# Security analysis of AF VFL client training 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** | 12: 7 hold, 5 fail (12 attacks) |
| **Findings** | 1 |

**Contents**


1 Protocol and scope

2 Findings

3 Properties examined

4 Limits of this analysis

Appendix A: Reading of each property

Appendix C: Claims and prover answers

Appendix D: How the analysis was run


## 1 Protocol and scope

The Naf_VFLTraining service is the interface a trusted Application Function exposes as a VFL client, so that the NWDAF, acting as the NF service consumer, can subscribe to VFL training reports. The AF holds the subscriptions and issues the training notifications; the NWDAF creates, updates and deletes its subscription and receives the notifications at a URI it names. Both are NFs of the same operator and obtain an access token for the single service scope naf-vfl-train from the NRF.

The exchange is a subscription creation that names a notification URI and a correlation identifier, an update that activates reporting, one notification from the AF, its acknowledgement, and finally the deletion of the subscription. When the exchange ends the AF is sure that the requests it accepted came from the NF whose credentials and token it verified, and the NWDAF is sure that the 201 and the notification came from the AF for the subscription it created. The AF is not made sure, by anything in this API, that the NF operating on a subscription is the one that created it.

The parties:

- **NWDAF**: NF service consumer in the trusted case. Holds: NF instance identity, TLS credentials, OAuth2 client credentials.
- **AF**: trusted VFL client, service producer. Holds: the subscriptions it holds, NRF key for token verification.
- **NRF**: authorization server. Holds: NRF signing key.

Figure 1 shows the exchange as the documents specify it.

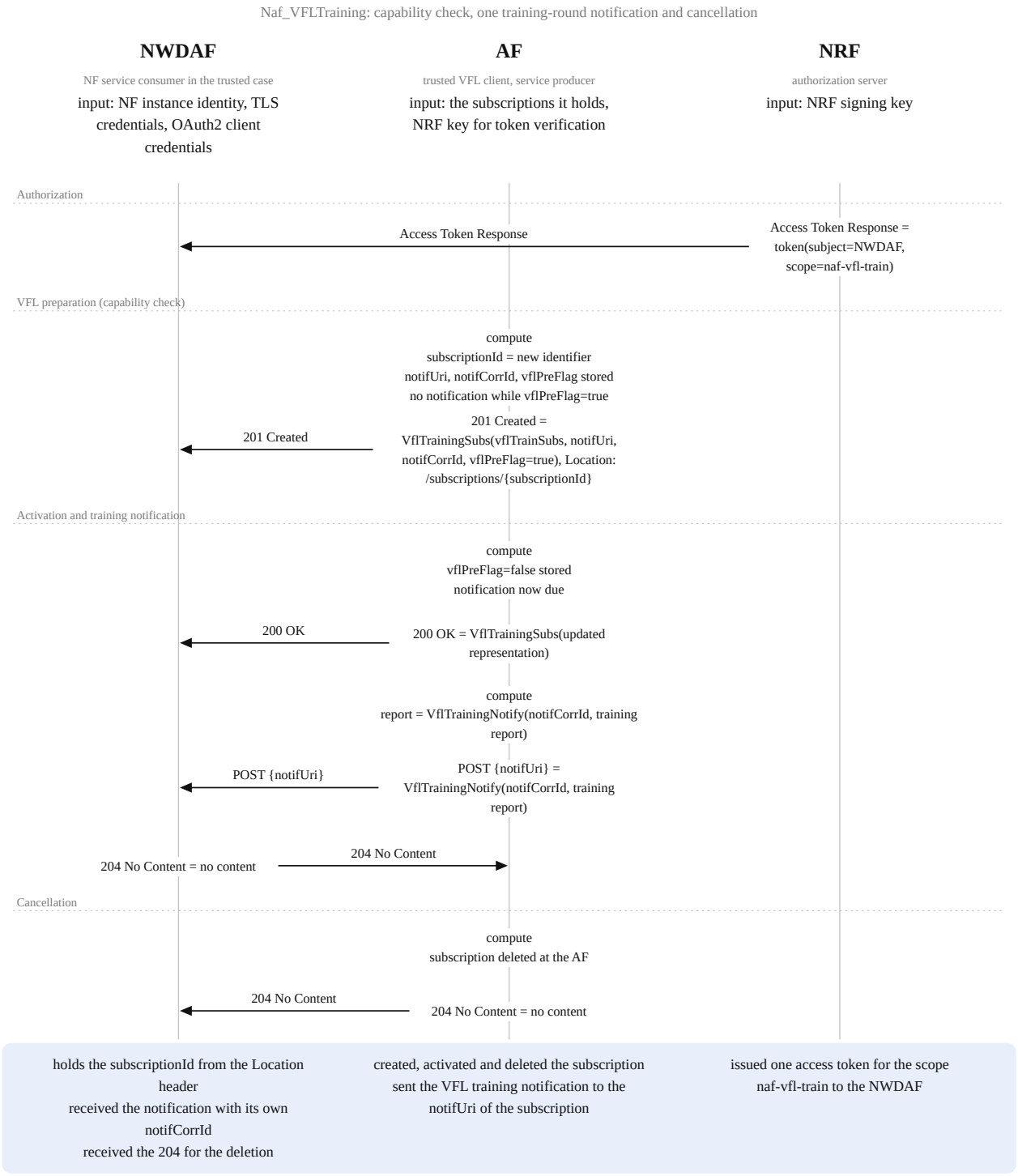



Figure 1. The protocol as specified.

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

## 2 Findings

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

The API gives the guarantees it is designed for only against an attacker that has the network alone: authentication, parameter agreement, token validity and the preparation flag all hold there. Against another authorized NF it does not: one legitimate NF can take over another consumer's subscription, receive its VFL training notifications and delete it. This is the one finding; it is new as a statement, although the documents state the two ingredients, the single service scope and the absence of any token-to-client binding at a resource server. The results rest on a model that simplifies TLS as per-endpoint encryption without session state, fixes two NF identities and treats the subscriptionId as knowable; the deletion property is answered on a four-step unrolled model. Tamarin+ Agent, which models TLS records and state as they are, would settle the replay question that the ProVerif model leaves as an artifact and would confirm the deletion behaviour beyond four steps.

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 One authorized NF can take over another consumer's VFL training 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, Z03, Z04, U02, U03 (section 3) |
| **Source text** | s0045, s0061, s0074, s0144, s0276 |
| **Shown by** | ProVerif attack queries `z03_update`, `z03_delete`, `z03_notif`, `a08_origin`, `a08_owner`; Figures 2, 3, 4, 5 and 6 |

**What happens.** A legitimate NF obtains its own access token for the single service scope naf-vfl-train from the NRF. It addresses another consumer's individual subscription by its resource URI and sends PUT or PATCH. The AF verifies the token, the scope and the subscriptionId, but never compares the token subject with the NF that created the subscription, so it accepts the update and replaces the stored notifUri, notifCorrId and preparation flag. The next VFL training notification of that subscription is delivered to the attacker's URI and decrypted with the attacker's own TLS key; the same missing check also lets the attacker delete the subscription and suppress its reporting.

**What the attacker gains.** The attacker receives the VFL training notification of that consumer's subscription in place of the consumer, can suppress reporting by setting the preparation flag back, and can stop it altogether by deleting the subscription. Neither party notices: the consumer only stops receiving notifications, and the AF has accepted a request that its token and scope authorized.

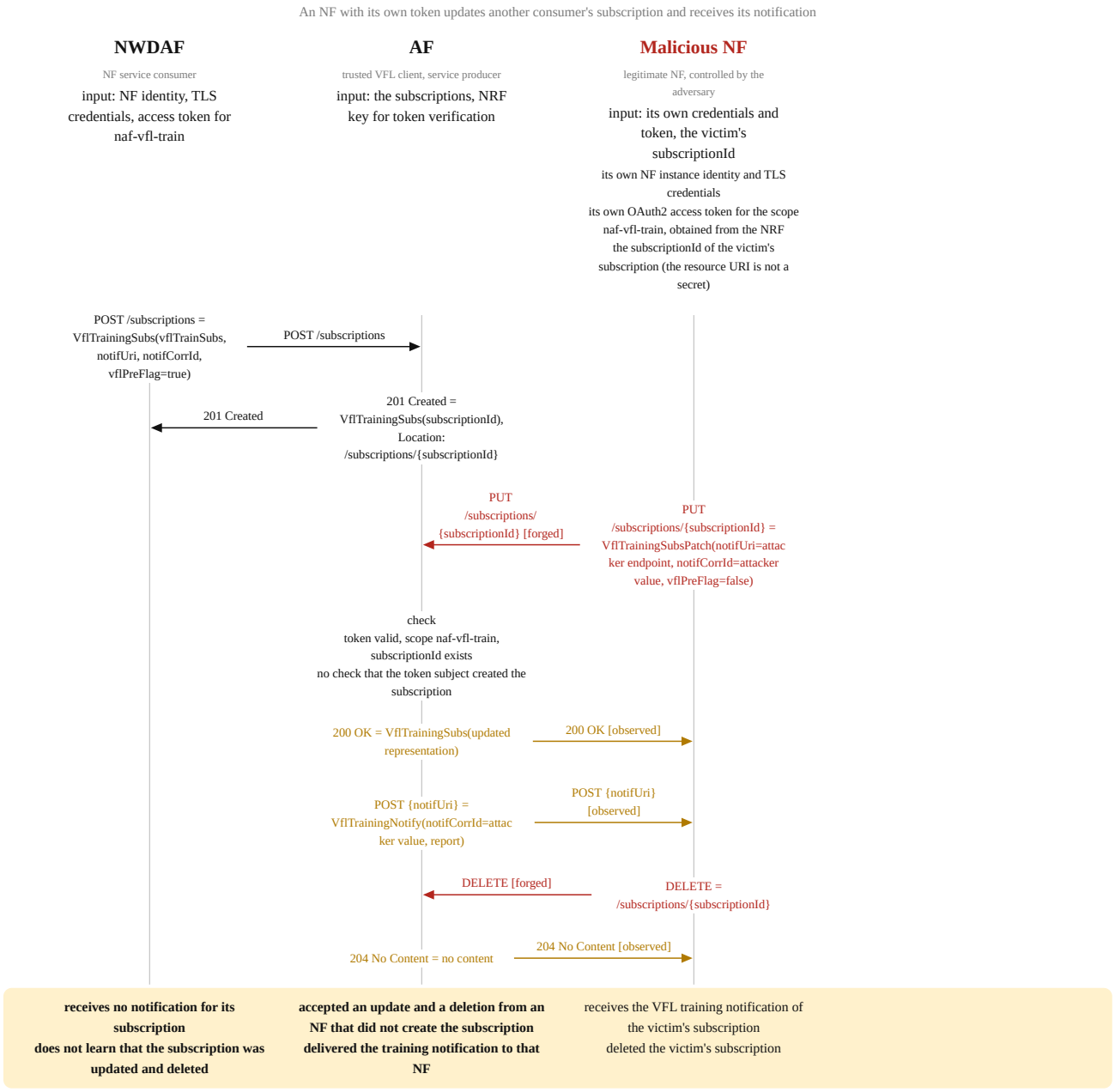



Figure 2. An NF with its own token updates another consumer's subscription and receives its notification. Property U03; attack queries z03_delete, z03_notif, z03_update.

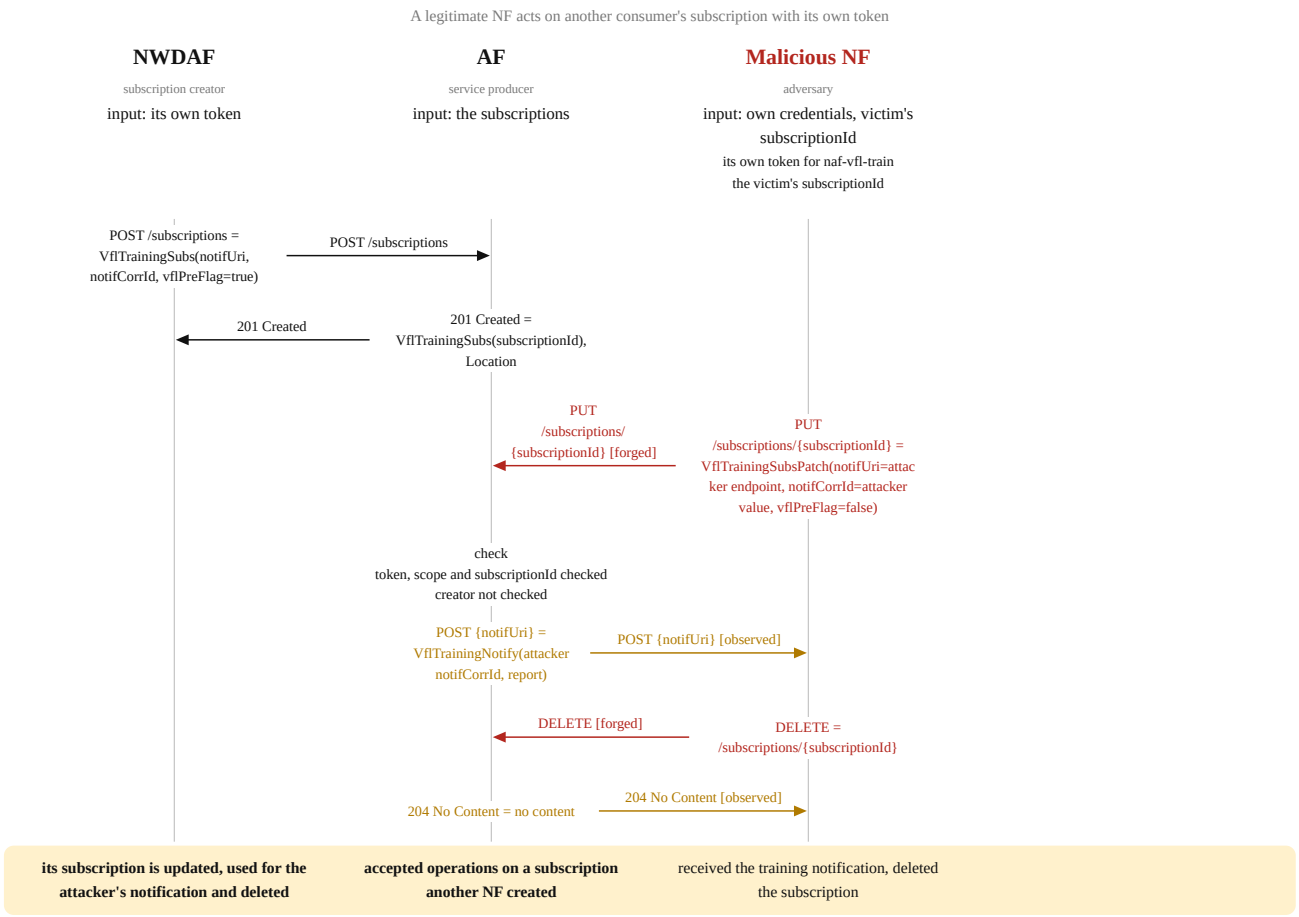



Figure 3. A legitimate NF acts on another consumer's subscription with its own token. Property Z03; attack queries z03_delete, z03_notif, z03_update.

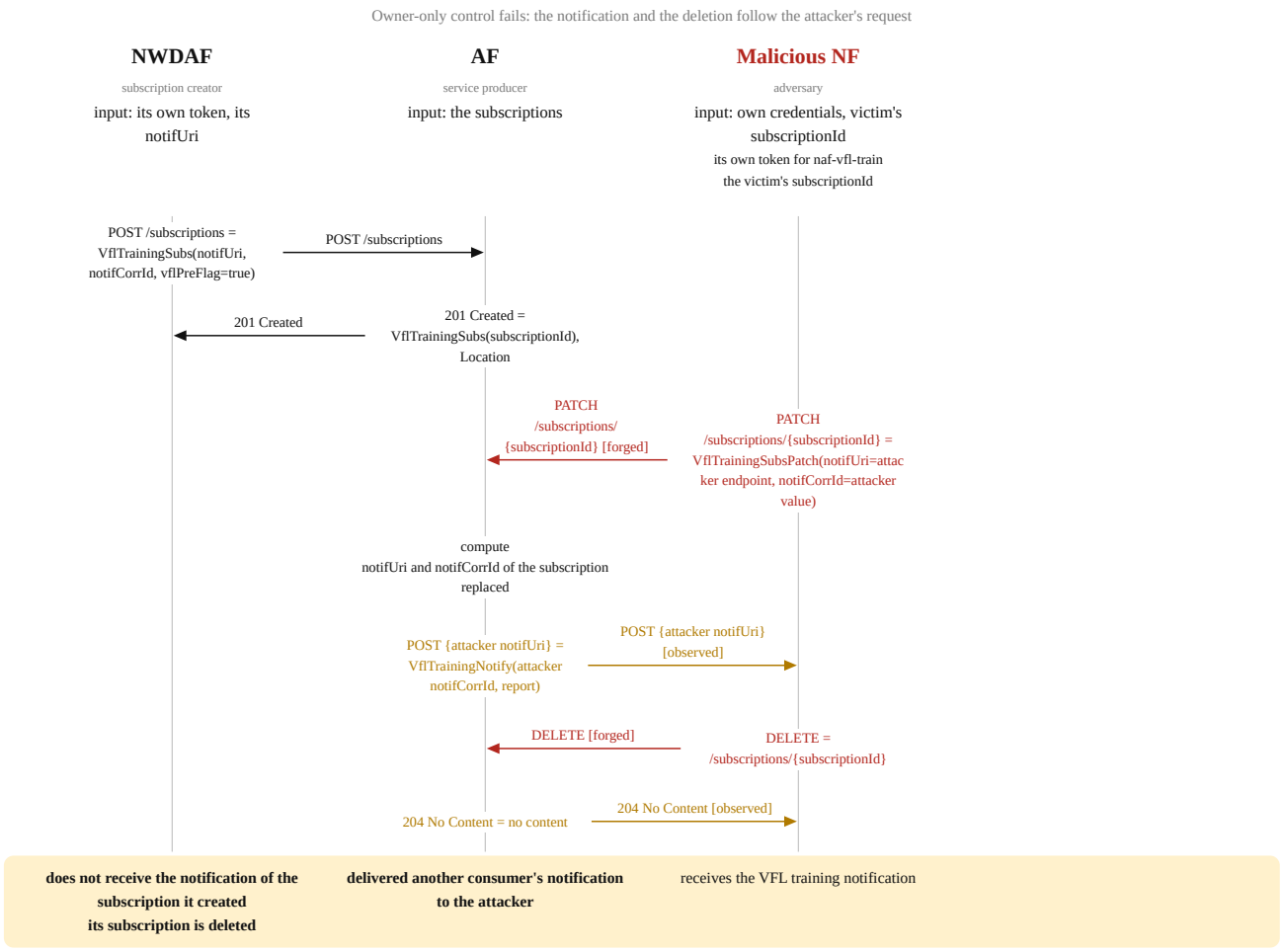



Figure 4. Owner-only control fails: the notification and the deletion follow the attacker's request. Property U02; attack queries z03_delete, z03_notif, z03_update.

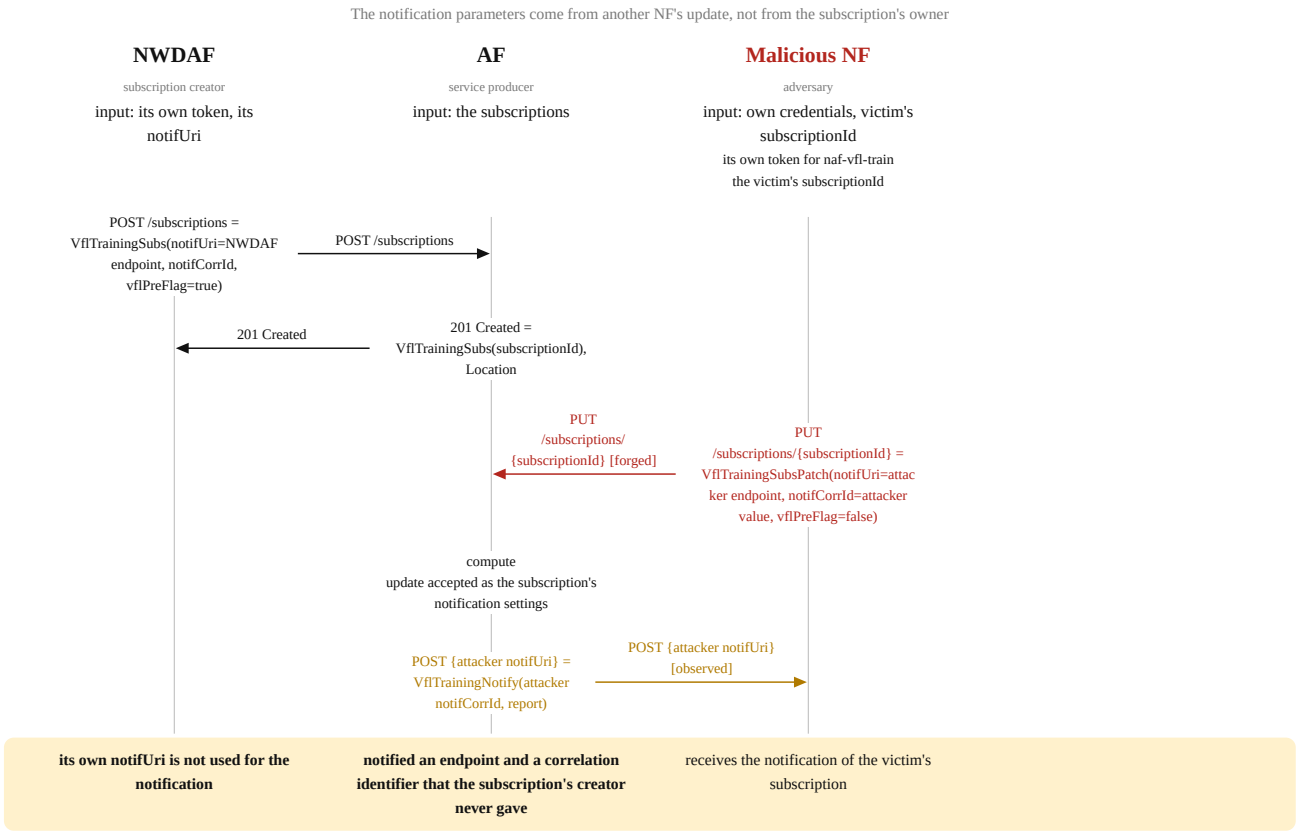



Figure 5. The notification parameters come from another NF's update, not from the subscription's owner. Property A08; attack queries a08_origin, a08_owner.

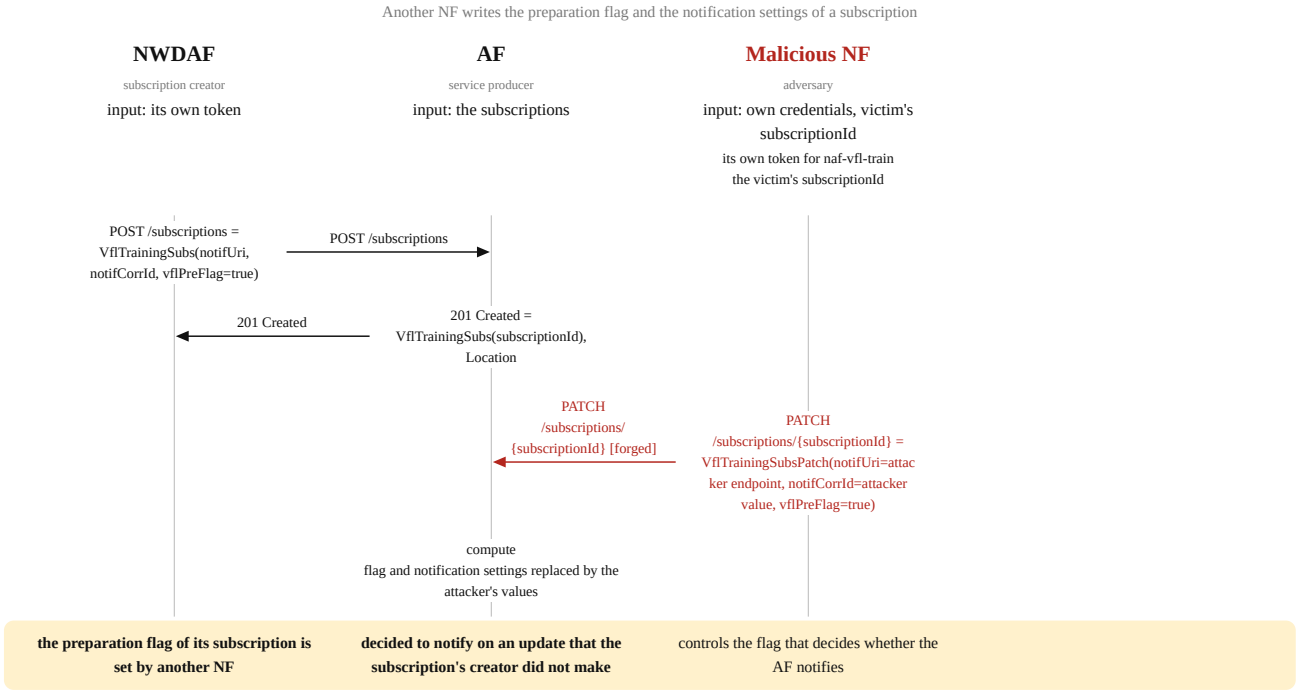



Figure 6. Another NF writes the preparation flag and the notification settings of a subscription. Property Z04; attack query a08_origin.

**Proposed change.** Have the AF record the NF instance identity that created each subscription and reject PUT, PATCH and DELETE when the subject of the access token is not that identity, and accept a notifUri only if it is an endpoint of that same identity. Statically, define the authorization at resource level instead of one scope for the whole

service, so that the token the NRF issues names the subscription the consumer may act on.

**Check of the change with ProVerif.** Not checked: A model of the change would need the AF to keep the creator identity of each subscription and compare it with the token subject, and the API to carry that identity in its token; that is a change of the protocol's checks and of the authorization, which I did not build as a variant within the session.

No figure was given for these attacks of a finding: A08: z03_delete, A08: z03_notif, A08: z03_update, U02: a08_origin, U02: a08_owner, U03: a08_origin, U03: a08_owner, Z03: a08_origin, Z03: a08_owner, Z04: a08_owner, Z04: z03_delete, Z04: z03_notif, Z04: z03_update.

## 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 | **VFL request peer authentication.** Whenever the AF accepts a subscription/update/deletion request, or the consumer accepts a notification, there exists a real peer run addressed to this party's identity. | no compromise | Holds | medium | yes |
| A03 | **Agreement on subscription parameters.** Every time the AF accepts a subscription creation/update, or the consumer accepts a notification, there exists a matching peer run, and both sides agree on the values of the subscription identifier, notifUri, notifCorrId, vflTrainSubs and vflPreFlag. | no compromise | Holds | low | yes |
| A04 | **Freshness of requests and notifications.** Every time the AF successfully creates a subscription or the consumer successfully accepts a notification, this corresponds to a previously unused peer run; the same request cannot be accepted twice as a new subscription or a new notification. | no compromise | Holds | low | yes |
| A08 | **Notification bound to approved subscription.** The notification delivery parameters the AF accepts (notifUri, notifCorrId, vflTrainSubs, vflPreFlag) must come from a subscription this consumer was previously approved for, and no notification may be sent while the preparation flag is true. | no compromise | Fails (2 attacks) | high | unclear |
| Z01 | **Token and scope for each operation.** Every subscription creation/update/deletion or notification delivery that is accepted must have a valid token from the authorization server (NRF), and the token's scope must cover the naf-vfl-train service. | no compromise | Holds | medium | yes |
| Z03 | **Per-subscription authorization.** The authorization used for a subscription update/deletion or notification delivery must be bound to this consumer's own subscription resource and the corresponding operation, and cannot be used for another consumer's subscription. | no compromise | Fails (3 attacks) | high | unclear |
| Z04 | **No notification after deletion.** After a subscription has been deleted (or before vflPreFlag is updated from true to false), the AF must not deliver any further notification to the consumer based on that old subscription. | no compromise | Fails (1 attack) | medium | unclear |

| ID | Property | Compromise assumed | Result | Value | In the documents |
|---|---|---|---|---|---|
| S05 | **Deleted subscription raises no notification.** After the subscription deletion takes effect, the old notifUri, notifCorrId or subscriptionId of that subscription must not trigger a new notification delivery. | no compromise | Holds within a bound (the changing value unrolled to 4 steps) | low | unclear |
| U01 | **AF identity and correlation bound.** When the AF accepts a subscription or the consumer accepts a notification, the AF identity, subscriptionId, notifCorrId and training round identifier must agree with the matching peer run. | no compromise | Holds | low | yes |
| U02 | **Owner-only subscription control.** An update/deletion of a given subscriptionId may only be initiated by the creator of that subscription; a notification may only be delivered to the notifUri provided by that subscriber, with a matching notifCorrId. | no compromise | Fails (3 attacks) | high | unclear |
| U03 | **Malicious NF confined to own subscription.** A malicious NF holding its own legitimate credentials cannot make the AF accept a substitution of subscription content across training rounds, nor perform updates or deletions on another consumer's subscription. | no compromise | Fails (3 attacks) | high | unclear |
| X01 | **OAuth2 token confidentiality in transit.** A network attacker never learns the access token an NF consumer obtained from the NRF for the naf-vfl-train scope. | no compromise | Holds | low | yes |

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

The questions asked, and the analyst's answers:

| Question | Answer | Rests on |
|---|---|---|
| Q1: AF/client identity bound to VFL correlation/round | The AF and the consumer do bind the subscriptionId and the notifCorrId: the consumer accepts a 201 or a notification only from the AF and only for the subscription and correlation identifier it created, and ProVerif proves this against the network attacker. The training round has no field in this API: the round number travels in the VflTrainingNotify data of TS 29.520, so the model binds a round only through the notification content of the subscription; substitution by a party that holds credentials is not prevented, which is the attack reported for U03. | U01, A03, U03 |
| Q2: Binding between the training subscription controller and the notification receiver | No. The AF verifies the token and the subscriptionId, but the API defines no per-resource authorization and the AF never compares the token subject with the NF that created the subscription, so a second legitimate NF can update or delete the subscription and can make the AF deliver the notification to its own notifUri instead of the one the subscriber gave. | U02, Z03, Z04 |
| Q3: Cross-round substitution and cross-subscription operations by a malicious legitimate NF | Yes, it can do both. With its own valid token for the single service scope it replaces the notifUri, notifCorrId and preparation flag of another consumer's subscription, receives that subscriber's VFL training notifications, and can delete the subscription; substituting the content of a round is the same operation on the same record, so the round of the next notification follows the attacker's update. | U03, U02, Z03, 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:

- TLS is modelled as a MAC under the sender's long-term key plus encryption under the receiver's long-term key; the handshake, session keys and record sequence numbers are not modelled, so replay of a recorded record is left to the abstraction.
- The model has two NF identities (one honest consumer, one malicious NF) and one AF identity; what holds for an arbitrary population of NFs is argued from these two, not proved.
- The subscriptionId is modelled as a function of the consumer identity, so it is treated as knowable by the attacker; a deployment whose subscriptionId is an unguessable identifier is not covered, and neither are concurrent sessions of one consumer.
- OAuth2 is optional in the source (access to the API may be authorized); the model always has the AF verify a token, so the answers are for the configurations where OAuth2 is in use.
- The AF's record is modelled for one subscription at a time: model.pv does not check its state, and bounded.pv unrolls it to four steps (creation, update and notification, deletion, one repeated update).
- Error handling (403 OVERLOAD, NOT_AVAILABLE_FOR_VFL_PROCESS, VFL_REQS_NOT_MET), HTTP redirection via the SCP, the NEF and untrusted-AF branch, and the content of VflTrainingSub and VflTrainingNotify (defined in TS 29.520) are outside the analysed flow.
- The strength of the cryptography, the secrecy of the NRF's signing key and availability are not analysed.

What the host records:

- The answers of Z04, S05 rest on an operation the model simplifies (Appendix D).
- Proved without a control query (Appendix C): Z01: z01_create, Z01: z01_update, Z01: z01_delete, Z03: z03_imp, U03: z03_imp. 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: VFL request peer authentication

Whenever the AF accepts a subscription/update/deletion request, or the consumer accepts a notification, there exists a real peer run addressed to this party's identity.

The AF accepts a subscription creation, update or deletion only when the request really came from the NF whose identity it names, and the consumer accepts a 201 or a notification only when the AF really sent it. The guarantee is the one TLS and OAuth2 client credentials are meant to give, and it holds for any number of sessions.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: It is the central authentication guarantee of the flow, proved mechanically, but no simplification touches it and the documents require it, so it is a confirmation rather than a discovery.
- Stated in the documents: yes. Where the documents say it: s0038, s0045, s0061, s0074, s0098, s0273
- Rests on: TLS is modelled as a MAC under the sender's long-term key plus encryption under the receiver's long-term key, without handshake or session state.; Only two NF identities are modelled; the claim is asked for the honest one, and the malicious NF is by construction a real peer of its own runs.; The token format and the NRF's signing are modelled as a MAC under a key the NRF shares with the AF.

## A03: Agreement on subscription parameters

Every time the AF accepts a subscription creation/update, or the consumer accepts a notification, there exists a matching peer run, and both sides agree on the values of the subscription identifier, notifUri, notifCorrId, vflTrainSubs and vflPreFlag.

The two sides agree on the subscriptionId, the notifUri, the notifCorrId and the preparation flag: what the consumer reads in the 201 and in the update response is what the AF accepted for its subscription, and the notification it accepts carries the correlation identifier of its own subscription.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: The API carries the same values in request and response, so the agreement follows from the message format; the result confirms that the model matches the source.
- Stated in the documents: yes. Where the documents say it: s0039, s0048, s0074, s0241
- Rests on: The update response is modelled as carrying the correlation identifier, as the 200 OK representation of the updated resource would.; TLS and token modelling as for A02.

## A04: Freshness of requests and notifications

Every time the AF successfully creates a subscription or the consumer successfully accepts a notification, this corresponds to a previously unused peer run; the same request cannot be accepted twice as a new subscription or a new notification.

A notification cannot be made to be accepted twice by an attacker on the network, and no acceptance of a request is found twice for the same consumer run except by replaying the recorded record, which the model accepts only because it treats TLS encryption as deterministic. The API therefore owes its freshness entirely to TLS: nothing in the subscription request or the notification carries a nonce, a sequence number or a timestamp.

- Result: Holds; answers what is asked: partly; usable with caveats
- Value low: The only failure ProVerif found is a model artifact, and what remains is that the API adds nothing to TLS for freshness; the documents already require replay protection at the SBI layer.
- Stated in the documents: yes. Where the documents say it: s0038, s0074, s0098
- Rests on: The replay of a creation is an artifact of modelling TLS as deterministic encryption; a TLS 1.3 record belongs to a handshake the attacker cannot repeat without the consumer's key.; No sequence number or round number is modelled, so freshness of a notification across rounds cannot be expressed.
- Open: Whether a recorded TLS record can be replayed in a deployment is settled by TLS, not by the API; the API itself defines no request nonce or sequence number.

### A08: Notification bound to approved subscription

The notification delivery parameters the AF accepts (notifUri, notifCorrId, vflTrainSubs, vflPreFlag) must come from a subscription the consumer was previously granted, and no notification may be sent while the preparation flag is true.

The AF notifies only after an accepted update that sets the preparation flag to false, and it always uses the notifUri and notifCorrId of that update. Against an attacker with the network alone that means the consumer's own values; against another legitimate NF it means the attacker's values, because that NF can make the update the AF accepts.

- Result: Fails (2 attacks); answers what is asked: partly; usable with caveats
- Value high: The attack is a real loss of another consumer's training reports, obtained with a credential every authorized NF can have; the documents do not state it.
- Stated in the documents: unclear. Source text it rests on: s0074, s0213, s0241, s0276
- Rests on: The subscriptionId is assumed knowable by the attacker, as a resource URI that is not a secret.; The AF checks the token and the service scope but no per-resource authorization, because the source defines none.
- Open: Whether the AF must accept a notifUri that belongs to another NF; the source does not say.

### Z01: Token and scope for each operation

Every subscription creation/update/deletion or notification delivery that is accepted must have a valid token from the authorization server (NRF), and the token's scope must cover the naf-vfl-train service.

Every creation, update and deletion the AF accepts rests on a token the NRF issued for the naf-vfl-train scope to that same NF identity, against an attacker with the network alone and against a legitimate NF holding its own token. The token cannot be presented for another NF's identity.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: It proves the authorization step works as intended, which is worth having, but it rests on the assumption that OAuth2 is deployed, which the source leaves to local configuration.
- Stated in the documents: yes. Where the documents say it: s0273, s0274, s0276
- Rests on: The AF is modelled as always verifying the token, while the source makes OAuth2 optional.; The NRF's signing key is modelled as a MAC key shared with the AF, and the token is modelled without lifetime or audience.

- Open: The source says access may be authorized with OAuth2, so the guarantee holds for the configurations that use it.

### Z03: Per-subscription authorization

The authorization used for subscription update/deletion or notification delivery must be bound to the consumer's own subscription resource and the corresponding operation, and must not be usable for another consumer's subscription.

No NF can act under another NF's identity: the AF accepts an operation only when the token it verifies was issued to the identity that authenticated at the transport layer. What the AF does not check is whether that identity owns the subscription it addresses, so any NF with a token for the service scope can update or delete another consumer's subscription and receive its notifications.

- Result: Fails (3 attacks); answers what is asked: partly; usable with caveats
- Value high: The attack is a loss of another consumer's subscription and training reports, reachable by every authorized NF of the same operator, and the API as specified has no check that prevents it.
- Stated in the documents: unclear. Source text it rests on: s0144, s0276, s0045, s0061
- Rests on: The subscriptionId is assumed knowable by the attacker; the model has two NF identities.; The AF's lookup and authorization are modelled as the source describes them: token, scope and the resource URI, with no owner comparison.
- Open: Whether a deployment is expected to add a resource-level check that the API does not define.

### Z04: No notification after deletion

After a subscription has been deleted (or before vflPreFlag is updated from true to false), the AF must not deliver any further notifications to the consumer on the basis of that old subscription.

The AF sends no notification while the preparation flag is still true, and after the deletion it refuses a repeated update of the same subscription instead of notifying again. Who may write the flag and the notification parameters is decided only by the token and the service scope, so another legitimate NF can suppress or redirect the notifications of a subscription.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: The flag behaviour is as the note says, but the possibility for another NF to write the flag is a consequence of the same missing owner check as the main finding.
- Stated in the documents: unclear. Source text it rests on: s0241, s0061, s0074
- Rests on: The deletion property is answered on the bounded model, with the AF unrolled to four steps and a single identity.; The preparation-flag property is answered on the full model against the network attacker; the malicious-NF answer is the attack recorded for A08.
- Open: Whether the AF must compare the token subject with the subscription's creator; the source does not say, and the API has no resource-level scope that would express it.

### S05: Deleted subscription raises no notification

After a subscription deletion takes effect, the old notifUri, notifCorrId or subscriptionId of that subscription must not be able to trigger a new notification delivery.

In the unrolled model a repeated update that arrives after the deletion is refused and produces no notification, so the old subscriptionId, notifUri and notifCorrId cannot keep the reporting alive by themselves. The answer covers four steps of the subscription's life and one repeated message.

- Result: Holds within a bound (the changing value unrolled to 4 steps); answers what is asked: partly; usable with caveats
- Value low: It confirms what the deletion is for, within a bound, and it shows the property depends on the AF dropping the record; the source says nothing that contradicts it.
- Stated in the documents: unclear. Source text it rests on: s0054, s0061, s0062
- Rests on: The answer is bounded: creation, update and notification, deletion, and one repeated update.; The bounded model carries one identity and one TLS key, so it says nothing about another NF's own fresh update after the deletion.; Deletion is modelled as the AF no longer holding the record; if the record were kept, the property would fail, which the control model bounded-buggy.pv shows.
- Open: How long after the deletion a repeated request could still be honoured in a real implementation; the model fixes the deletion at one point of the unrolled run.

### U01: AF identity and correlation bound

When the AF accepts a subscription or the consumer accepts a notification, the AF identity, subscriptionId, notifCorrId and training-round identifier must match those of the corresponding peer run.

The consumer accepts a 201 or a notification only from the AF, only for the subscriptionId the AF returned to it, and only with the notifCorrId of its own subscription. A round number is not part of this API; the round travels inside the notification data of TS 29.520, and the model binds it only through the subscription's content.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: The binding of subscriptionId and notifCorrId follows from the message format; the round part of the question cannot be asked of this API.
- Stated in the documents: yes. Where the documents say it: s0039, s0074, s0144, s0235
- Rests on: The training-round identifier is not modelled because the selected units do not define it.; TLS and token modelling as for A02.
- Open: The training round has no field in this API, so round binding is only through the notification content of the subscription.

### U02: Owner-only subscription control

An update/deletion of a subscriptionId may only be initiated by the creator of that subscription; a notification may only be delivered to the notifUri provided by that subscriber and only with a matching notifCorrId.

Against an attacker with the network alone, notifications go only to the notifUri and notifCorrId that the consumer provided. Against another legitimate NF they do not: that NF can update or delete the subscription and make the AF deliver the notification to its own URI.

- Result: Fails (3 attacks); answers what is asked: partly; usable with caveats
- Value high: It is the same weakness as the main finding, stated as the subscription owner's question, and the documents do not state it.
- Stated in the documents: unclear. Source text it rests on: s0045, s0061, s0074, s0144
- Rests on: The subscriptionId is assumed knowable by the attacker; the model has two NF identities.; The AF checks only the token and the service scope for an operation on a subscription.
- Open: Whether the AF is expected to record the creator of a subscription and compare it with the token subject.

### U03: Malicious NF confined to own subscription

A malicious NF holding its own legitimate credentials cannot make the AF accept a replacement of subscription content across training rounds, nor perform an update or delete on another consumer's subscription.

A legitimate NF cannot get anything accepted under another NF's identity, but it can act on another consumer's subscription with its own valid token: it replaces the notifUri, notifCorrId and preparation flag, receives that subscription's training notifications, and can delete the subscription.

- Result: Fails (3 attacks); answers what is asked: partly; usable with caveats
- Value high: It answers the malicious-legitimate-NF question with a concrete attack that the documents do not state.
- Stated in the documents: unclear. Source text it rests on: s0045, s0061, s0144, s0276
- Rests on: The subscriptionId is assumed knowable by the attacker; the model has two NF identities.; Only the subscription's control values are modelled; the training-round content of TS 29.520 is not, so a round is affected only through the subscription's fields.
- Open: Whether a deployment adds a resource-level authorization that the API does not define.

### X01: OAuth2 token confidentiality in transit

A network attacker never learns the access token an NF consumer obtained from the NRF for the naf-vfl-train scope.

The access token the NWDAF obtains from the NRF never becomes known to an attacker that has only the network: it travels only inside the channel to the consumer and is bound to that consumer's identity. The control shows that the same query does detect a token that is published, as the malicious NF publishes its own.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: It confirms what TLS gives and the documents already require.
- Stated in the documents: yes. Where the documents say it: s0273, s0274, s0098
- Rests on: The token travels inside the modelled TLS channel; a leak from storage at the consumer is not modelled.

## 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_create | checked | | 4 proved |
| A03 | proved | | a03_created | checked | | 3 proved |
| A04 | proved | | a04_notif | checked | a04_create (model_artifact) | 1 proved, 1 attack |
| A08 | proved | | a08_net | checked | a08_origin (holds_in_source); a08_owner (holds_in_source) | 1 proved, 2 attack |
| Z01 | proved | | z01_create | | | 6 proved |
| Z03 | proved | | z03_imp | | z03_update (holds_in_source); z03_delete (holds_in_source); z03_notif (holds_in_source) | 1 proved, 3 attack |
| Z04 | proved | | a08_net | checked | a08_origin (holds_in_source) | 1 proved, 1 bounded, 1 attack |
| S05 | bounded | the changing value unrolled to 4 steps | s05 | checked | | 2 bounded |
| U01 | proved | | u01_create | checked | | 2 proved |
| U02 | proved | | z03_notif_net | checked | z03_update (holds_in_source); z03_delete (holds_in_source); z03_notif (holds_in_source) | 1 proved, 3 attack |
| U03 | proved | | z03_imp | | z03_update (holds_in_source); z03_delete (holds_in_source); z03_notif (holds_in_source) | 1 proved, 3 attack |
| X01 | proved | | x01_token | checked | | 1 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 |
|---|---|---|---|---|---|
| A02 | The AF accepts a VFL training subscription creation under an NF identity only if a real run of that NF sent that creation request (peer authentication), against the attacker that has the network alone. | `network` | proved | a02_create | checked |
| A02 | The AF accepts a subscription update under an NF identity only if a real run of that NF sent that update, against the attacker that has the network alone. | `network` | proved | a02_update | checked |
| A02 | The AF accepts a subscription deletion under an NF identity only if a real run of that NF sent that deletion, against the attacker that has the network alone. | `network` | proved | a02_delete | checked |

| target | claim | attacker | answer | query | control |
|---|---|---|---|---|---|
| A02 | The consumer accepts a VFL training notification only if the AF really sent that notification to it, against the attacker that has the network alone. | `network` | proved | a02_notif | checked |
| A03 | The notification the consumer accepts carries the subscriptionId, notifUri and notifCorrId of its own subscription at the AF, against the attacker that has the network alone. | `network` | proved | a03_notif | checked |
| A03 | The subscriptionId, notifUri, notifCorrId and vflPreFlag the consumer reads in the 201 response are exactly the ones the AF accepted for that creation, against the attacker that has the network alone. | `network` | proved | a03_created | checked |
| A03 | The subscriptionId, notifUri, notifCorrId and vflPreFlag the consumer sees in the update response are the ones the AF accepted for that subscription of that consumer, against the attacker that has the network alone. | `network` | proved | a03_update | checked |
| A04 | One creation request is not accepted twice: two acceptances of the same NF's creation must correspond to two runs of that NF. | `network` | attack (model_artifact) | a04_create | |
| A04 | A notification is accepted at most once for one send of the AF: the consumer cannot be made to accept the same notification twice. | `network` | proved | a04_notif | checked |
| A08 | The notification parameters are those of the consumer's own approved subscription: only the subscription owner's update may set them. | `malicious_nf` | attack (holds_in_source) | a08_origin (the control of a08_net) | |
| A08 | A notification for the victim's subscription is never delivered to another NF's notifUri. | `malicious_nf` | attack (holds_in_source) | a08_owner | |
| A08 | Every notification the AF sends uses the notifUri and notifCorrId of an update the subscription owner itself made with vflPreFlag set to false; no notification is sent while the flag is still true. | `network` | proved | a08_net | checked |
| Z01 | Every creation the AF accepts rests on an access token the NRF issued for the scope naf-vfl-train to the same NF identity, against the attacker that has the network alone. | `network` | proved | z01_create | none |
| Z01 | Every update the AF accepts rests on an access token issued for the naf-vfl-train scope to the same NF identity, against the attacker that has the network alone. | `network` | proved | z01_update | none |
| Z01 | Every deletion the AF accepts rests on an access token issued for the naf-vfl-train scope to the same NF identity, against the attacker that has the network alone. | `network` | proved | z01_delete | none |
| Z01 | Every creation the AF accepts rests on a token the NRF issued for the naf-vfl-train scope to that NF, here for the malicious NF that holds its own valid token. | `malicious_nf` | proved | z01_create | none |

| target | claim | attacker | answer | query | control |
|---|---|---|---|---|---|
| Z01 | Every update the AF accepts rests on a token the NRF issued for the naf-vfl-train scope to that NF, here for the malicious NF that holds its own valid token. | `malicious_nf` | proved | z01_update | none |
| Z01 | Every deletion the AF accepts rests on a token the NRF issued for the naf-vfl-train scope to that NF, here for the malicious NF that holds its own valid token. | `malicious_nf` | proved | z01_delete | none |
| Z03 | An NF holding a valid token for the service scope cannot update a subscription another consumer created. | `malicious_nf` | attack (holds_in_source) | z03_update | |
| Z03 | An NF holding a valid token for the service scope cannot delete a subscription another consumer created. | `malicious_nf` | attack (holds_in_source) | z03_delete | |
| Z03 | Notifications of a subscription reach only the notifUri and notifCorrId that the subscription's creator provided. | `malicious_nf` | attack (holds_in_source) | z03_notif (the control of z03_notif_net) | |
| Z03 | No NF can get an operation accepted under another NF's identity: the AF accepts an update only when the token it verifies was issued to the same identity that authenticated at the transport layer. | `network` | proved | z03_imp | none |
| Z04 | A notification for a subscription is sent only after an update by the subscription's owner that sets vflPreFlag to false. | `malicious_nf` | attack (holds_in_source) | a08_origin (the control of a08_net) | |
| Z04 | After the subscription is deleted, a repeated (replayed) update of the same subscription does not lead to a notification: the AF refuses it. | `network` | bounded (the changing value unrolled to 4 steps) | s05 | checked |
| Z04 | No notification is sent for a subscription while its vflPreFlag is still true: the AF notifies only after an update with the flag set to false. | `network` | proved | a08_net | checked |
| S05 | After deletion takes effect, a replayed update carrying the old subscriptionId, notifUri and notifCorrId does not produce a new notification. | `network` | bounded (the changing value unrolled to 4 steps) | s05 | checked |
| S05 | After deletion takes effect, neither a replayed update nor a fresh update from another NF carrying the old subscriptionId produces a new notification. | `malicious_nf` | bounded (the changing value unrolled to 4 steps) | s05 | checked |
| U01 | The consumer accepts a 201 only from the AF, for the subscriptionId the AF created for it and the notifCorrId it sent. | `network` | proved | u01_create | checked |
| U01 | The consumer accepts a notification only from the AF, for its own subscriptionId and the notifCorrId of its own subscription. | `network` | proved | u01_notif | checked |
| U02 | Under the network attacker, a notification is sent only to the notifUri and notifCorrId that the consumer whose subscription it is provided. | `network` | proved | z03_notif_net | checked |
| U02 | Only the NF that created a subscription may update it. | `malicious_nf` | attack (holds_in_source) | z03_update | |
| U02 | Only the NF that created a subscription may delete it. | `malicious_nf` | attack (holds_in_source) | z03_delete | |

| target | claim | attacker | answer | query | control |
|---|---|---|---|---|---|
| U02 | Notifications go only to the notifUri and notifCorrId the subscription's creator provided. | `malicious_nf` | attack (holds_in_source) | z03_notif (the control of z03_notif_net) | |
| U03 | Without credentials beyond the network, no NF can get an operation accepted under another NF's identity. | `network` | proved | z03_imp | none |
| U03 | A legitimate NF holding its own credentials cannot replace the content of another consumer's subscription. | `malicious_nf` | attack (holds_in_source) | z03_update | |
| U03 | A legitimate NF holding its own credentials cannot delete another consumer's subscription. | `malicious_nf` | attack (holds_in_source) | z03_delete | |
| U03 | A legitimate NF holding its own credentials cannot make the AF deliver another consumer's training notification to it. | `malicious_nf` | attack (holds_in_source) | z03_notif (the control of z03_notif_net) | |
| X01 | The access token the NWDAF obtains from the NRF for the naf-vfl-train scope never becomes known to an attacker that has only the network. | `network` | proved | x01_token | checked |

# 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-20261114T232842, 2026-11-14. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 51, prover 11 s, wall 13 min, cost 0.129 USD.

ProVerif: 86 proofs run, 86 decided, 0 without an answer, 10 s, 57 answers recorded.

## What was analysed

| | |
|---|---|
| **Protocol** | AF VFL client training API |
| **Version** | 19.2.0 |
| **Analysed** | A trusted AF acts as a VFL client and the NWDAF server calls Naf_VFLTraining directly; one capability check, one training round notification and cancellation, excluding the NEF/untrusted AF branch and global multi-client scheduling. |
| **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. Each property must have a reachability/positive control. Fixed to Rel-19. Distinguishes between an external network attacker and 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. A monotonic table cannot model real deletion, and 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 with corrected table positioning; the previous run was rejected by the implementation version lock during preparation, and the old inputs cannot be reused by bypassing the version lock. |
| **Source text** | 322 numbered passages (153032 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 | 5.2 | Naf_VFLTraining Service |
| main | 5.2.1 | Service Description |
| main | 5.2.1.1 | Overview |
| main | 5.2.1.2 | Service Architecture |
| main | 5.2.1.3 | Network Functions |
| main | 5.2.1.3.1 | Application Function (AF) |
| main | 5.2.1.3.2 | NF Service Consumers |
| main | 5.2.2 | Service Operations |
| main | 5.2.2.1 | Introduction |
| main | 5.2.2.2 | Naf_VFLTraining_Subscribe |
| main | 5.2.2.2.1 | General |
| main | 5.2.2.2.2 | VFL Training Subscription Creation |

| Document | Section | Title |
|---|---|---|
| main | 5.2.2.2.3 | VFL Training Subscription Update |
| main | 5.2.2.3 | Naf_VFLTraining_Unsubscribe |
| main | 5.2.2.3.1 | General |
| main | 5.2.2.3.2 | VFL Training Subscription Deletion |
| main | 5.2.2.4 | Naf_VFLTraining_Notify |
| main | 5.2.2.4.1 | General |
| main | 5.2.2.4.2 | VFL Training Notification |
| main | 6.1 | Naf_VFLTraining Service API |
| main | 6.1.1 | Introduction |
| main | 6.1.2 | Usage of HTTP |
| main | 6.1.2.1 | General |
| main | 6.1.2.2 | HTTP standard headers |
| main | 6.1.2.2.1 | General |
| main | 6.1.2.2.2 | Content type |
| main | 6.1.2.3 | HTTP custom headers |
| main | 6.1.3 | Resources |
| main | 6.1.3.1 | Overview |
| main | 6.1.3.2 | Resource: VFL Training Subscriptions |
| main | 6.1.3.2.1 | Description |
| main | 6.1.3.2.2 | Resource Definition |
| main | 6.1.3.2.3 | Resource Standard Methods |
| main | 6.1.3.2.3.1 | POST |
| main | 6.1.3.2.4 | Resource Custom Operations |
| main | 6.1.3.3 | Resource: Individual VFL Training Subscription |
| main | 6.1.3.3.1 | Description |
| main | 6.1.3.3.2 | Resource Definition |
| main | 6.1.3.3.3 | Resource Standard Methods |
| main | 6.1.3.3.3.1 | GET |
| main | 6.1.3.3.3.2 | PUT |
| main | 6.1.3.3.3.3 | PATCH |
| main | 6.1.3.3.3.4 | DELETE |
| main | 6.1.4 | Custom Operations without associated resources |
| main | 6.1.5 | Notifications |
| main | 6.1.5.1 | General |
| main | 6.1.5.2 | VFL Training Notification |
| main | 6.1.5.2.1 | Description |

| Document | Section | Title |
|---|---|---|
| main | 6.1.5.2.2 | Target URI |
| main | 6.1.5.2.3 | Standard Methods |
| main | 6.1.5.2.3.1 | POST |
| main | 6.1.6 | Data Model |
| main | 6.1.6.1 | General |
| main | 6.1.6.2 | Structured data types |
| main | 6.1.6.2.1 | Introduction |
| main | 6.1.6.2.2 | Type: VflTrainingSubs |
| main | 6.1.6.2.3 | Type: VflTrainingSubsPatch |
| main | 6.1.6.3 | Simple data types and enumerations |
| main | 6.1.6.3.1 | Introduction |
| main | 6.1.6.3.2 | Simple data types |
| main | 6.1.6.4 | Data types describing alternative data types or combinations of data types |
| main | 6.1.6.5 | Binary data |
| main | 6.1.6.5.1 | Binary Data Types |
| main | 6.1.7 | Error Handling |
| main | 6.1.7.1 | General |
| main | 6.1.7.2 | Protocol Errors |
| main | 6.1.7.3 | Application Errors |
| main | 6.1.8 | Feature negotiation |
| main | 6.1.9 | Security |
| main | 6.1.10 | HTTP redirection |

The purpose of the protocol as the source step located it in the documents: A trusted AF acts as a VFL client and the NWDAF server calls Naf_VFLTraining directly; one capability check, one training round notification and cancellation, excluding the NEF/untrusted AF branch and global multi-client scheduling.

## 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 |
|---|---|---|---|---|
| A02 | VFL request peer authentication | The AF (trusted VFL client) is assured that an accepted subscription creation/update/deletion request comes from the real NWDAF it names; the consumer is assured a notification comes from the AF. | TS 29.500/TS 33.501 require mutual authentication for direct NF-to-NF communication; the API uses TLS for the trusted AF and OAuth2 client credentials. | ts-33-501:2871, main:931, s0009, s0011, s0038, s0045, s0061, s0074, s0098, s0099 |

| ID | Name | About | What the documents say | Where |
|---|---|---|---|---|
| A03 | Agreement on subscription parameters | AF and consumer agree on subscriptionId, notifUri, notifCorrId, vflTrainSubs and vflPreFlag for the subscription they both act on. | The VflTrainingSubs representation carries notifUri, notifCorrId and vflPreFlag, which the AF echoes and later uses; notification delivery uses the notifCorrId of the subscription. | main:1126, s0038, s0039, s0074, s0241, s0244 |
| A04 | Freshness of requests and notifications | Every AF acceptance of a request and every consumer acceptance of a notification corresponds to an unused peer run; one request cannot be accepted twice. | TS 33.501 requires replay protection for SBI signalling; the API itself defines no request nonce or sequence number. | ts-33-501:2865, ts-33-501:2915, s0038, s0061, s0074 |
| A08 | Notification bound to approved subscription | The notification parameters (notifUri, notifCorrId, vflTrainSubs) used by the AF are those of the consumer's accepted subscription, and no notification is sent while vflPreFlag is true. | The VflTrainingNotify request is sent to the notifUri received during creation/update of the corresponding subscription; the vflPreFlag note forbids notifications before the flag is set to false. | main:1126, main:1103, s0074, s0213, s0241 |
| Z01 | Token and scope for each operation | Every accepted creation, update, deletion and notification delivery rests on an access token the NRF issued for the naf-vfl-train scope to that consumer identity. | Access to the API may be authorized with OAuth2 client credentials, NRF as authorization server, with a single scope naf-vfl-train. | rfc6749:2859, s0273, s0274, s0276 |
| Z03 | Per-subscription authorization | An NF holding a valid token for the service scope can only update, delete or receive notifications of the subscriptions it created. | The API defines a single service-level scope and no scope at resource or operation level; OAuth2 provides no method for a resource server to check that a token was issued to the presenting client. | rfc6749:2879, s0144, s0276 |
| Z04 | No notification after deletion | After a subscription is deleted, or before vflPreFlag is set to false, the AF sends no notification for it. | Unsubscribe requests deletion of the subscription at the AF; notifications go to a previously subscribed consumer only. | main:1126, s0054, s0061, s0062, s0074, s0241 |
| S05 | Deleted subscription raises no notification | After deletion takes effect, the old subscriptionId, notifUri and notifCorrId cannot cause a new notification. | Deletion removes the individual subscription resource at the AF; notifications are for previously subscribed consumers. | main:1126, s0054, s0061, s0062, s0074 |
| U01 | AF identity and correlation bound | The consumer accepts a 201 or a notification only from the AF, with the subscriptionId it was given and the notifCorrId of its own subscription; the training round of the notification is the one the subscription is for. | The 201 response carries the Location of the created resource; notifications are sent to the notifUri and carry the association established by the subscription. | main:1126, s0039, s0074, s0144, s0241 |
| U02 | Owner-only subscription control | Only the NF that created a subscription may update or delete it, and notifications go only to the notifUri and notifCorrId that NF provided. | The individual subscription resource is managed by the AF; no per-resource authorization is defined, only the service scope. | rfc6749:2879, s0045, s0061, s0074, s0144, s0276 |
| U03 | Malicious NF confined to own subscription | A legitimate NF holding its own credentials cannot replace the content (notifUri, notifCorrId, training round) of another consumer's subscription, nor update or delete it. | NFs in the same trust domain are mutually authenticated, but the API authorizes on a single service scope only. | ts-33-501:2859, ts-33-501:2875, rfc6749:2879, s0045, s0061, s0144, s0273, s0276 |
| X01 | OAuth2 token confidentiality in transit | The OAuth2 access token of the NF service consumer, which must stay confidential between the NRF, the consumer and the AF. | RFC 6749 requires access token credentials to be kept confidential in transit and used only over TLS; the API relies on TLS between the NFs. | rfc6749:2859, main:1141 |

Added from the documents by the analyst: X01.

## 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: 27. Not taken: 371. 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 | Framework-level guidance that points at the per-section threats; it states no requirement of the API. |
| P003 to P006 | rfc6749:2790-2823 | Z01, `malicious_nf` | |
| P007 to P010 | rfc6749:2827-2850 | Z01, `malicious_nf` | |
| P011 to P014 | rfc6749:2859-2877 | Z01, `malicious_nf` | |
| P015 | rfc6749:2879-2881 | Z03, Z01, `malicious_nf` | |
| P016 to P021 | rfc6749:2885-2917 | No goal stated | Refresh tokens are not used by the client-credentials grant this API authorizes with. |
| P022 to P025 | rfc6749:2921-2943 | No goal stated | The authorization-code grant is not used by the SBI client-credentials flow. |
| P026 to P029 | rfc6749:2947-2980 | No goal stated | Redirection-URI manipulation belongs to the authorization-code grant, which this API does not use. |
| P030 to P033 | rfc6749:2984-3003 | No goal stated | The resource-owner password grant is not used. |
| P034 to P035 | rfc6749:3013-3019 | Z01, `network` | |
| P036 | rfc6749:3023-3028 | A02, U01, `network` | |
| P037 to P039 | rfc6749:3032-3041 | No goal stated | Credentials-guessing attacks concern user-chosen passwords; the SBI client authenticates with a certificate, so no guessable secret is in the flow. |
| P040 to P042 | rfc6749:3045-3066 | No goal stated | Phishing attacks involve a resource owner and a user-agent; neither takes part in this SBI flow. |
| P043 to P048 | rfc6749:3070-3110 | No goal stated | CSRF is about browser requests carrying ambient credentials; no browser is involved and the consumer authenticates at the TLS and OAuth2 layer. |
| P049 to P050 | rfc6749:3119-3138 | No goal stated | Clickjacking is a user-agent concern, not part of this flow. |
| P051 to P052 | rfc6749:3142-3150 | No goal stated | Code injection and input validation concern API implementations, not a property of this message flow. |
| P053 to P054 | rfc6749:3154-3167 | Outside the scope | Open redirectors concern HTTP redirection, which in this flow is done by the SCP and is outside the modelled sub-flow. |
| P055 to P062 | rfc6749:3170-3210 | No goal stated | Implicit-grant token misuse does not apply to the client-credentials grant. |
| P063 | ts-33-501:2491-2491 | No goal stated | Bidding-down is about negotiating security features; the Naf_VFLTraining API defines no optional feature in Rel-19 (Table 6.1.8-1 is empty). |
| P064 to P074 | ts-33-501:2495-2517 | Outside the scope | Requirements on subscription authentication and on 5GC/NG-RAN keys for the UE, which this API sub-flow does not touch. |

| Paragraphs | Where | Taken up as | Why not |
|---|---|---|---|
| P075 to P213 | ts-33-501:2523-2853 | Outside the scope | Requirements on the UE, the gNB, the ng-eNB and NG-RAN key handling; they belong to the radio and UE parts of TS 33.501, which the task leaves out. |
| P214 | ts-33-501:2859-2859 | U03, `malicious_nf` | |
| P215 | ts-33-501:2865-2865 | A04, `network` | |
| P216 | ts-33-501:2867-2867 | Z01, `malicious_nf` | |
| P217 | ts-33-501:2869-2869 | No goal stated | Topology hiding is a property of discovery, not of the subscription and notification flow analysed here. |
| P218 | ts-33-501:2871-2871 | A02, U01, `network` | |
| P219 | ts-33-501:2873-2873 | A02 | |
| P220 | ts-33-501:2875-2875 | U02, Z03, A08, `malicious_nf` | |
| P221 to P222 | ts-33-501:2879-2881 | No goal stated | Descriptive text on what the NRF does; it states no requirement beyond P223-P227. |
| P223 to P225 | ts-33-501:2883-2887 | Z01, `malicious_nf` | |
| P226 to P227 | ts-33-501:2889-2891 | Z01, `malicious_nf` | |
| P228 to P234 | ts-33-501:2895-2907 | Outside the scope | NEF security requirements belong to the untrusted-AF branch, which the task excludes. |
| P235 to P241 | ts-33-501:2911-2923 | Outside the scope | SCP requirements concern indirect communication, which the task leaves out. |
| P242 to P312 | ts-33-501:2929-3075 | Outside the scope | End-to-end interconnection, SEPP and roaming-intermediary requirements belong to inter-PLMN roaming, which this single-operator sub-flow does not touch. |
| P313 to P314 | ts-33-501:3079-3081 | Outside the scope | IPUPS is an inter-PLMN border function, outside the analysed sub-flow. |
| P315 to P318 | ts-33-501:3085-3091 | No goal stated | The 3gpp-Sbi-Originating-Network-Id header requirement states no property of this subscription and notification flow. |
| P319 to P327 | ts-33-501:3097-3115 | Outside the scope | Signalling-monitoring requirements concern monitoring infrastructure that the task leaves out. |
| P328 to P337 | ts-33-501:3121-3141 | Outside the scope | Security visibility and configurability concern operator deployment, not the API message flow. |
| P338 to P375 | ts-33-501:3149-3231 | Outside the scope | Algorithm identifiers and selection, 5G-RG and NSSAAF requirements belong to other parts of TS 33.501. |
| P376 to P398 | ts-33-501:6540-11901 | Outside the scope | N3, EAP-TLS annexes, IAB, MBS and NEF annex requirements belong to parts of the system the task leaves out. |

## The parties and the protected values, one by one

The host listed the 20 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: 12. Without: 8. 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 holds the subscriptions and originates the notifications; a dishonest AF breaks every property asked about the service producer, and the documents ask none of them against the trusted AF (s0009). |
| S02 | NWDAF | party | `malicious_nf` | |
| S03 | NRF | party | Everything lost by definition | The NRF is the root of trust for the token; with it controlled the token property is lost by definition and the documents ask none of them under it. |
| S04 | Malicious legitimate NF | party | `malicious_nf` | |
| S05 | NF instance identity | value | `malicious_nf` | |
| S06 | Subscription resource | value | `malicious_nf` | |
| S07 | subscriptionId | value | `malicious_nf` | |
| S08 | notifUri | value | Nothing gained | Knowing the notifUri does not deliver the notifications, which go over TLS to the endpoint the URI names; substituting one's own URI is the update claim of Z03/U02. |
| S09 | notifCorrId | value | Nothing gained | Knowing the notifCorrId gives no access to the notification; the consumer checks it only to match its own subscription. |
| S10 | vflTrainSubs | value | Nothing gained | No target asks for confidentiality of the training-set description; it matters as the value the AF must accept from its owner, which reading it does not change. |
| S11 | vflPreFlag | value | `malicious_nf` | |
| S12 | Notification request | value | `malicious_nf` | |
| S13 | suppFeat | value | Nothing gained | Rel-19 defines no optional feature for this API (Table 6.1.8-1 is empty), so the attribute carries nothing. |
| S14 | OAuth2 access token | value | `malicious_nf` | |
| S15 | naf-vfl-train scope | value | `malicious_nf` | |
| S16 | access token subject and audience | value | `malicious_nf` | |
| S17 | AF identity | value | Everything lost by definition | The AF identity is the trust anchor of the consumer's authentication; with it held, every assurance about the producer is lost by definition. |
| S18 | Training-round identifier | value | `malicious_nf` | |
| S19 | Identity of the subscription creator | value | `malicious_nf` | |
| S20 | Subscription content | value | Nothing gained | The subscription content matters as the value the AF accepts from its owner (A03/A08); reading it changes no property asked here. |

## 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 | | |
| `malicious_nf` | its own NF identity, its own TLS credentials and its own OAuth2 access token for the naf-vfl-train scope, obtained legitimately from the NRF | Any authorized NF may obtain a token for the service scope; TS 33.501 assumes trust zones and requires NFs to validate incoming messages, so a legitimate but dishonest NF is in scope. | ts-33-501:2859, ts-33-501:2875, main:1147 |

| ID | `network` | `malicious_nf` |
|---|---|---|
| A02 | Holds | |
| A03 | Holds | |
| A04 | Holds | |
| A08 | Holds | Fails |
| Z01 | Holds | Holds |
| Z03 | Holds | Fails |
| Z04 | Holds | Fails |
| S05 | Holds | Holds |
| U01 | Holds | |
| U02 | Holds | Fails |
| U03 | Holds | Fails |
| X01 | Holds | |

## Source and targets

Source: 322 units, 153032 bytes, frozen in source-delivery-01.json. Targets: 11 approved (A02, A03, A04, A08, Z01, Z03, Z04, S05, U01, U02, U03); 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-1e783448602f196b8ceaeba2251f

This step as it reports itself: 1 requests (1 failed), 0.000 USD, 81443 tokens in and 5532 out, 262 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

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 |
|---|---|---|---|---|
| VFL Training Subscription record at the AF (vflTrainSubs, notifUri, notifCorrId, vflPreFlag, creator) | AF (VFL client, service producer) | created by POST on the collection resource (s0038, s0039), changed by PUT/PATCH on the individual resource (s0045, s0048), removed by DELETE (s0061, s0062) | on every request addressed to the individual subscription: the AF looks the resource up by subscriptionId and checks the token, but does not compare the token subject with the creator (s0144, s0273, s0276) | s0038, s0039, s0045, s0061, s0144, s0267, s0276 |
| vflPreFlag (preparation flag of the subscription) | AF | set true at creation for VFL preparation, set false by the update that activates notification (s0241) | before the AF sends a notification: no notification until the flag is false (s0241 NOTE) | s0241, s0045 |
| notifCorrId (notification correlation identifier) | AF and NWDAF | set at creation or replaceable by PUT/PATCH (s0241, s0244) | the consumer matches the notifCorrId of a notification against its own subscription | s0241, s0244, s0074 |

| pattern | result | target | shown by ProVerif | steps |
|---|---|---|---|---|
| replay | nothing found | | | The recorded update is delivered again after the deletion; the unrolled AF reads it and refuses it (bounded.pv query refdel is falsified, so the step really happens) and no notification follows (query s05 verified). A replayed creation is accepted as a second subscription only because the model's TLS is deterministic encryption; that is the model artifact recorded under A04. |
| repeat_one_time_action | nothing found | | | The notification phase runs once per subscription and the AF then accepts only the deletion; a second update is not accepted in its place in the unrolled model. |
| desynchronisation | nothing found | | | This pattern is real for the subscription record but is shown on the unbounded model: a second NF's update replaces the record's notifUri and notifCorrId and the consumer never sees the notification of the subscription it still owns. It is recorded as the attack answers of U03 and Z03 (query z03_notif on model.pv); bounded.pv carries no second identity, so the same run cannot be shown there. |

| pattern | result | target | shown by ProVerif | steps |
|---|---|---|---|---|
| update_before_check | nothing found | | | The AF changes the stored record on an update without checking that the token subject is the subscription's creator; that is recorded as the attack answers of Z03 (query z03_update on model.pv). In bounded.pv the only writer is the consumer whose key encrypts the update, so the missing check has no visible effect there. |
| no_rollback_on_failure | nothing found | | | A run abandoned before the deletion leaves the subscription and its notifUri in place, but no notification is issued after that point within the subscription's own lifecycle; the source says nothing about retry, so nothing beyond that is claimed. |
| race_between_sessions | nothing found | | | The model gives one subscriptionId per consumer identity, so two sessions of one consumer are not distinguished and a race on the same record is not modelled. Concurrent sessions of one consumer against a deployment that issues one subscriptionId per creation are outside what this model can say. |
| distinguishable_failure | nothing found | | | The AF's 403 causes (OVERLOAD, NOT_AVAILABLE_FOR_VFL_PROCESS, VFL_REQS_NOT_MET) are not modelled and the source does not say whether they tell the requester anything about the subscription; the model only has accept and no-reply. |
| wrap_or_window | nothing found | | | The API carries no counter, sequence number or acceptance window; the subscriptionId is an identifier, not a number, so nothing can wrap. |

What the analyst named for targets:

| step | targets | features | reason |
|---|---|---|---|
| 38 | Z04, S05 | changing_state | The AF keeps a subscription record whose state changes: created in preparation (vflPreFlag true), activated by the update, notified once, and deleted. Z04 and S05 ask what happens to a message that arrives after the deletion, which is a value that changes across steps; model.pv leaves the check out and bounded.pv unrolls four steps of the AF. |
| 38 | A02, A03, A04, A08, Z01, Z03, U01, U02, U03, X01 | plain_primitives, many_sessions | The model uses only symmetric encryption, MAC and a private key-derivation function for the TLS-style endpoint channel, and a MAC for the NRF's token; there is no exclusive-or or Diffie-Hellman algebra. The queries are asked for any number of sessions of both the consumer and the AF; the only fixed bound is the single honest and single malicious identity. |

## The analyst's closing note

Model: model.pv gives the Naf_VFLTraining flow of the trusted AF and the NWDAF (creation with notifUri/notifCorrId/vflPreFlag as capability check, activation update, one notification with acknowledgement, deletion), with TLS modelled as per-endpoint MAC-plus-encryption, OAuth2 client credentials with the NRF as authorization server and a single scope naf-vfl-train, plus a second, malicious but legitimate NF holding its own token. bounded.pv unrolls the AF to four steps (create, update/notify, delete, one repeated update); bounded-buggy.pv is the control in which the deletion check is left out.

Established: against the network attacker, authentication of the AF and of the consumer (A02, four claims with controls), agreement on subscriptionId/notifUri/notifCorrId/vflPreFlag (A03), no double acceptance of a notification (A04 second claim), notification only after the owner's update with vflPreFlag false (A08 network claim, Z04 flag claim), token validity and subject binding for every accepted operation (Z01, both attackers), consumer-side binding of AF and correlation identifiers (U01), no operation accepted under another NF's identity (Z03/U03 network claims), and token secrecy (X01, control falsified). S05 and Z04's deletion claim hold on the bounded model with 4 steps, and the control shows they depend on the AF dropping the record.

Finding (one, high, novelty unclear): a legitimate NF with its own token for the single service scope can update or delete another consumer's subscription and make the AF deliver that consumer's VFL training notification to its own notifUri, and can suppress reporting by rewriting vflPreFlag. The documents state the ingredients (one service scope, no resource-level scope; RFC 6749's statement that a resource server cannot bind a token to a client) but not the consequence. Recommendation: record the creator identity and compare it with the token subject, or define resource-level scopes; not checked on a variant model.

Limits: TLS without handshake or record state, so the creation replay ProVerif found is a model artifact (submitted as such); two identities only; subscriptionId treated as knowable; OAuth2 assumed deployed though the source makes it optional; deletion answered within four unrolled steps. Tamarin+ Agent would settle the TLS replay question and the deletion behaviour beyond the bound.