# Security analysis of Review ML model provision with SBA isolation

19.8.0 · ProVerif+ Agent report

| | |
|---|---|
| **Documents** | 5G System; Network Data Analytics Services; Stage 3 (protocol definition); 5G System; Principles and Guidelines for Services Definition; Stage 3 (supporting); Security architecture and procedures for 5G system (supporting) |
| **Properties examined** | 9: 3 hold, 6 fail (6 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_MLModelProvision service lets an NF service consumer obtain ML models from the NWDAF that contains the Model Training Logical Function without carrying the model itself: the consumer subscribes to ML model events and is sent a notification that names the model and the address where the model file can be fetched. The consumer holds its TLS client credential and obtains an OAuth2 access token from the NRF; the NWDAF holds the token key shared with the NRF and keeps the subscriptions; the NRF is the authorization server.

The consumer subscribes with a POST to the subscriptions resource, giving its callback URI, a notification correlation identifier and the events it wants. The NWDAF checks the token, creates a subscription with an identifier it assigns, returns that identifier in the Location header, and later posts a notification to the callback URI carrying the identifier and the model file address. The consumer stores the notification and answers 204, and it deletes the subscription by its identifier when it no longer wants the events.

The parties:

- **NF service consumer**: NWDAF-AnLF, NWDAF-MTLF or LMF. Holds: TLS client certificate and key; registered SBI endpoint.
- **NWDAF (MTLF)**: publishes the ML models. Holds: TLS server certificate; key shared with the NRF; subscriptions.
- **NRF**: OAuth2 authorization server. Holds: TLS server certificate; key shared with the NWDAF.

Figure 1 shows the exchange as the documents specify it.

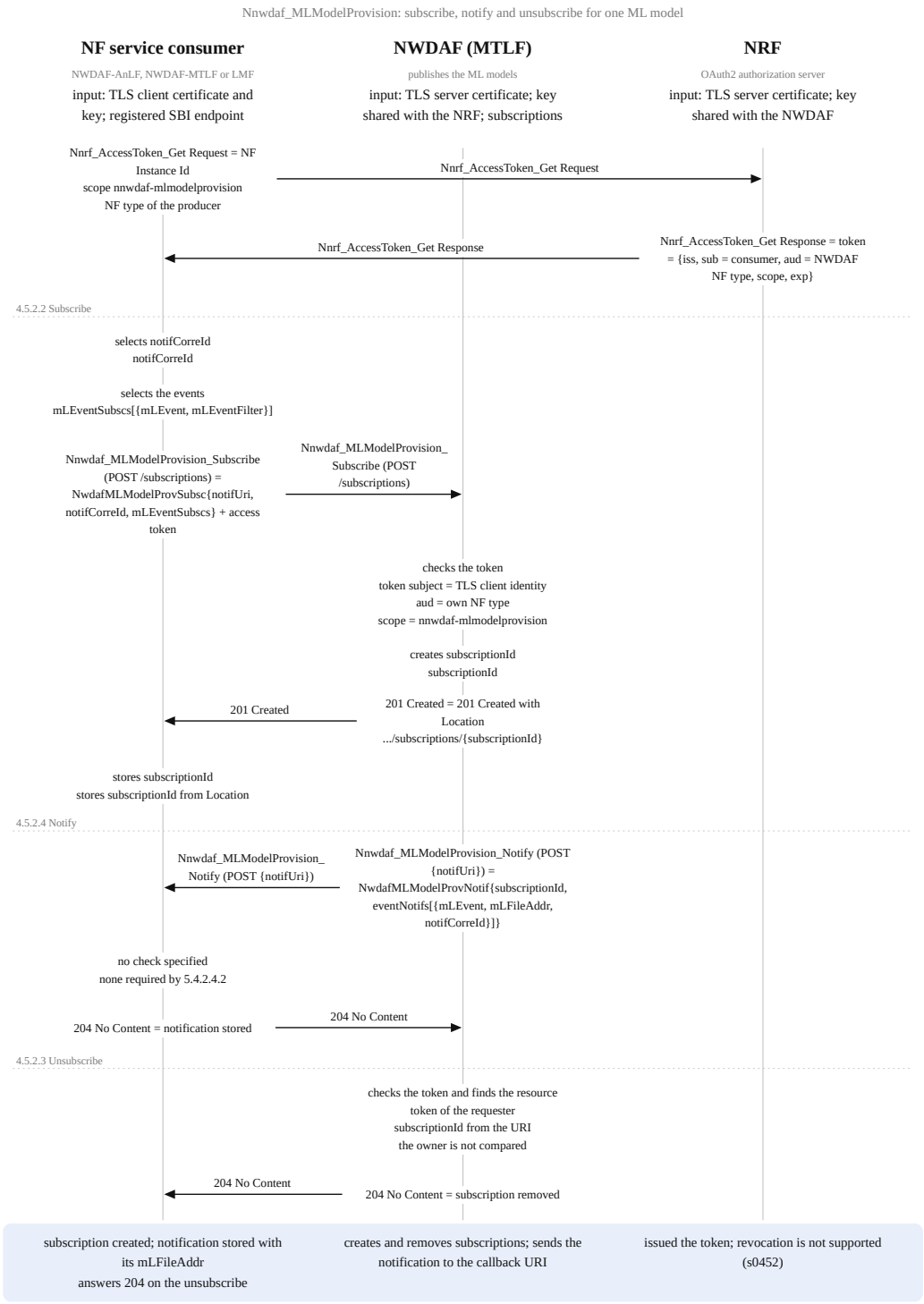



Figure 1. The protocol as specified.

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.

Within the analysed sub-flow the transport and OAuth2 parts hold: no party can forge another TLS endpoint, and every subscribe and unsubscribe the NWDAF executes is backed by an NRF token whose subject is the instance it sees over the connection. There are two findings. A registered NF that holds its own certificate and token can post a notification to another consumer's callback URI, and the consumer stores it with that NF's model file address and correlation identifier, because the notification operation is outside OAuth2 and the consumer is only required to store what it accepts. A consumer with a valid service-scope token that knows another consumer's subscriptionId can delete that subscription, because the API defines a single service scope and the unsubscribe never compares the requester with the owner. Both were checked as changes on the model: binding the notification to the producer and to the subscription, and comparing the owner on the unsubscribe, remove the attacks without losing the normal run or any other recorded answer. All nine targets have an answer for the attacker they name, and every proof except the one about the notification content has a control that fails; the remaining answers rest on the simplifications listed under limits, chiefly the idealised TLS channel and the monotonic subscription store.

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 registered NF can plant a model notification in another consumer's callback URI

| | |
|---|---|
| **Kind** | Weakness of the protocol as specified |
| **New** | Unclear: the documents state it in part, or could not be searched (ts-33-501:7892) |
| **Value** | medium |
| **Properties** | A03, A08, A03.7, A08.9 (section 3) |
| **Source text** | s0108, s0123, s0410 |
| **Shown by** | ProVerif attack queries `a03_nf`, `a08_nf`; Figures 2, 3, 4 and 5 |

**What happens.** The notification operation carries no authorization and the consumer is only required to store what it accepts, so the transport is its whole defence, and the transport proves which NF sent a message, not that it is the NWDAF. A malicious NF posts a NnwdafMLModelProvNotif-shaped body to the consumer's callback URI over its own TLS connection and the consumer stores it with the sender's notifCorreId and model file address. The NWDAF's own notification for the same subscription is unaffected and the consumer cannot tell the two apart.

**What the attacker needs.** A registered NF service consumer with its own TLS client certificate, its own NRF-issued access token for the service scope, and the callback URI the target consumer registered. The callback URI is modelled as derivable from the registered SBI endpoint; if it were unguessable, the attacker would need it to leak first.

**What the attacker gains.** The consumer stores the notification: the model file address it would fetch the ML model from, and the correlation identifier of its subscription, are then the values of the malicious NF. Model provisioning for that consumer is redirected to an attacker-chosen address, and the model identifier it records is not the one the NWDAF published.

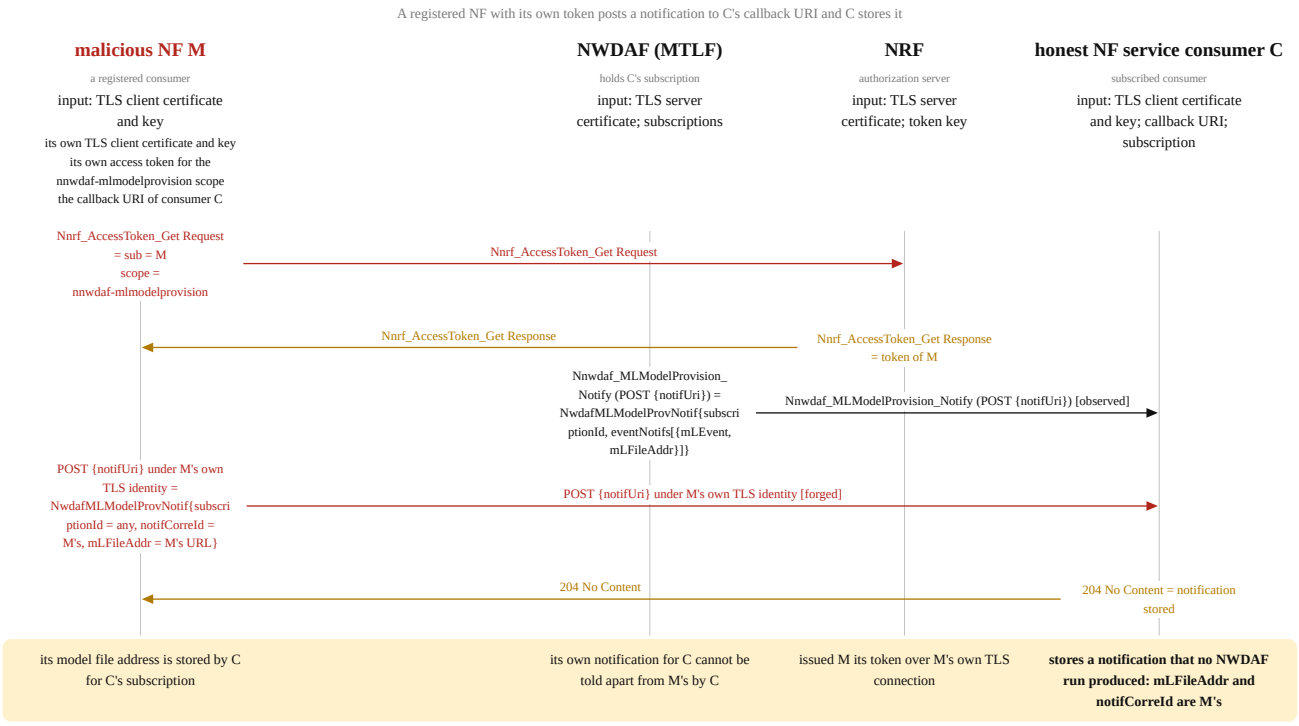



Figure 2. A registered NF with its own token posts a notification to C's callback URI and C stores it. Property A03; attack query a03_nf.

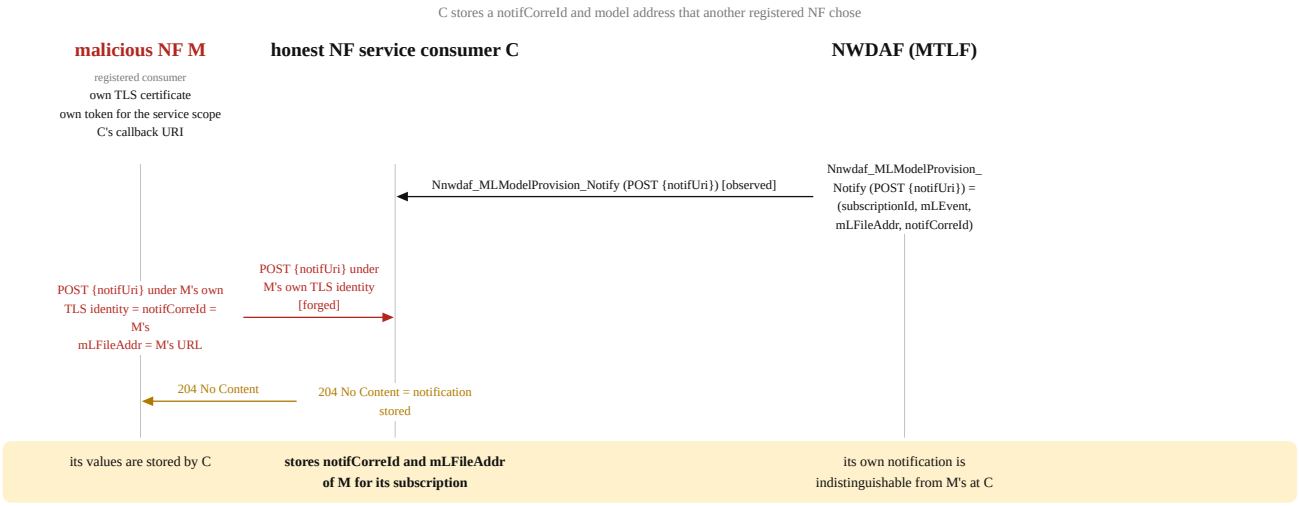



Figure 3. C stores a notifCorreId and model address that another registered NF chose. Property A08; attack query a08_nf.

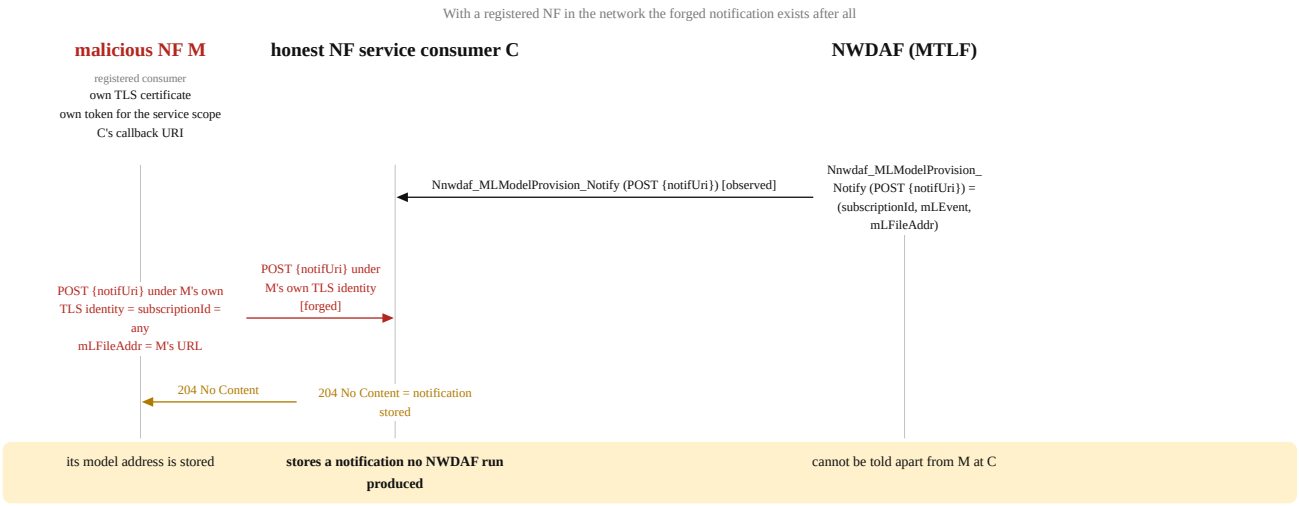



Figure 4. With a registered NF in the network the forged notification exists after all. Property A03.7; attack query a03_nf.

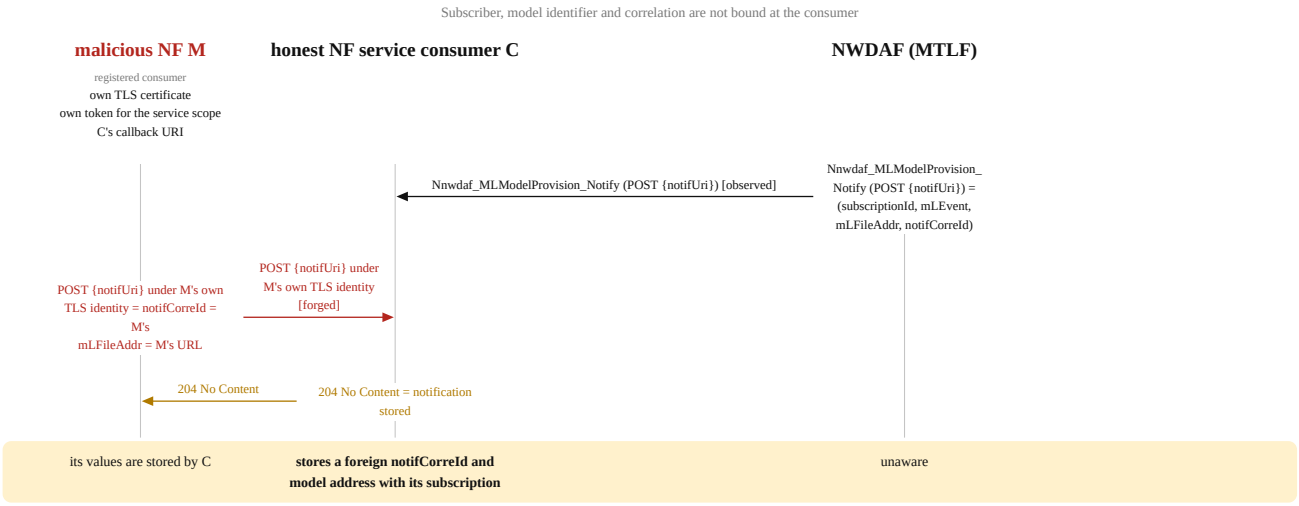



Figure 5. Subscriber, model identifier and correlation are not bound at the consumer. Property A08.9; attack query a08_nf.

**Proposed change.** Require the consumer, before storing a notification, to verify that it was sent by the NF instance that holds its subscription and that the subscriptionId it carries is the one returned in the Location header of its subscription, and to discard the request otherwise. Checked on the model with that check added: the two attack queries are verified there, the normal run is still found and every other recorded answer is unchanged. A stronger variant, which also helps when the subscription identifier is known to an attacker, is for the NWDAF to authorise the notification as it authorises the other operations instead of leaving the notification operation outside OAuth2.

**Check of the change with ProVerif.** The change was put into a copy of the model: The consumer verifies, before storing a notification, that it was sent by the NWDAF instance it subscribed with and that its subscriptionId is the one the NWDAF returned for its subscription, and discards the request otherwise. On the changed model ProVerif proves `a03_nf`: the attack is no longer possible there. The normal run of the protocol is still found. Of the other recorded answers, 23 still hold and 2 changed: `a08_nf` (A08, was attack) is now verified; `a08_nf` (A08.9, was attack) is now verified. This is a first check on the model: it shows that the change removes this attack there, not that it introduces no other problem. Changed model: `records/fixes/0048-fix-notif.pv`.

## 2.2 The unsubscribe is not bound to the owner of the subscription

| | |
|---|---|
| **Kind** | Weakness of the protocol as specified |
| **New** | Unclear: the documents state it in part, or could not be searched (ts-29-501:3967) |
| **Value** | medium |
| **Properties** | Z03, Z03.8 (section 3) |
| **Source text** | s0095, s0192, s0342, s0395 |
| **Shown by** | ProVerif attack query `z03_other`; Figures 6 and 7 |

**What happens.** The Nnwdaf_MLModelProvision API defines one service-level scope and no resource or operation scope, so a token grants every operation of the service to every consumer allowed to use it. The unsubscribe finds the subscription by the subscriptionId in the URI, checks the requester's token subject and scope, and never compares the requester with the consumer that created the subscription. A consumer that holds another consumer's subscriptionId therefore deletes that subscription and is answered 204 No Content.

**What the attacker needs.** Its own valid access token for the nnwdaf-mlmodelprovision service scope plus the subscriptionId of another consumer's subscription, which it obtains from an intermediary that handles the resource URI (an SCP in indirect communication, under the trust assumption of TS 33.501 5.9.2.4) or because the identifier is not unguessable.

**What the attacker gains.** The consumer's subscription is removed without its request, so it receives no further model notifications while it believes it is subscribed; a model update or a first model for its analytics then never arrives. The consumer learns nothing about who did it.

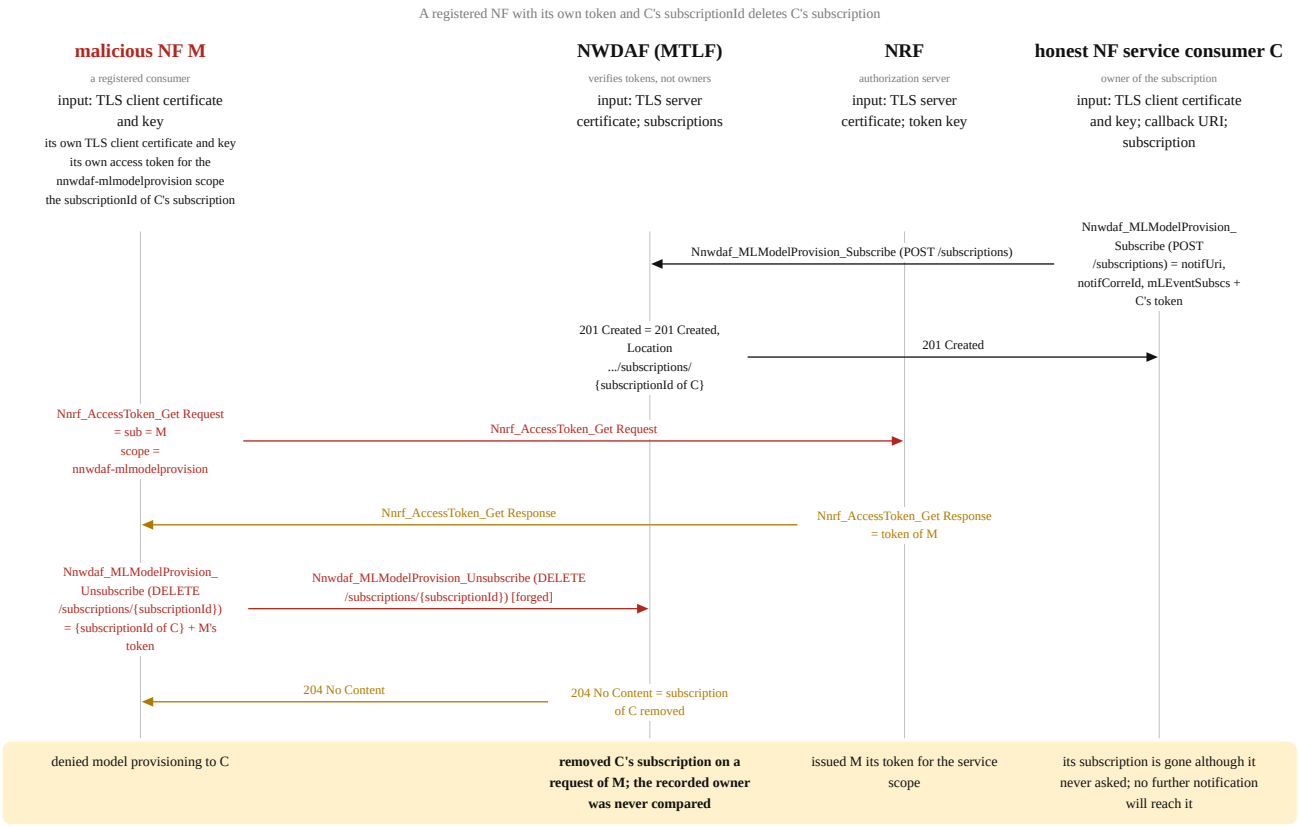



Figure 6. A registered NF with its own token and C's subscriptionId deletes C's subscription. Property Z03; attack query z03_other.

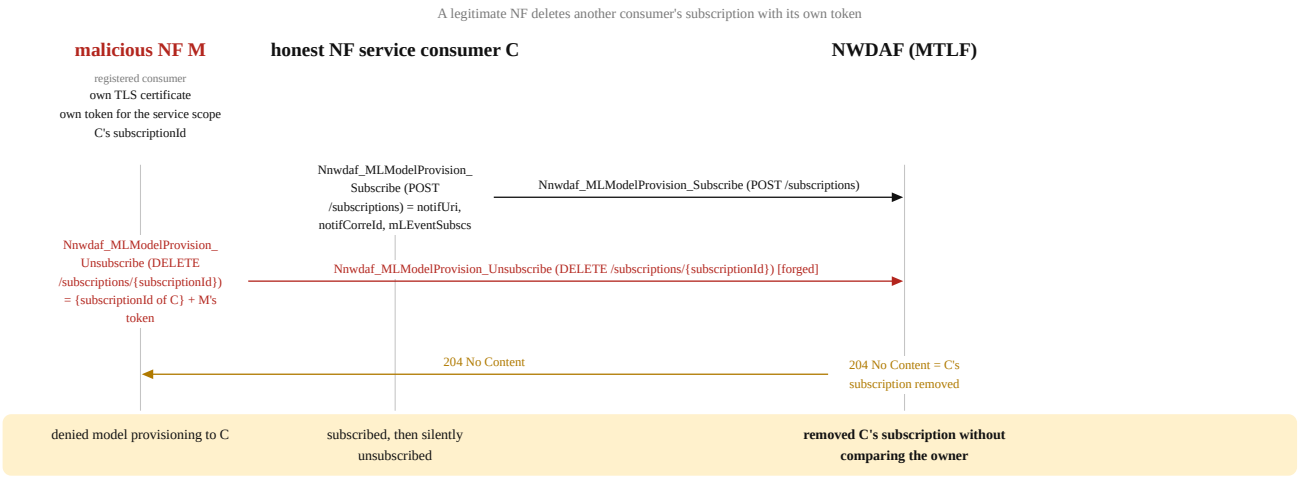



Figure 7. A legitimate NF deletes another consumer's subscription with its own token. Property Z03.8; attack query z03_other.

**Proposed change.** Require the NWDAF, on an unsubscribe, to compare the owner recorded for the subscription identified in the request URI with the requester authenticated by the access token subject and the TLS client certificate, and to refuse with an error answer when they differ. Checked on the model with that check added: the cross-owner delete query is verified there, the normal run is still found and every other recorded answer is unchanged. Alternatively define a resource-level scope for the subscription resource, as TS 29.501 4.10 recommends for APIs whose operations must not all be open to every consumer.

**Check of the change with ProVerif.** The change was put into a copy of the model: The NWDAF compares the owner recorded for the subscription identified by the subscriptionId in the request URI with the requester authenticated by the access token subject and the TLS client certificate, and refuses the unsubscribe with an error answer when they differ, instead of removing the subscription. On the changed model ProVerif proves `z03_other`: the attack is no longer possible there. The normal run of the protocol is still found. The other 25 recorded answers are unchanged. This is a first check on the model: it shows that the change removes this attack there, not that it introduces no other problem. Changed model: `records/fixes/0045-fix-owner.pv`.

No figure was given for these attacks of a finding: A03.7: a08_nf, A03: a08_nf, A08.9: a03_nf, A08: a03_nf.

# 3 Properties examined

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

| ID | Property | Compromise assumed | Result | Value | In the documents |
|---|---|---|---|---|---|
| A02 | **NWDAFauthenticates the consumer endpoint.** Every timeNWDAF（MTLFC must exist when accepting a subscription/update/cancellation request and claiming that the peer is a consumer CNWDAFA real run for the opposite end. | no compromise | Holds | low | yes |
| A03 | **Notification authenticity for the subscriber.** Every time a consumer accepts a notification and claims to be fromNWDAF, and must exist for S subscriptionsNWDAFRun a true notification for the same subscription S, andsubscriptionId、notifCorreId、notifUri, Model markings are consistent. | no compromise | Fails (1 attack) | medium | unclear |

| ID | Property | Compromise assumed | Result | Value | In the documents |
|---|---|---|---|---|---|
| A08 | **Notification fields bound to the subscription.** Every time a consumer accepts a notice,notifCorreId、notifUri, the model identification must be derived from the consumer's own subscription request, andNWDAFThe records are identical to the subscriptions. | no compromise | Fails (1 attack) | medium | unclear |
| Z01 | **Token authorization of subscription operations.** Every timeNWDAFSubmit a subscription to create, update or delete, must existNRFIssued for the consumer, subject corresponds to the customer's identity, AudienceNWDAF, scope contains nnwdaf-mlmodelprosision and has not expired a valid licence. | no compromise | Holds | low | yes |
| Z03 | **Subscription ownership beyond service scope.** Whenever NWDAF executes Subscription updates or deletions on the basis of a licence, the permission subject must be consistent with the identity of the applicant and the operated subscriptionId belongs to the subproject. | no compromise | Fails (1 attack) | medium | unclear |
| Z05 | **Cross-subscription isolation from a maliciousNF.** Other consumer C subscriptions and notifications remain confidential and authentic, subject to malicious consumer M controlling its own documents and its own subscriptions: M cannot obtain the content of C notifications, nor can it allowNWDAFSend C notifications to MnotifUri。 | no compromise | Holds | low | yes |
| A03.7 | **Notification authenticity under normalSBA.** In normalSBAUnder the end-point authentication encryption model, the notification to be accepted by the consumer must be responsiveNWDAFSynchronising "%s" | no compromise | Fails (1 attack) | medium | unclear |
| Z03.8 | **Independent basis for cross-subscription operations.** Whenever NWDAF executes Subscription updates or deletions on the basis of a licence, the permission subject must be consistent with the identity of the applicant and the operated subscriptionId belongs to the subproject. | no compromise | Fails (1 attack) | medium | unclear |
| A08.9 | **Binding of subscriber, model id and correlation.** Every time a consumer accepts a notice,notifCorreId、notifUri, the model identification must be derived from the consumer's own subscription request, andNWDAFThe records are identical to the subscriptions. | no compromise | Fails (1 attack) | medium | unclear |

The questions asked, and the analyst's answers:

| Question | Answer | Rests on |
|---|---|---|
| Q1: NormalSBANext round to notify if the forgeries have disappeared. | Yes for the network attacker: with only the network no forged notification can be stored, because the stored message is the one its TLS peer sent. No for a malicious legitimate NF: a registered NF with its own certificate and token posts a notification to the consumer's callback URI and the consumer stores it, so the forgery trace returns as soon as that attacker is considered. | A03.7, A03, A08.9 |
| Q2: LegalNFWhether cross-subscription has an independent regulatory basis and authentication prerequisite | No. TS 29.552 defines one scope for the whole service and no resource or operation scope, and neither its unsubscribe clause nor TS 33.501 gives a basis or an authentication precondition for acting on another consumer's subscription: the NWDAF checks the requester's token and finds the resource by its identifier, and the owner is never compared. With the identifier withheld, ownership rests only on the identifier. | Z03.8, Z03, Z01 |
| Q3: subscriber, model identifier, notion coordinationIDandURLTie | The transport binds the message to the NF that sent it, and the NWDAF binds the correlation identifier and the model address to the subscription it holds; the consumer binds nothing. A notification from any registered NF is stored with that NF's correlation identifier and model address, so the subscriber, the model identifier and the URL are not bound to each other at the consumer. | A08.9, A08, Z05 |

# 4 Limits of this analysis

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

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

- TLS is an idealised endpoint channel: pairwise keys derived from the two NF identities, no handshake, no sequence numbers, no rekeying. A recorded request can be replayed in the model, and no target rests on that.
- The download of the ML model from the address in the notification is not modelled, as the task excludes it; the model address is a symbolic value whose only property examined is that it stays inside the consumer's TLS channel.
- The subscription store is a monotonic table: a deletion is an event and the record is not removed, so nothing about what happens after an unsubscribe is asked.
- Token expiration is not modelled in time and token revocation is not modelled at all; TS 33.501 states that revocation is not supported.
- The callback URI of a consumer is modelled as derivable from its registered SBI endpoint, which the documents do not state; the notification injection attack rests on that choice.
- Only direct communication is modelled. Indirect communication through an SCP or SEPP, roaming and interconnect, which TS 33.501 treats separately and where there is no mutual authentication between consumer and producer, are left out.
- The PUT operation that updates a subscription, the immediate-report and failure-report branches, the 403 and 500 answers and the error handling of TS 29.552 5.4.7 are not modelled; the task scopes the analysis to the first round of subscribe, notify and unsubscribe for a single model.
- Tokens are MACs under a key shared by the NRF and the NWDAF, not JWS signatures; the model represents the checks the producer makes, not the signature scheme, and it says nothing about computational strength.
- The attacker that holds another consumer's subscriptionId is given that knowledge explicitly; the model does not show how it is obtained.

What the host records:

- Proved without a control query (Appendix C): Z05: z05_sec. A control is a variant of the property that must fail; without one, nothing shows that the property as written can fail in this model.

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

# Appendix A: Reading of each property

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

## A02: NWDAF authenticates the consumer endpoint

Every timeNWDAF（MTLFC must exist when accepting a subscription/update/cancellation request and claiming that the peer is a consumer CNWDAFA real run for the opposite end.

The NWDAF accepts a subscription or a deletion only from the NF instance it sees as its TLS peer: every subscription it creates carries the correlation identifier of that instance's own request, and every deletion is the request of the instance named in it. A request cannot be made to look as if it came from another NF, against the network alone as well as against another registered NF.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: This is the mutual authentication and token-to-certificate binding the documents already require (s0470, s0478); the analysis confirms it rather than finding anything new.
- Stated in the documents: yes. Where the documents say it: s0470, s0478
- Rests on: TLS is modelled as authenticated encryption between two NF identities, without the handshake and without sequence numbers, so a recorded request can be replayed in the model; no target rests on that.; The token subject and the TLS client identity are required to be equal, as TS 33.501 13.4.1.1.2 step 2 requires; the NRF only issues a token to the instance it is talking to.; The subscriptions are kept in a table that only grows.
- Open: Indirect communication through an SCP, where TS 33.501 notes there is no mutual authentication between consumer and producer, is not modelled.
- Next step: Model the SCP variant to see whether the binding still holds when the two endpoints do not authenticate each other.

## A03: Notification authenticity for the subscriber

Every time a consumer accepts a notice and claims that it comes fromNWDAF, and must exist for S subscriptionsNWDAFRun a true notification for the same subscription S, andsubscriptionId、 notifCorreId、 notifUri, Model markings are consistent.

For an attacker with the network alone no forged notification exists: what a consumer stores was sent, under its own endpoint identity, by the NF it sees as peer. For a registered NF that holds its own certificate and its own token the property fails: it posts a notification to the consumer's callback URI and the consumer stores it, so the stored notification is not the notification of the NWDAF that holds the subscription.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: TS 33.501 states that OAuth2 does not apply to the notification operation and TS 29.552 only requires the consumer to store the notification; neither states that a notification can be injected by another NF, nor requires the consumer to verify the sender.
- Stated in the documents: unclear. Source text it rests on: s0108, s0123, s0410
- Rests on: The consumer's callback URI is modelled as derived from its registered SBI endpoint, hence known to any NF; the documents do not say whether it is unguessable.; The consumer performs only the checks TS 29.552 states, that is none on the sender or on the correlation; the attack depends on that, and TS 33.501 5.9.2.1 requires

validation of incoming messages only in general terms.; The model file address is a fixed symbolic value; the download protocol is not modelled.

- Open: Whether a consumer is required by TS 33.501 5.9.2.1 to reject a notification for a subscription identifier it does not hold.
- Next step: Ask the 29.552 editors to state the validation a consumer must perform on a notification; the checked change in the report shows that binding the notification to the producer and to the subscription removes the attack.

### A08: Notification fields bound to the subscription

Whenever a consumer accepts a notice, the model identification of notifCorreId, notifUri must come from the consumer ' s own subscription request and be consistent with the subscription to the NWDAF record.

The values inside a stored notification are exactly the values of the message the consumer accepted, so a network attacker cannot substitute any of them. But the correlation identifier and the model address a consumer stores are not tied to its own subscription: another registered NF supplies its own values and the consumer stores them.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: The documents describe the notification correlation identifier as optional and say the consumer stores the notification, but never state that a consumer can be given another party's values, nor require a check.
- Stated in the documents: unclear. Source text it rests on: s0112, s0114, s0123
- Rests on: The callback URI is modelled as derivable from the registered SBI endpoint.; The consumer checks nothing about the notification, as TS 29.552 does not require it to.; The correlation identifier is only echoed by the NWDAF; nothing makes the consumer compare it.
- Open: Whether the consumer is expected to verify notifCorreId and the model identifier at all.
- Next step: Same as A03: state the checks of the consumer, then re-run the model with them.

### Z01: Token authorization of subscription operations

Every timeNWDAFSubmit a subscription to create, update or delete, must existNRFIssued for the consumer, subject corresponds to the customer's identity, AudienceNWDAF, scope contains nnwdaf-mlmodelprosision and has not expired a valid licence.

Before creating or deleting a subscription the NWDAF is assured by an NRF-issued token that the requester is the instance it sees over TLS, that the token is for this producer NF type and that its scope covers the service. A consumer without a token of its own cannot make the NWDAF act at all.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: The checks are exactly those TS 33.501 13.4.1.1.2 requires; the analysis confirms them rather than finding anything new.
- Stated in the documents: yes. Where the documents say it: s0478, s0480, s0483, s0485
- Rests on: Expiration is not modelled in time: the model shows the subject, audience and scope checks and the binding of the token to the TLS identity, not the expiration check of TS 33.501 13.4.1.1.2.; Tokens are modelled as MACs under a key shared by the NRF and the NWDAF rather than as JWS signatures, so the model represents the checks and not the signature scheme.; Only the producer NF type is used as audience; instance-level audiences and slice claims are not modelled.
- Open: Token expiry and the absence of revocation (TS 33.501 states revocation is not supported) are outside the model.

- Next step: A model with time or with a nonce sequence could ask whether an expired or revoked token is still accepted.

### Z03: Subscription ownership beyond service scope

Whenever NWDAF executes Subscription updates or deletions on the basis of a licence, the subproject of the licence must be consistent with the identity of the applicant and the operated subscriptionId belongs to the subproject.

The token binds the requester to its own TLS identity, so no consumer acts under another consumer's token; but nothing binds the subscription to its owner. A consumer with a valid service-scope token that knows another consumer's subscription identifier deletes that subscription, and the NWDAF answers 204 as if the owner had asked. Without the identifier the deletion does not happen.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: TS 29.501 4.10 warns that an API defined with a service-level scope only must not grant inappropriate write or delete access to undue consumers, and TS 29.552 defines exactly one scope for the whole API; neither document states that this API is affected.
- Stated in the documents: unclear. Source text it rests on: s0342, s0392, s0395
- Rests on: The attack needs the attacker to know another consumer's subscription identifier; the model gives it through a separate process, since the identifier is otherwise only sent inside the owner's TLS channel.; The subscription store is a monotonic table: the deletion is an event and the record is not removed, so no property about what happens after a deletion is asked.; The API defines one service-level scope and no resource or operation scope, as the model has it.
- Open: Whether subscription identifiers are confidential or unguessable in a deployment; the documents require guessing protection only for tokens and client credentials.
- Next step: Clarify in TS 29.552 whether an unsubscribe must be checked against the owner, or define a resource-level scope; the checked change in the report adds the ownership check.

### Z05: Cross-subscription isolation from a malicious NF

Other consumer C subscriptions and notifications remain confidential and authentic, provided that the malicious consumer M controls his own documents and his own subscriptions: M cannot obtain the content of the C notification, nor can it allowNWDAFSend C notifications to MnotifUri。

The NWDAF notifies only a subscription it created and only at the callback URI of that subscription's owner, so another NF cannot make it deliver a subscription's notification elsewhere, and the model address never leaves the TLS channel between the NWDAF and the holder. What another NF can still do is post its own message to the callback URI, which is the weakness recorded under A03 and A08.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: Mutual authentication and protection between NFs, and the notification being sent to the callback URI of the subscription, are what TS 33.501 5.9.2.1 and TS 29.552 5.4.2.4.2 require; the analysis confirms them.
- Stated in the documents: yes. Where the documents say it: s0108, s0123
- Rests on: The model address is a symbolic value in the notification; the download of the model file from it is not modelled.; The table of subscriptions only grows, so the effect of a deletion on later notifications is not represented.; The callback URI is modelled as derivable from the registered SBI endpoint.; The secrecy claim has no control: the model contains no process that reveals the consumer's TLS key, and adding one would falsify the other targets' answers.
- Open: Whether a notification continues to be sent, or is refused, after the subscription was deleted.

- Next step: A model that represents removal would settle the behaviour after unsubscribe.

### A03.7: Notification authenticity under normal SBA

It's normal.SBAUnder the end-point authentication encryption model, the notification to be accepted by the consumer must be responsiveNWDAFSynchronising "%s"

Under the endpoint-authenticated transport the forging trace of the external attacker is gone: it takes a registered NF with its own certificate and its own token to make the consumer store a notification the NWDAF never sent. The forgery therefore disappears for the network attacker and returns for a malicious legitimate NF.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: The documents state that notifications are outside OAuth2 but not that a consumer can be made to store a foreign notification.
- Stated in the documents: unclear. Source text it rests on: s0123, s0410
- Rests on: The callback URI is assumed derivable from the registered SBI endpoint.; The consumer stores the notification without checking the sender or the subscription identifier.
- Open: The validation a consumer owes under TS 33.501 5.9.2.1.
- Next step: Same as A03.

### Z03.8: Independent basis for cross-subscription operations

Whenever NWDAF executes Subscription updates or deletions on the basis of a licence, the subproject of the licence must be consistent with the identity of the applicant and the operated subscriptionId belongs to the subproject.

The second part of the question has a negative answer as far as the documents go: TS 29.552 defines one scope for the whole service and its unsubscribe clause binds nothing to the owner, so the only precondition for a deletion is the requester's own token and the subscription identifier. A legitimate NF that holds another consumer's identifier deletes that subscription; with the identifier withheld, ownership holds by the identifier alone.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: TS 29.501 4.10 names the risk of a service-scope-only API granting inappropriate delete access and requires it to be assessed; that the assessment was not made for this API is not stated anywhere.
- Stated in the documents: unclear. Source text it rests on: s0192, s0095, s0395
- Rests on: The identifier knowledge is an extra ability given to the attacker, not something the protocol reveals.; The model represents the event of deleting a subscription, not the removal of the record.; The error paths of TS 29.552 5.4.7 are not modelled.
- Open: Whether a resource-level scope or an ownership check exists in deployments.
- Next step: Add the ownership check or a resource-level scope to TS 29.552; the checked change in the report does the first.

### A08.9: Binding of subscriber, model id and correlation

Whenever a consumer accepts a notice, the model identification of notifCorreId, notifUri must come from the consumer ' s own subscription request and be consistent with the subscription to the NWDAF record.

The subscriber is bound to the message only by the transport: the stored values are the ones the NF that sent the message wrote. Nothing in the consumer binds the model identifier, the correlation identifier or the callback URI to its own subscription, so a foreign notification is stored with foreign values, and the model address it carries is the one the consumer would use.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value medium: The documents let the NWDAF echo notifCorreId and supply the model address but do not state that these are bound to the subscription, nor that a consumer can be given another party's values.
- Stated in the documents: unclear. Source text it rests on: s0112, s0114, s0123
- Rests on: The callback URI is modelled as derivable from the registered SBI endpoint.; The model address is symbolic; the download of the model is not modelled.; The consumer checks nothing, as TS 29.552 does not require it to.
- Open: Whether a consumer is expected to compare the notification with its subscription.
- Next step: Same as A08.

## Appendix C: Claims and prover answers

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

| target | level | bound | lemma | control | attacks | claims |
|---|---|---|---|---|---|---|
| A02 | proved | | a02_sub | checked | | 4 proved |
| A03 | proved | | a03_end | checked | a03_nf (depends_on_ambiguity) | 1 proved, 1 attack |
| A08 | proved | | a08_end | checked | a08_nf (depends_on_ambiguity) | 1 proved, 1 attack |
| Z01 | proved | | z01_sub | checked | | 4 proved |
| Z03 | proved | | z03_own | checked | z03_other (depends_on_ambiguity) | 2 proved, 1 attack |
| Z05 | proved | | z05_dest | 2 of 4 | | 4 proved |
| A03.7 | proved | | a03_end | checked | a03_nf (depends_on_ambiguity) | 1 proved, 1 attack |
| Z03.8 | proved | | z03_own | checked | z03_other (depends_on_ambiguity) | 2 proved, 1 attack |
| A08.9 | proved | | a08_end | checked | a08_nf (depends_on_ambiguity) | 1 proved, 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 |
|---|---|---|---|---|---|
| A02 | Every subscription creation the NWDAF accepts carries the notifCorreId of a subscribe request sent by the very NF instance the NWDAF sees as its TLS peer; no other party creates a subscription in that consumer's name. The attacker has the network only. | `network` | proved | a02_sub | checked |
| A02 | Every unsubscribe the NWDAF executes (for the owner or for another consumer) was sent by the same NF instance that the delete request names as its TLS peer, and carries that instance's subscription identifier. The attacker has the network only. | `network` | proved | a02_del | checked |
| A02 | Every subscription creation the NWDAF accepts carries the notifCorreId of a subscribe request sent by the NF instance it names as its TLS peer, also when another registered NF with its own certificate and its own service-scope token is present. | `malicious_nf` | proved | a02_sub | checked |
| A02 | Every unsubscribe the NWDAF executes was sent by the NF instance the request names over TLS, also when another registered NF with its own certificate and token is present; a delete request of the malicious NF appears as a request of the malicious NF and not of the consumer. | `malicious_nf` | proved | a02_del | checked |
| A03 | A notification stored by a consumer corresponds to a real Nnwdaf_MLModelProvision_Notify run of the NWDAF for the same subscription; the attacker holds its own NF certificate, its own client-credentials token for the service scope, and knows the consumer's callback URI. | `malicious_nf` | attack (depends_on_ambiguity) | a03_nf | |
| A03 | A notification a consumer stores was produced, under its own TLS endpoint identity, by the NF the consumer sees as its peer: with the network alone no message can be stored that some endpoint did not itself send. | `network` | proved | a03_end | checked |

| target | claim | attacker | answer | query | control |
|---|---|---|---|---|---|
| A08 | The notifCorreId and the model address stored by consumer C come from C's own subscription request and from the NWDAF that holds it; the attacker holds its own NF certificate, its own token for the service scope and knows C's callback URI. | `malicious_nf` | attack (depends_on_ambiguity) | a08_nf | |
| A08 | The subscriptionId, notifCorreId and model address stored by a consumer are exactly the values the peer that sent the message wrote into it; the network attacker cannot substitute any of them. | `network` | proved | a08_end | checked |
| Z01 | Every subscription the NWDAF creates is backed by an access token that the NRF issued to that same NF instance: the token subject matches the instance the NWDAF sees over TLS, its audience covers the producer NF type and its scope covers the service. The attacker has the network only. | `network` | proved | z01_sub | checked |
| Z01 | Every unsubscribe the NWDAF executes is backed by an access token that the NRF issued to the same NF instance, with the subject matching the TLS peer, the audience covering the producer and the scope covering the service. The attacker has the network only. | `network` | proved | z01_del | checked |
| Z01 | Also when a registered NF with its own certificate and its own token is present, every subscription the NWDAF creates is backed by a token the NRF issued to that same instance. | `malicious_nf` | proved | z01_sub | checked |
| Z01 | Also when another registered NF is present, every unsubscribe the NWDAF executes is backed by a token the NRF issued to the same instance, so the malicious NF acts only under its own token. | `malicious_nf` | proved | z01_del | checked |
| Z03 | A subscription is deleted only by the consumer that owns it; a delete for another consumer's subscription happens only when the attacker already holds that subscriptionId. The attacker has the network only. | `network` | proved | z03_own | checked |
| Z03 | The malicious NF, holding its own certificate and its own service-scope token, deletes another consumer's subscription only if it holds that subscriptionId; the service scope alone gives it no access to a subscription it does not know the identifier of. | `malicious_nf` | proved | z03_own | checked |
| Z03 | The NWDAF relates an unsubscribe to the consumer that owns the subscription, not merely to a consumer allowed to use the service: a malicious NF holding its own token and another consumer's subscriptionId must not be able to delete that subscription. | `subscription_id_known` | attack (depends_on_ambiguity) | z03_other (the control of z03_own) | |
| Z05 | The model address a notification carries is never learned by an attacker with the network only; it reaches the subscription holder inside its own TLS channel and nowhere else. | `network` | proved | z05_sec | none |
| Z05 | A malicious NF holding its own certificate and token obtains neither another consumer's subscription nor the content of its notification: the model address travels only inside the consumer's TLS channel with the NWDAF. | `malicious_nf` | proved | z05_sec | none |
| Z05 | The NWDAF sends a notification only for a subscription it created and only to that subscription's owner over the owner's callback URI, so nothing on the network makes it deliver a subscription's notification to another party. | `network` | proved | z05_dest | checked |
| Z05 | A malicious NF cannot make the NWDAF deliver a notification for a subscription to a callback URI other than the owner's, nor obtain the subscription of another consumer from the NWDAF. | `malicious_nf` | proved | z05_dest | checked |

| target | claim | attacker | answer | query | control |
|---|---|---|---|---|---|
| A03.7 | A notification accepted by a consumer corresponds to a real NWDAF notification run for the same subscription; the attacker is a registered NF with its own certificate and token that knows the consumer's callback URI. | `malicious_nf` | attack (depends_on_ambiguity) | a03_nf | |
| A03.7 | Under the SBA endpoint-authenticated transport a notification a consumer stores was sent under its own identity by the NF it sees as peer, so with the network alone the trace of a forged notification does not exist. | `network` | proved | a03_end | checked |
| Z03.8 | There is no independent basis for one NF to update or delete another consumer's subscription: a subscription is deleted only by its owner unless the subscriptionId is already held by the attacker. | `network` | proved | z03_own | checked |
| Z03.8 | A legitimate NF holding its own service-scope token has no basis to act on another consumer's subscription: the only precondition the NWDAF checks is the token of the requester, so ownership rests on the subscriptionId not being known. | `malicious_nf` | proved | z03_own | checked |
| Z03.8 | The NWDAF must not let a consumer allowed to use the service delete a subscription it does not own, even when that consumer presents its own valid token and names the subscriptionId. | `subscription_id_known` | attack (depends_on_ambiguity) | z03_other (the control of z03_own) | |
| A08.9 | The model identifier and notifCorreId in a notification C stores belong to C's own subscription and come from the NWDAF that holds it; the attacker is a registered NF with its own certificate and token that knows C's callback URI. | `malicious_nf` | attack (depends_on_ambiguity) | a08_nf | |
| A08.9 | The notifCorreId, subscriptionId and model address stored by C are exactly the values the NF it sees as peer sent; the network attacker cannot substitute any of them. | `network` | proved | a08_end | checked |

# Appendix D: How the analysis was run

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

ProVerif+ Agent session session-20261115T084034, 2026-11-15. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 66, prover 8 s, wall 19 min, cost 0.157 USD.

ProVerif: 132 proofs run, 132 decided, 0 without an answer, 5 s, 49 answers recorded.

## What was analysed

| | |
|---|---|
| **Protocol** | Review ML model provision with SBA isolation |
| **Version** | 19.8.0 |
| **Analysed** | First round of reviewNnwdaf_MLModelProvision: Single modelURLSubscriptions/notices/cancellations; Pre-standard end-point authentication encryption model. External assailants can't fake anything else.TLSEndpoints; An additional malice consumer who owns his own token. It's forbidden to do so.SBIIt's clear that the assailants are normal.TLSDeployment; modelURLSeparated from actual downloads, undefined download protocols do not include conclusions. |
| **Not analysed** | Symbolic models do not support the calculation of algorithms as safe or as resistant to quantum security; The trajectory of the attack is established only under clear assumptions and is not directly equivalent to a normative loophole. Each nature must be subject to accessibility/proportional controls. FixedRel-19。 Distinguishing between an outside cyber attacker and an ill-intentioned and legitimate person holding his own evidence.NF； TLSAuthenticate encryption modeling by endpoint, validNFCan not open messageNFIt's...TLSSession.OAuth2Subject/udience/scope/expiry andTLS(a) The identity is bound in accordance with the attached baseline; Service scope does not automatically equal ownership of resources. Single-modules cannot simulate real deletions, and the life cycle of a limited state must be clearly defined. Targets are specified.APISubprocess, not complete 5G secure. |
| **Source text** | 659 numbered passages (338873 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 | 4.5 | Nnwdaf_MLModelProvision Service |
| main | 4.5.1 | Service Description |
| main | 4.5.1.1 | Overview |
| main | 4.5.1.2 | Service Architecture |
| main | 4.5.1.3 | Network Functions |
| main | 4.5.1.3.1 | Network Data Analytics Function (NWDAF) |
| main | 4.5.1.3.2 | NF Service Consumers |
| main | 4.5.2 | Service Operations |
| main | 4.5.2.1 | Introduction |
| main | 4.5.2.2 | Nnwdaf_MLModelProvision_Subscribe service operation |
| main | 4.5.2.2.1 | General |

| Document | Section | Title |
|---|---|---|
| main | 4.5.2.2.2 | Subscription for event notifications |
| main | 4.5.2.2.3 | Update subscription for event notifications |
| main | 4.5.2.3 | Nnwdaf_MLModelProvision_Unsubscribe service operation |
| main | 4.5.2.3.1 | General |
| main | 4.5.2.3.2 | Unsubscribe from event notifications |
| main | 4.5.2.4 | Nnwdaf_MLModelProvision_Notify service operation |
| main | 4.5.2.4.1 | General |
| main | 4.5.2.4.2 | Notification about subscribed event |
| main | 5.4 | Nnwdaf_MLModelProvision Service API |
| main | 5.4.1 | Introduction |
| main | 5.4.2 | Usage of HTTP |
| main | 5.4.2.1 | General |
| main | 5.4.2.2 | HTTP standard headers |
| main | 5.4.2.2.1 | General |
| main | 5.4.2.2.2 | Content type |
| main | 5.4.2.3 | HTTP custom headers |
| main | 5.4.3 | Resources |
| main | 5.4.3.1 | Resource Structure |
| main | 5.4.3.2 | Resource: NWDAF ML Model Provision Subscriptions |
| main | 5.4.3.2.1 | Description |
| main | 5.4.3.2.2 | Resource definition |
| main | 5.4.3.2.3 | Resource Standard Methods |
| main | 5.4.3.2.3.1 | POST |
| main | 5.4.3.2.4 | Resource Custom Operations |
| main | 5.4.3.3 | Resource: Individual NWDAF ML Model Provision Subscription |
| main | 5.4.3.3.1 | Description |
| main | 5.4.3.3.2 | Resource definition |
| main | 5.4.3.3.3 | Resource Standard Methods |
| main | 5.4.3.3.3.1 | PUT |
| main | 5.4.3.3.3.2 | DELETE |
| main | 5.4.3.3.4 | Resource Custom Operations |
| main | 5.4.4 | Custom Operations without associated resources |
| main | 5.4.5 | Notifications |
| main | 5.4.5.1 | General |
| main | 5.4.5.2 | Event Notification |
| main | 5.4.5.2.1 | Description |

| Document | Section | Title |
|---|---|---|
| main | 5.4.5.2.2 | Operation Definition |
| main | 5.4.6 | Data Model |
| main | 5.4.6.1 | General |
| main | 5.4.6.2 | Structured data types |
| main | 5.4.6.2.1 | Introduction |
| main | 5.4.6.2.2 | Type NwdafMLModelProvSubsc |
| main | 5.4.6.2.3 | Type MLEventSubscription |
| main | 5.4.6.2.4 | Void |
| main | 5.4.6.2.5 | Type NwdafMLModelProvNotif |
| main | 5.4.6.2.6 | Type MLEventNotif |
| main | 5.4.6.2.7 | Type FailureEventInfoForMLModel |
| main | 5.4.6.2.8 | Type MLModelAddr |
| main | 5.4.6.2.9 | Void |
| main | 5.4.6.2.10 | Void |
| main | 5.4.6.2.11 | Type MLRepEventCondition |
| main | 5.4.6.2.12 | Type InputDataInfo |
| main | 5.4.6.2.13 | Type ModelProvisionParamsExt |
| main | 5.4.6.2.14 | Type AdditionalMLModelInformation |
| main | 5.4.6.2.15 | Type MLModelAdrf |
| main | 5.4.6.2.16 | Type TrainInputDataInfo |
| main | 5.4.6.2.17 | Type InferenceDataForModelTrain |
| main | 5.4.6.2.18 | Type VflInfo |
| main | 5.4.6.3 | Simple data types and enumerations |
| main | 5.4.6.3.1 | Introduction |
| main | 5.4.6.3.2 | Simple data types |
| main | 5.4.6.3.3 | Enumeration: FailureCode |
| main | 5.4.6.3.4 | Enumeration: MLModelMetric |
| main | 5.4.6.3.5 | Enumeration: VflTrainingStatus |
| main | 5.4.6.3.6 | Enumeration: PositioningCase |
| main | 5.4.7 | Error handling |
| main | 5.4.7.1 | General |
| main | 5.4.7.2 | Protocol Errors |
| main | 5.4.7.3 | Application Errors |
| main | 5.4.8 | Feature negotiation |
| main | 5.4.9 | Security |
| ts-29-501 | 4.10 | Scopes definition for OAuth2.0 access token |

| Document | Section | Title |
|---|---|---|
| ts-33-501 | 13.4 | Authorization of NF service access |
| ts-33-501 | 13.4.1 | OAuth 2.0 based authorization of Network Function service access |
| ts-33-501 | 13.4.1.0 | General |
| ts-33-501 | 13.4.1.1 | Service access authorization within the PLMN |
| ts-33-501 | 13.4.1.1.1 | OAuth 2.0 roles |
| ts-33-501 | 13.4.1.1.2 | Service Request Process |
| ts-33-501 | 13.4.1.1.3 | Access token requests in deployments with several NRFs |
| ts-33-501 | 13.4.1.1A | Service access authorization in interconnect scenarios |
| ts-33-501 | 13.4.1.2 | Service access authorization in roaming scenarios |
| ts-33-501 | 13.4.1.2.1 | OAuth 2.0 roles |
| ts-33-501 | 13.4.1.2.2 | Service Request Process |
| ts-33-501 | 13.4.1.3 | Service access authorization in indirect communication scenarios |
| ts-33-501 | 13.4.1.3.1 | Authorization for indirect communication without delegated discovery procedure |
| ts-33-501 | 13.4.1.3.1.1 | With mutual authentication between NF Service Consumer and NRF at the transport layer |
| ts-33-501 | 13.4.1.3.1.2 | Without mutual authentication between NF and NRF at the transport layer |
| ts-33-501 | 13.4.1.3.2 | Authorization for indirect communication with delegated discovery procedure |
| ts-33-501 | 13.4.1.4 | Service access authorization in inter NF mobility scenario |
| ts-33-501 | 13.4.1.5 | Service access authorization in interconnect and roaming scenarios with indirect communication |
| ts-33-501 | 13.4.1.5.1 | General |
| ts-33-501 | 13.4.1.5.2 | Authorization with NF selection at source network |
| ts-33-501 | 13.4.1.5.2.1 | General |
| ts-33-501 | 13.4.1.5.2.2 | Authorization for indirect communication with or without delegated discovery procedure |
| ts-33-501 | 13.4.1.5.3 | Authorization with NF selection at target network |
| ts-33-501 | 13.4.1.5.3.1 | General |
| ts-33-501 | 13.4.1.5.3.2 | Authorization for indirect communication without delegated discovery procedure |
| ts-33-501 | 13.4.1.5.3.3 | Authorization for indirect communication with delegated discovery procedure |

The process of the review as the source step listed it in the documents:Nnwdaf_MLModelProvision: Single modelURLSubscriptions/notices/cancellations; Pre-standard end-point authentication encryption model. External assailants can't fake anything else.TLSEndpoints; An additional malice consumer who owns his own token. It's forbidden to do so.SBIIt's clear that the assailants are normal.TLSDeployment; modelURLSeparated from actual downloads, undefined download protocols do not include conclusions.

## The properties against the documents

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

| ID | Name | About | What the documents say | Where |
|---|---|---|---|---|
| A02 | NWDAF authenticates the consumer endpoint | The NWDAF containing MTLF is assured that a subscribe or unsubscribe request it accepts was sent by the NF instance it names as its peer, whose identity is taken from the TLS client certificate. | TS 33.501 13.4.1.1.2 step 2 requires the producer to check the token subject against the NF Instance ID in the consumer's TLS client certificate, and 5.9.2.1 requires mutual authentication for direct communication between consumer and producer. | s0470, s0478, s0399 |
| A03 | Notification authenticity for the subscriber | The NF service consumer is assured that a notification it stores was sent by the NWDAF holding its subscription and belongs to that subscription. | The consumer is only told to store the accepted notification and answer 204 (s0123); TS 33.501 13.4.1.0 says the OAuth2 framework does not apply to the notification operation (s0410); TS 33.501 5.9.2.1 requires each NF to validate all incoming messages against the protocol specification and network state (ts-33-501:2875). | ts-33-501:2875, ts-33-501:7892, s0108, s0123, s0410 |
| A08 | Notification fields bound to the subscription | The values the consumer stores from a notification (notifCorreId, notifUri, model identifier and model file address) are the ones belonging to its own subscription request. | TS 29.552 5.4.2.4.2 lets the NWDAF echo notifCorreId (s0114) and carries the model address mLFileAddr (s0112); the consumer is only required to store the notification and answer 204 (s0123), while TS 33.501 5.9.2.1 requires each NF to validate incoming messages against network state (ts-33-501:2875). | s0112, s0114, s0123 |
| Z01 | Token authorization of subscription operations | The NWDAF is assured, by an access token issued by the NRF, that the consumer is allowed to perform the subscribe or unsubscribe operation: subject matches the requesting instance, audience covers this producer, scope covers the service, and the token has not expired. | TS 29.552 5.4.9 (s0339, s0340) makes access authorized by OAuth2 with the NRF as authorization server; TS 33.501 13.4.1.1.2 step 2 lists the checks on issuer, integrity, subject against the TLS certificate, audience, scope and expiration (s0472-s0485). | s0339, s0340, s0342, s0478, s0480, s0483, s0485 |
| Z03 | Subscription ownership beyond service scope | The NWDAF relates an unsubscribe to the consumer that owns the subscription, not merely to a consumer that is allowed to use the service. | TS 29.501 4.10 says a 5GC API defined with a service-level scope only grants all its service operations to all consumers allowed to access the service and that it must be ensured this does not grant inappropriate write or delete access to undue consumers (s0392, s0395); TS 29.552 5.4.9 defines one scope for the entire API and no resource or operation scope (s0342), and nothing in the unsubscribe clause binds the request to the owner (s0095-s0098). | s0342, s0395, s0095, s0339 |
| Z05 | Cross-subscription isolation from a malicious NF | The honest NF service consumer C: its subscription and its notification content stay with it when another NF in the same network, holding its own credentials, is malicious. | TS 33.501 5.9.2.1 and 5.9.2.4 require mutual authentication and confidentiality, integrity and replay protection between NFs and through the SCP, which the documents state as a trust assumption (ts-33-501:2913, ts-33-501:2921); the notification operation is outside OAuth2 (s0410) and the consumer only stores what it receives (s0123). | s0108, s0123, s0410, s0342 |
| A03.7 | Notification authenticity under normal SBA | Under the SBA endpoint-authenticated transport, the consumer is assured that a notification it accepts corresponds to a real notification run of the NWDAF for the same subscription. | Same as A03: the notification operation is outside OAuth2 (s0410) and the consumer is only told to store the notification (s0123); mutual authentication is required for direct communication (ts-33-501:2871). | s0108, s0123, s0410 |
| Z03.8 | Independent basis for cross-subscription operations | Whether the NWDAF has an independent specification basis and an authentication precondition for an NF to update or delete a subscription belonging to another consumer. | TS 29.501 4.10 requires the API to be assessed for such a risk when only a service-level scope exists (s0392, s0395); TS 29.552 5.4.9 defines exactly one service-level scope (s0342) and the unsubscribe clause states no ownership check (s0095-s0098). | s0342, s0392, s0395, s0095 |

| ID | Name | About | What the documents say | Where |
|---|---|---|---|---|
| A08.9 | Binding of subscriber, model id and correlation | The notifCorreId, notifUri and model identifier in a stored notification belong to that consumer's own subscription request rather than to another subscription or to another NF's choice. | Same as A08: the NWDAF may echo notifCorreId (s0114) and supplies mLFileAddr (s0112); the consumer is only required to store the notification (s0123); TS 33.501 5.9.2.1 requires validation of incoming messages against network state (ts-33-501:2875). | s0112, s0114, s0123 |

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

## The security statements of the documents, paragraph by paragraph

The host listed the 398 paragraphs of the sections in which the documents say what the protocol protects and against what, and the analyst said of each what it made of it: a property, an attacker, or why neither. Taken up: 24. Not taken: 374. Which paragraphs are listed follows from the layout of the standard; where each went is the analyst's judgement.

| Paragraphs | Where | Taken up as | Why not |
|---|---|---|---|
| P001 to P002 | rfc6749:2778-2786 | No goal stated | Introductory scope of the OAuth2 security considerations and a pointer to a separate threat-model document; it states no property of this protocol. |
| P003 to P006 | rfc6749:2790-2823 | Z01 | |
| P007 | rfc6749:2827-2829 | A02, Z01, `malicious_nf` | |
| P008 | rfc6749:2831-2838 | Z01 | |
| P009 to P010 | rfc6749:2840-2850 | Outside the scope | Resource-owner authentication and repeated authorization requests belong to the authorization-code profile; this API uses the client-credentials grant with no resource owner. |
| P011 | rfc6749:2859-2865 | Z01, `malicious_nf` | |
| P012 | rfc6749:2867-2868 | Outside the scope | About the implicit grant, which this API does not use. |
| P013 | rfc6749:2870-2872 | Z01 | |
| P014 | rfc6749:2874-2877 | Z03, `malicious_nf` | |
| P015 | rfc6749:2879-2881 | Z01 | |
| P016 to P033 | rfc6749:2885-3003 | Outside the scope | Refresh tokens, authorization codes, redirection URIs and resource-owner password credentials are parts of OAuth2 grants this API does not use. |
| P034 | rfc6749:3013-3015 | Z01 | |
| P035 | rfc6749:3017-3019 | No goal stated | Advice about what may be put inside state and scope parameters, not a property of this exchange. |

| Paragraphs | Where | Taken up as | Why not |
|---|---|---|---|
| P036 | rfc6749:3023-3028 | A02 | |
| P037 to P038 | rfc6749:3032-3038 | Z01 | |
| P039 | rfc6749:3040-3041 | Outside the scope | About credentials intended for end-user usage; this API has no end-user credentials. |
| P040 to P062 | rfc6749:3045-3210 | Outside the scope | Phishing, CSRF, clickjacking, code injection, open redirectors and impersonation in the implicit grant concern browser-based OAuth2 clients; the SBI exchange analysed here has no user agent and no redirection to clients. |
| P063 to P213 | ts-33-501:2491-2853 | Outside the scope | Access-stratum, RAN, subscriber-privacy and key requirements of TS 33.501 clauses 5.1 to 5.8 concern the UE, gNB, UDM and AUSF; the task scopes the analysis to the Nnwdaf_MLModelProvision API sub-flow between NFs. |
| P214 | ts-33-501:2859-2859 | No goal stated | An assumption about trust zones that refers the reader to clauses 5.9.2 to 5.9.4; it states no property of its own. |
| P215 to P217 | ts-33-501:2865-2869 | Outside the scope | About NF service registration and discovery, not about the ML model provision exchange. |
| P218 | ts-33-501:2871-2871 | A02, `malicious_nf` | |
| P219 | ts-33-501:2873-2873 | `subscription_id_known` | |
| P220 | ts-33-501:2875-2875 | A03, A08 | |
| P221 to P224 | ts-33-501:2879-2885 | Z01 | |
| P225 | ts-33-501:2887-2887 | No goal stated | A lead-in sentence announcing the list of NRF requirements. |
| P226 to P227 | ts-33-501:2889-2891 | Z01 | |
| P228 to P234 | ts-33-501:2895-2907 | Outside the scope | About the NEF and the Application Function, which are not parties of this API sub-flow. |
| P235 to P239 | ts-33-501:2911-2919 | Outside the scope | About mutual authentication and protection of direct SCP links; the model of this API uses direct communication without an SCP, which is the configuration the task fixes. |
| P240 | ts-33-501:2921-2921 | `subscription_id_known` | |
| P241 | ts-33-501:2923-2923 | Outside the scope | About the 3gpp-Sbi-Originating-Network-Id header, which is not part of the modelled exchange. |

| Paragraphs | Where | Taken up as | Why not |
|---|---|---|---|
| P242 to P314 | ts-33-501:2929-3081 | Outside the scope | Interconnection, SEPP, N32-f and IPUPS requirements concern roaming and interconnect, which the task leaves out. |
| P315 to P318 | ts-33-501:3085-3091 | Outside the scope | Requirements on the 3gpp-Sbi-Originating-Network-Id header; not a property of the subscribe, notify and unsubscribe exchange. |
| P319 to P337 | ts-33-501:3097-3141 | Outside the scope | Signalling monitoring and security configurability are network-management abilities outside the analysed API sub-flow. |
| P338 to P369 | ts-33-501:3149-3215 | The prover cannot be asked | Ciphering and integrity algorithm identifiers and algorithm negotiation: the strength of a primitive and the choice of algorithm cannot be asked of ProVerif, and the task excludes computational strength from the scope. |
| P370 to P398 | ts-33-501:3219-11901 | Outside the scope | Requirements on the 5G-RG, NSSAAF, N3, EAP-TLS, IAB, MBSF and MBSTF concern other parts of the 5G system than this API sub-flow. |

## 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: 6. Left out: 30. 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/evidence confidentiality | Outside the scope | The long-term secrets are the TLS end-entity private keys and the NRF token key; their confidentiality is the TLS and OAuth2 baseline the task fixes as given (endpoint authentication), not this API sub-flow. |
| K02 | Load and End-to-End Confidentiality | Z05 | |
| K03 | Indistinguishability of confidentiality/secretity | Not applicable | The protocol hides no chosen secret value; NF identities are public and there is no secret between two candidate values to distinguish. |
| K04 | Key uses, directional statements separated | Not applicable | The API establishes no key hierarchy and derives no key of its own; transport keys belong to TLS. |
| D01 | Information integrity and authenticity of data sources | A02, A03 | |
| A01 | (Aliveness) | A02 | |
| A04 | Single-shot Consistency Authentication | A03 | |
| A05 | Secret Key Authentication | Not applicable | No key is established by this protocol. |
| A06 | Visible Key Confirmation | Not applicable | No key agreement, so no key confirmation is claimed; the 204 answer of a notification is an HTTP acknowledgement, not a key confirmation. |
| A07 | Identity - Certificate - Key binding | Z01, Z03 | |
| R01 | Certification/challenging novelty | Not applicable | The specification claims no challenge or freshness mechanism for authentication; the token expiration is a validity limit, handled by Z01. |
| R02 | Message Resume | The prover cannot be asked | Replay protection is delegated to the TLS transport and the endpoint-channel model has no sequence numbers, so a ProVerif answer about replaying a protected request would be an artefact of the model. |

| Entry | Property of the catalogue | Examined as | Why not |
|---|---|---|---|
| R03 | Operations at most once | The prover cannot be asked | At-most-once semantics of subscribe, like replay, rest on transport properties the endpoint-channel model does not represent; the specification claims none. |
| R04 | Message acceptance order/window security | Not applicable | The API has no ordering or acceptance window of its own. |
| R05 | Anti-degrading/security strategy maintained | Not applicable | Feature negotiation in TS 29.552 clause 5.4.8 negotiates optional functionality, not a security level or suite that could be lowered. |
| C01 | Forward confidentiality (FS) | Not applicable | No session key is established by this protocol. |
| C02 | Anti-key disclosure fake (KCI) | Not applicable | No authenticated key exchange is performed here; TLS provides it and is modelled as endpoint authentication. |
| C03 | Known key security/dissemination quarantine | Not applicable | There is no multi-session key graph in this API; each exchange uses the transport credentials of its two endpoints. |
| C04 | Temporary secret/partial state leaks resilience | Not applicable | The API keeps no ephemeral key or partial state whose leakage could be tolerated; the documents claim none. |
| C05 | Security restored after leak (PCS) | Not applicable | There is no rekeying or ratchet in this API. |
| C06 | Enhanced authentication after disclosure of a certificate of registration | Not applicable | No PAKE or enhanced registration record is used; client authentication is by TLS certificate. |
| C07 | Anti-offline dictionary attack | Not applicable | The protocol uses no password or other low-entropy secret, so there is nothing to guess offline. |
| C08 | Specify the target to be maintained under the inactivation | Not applicable | The documents claim no protection when a primitive fails; the API relies on TLS, OAuth2 and its primitives as given. |
| Z02 | Delegation does not extend | Not applicable | There is no delegation chain: the SCP forwards messages but is not delegated authority, and the task excludes indirect communication from the model. |
| Z04 | Avoidance and expiry | Z01 | |
| S01 | Encryption uniqueness | Not applicable | The API uses no nonce-sensitive AEAD of its own; TLS nonces are the transport baseline. |
| S02 | Re-transmit/key weight installation security | Not applicable | No key is installed or reinstalled by this API. |
| S03 | Anti-roll and state monotony | Not applicable | The API has no versioned security state that could be rolled back. |
| S04 | Consistency in security context across phases | Not applicable | There are no registration, migration or recovery stages in this API sub-flow; the subscription lifecycle is covered by A03 and A08. |
| S05 | Termination and old state lapses | The prover cannot be asked | That an unsubscribed subscription receives no further notification cannot be asked of ProVerif here: the model keeps subscriptions in a monotonic table, which cannot represent removal. |
| P01 | Identity anonymity | Not applicable | NF instance identities are public on the SBI and are carried in TLS certificates; the documents claim no anonymity for them. |
| P02 | Interconnectedness of sessions | Not applicable | The API uses stable NF identities throughout; the documents claim no unlinkability. |
| P03 | User/evidence Existence Hide | Not applicable | The API has no user or credential existence query; a subscription resource is addressed by a URI the requester must already know, which is covered as the identifier knowledge of target Z03. |

| Entry | Property of the catalogue | Examined as | Why not |
|---|---|---|---|
| V01 | Conditional progress/ error recovery | The prover cannot be asked | Progress and error recovery are liveness questions the prover cannot be asked. |
| V02 | No unexpected death locks. | The prover cannot be asked | Absence of deadlock is a liveness question the prover cannot be asked. |
| V03 | Abuse of restricted resources/anti-magnification | The prover cannot be asked | Resource abuse and amplification are cost questions, which only a count of events approximates; the documents claim nothing here. |

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

The host listed the 22 parties and protected values that the drafted properties name, and the analyst said of each which attacker controls the party or holds the value, or why no such attacker was set. With an attacker: 7. Without: 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 Consumers of services | party | `malicious_nf` | |
| S02 | NWDAF | party | Everything lost by definition | The NWDAF is the party whose decisions every target takes as authoritative; with it dishonest, no target of this API asserts anything about the consumer's assurance. |
| S03 | NRF | party | Everything lost by definition | The NRF is the authorization server; if it issues tokens at will, every authorization property of this API is lost and the documents ask none of them under a dishonest NRF. |
| S04 | Malicious NF service consumers M | party | `malicious_nf` | |
| S05 | Honesty NF service consumer C | party | Nothing gained | An attacker holding C's own TLS key and token is the malicious_nf attacker already set for every property about C; nothing further is gained for the properties about other consumers. |
| S06 | NF Example Identity | value | Nothing gained | NF instance identities are public and appear in the TLS certificates. |
| S07 | TLS Endpoint Identity | value | Nothing gained | The TLS endpoint identity is the public certificate identity; holding it without the private key gives the attacker nothing, and holding the private key is the malicious_nf attacker. |
| S08 | subscriptionId | value | `subscription_id_known` | |
| S09 | notifCorreId | value | Nothing gained | The notification correlation identifier is chosen by the consumer and echoed back; the model shows the consumer checks nothing about it, so knowing it adds nothing to the malicious_nf attacker. |
| S10 | notifUri | value | `malicious_nf` | |
| S11 | modelUniqueId / mLFileAddr | value | Outside the scope | The model file address points to a download that the specification leaves undefined and the task excludes. |
| S12 | modelUniqueId | value | Outside the scope | The model identifier belongs to the excluded download protocol. |
| S13 | mLFileAddr | value | Outside the scope | The model file address belongs to the excluded download protocol. |
| S14 | OAuth2 access token | value | Nothing gained | The producer checks the token subject against the TLS client identity, so a token without the matching client credential is not usable; the attacker that holds its own valid token is the malicious_nf attacker. |
| S15 | scope | value | Nothing gained | The scope is a public constant of the API. |
| S16 | audience | value | Nothing gained | The audience is the public NF type of the producer. |

| ID | Party or value | Kind | Attacker | Why none |
|---|---|---|---|---|
| S17 | subject | value | Nothing gained | The subject is the public NF instance identity of the consumer. |
| S18 | expiration | value | Nothing gained | The expiration time is a validity limit; the documents state that revocation is not supported, and its effect is part of the claims of Z01. |
| S19 | access token | value | Nothing gained | Same as S14: the access token alone, without the consumer's TLS client credential, authorizes nothing. |
| S20 | Subscriptions to C | value | `malicious_nf`, `subscription_id_known` | |
| S21 | C. Content of the notice | value | `malicious_nf` | |
| S22 | Contents of notice | value | `malicious_nf` | |

## The properties under each attacker

The attackers are the analyst's, set before a query was written; each answer was recorded under one of them. A cell is the prover's answer for the property under that attacker; "No answer" is an attacker set for the property under which the prover left a claim undecided or nothing was recorded; an empty cell is an attacker the analyst did not set for the property.

| Attacker | Has | Basis | Where |
|---|---|---|---|
| `network` | the network only: it reads, drops, changes and sends messages, and holds no secret of a party | | |
| `malicious_nf` | It is a registered NF service consumer M: it holds its own TLS client certificate and key, it obtains its own access token from the NRF for the nnwdaf-mlmodelprovision service scope, it registers its own callback URI, and it can open a mutually authenticated SBI connection to any other NF's endpoint from the start of the run. | The task scopes this attacker in explicitly; TS 29.501 4.10 says a service-level-scope API grants all its operations to all consumers allowed to access the service (s0395), TS 29.552 5.4.9 defines one service scope for the whole API (s0342), TS 33.501 13.4.1.0 excludes the notification operation from OAuth2 (s0410, ts-33-501:7892), and TS 33.501 5.9.2.1 requires only mutual authentication, so any registered NF is an authenticated peer of any other (ts-33-501:2871). | s0342, s0395, s0410 |
| `subscription_id_known` | The malicious NF additionally knows the subscriptionId of another consumer's subscription, obtained from an intermediary that handles the resource URI (SCP in indirect communication, per the trust assumption of TS 33.501 5.9.2.4) or from the fact that nothing in the specification requires the identifier to be unguessable. | TS 33.501 5.9.2.4 NOTE 1 states that all parties communicating through the SCP need to trust the SCP to correctly handle the messages passing through it (ts-33-501:2921), and the specification describes subscriptionId only as a string identifying a subscription (s0192) without a confidentiality or unguessability requirement; RFC 6749 10.10 requires guessing protection only for tokens and client credentials (rfc6749:3032). | ts-33-501:2921, rfc6749:3032, s0192 |

| ID | `network` | `malicious_nf` | `subscription_id_known` |
|---|---|---|---|
| A02 | Holds | Holds | |
| A03 | Holds | Fails | |
| A08 | Holds | Fails | |
| Z01 | Holds | Holds | |
| Z03 | Holds | Holds | Fails |

| ID | network | malicious_nf | subscription_id_known |
|---|---|---|---|
| Z05 | Holds | Holds | |
| A03.7 | Holds | Fails | |
| Z03.8 | Holds | Holds | Fails |
| A08.9 | Holds | Fails | |

## Notes on the prover and the model

What the analyst named for targets:

| step | targets | features | reason |
|---|---|---|---|
| 42 | A02, Z01, Z05 | plain_primitives, many_sessions | These targets rest only on the NRF-issued token (MAC over subject, audience and scope), on the transport channel modelled as authenticated encryption between endpoints, and on hold for any number of consumer, NWDAF and NRF role instances run in parallel. No arithmetic, no Diffie-Hellman algebra and no value that changes across steps. |
| 42 | A03, A08, A03.7, A08.9 | plain_primitives, many_sessions, other | These targets rest on the transport as authenticated encryption with the sender identity taken from the certificate, and on any number of sessions. Two modelling choices matter for them: the consumer's callback URI is modelled as a public function of its registered SBI endpoint (the documents do not say whether it is unguessable), and the consumer's notification acceptance performs only the checks TS 29.552 states, i.e. none on the sender or on the correlation. |
| 42 | Z03, Z03.8 | other | The subscription store is a monotonic table: a subscription is created and looked up, and removal is not modelled, so nothing in these targets depends on a value that changes (no counter, sequence number or replaced key). What they do depend on is the recorded owner of a subscription and on whether the malicious NF knows another consumer's subscriptionId, which is given by an extra attacker process. |

## The analyst's closing note

Nine targets, all answered for every attacker set for them, on model.pv (ProVerif, any number of sessions). All 27 recorded answers still hold on the final model; every attack has a figure and every proof a control that fails, except the secrecy claim in Z05.

Established by proof: the SBA transport and the OAuth2 binding hold. No party can forge another TLS endpoint's message (A02, and the endpoint part of A03/A08), and every subscribe and unsubscribe the NWDAF executes is backed by an NRF token whose subject is the instance it sees over the connection, with the producer NF type as audience and the single service scope (Z01, A02). The NWDAF notifies only a subscription it created and only at that subscription's owner's callback URI, so no NF can make it deliver a notification elsewhere, and the model address never leaves the consumer's TLS channel (Z05). The "identifier withheld" reading of Z03 is proved as well: a cross-owner deletion happens only after the attacker obtained the identifier.

Two findings, both checked as protocol changes. First, a registered NF with its own certificate and token posts a notification to another consumer's callback URI and the consumer stores it with that NF's notifCorreId and model file address: the notification operation is outside OAuth2 and the consumer is only required to store what it accepts. Binding the notification at the consumer to the producer instance and to its subscriptionId removes the attack without losing the normal run or any other answer. Second, a consumer with a valid service-scope token that knows another consumer's subscriptionId deletes that subscription, because the API defines one service scope and no resource or operation scope and the unsubscribe never compares the requester with the recorded owner; adding the owner comparison removes it. Both are marked novelty unclear: TS 29.501 4.10 warns of the delete-access risk in general terms, and neither document states either failure.

What blocks more: no bounded.pv, because no target rests on a value that changes across steps — the subscription store is monotonic and removal is not modelled, so nothing about behaviour after an unsubscribe is asked. The other simplifications are in limits: the TLS channel has no handshake or sequence numbers (replay is therefore possible in the model and no target rests on it), the callback URI is assumed derivable from the registered SBI endpoint (the notification injection rests on that), token expiry and revocation are not modelled, and indirect communication through an SCP, the PUT operation, the error paths and the model download are outside the model. The two findings involve no algebraic reasoning, so Tamarin+ Agent would confirm rather than change them; it would be the tool to settle post-unsubscribe behaviour and the SCP variants.