Security analysis of NWDAF model monitoring: registration, subscription and feedback 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	9: 1 hold, 8 fail (10 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_MLModelMonitor service of the 5G core lets one NWDAF monitor the analytics accuracy of an ML model used by another. Two roles take part: the NWDAF containing MTLF, which uses and trains the model and wants to be told how accurate it is, and the NWDAF containing AnLF, which runs the model in analytics and observes its accuracy. Both hold an OAuth2 access token of the single service scope, issued by the NRF.

The NWDAF containing AnLF first registers the use and monitoring capability of the ML model at the NWDAF containing MTLF and receives a registration identifier. The NWDAF containing MTLF then subscribes at the NWDAF containing AnLF for the accuracy monitoring event, naming the ML model identifiers, its callback address and a notification correlation identifier, and receives a subscription identifier. The NWDAF containing AnLF notifies the subscribed event to that callback address with the correlation identifier, the accuracy information of the model and analytics feedback, and the NWDAF containing MTLF stores the notification and answers 204 No Content. Either side can end the arrangement with a DELETE on the individual registration or subscription.

The parties:

- **NWDAF containing MTLF**: uses and trains the ML model, subscribes for its accuracy. Holds: OAuth2 access token of scope nnwdaf-mlmodelmonitor.
- **NWDAF containing AnLF**: runs the model, observes its accuracy, notifies. Holds: OAuth2 access token of scope nnwdaf-mlmodelmonitor; the ML models it runs in analytics.
- **NRF**: issues the access token. Holds: authorization server.

Figure 1 shows the exchange as the documents specify it.

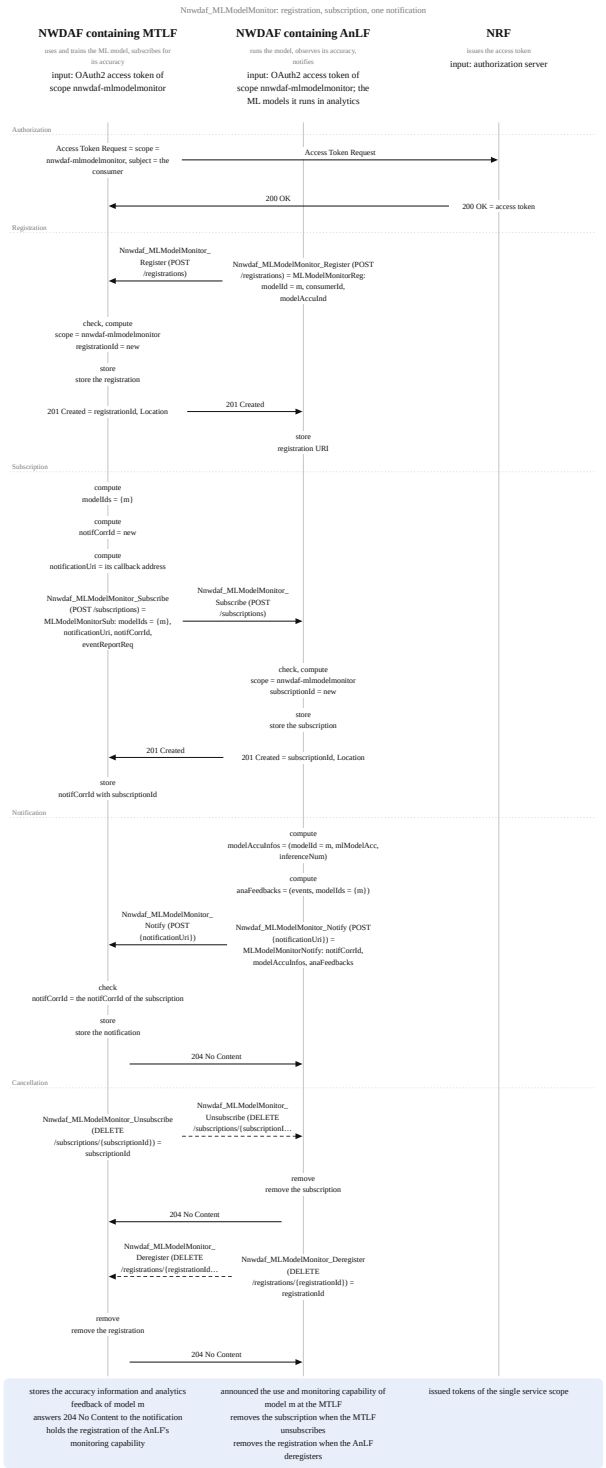


Figure 1. The protocol as specified.

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

2 Findings

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

The service does not give what its notification flow is meant to give: a notification is accepted on the strength of a correlation string that the consumer picks and sends in the clear, because the notify operation carries no OAuth2 token and the documents define no other authentication for it. Forged and repeated notifications are therefore stored by the NWDAF containing MTLF, and the model identifiers they carry are never compared with the subscription. What does hold is that an accepted notification carries the identifier the consumer itself chose, that no Register or Subscribe request is acted on without a token of the service scope, and that the producer replaces the value it keeps when a subscription is deleted. Those two gaps are not stated by the documents, so they are findings; the single-scope authorization gap of Z03 is stated by TS 29.501 clause 4.10 and is reproduced, not found. The answers rest on a model in which the SBI transport is the attacker's channel, so the notification attacks hold for a party on the SBI path and depend on where TLS terminates; the state-dependent answers come from two unrolled monitoring intervals.

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 notification is accepted on a correlation string alone: an on-path party can inject accuracy data

Kind	Weakness of the protocol as specified
New	Unclear: the documents state it in part, or could not be searched
Value	medium
Properties	D01, A03, A03.8, A08, A08.9 (section 3)
Source text	s0073, s0074, s0076, s0122, s0123, s0132, s0319
Shown by	ProVerif attack queries `payload`, `corr_bound`, `model_bound`, `fb_bound`; Figures 2, 3, 4, 5 and 6

What happens. The NWDAF containing MTLF accepts a notification on the strength of the notification correlation identifier alone. That identifier is chosen by the consumer and carried in the subscription request, and TS 29.500 clause 6.7.3 states that the notify operation uses no OAuth2 access token in this release; the documents define no signature or MAC over a notification. A party that reads the subscription request can post its own MLModelMonitorNotify body, and the consumer stores it and answers 204. The modelIds in modelAccuInfos and anaFeedbacks are not compared with the subscription either, so the notification may name another model of the attacker's choice; the only related requirement in the documents is a NOTE that mLEvent, mLEventFilter and tgtUe should match what the AnLF registered.

What the attacker needs. A position from which the attacker reads the HTTP body of the subscription request, so that it learns the notifCorrId, for instance an SCP where TLS terminates as TS 29.500 clauses 6.7.2 and 6.10.9 allow, or a notifCorrId that can be guessed. It needs no token and no key.

What the attacker gains. The NWDAF containing MTLF stores accuracy information and analytics feedback for a model that the NWDAF containing AnLF never reported, and bases its monitoring of the model and its indication whether the accuracy requirement is met on it. The same run attributes data to another ML model of the attacker's choice.

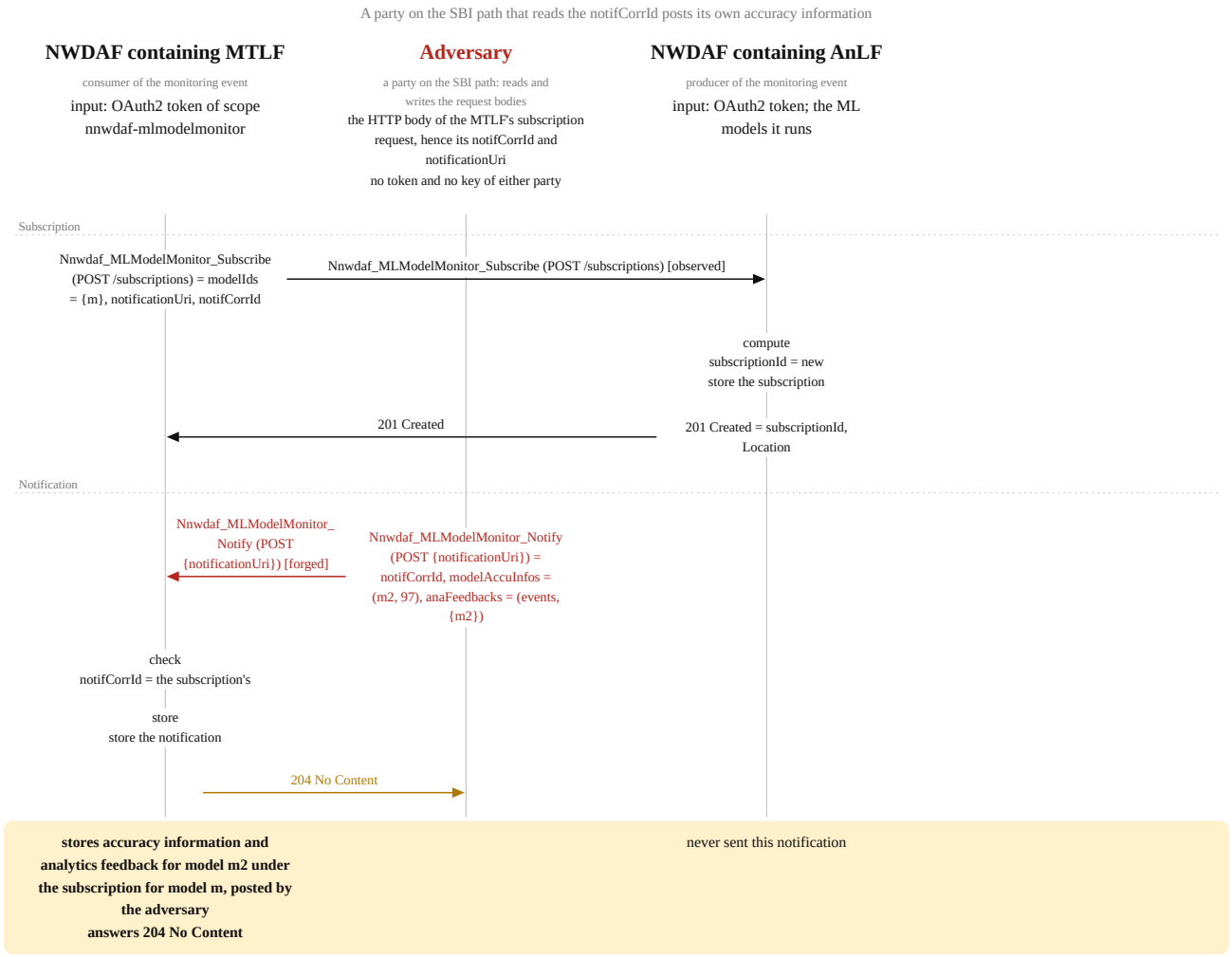


Figure 2. A party on the SBI path that reads the notifCorrId posts its own accuracy information. Property D01; attack query payload.

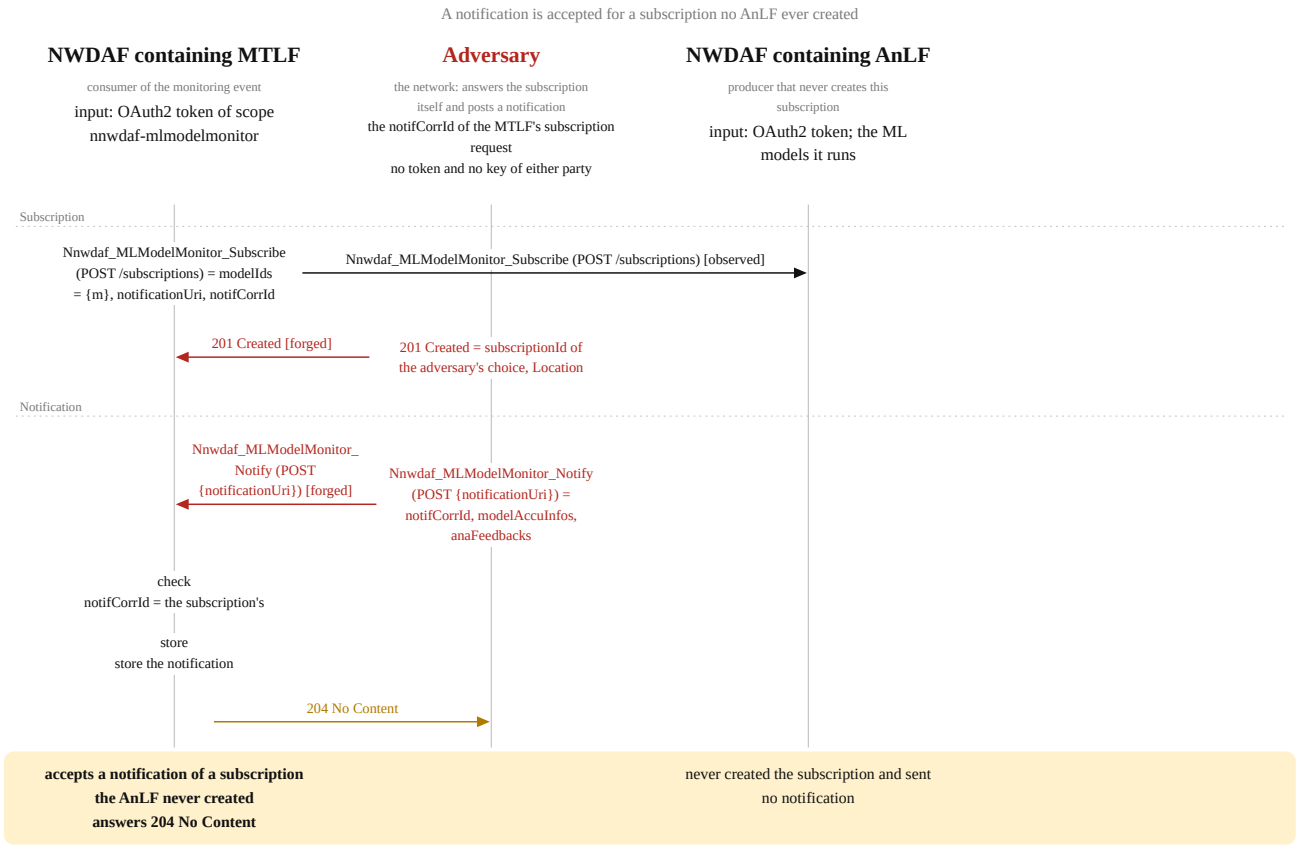


Figure 3. A notification is accepted for a subscription no AnLF ever created. Property A03; attack query corr_bound.

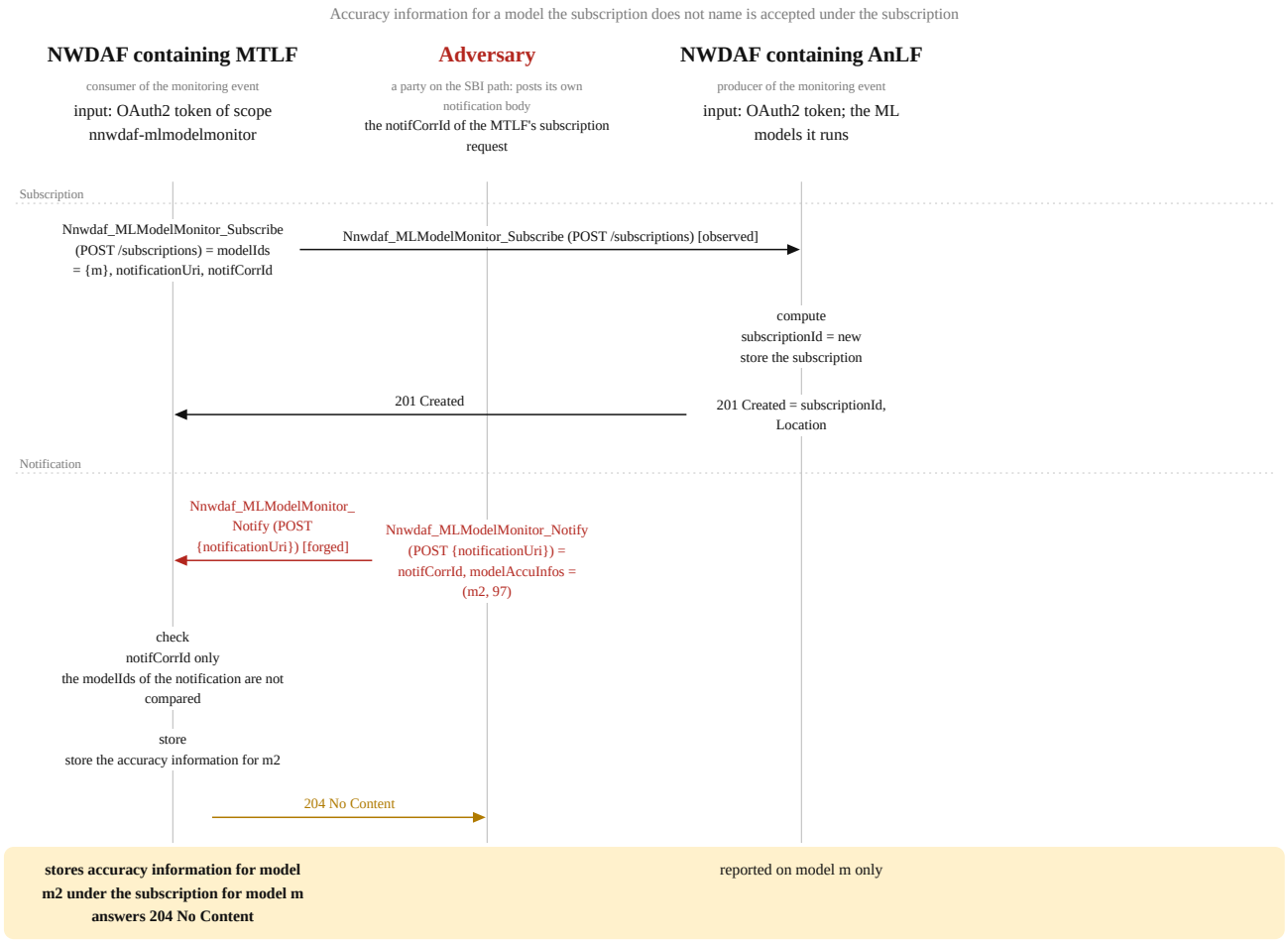


Figure 4. Accuracy information for a model the subscription does not name is accepted under the subscription. Property A08; attack query model_bound.

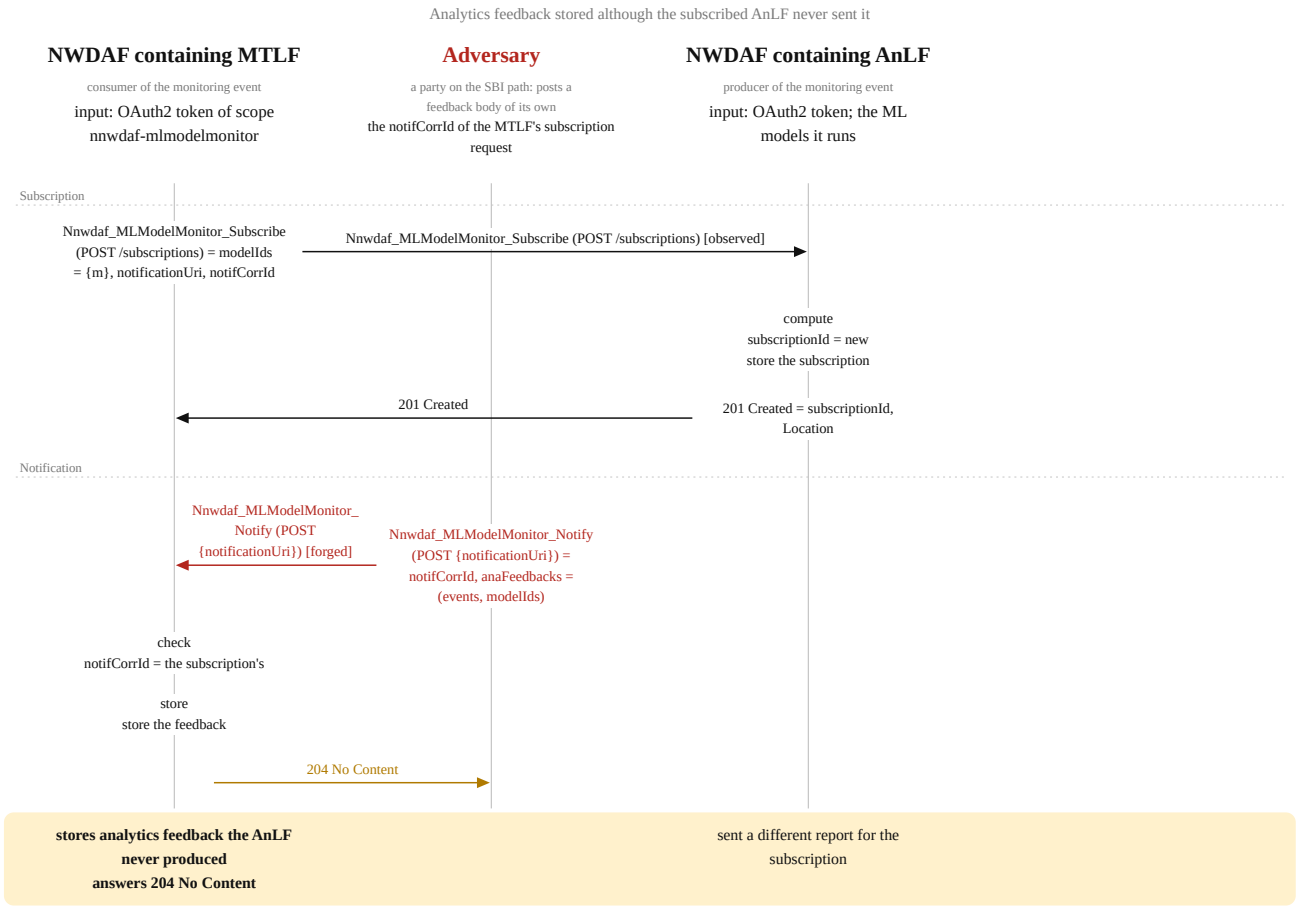


Figure 5. Analytics feedback stored although the subscribed AnLF never sent it. Property A03.8; attack query payload.

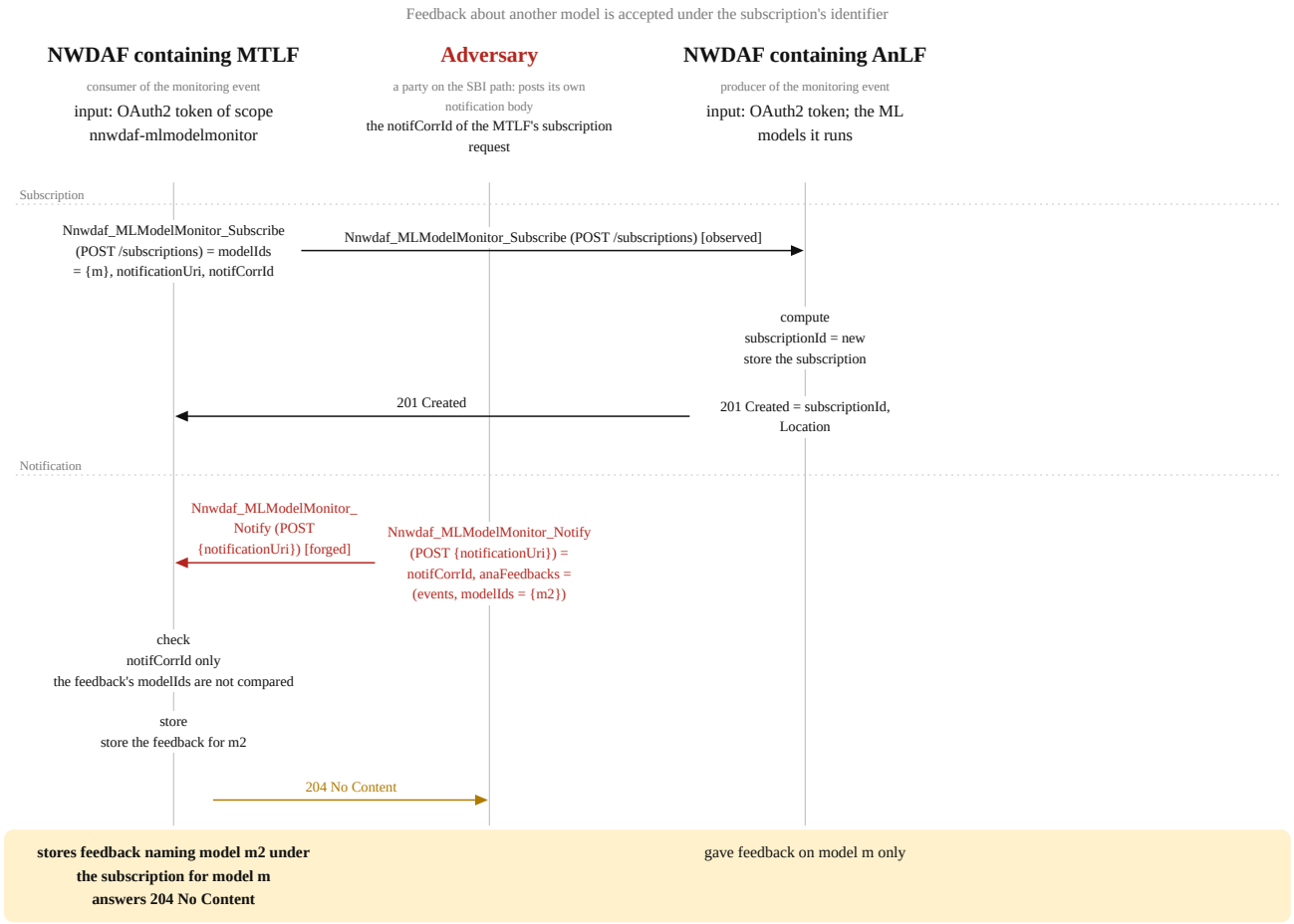


Figure 6. Feedback about another model is accepted under the subscription's identifier. Property A08.9; attack query fb_bound.

Proposed change. Require the notify request to be authenticated as coming from the NF service producer of that subscription, and have the consumer reject a notification whose sender is not the producer it subscribed to. The cheapest form is to use the OAuth2 access token for the notify operation, as for Subscribe and Unsubscribe, and to state that the consumer compares the token subject with the producer instance of the subscription, since TS 29.500 clause 6.7.3 currently excludes the notify operation from token use. In addition, require the consumer to compare the modelIds of modelAccuInfos and anaFeedbacks with the modelIds of the subscription, which the NOTE in clause 5.6.6.2.3 only recommends.

Check of the change with ProVerif. Not checked: The change is at the level of the request contents, but it cannot be shown to work in this model: the model gives the attacker the subscription request itself, so any value carried in it is known to the attacker and no application-level addition to the notification helps there. Checking it needs a model in which the transport hides the request from the attacker, which the task's boundaries exclude.

2.2 A cancelled subscription keeps taking notifications, and a repeated notification is stored again

Kind	Weakness of the protocol as specified
New	Unclear: the documents state it in part, or could not be searched
Value	low
Properties	Z04, A04 (section 3)
Source text	s0109, s0110, s0112, s0123, s0132, s0338
Shown by	ProVerif attack queries `late_accepted`, `replayed`, `no_replay`; Figures 7 and 8

What happens. A cancellation is not a boundary for the consumer. After the NWDAF containing MTLF has unsubscribed and been answered 204 No Content, it still matches incoming notifications against the correlation identifier of the subscription it removed, and stores one that arrives. In the same way a notification it has already stored is stored again when the identical body is delivered a second time, since no report identifier and no duplicate detection is defined.

What the attacker needs. Nothing beyond the network for the in-flight case: the producer's own notification that was already on its way is stored after the cancellation. For a repeat, the same notification body has to be delivered twice, which needs the notifCorrId only if the repeat is made up rather than copied.

What the attacker gains. The monitoring record of a model keeps values for a cancelled subscription and duplicate values for one report, so the data the MTLF uses to judge the accuracy of the model, and the feedback about which analytics and models were used for actions, can be stale or multiplied.

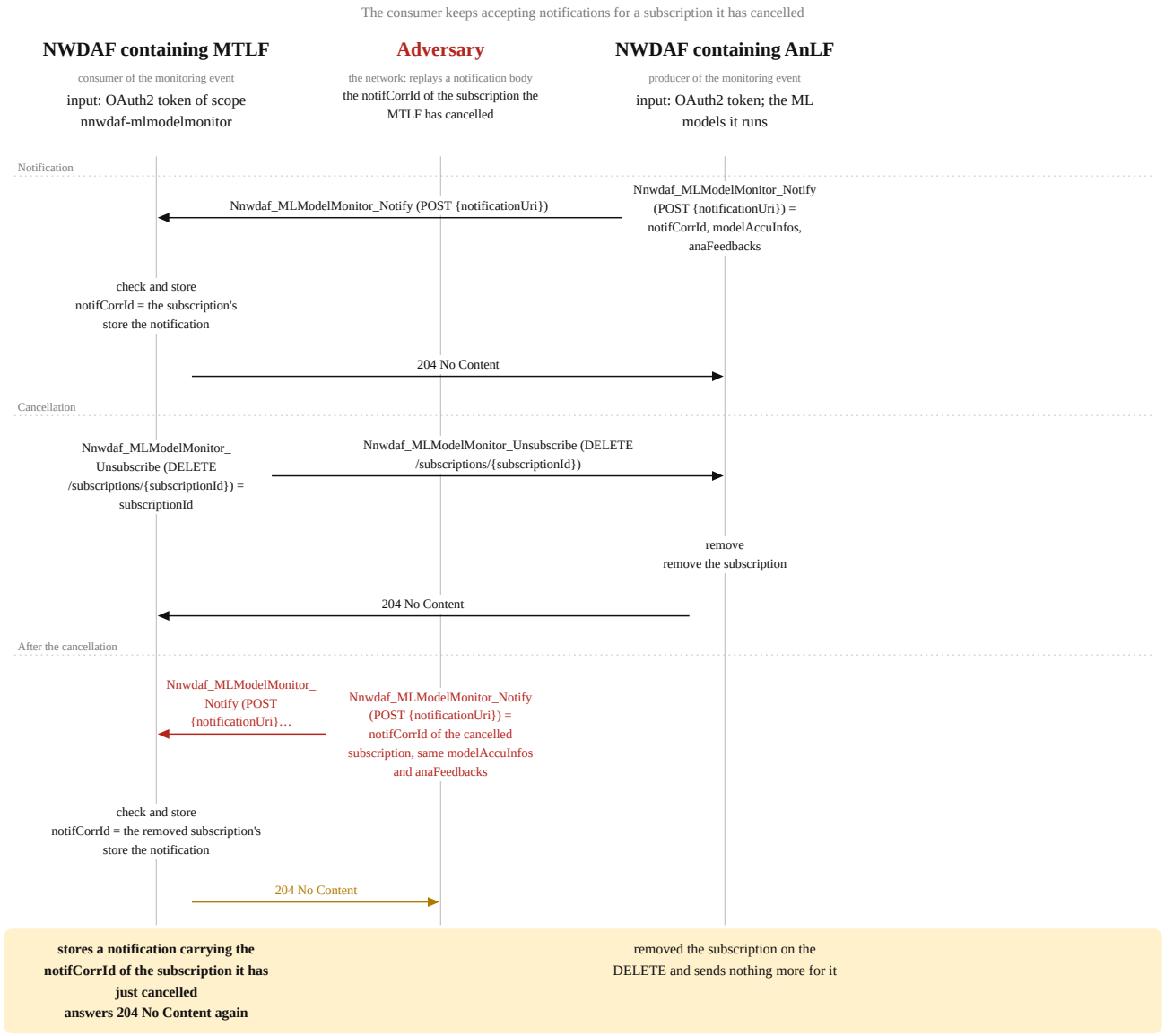


Figure 7. The consumer keeps accepting notifications for a subscription it has cancelled. Property Z04; attack query late_accepted.

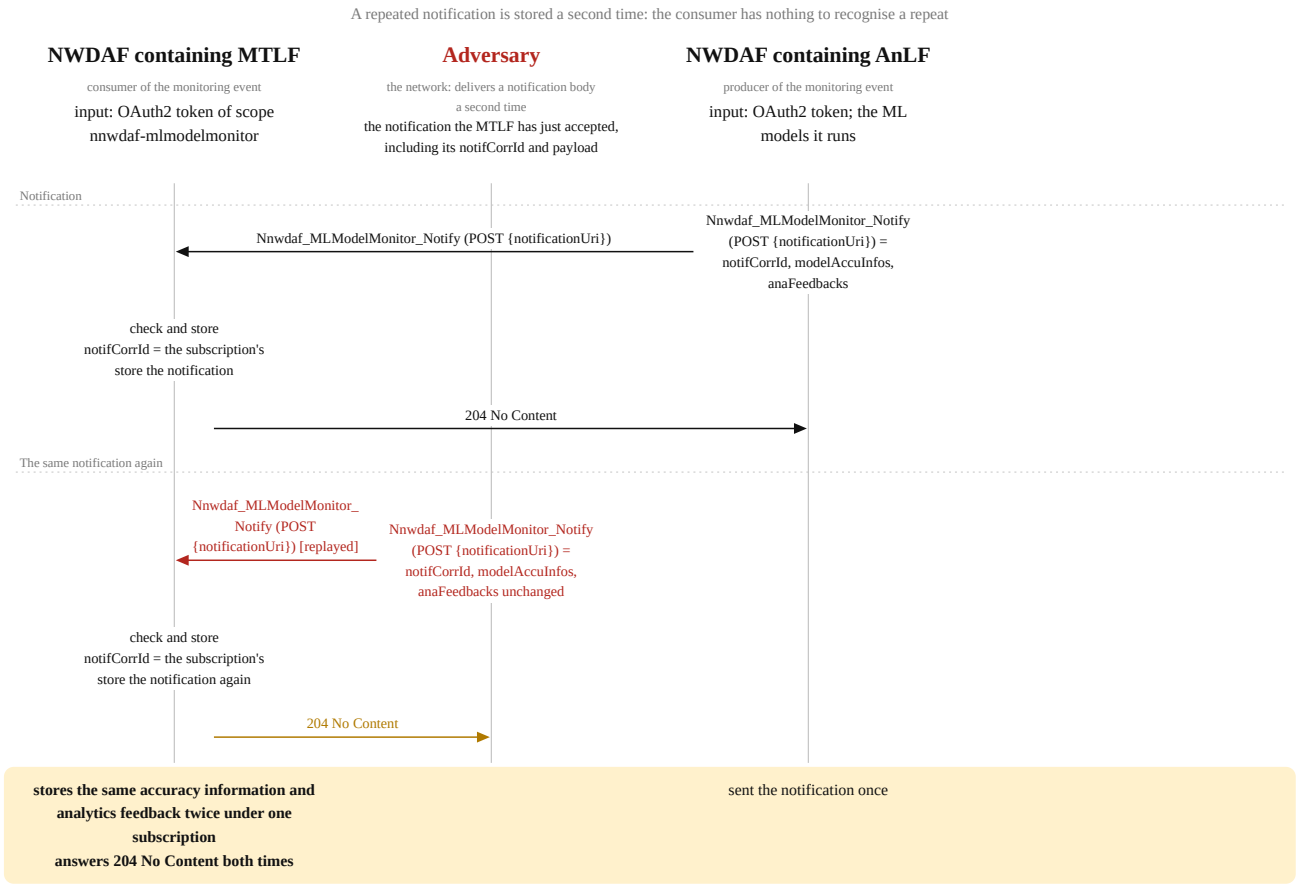


Figure 8. A repeated notification is stored a second time: the consumer has nothing to recognise a repeat. Property A04; attack queries no_replay, replayed.

Proposed change. Have the consumer drop the notifCorrId of a subscription when it cancels it, and state that a notification received for a subscription that is no longer active is discarded and not stored. Add an identifier to MLModelMonitorNotify, or state that notifCorrId together with the reporting interval identifies a report, so that a repeated report can be recognised.

Check of the change with ProVerif. Not checked: The change is a rule about when the consumer may store a notification, and checking it would only confirm that a consumer which discards what it has been told to discard does not store it: the models here give the consumer no history to reason about beyond the two steps already unrolled. What the check could not show is whether real consumers implement such a discard, because the documents require none.

No figure was given for these attacks of a finding: A03.8: corr_bound, A03.8: fb_bound, A03.8: model_bound, A03: fb_bound, A03: model_bound, A03: payload, A04: late_accepted, A08.9: corr_bound, A08.9: model_bound, A08.9: payload, A08: corr_bound, A08: fb_bound, A08: payload, D01: corr_bound, D01: fb_bound, D01: model_bound, Z04: no_replay, Z04: replayed.

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 matches a real subscription.** Whenever the MTLF accepts a notification, there is a real subscription run of an AnLF whose subscriber identity, notificationUri, modelIds and notifCorrId agree with the accepted notification (non-injective).	no compromise	Attack holds (stated in the documents)	medium	yes
A04	**One acceptance per subscription run.** Whenever the MTLF accepts a notification, there is a distinct AnLF subscription run that matches it (injective), and the same subscription run is not accepted twice.	no compromise	Fails (3 attacks)	low	unclear
A08	**Notification fields bound to the subscription.** In the notifications the MTLF accepts, notifCorrId, modelIds and the feedback content correspond to the values approved in its subscription request, and are not replaced or misattached.	no compromise	Fails (1 attack)	medium	unclear
D01	**Notifications come from the subscribed AnLF.** Every notification the MTLF accepts corresponds to a real send by the permitted sender (the subscribed AnLF), with payload and correlation fields consistent.	no compromise	Attack holds (stated in the documents)	medium	yes
Z03	**Token authorizes subject and operation.** For a registration/subscription/notification operation submitted with a token, the subject, audience and operation agree with the authorization granted when the token was issued.	no compromise	Attack holds (stated in the documents)	low	yes
Z04	**Nothing takes effect after cancellation.** After a registration is deleted or a subscription cancellation takes effect, no new notification acceptance or monitoring operation based on that registration/subscription is produced.	no compromise	Fails (1 attack)	medium	unclear
A08.7	**Correlation and URI match subscription.** In the notifications the MTLF accepts, notifCorrId, notificationUri and modelIds correspond to the values in its subscription request.	no compromise	Holds	low	no
A03.8	**Accepted feedback matches a real send.** Whenever the MTLF accepts a feedback notification, there is a real send run of an AnLF that matches it.	no compromise	Attack holds (stated in the documents)	medium	yes
A08.9	**Feedback not cross-wired to a model.** In the notifications the MTLF accepts, the modelIds the feedback content belongs to and the notifCorrId correspond to its subscription request, and are not cross-wired across models or subscriptions.	no compromise	Fails (1 attack)	medium	unclear

"Attack holds (stated in the documents)": the prover found the attack, and the documents themselves already state it. The property fails; this is a known result, not a finding (Appendix B).

The questions asked, and the analyst's answers:

Question	Answer	Rests on
Q1: consistent association of notifCorrId with the subscriber, the callback URI and the modelIds	The consumer does check the correlation identifier and its own callback address, so an accepted notification always carries the notifCorrId the MTLF itself chose. The modelIds are not checked: a notification naming another ML model is accepted under the subscription, and because the correlation identifier is not a secret, consistency with the producer's subscription does not hold either.	A08.7, A03, A08
Q2: an accepted feedback corresponds to a send event of the corresponding AnLF	No. An accepted notification need not correspond to a send of the NWDAF containing AnLF: a party that learns the notifCorrId can post the notification itself, and the MTLF stores it as monitored data.	D01, A03.8

Question	Answer	Rests on
Q3: feedback across models/subscriptions cannot be misattributed (as a requirement to be verified, first check the receiving side's checking rules)	No. Nothing in the API requires the consumer to compare the modelIds of modelAccuInfos or anaFeedbacks with its subscription, so feedback about another model is accepted under the subscription's correlation identifier; the only related text is a NOTE that mLEvent, mLEventFilter and tgtUe should match the registration.	A08.9, A08

4 Limits of this analysis

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

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

- The SBI transport is modelled as the attacker's channel: TLS (TS 29.500 6.7.2) and OAuth2 token issuance and validation (TS 33.501, TS 29.510) are not modelled, only the scope check of the token.
- Whether a deployment lets an on-path element read the HTTP bodies, as where TLS terminates at an SCP, is left open; the notification attacks are submitted as depending on that ambiguity.
- Accuracy values, thresholds, model metrics and the feedback contents are opaque data; nothing about their meaning or correctness is analysed.
- Deregistration and unsubscription are covered at one instance only: the consistency of a registration and a subscription across NF instances is outside the scope, as the task states.
- One ML model and one consumer identity per process are modelled; the answer on per-model authorization comes from the documented single scope, not from a model of the NRF.
- The consumer's behaviour after it has unsubscribed is modelled as storing whatever carries the old notifCorrId; the documents state no rule, so that attack rests on an absent rule rather than on a modelled check.
- bounded.pv covers two monitoring intervals of one subscription; longer histories and the deregistration side of Z04 are not unrolled.
- No algebraic operation (exclusive-or, Diffie-Hellman) plays a role in any answer.

What the host records:

- The answers of A04, Z04 rest on an operation the model simplifies (Appendix D).

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 matches a real subscription

Whenever the MTLF accepts a notification, there is a real subscription run of an AnLF whose subscriber identity, notificationUri, modelIds and notifCorrId agree with the accepted notification (non-injective).

The MTLF accepts a notification for a subscription the AnLF never created: it matches only the correlation identifier it chose, and that identifier travels in the clear in the subscription request. So the existence half of the property, that an accepted notification belongs to a real subscription run, fails whenever a party can read that request.

- Result: Attack holds (stated in the documents); answers what is asked: partly; usable with caveats
- Value medium: The trace is the forged-notification run of the first finding; the documents state the mechanism, that no token is used for notify, but not that a notification can come from another party.
- Stated in the documents: yes. Where the documents say it: s0076, s0123, s0132
- Rests on: The SBI body is readable by the attacker; the model has no TLS.; The subscription body carries no consumer identity, so the subscriber is taken from the token subject.; One ML model instance is modelled and the accuracy values are opaque.
- Open: The attack needs the notifCorrId, so it holds for a party on the SBI path and not for an attacker that has the network alone.
- Next step: A model in which the transport hides the subscription request from the attacker would separate the two readings, or a clarification in the specification of what the consumer must check besides the notifCorrId.

A04: One acceptance per subscription run

Whenever MTLF accepts a notification, there is a distinct AnLF subscription run that matches it (injectivity), and the same corresponds to that subscription run not being accepted twice.

One notification can be accepted more than once, and a subscription stays in force, so several acceptances belong to one subscription run. The injectivity reading of the target is not a property of this protocol; what matters is that a repeat is stored again and that the consumer has no identifier to tell a repeat from a new report.

- Result: Fails (3 attacks); answers what is asked: partly; usable with caveats
- Value low: Duplicate monitoring data is a data-quality problem here, and the source neither forbids repeats nor requires detection.
- Stated in the documents: unclear. Source text it rests on: s0123, s0132, s0338
- Rests on: bounded.pv unrolls the consumer to two acceptances and the producer to two intervals.; The repeat needs the notifCorrId only when it is made up rather than copied.
- Open: Whether a repeated notification should be rejected at all is not settled by the documents.
- Next step: The at-most-once question needs a rule in the specification, such as a report identifier, or a stated discard after cancellation; the injectivity reading should be dropped from the target.

A08: Notification fields bound to the subscription

In a notification accepted by MTLF, the notifCorrId, modelIds and feedback contents correspond to the values approved in its subscription request, and are not replaced or cross-wired.

The modelIds carried in the accuracy information are not compared with the subscription, so a notification about a different ML model is accepted under the subscription's correlation identifier. The documents require no such comparison; only a NOTE recommends that related fields match the registration.

- Result: Fails (1 attack); answers what is asked: no; usable with caveats
- Value medium: The absence of a modelIds check is a real gap in the checks the consumer is asked to perform, and the documents do not state it.
- Stated in the documents: unclear. Source text it rests on: s0074, s0125, s0319
- Rests on: The attacker reads the subscription request to learn the notifCorrId.; The accuracy values are opaque; only the model identifiers are compared.
- Open: Whether the consumer is expected to check modelIds is left to the implementation.
- Next step: A model of the transport that hides the subscription request, or an explicit consumer check of modelIds against the subscription, would settle which model identifiers may appear.

D01: Notifications come from the subscribed AnLF

Every notification accepted by MTLF corresponds to a real send by the permitted sender (the subscribed AnLF), with the payload consistent with the correlation fields.

An accepted notification does not correspond to a send of the subscribed AnLF: a party that knows the correlation identifier posts its own body and the consumer stores it. The only protection the documents give the notify operation is the transport, since no token is used for it.

- Result: Attack holds (stated in the documents); answers what is asked: no; usable with caveats
- Value medium: This is the central gap of the first finding, and TS 29.500 clause 6.7.3 states that the notify operation uses no token without saying what that costs.
- Stated in the documents: yes. Where the documents say it: s0117, s0122, s0132
- Rests on: No application-level protection of the notify request is modelled, because the documents define none.; The attacker reads the subscription request; the model gives it every body on the SBI.
- Open: Nothing here distinguishes a legitimate in-flight notification from an injected one; the model shows that both are accepted.
- Next step: An authenticated notify request, or a MAC under a key that is not carried in the subscription, is needed to make this hold; the second is outside what this model can show.

A08.7: Correlation and URI match subscription

In a notification accepted by MTLF, the notifCorrId, notificationUri and modelIds correspond to the values in its subscription request.

The correlation identifier and the callback address of an accepted notification are the ones the MTLF itself used, so the consumer's check on the identifier works as specified. This is all the binding the protocol defines.

- Result: Holds; answers what is asked: yes; usable
- Value low: It is the mechanism the specification states, proved where it holds; the control shows that the same query fails when a field the consumer does not check is added.

- Stated in the documents: no. Source text it rests on: s0075, s0076, s0123
- Rests on: The callback address is the consumer's own endpoint, so only the notifCorrId is really compared.; The model does not consider a consumer that reuses one identifier across subscriptions.
- Next step: Nothing further; the property holds as far as the model can ask it.

A03.8: Accepted feedback matches a real send

Whenever MTLF accepts a feedback notification, there is a real send run of the AnLF that matches it.

The analytics feedback the MTLF stores was not necessarily produced by the AnLF: a party that knows the correlation identifier sends its own feedback body and the consumer stores it. So the feedback record of a model can be filled by someone other than the producer.

- Result: Attack holds (stated in the documents); answers what is asked: no; usable with caveats
- Value medium: Same gap as the first finding, stated for the feedback payload; the documents do not state it.
- Stated in the documents: yes. Where the documents say it: s0117, s0126, s0132
- Rests on: No sender authentication of the notification is modelled, as the documents define none.; The attacker reads the subscription request.
- Open: Whether an in-flight notification from the AnLF should be rejected is not settled.
- Next step: An authenticated notify request is the change that would make this hold.

A08.9: Feedback not cross-wired to a model

In a notification accepted by MTLF, the modelIds to which the feedback contents belong and the notifCorrId correspond to its subscription request, and are not cross-wired across models or subscriptions.

Feedback naming a model other than the subscribed one is accepted under the subscription's identifier, so monitoring data can be attributed to the wrong ML model. Nothing in the API requires the consumer to compare the two sets of model identifiers.

- Result: Fails (1 attack); answers what is asked: no; usable with caveats
- Value medium: The missing modelIds check is a gap the documents do not state; only a NOTE hints at it.
- Stated in the documents: unclear. Source text it rests on: s0074, s0126, s0328
- Rests on: The attacker reads the subscription request.; The feedback contents are opaque; only the model identifiers are modelled.
- Open: The documents only recommend in a NOTE that related fields match what the producer registered.
- Next step: Requiring the consumer to compare the feedback's modelIds with the subscription, and to authenticate the sender, would settle it.

Z03: Token authorizes subject and operation

For a registration/subscription/notification operation submitted with a token, the subject, audience and operation are consistent with the authorization at the time the token was issued.

A Register or Subscribe request is acted on only with a token of the service scope, and that is the whole of the authorization: the API defines one scope for the entire service, so any NF holding it may register and subscribe for any ML model, including a model another NF uses. The documents state this consequence themselves.

- Result: Attack holds (stated in the documents); answers what is asked: yes; usable

- Value low: The documents state it in so many words: an API with a service-level scope only grants access to all its operations to all consumers allowed to access the service.
- Stated in the documents: yes. Where the documents say it: ts-29-501:1801, main:19224, s0344, s0345, s0347
- Rests on: The token is modelled as an unforgeable term whose scope the producer checks; issuance, expiry and audience are not modelled.; The rogue NF is given its token, which every NF obtains from the NRF; that is not a leaked secret.
- Next step: Nothing further for the authorization structure; the single-scope answer is what TS 29.501 clause 4.10 gives.

Z04: Nothing takes effect after cancellation

After a registration is deleted or a subscription cancellation takes effect, no new notification acceptances or monitoring operations based on that registration/subscription are produced.

After the NWDAF containing MTLF has cancelled its subscription and been answered 204, it still accepts and stores a notification carrying the identifier of the removed subscription, and a repeated notification is stored again. On the producer side the opposite holds: once the DELETE has been processed the AnLF sends nothing more for that subscription, which depends on the value it keeps being replaced.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: The lifecycle gap is real and unstated, but its effect is stale or repeated monitoring data rather than a loss of secrecy; the producer side of the property does hold.
- Stated in the documents: unclear. Source text it rests on: s0109, s0111, s0112, s0132, s0338
- Rests on: The unrolled model covers two monitoring intervals of one subscription; the deregistration side is not unrolled.; The consumer's continuing to match the old notifCorrId is a modelling decision taken because the documents state no rule.; The consumer-side attack needs the notifCorrId; a legitimate in-flight notification would also be accepted, which no query isolates.
- Open: Whether a notification must be discarded after the subscription was cancelled is not settled by the documents.; The reading where an NF cancels another consumer's subscription is recorded as an unconfirmed candidate in the state review.
- Next step: Unrolling the deregistration side and a rule for notifications in flight would answer the rest; under the rogue NF an answer needs a model in which the DELETE carries the caller's identity, which the subscription body does not provide.

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		corr_own	checked	corr_bound (depends_on_ambiguity)	1 proved, 1 attack
A04	reachable		reach_accepted		no_replay (depends_on_ambiguity); inj_sub (holds_in_source); replayed (holds_in_source)	3 attack
A08	reachable		reach_accepted		model_bound (depends_on_ambiguity)	1 attack
D01	reachable		reach_accepted		payload (depends_on_ambiguity)	1 attack
Z03	proved		no_bad_scope_sub	checked	rogue_sub (holds_in_source)	2 proved, 1 attack
Z04	bounded	the changing value unrolled to 2 steps	late_sent	checked	late_accepted (holds_in_source)	1 bounded, 1 attack
A08.7	proved		corr_own	checked		1 proved
A03.8	reachable		reach_accepted		payload (depends_on_ambiguity)	1 attack
A08.9	reachable		reach_accepted		fb_bound (depends_on_ambiguity)	1 attack

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	The notifCorrId and notificationUri of an accepted notification are those of a subscription request the MTLF itself made.	`on_path`	proved	corr_own	checked
A03	The accepted notification belongs to a subscription run really created at the AnLF, with the same subscriber identity, notificationUri, modelIds and notifCorrId. The attacker has, beyond the network, the bodies of the SBI requests (it reads the subscription request and post to the notificationUri).	`on_path`	attack (depends_on_ambiguity)	corr_bound	
A04	One notification is not accepted twice: each acceptance is matched by a distinct send of the AnLF. The attacker has, beyond the network, the bodies of the SBI requests.	`on_path`	attack (depends_on_ambiguity)	no_replay	
A04	Each acceptance is matched by a distinct subscription run, so one subscription run is not matched by two acceptances.	`on_path`	attack (holds_in_source)	inj_sub	
A04	One notification is not accepted twice: a repeated notification is not stored again under the same subscription.	`on_path`	attack (holds_in_source)	replayed	

target	claim	attacker	answer	query	control
A08	The modelIds of the accuracy information in an accepted notification are the modelIds of the MTLF's subscription, not substituted. The attacker has, beyond the network, the bodies of the SBI requests.	on_path	attack (depends_on_ambiguity)	model_bound (the control of corr_own)	
D01	An accepted notification corresponds to a real send of the NWDAF containing AnLF the MTLF subscribed to, with the same payload and correlation fields. The attacker has, beyond the network, the bodies of the SBI requests.	on_path	attack (depends_on_ambiguity)	payload	
Z03	A Register or Subscribe operation on an ML model is performed only for a subject that the token's authorization covers for that model. The attacker has, beyond the network, a valid OAuth2 access token of the single service scope.	rogue_nf	attack (holds_in_source)	rogue_sub (the control of no_bad_scope)	
Z03	A Subscribe request is acted on only if it carries an OAuth2 access token whose scope is the single service scope nnwdaf-mlmodelmonitor.	network	proved	no_bad_scope_sub	checked
Z03	A Register request is acted on only if it carries an OAuth2 access token whose scope is the single service scope nnwdaf-mlmodelmonitor.	network	proved	no_bad_scope	checked
Z04	After the MTLF has cancelled its subscription (Unsubscribe answered 204), no notification based on that subscription is accepted any more.	on_path	attack (holds_in_source)	late_accepted	
Z04	After the AnLF has processed the DELETE that removes a subscription, it sends no further notification for that subscription.	rogue_nf	bounded (the changing value unrolled to 2 steps)	late_sent	checked
A08.7	The notifCorrId and notificationUri of an accepted notification are those of the MTLF's own subscription request.	on_path	proved	corr_own	checked
A03.8	An accepted feedback notification corresponds to a real send of the subscribed AnLF, so the analytics feedback and accuracy information the MTLF stores were produced by that producer. The attacker has, beyond the network, the bodies of the SBI requests.	on_path	attack (depends_on_ambiguity)	payload	
A08.9	The modelIds of the analytics feedback in an accepted notification are the modelIds of the MTLF's subscription, so feedback is not attributed to another model. The attacker has, beyond the network, the bodies of the SBI requests.	on_path	attack (depends_on_ambiguity)	fb_bound	

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-20261114T182230, 2026-11-14. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 57, prover 5 s, wall 17 min, cost 0.136 USD.

ProVerif: 92 proofs run, 92 decided, 0 without an answer, 3 s, 27 answers recorded.

What was analysed

Protocol	NWDAF model monitoring: registration, subscription and feedback notification
Version	19.8.0
Analysed	Fixed MTLF/AnLF two roles; first a monitoring subscription with fixed modelIds and one feedback, the registration relationship is modelled separately, it cannot be assumed without basis that every subscription is preceded by a registration. The accuracy value is treated as opaque data.
Not analysed	The truth of the accuracy values, model accuracy, convergence or threshold statistics are not verified; where the API leaves some checks to the implementation/policy, a check cannot be added on one's own and the original protocol then claimed to be secure; the global consistency of deregistration and unsubscription is handled separately. Only the properties under the explicit model and attacker conditions are proved, missing checks, a complete secure channel or all-honest NFs cannot be substituted for the conclusion to be proved.
Source text	347 numbered passages (63012 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	4.7	Nnwdaf_MLModelMonitor Service
main	4.7.1	Service Description
main	4.7.1.1	Overview
main	4.7.1.2	Service Architecture
main	4.7.1.3	Network Functions
main	4.7.1.3.1	Network Data Analytics Function (NWDAF)
main	4.7.1.3.2	NF Service Consumers
main	4.7.2	Service Operations
main	4.7.2.1	Introduction
main	4.7.2.2	Nnwdaf_MLModelMonitor_Register service operation
main	4.7.2.2.1	General
main	4.7.2.2.2	Registering the monitoring of the analytics accuracy of an ML Model
main	4.7.2.3	Nnwdaf_MLModelMonitor_Deregister service operation
main	4.7.2.3.1	General
main	4.7.2.3.2	Deregistering the monitoring of the analytics accuracy of an ML Model

Document	Section	Title
main	4.7.2.4	Nnwdaf_MLModelMonitor_Subscribe service operation
main	4.7.2.4.1	General
main	4.7.2.4.2	Subscription for monitoring notifications
main	4.7.2.4.3	Update of subscription for monitoring notifications
main	4.7.2.5	Nnwdaf_MLModelMonitor_Unsubscribe service operation
main	4.7.2.5.1	General
main	4.7.2.5.2	Unsubscribe from monitoring notifications
main	4.7.2.6	Nnwdaf_MLModelMonitor_Notify service operation
main	4.7.2.6.1	General
main	4.7.2.6.2	Notification about subscribed event
main	5.6	Nnwdaf_MLModelMonitor Service API
main	5.6.1	Introduction
main	5.6.2	Usage of HTTP
main	5.6.2.1	General
main	5.6.2.2	HTTP standard headers
main	5.6.2.2.1	General
main	5.6.2.2.2	Content type
main	5.6.2.3	HTTP custom headers
main	5.6.3	Resources
main	5.6.3.1	Resource Structure
main	5.6.3.2	Resource: NWDAF ML model monitoring registrations
main	5.6.3.2.1	Description
main	5.6.3.2.2	Resource Definition
main	5.6.3.2.3	Resource Standard Methods
main	5.6.3.2.3.1	POST
main	5.6.3.2.4	Resource Custom Operations
main	5.6.3.3	Resource: Individual NWDAF ML model monitoring registration
main	5.6.3.3.1	Description
main	5.6.3.3.2	Resource definition
main	5.6.3.3.3	Resource Standard Methods
main	5.6.3.3.3.1	DELETE
main	5.6.3.3.4	Resource Custom Operations
main	5.6.3.4	Resource: NWDAF ML model monitoring Subscriptions
main	5.6.3.4.1	Description
main	5.6.3.4.2	Resource Definition
main	5.6.3.4.3	Resource Standard Methods

Document	Section	Title
main	5.6.3.4.3.1	POST
main	5.6.3.4.4	Resource Custom Operations
main	5.6.3.5	Resource: Individual NWDAF ML model monitoring Subscription
main	5.6.3.5.1	Description
main	5.6.3.5.2	Resource definition
main	5.6.3.5.3	Resource Standard Methods
main	5.6.3.5.3.1	PUT
main	5.6.3.5.3.2	DELETE
main	5.6.3.5.4	Resource Custom Operations
main	5.6.4	Custom Operations without associated resources
main	5.6.5	Notifications
main	5.6.5.1	General
main	5.6.5.2	Event Notification
main	5.6.5.2.1	Description
main	5.6.5.2.2	Operation Definition
main	5.6.6	Data Model
main	5.6.6.1	General
main	5.6.6.2	Structured data types
main	5.6.6.2.1	Introduction
main	5.6.6.2.2	Type MLModelMonitorReg
main	5.6.6.2.3	Type MLModelMonitorSub
main	5.6.6.2.4	Type MLModelMonitorNotify
main	5.6.6.2.5	Type MLModelAccuracyInfo
main	5.6.6.2.6	Type AnalyticsFeedback
main	5.6.7	Error handling
main	5.6.7.1	General
main	5.6.7.2	Protocol Errors
main	5.6.7.3	Application Errors
main	5.6.8	Feature negotiation
main	5.6.9	Security

The purpose of the protocol as the source step located it in the documents: fix the two roles MTLF/AnLF; first carry out the monitoring subscription and one feedback for fixed modelIds; model the registration relationship separately; do not assume without basis that every subscription must be preceded by a registration. Treat the accuracy value as opaque data.

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 matches a real subscription	The NWDAF containing MTLF is assured that every notification it accepts belongs to a subscription run it made at the NWDAF containing AnLF, with the same subscriber identity, notificationUri, modelIds and notifCorrId; non-injective.	The consumer sends modelIds, notificationUri and notifCorrId in the subscription request (s0074, s0075, s0076), the AnLF notifies with the notifCorrId (s0122, s0123) and the consumer stores the notification (s0132). The documents state no check that binds an accepted notification back to the subscription beyond the notifCorrId, and no application error for an unknown notifCorrId is defined (s0338).	s0008, s0069, s0073, s0076, s0084, s0117, s0123, s0132
A04	One acceptance per subscription run	The NWDAF containing MTLF's acceptance of a notification is matched by a distinct subscription run of the NWDAF containing AnLF, so that one notification is not accepted twice under one subscription.	The subscription is created with a subscriptionId (s0086) and notifications carry the notifCorrId (s0123); the documents define no message-level replay protection, no duplicate detection and no application error for a repeated notification (s0338), and they leave anti-replay to the transport (TS 29.500 clause 6.7.2).	s0076, s0086, s0123, s0132
A08	Notification fields bound to the subscription	The NWDAF containing MTLF is assured that the notifCorrId, modelIds and notificationUri of an accepted notification, and the feedback it carries, are the ones of its own subscription request and are neither substituted nor cross-wired.	The subscription fixes modelIds, notificationUri and notifCorrId (s0074, s0075, s0076); the notification carries notifCorrId plus modelAccuInfos and anaFeedbacks, each naming modelIds (s0123, s0125, s0126, s0322, s0328). The documents only recommend, in a NOTE, that mLEvent, mLEventFilter and tgtUe should match what was registered (s0319), and state no mandatory check of a notification's modelIds against the subscription.	s0074, s0075, s0076, s0123, s0124, s0125, s0126, s0319
D01	Notifications come from the subscribed AnLF	The NWDAF containing MTLF is assured that every notification it accepts corresponds to a real send by the NWDAF containing AnLF it subscribed to, with the same payload and correlation fields.	The Notify operation is used by the NWDAF containing AnLF to notify the subscriber (s0117, s0288), and the consumer stores the notification and answers 204 (s0132). The documents define no signature, MAC or token for a notification; TS 29.500 clause 6.7.3 states that the notify operation uses no OAuth2 access token in this release and TS 29.500 clause 6.7.2 leaves message protection to TLS.	s0117, s0122, s0126, s0132, s0288
Z03	Token authorizes subject and operation	The NWDAF containing MTLF and the NWDAF containing AnLF are assured that the subject, the audience and the operation of a Register, Subscribe or Notify request are those the OAuth2 token was issued for.	Access to the API may be authorized by OAuth2 with the NRF as authorization server (s0344), the consumer shall obtain a token before consuming the service (s0345), and the API defines a single scope for the entire service and no additional scopes at resource or operation level (s0347). TS 29.501 clause 4.10 states that an API with a service-level scope only grants access to all its service operations to all consumers allowed to access the service, and TS 29.500 clause 6.7.3 adds that the notify operation uses no token.	main:19224, ts-29-501:1801, ts-29-500:5032, s0344, s0345, s0347

ID	Name	About	What the documents say	Where
Z04	Nothing takes effect after cancellation	The NWDAF containing MTLF and the NWDAF containing AnLF are assured that once a subscription has been cancelled (Unsubscribe, s0109 to s0112) or a registration deleted (Deregister, s0061 to s0064), no notification or monitoring step based on that subscription or registration still takes effect.	The NWDAF shall remove the subscription on DELETE and answer 204 (s0110, s0111, s0112) and shall remove the registration on DELETE (s0062, s0063, s0064). The documents state no rule on notifications already in flight or repeated after the removal, and put the global consistency of deregistration and unsubscription aside (task boundaries).	s0061, s0063, s0064, s0109, s0111, s0112
A08.7	Correlation and URI match subscription	The NWDAF containing MTLF is assured that an accepted notification's notifCorrId and notificationUri are the ones of its own subscription request.	The consumer supplies notificationUri and notifCorrId in the subscription (s0075, s0076) and notifications are posted to {notificationUri} with the notifCorrId (s0122, s0123).	s0073, s0074, s0075, s0076, s0123
A03.8	Accepted feedback matches a real send	The NWDAF containing MTLF is assured that an accepted feedback notification corresponds to a real send of the NWDAF containing AnLF, so that the analytics feedback and accuracy information it stores were produced by the subscribed producer.	The AnLF notifies the subscribed event with anaFeedbacks and modelAccuInfos (s0117, s0122, s0126) and the consumer stores it (s0132); no per-message authentication of the notify request is defined (TS 29.500 clause 6.7.3: no token for notify).	s0117, s0118, s0122, s0126, s0132
A08.9	Feedback not cross-wired to a model	The NWDAF containing MTLF is assured that the analytics feedback it accepts for a model is not attributed to another model or another subscription than the one it asked for.	The subscription fixes modelIds (s0074) and notifCorrId (s0076), while a notification carries anaFeedbacks whose modelIds are named per element (s0126, s0328); the documents require no comparison of these modelIds with the subscription, only a NOTE-level should for mLEvent, mLEventFilter and tgtUe (s0319).	s0074, s0076, s0123, s0126, s0328

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

The catalogue of properties, entry by entry

The properties were drafted from a catalogue of security and privacy properties. The draft left out 36 of its entries, and the analyst went through each of them: examined here as a property, or left out with the reason. Examined: 8. Left out: 28. That the documents do not speak of a property is not a reason the analyst may give.

Entry	Property of the catalogue	Examined as	Why not
K01	Key/credential confidentiality	Outside the scope	No long-term key or secret credential is defined by this API. The OAuth2 access token is a bearer credential whose protection is at the transport level and is defined in TS 33.501 and TS 29.500, which the task leaves out. The token's authorization meaning is covered by Z03.
K02	Payload and end-to-end confidentiality	Outside the scope	Confidentiality of the accuracy and feedback payload on the SBI is provided, if at all, by TS 33.501 transport security, which the task leaves out. The only application-level control over who obtains the monitoring data is the authorization that Z03 examines.
K03	Strong confidentiality/indistinguishability of secret values	Not applicable	The API defines no secret value whose indistinguishability is required.

Entry	Property of the catalogue	Examined as	Why not
K04	Key usage, direction and session separation	Not applicable	No keys are derived or used by this API.
A01	Peer participation (Aliveness)	A03	
A02	Weak consistent authentication	A03	
A05	Implicit key authentication	Not applicable	The protocol establishes no key.
A06	Explicit key confirmation	Not applicable	No key is established or confirmed.
A07	Identity—credential—key binding	Z03	
R01	Authentication/challenge freshness	Not applicable	The API defines no challenge, counter, timestamp or ticket freshness mechanism.
R02	Message-level replay resistance	A04	
R03	Business operation at most once	A04	
R04	Message acceptance order/window security	Not applicable	The API defines no sequence number, ordering rule or acceptance window.
R05	Resistance to downgrade/preservation of security policy	Not applicable	The supported-features table of the API is empty (s0341, s0342), so there is no negotiation to lower.
C01	Forward secrecy (FS)	Not applicable	No key establishment, so forward secrecy does not apply.
C02	Resistance to key-compromise impersonation (KCI)	Not applicable	No long-term-key authentication of messages by the API.
C03	Known-key security/compromise isolation	Not applicable	No key hierarchy or session keys in the API.
C04	Resilience to ephemeral secret/partial state leakage	Not applicable	No ephemeral secrets or component state whose leak the API protects against.
C05	Post-compromise security recovery (PCS)	Not applicable	No ratchet or key update mechanism.
C06	Enhanced authentication after compromise of registration credentials	Not applicable	No password-authenticated key exchange and no registration record that authenticates a client.
C07	Resistance to offline dictionary attacks	Not applicable	No password or other human-chosen secret is used.
C08	Goal preservation under failure of a designated primitive	Outside the scope	The API defines no cryptographic primitive of its own. The primitives in use belong to TLS and to the OAuth2 framework, which the task leaves out.
Z01	Correctness of authorization enforcement	Z03	
Z02	No expansion of delegated authority	Not applicable	The API has no delegation or proxy chain. SCPs forward requests without delegated authorization.
Z05	Cross-tenant/cross-service security isolation	Not applicable	The API has no tenant or domain isolation semantics of its own. PLMN separation is not part of the modelled clauses.
S01	Uniqueness of cryptographic nonces	Not applicable	No nonce-sensitive cryptographic primitive is used by the API.
S02	Security of retransmission/key reinstallation	Not applicable	No key installation or retransmission state machine in the API.

Entry	Property of the catalogue	Examined as	Why not
S03	Resistance to rollback and state monotonicity	Z04	
S04	Consistency of security context across phases	Outside the scope	The task states that the registration relation is modelled separately and that no subscription may be assumed to require a prior registration, and it puts the global consistency of deregistration and unsubscription aside. Cross-phase context consistency of this kind is therefore outside the scope.
S05	Termination and invalidation of old state	Z04	
P01	Identity anonymity	Not applicable	The parties are identified NF instances. No anonymity goal is stated or implied.
P02	Session unlinkability	Not applicable	No unlinkability goal is stated. NF identities are visible by design in the SBI.
P03	Hiding of user/credential existence	Not applicable	The API has no login or discovery query that could enumerate users or credentials.
V01	Conditional progress/error recovery	The prover cannot be asked	Progress and termination cannot be asked of the prover: it answers properties of traces, not that a run always completes.
V02	No unexpected deadlock	The prover cannot be asked	Absence of deadlock is a branching-time property the prover cannot be asked.
V03	Limited resource abuse/resistance to amplification	The prover cannot be asked	Resource consumption and amplification are quantitative properties the prover cannot be asked.

The parties and the protected values, one by one

The host listed the 13 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: 6. Without: 7. 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	NWDAF containing MTLF	party	`rogue_nf`	
S02	NWDAF containing AnLF	party	Nothing gained	The AnLF is the origin of the accuracy information and analytics feedback it notifies. Controlling it gives only data it produces anyway. The notification-binding targets still hold because they ask for a real send of that party, and the truthfulness of the accuracy values is excluded by the task.
S03	NRF	party	Everything lost by definition	The NRF is the authorization server and the trust anchor for discovery. With it controlled every authorization property is void by construction, and the documents ask no property of this API under a dishonest NRF.
S04	notifCorrId	value	`on_path`	
S05	notificationUri	value	`on_path`	
S06	modelIds	value	Nothing gained	The ML model identifiers are small integers carried in the clear in the subscription and in every notification. Knowing them adds nothing to an attacker that reads the request, and they do not unlock anything by themselves.
S07	Subscriber identity	value	Nothing gained	The subscription body carries no consumerId: the subscriber is known only from the token subject and the transport peer, both visible to the AnLF by design. NF identities are not secrets in the SBI.

ID	Party or value	Kind	Attacker	Why none
S08	subscriptionId	value	`on_path`	
S09	anaFeedbacks	value	Nothing gained	The analytics feedback is the notification payload. No target asks for its confidentiality, and reading it needs the payload, which the on_path attacker already has. The task keeps the meaning of the feedback out of scope.
S10	modelAccuInfos	value	Nothing gained	The accuracy information is the notification payload. Its values are opaque data by the task statement and no confidentiality target is asked of them.
S11	OAuth2 access token	value	`rogue_nf`	
S12	scope nnwdaf-mlmodelmonitor	value	Nothing gained	The scope is the public service name. Holding the name grants nothing without a token, and the token is the attacker of S11.
S13	registrationId	value	`on_path`	

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		
`on_path`	reads and can rewrite the HTTP request and response bodies exchanged between the NWDAF containing MTLF and the NWDAF containing AnLF, so it learns notifCorrId, notificationUri, modelIds, subscriptionId and registrationId, and it can post requests to any of the two NFs' resource and callback URIs. It holds no OAuth2 token of its own.	The API's own messages carry no application-level protection: TLS is the only stated protection of SBI messages (TS 29.500 clause 6.7.2), SCPs mediate the requests and may terminate TLS (TS 29.500 clause 6.10.9), and the notify operation uses no OAuth2 access token at all in this release (TS 29.500 clause 6.7.3), while the documents state no signature or MAC over a notification.	ts-29-500:4980, ts-29-500:5032, s0344, s0122
`rogue_nf`	a valid OAuth2 access token of the single scope nnwdaf-mlmodelmonitor, which every NF obtains from the NRF, and the addresses of the service; it can invoke Register, Deregister, Subscribe and Unsubscribe with parameters of its choice (any modelIds, its own notificationUri and notifCorrId), and it receives the notifications it subscribed to. It does not read other NFs' transport-protected traffic.	Every NF service consumer shall obtain a token before consuming the service (s0345) and the API defines a single scope for the entire service with no resource- or operation-level scope (s0347), so a service-level scope grants generic access to all operations (TS 29.501 clause 4.10).	ts-29-501:1801, main:19224, s0344, s0345, s0347

ID	`network`	`on_path`	`rogue_nf`
A03		Fails	
A04		Fails	
A08		Fails	
D01		Fails	
Z03	Holds		Fails
Z04		Fails	Holds

ID	network	on_path	rogue_nf
A08.7		Holds	
A03.8		Fails	
A08.9		Fails	

Source and targets

Source: 347 units, 63012 bytes, frozen in source-delivery-01.json. Targets: 6 approved (A03, A04, A08, D01, Z03, Z04); 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-188d5da04c35c21d9e19487f1a09

This step as it reports itself: 1 requests (1 failed), 0.000 USD, 46859 tokens in and 3348 out, 151 s.

Notes on the prover and the model

- A04: answered in ProVerif, and the analyst names changing_state for this target
- Z04: 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+ models these as they are.

A bound in steps is an answer on bounded.pv: the analyst's second ProVerif model, in which the party that keeps a value that changes is unrolled to that many steps with its check in place. That model does less than the protocol does, so the property holds for those steps and no further, and an attack found there is an attack. The number of steps is the analyst's statement.

State review by the analyst

This part is the analyst's own reading of the source, not an answer of the prover. "yes" under shown by ProVerif means that ProVerif found a trace with these steps in the unrolled model, so the steps are possible; that they break a target is said only by an attack in the table of answers, and what else they cost is the analyst's reading. "nothing found" says which patterns were looked at and proves nothing.

value	kept by	changes when	checked when	refs
the set of ML model monitoring subscriptions, each with its subscriptionId, modelIds, notificationUri and notifCorrId	NWDAF containing AnLF	On Subscribe (a new subscription is created, assigned a subscriptionId and stored) and on Unsubscribe (the subscription is removed); the removal is the step Z04 turns on.	When the AnLF sends a notification for a subscription (s0122), and when a DELETE names a subscriptionId (s0110).	s0073, s0085, s0086, s0087, s0109, s0110, s0111, s0114
the set of ML model monitoring registrations, each with its registrationId, modelId and consumer identifier	NWDAF containing MTLF	Created on Register (registrationId assigned, registration stored) and removed on Deregister.	When a DELETE names a registrationId (s0061), and when the registration is the reason for monitoring the model.	s0038, s0048, s0049, s0050, s0051, s0061, s0062, s0063, s0064, s0066

value	kept by	changes when	checked when	refs
the notifCorrId the consumer attributes to its subscription	NWDAF containing MTLF	Chosen when the subscription is made; the source gives no step that invalidates it on Unsubscribe.	At every notification: the consumer compares the notifCorrId in MLModelMonitorNotify with the one of its subscription (s0123, s0132).	s0076, s0110, s0111, s0112, s0123, s0132, s0338

pattern	result	target	shown by ProVerif	steps
replay	candidate	A04	yes: ProVerif falsifies `replayed` on bounded.pv	The MTLF has accepted a notification for its subscription and answered 204 (s0132). / The same MLModelMonitorNotify body, with the same notifCorrId, modelAccuInfos and anaFeedbacks (s0122, s0123), is posted to the notificationUri again. / The MTLF finds the same notifCorrId among the subscriptions it made, stores the notification a second time and answers 204 again (s0132); no duplicate detection and no application error for a repeated notification is defined (s0338).
repeat_one_time_action	candidate	A04	no: not tried with ProVerif	The MTLF sends POST /subscriptions with its modelIds, notificationUri and notifCorrId (s0073) and the response is lost on the way back, so it sends the same request again. / On the second POST the AnLF again creates a new subscription, assigns a new subscriptionId and stores it (s0085, s0086, s0087); the documents require no idempotency key and no duplicate detection. / Both subscriptions later notify the same notificationUri with the same notifCorrId (s0122), so every monitoring report is delivered twice under one identifier and the MTLF cannot tell the two subscriptions apart.
desynchronisation	candidate	Z04	yes: ProVerif falsifies `late_accepted` on bounded.pv	The MTLF unsubscribes: DELETE on its subscription, and the AnLF removes the subscription and answers 204 (s0110, s0111, s0112). / The AnLF had already sent a notification for that subscription (s0122) which reaches the MTLF after its 204; the two parties are then in different states and nothing in the documents brings them back in step. / The MTLF keeps matching incoming notifications against the notifCorrId of the subscription it has removed, so it accepts and stores the late one (s0132); it kept no record that would let it refuse, and no rule tells it to.
update_before_check	nothing found			Looked at the token check and the removal: the subscription is created only after the scope check, and removed only after a DELETE the producer accepted. No value is changed before the message that changes it is verified.
no_rollback_on_failure	nothing found			The AnLF replaces the value it keeps when it accepts the DELETE (verified on bounded.pv as late_sent) and leaves it unchanged when it refuses it (s0114), which is what the source says. The consumer side that fails to move is recorded under desynchronisation.
race_between_sessions	candidate		no: not tried with ProVerif	The MTLF's subscription is created and its subscriptionId appears in the Location header of the 201 response (s0088); a party on the SBI path reads it. / That party sends DELETE on /subscriptions/{subscriptionId} with its own OAuth2 access token: the API has one service scope (s0347), so the only check the AnLF can apply to the caller is that scope. / The subscription body carries no consumerId (s0073 to s0076), so the AnLF has stored nothing that would let it compare the caller with the subscriber; it removes the MTLF's subscription and answers 204 (s0111, s0112), and the MTLF's monitoring of the model stops.

pattern	result	target	shown by ProVerif	steps
distinguishable_failure	candidate		no: not tried with ProVerif	A caller sends DELETE on /subscriptions/{subscriptionId} once with a valid token and an existing identifier, and once with a valid token and an identifier that does not exist. / The first is answered 204 No Content (s0112), the second with an HTTP error (s0114) or 401/403 if the token is not accepted (TS 29.500 6.7.3). / The difference tells the caller whether that subscriptionId exists at the AnLF, which is what an NF that wants to cancel someone else's monitoring needs; the subscriptionId is a URI segment assigned by the AnLF (s0088), so the gain is bounded by how predictable it is, and no target asks for existence hiding.
wrap_or_window	nothing found			Looked at every value: none is a counter, a sequence number or a window. The notifCorrId is an opaque string the consumer picks, subscriptionId and registrationId are assigned once, and the API defines no acceptance window or ordering rule (s0338, s0342).

What the analyst named for targets:

step	targets	features	reason
31	Z04	changing_state	Z04 is about the set of live subscriptions and registrations, which the AnLF and the MTLF keep and change: a subscription is created on Subscribe and removed on Unsubscribe, and whether a later notification still takes effect depends on that value. model.pv leaves the removal out (a store that only grows), so the removal is unrolled to two monitoring intervals in bounded.pv.
31	A03, A04, A08, D01, A08.7, A03.8, A08.9	plain_primitives, many_sessions	The notification-binding answers rest on no algebraic primitive: the model has only an unforgeable OAuth2 token term and opaque accuracy data, and the SBI transport is modelled as the Dolev-Yao channel with any number of sessions of both parties. What is simplified is the transport and the token framework, not an operation with laws.
31	Z03	plain_primitives	Z03 rests on the OAuth2 access token only, modelled as an unforgeable term whose scope the producers check; the token framework itself (issuance, expiry, audience) is not modelled, and no operation with algebraic laws is involved.
42	A04	plain_primitives, changing_state, many_sessions	The replay answer for A04 rests on the state the consumer keeps for its subscription: whether a repeat of a notification it already stored is recognised depends on what it retains, which model.pv does not unroll, so the answer is taken from bounded.pv where that state is unrolled to two steps.

The analyst's closing note

I modelled the Nnwdaf_MLModelMonitor exchange of TS 29.503 clause 4.7 between the NWDAF containing AnLF and the NWDAF containing MTLF: Register, Subscribe with modelIds, notificationUri and notifCorrId, one Notify carrying the accuracy information and analytics feedback, and the DELETE operations for unsubscribe and deregister. Registration is modelled on its own, no subscription is required to follow a registration, and the producers check only the OAuth2 service scope as the documents require. All nine targets have an answer for every claim under every attacker set for it, each proof carries a control, and the figures and the reading are recorded.

Established: an accepted notification does carry the notifCorrId and callback address the consumer itself chose (A08.7 and one claim of A03, proved; the control shows the same query fails for a field the consumer does not check); no Register or Subscribe request is acted on without a token of the service scope (Z03, proved); once the AnLF has processed the DELETE it sends nothing more for that subscription (Z04, bounded with the subscription store unrolled to two intervals, control falsifies it).

Two findings, neither stated by the documents. First, a notification is accepted on a correlation string alone: the notify operation carries no OAuth2 token (TS 29.500 clause 6.7.3), no signature or MAC is defined, and the modelIds in the notification are never compared with the subscription, so a party on the SBI path that reads the subscription request can inject accuracy information and feedback for any model (D01, A03, A03.8, A08, A08.9 falsified). Second, a cancellation is not a boundary: the consumer keeps accepting notifications for a subscription it has cancelled and stores a repeated notification again (Z04, A04 falsified in both the plain and the unrolled model). The single-scope authorization gap of Z03 is reproduced as documented, not found (TS 29.501 clause 4.10).

Limits: the SBI transport is the attacker's channel in the model, so every attack that needs the notifCorrId is submitted as depending on where TLS terminates and on whether an on-path element is in the threat model; the per-model authorization question could not be asked of a model of the NRF; the unrolled model covers two monitoring intervals of one subscription and not the deregistration side; no answer rests on exclusive-or or Diffie-Hellman. Three state-review patterns are candidates with a query (replay, desynchronisation) or a written trace (retried Subscribe, cancellation by another NF, distinguishable failures).