AF identity field in the notification body, so the model supplies what TLS and the URI would carry.; TLS endpoint authentication is assumed.

Q3: Subscription state after cancel and update

The finite-state behaviour after cancelling and updating a subscription, and the handling of error responses.

With the subscription state kept, the AF modifies the subscription on PUT, notifies the modified content, removes it on DELETE and answers, and a repeated DELETE is refused; after the 204 the consumer accepts no further notification. The finite state is therefore consistent between the two sides in the unrolled run, and the source defines nothing beyond it.

- Result: Holds within a bound (the changing value unrolled to 3 steps); answers what is asked: partly; usable with caveats
- Value medium: It gives the state machine the specification leaves implicit, and separates what holds with a state from what the source does not require.
- Stated in the documents: no. Source text it rests on: s0192, s0205, s0218, s0275
- Rests on: bounded.pv unrolls three steps of one subscription and does not model PATCH, retries or failure answers.; The consumer drops its subscription on the 204 although the source does not require it to.
- Open: The error answers: 403, 404 and the 29.500 codes are not modelled, so nothing is said about what a refusal reveals.

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
A02	proved		a02_req			2 proved
A03	proved		a03_bind			1 proved
A08	proved		s04_notif_uri		a08_upd_own (holds_in_source); a08_del_own (holds_in_source)	1 proved, 2 attack
Z01	proved		z01_sub			3 proved
Z03	proved		z01_sub			1 proved
Z04	bounded	the changing value unrolled to 3 steps	b_notif_created	1 of 2		2 bounded
S05	bounded	the changing value unrolled to 3 steps	b_afterdel	checked		1 bounded
S04	proved		s04_notif_uri			1 proved
R02	none				r02_once (holds_in_source)	1 attack
R03	bounded	the changing value unrolled to 3 steps	b_del_once	checked	r03_once_sub (holds_in_source); r03_once_del (holds_in_source)	1 bounded, 2 attack
K02	proved		k02_secret	checked		1 proved
K01	proved		k01_secret	checked		1 proved
U01	proved		a02_notif			1 proved
U02	proved		a03_bind			1 proved
Q3	bounded	the changing value unrolled to 3 steps	b_notif_created			1 bounded

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
A02	The AF accepts a subscribe request only from the NF service consumer whose identity the request carries: every subscription the AF creates corresponds to a subscribe request that consumer actually sent.	`network`	proved	a02_req	none
A02	The consumer accepts a notification only if it was produced by the AF it selected, with the same notifCorrId, notifUri and results that AF sent for that subscription.	`network`	proved	a02_notif	none
A03	Every notification the consumer accepts corresponds to a subscription the AF created for that consumer, with the same notifCorrId, notifUri and vflInferAnaSubs.	`network`	proved	a03_bind	none
A08	Every notification the AF sends for a subscription goes to the notifUri registered in that subscription, carrying its notifCorrId.	`network`	proved	s04_notif_uri	none

target	claim	attacker	answer	query	control
A08	A PUT on an Individual VFL Inference Subscription acts only on a subscriptionId the requester itself created. The attacker has, beyond the network, its own valid NF credentials and a token with the naf-vflinference scope, and it knows another consumer's subscriptionId from the resource URI.	`uri_reader`	attack (holds_in_source)	a08_upd_own	
A08	A DELETE acts only on a subscriptionId the requester itself created. The attacker has its own valid NF credentials and scope token, and knows another consumer's subscriptionId from the resource URI.	`uri_reader`	attack (holds_in_source)	a08_del_own	
Z01	A subscription is created only for a request that carried a token the NRF issued for that consumer and that AF instance with the naf-vflinference scope and an expiry that has not passed.	`network`	proved	z01_sub	none
Z01	A PUT is accepted only for a request that carried a token the NRF issued for that consumer and that AF instance with the naf-vflinference scope and a valid expiry.	`network`	proved	z01_upd	none
Z01	A DELETE is accepted only for a request that carried a token the NRF issued for that consumer and that AF instance with the naf-vflinference scope and a valid expiry.	`network`	proved	z01_del	none
Z03	A token is accepted only at the AF instance and for the consumer it was issued for: an operation the AF accepts implies an NRF-issued token whose subject is that consumer and whose audience is that AF instance.	`network`	proved	z01_sub	none
Z04	A notification is sent only for a subscription the AF still holds, and after an accepted PUT it carries the modified vflInferAnaSubs.	`network`	bounded (the changing value unrolled to 3 steps)	b_notif_created	none
Z04	After the DELETE has been answered 204, the consumer accepts no further notification for that subscriptionId.	`network`	bounded (the changing value unrolled to 3 steps)	b_afterdel	checked
S05	After the unsubscribe has been accepted, the old subscriptionId and notifUri pair triggers no notification that the consumer accepts.	`network`	bounded (the changing value unrolled to 3 steps)	b_afterdel	checked
S04	Every notification the AF sends carries the notifCorrId and goes to the notifUri of one and the same subscription it created.	`network`	proved	s04_notif_uri	none
R02	The same logical notification, the same notifCorrId with the same vflInferResults, is accepted by the consumer at most once.	`network`	attack (holds_in_source)	r02_once	
R03	A repeated subscribe request creates at most one subscription resource.	`network`	attack (holds_in_source)	r03_once_sub	
R03	A repeated unsubscribe request deletes at most once.	`network`	attack (holds_in_source)	r03_once_del	
R03	With the subscription state kept, a replayed DELETE is refused: the AF that holds the subscription accepts it once, and a state of the resource records the removal.	`network`	bounded (the changing value unrolled to 3 steps)	b_del_once	checked
K02	The vflInferResults and the user identifiers of a subscription are kept from the network and from other NFs: the attacker learns no user identifier while a notification is accepted.	`network`	proved	k02_secret	checked

target	claim	attacker	answer	query	control
K01	The OAuth2 access token is known only to the NRF, the consumer it was issued to and the AF it is addressed to; no other NF and no network attacker learns it during the exchange.	network	proved	k01_secret	checked
U01	The results the consumer accepts are exactly the results the AF computed for a subscription with that correlation id, notifUri and analysis description, so a result is tied to the subscribed request rather than to a sample or model version the source does not carry.	network	proved	a02_notif	none
U02	The consumer accepts a notification only from the AF instance it subscribed at and only for a subscription it created itself.	network	proved	a03_bind	none
Q3	The finite state after PUT and DELETE: the AF modifies and then removes the subscription and its answers report that state, while a second notification is accepted only for the modified content.	network	bounded (the changing value unrolled to 3 steps)	b_notif_created	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-20261114T232417, 2026-11-14. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 48, prover 5 s, wall 14 min, cost 0.136 USD.

ProVerif: 91 proofs run, 91 decided, 0 without an answer, 3 s, 23 answers recorded.

What was analysed

Protocol	AF VFL client inference API
Version	19.2.0
Analysed	Trusted AF as VFL client, NWDAF server calls Naf_VFLInference directly; model pre-trained, one bounded session per inference request/report; does not include NEF, training, or numerical algorithms.
Not analysed	The symbolic model does not prove computational or post-quantum security of the algorithms; attack traces hold only under explicit assumptions and are not directly equivalent to specification vulnerabilities. Every property must have a reachability/positive control. Fixed to Rel-19. Separately distinguishes the external network attacker from a malicious legitimate NF holding 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. Monotonic tables cannot model real deletion, so the finite-state lifecycle must be explicitly bounded. The goal is the specified API sub-flow, not full 5G security. This is a re-run of the same protocol scope under the implementation version after correcting the table placement; the previous run was rejected by the implementation version lock during the preparation phase, and the version lock cannot be bypassed to reuse the old inputs.
Source text	336 numbered passages (155963 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	5.3	Naf_VFLInference Service
main	5.3.1	Service Description
main	5.3.1.1	Overview
main	5.3.1.2	Service Architecture
main	5.3.1.3	Network Functions
main	5.3.1.3.1	Application Function (AF)
main	5.3.1.3.2	NF Service Consumers
main	5.3.2	Service Operations
main	5.3.2.1	Introduction
main	5.3.2.2	Naf_VFLInference_Subscribe service operation
main	5.3.2.2.1	General
main	5.3.2.2.2	Subscription for VFL inference event notifications

Document	Section	Title
main	5.3.2.2.3	Update subscription for event notifications
main	5.3.2.2.4	Partial update subscription for event notifications
main	5.3.2.3	Naf_VFLInference_Unsubscribe service operation
main	5.3.2.3.1	General
main	5.3.2.3.2	Unsubscribe from VFL inference notifications
main	5.3.2.4	Naf_VFLInference_Notify service operation
main	5.3.2.4.1	General
main	5.3.2.4.2	Notification about subscribed event
main	6.2	Naf_VFLInference Service API
main	6.2.1	Introduction
main	6.2.2	Usage of HTTP
main	6.2.2.1	General
main	6.2.2.2	HTTP standard headers
main	6.2.2.2.1	General
main	6.2.2.2.2	Content type
main	6.2.2.3	HTTP custom headers
main	6.2.3	Resources
main	6.2.3.1	Resource Structure
main	6.2.3.2	Resource: VFL Inference Subscriptions
main	6.2.3.2.1	Description
main	6.2.3.2.2	Resource definition
main	6.2.3.2.3	Resource Standard Methods
main	6.2.3.2.3.1	POST
main	6.2.3.2.4	Resource Custom Operations
main	6.2.3.3	Resource: Individual VFL Inference Subscription
main	6.2.3.3.1	Description
main	6.2.3.3.2	Resource definition
main	6.2.3.3.3	Resource Standard Methods
main	6.2.3.3.3.1	GET
main	6.2.3.3.3.2	PUT
main	6.2.3.3.3.3	PATCH
main	6.2.3.3.3.4	DELETE
main	6.2.3.3.4	Resource Custom Operations
main	6.2.4	Custom Operations without associated resources
main	6.2.5	Notifications
main	6.2.5.1	General

Document	Section	Title
main	6.2.5.2	VFL Inference Event Notification
main	6.2.5.2.1	Description
main	6.2.5.2.2	Operation Definition
main	6.2.6	Data Model
main	6.2.6.1	General
main	6.2.6.2	Structured data types
main	6.2.6.2.1	Introduction
main	6.2.6.2.2	Type VflInferSub
main	6.2.6.2.3	Type VflInferSubPatch
main	6.2.6.3	Simple data types and enumerations
main	6.2.6.3.1	Introduction
main	6.2.6.3.2	Simple data types
main	6.2.7	Error handling
main	6.2.7.1	General
main	6.2.7.2	Protocol Errors
main	6.2.7.3	Application Errors
main	6.2.8	Feature negotiation
main	6.2.9	Security
main	6.2.10	HTTP redirection

The source step located the purpose of the protocol in the documents as follows: the trusted AF acts as the VFL client, and the NWDAF server calls Naf_VFLInference directly; the model is pre-trained; one inference request/report is a bounded session; NEF, training and numerical algorithms are out of scope.

The properties against the documents

Before writing a query, 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 do exist in the documents.

ID	Name	About	What the documents say	Where
A02	Request and notification authenticity	The AF is assured that a subscribe/update/delete request comes from the NF service consumer it names; the NF service consumer is assured that a notification comes from the AF it selected.	The service is consumed by NWDAF (trusted AF) or NEF (untrusted AF) only, and access may be authorized by OAuth2; mutual authentication of consumer and producer and validation of every incoming message are required by the SBI baseline; the source states no message-level authentication beyond TLS and OAuth2.	ts-33-501:2871, ts-33-501:2875, s0004, s0011, s0013, s0014, s0031, s0090, s0288, s0289

ID	Name	About	What the documents say	Where
A03	Notification matches a matching run	Each notification the consumer accepts corresponds to a subscription the consumer created, with the same notifCorrId, notifUri and subscription content; each subscription the AF creates corresponds to a consumer request.	The VflInferSub body carries notifUri, notifCorrId and vflInferAnaSubs (s0037-s0040); the notification carries notifCorrId and vflInferResults to {notifUri} (s0095-s0097). The source states no correspondence requirement in words; it is implied by the resource description.	s0037, s0038, s0039, s0040, s0095, s0096, s0097
A08	Operations bound to own subscriptionId	PUT, PATCH and DELETE on an Individual VFL Inference Subscription, and the notification sent for it, act for and towards the consumer that owns that subscriptionId.	The resource URI carries {subscriptionId} (s0044, s0082) and only the service-wide scope naf-vflinference exists, with no scope at resource or operation level (s0291); the source requires no ownership check for PUT/PATCH/DELETE.	ts-33-501:2875, s0044, s0045, s0082, s0190, s0218, s0291
Z01	Valid scoped token required	Every subscribe, update, partial update, delete and notification handling at the AF is authorized by a token the NRF issued, with the naf-vflinference scope and an expiry that has not passed.	Access may be authorized with OAuth2 client credentials, NRF as authorization server; the consumer shall obtain a token before consuming the service; the API defines the single scope naf-vflinference (s0288-s0291).	rfc6749:2859, ts-33-501:2883, s0288, s0289, s0290, s0291
Z03	Token bound to subject and audience	A token issued for one NF service consumer and one AF instance cannot be used to act at another AF instance or on another consumer's subscription.	The token is obtained from the NRF before consuming the service (s0288-s0290); RFC 6749 states that OAuth2 itself gives the resource server no way to check that a token presented by a client was issued to that client, which is the gap the SBI baseline must close by audience and subject checks.	rfc6749:2879, ts-33-501:2885, s0288, s0289, s0291
Z04	Deleted subscription stops notifications	After a DELETE is accepted for a subscriptionId, the AF sends no further notification for it; after a PUT or PATCH the notification follows the modified subscription.	On DELETE the AF removes the subscription and answers 204 (s0083-s0085); on PUT and PATCH it modifies the resource (s0190, s0197-s0209). The source does not say when the removed subscription stops producing notifications.	s0083, s0084, s0085, s0190, s0200, s0201
S05	No notification after unsubscribe	Once an unsubscribe is accepted, the old subscriptionId and notifUri pair cannot trigger a new notification.	The AF removes the subscription and answers 204 on DELETE (s0083-s0085); no retention or cache rule is stated.	s0082, s0083, s0084, s0085
S04	Notification fields from same subscription	The notifCorrId carried by a notification and the notifUri it is sent to both come from the same subscription, as registered at creation or update.	notifUri and notifCorrId are registered in the subscription (s0038, s0039) and the notification is sent to {notifUri} with that notifCorrId (s0095, s0096).	s0038, s0039, s0044, s0095, s0096
R02	Notification accepted at most once	The same logical notification, the same notifCorrId with the same vflInferResults, is accepted by the consumer at most once.	The AF notifies a subscribed, one-time or periodic, VFL inference event (s0022, s0095) and the consumer stores the notification and answers 204 (s0098); no notification sequence number or duplicate detection is specified.	ts-33-501:2865, s0022, s0095, s0096, s0097, s0098
R03	Subscribe and unsubscribe applied once	A repeated subscribe request creates at most one subscription resource, and a repeated unsubscribe deletes at most once.	POST creates a new Individual VFL Inference Subscription and answers 201 with a new subscriptionId (s0043, s0044); DELETE answers 204 when the resource matching subscriptionId was deleted (s0084, s0085, s0218). No idempotency key or retransmission rule is specified.	ts-33-501:2865, ts-33-501:2875, s0043, s0044, s0084, s0085, s0218

ID	Name	About	What the documents say	Where
K02	Results kept from other NFs	vflInferResults and the user identifiers of a subscription (supis, gpsis, intGroupIds, exterGroupIds) are visible only to the subscribing consumer and the AF.	The subscription carries user identifiers and the notification carries vflInferResults (s0045, s0097, s0262); transport protection is TLS per TS 33.501 (s0115) and the consumer is told to store the notification (s0098).	rfc6749:2859, ts-33-501:2865, s0045, s0097, s0115, s0262
K01	Access token kept from other NFs	The OAuth2 access token of a consumer is known only to the NRF, that consumer and the AF it is addressed to.	Access tokens are credentials that must be kept confidential in transit and storage and shared only among the authorization server, the resource server and the client (RFC 6749); the token is obtained from the NRF before consuming the service (s0288, s0289).	rfc6749:2859, rfc6749:2870, s0288, s0289
U01	Results match the subscribed request	The vflInferResults the consumer accepts are the results the AF produced for that consumer's subscription, i.e. for the analysis description of that subscription and the pre-trained model version.	The subscription describes the subscribed analytics event in vflInferAnaSubs (s0040) and the notification carries the results for it (s0097); the structure of the results is defined in TS 29.520, outside the selected units, so the source states no field tying a result to a sample or a model version.	s0040, s0045, s0097, s0262
U02	Notification from selected AF	The notification the consumer accepts comes from the AF instance the consumer selected and belongs to a subscription of that consumer.	The consumer subscribes at the AF it discovered (s0029, s0036) and the AF notifies the {notifUri} of that subscription (s0095); no AF identity field is defined for the notification body.	ts-33-501:2871, s0029, s0036, s0095, s0096
Q3	Subscription state after cancel and update	The finite state of a subscription after PUT, PATCH and DELETE, and the error responses of those methods, in particular whether a deleted subscription still produces notifications and whether a consumer can be brought to keep a different state than the AF.	PUT answers 200 or 204, PATCH 200 or 204, DELETE 204, with 403 and the 29.500 error codes (s0192, s0205, s0218, s0274-s0282); the source defines no state machine beyond removing and modifying the resource.	s0192, s0205, s0218, s0275, s0282

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: 39. Not taken: 359. 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	Introduction to the security considerations, advice to implementers and a reference to the OAuth threat model; states no property of this API.
P003 to P006	rfc6749:2790-2823	Z01, K01	
P007 to P010	rfc6749:2827-2850	A02, Z03, `malicious_nf`	
P011 to P014	rfc6749:2859-2877	K01, Z01	
P015	rfc6749:2879-2881	Z03	

Paragraphs	Where	Taken up as	Why not
P016 to P021	rfc6749:2885-2917	Outside the scope	Refresh tokens are not part of the client credentials grant this API uses (s0288-s0289).
P022 to P025	rfc6749:2921-2943	Outside the scope	The authorization code grant is not used; the NRF issues tokens to NF service consumers with the client credentials grant.
P026 to P029	rfc6749:2947-2980	Outside the scope	No authorization code flow and hence no authorization response redirection URI in this grant.
P030 to P033	rfc6749:2984-3003	No goal stated	Warnings about resource owner password credentials, a grant this API does not use and about which it states no property.
P034 to P035	rfc6749:3013-3019	K01, K02	
P036	rfc6749:3023-3028	A02, U02	
P037 to P039	rfc6749:3032-3041	The prover cannot be asked	The strength of client credentials and the number of guesses needed are outside what a symbolic prover can answer.
P040 to P042	rfc6749:3045-3066	No goal stated	Phishing against a resource owner; this API has no end user or user agent, so it states no property here.
P043 to P048	rfc6749:3070-3110	No goal stated	Cross-site request forgery needs a browser session with cookies; NF service consumers use client credentials over TLS, so no property of this API is stated.
P049 to P050	rfc6749:3119-3138	No goal stated	Clickjacking needs a user agent and framed content; not present in this API.
P051 to P052	rfc6749:3142-3150	Z01, A08	
P053 to P054	rfc6749:3154-3167	No goal stated	Open redirectors at the authorization endpoint; the redirections of this API are performed by an SCP and are outside the selected units.
P055 to P062	rfc6749:3170-3210	No goal stated	Misuse of an access token to impersonate a resource owner in the implicit grant; the implicit grant is not used and there is no resource owner.
P063	ts-33-501:2491-2491	The prover cannot be asked	Bidding down of security features is about algorithm negotiation strength, which a symbolic prover cannot judge.
P064 to P072	ts-33-501:2495-2511	Outside the scope	Subscription authentication, serving network authentication and UE authorization in the 5G access network; the task fixes the Naf_VFLInference API sub-procedure.
P073 to P074	ts-33-501:2515-2517	Outside the scope	Key lengths for AS/NAS/UP protection on the radio and N3 interfaces.
P075 to P078	ts-33-501:2523-2529	Outside the scope	Requirements on the UE and ng-eNB.
P079 to P089	ts-33-501:2533-2553	Outside the scope	User data and signalling confidentiality between UE and network.
P090 to P103	ts-33-501:2557-2583	Outside the scope	User data and signalling integrity between UE and network.
P104 to P110	ts-33-501:2587-2599	Outside the scope	Secure storage of subscription credentials in the UE and UICC.
P111 to P125	ts-33-501:2603-2631	Outside the scope	Subscriber privacy of SUPI on the air interface; the user identifiers carried in a VFL subscription are covered by K02 instead.
P126	ts-33-501:2637-2637	Outside the scope	General requirements on the gNB.

Paragraphs	Where	Taken up as	Why not
P127 to P136	ts-33-501:2641-2659	Outside the scope	gNB user data and signalling confidentiality.
P137 to P147	ts-33-501:2663-2683	Outside the scope	gNB user data and signalling integrity.
P148 to P156	ts-33-501:2687-2703	Outside the scope	gNB setup and configuration.
P157 to P158	ts-33-501:2707-2709	Outside the scope	Key management inside the gNB.
P159 to P160	ts-33-501:2713-2715	Outside the scope	Handling of user plane data in the gNB.
P161 to P162	ts-33-501:2719-2721	Outside the scope	Handling of control plane data in the gNB.
P163 to P168	ts-33-501:2725-2735	Outside the scope	Secure environment of the gNB.
P169 to P174	ts-33-501:2739-2749	Outside the scope	gNB F1 interface requirements.
P175 to P176	ts-33-501:2753-2755	Outside the scope	gNB E1 interface requirements.
P177 to P178	ts-33-501:2759-2761	Outside the scope	Requirements on the ng-eNB.
P179 to P185	ts-33-501:2767-2779	Outside the scope	Signalling data confidentiality on the NG interface.
P186 to P192	ts-33-501:2783-2795	Outside the scope	Signalling data integrity on the NG interface.
P193 to P196	ts-33-501:2799-2805	Outside the scope	Subscriber privacy on the NG interface.
P197 to P198	ts-33-501:2809-2811	Outside the scope	Requirements on the SEAF.
P199 to P201	ts-33-501:2819-2823	Outside the scope	Generic requirements on UDM/UDR subscription data handling.
P202 to P207	ts-33-501:2827-2837	Outside the scope	Subscriber privacy requirements on UDM and SIDF.
P208 to P210	ts-33-501:2841-2845	Outside the scope	Requirements on the AUSF.
P211 to P213	ts-33-501:2849-2853	Outside the scope	Requirements on the UDR.
P214	ts-33-501:2859-2859	`network`	
P215 to P220	ts-33-501:2865-2875	A02, Z01, R02, R03, `network`	
P221 to P227	ts-33-501:2879-2891	Z01, Z03, K01, `malicious_nrf`	
P228 to P234	ts-33-501:2895-2907	Outside the scope	NEF exposure of an untrusted AF is excluded by the task, which fixes the trusted-AF direct call.
P235 to P241	ts-33-501:2911-2923	`uri_reader`	

Paragraphs	Where	Taken up as	Why not
P242 to P254	ts-33-501:2929-2953	Outside the scope	Roaming and inter-PLMN security, including SEPP; the modelled path is a single trust domain.
P255 to P292	ts-33-501:2957-3031	Outside the scope	Requirements for the Security Edge Protection Proxy; no roaming in the selected path.
P293 to P303	ts-33-501:3035-3055	Outside the scope	Messages generated by roaming intermediaries; no roaming.
P304 to P312	ts-33-501:3059-3075	Outside the scope	Protection of attributes across roaming interfaces.
P313 to P314	ts-33-501:3079-3081	Outside the scope	IPUPS functionality at inter-PLMN boundaries.
P315 to P318	ts-33-501:3085-3091	No goal stated	The originating-network header carries a PLMN identity; it states no property of this API and the modelled path is single-PLMN.
P319 to P323	ts-33-501:3097-3105	Outside the scope	Configuration of signalling monitoring.
P324 to P327	ts-33-501:3109-3115	Outside the scope	Streaming of signalling monitoring data.
P328 to P334	ts-33-501:3121-3133	Outside the scope	Security visibility, a deployment capability.
P335 to P337	ts-33-501:3137-3141	Outside the scope	Security configurability of the operator.
P338 to P347	ts-33-501:3149-3167	Outside the scope	Ciphering algorithm identifiers on the radio interface.
P348 to P357	ts-33-501:3171-3189	Outside the scope	Integrity algorithm identifiers on the radio interface.
P358 to P369	ts-33-501:3193-3215	The prover cannot be asked	Requirements on algorithm selection and strength, which a symbolic prover cannot judge.
P370 to P371	ts-33-501:3219-3221	Outside the scope	Requirements on the 5G-RG.
P372 to P375	ts-33-501:3225-3231	Outside the scope	Requirements on the NSSAAF.
P376 to P380	ts-33-501:6540-6548	Outside the scope	Security requirements and procedures on N3.
P381 to P388	ts-33-501:9656-10897	Outside the scope	EAP-TLS examples for subscription identifier privacy and IAB-node requirements.
P389 to P394	ts-33-501:10901-10917	Outside the scope	IAB donor and IAB F1 requirements.
P395 to P396	ts-33-501:11439-11443	Outside the scope	Requirements of MBSF and MBSTF.
P397 to P398	ts-33-501:11899-11901	Outside the scope	NEF security requirements for untrusted AF exposure, excluded by the task.

The parties and the protected values, one by one

The host listed the 18 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: 7. Without: 11. 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	AF	party	Everything lost by definition	The AF is the trusted VFL client the task fixes (s0004, s0013); an attacker in its place produces the results and the notifications itself, so result correctness, notification content and token validation are lost by definition, and the source asks no property of a dishonest AF.
S02	NF service consumer	party	`malicious_nf`	
S03	NRF	party	`malicious_nrf`	
S04	Subscription resource	value	`uri_reader`	
S05	Notification	value	Nothing gained	A notification is protected by TLS between AF and consumer; a copy of it gives only the replay, which is the attacker of R02.
S06	notifCorrId	value	Nothing gained	notifCorrId is an opaque identifier chosen by the consumer; it authenticates nothing by itself.
S07	notifUri	value	Nothing gained	The notifUri is the consumer's own callback address; knowing it does not let another party read traffic sent to it.
S08	vflInferAnaSubs	value	Nothing gained	A description of the analytics event subscribed to; knowing it gives no access to the subscription resource.
S09	vflInferResults	value	`malicious_nf`	
S10	subscriptionId	value	`uri_reader`	
S11	OAuth2 access token	value	`malicious_nf`	
S12	naf-vflinference scope	value	Nothing gained	The scope string naf-vflinference is public and defined by the specification (s0291).
S13	AF instance identity	value	Nothing gained	The AF NF instance identity is public; it is discovered via the NRF.
S14	Notification message	value	Nothing gained	The notification message is protected by TLS; a copy in the hands of the network is the attacker of R02.
S15	supis/gpsis/intGroupIds/exterGroupIds	value	`malicious_nf`	
S16	Input samples	value	Outside the scope	The input samples and the numerical algorithm are excluded by the task; the API carries only their results.
S17	Model version	value	Outside the scope	The model version is not a field of the messages in the selected units (VflInferResult is defined in TS 29.520).
S18	Error response	value	Nothing gained	ProblemDetails and the HTTP error codes carry no protected value; the source even allows 403 without a body.

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		

Attacker	Has	Basis	Where
`malicious_nf`	A legitimate NF with its own NF identity, its own TLS credentials and a valid OAuth2 access token with the naf-vflinference scope, from the start.	A NF service consumer may be a dishonest NWDAF or NEF; the task distinguishes a malicious legitimate NF holding its own credentials from an external network attacker, and TS 33.501 separates trust domains.	ts-33-501:2859, ts-33-501:2865
`uri_reader`	The subscriptionId and the other resource URI parts of another consumer's subscription, as an intermediate node sees them (Location header of s0044, request URI of s0082, redirection of s0293/s0295), before the notification is sent.	Requests carry {subscriptionId} in the resource URI and may be redirected via an SCP, which handles the URI; the source does not say that a subscriptionId is confidential to its owner.	ts-33-501:2911, s0044, s0082, s0293, s0295
`malicious_nrf`	The authorization server role: it can issue tokens for any subject, audience and scope, and it holds the token signing key.	The NRF is the authorization server and issues the tokens the AF validates (s0288-s0290); a compromised authorization server invalidates every token-based check.	ts-33-501:2885, s0288, s0289, s0290

ID	`network`	`malicious_nf`	`uri_reader`	`malicious_nrf`
A02	Holds			
A03	Holds			
A08	Holds		Fails	
Z01	Holds			
Z03	Holds			
Z04	Holds			
S05	Holds			
S04	Holds			
R02	Fails			
R03	Fails			
K02	Holds			
K01	Holds			
U01	Holds			
U02	Holds			
Q3	Holds			

Source and targets

Source: 336 units, 155963 bytes, frozen in source-delivery-01.json. Targets: 14 approved (A02, A03, A08, Z01, Z03, Z04, S05, S04, R02, R03, K02, K01, U01, U02); 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-1ebf059b4d1aa0ece99bbb214373

This step as it reports itself: 1 requests (1 failed), 0.000 USD, 82562 tokens in and 5293 out, 260 s.

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
- R02: answered in ProVerif, and the analyst names changing_state for this target
- R03: answered in ProVerif, and the analyst names changing_state for this target
- Q3: 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 subscription resource (subscriptionId, notifUri, notifCorrId, vflInferAnaSubs)	AF (VFL client), as the Individual VFL Inference Subscription resource	Created by POST on the subscriptions resource (s0043, s0044); modified by PUT (s0190) and PATCH; removed by DELETE (s0083, s0084).	On every PUT/PATCH/DELETE the AF looks the subscriptionId up before it acts, and on every notification it reads the subscription it notifies (s0095).	s0043, s0044, s0083, s0084, s0085, s0190, s0218
the notification state of a subscription (notifCorrId, the results already consumed)	NF service consumer (NWDAF or NEF)	Whenever the AF notifies an event of that subscription (s0095); the consumer stores the notification and answers 204 (s0098).	On every incoming notification POST, against the notifCorrId and notifUri registered locally (s0096).	s0095, s0096, s0098, s0100
the subscriptionId	AF and NF service consumer	Assigned once, by the AF, when the subscription is created (s0044).	Compared in the resource URI of PUT, PATCH, DELETE and of the redirect Location (s0082, s0193, s0219), and by the consumer when it uses the Location it was given.	s0044, s0082, s0193, s0219

pattern	result	target	shown by ProVerif	steps
replay	candidate	R02	yes: ProVerif falsifies `b_afterdel_ctl` on bounded.pv	The AF sends the notification POST for a subscription to {notifUri} with notifCorrId and vflInferResults (s0095-s0097). / The attacker keeps a copy of that POST instead of dropping it. / The consumer accepts it and answers 204, storing the notification (s0098). / The attacker delivers the same POST again; the consumer accepts and stores it again, because neither the message nor any consumer state carries a sequence number or a duplicate marker.

pattern	result	target	shown by ProVerif	steps
repeat_one_time_action	candidate	R03	yes: ProVerif falsifies `b_del_once_ctl` on bounded.pv	The consumer sends DELETE on {apiRoot}/naf-vflinference/<apiVersion>/subscriptions/{subscriptionId}; the AF removes the subscription and answers 204 (s0082-s0085). / The attacker re-sends the same DELETE. / The AF accepts it again and answers 204 a second time: nothing in the source records that the resource has already been removed, and no 404 is required for a subscriptionId that was deleted.
repeat_one_time_action	candidate	R03	no: ProVerif says verified for `b_del_once` on bounded.pv	If the AF keeps the subscription as state until it is removed, the repeated DELETE finds no subscription and is refused; b_del_once shows this for one unrolled subscription.
desynchronisation	candidate		no: not tried with ProVerif	A second NF with a valid token and the naf-vflinference scope learns the victim's subscriptionId (s0044, s0082) and sends PUT with its own notifUri and its own vflInferAnaSubs (s0190). / The AF accepts the PUT and modifies the subscription; the victim consumer keeps its own notifCorrId and notifUri and believes the subscription unchanged. / The AF then notifies the modified subscription to the attacker's notifUri, so the victim's inference results reach the attacker while the victim notices nothing; if the attacker deletes the subscription instead, the victim waits for notifications that never come.
no_rollback_on_failure	candidate		no: not tried with ProVerif	The consumer sends DELETE; the AF removes the subscription and sends 204 (s0083-s0085). / The attacker drops the 204, so the consumer keeps the subscription and its notifCorrId and believes the unsubscribe did not happen. / The consumer retries or waits; either way the two sides keep different states, and the source defines no way for the consumer to learn that the resource is already gone.
update_before_check	nothing found			The token is checked before the subscription is looked up in both AF roles, so no acceptance follows a change of state, and no query is needed to see the order.
race_between_sessions	nothing found			Every AF role looks the subscription up before it acts; a request for an unknown subscriptionId does not change what is stored.
distinguishable_failure	nothing found			The source lets the AF answer 403 without a body (s0155) and refers failures to 29.500 (s0275, s0276); the model gives no failure answer, so this pattern is outside what it can show.
wrap_or_window	nothing found			The API has no counter, sequence number or acceptance window; its identifiers are opaque strings.

What the analyst named for targets:

step	targets	features	reason
22	R02, R03, Z04, S05, Q3	changing_state	These targets rest on values that change across steps: the AF's subscription resource (created by POST, modified by PUT/PATCH, removed by DELETE) and the consumer's knowledge of the notification it has already consumed. In model.pv the store is a monotone table and the removal is not modelled at all, so the ordering questions ('after the delete no notification', 'a repeated request changes the resource once') are only answered on bounded.pv, where the AF and the consumer are unrolled to a fixed number of steps.

The analyst's closing note

All fifteen targets have answers. Proved on model.pv: request and notification authenticity, the notification-to-subscription binding, the notifCorrId/notifUri pairing, the token requirement for subscribe, update and delete together with its subject and audience binding, and the confidentiality of the access token and of the user identifiers in the results, both with a control that fails. Proved on bounded.pv, one subscription unrolled to three steps: a notification is sent only for a subscription the AF holds and carries the modified content after a PUT, the unsubscribe is applied once, and after the 204 the consumer accepts no more notifications for that subscriptionId; the last two have controls that fail. Two findings. First and most important: the API has one service-wide OAuth2 scope and specifies no check that the requester of PUT, PATCH or DELETE owns the subscriptionId, so any NF with a valid token that knows one subscriptionId replaces or deletes another consumer's subscription, and a replaced notifUri diverts that consumer's vflInferResults; the remedy was checked with check_fix — binding the stored subscription to the token subject makes both attack queries verify, keeps the normal run and leaves all sixteen other recorded answers unchanged. Second: no duplicate detection anywhere, so a replayed notification is stored twice, a replayed subscribe creates a second subscription and a replayed DELETE is accepted again; that recommendation is not checked, because a counter that changes per notification can only be modelled in the three-step unrolled chain and the counter itself would be invented by the recommendation. Everything rests on TLS endpoint authentication, on a symmetric OAuth2 token, on a frozen notion of time that leaves the expiry check untested, and on the AF being the trusted party the task fixes; PATCH, the error answers and the result structure of TS 29.520 are outside the model. A control in the report's style could not be built for the correspondence proofs inside one file, and the report names which proofs have none.