Security analysis of NWDAF model training event subscription and result notification

This report in another language: 中文.

19.8.0 · ProVerif+ Agent report

Documents	5G System; Network Data Analytics Services; Stage 3 (protocol definition)
Properties examined	11: 8 hold, 3 fail (5 attacks)
Findings	2

Contents

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

1 Protocol and scope

The Nnwdaf_MLModelTraining service lets an NF service consumer subscribe to machine-learning model training events at the NWDAF that contains the Model Training Logical Function, and lets that NWDAF post the training results back to the consumer. Three parties appear: the consumer, the NWDAF with MTLF, and the NRF, which acts as the OAuth2 authorisation server for the nnwdaf-mlmodeltraining scope.

The consumer holds its service token, a notification correlation identifier it invents and the URI where it wants notifications; the NWDAF holds the created subscription with its subscription identifier. The exchange is a subscription with a created-subscription response, one or more notifications carrying the model information or a termination request, and an unsubscribe.

When the exchange ends, the consumer knows that each notification it stored belongs to a subscription it created itself, that it really came from the NWDAF unaltered, and that the NWDAF created the subscription only after checking a token of the service scope. It does not learn when a notification is a repeat or how a termination relates to the notifications around it.

The parties:

- **NF service consumer**: subscribes, stores the notifications it accepts. Holds: token for scope nnwdaf-mlmodeltraining, notifCorreId, notifUri, mLEvent.
- **NWDAF (MTLF)**: creates and stores subscriptions, trains, notifies. Holds: OAuth2 token verification key, stored subscriptions.
- **NRF**: OAuth2 authorisation server. Holds: client credentials of the consumer.

Figure 1 shows the exchange as the documents specify it.



Figure 1. The protocol as specified.

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

2 Findings

The analysis has 2 findings: what follows is not stated in the documents. For 1 of them it is unclear whether the documents state it.

Within the scope analysed the correlation checks hold: an accepted notification always belongs to a subscription the consumer itself created, with matching notifCorreId, notifUri and mLEvent, and it was really sent by the NWDAF unaltered, because the consumer correlates on identifiers it chose and the SBA transport protects the messages. Two gaps remain and both are findings. Nothing identifies a notification instance or orders the notification stream, so a withheld or repeated notification is accepted again, also after the termination request and after the unsubscribe. And no field says what kind of notification it is, so a termination that also carries the final aggregated model is read as an ordinary result. The token checks of the subscribe and unsubscribe operations hold. The results rest on a model that treats the SBA transport as an authenticated encrypted channel and models a single consumer, a single provider and at most two notifications per subscription.

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 termination notification carrying the final model is read as an ordinary result

Kind	Weakness of the protocol as specified
New	Unclear: the documents state it in part, or could not be searched
Value	medium
Properties	U01 (section 3)
Source text	s0128, s0131, s0133, s0321, s0337
Shown by	ProVerif attack queries `u01_cls`, `x01_both`; Figure 2

What happens. The notification has no type field. The consumer must decide from the fields present whether it received model information or a termination request, and the specification makes termTrainReq and mLModelInfos well formed together: NOTE 1 requires at least one of delayEventNotif, mLModelInfos and termTrainReq and makes only delayEventNotif mutually exclusive with the other two. A consumer that looks at mLModelInfos first therefore classifies a termination that also carries the final aggregated model as a training result and never learns that the subscription ended. The same shape appears elsewhere in the service, where TrainingUnsubscribeInfo carries termCause FL_FINISHED together with mLModelInfos.

What the attacker needs. No attacker is needed: the NWDAF itself sends this well-formed notification when it finishes a training whose final model it reports. An attacker on the network needs only to reorder the two notifications to reach the same state.

What the attacker gains. The consumer keeps training on a subscription that has ended, waits for notifications that will never come, or treats a final model as an intermediate result, and no field in the stored notification lets it recover the distinction.

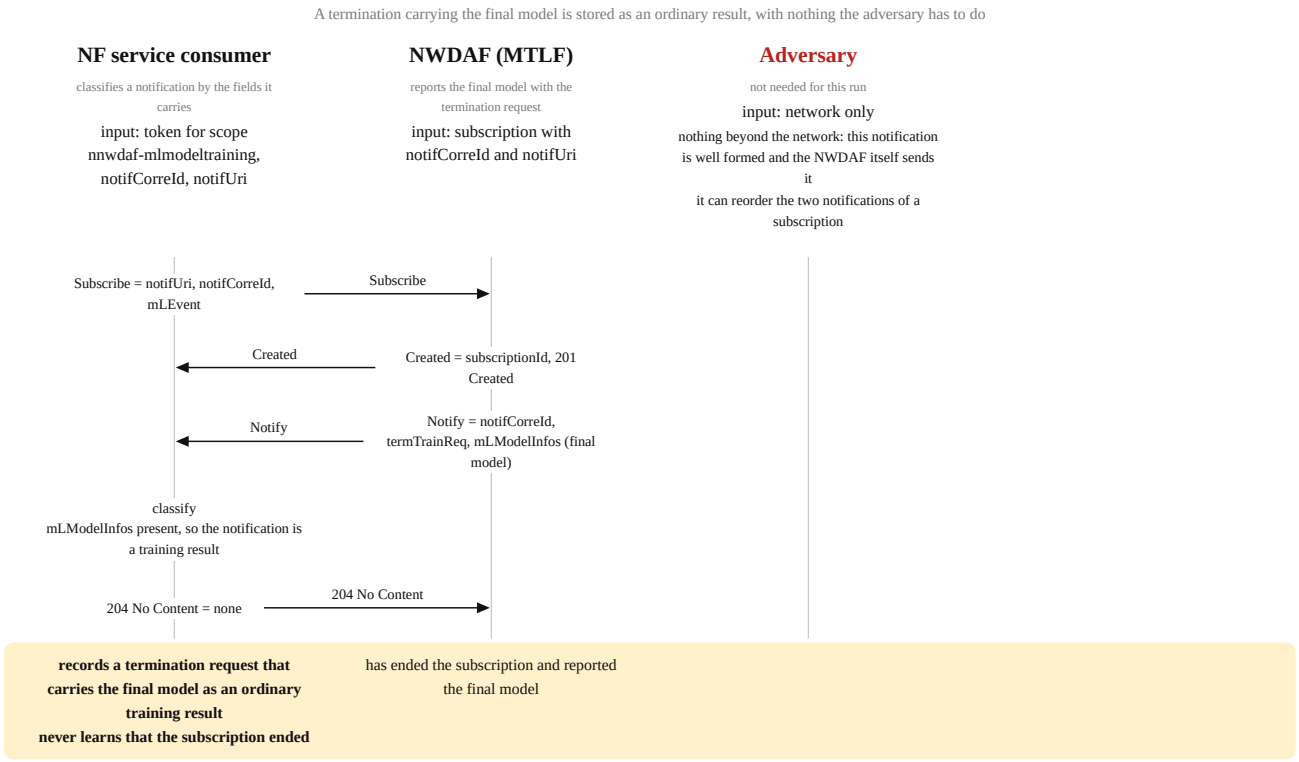


Figure 2. A termination carrying the final model is stored as an ordinary result, with nothing the adversary has to do. Property U01; attack queries u01_cls, x01_both.

Proposed change. Give NwdafMLModelTrainNotif an explicit notification type, or state that termTrainReq is decisive when present so that a notification carrying it is a termination whatever else it carries, and say in NOTE 1 for which training kinds the combination termTrainReq plus mLModelInfos is meant to occur.

Check of the change with ProVerif. Not checked: Checking it needs a notification type that every notification carries, which is a change of the message and of all its producers; the model can only check a consumer-side precedence rule, not that the notifications become distinguishable.

2.2 Repeated and withheld notifications are accepted again, also after termination and unsubscribe

Kind	Weakness of the protocol as specified
New	Yes: the documents do not state it
Value	medium
Properties	R02, S05 (section 3)
Source text	s0128, s0129, s0138, s0321
Shown by	ProVerif attack queries `r02`, `s05_term`, `s05_unsub`; Figures 3 and 4

What happens. Nothing in the notification identifies the notification instance or its position in the stream: notifCorreId identifies the subscription and roundInd is optional and unconstrained. A consumer that is offered a notification it has already accepted stores it again. A notification withheld by the adversary and released after the termination request, or after the unsubscribe has been answered, is accepted as a fresh notification, although termTrainReq promises that no further notifications for that subscription will be provided.

What the attacker needs. The adversary needs only the network: it reads a notification it cannot forge, keeps it, and delivers it again later or twice. It needs no key and breaks neither the transport protection nor the token check.

What the attacker gains. The consumer's stored record of a subscription can contain the same result twice, and it can contain a result that arrived after it was told the subscription ended or after it unsubscribed, so analytics computed from the stored notifications may count a training result twice or after the fact.

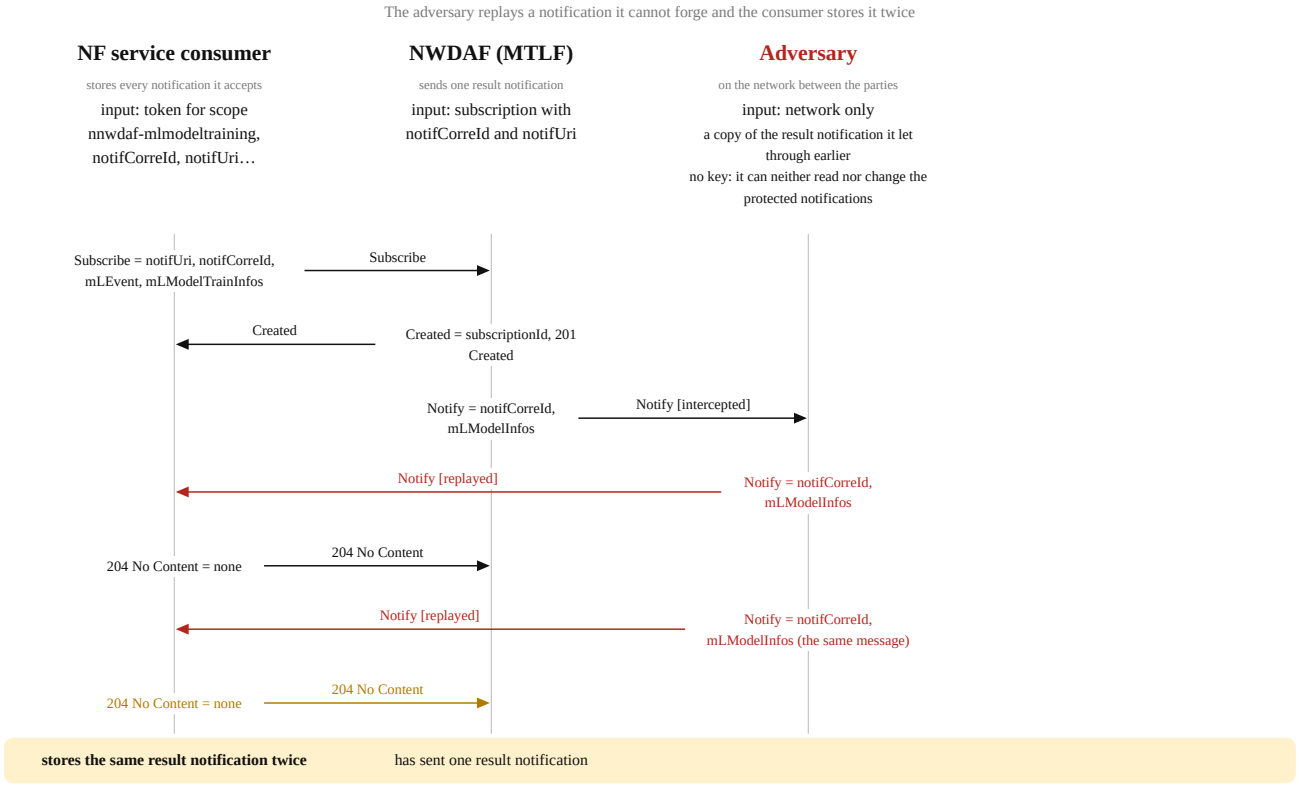


Figure 3. The adversary replays a notification it cannot forge and the consumer stores it twice. Property R02; attack query r02.

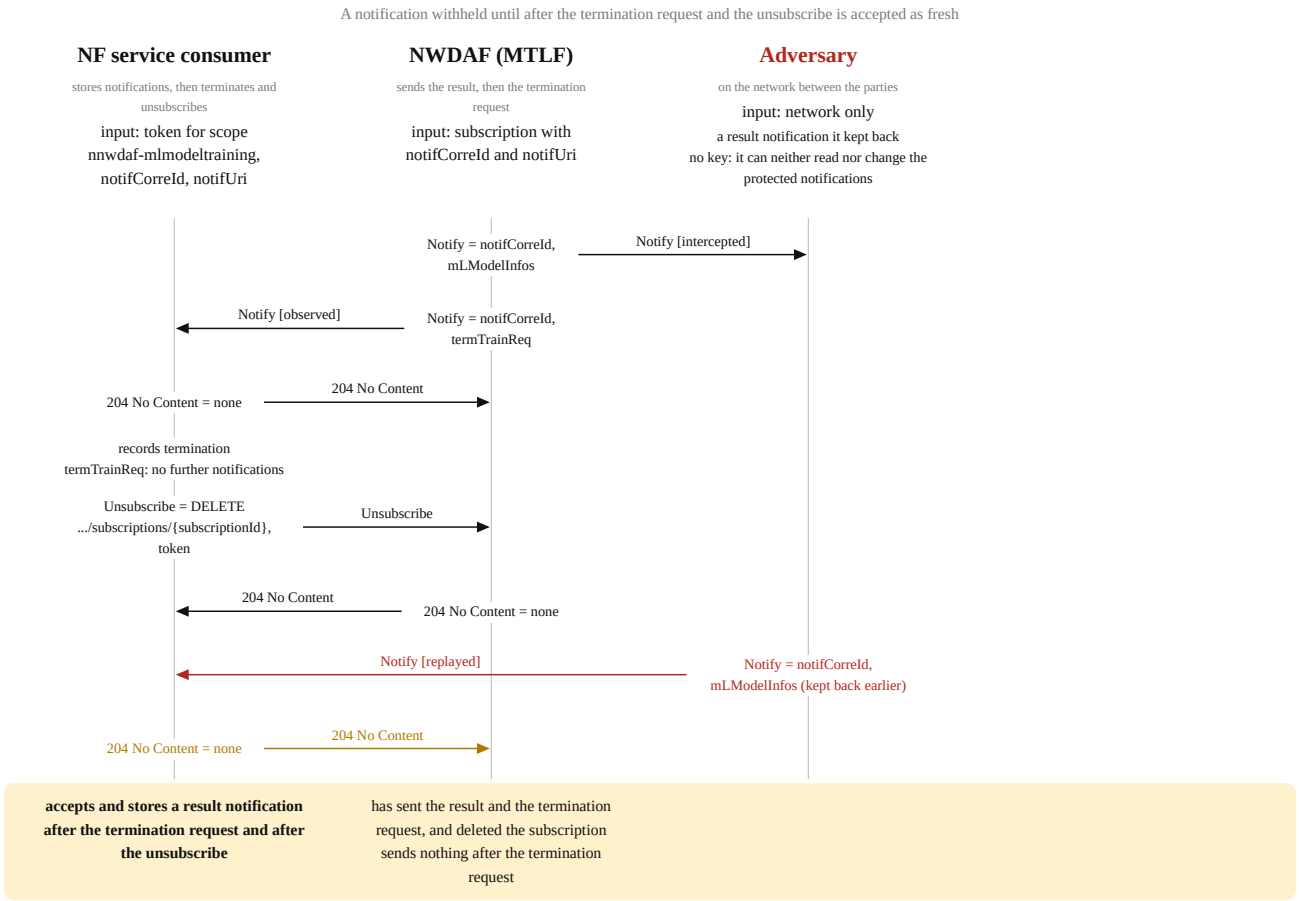


Figure 4. A notification withheld until after the termination request and the unsubscribe is accepted as fresh. Property S05; attack queries s05_term, s05_unsub.

Proposed change. Carry a per-subscription notification sequence number in NwdafMLModelTrainNotif, make roundInd mandatory where rounds exist, and require the consumer to store and check that a notification's number is greater than the last one it stored for that subscription, dropping anything else.

Check of the change with ProVerif. Not checked: The check needs a value that changes per subscription and is compared at every notification, which this model does not have: ProVerif forgets how often and in what order a step ran, so the property would have to be shown on an unrolled bounded model or with Tamarin+ Agent.

No figure was given for these attacks of a finding: R02: s05_term, R02: s05_unsub, S05: r02.

3 Properties examined

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

ID	Property	Compromise assumed	Result	Value	In the documents
A03	**notification tied to own subscription.** Every time the consumer accepts a training event notification, there is a subscription genuinely initiated by that consumer whose notifCorreId and notifUri match it.	no compromise	Holds	low	yes

ID	Property	Compromise assumed	Result	Value	In the documents
A08	**notification context matches subscription.** For every notification the consumer accepts, its context fields such as notifCorreId, event identifier and round all correspond to a subscription previously approved for the same consumer and to its event description.	no compromise	Holds	low	unclear
D01	**notification authenticity and integrity.** Every training event notification the consumer accepts corresponds to a notification genuinely sent by the NWDAF for that subscription, and the notification payload is the same as when it was sent.	no compromise	Holds	medium	unclear
R02	**one acceptance per notification instance.** For the same logical notification instance of the same subscription, the consumer accepts it at most once.	no compromise	Fails (1 attack)	medium	no
S05	**nothing accepted after termination.** After a termination notification or an unsubscribe takes effect, the consumer no longer accepts new training event notifications associated with that terminated subscription.	no compromise	Fails (2 attacks)	medium	unclear
Z01	**service operations covered by token.** Every time a subscribe, update, unsubscribe or notify operation is accepted, there is a valid token issued by the authorization server that covers the nnwdaf-mlmodeltraining scope.	no compromise	Holds	medium	yes
Z03	**token bound to its subscription.** A token held by a consumer can be used only for subscriptions and operations it initiated itself, and cannot be used for another consumer's subscriptions or notifications.	no compromise	Holds	medium	unclear
A03.8	**notifCorreId equals own subscription.** For every notification the consumer accepts, its notifCorreId equals the notifCorreId in the subscription the consumer genuinely initiated.	no compromise	Holds	low	yes
A08.9	**no mixing of subscriptions.** For every notification the consumer accepts, its subscription association identifier and event parameters all correspond to the same consumer's own subscription and do not come from another subscription.	no compromise	Holds	medium	unclear
U01	**termination distinguished from result.** For every notification the consumer accepts, termTrainReq and mLModelInfos do not appear at the same time, and the consumer classifies it as a termination event or a result event according to the actual fields.	no compromise	Fails (2 attacks)	medium	unclear
X01	**notification carries at least one report field.** Every notification accepted by the consumer carries at least one of delayEventNotif, mLModelInfos or termTrainReq, and never carries delayEventNotif together with mLModelInfos or with termTrainReq.	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: the notification's notifCorreId is bound to the requested job	Yes, but only by identifier equality: the notification's notifCorreId must be the value the consumer chose for that subscription, and because the message is protected and the value is fresh, the consumer can bind a notification to the job it submitted; nothing else in the notification ties it to the request.	A03, A03.8, D01
Q2: training result/failure notifications are not mixed into other subscriptions	A training result or a termination is not delivered to another subscription's handler: the correlation identifier and the subscription identifier both have to match the consumer's own subscription, so a result cannot arrive under the wrong subscription.	A08.9, A08, A03

Question	Answer	Rests on
Q3: termination notifications and training result notifications are distinguished as different events	No, they are not reliably distinct: a termination may also carry the final aggregated model, and since the notification has no type field a consumer that reads model information first treats the termination as an ordinary result. A termination with no model information is distinguishable, which is the only case the model shows to be safe.	U01

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 SBA transport is modelled as authenticated encryption under one long-term key per pair of parties (consumer to NWDAF, consumer to NRF); TLS details, SEPP/SCP paths and the transport's own replay handling are not modelled, and no transport or token key is ever compromised.
- One consumer identity, one NWDAF and one token key: cross-consumer use of a subscription identifier is not expressible, so the token-scope answer speaks only about the network attacker.
- At most two notifications per subscription are modelled (one result, one termination request), and round numbers, statusReport and the multi-round training state are left out, so nothing is said about the order of notifications within a round or across many rounds.
- The NWDAF deletes no stored subscription on unsubscribe: the model shows only that the honest provider sends nothing more, so post-unsubscribe behaviour of a dishonest provider is not covered.
- The OAuth2 token is modelled as a MAC over the service scope and consumer identity; token lifetimes, audiences, per-resource scopes and NRF-side authorisation policy are out of scope.
- The update and partial-update operations (PUT, PATCH), the partial-success path with failEventReports, the delay notification and the unsubscribe-info operation are not modelled.
- The correctness of the trained model and any confidentiality of the model information are not modelled and are outside the task.

What the host records:

- Proved without a control query (Appendix C): A03: a03, A08: a03, D01: d01, Z01: z01_unsub, Z03: z03, A03.8: a03, A08.9: a08_9, X01: x01_empty. A control is a variant of the property that must fail; without one, nothing shows that the property as written can fail in this model.

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

Appendix A: Reading of each property

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

A03: notification tied to own subscription

Every time the consumer accepts a training event notification, there is a subscription genuinely initiated by that consumer whose notifCorreId and notifUri match it.

Every notification the consumer stores belongs to a subscription request the consumer itself sent, with the same notifCorreId, the same mLEvent and its own notifUri. This holds against the network attacker because the consumer checks identifiers it chose itself and the transport keeps the message intact.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: The specification itself says the notification correlation identifier shall be set to the same value as in the subscription, so the result confirms a stated rule under an explicit transport assumption rather than finding something new.
- Stated in the documents: yes. Where the documents say it: main:27562
- Rests on: the SBA transport is modelled as authenticated encryption under a key shared by the consumer and the NWDAF, so the attacker cannot forge or change a notification; the consumer is modelled as checking notifCorreId, notifUri and mLEvent against its own subscription before accepting; one consumer identity and at most two notifications per subscription
- Next step: Nothing; a two-consumer model would be needed only for the cross-subscription reading of the token scope.

A08: notification context matches subscription

For every notification the consumer accepts, its context fields such as notifCorreId, event identifier and round all correspond to a subscription previously approved for the same consumer and to its event description.

The event identifier of an accepted notification is an event the consumer subscribed to under that same correlation identifier. The round and status context fields are not modelled, so nothing is said about whether the round of a notification matches the subscription's expectations.

- Result: Holds; answers what is asked: partly; usable with caveats
- Value low: The documents say the notification reports the model information for the subscription identified by notifCorreId but never state a correspondence between the subscribed event identifier and the notified one, and the round context is outside the modelled scope.
- Stated in the documents: unclear
- Rests on: only the event identifier mLEvent is modelled as part of the notification context; roundInd and statusReport are absent; the transport protects the notification from being changed
- Open: the round context of notifications (roundInd, statusReport)
- Next step: Unfold the multi-round training state in a bounded model and ask whether the round of a notification can be accepted out of order.

A03.8: notifCorreId equals own subscription

For every notification the consumer accepts, its notifCorreId equals the notifCorreId in the subscription the consumer genuinely initiated.

The correlation identifier of every accepted notification equals the one of the subscription request the consumer really sent, and no other party can produce it because it is chosen freshly by the consumer and travels only inside the protected messages.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: The value equality is exactly what the attribute description prescribes; the proof adds only the transport assumption that makes it unguessable.
- Stated in the documents: yes. Where the documents say it: main:27562
- Rests on: the transport credential is never compromised; the consumer checks the identifier before accepting
- Next step: Nothing.

A08.9: no mixing of subscriptions

For every notification the consumer accepts, its subscription association identifier and event parameters all correspond to the same consumer's own subscription and do not come from another subscription.

A notification is not mixed in from another subscription: the subscription identifier it carries was assigned by the NWDAF for a subscription created under the same correlation identifier, so a consumer holding several subscriptions keeps them apart by the identifiers rather than by any cryptographic binding.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The documents define notifCorreId as identifying the subscription a notification relates to but say nothing about a cryptographic binding of a notification to its subscription.
- Stated in the documents: unclear
- Rests on: the binding is by identifier equality inside the protected messages, not by a per-subscription key or signature; one consumer identity, so two subscriptions of different consumers are not modelled
- Open: mixing between subscriptions of two different consumers
- Next step: A model with two consumer identities and a shared notification endpoint would settle the cross-consumer reading.

D01: notification authenticity and integrity

Every training event notification the consumer accepts corresponds to a notification that the NWDAF really sent for that subscription, and the notification payload is the same as when it was sent.

Every notification the consumer receives was really sent by the NWDAF for that subscription and its payload was not altered; the guarantee comes entirely from the transport credential, since the notification carries no signature or MAC of its own.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: The 29.500 and 33.501 documents require TLS at the transport layer for service-based interfaces, which is what the model assumes, but neither they nor the service specification state notification authenticity as a property, and TS 33.501/29.500 are outside the selected source units.
- Stated in the documents: unclear

- Rests on: the whole protection is the modelled SBA transport; without it the payload is unauthenticated; the NWDAF is honest about the content of the model information
- Open: protection when indirect communication through SEPP or SCP terminates the transport
- Next step: Model the SEPP/SCP hop to see whether the notification is protected end to end.

R02: one acceptance per notification instance

For the same logical notification instance of the same subscription, the consumer accepts it at most once.

A notification that is delivered twice is stored twice. Nothing in the notification distinguishes two instances of it, so the consumer has no way to notice the repetition, and the attack needs nothing but the network.

- Result: Fails (1 attack); answers what is asked: yes; usable with caveats
- Value medium: I searched the documents for replay, duplicate notification and ordering rules and found no statement about repetition or de-duplication of notifications; the only protection mentioned is TLS at the transport layer, which does not order two notification requests.
- Stated in the documents: no
- Rests on: the transport is assumed not to reject a repeated application message; the source says nothing about notification repetition; the consumer stores every notification it accepts without de-duplication
- Open: whether a real deployment's transport rejects a repeated request
- Next step: State a notification sequence number in the type and check it at the consumer, then re-run the property.

S05: nothing accepted after termination

After a termination notification or an unsubscribe takes effect, the consumer no longer accepts new training event notifications associated with that terminated subscription.

The promise that a termination request ends the notifications, and that an unsubscribe ends them, is not enforced at the consumer: a notification withheld by the adversary and released afterwards is still accepted and stored, so the stored record can hold a result that arrived after the end of the subscription.

- Result: Fails (2 attacks); answers what is asked: yes; usable with caveats
- Value medium: termTrainReq states that no further notifications related to the subscription will be provided, and the unsubscribe deletes the subscription, but the documents say nothing about notifications already in flight or about how a consumer recognises one; the weakness is in what the text does not say.
- Stated in the documents: unclear
- Rests on: the consumer keeps no state that marks a subscription as ended, which is what the source leaves open; the NWDAF deletes nothing in the model and simply sends nothing more; the transport is assumed not to reject a re-delivered notification
- Open: the behaviour of a dishonest NWDAF that keeps notifying after a termination
- Next step: Add a per-subscription end marker or sequence number and require the consumer to drop anything that follows it.

Z01: service operations covered by token

Every subscribe, update, unsubscribe or notify operation that is accepted has a valid token issued by the authorisation server that covers the nnwdaf-mlmodeltraining scope.

The NWDAF creates a subscription and accepts an unsubscribe only after checking a token that the NRF issued for the nnwdaf-mlmodeltraining scope, and the check is what the property rests on: with the check removed an acceptance happens that never verified a token. The notification direction is not covered, since the specification defines no token for the Notify operation, and the update operations are not modelled.

- Result: Holds; answers what is asked: partly; usable with caveats
- Value medium: The security section states that access may be authorised with OAuth2 client credentials and that the consumer shall obtain a token before consuming the service; the result restates that rule for subscribe and unsubscribe and makes explicit that notifications carry none.
- Stated in the documents: yes. Where the documents say it: main:27946, main:27948
- Rests on: the token is modelled as a MAC over the service scope and the consumer identity with a key held by the NRF and the NWDAF; the Notify operation is modelled without any token, as the source defines none; the update and partial-update operations are not modelled
- Open: whether the Notify operation is meant to be token-authorised; the update and partial-update operations
- Next step: Read the API access-control clauses of TS 29.500 for the Notify direction.

Z03: token bound to its subscription

A token held by one consumer can be used only for the subscriptions and operations that consumer itself initiated, and cannot be used for another consumer's subscriptions or notifications.

An operation on a subscription is accepted only for a subscription the same consumer established, because the identifiers travel only inside the protected exchange. The token is bound to the service, not to a subscription, so any authorised consumer that learns a subscription identifier could use its own token on it; with a single consumer in the model, that reading is answered only for the attacker on the network.

- Result: Holds; answers what is asked: partly; usable with caveats
- Value medium: The documents state the single service-wide scope and the absence of resource-level scopes, which is the cause of the caveat, but they never say what a consumer holding a known subscription identifier may do with it.
- Stated in the documents: unclear
- Rests on: a single consumer identity, so cross-consumer use is not expressible; the subscription identifier is unguessable and travels only inside the protected messages; the token carries only the service scope, as the specification says
- Open: cross-consumer use of a subscription identifier
- Next step: Model two consumers with valid tokens for the same scope and ask whether one can unsubscribe or modify the other's subscription once it knows the identifier.

U01: termination distinguished from result

Every notification the consumer accepts does not carry termTrainReq and mLModelInfos at the same time, and the consumer classifies it as a termination event or a result event according to the actual fields.

The consumer cannot reliably tell a termination from a training result. A notification may carry termTrainReq together with mLModelInfos, and a consumer that classifies by the presence of model information reads such a termination as an ordinary result and never learns that the subscription ended.

- Result: Fails (2 attacks); answers what is asked: yes; usable with caveats

- Value medium: NOTE 1 of the notification type states the field rules and leaves termTrainReq and mLModelInfos compatible, and TrainingUnsubscribeInfo shows the same combination elsewhere, but the documents never say how a consumer is to tell a termination from a result, so the consequence is not stated even though the cause is.
- Stated in the documents: unclear
- Rests on: the consumer's classification rule is assumed to test mLModelInfos before termTrainReq, which is one of the two possible readings of the field rules; only one result notification and one termination notification per subscription are modelled
- Open: which field a conformant consumer is meant to give precedence
- Next step: Ask the specification editors whether termTrainReq is decisive, or require an explicit notification type.

X01: notification carries at least one report field

Every notification accepted by the consumer carries at least one of delayEventNotif, mLModelInfos or termTrainReq, and never carries delayEventNotif together with mLModelInfos or with termTrainReq.

Every notification the consumer accepts carries at least one report field, so a notification with no report at all does not occur. The other half of the rule, that delayEventNotif never comes together with the other two fields, is not modelled, because delay notifications belong to the Federated Learning part that the scope leaves out.

- Result: Holds; answers what is asked: partly; usable with caveats
- Value low: NOTE 1 states exactly this field rule, so the result reproduces a stated requirement rather than finding anything.
- Stated in the documents: yes. Where the documents say it: s0321
- Rests on: delay notifications are not modelled, so their exclusivity is not exercised; the NWDAF is honest about the fields it fills in
- Open: the delayEventNotif exclusivity
- Next step: Model the delay notification of the Federated Learning case to exercise the second half of the rule.

Appendix C: Claims and prover answers

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

target	level	bound	lemma	control	attacks	claims
A03	proved		a03			1 proved
A08	proved		a03			1 proved
D01	proved		d01			1 proved
R02	none				r02 (depends_on_ambiguity)	1 attack
S05	none				s05_term (depends_on_ambiguity); s05_unsub (depends_on_ambiguity)	2 attack
Z01	proved		z01	1 of 2		2 proved
Z03	proved		z03			1 proved
A03.8	proved		a03			1 proved
A08.9	proved		a08_9			1 proved
U01	none				x01_both (holds_in_source); u01_cls (holds_in_source)	2 attack
X01	proved		x01_empty			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
A03	Every notification the consumer accepts belongs to a subscription request that this same consumer really sent, with the same notifCorreId and the same mLEvent.	`network`	proved	a03	none
A08	The mLEvent of an accepted notification is an event the consumer subscribed to under the same notifCorreId; the round context (roundInd, statusReport) is not modelled.	`network`	proved	a03	none
D01	Every notification the consumer receives was really sent by the NWDAF for that subscription and its payload was not altered (the transport credential is the only thing protecting it).	`network`	proved	d01	none
R02	The consumer accepts at most one instance of the same notification; in fact it accepts a repeated notification again, because nothing in the notification identifies the instance.	`network`	attack (depends_on_ambiguity)	r02	
S05	After the consumer has accepted a termination notification (termTrainReq) for a subscription, it accepts no further notification for that subscription; in fact it accepts the same termination notification again.	`network`	attack (depends_on_ambiguity)	s05_term	
S05	After the unsubscribe of a subscription has been accepted, the consumer accepts no further notification for it; in fact a notification sent before the unsubscribe is accepted after it.	`network`	attack (depends_on_ambiguity)	s05_unsub	
Z01	Every subscription request the NWDAF accepts is authorised by a token that the NRF issued for the nnwdaf-mlmodeltraining scope.	`network`	proved	z01	checked

target	claim	attacker	answer	query	control
Z01	Every unsubscribe the NWDAF accepts is authorised by a token that the NRF issued for the nnwdaf-mlmodeltraining scope.	`network`	proved	z01_unsub	none
Z03	An operation on a subscription is accepted only for a subscription that this consumer established; another party cannot act on it without knowing its identifiers.	`network`	proved	z03	none
A03.8	The notifCorreId of an accepted notification equals the notifCorreId of the subscription request the consumer really sent.	`network`	proved	a03	none
A08.9	The subscription identifier carried by an accepted notification is one the NWDAF assigned to a subscription created for that same notifCorreId, so a notification is not mixed in from another subscription.	`network`	proved	a08_9	none
U01	A notification never carries termTrainReq together with mLModelInfos; in fact the source permits it and the NWDAF can send a termination that also carries the final model.	`network`	attack (holds_in_source)	x01_both	
U01	The consumer classifies a notification by the fields it carries, so a termination notification that also carries the final model is classified as a training result and the consumer does not learn that the subscription ended.	`network`	attack (holds_in_source)	u01_cls	
X01	Every notification the consumer accepts carries at least one report field, so the consumer never has to handle a notification with no report at all.	`network`	proved	x01_empty	none

Appendix D: How the analysis was run

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

ProVerif+ Agent session session-20261114T184451, 2026-11-14. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 56, prover 5 s, wall 13 min, cost 0.119 USD.

ProVerif: 101 proofs run, 93 decided, 8 without an answer, 3 s, 14 answers recorded.

What was analysed

Protocol	NWDAF model training event subscription and result notification
Version	19.8.0
Analysed	A single training provider, a single consumer, a single training task; model training is opaque internal computation; ordinary event subscription/notification is verified first, without enabling full horizontal/vertical federated learning.
Not analysed	This is not the full HFL/VFL protocol, and it does not prove that training results are correct; repeated subscriptions, re-subscription after termination and round state need to be developed separately. It proves only the properties under the explicit model and attacker conditions; missing checks, a full secure channel or an all-honest NF cannot stand in for the conclusions to be proved.
Source text	379 numbered passages (88264 bytes) from the sections listed below. A reference such as s0001 in this report is one of these passages

The sections of the documents the analysis is built from:

Document	Section	Title
main	4.6	Nnwdaf_MLModelTraining Service
main	4.6.1	Service Description
main	4.6.1.1	Overview
main	4.6.1.2	Service Architecture
main	4.6.1.3	Network Functions
main	4.6.1.3.1	Network Data Analytics Function (NWDAF)
main	4.6.1.3.2	NF Service Consumers
main	4.6.2	Service Operations
main	4.6.2.1	Introduction
main	4.6.2.2	Nnwdaf_MLModelTraining_Subscribe service operation
main	4.6.2.2.1	General
main	4.6.2.2.2	Subscription for event notifications
main	4.6.2.2.3	Update subscription for event notifications
main	4.6.2.2.4	Partial update subscription for event notifications
main	4.6.2.3	Nnwdaf_MLModelTraining_Unsubscribe service operation
main	4.6.2.3.1	General

Document	Section	Title
main	4.6.2.3.2	Unsubscribe from event notifications
main	4.6.2.3.3	Unsubscribe from event notifications providing additional information
main	4.6.2.4	Nnwdaf_MLModelTraining_Notify service operation
main	4.6.2.4.1	General
main	4.6.2.4.2	Notification about subscribed event
main	5.5	Nnwdaf_MLModelTraining Service API
main	5.5.1	Introduction
main	5.5.2	Usage of HTTP
main	5.5.2.1	General
main	5.5.2.2	HTTP standard headers
main	5.5.2.2.1	General
main	5.5.2.2.2	Content type
main	5.5.2.3	HTTP custom headers
main	5.5.3	Resources
main	5.5.3.1	Resource Structure
main	5.5.3.2	Resource: NWDAF ML Model Training Subscriptions
main	5.5.3.2.1	Description
main	5.5.3.2.2	Resource definition
main	5.5.3.2.3	Resource Standard Methods
main	5.5.3.2.3.1	POST
main	5.5.3.2.4	Resource Custom Operations
main	5.5.3.3	Resource: Individual NWDAF ML Model Training Subscription
main	5.5.3.3.1	Description
main	5.5.3.3.2	Resource definition
main	5.5.3.3.3	Resource Standard Methods
main	5.5.3.3.3.1	PUT
main	5.5.3.3.3.2	PATCH
main	5.5.3.3.3.3	DELETE
main	5.5.3.3.4	Resource Custom Operations
main	5.5.3.3.4.1	Overview
main	5.5.3.3.4.2	Operation: unsubscribe-info
main	5.5.3.3.4.2.1	Description
main	5.5.3.3.4.2.2	Operation Definition
main	5.5.4	Custom Operations without associated resources
main	5.5.5	Notifications
main	5.5.5.1	General

Document	Section	Title
main	5.5.5.2	Event Notification
main	5.5.5.2.1	Description
main	5.5.5.2.2	Operation Definition
main	5.5.6	Data Model
main	5.5.6.1	General
main	5.5.6.2	Structured data types
main	5.5.6.2.1	Introduction
main	5.5.6.2.2	Type NwdafMLModelTrainSubsc
main	5.5.6.2.3	Type NwdafMLModelTrainSubscPatch
main	5.5.6.2.4	Void
main	5.5.6.2.5	Type MLModelTrainInfo
main	5.5.6.2.6	Type MLTrainReportInfo
main	5.5.6.2.7	Type FailureEventInfoForMLModelTrain
main	5.5.6.2.8	Type NwdafMLModelTrainNotif
main	5.5.6.2.9	Void
main	5.5.6.2.10	Type DataAvReq
main	5.5.6.2.11	Type DelayEventNotif
main	5.5.6.2.12	Type StatusReportInfo
main	5.5.6.2.13	Type TrainDataInfo
main	5.5.6.2.14	Type TrainingUnsubscribeInfo
main	5.5.6.3	Simple data types and enumerations
main	5.5.6.3.1	Introduction
main	5.5.6.3.2	Simple data types
main	5.5.6.3.3	Enumeration: FailureCodeTrain
main	5.5.6.3.4	Enumeration: TermTrainCause
main	5.5.6.3.5	Enumeration: DelayCause
main	5.5.6.3.6	Enumeration: FLServerTermCause
main	5.5.7	Error handling
main	5.5.7.1	General
main	5.5.7.2	Protocol Errors
main	5.5.7.3	Application Errors
main	5.5.8	Feature negotiation
main	5.5.9	Security

The source step located the purpose of the protocol in the documents as follows: a single training provider, a single consumer, a single training task; model training is treated as opaque internal computation; ordinary event subscription/notification is verified first, and full horizontal/vertical federated learning is not enabled.

The properties against the documents

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

ID	Name	About	What the documents say	Where
A03	notification tied to own subscription	The NF service consumer is assured that every notification it accepts belongs to a subscription it created itself, with the same notifCorreId and addressed to its own notifUri.	notifCorreId is assigned by the consumer for the requested notifications and the notification shall carry the same value (s0042, s0129, s0321); the NWDAF posts to the notifUri received in the subscription (s0128).	s0042, s0128, s0129, s0321
A08	notification context matches subscription	The NF service consumer is assured that the event identifier and the round/context fields of an accepted notification correspond to the events it subscribed to, not to another subscription or another event.	the subscription describes the subscribed events with mLEvent (s0038 to s0040) and the notification reports mLModelInfos and roundInd for the subscription identified by notifCorreId (s0131, s0136, s0321).	s0038, s0039, s0040, s0131, s0136, s0321
D01	notification authenticity and integrity	The NF service consumer is assured that an accepted notification was really sent by the NWDAF containing MTLF for that subscription and that its payload was not altered on the way.	the NWDAF shall invoke the Notify operation to notify the subscribed event (s0128) and the consumer stores what it accepts and answers 204 (s0138); the documents leave message protection to the transport layer of TS 29.500 and TS 33.501.	s0128, s0138
R02	one acceptance per notification instance	The NF service consumer accepts at most one instance of the same logical notification of a subscription.	The documents say nothing about repetition or de-duplication of a notification: notifCorreId identifies the subscription, not the notification instance (s0129, s0321), and the consumer only stores the notification (s0138).	s0129, s0138, s0321
S05	nothing accepted after termination	After a termination notification or an effective unsubscribe, the NF service consumer accepts no further notification for that subscription.	termTrainReq indicates that the subscription is requested to be terminated and that no further notifications related to this subscription will be provided (s0133, s0321); an unsubscribe deletes the subscription (s0101 to s0110).	s0105, s0108, s0133, s0321
Z01	service operations covered by token	The NWDAF is assured, and the consumer can rely, that every subscription operation it accepts was authorised by a token issued by the NRF for the nnwdaf-mlmodeltraining scope.	access to the API may be authorised with OAuth2 client credentials, the NRF is the authorisation server and the consumer shall obtain a token before consuming the service (s0375, s0376); the API defines the single scope nnwdaf-mlmodeltraining (s0378).	s0375, s0376, s0378
Z03	token bound to its subscription	A token held by one NF service consumer cannot be used to act on another consumer's subscription or to receive its notifications.	the API defines a single service-wide scope and no additional scopes at resource or operation level (s0378), so the documents bind the token to the service, not to a subscription.	s0206, s0378
A03.8	notifCorreId equals own subscription	The NF service consumer is assured that the notification correlation identifier of an accepted notification equals the notifCorreId of the subscription request it really sent.	notifCorreId is assigned by the consumer and the notification shall carry the same value as the subscription (s0042, s0129, s0321).	s0042, s0129, s0321
A08.9	no mixing of subscriptions	The NF service consumer is assured that an accepted notification and its payload belong to one of its own subscriptions and are not mixed from another subscription.	notifCorreId identifies the subscription to which the notification relates (s0321) and mLModelInfos reports the subscribed event (s0131).	s0129, s0131, s0321

ID	Name	About	What the documents say	Where
U01	termination distinguished from result	The NF service consumer can tell a termination notification from a training result notification, and a notification never carries termTrainReq together with mLModelInfos.	a notification carries at least one of delayEventNotif, mLModelInfos or termTrainReq; delayEventNotif is mutually exclusive with each of the other two, but termTrainReq and mLModelInfos are not declared mutually exclusive (s0321 NOTE 1); the notification has no explicit type field (s0128 to s0137).	s0128, s0131, s0133, s0321
X01	notification carries at least one report field	The NF service consumer's well-formedness assumption about the notification body it stores.	NOTE 1 of the NwdafMLModelTrainNotif definition states this rule and makes only the two delayEventNotif combinations mutually exclusive.	s0321

Added from the documents by the analyst: X01.

The parties and the protected values, one by one

The host listed the 15 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: 0. Without: 15. The list follows from the properties; what was made of each entry is the analyst's judgement.

ID	Party or value	Kind	Attacker	Why none
S01	NF service consumer	party	Nothing gained	A dishonest NF service consumer holds exactly what the service scope grants every registered consumer: its own token, its own subscription identifiers and the transport credential. It learns no other consumer's notifCorreId, notifUri or subscriptionId from the protocol, so no property asked here changes.
S02	NWDAF	party	Nothing gained	A dishonest NWDAF learns a consumer's identifiers only through that consumer's own subscription, and the correctness of the trained model is outside the task. The properties asked here are about messages the NWDAF really sends; a dishonest provider sending its own notifications does not break them.
S03	NRF	party	Nothing gained	A compromised authorisation server can mint tokens for the service scope, but it holds none of the transport credentials that the messages are protected with, and any registered consumer can obtain a token of the same scope anyway; no property asked here becomes answerable only under it.
S04	notifCorreId	value	Nothing gained	notifCorreId is chosen by the consumer and only ever travels inside the protected messages; holding it gives no ability to make a notification be accepted without the transport credential.
S05	notifUri	value	Nothing gained	notifUri is chosen by the consumer and carried inside the protected subscription request; knowing it gives nothing without the transport credential.
S06	subscriptionId	value	Nothing gained	Holding a subscriptionId gives no ability to act on the subscription in the model: a request still needs a valid token and the protected channel. What it would allow is stated as the caveat of the token-scope target, because the API defines no resource-level scope.
S07	mLEvent	value	Nothing gained	mLEvent is a public event identifier that both parties legitimately know; it identifies what was subscribed, not who may receive it.
S08	roundInd	value	Outside the scope	Round numbers belong to the multi-round training state that the task's boundaries leave out; no target asks for the state of the rounds.
S09	mLModelInfos	value	Nothing gained	No target and no statement of the documents asks for the confidentiality of the trained model; the payload is protected by the transport in the model, and its correctness is out of scope.
S10	termTrainReq	value	Nothing gained	termTrainReq is a reason code carried in a protected notification; holding it gives no ability to terminate or continue a subscription.

ID	Party or value	Kind	Attacker	Why none
S11	notification payload	value	Nothing gained	The notification payload is only meaningful inside a notification that the consumer has correlated to its own subscription; holding a payload alone breaks no property asked here.
S12	notification instance	value	Nothing gained	Nothing identifies a notification instance beyond the subscription correlation, which is exactly what the de-duplication target asks about; holding an instance gives no key material.
S13	OAuth2 token	value	Nothing gained	A token authorises the service scope only, and every registered consumer can obtain one of the same scope; requests still need the transport credential, so holding a token adds nothing.
S14	scope nnwdaf-mlmodeltraining	value	Nothing gained	The scope is public and identical for every consumer of the service.
S15	failEventReports	value	Outside the scope	failEventReports belongs to the partial-success path of the subscribe operation, which is not part of the plain run modelled.

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		

ID	`network`
A03	Holds
A08	Holds
D01	Holds
R02	Fails
S05	Fails
Z01	Holds
Z03	Holds
A03.8	Holds
A08.9	Holds
U01	Fails
X01	Holds

Source and targets

Source: 379 units, 88264 bytes, frozen in source-delivery-01.json. Targets: 8 approved (A03, A08, D01, R02, S05, Z01, Z03, 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 source package protocol-b33fef4f0dbf4eb9d2c66b76d4e6

This step as it reports itself: 2 requests (2 failed), 0.000 USD, 116167 tokens in and 7679 out, 169 s.

Notes on the prover and the model

What the analyst named for targets:

step	targets	features	reason
49	A03, A08, A03.8, A08.9, D01, R02, S05, Z01, Z03, U01, X01	plain_primitives, many_sessions	All answers use only symmetric authenticated encryption and MACs for the SBA/HTTP layer and a MAC-based OAuth2 token, with any number of consumer/NWDAF sessions. The protocol keeps no counter, sequence number or key that is installed and replaced, so no property depends on a value that changes; the failures are exactly the absence of such a value (no notification identifier/sequence). Round numbers (roundInd), which would be such a value, are outside the task's boundaries and are not modelled.

The analyst's closing note

Eleven targets answered, six of them proved and three falsified, two findings.

Proved on the model: every notification the consumer accepts belongs to a subscription it created itself, with matching notifCorreId, notifUri and mLEvent, and was really sent by the NWDAF unaltered (A03, A03.8, A08 for the event identifier, A08.9, D01); the NWDAF accepts a subscribe and an unsubscribe only with a token the NRF issued for the nnwdaf-mlmodeltraining scope (Z01, both claims, with a control in which the unchecked variant accepts without verifying a token); an operation on a subscription is accepted only for the consumer's own subscription (Z03); and every accepted notification carries at least one report field (X01, first half).

Falsified, and reported as attacks: a notification delivered twice is stored twice (R02), a notification withheld and released after the termination request or after the unsubscribe is accepted as fresh (S05), and a termination notification that also carries the final aggregated model is classified as an ordinary training result (U01). The first two are one finding (nothing identifies a notification instance or orders the stream), the third is the second finding (the notification has no type field and NOTE 1 makes termTrainReq and mLModelInfos compatible).

These all rest on a model that treats the SBA transport as an authenticated encrypted channel and that keeps one consumer, one provider and at most two notifications per subscription. What remains open: cross-consumer use of a subscription identifier (needs a two-consumer model), notification ordering and de-duplication (needs a per-subscription counter, so a bounded model or Tamarin+ Agent), the multi-round state and the delay notification of the Federated Learning case, and the update operations.