Security analysis of AF training server subscription

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	10: 6 hold, 2 fail (3 attacks), 2 without an answer
Findings	1

Contents

1 Protocol and scope

2 Findings

3 Properties examined

4 Limits of this analysis

Appendix A: Reading of each property

Appendix C: Claims and prover answers

Appendix D: How the analysis was run

1 Protocol and scope

The Naf_Training service is the service-based interface of an Application Function acting as the VFL server. A Network Data Analytics Function, or a Network Exposure Function when the AF is untrusted, subscribes to the training events of an existing training task, receives the training reports at the callback URI it registered, updates the subscription and later deletes it. The AF holds the training subscriptions and the reports; the NRF issues the OAuth2 access token with which a consumer is authorized.

The consumer is sure that a notification it accepts was sent by the AF for its own subscription. The AF is sure that every request it serves carries a token the NRF issued for the consumer that presents it. Neither is sure that the consumer sending a request is the one that created the subscription, and that is where the exchange breaks.

The parties:

- **NWDAF (or NEF)**: NF service consumer. Holds: access token (naf-train scope), its notifUri, its notifCorreId.
- **AF**: VFL server, NF service producer. Holds: the subscriptions, the training reports.
- **NRF**: authorization server. Holds: token key.

Figure 1 shows the exchange as the documents specify it.

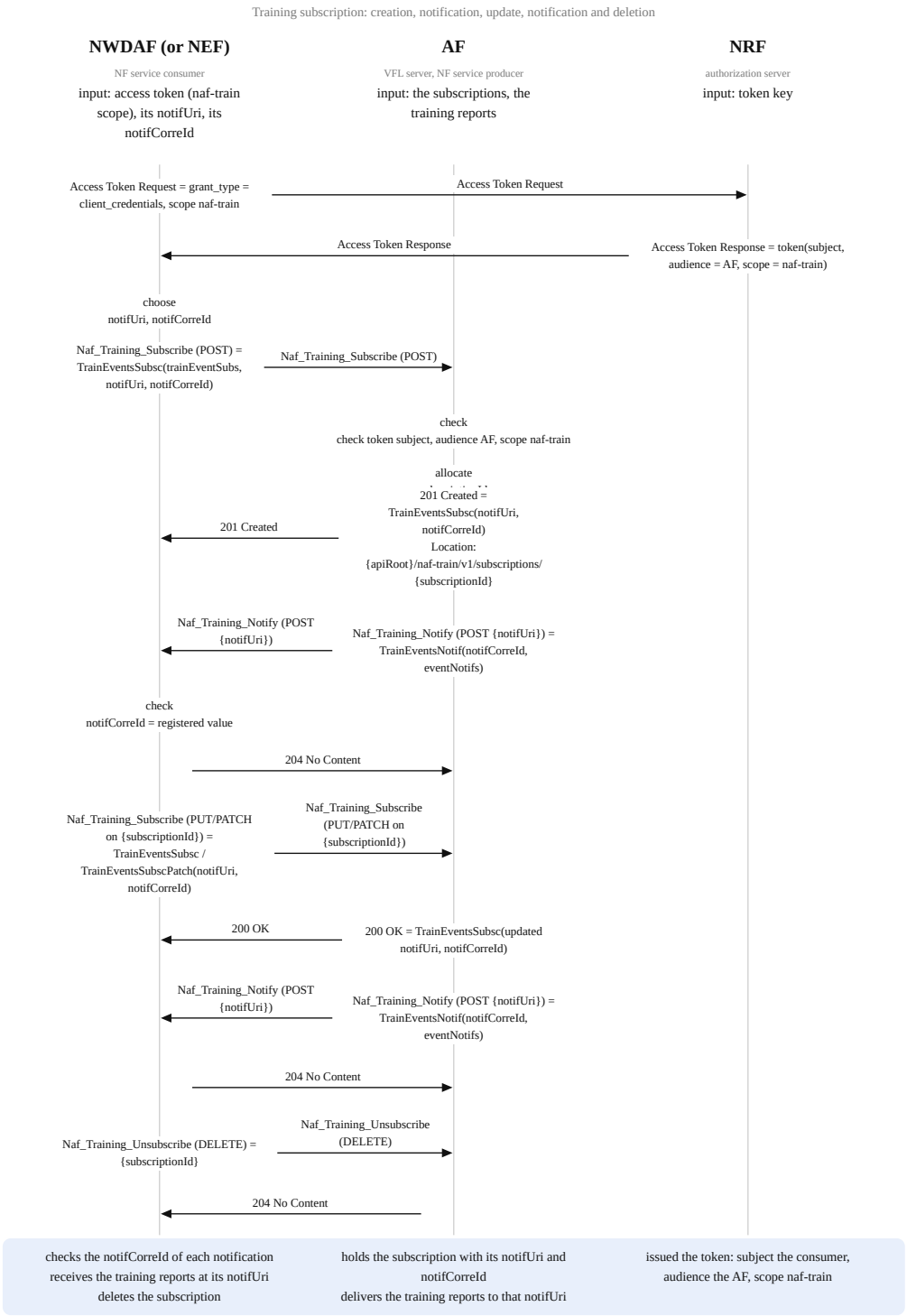


Figure 1. The protocol as specified.

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 authorization requirement holds and the notification binding holds: the AF applies a creation, update or deletion only with a token the NRF issued for the requesting consumer, and every notification goes to the endpoint and correlation identifier registered for its subscription, for any number of sessions. The finding is the other half of the authorization: because the API defines one service-level scope and no scope at resource level, a second legitimate NF that knows another consumer's subscription URI updates that subscription to receive its training reports, and deletes it. The documents neither state nor forbid this, so it is a gap to close in the specification. Everything that rests on the AF's subscription record is answered only for three unrolled steps, the retried creation and the time after a deletion are answered at most by reachability, no proof carries a control variant, and the recommended fix was not verified on a model.

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 A legitimate NF can take over another consumer's training subscription and receive its reports

Kind	Weakness of the protocol as specified
New	Unclear: the documents state it in part, or could not be searched (main:5855, ts-29-500:4996, rfc6749:2827)
Value	high
Properties	Z01.10, Z03 (section 3)
Source text	s0045, s0061, s0134, s0288
Shown by	ProVerif attack queries `z0110_upd`, `z0110_del`; Figures 2, 3 and 4

What happens. A second NF service consumer sends PUT on the Individual Training Subscription of another consumer, with its own valid naf-train token, and asks that the notifUri of that subscription be its own endpoint; the AF validates the token, finds the expected audience and scope, and applies the update, because the only authorization it applies is the service-level scope. The AF then sends the notifications of that subscription to the second NF. With the same token the second NF can also send DELETE on that subscription, and the AF deletes it.

What the attacker needs. A legitimate NF service consumer with its own OAuth2 token for the naf-train scope, which the NRF issues to any consumer, and the Individual Training Subscription URI of the other consumer. That URI has the fixed public form {apiRoot}/naf-train/v1/subscriptions/{subscriptionId} and nothing in the specification requires the subscriptionId to be unguessable or private.

What the attacker gains. The training reports of that subscription - the analytics event, the accuracy, the termination cause, the VFL correlation identifier - are delivered to the other NF, and the subscription can be removed under the owner. Nothing tells the owner: it simply stops receiving reports.

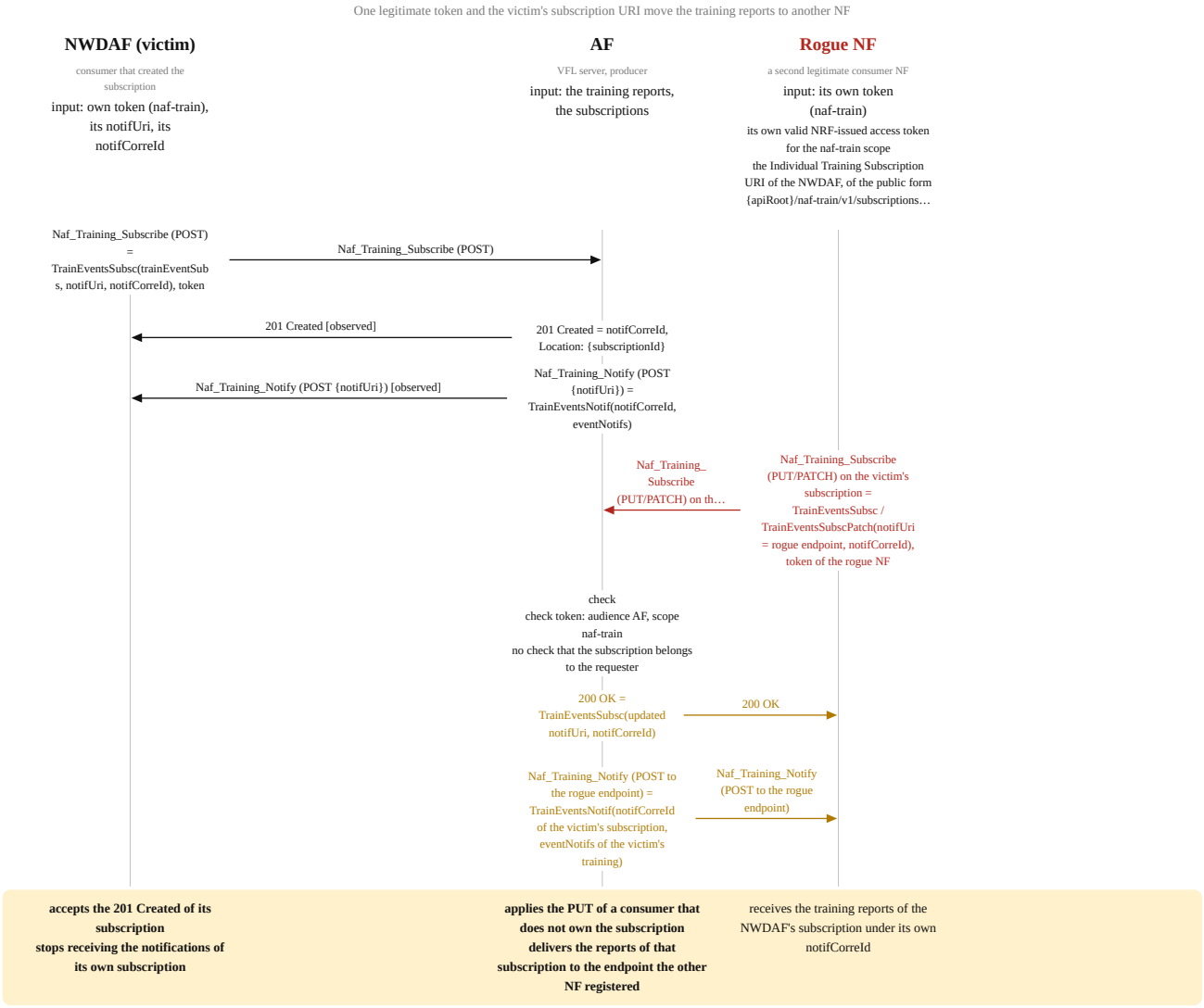


Figure 2. One legitimate token and the victim's subscription URI move the training reports to another NF. Property Z01.10; attack query z0110_upd.

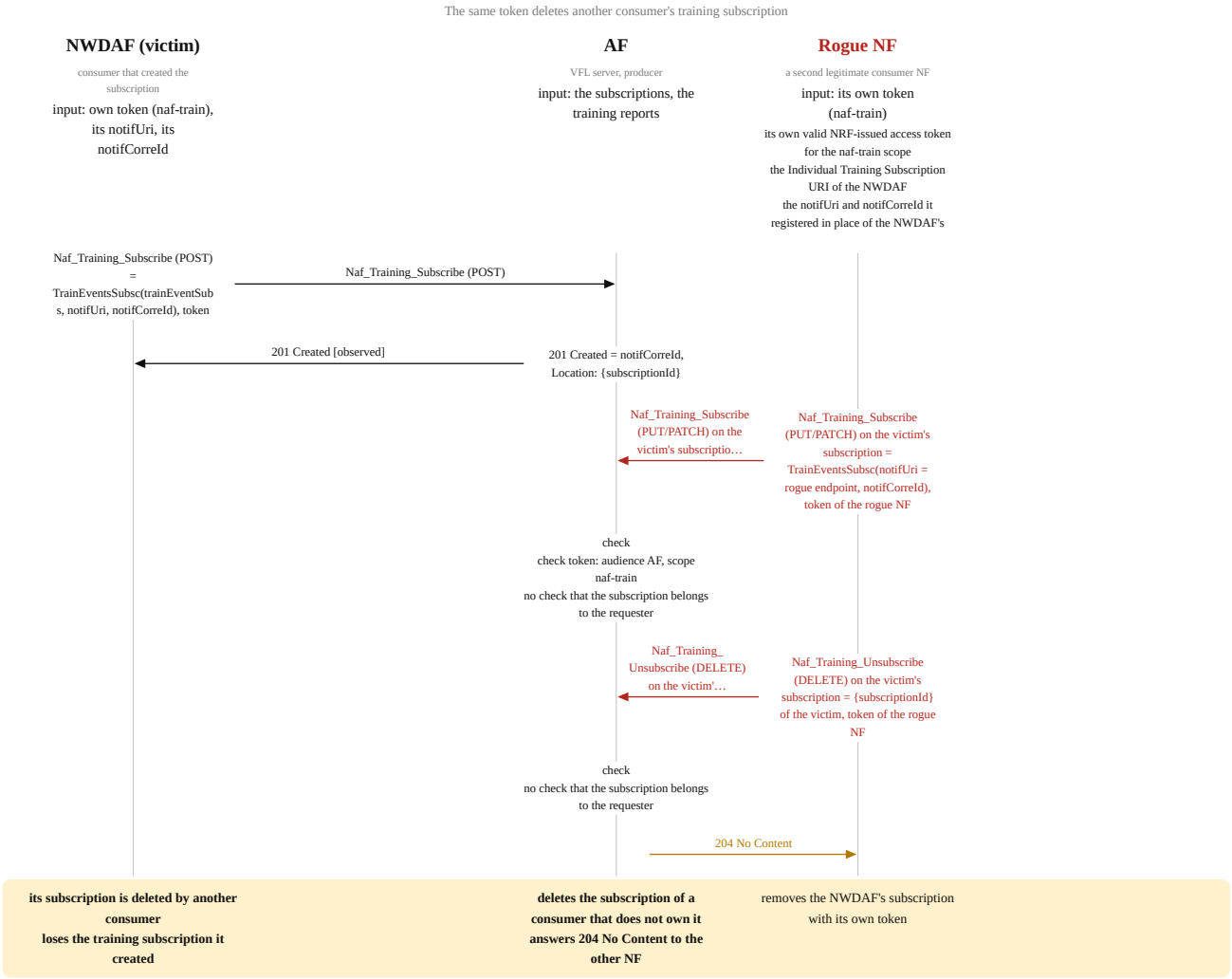


Figure 3. The same token deletes another consumer's training subscription. Property Z01.10; attack query z0110_del.

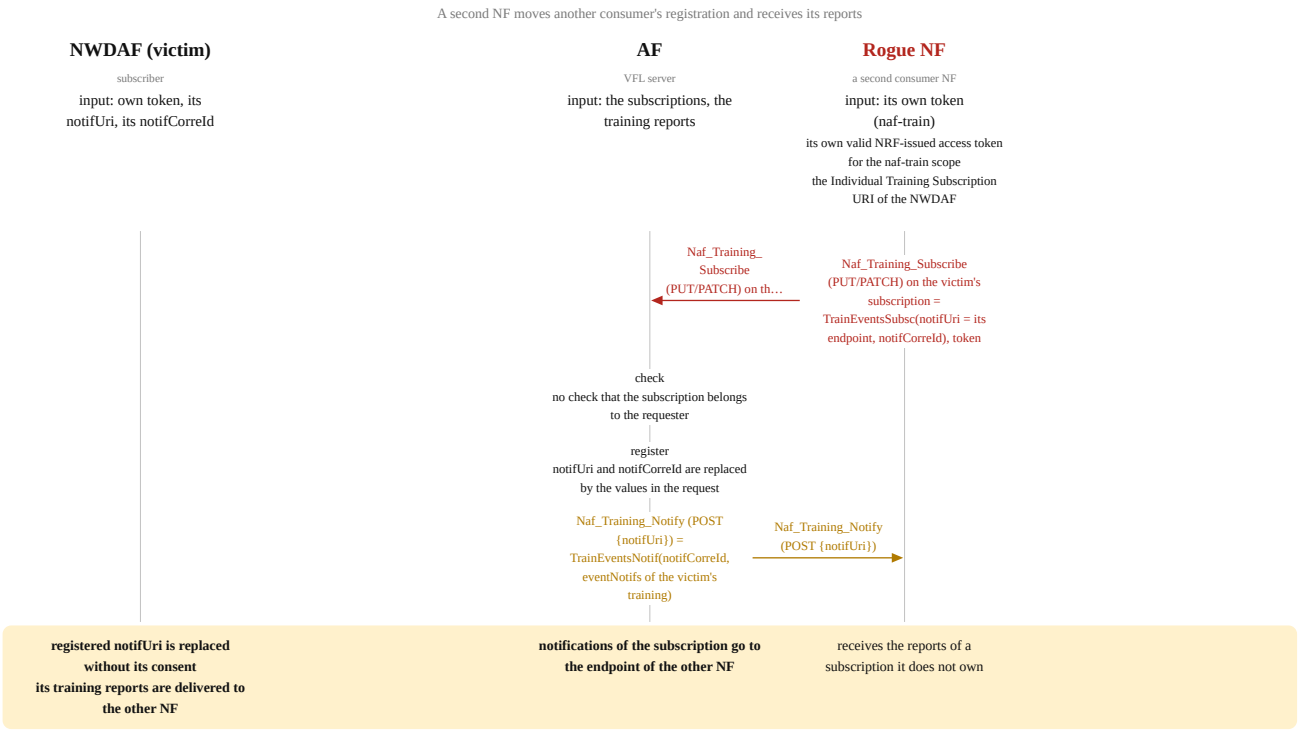


Figure 4. A second NF moves another consumer's registration and receives its reports. Property Z03; attack query z0110_upd.

Proposed change. Bind each Training Subscription to the consumer that created it and check the binding on every later request: the AF stores the token subject of the successful POST and requires, on PUT, PATCH, GET and DELETE of the Individual Training Subscription, that the token subject equals the stored subject. Alternatively clause 6.3.9 defines an operation- and resource-level scope for these operations, as TS 29.500 allows where a specific authorization is required, and the AF requires it on the Individual Training Subscription resource.

Check of the change with ProVerif. Not checked: The change needs a per-subscription owner record at the AF and a comparison on PUT, PATCH, GET and DELETE. bounded.pv keeps such a record for three steps but does not carry the identity of the consumer that created it, so the fix would have to be modelled first; that was beyond this session and the change is therefore not verified.

No figure was given for these attacks of a finding: Z03: z0110_del.

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
Z01	**Authorization of every subscription operation.** Every time the AF accepts a creation, update or deletion of a training subscription, there must be a valid grant held by the requesting consumer and issued by the authorization server.	no compromise	Holds	medium	yes
Z03	**Notification bound to its subscription.** Every training notification the AF sends goes to the notifUri registered when that subscription was created/updated, and carries the notifCorreId corresponding to that subscription; a grant must not be movable to another consumer or another subscription.	no compromise	Fails (1 attack)	high	unclear

ID	Property	Compromise assumed	Result	Value	In the documents
D01	**Notification only after a real send.** Every time the consumer accepts a training notification, there is a real send by the AF to that consumer, and the notifCorreId matches the eventNotifs content.	no compromise	Holds	low	yes
A03	**Matching notification delivery, no replay.** Every time the consumer accepts a notification, there is a matching send by the AF, with the consumer identity, notifUri, notifCorreId and event content all consistent.	no compromise	Holds	low	yes
A08	**Created resource matches the request.** The subscription resource and Location returned by the AF correspond to the TrainEventsSubsc parameters submitted by the consumer; later notifications use the notifUri and notifCorreId registered for that subscription.	no compromise	Holds	low	yes
R03	**One side effect per operation.** The same logical subscription creation/update/deletion operation commits at most one side effect at the AF.	no compromise	Reachable only	low	unclear
S05	**Nothing after deletion.** After a subscription deletion takes effect, the AF no longer sends notifications for that subscription to that subscription's notifUri, and the old subscription identifier cannot trigger notifications again.	no compromise	No answer	low	unclear
Z03.8	**notifUri and notifCorreId binding (Q1).** Every training notification the AF sends goes to the notifUri registered for that subscription and carries the notifCorreId corresponding to that subscription.	no compromise	Holds	low	yes
U01	**Report bound to its training task (Q2).** Every training report the consumer accepts corresponds to a real send by the AF for the same training task (event, vflCorrId), and the report category is consistent.	no compromise	Holds	low	yes
Z01.10	**No cross-consumer subscription control (Q3).** Every time the AF accepts a subscription update or deletion, there must be a valid grant held by the requesting consumer; an NF holding its own token must not be able to update or delete another NF's subscription without authorization for that subscription.	no compromise	Fails (2 attacks)	high	unclear

The questions asked, and the analyst's answers:

Question	Answer	Rests on
Q1: Whether the training event subscriber is bound to the notification URI and its related identifiers	The binding holds in the direction the AF controls: every notification goes to the notifUri registered for that subscription and carries its notifCorreId, proved for any number of sessions. What is not bound is the registration itself: another legitimate NF can PUT a new notifUri into a subscription it does not own, and the reports then go to it under its own notifCorreId. The specification never says how the AF checks that a notifUri belongs to the consumer that submits it.	Z03.8, Z03, Z01.10
Q2: Whether success/failure/termination reports cannot be mixed across training tasks	A party on the network cannot mix up reports between tasks: every report the consumer accepts corresponds to a notification the AF sent, with the same event, the same notifCorreId and the same VFL correlation identifier, and each notification matches a distinct send. Mixing arises only through the takeover: the second NF receives another task's reports already sorted under its own notifCorreId, so they cannot be told apart from its own.	U01, A03, Z01.10
Q3: Whether an NF holding its own token can update/cancel other subscriptions, distinguishing policy conditions per the specification	With its own token and the other consumer's subscription URI, a legitimate NF updates that subscription and receives its reports; the token check passes because the API defines one service-level scope and no scope at resource level. The separation between consumers thus rests on a policy that clause 6.3.9 leaves to local configuration and on the subscription URI not being known, not on an authorization the AF can check. The deletion half of the question could not be settled.	Z01.10, Z01, Z03

4 Limits of this analysis

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

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

- The AF's subscription record is unrolled to three steps in bounded.pv (creation, update, deletion) and is not replicated; a fourth or later request on the same record, and the interleaving of two subscriptions, are outside what was analysed.
- In model.pv the AF serves one subscription per process instance, so nothing there speaks about a request that arrives after the lifecycle it holds, and the deletion path was not reachable there at all.
- TLS is one private channel per endpoint and the token is a MAC only the NRF can produce; token expiry, TLS key material and the strength of either primitive are not modelled.
- The individual subscription URI is treated as an identifier another NF can know; if subscriptionId were unguessable and never leaked, the takeover would need an additional disclosure of the URI, which the specification neither requires nor forbids.
- The AF's training computation, the VFL clients, an untrusted AF or NEF, HTTP redirection (307, 308), the SCP, feature negotiation and the OAuth2 exchange with the NRF beyond issuing a token are outside the model.
- Notification delivery is one POST per notification with a 204 acknowledgement; retries, several notifications per subscription and the ordering between a deletion and a notification in flight are not modelled.
- The proofs carry no control variant, so the report says of each that nothing shows the property can fail in this model; the one control attempted became a proof when the model was reduced.
- The recommended fix was not verified on a model, and the retried creation (R03) could only be answered at reachability, because no model here sends a creation request twice.

What the host records:

- No answer for: R03, S05.
- The answers changed after the reading was written for: R03.
- The answers of Z01, R03, S05, Z01.10 rest on an operation the model simplifies (Appendix D).
- The prover gave no answer (Appendix C): S05: s05, Z01.10: z0110_del. The property was asked and the prover neither proved it nor found an attack, so the property is shown without an answer.
- Set by the analyst and without a proof or an attack (Appendix D): R03 under `legal_nf`.
- Proved without a control query (Appendix C): Z01: z01_upd, Z01: z01_create, Z01: z01_del, Z03: z03_reg, D01: d01, A03: a03, A08: a08, A08: z03_reg, Z03.8: z03_reg, U01: d01. 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.

Z01: Authorization of every subscription operation

Every time the AF accepts the creation, update or deletion of a training subscription, the request must carry a valid grant issued by the authorization server and held by the consumer making the request.

Every creation, update and deletion the AF applies is backed by an access token the NRF issued for the consumer named in the request, with the AF as audience and the naf-train scope, and an NF without a token cannot make the AF accept one. The deletion is covered on the unrolled model, where the record is created, replaced and removed. The check says whose token it is, not what that consumer is entitled to do with the resource it names.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: It confirms the token requirement the specification and TS 29.500 state and shows the check holds for what it checks; little is new for a reader.
- Stated in the documents: yes. Where the documents say it: main:5849, main:5855
- Rests on: The token is a MAC under a key shared by the NRF and the AF; its expiry, its format and the channel to the NRF are not modelled.; The token subject is taken as the identity of the TLS peer, as the baseline binds them; no client certificate is carried.; The deletion is answered for the three steps the record is unrolled to, not for any number of requests.

Z03: Notification bound to its subscription

Every training notification the AF sends goes to the notifUri registered when the subscription was created or updated and carries the notifCorreId corresponding to that subscription; the grant cannot be moved to another consumer or another subscription and used there.

Each notification the AF sends goes to the notifUri its subscription registered, with that subscription's notifCorreId, for any number of sessions. The registration itself is not protected: a second NF with its own valid token overwrites the notifUri of a subscription it did not create, and the AF then delivers that consumer's training reports to the second NF.

- Result: Fails (1 attack); answers what is asked: yes; usable with caveats
- Value high: The failure is not stated in the documents: they say the API has one service-level scope but never that this lets another NF move a registration and receive the reports.
- Stated in the documents: unclear. Source text it rests on: main:5124, s0288, s0134
- Rests on: The attacking NF is assumed to know the individual subscription URI; the specification fixes its form and requires nothing about the secrecy of the identifier.; The notification path is a private channel per endpoint, that is TLS between the AF and the consumer; a network attacker can neither read nor forge it, and that part is assumed, not proved.

D01: Notification only after a real send

Every time the consumer accepts a training notification, there is a real send from the AF to that consumer, and the notifCorreId matches the content of eventNotifs.

Every notification the consumer accepts was sent by the AF for that consumer's subscription, with the same notifCorreId and the same report content. A party on the network cannot produce a notification the consumer would accept, because the callback endpoint is bound to the AF.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: It is the property the procedure implies once the notification path is protected, and it holds; nothing new for a reader.
- Stated in the documents: yes. Where the documents say it: main:1425, main:5124
- Rests on: TLS between the AF and the consumer is a private channel per endpoint, so the network has no way to write on it.; One notification per subscription print is modelled, with a 204 acknowledgement.

A03: Matching notification delivery, no replay

Every time the consumer accepts a notification, there is a matching send from the AF, and the consumer identity, notifUri, notifCorreId and event content all agree.

Each notification the consumer accepts matches a distinct notification of the AF for that subscription and that report, so a recorded notification cannot be presented twice under the same correlation identifier, for the sessions the model covers.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: It holds for the same reason as the previous property and says nothing the documents do not already assume.
- Stated in the documents: yes. Where the documents say it: main:1425
- Rests on: The answer comes back on the AF's own endpoint channel, which is the TLS binding; no replay rule of the specification is modelled because none exists.

A08: Created resource matches the request

The subscription resource returned by the AF and the Location correspond to the TrainEventsSubsc parameters submitted by the consumer; subsequent notifications use the notifUri and notifCorreId registered for that subscription.

The 201 Created the consumer accepts names a subscription the AF created for that same consumer. It echoes the same notifCorreId and the same notifUri. Every later notification of that subscription uses the registered endpoint and correlation identifier.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: It is the echo of the parameters the procedure requires, proved rather than assumed, but not in doubt.
- Stated in the documents: yes. Where the documents say it: main:1065, main:1425
- Rests on: The AF creates a fresh identifier for every accepted POST; the model cannot show how it is chosen or how long it lives.

R03: One side effect per operation

The same logical subscription create/update/delete operation commits a side effect at most once on the AF side.

The answers of this target changed after this reading was written.

Nothing is established beyond the situation: a creation request reaches the AF and produces an Individual Training Subscription with its own identifier. The AF keeps no request identifier, and the specification defines no idempotency rule. A consumer that retries after a lost 201 would therefore end up with two subscriptions. A derivation of the prover says as much, but no model here sends the request twice, so it is not a trace.

- Result: Reachable only; answers what is asked: no; not usable
- Value low: The prover gave only a reachability; the duplicate effect is a reading of the specification, not a result.
- Stated in the documents: unclear
- Rests on: The AF creates a subscription for every accepted POST and keeps no record of the request that produced it.; No retry, no timeout and no request identifier is modelled.
- Open: Whether a retried POST creates a second subscription, which is what the target asks.

S05: Nothing after deletion

Once a subscription deletion takes effect, the AF no longer sends notifications of that subscription to the subscription's notifUri, and the old subscription identifier cannot trigger notifications again.

Only the situation is established: a deletion is answered 204 No Content and, in the unrolled AF, a request that names the deleted subscription afterwards produces no notification and no answer. Whether a notification can still reach the consumer after its deletion, which is what the target asks, is not shown, because the model cannot order a notification and a deletion.

- Result: No answer; answers what is asked: no; not usable
- Value low: Reachability of a deletion alone answers nothing about what follows it; the property itself is not established.
- Stated in the documents: unclear
- Rests on: The AF's record is unrolled to three steps and a request after the deletion is answered by a process that emits nothing; that is a model choice, not a statement of the specification.; No notification in flight is modelled, and the specification says nothing about their ordering.
- Open: What the AF does with a notification or a request that is in flight when the deletion is applied; the model has no such overlap.

Z03.8: notifUri and notifCorreId binding (Q1)

Every training notification the AF sends goes to the notifUri registered for that subscription and carries the notifCorreId corresponding to that subscription.

Every notification the AF sends in a subscription goes to the notifUri that subscription registered and carries its notifCorreId, so a consumer recognises its own subscriptions from the notification alone, for any number of sessions.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: It is the correlation the procedure states, proved for any number of sessions; useful as a basis, not a result on its own.
- Stated in the documents: yes. Where the documents say it: main:1425, main:5124
- Rests on: The consumer keeps the notifCorreId it chose and compares it with the one in the notification; the model matches it exactly.; The AF registers a new notifCorreId when the subscription is updated.

U01: Report bound to its training task (Q2)

Every training report the consumer accepts corresponds to a real send by the AF for the same training task (event, vflCorrId), and the report category is the same.

Every training report the consumer accepts corresponds to a notification the AF sent for the same event and the same report, so one task's report cannot be presented as another task's report. The correlation of a report to its task rests on the AF and on nothing the consumer can check beyond the notifCorreId.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: It holds and it is what the documents describe; the sharp question, whether another task's report reaches a consumer that did not ask for it, is answered by the takeover, not here.
- Stated in the documents: yes. Where the documents say it: main:5124
- Rests on: The report content is a fresh value of the AF per notification; the model does not distinguish the report categories beyond carrying one content value.; vflCorrId is carried as part of the report and is not something the consumer can verify.

Z01.10: No cross-consumer subscription control (Q3)

Every time the AF accepts a subscription update or deletion, the requesting consumer must hold a valid authorization; an NF holding its own token cannot update or delete another NF's subscription without authorization for that subscription.

A second NF holding its own valid naf-train token and the URI of another consumer's subscription updates that subscription so that its notifications come to the second NF, and the AF applies the change. With the same token it deletes the subscription, and the AF removes it. The AF checks the token, the audience and the scope, but nothing ties a subscription to the consumer that created it, so the service-level scope authorizes every consumer for every subscription.

- Result: Fails (2 attacks); answers what is asked: yes; usable with caveats
- Value high: It is a weakness of the specified authorization, not stated in the documents, with a concrete impact on the training reports and the subscription of another consumer; it needs the subscription URI to be known, which the specification neither requires nor forbids.
- Stated in the documents: unclear. Source text it rests on: s0288, s0134, s0045, s0061
- Rests on: The attacking NF is assumed to know the other consumer's subscription URI; the specification fixes the URI form and requires nothing about the secrecy of the subscriptionId.; The AF authorizes only on the token and the service-level scope, as clause 6.3.9 describes; a deployment that checks ownership by local configuration would not be attacked by these traces.; The deletion is shown on the unrolled model, for three steps of the record.

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
Z01	proved		z01_create			2 proved, 1 bounded
Z03	proved		z03_reg		z0110_upd (depends_on_ambiguity)	1 proved, 1 attack
D01	proved		d01			1 proved
A03	proved		a03			1 proved
A08	proved		a08			2 proved
R03	reachable		r03			1 reachable
S05	reachable		s05			1 reachable, 1 undecided
Z03.8	proved		z03_reg			1 proved
U01	proved		d01			1 proved
Z01.10	none				z0110_upd (depends_on_ambiguity); z0110_del (depends_on_ambiguity)	2 attack, 1 undecided

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
Z01	The AF applies the update of a Training Subscription only with an access token the NRF issued for the requesting consumer.	`legal_nf`	proved	z01_upd	none
Z01	The AF accepts the creation of a Training Subscription only with an access token the NRF issued for the requesting consumer, for the AF as audience and with the naf-train scope.	`network`	proved	z01_create	none
Z01	The AF applies the deletion of a Training Subscription only with an access token the NRF issued for the requesting consumer.	`legal_nf`	bounded (the changing value unrolled to 3 steps)	z01_del	none
Z03	Each notification the AF sends for a subscription goes to the notifUri and with the notifCorreId registered for that subscription.	`legal_nf`	proved	z03_reg	none
Z03	The registration of a subscription cannot be moved to another consumer: no other NF with its own valid naf-train token can change the notifUri of a subscription it does not own.	`legal_nf`	attack (depends_on_ambiguity)	z0110_upd	
D01	Every notification the consumer accepts corresponds to a notification the AF sent for that subscription, with the same notifCorreId and the same report content.	`network`	proved	d01	none
A03	Every notification the consumer accepts matches a distinct notification of the AF for the same subscription and the same report; a notification cannot be replayed under the same correlation identifier.	`network`	proved	a03	none

target	claim	attacker	answer	query	control
A08	The 201 Created representation and the Location URI the AF returns correspond to the TrainEventsSubsc the consumer submitted (same notifCorreId, same notifUri).	`network`	proved	a08	none
A08	Later notifications use the notifUri and the notifCorreId registered in the subscription.	`network`	proved	z03_reg	none
R03	A creation request reaches the AF and produces one Individual Training Subscription with its own identifier.	`legal_nf`	reachable	r03	
S05	After the DELETE is answered 204 No Content the AF sends no further notification for that subscription and the identifier cannot be used again to obtain notifications.	`legal_nf`	undecided	s05	
S05	A deletion is answered 204 No Content and, in the unrolled AF, a request that names the deleted subscription afterwards produces no notification and no answer.	`legal_nf`	reachable	s05	
Z03.8	Every notification the AF sends for a subscription goes to the notifUri registered in that subscription and carries that subscription's notifCorreId, so a consumer recognises the notifications of its own subscriptions.	`legal_nf`	proved	z03_reg	none
U01	Every training report the consumer accepts corresponds to a notification the AF sent for the same training task (event, vflCorrId) with one report category.	`network`	proved	d01	none
Z01.10	Another NF with its own valid naf-train token cannot update a subscription it does not own, nor move its notifications to itself.	`legal_nf`	attack (depends_on_ambiguity)	z0110_upd	
Z01.10	Another NF with its own valid naf-train token cannot delete a subscription it does not own.	`legal_nf`	undecided	z0110_del	
Z01.10	Another NF with its own valid naf-train token cannot delete a subscription it does not own.	`legal_nf`	attack (depends_on_ambiguity)	z0110_del	

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-20261114T223818, 2026-11-14. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 71, prover 289 s, wall 20 min, cost 0.154 USD.

ProVerif: 74 proofs run, 60 decided, 14 without an answer, 215 s, 19 answers recorded.

What was analysed

Protocol	AF training server subscription
Version	19.2.0
Analysed	A trusted AF as the VFL server and the NWDAF as the NF consumer; one fixed TrainEventsSubsc creation/update/Notify/cancellation for an existing training task. The concrete VFL client computation is within the boundary; the event API is not treated as the complete training algorithm; untrusted AF/NEF are excluded.
Not analysed	The symbolic model does not prove the computational security or post-quantum resistance 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. Rel-19 is fixed. The external network attacker and the malicious legitimate NF holding its own credentials are distinguished; TLS authenticated encryption is modelled per endpoint, and a legitimate NF cannot read the TLS session of another NF. 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 target is the specified API sub-flow, not complete 5G security.
Source text	338 numbered passages (198526 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	5.4	Naf_Training Service
main	5.4.1	Service Description
main	5.4.1.1	Overview
main	5.4.1.2	Service Architecture
main	5.4.1.3	Network Functions
main	5.4.1.3.1	Application Function (AF)
main	5.4.1.3.2	NF Service Consumers
main	5.4.2	Service Operations
main	5.4.2.1	Introduction
main	5.4.2.2	Naf_Training_Subscribe
main	5.4.2.2.1	General
main	5.4.2.2.2	Training Subscription Creation
main	5.4.2.2.3	Training Subscription Update

Document	Section	Title
main	5.4.2.3	Naf_Training_Unsubscribe
main	5.4.2.3.1	General
main	5.4.2.3.2	Training Subscription Deletion
main	5.4.2.4	Naf_Training_Notify
main	5.4.2.4.1	General
main	5.4.2.4.2	Training Notification
main	6.3	Naf_Training Service API
main	6.3.1	Introduction
main	6.3.2	Usage of HTTP
main	6.3.2.1	General
main	6.3.2.2	HTTP standard headers
main	6.3.2.2.1	General
main	6.3.2.2.2	Content type
main	6.3.2.3	HTTP custom headers
main	6.3.3	Resources
main	6.3.3.1	Overview
main	6.3.3.2	Resource: Training Subscriptions
main	6.3.3.2.1	Description
main	6.3.3.2.2	Resource Definition
main	6.3.3.2.3	Resource Standard Methods
main	6.3.3.2.3.1	POST
main	6.3.3.2.4	Resource Custom Operations
main	6.3.3.3	Resource: Individual Training Subscription
main	6.3.3.3.1	Description
main	6.3.3.3.2	Resource Definition
main	6.3.3.3.3	Resource Standard Methods
main	6.3.3.3.3.1	GET
main	6.3.3.3.3.2	PUT
main	6.3.3.3.3.3	PATCH
main	6.3.3.3.3.4	DELETE
main	6.3.4	Custom Operations without associated resources
main	6.3.5	Notifications
main	6.3.5.1	General
main	6.3.5.2	Training Notification
main	6.3.5.2.1	Description
main	6.3.5.2.2	Target URI

Document	Section	Title
main	6.3.5.2.3	Standard Methods
main	6.3.5.2.3.1	POST
main	6.3.6	Data Model
main	6.3.6.1	General
main	6.3.6.2	Structured data types
main	6.3.6.2.1	Introduction
main	6.3.6.2.2	Type: TrainEventsSubsc
main	6.3.6.2.3	Type: TrainEventsSubscPatch
main	6.3.6.2.4	Type: EventSubsc
main	6.3.6.2.5	Type: MlModelMonitorInfo
main	6.3.6.2.6	Type: TrainEventsNotif
main	6.3.6.2.7	Type: EventNotif
main	6.3.6.3	Simple data types and enumerations
main	6.3.6.3.1	Introduction
main	6.3.6.3.2	Simple data types
main	6.3.6.4	Data types describing alternative data types or combinations of data types
main	6.3.6.5	Binary data
main	6.3.6.5.1	Binary Data Types
main	6.3.7	Error Handling
main	6.3.7.1	General
main	6.3.7.2	Protocol Errors
main	6.3.7.3	Application Errors
main	6.3.8	Feature negotiation
main	6.3.9	Security
main	6.3.10	HTTP redirection

The purpose of the protocol as the source step located it in the documents: A trusted AF as the VFL server and the NWDAF as the NF consumer; one fixed TrainEventsSubsc creation/update/Notify/cancellation for an existing training task. The concrete VFL client computation is within the boundary; the event API is not treated as the complete training algorithm; untrusted AF/NEF are excluded.

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
Z01	Authorization of every subscription operation	The AF (VFL server) is assured that it accepts the creation, update and deletion of a Training Subscription only from an NF service consumer presenting a valid NRF-issued access token for the AF as audience with the naf-train scope.	Access may be authorized with OAuth2 client credentials, the NRF being the authorization server; the consumer shall obtain a token before consuming the service; the API defines the single scope naf-train. TS 29.500 requires the producer to validate the token, its expiry and its access scope before allowing access to the requested resource.	main:5849, main:5851, main:5855, ts-29-500:4994, s0285, s0286, s0288
Z03	Notification bound to its subscription	The subscribing NF consumer is assured that each notification is delivered to the notifUri registered for its subscription with that subscription's notifCorreId, and that the registration cannot be moved to another consumer.	The AF shall POST the notification to {notifUri}, the value received during the creation/update of the corresponding subscription; notifCorreId is mandatory in TrainEventsSubsc and in TrainEventsNotif.	main:1425, main:5124, s0074, s0213, s0241, s0253
D01	Notification only after a real send	The subscribing NF consumer is assured that every notification it accepts corresponds to a notification the AF actually sent for its subscription, with the same notifCorreId and the same report content.	The AF sends TrainEventsNotif to {notifUri} and the consumer acknowledges with 204 No Content; the protection of the path is left to TLS between the endpoints.	main:1425, main:5124, s0074, s0075, s0253
A03	Matching notification delivery, no replay	The subscribing NF consumer is assured that each notification it accepts matches a distinct notification of the AF for the same subscription and report.	TrainEventsNotif carries the notifCorreId of the corresponding subscription and is answered with 204 No Content; no replay or de-duplication rule is defined.	main:1425, main:5124, s0074, s0075, s0253
A08	Created resource matches the request	The subscribing NF consumer is assured that the 201 Created representation and the Location URI correspond to the TrainEventsSubsc it submitted, and that later notifications use the notifUri and notifCorreId it registered.	The AF answers 201 Created with a TrainEventsSubsc representation and a Location header {apiRoot}/naf-train/<apiVersion>/subscriptions/{subscriptionId}; notifUri and notifCorreId are mandatory in TrainEventsSubsc.	main:1065, main:1067, s0038, s0039, s0134, s0241
R03	One side effect per operation	The AF's Training Subscription store: one logical creation, update or deletion request causes at most one side effect at the AF.	The operations are described one request and one success response at a time; no idempotency key, request identifier or retry rule is defined.	main:1065, s0027, s0030, s0054, s0062

ID	Name	About	What the documents say	Where
S05	Nothing after deletion	The subscribing NF consumer is assured that after its DELETE is answered 204 No Content the AF sends no further notification for that subscription and the identifier cannot be used again to obtain notifications.	Deletion: DELETE on the Individual Training Subscription URI answered with 204 No Content; nothing is said about notifications in flight at deletion.	main:1148, s0054, s0061, s0062, s0074
Z03.8	notifUri and notifCorreId binding (Q1)	The subscriber (Q1): every notification goes to the notifUri registered in that subscription and carries that subscription's notifCorreId, so a consumer recognises the notifications of its own subscriptions.	As stated for the notification procedure: {notifUri} is the value from the creation/update of the corresponding subscription and notifCorreId is echoed in TrainEventsNotif.	main:1425, main:5124, s0074, s0213, s0241, s0253
U01	Report bound to its training task (Q2)	The subscriber (Q2): every training report it accepts corresponds to a notification the AF sent for the same training task (event, vflCorrId) with one report category.	EventNotif carries the subscribed event, vflCorrId and the training indication or termination cause; TrainEventsNotif carries the notifCorreId; no rule on vflCorrId is defined.	main:5124, s0253, s0256
Z01.10	No cross-consumer subscription control (Q3)	The subscribing NF consumer (Q3): another NF service consumer holding its own valid naf-train token cannot update or delete its subscription, nor move its notifications elsewhere, without an authorization specific to that subscription.	Update and deletion are addressed to the Individual Training Subscription URI and authorised only by the token; the API defines one service-level scope and no scope at resource or operation level, while TS 29.500 allows an operation- and resource-level scope where a specific authorization is required.	main:5855, ts-29-500:4996, s0045, s0061, s0288

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: 33. Not taken: 365. Which paragraphs are listed follows from the layout of the standard; where each went is the analyst's judgement.

Paragraphs	Where	Taken up as	Why not
P001 to P002	rfc6749:2778-2786	No goal stated	Introductory list of OAuth2 threats; the paragraphs that follow are taken up individually.
P003 to P006	rfc6749:2790-2823	Z01, `legal_nf`	
P007 to P010	rfc6749:2827-2850	Z01.10, `legal_nf`	

Paragraphs	Where	Taken up as	Why not
P011 to P015	rfc6749:2859-2881	Z01, Z01.10, `legal_nf`	
P016 to P033	rfc6749:2885-3003	No goal stated	Refresh tokens, authorization codes, redirection URIs, resource owner password credentials: this API uses the client credentials grant only.
P034 to P035	rfc6749:3013-3019	Z03, D01	
P036	rfc6749:3023-3028	Z01, D01	
P037 to P039	rfc6749:3032-3041	Z01	
P040 to P052	rfc6749:3045-3150	Outside the scope	Phishing, CSRF, clickjacking and code injection are browser and user-agent concerns; the SBI has no user agent.
P053 to P054	rfc6749:3154-3167	Outside the scope	Open redirectors concern the authorization-request redirection URI; the redirects of this API are outside the modelled procedure.
P055 to P062	rfc6749:3170-3210	No goal stated	Token misuse in the implicit grant; the implicit grant is not used and there is no resource owner.
P063	ts-33-501:2491-2491	Outside the scope	Bidding down of ciphering and integrity algorithms; primitive strength is excluded by the scope.
P064 to P072	ts-33-501:2495-2511	Outside the scope	UE and network authentication and authorization in 5GS access, not this API procedure.
P073 to P078	ts-33-501:2515-2529	Outside the scope	Keys and general access-network security requirements of the 5G access stratum.
P079 to P125	ts-33-501:2533-2631	Outside the scope	Air-interface signalling protection, secure storage of subscription credentials and subscriber privacy; none is part of an SBI service procedure.
P126 to P198	ts-33-501:2637-2811	Outside the scope	gNB, ng-eNB, F1/E1 and SEAF requirements: RAN-side security.
P199 to P213	ts-33-501:2819-2853	Outside the scope	Requirements on UDM, AUSF, UDR and SIDF; the NRF is the authorization server here.
P214	ts-33-501:2859-2859	Z01.10	
P215 to P220	ts-33-501:2865-2875	Z01, `legal_nf`	
P221 to P227	ts-33-501:2879-2891	Z01	
P228 to P234	ts-33-501:2895-2907	Outside the scope	NEF security requirements for its exposure function; the NEF appears here only as a consumer.
P235 to P241	ts-33-501:2911-2923	Outside the scope	SCP requirements: the SCP deployment variant is outside the modelled procedure.
P242 to P254	ts-33-501:2929-2953	Outside the scope	SEPP and interconnect security: roaming is outside the scope.
P255 to P303	ts-33-501:2957-3055	Outside the scope	SEPP requirements, N32-f protection and roaming intermediaries: interconnect.
P304 to P312	ts-33-501:3059-3075	Outside the scope	Attribute protection over N32-f, including the authorization token: interconnect.
P313 to P314	ts-33-501:3079-3081	Outside the scope	IPUPS forwarding rules for GTP-U.

Paragraphs	Where	Taken up as	Why not
P315 to P318	ts-33-501:3085-3091	No goal stated	Originating PLMN header handling, deployment specific; it states no property of this API.
P319 to P327	ts-33-501:3097-3115	Outside the scope	Signalling monitoring configuration and streaming.
P328 to P337	ts-33-501:3121-3141	Outside the scope	Security visibility and configurability requirements on operators.
P338 to P369	ts-33-501:3149-3215	Outside the scope	Algorithm identifiers and selection: primitive strength is excluded.
P370 to P375	ts-33-501:3219-3231	Outside the scope	Requirements on 5G-RG and NSSAAF.
P376 to P380	ts-33-501:6540-6548	Outside the scope	Security requirements and procedures on N3.
P381 to P398	ts-33-501:9656-11901	Outside the scope	EAP-TLS subscription identifier privacy, IAB, MBS and NEF annexes: none concerns this API.

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: 19. Without: 1. 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	Outside the scope	The AF is fixed by the task as a trusted VFL server; a dishonest AF knows every subscription and every training report by definition, and the scope excludes an untrusted AF.
S02	NWDAF	party	`legal_nf`	
S03	NEF	party	`legal_nf`	
S04	NRF	party	`legal_nf`	
S05	NWDAF/NEF	party	`legal_nf`	
S06	OAuth2 access token	value	`legal_nf`	
S07	Training Subscription resource	value	`legal_nf`	
S08	notifUri	value	`legal_nf`	
S09	notifCorreId	value	`legal_nf`	
S10	subscriptionId	value	`legal_nf`	
S11	training event report	value	`legal_nf`	
S12	TrainEventsNotif content	value	`legal_nf`	
S13	eventNotifs	value	`legal_nf`	
S14	TrainEventsSubsc parameters	value	`legal_nf`	
S15	Location URI	value	`legal_nf`	
S16	event	value	`legal_nf`	

ID	Party or value	Kind	Attacker	Why none
S17	vflCorrId	value	`legal_nf`	
S18	termCause	value	`legal_nf`	
S19	vflNotOnTimeInd	value	`legal_nf`	
S20	training report content	value	`legal_nf`	

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		
`legal_nf`	A legitimate NF service consumer (NWDAF or NEF) with its own valid NRF-issued access token for the naf-train scope, acting from its own TLS endpoints, and with the public resource structure {apiRoot}/naf-train/v1/subscriptions/{subscriptionId}, so it can address another consumer's Individual Training Subscription resource. It cannot read another NF's TLS session and cannot forge a token for another consumer.	NWDAF and NEF are the known consumer NFs and hold their own credentials; the API defines one service-level scope, so any consumer's token carries the same authority; the Location URI structure of an Individual Training Subscription is public and nothing requires the subscriptionId to be secret.	main:5849, main:5855, ts-29-500:4996, s0010, s0011, s0012, s0134, s0288

ID	`network`	`legal_nf`
Z01	Holds	Holds
Z03		Fails
D01	Holds	
A03	Holds	
A08	Holds	
R03		No answer
S05		No answer
Z03.8	Holds	Holds
U01	Holds	
Z01.10		Fails

Source and targets

Source: 338 units, 198526 bytes, frozen in source-delivery-01.json. Targets: 8 approved (Z01, Z03, D01, A03, A08, R03, S05, U01); 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-eae0246ed96fbe0bdc5d0a5395bb

This step as it reports itself: 1 requests (1 failed), 0.000 USD, 99457 tokens in and 4220 out, 257 s.

Notes on the prover and the model

- Z01: 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
- S05: answered in ProVerif, and the analyst names changing_state for this target
- Z01.10: 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.

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 Training Subscription record: subscriptionId, notifUri, notifCorreId and the subscribed events	the AF (VFL server), one record per subscription (bounded.pv unrolls it to three steps: creation, update, deletion)	created by the POST (the AF allocates subscriptionId), its notifUri and notifCorreId replaced by a PUT or PATCH on the resource, and removed by a DELETE.	on every request that names the Individual Training Subscription URI: the AF looks the resource up and checks the token of the request; the token check is the only comparison and there is no comparison between the requester and the consumer that created the record.	s0038, s0039, s0045, s0061, s0288
the callback URI notifUri	the AF, per subscription	replaced by the notifUri of a PUT or PATCH on the subscription.	used, not compared: the AF posts each notification to the notifUri the record holds and the specification requires no check that the URI belongs to the consumer.	s0045, s0074, s0213
the notification correlation identifier notifCorreId	the AF per subscription and the consumer per subscription	replaced by the notifCorreId of a PUT or PATCH (optional in TrainEventsSubscPatch).	compared by the consumer with the value it registered, in every notification it accepts.	s0241, s0244, s0253

pattern	result	target	shown by ProVerif	steps
race_between_sessions	candidate	Z01.10	yes: ProVerif falsifies `z0110_del` on bounded.pv	The NWDAF creates a subscription and the AF stores its record (s0038, s0039). / While that record is still held, a second NF with its own valid token sends PUT on the same resource and the AF replaces the notifUri and the notifCorreId stored in it (s0045, s0048) - ProVerif finds this on bounded.pv and on model.pv. / The AF then notifies at the endpoint of the second NF (s0074), so the reports of the NWDAF's subscription leave it (this is the confirmed trace of z0110_upd on model.pv and bounded.pv). / After the update the second NF sends DELETE on the same resource and the AF deletes the record (s0061, s0062), which ProVerif confirms on bounded.pv in the trace of z0110_del.
desynchronisation	candidate	Z01.10	yes: ProVerif falsifies `z0110_upd` on bounded.pv	The NWDAF registers notifUri_A and notifCorreId_A; the AF stores them (s0038, s0039). / A second NF replaces them with its own values by PUT on the same resource (s0045, s0048). / The AF holds notifUri_B/notifCorreId_B while the NWDAF still expects its own, and the reports go to the second NF (s0074). / Nothing in the procedure tells the NWDAF that its registration changed: there is no notification of a subscription change and the consumer does not read the resource back (GET exists in clause 6.3.3.3.3.1 but is not part of this procedure), so the two sides stay apart.
replay	candidate		no: not tried with ProVerif	The NF service consumer sends the POST with TrainEventsSubsc(notifUri, notifCorreId) (s0038) and does not receive the 201 Created, for instance because the response is lost. / The consumer sends the same POST again, as an HTTP client does after a timeout. / The AF has nothing with which to recognise the second request: it creates a second Individual Training Subscription with a new subscriptionId, the same notifCorreId and the same notifUri, and answers 201 again (s0039, clause 6.3.3.2.3.1). / The consumer holds one logical subscription while the AF holds two, and both deliver the same training reports to the same endpoint.
repeat_one_time_action	nothing found	S05		
update_before_check	nothing found			
no_rollback_on_failure	nothing found			
distinguishable_failure	nothing found			
wrap_or_window	nothing found			

What the analyst named for targets:

step	targets	features	reason
47	Z01, Z03, D01, A03, A08, Z03.8, U01, Z01.10	plain_primitives, many_sessions	The access token is a MAC over (subject, audience, scope) under a key shared between the NRF and the AF, and the notification path is a private channel per endpoint; no other algebra is used. The properties are asked for any number of sessions of the consumer and of the AF, which ProVerif covers, but the AF is a chain of one subscription per process instance rather than a table, so the model does not keep a subscription record across arbitrary later requests.

step	targets	features	reason
47	R03, S05	changing_state	Both targets rest on the AF's Training Subscription record: it is created by the POST, its notifUri and notifCorreId are replaced by the PUT/PATCH, and it is removed by the DELETE. ProVerif's store for such a value only grows (a table) or ends in inconclusive (a cell on a private channel), so the removal and the replacement could not be modelled for any number of requests; the answers recorded for these two are on model.pv only and are undecided, the bounded unrolling of the AF and the state review not having been produced in this session.
65	Z01, Z01.10	changing_state	The deletion claim of Z01 and both claims of Z01.10 rest on the AF's Training Subscription record, which is created, replaced and removed; they are answered on bounded.pv, where that record is unrolled to three steps (creation, update, deletion) and the AF that keeps it is a chain of process macros. What the bound leaves out is a fourth or later request on the same record and any interleaving of two subscriptions.

The analyst's closing note

Two authorization properties fail and the notification properties hold. Proved: the AF applies a creation and an update only with a token the NRF issued for the requesting consumer, and (on the unrolled model) the same for a deletion; every notification goes to the notifUri and with the notifCorreId registered in its subscription, for any number of sessions; the consumer accepts only notifications the AF sent, each matching a distinct send; the 201 Created echoes the TrainEventsSubsc it submitted. Attacked: a second legitimate NF consumer, with nothing but its own naf-train token and the Individual Training Subscription URI of another consumer, sends PUT on that subscription, the AF applies it because it authorizes on the service-level scope alone, and the victim's training reports are then delivered to the second NF; with the same token it deletes the subscription. That is the finding, one weakness with two attack traces, novelty unclear (the documents state the single service scope and never its consequence) and depends_on_ambiguity for the assumption that the subscription URI is knowable. Open: whether a retried creation produces a second subscription (answered only at reachability) and what happens to a notification in flight when the deletion is applied. Limits: everything about the subscription record is answered for three unrolled steps only, no proof carries a control variant, and the recommended fix - bind the subscription to the token subject of its creator and check it on PUT, PATCH and DELETE - was not verified on a model. Tamarin+ Agent, which models the record and the algebra as they are, would settle the retry, the post-deletion ordering and check the fix.