Security analysis of ADRF Model Repository: Storage Confirmation and Retrieval

This report in another language: 中文.

19.7.0 · ProVerif+ Agent report

Documents	5G System; Analytics Data Repository Services; Stage 3 (protocol definition)
Properties examined	10: 6 hold, 4 fail (4 attacks)
Findings	1

Contents

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

1 Protocol and scope

The Analytics Data Repository Function offers the ML model management service: an NF service consumer, normally the NWDAF containing the model training logic, stores ML models in the repository, and consumers such as the LMF retrieve them later. A storage request carries the model identifier, the address of the model file and optionally the list of consumers allowed to retrieve it; the repository answers with the created record and a storage transaction identifier. A retrieval names a storage transaction identifier or model identifiers, and the repository answers with the record, that is with the model identifier and the address of the model file.

When a storage request ends, the consumer is sure that the record it is shown and the storage transaction identifier it receives are its own. When a retrieval ends, the consumer is sure that the entry is for the model identifier it asked for and that the repository checked that it is either the consumer that stored the model or one the storing NWDAF listed. The repository is sure, for each request, of the identity of the consumer that sent it, because access is authorised by an OAuth2 token.

The parties:

- **NWDAF containing MTLF**: NF service consumer that stores the model. Holds: nfInstanceId, OAuth2 access token.
- **ADRF**: ML model store. Holds: SBI credentials.
- **LMF**: NF service consumer that retrieves the model. Holds: nfInstanceId, OAuth2 access token.

Figure 1 shows the exchange as the documents specify it.

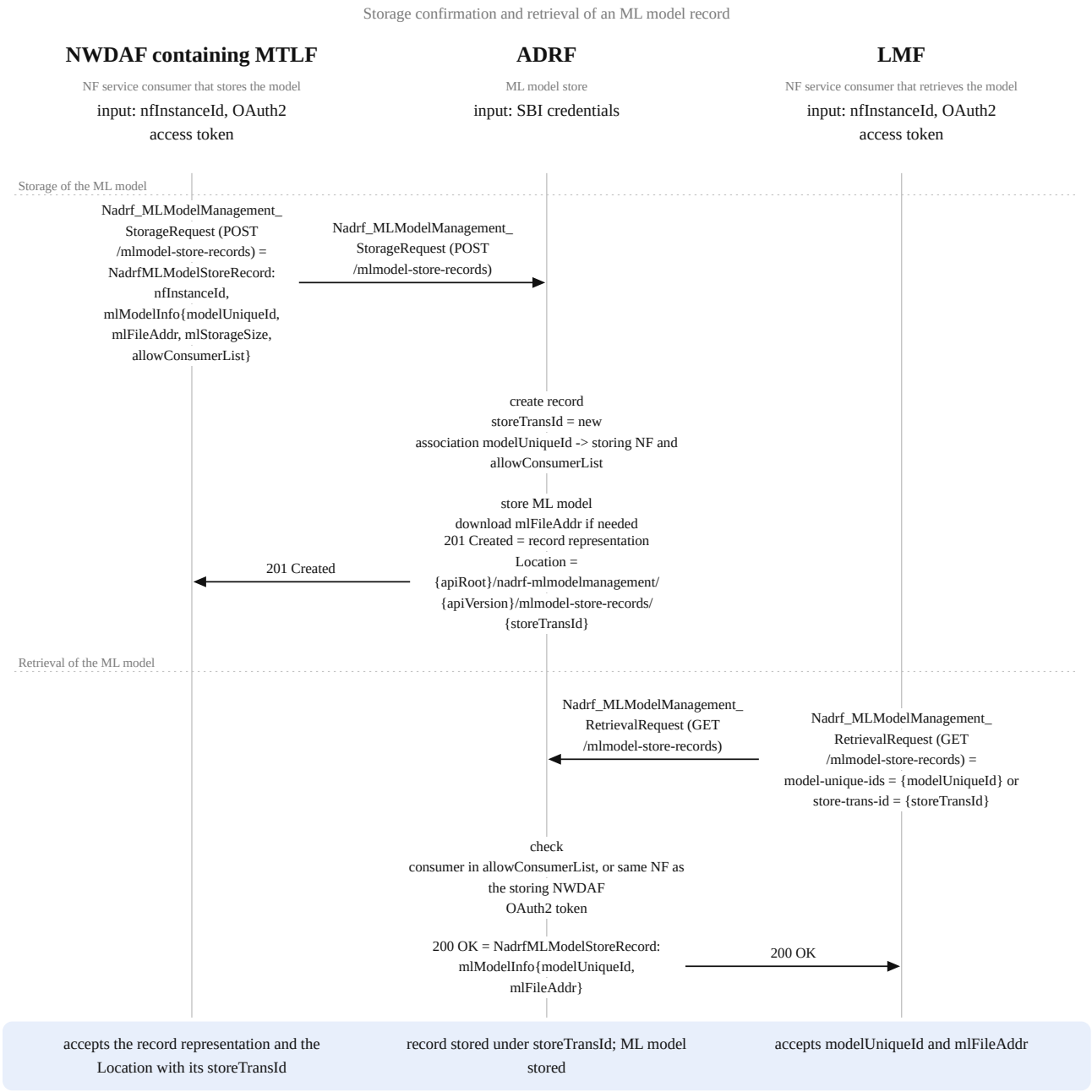


Figure 1. The protocol as specified.

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

2 Findings

The analysis has 1 finding: what follows is not stated in the documents.

Within the scope analysed the storage confirmation and the retrieval checks hold: the 201 answers the consumer's own request, and the repository returns a record only to the storing NWDAF or to a consumer the storing NWDAF listed with a valid token. One weakness is new: any registered NF consumer can register a second record under an existing model identifier, and the repository may then answer an authorised consumer with that record's file address, so the consumer downloads a file of the attacker's choosing. The documents do not state this and do not forbid it. The

results rest on the assumption that the 3GPP service based interface is authenticated and integrity protected, which the documents delegate to other specifications, and on the reading of the 403 condition as 'listed or storing NF'. The proofs and the attack otherwise cover any number of sessions.

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 consumer can take over another NF's model identifier and have its file address returned

Kind	Weakness of the protocol as specified
New	Yes: the documents do not state it
Value	high
Properties	X01, A08, A08.8, Z03 (section 3)
Source text	s0048, s0080, s0081, s0255
Shown by	ProVerif attack query `a08_addr`; Figures 2, 3, 4 and 5

What happens. An NF service consumer with a token for the service scope stores a record that carries the modelUniqueId the NWDAF already registered, together with its own mlFileAddr and an allowConsumerList that names the consumer it wants to reach. The ADRF keeps a record per storage request, so both records now carry the same identifier. When the LMF retrieves by that modelUniqueId, the ADRF accepts its request because the allowed list registered for the identifier matches, and answers with the rogue record, so the LMF receives the rogue file address instead of the NWDAF's.

What the attacker needs. A registered NF instance and an OAuth2 token for the single service scope nadrf-mlmodelmanagement, which s0295 gives for the whole service. No interception and no stolen credential are needed; the rogue consumer chooses the model identifier, the file address and the allowed list of its own request.

What the attacker gains. A consumer that retrieves a model by identifier can be handed an attacker-chosen file address and download a malicious model in place of the one the NWDAF trained, while the identifier it sees is the expected one.

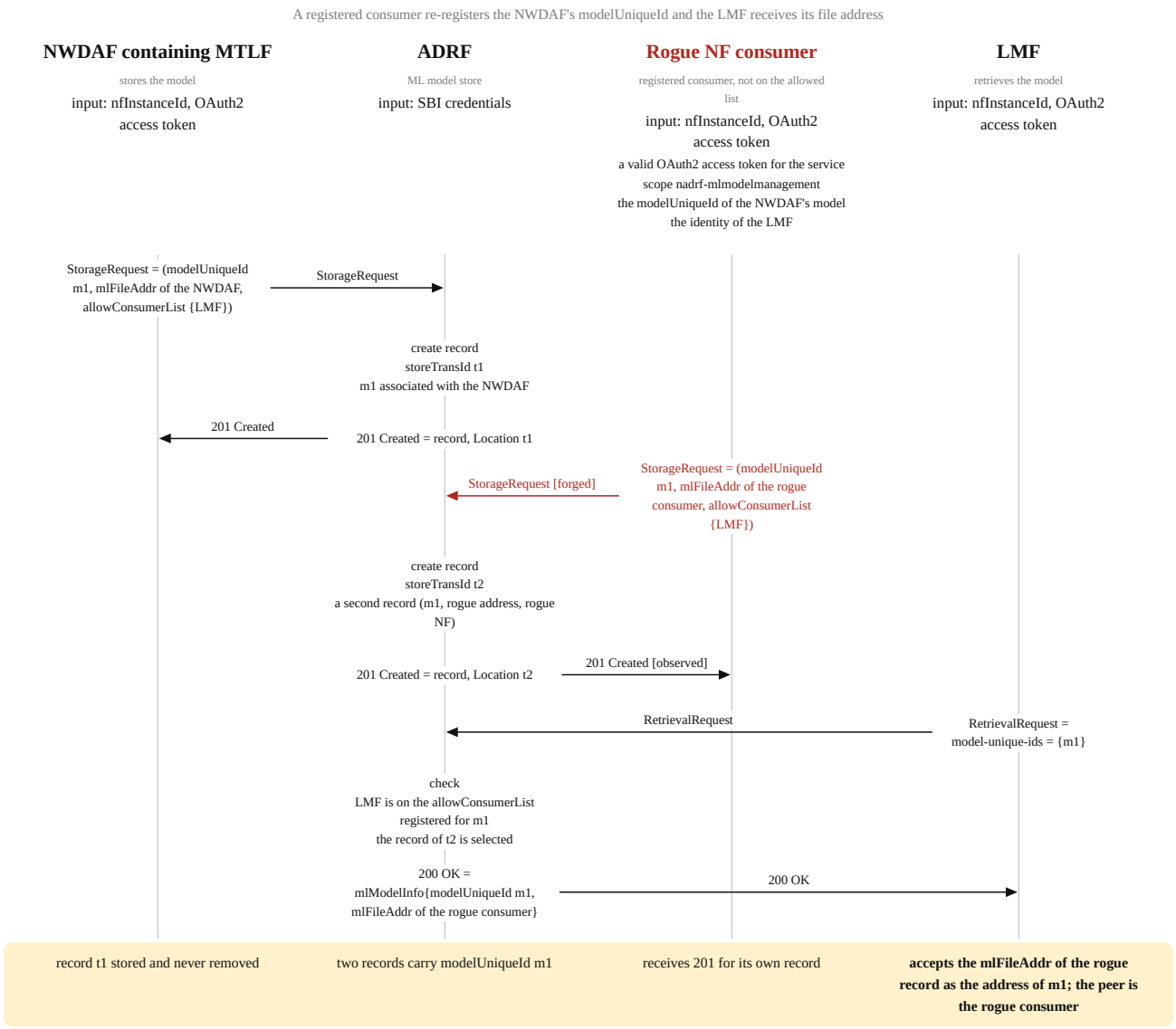


Figure 2. A registered consumer re-registers the NWDAF's modelUniqueId and the LMF receives its file address. Property X01; attack query a08_addr.

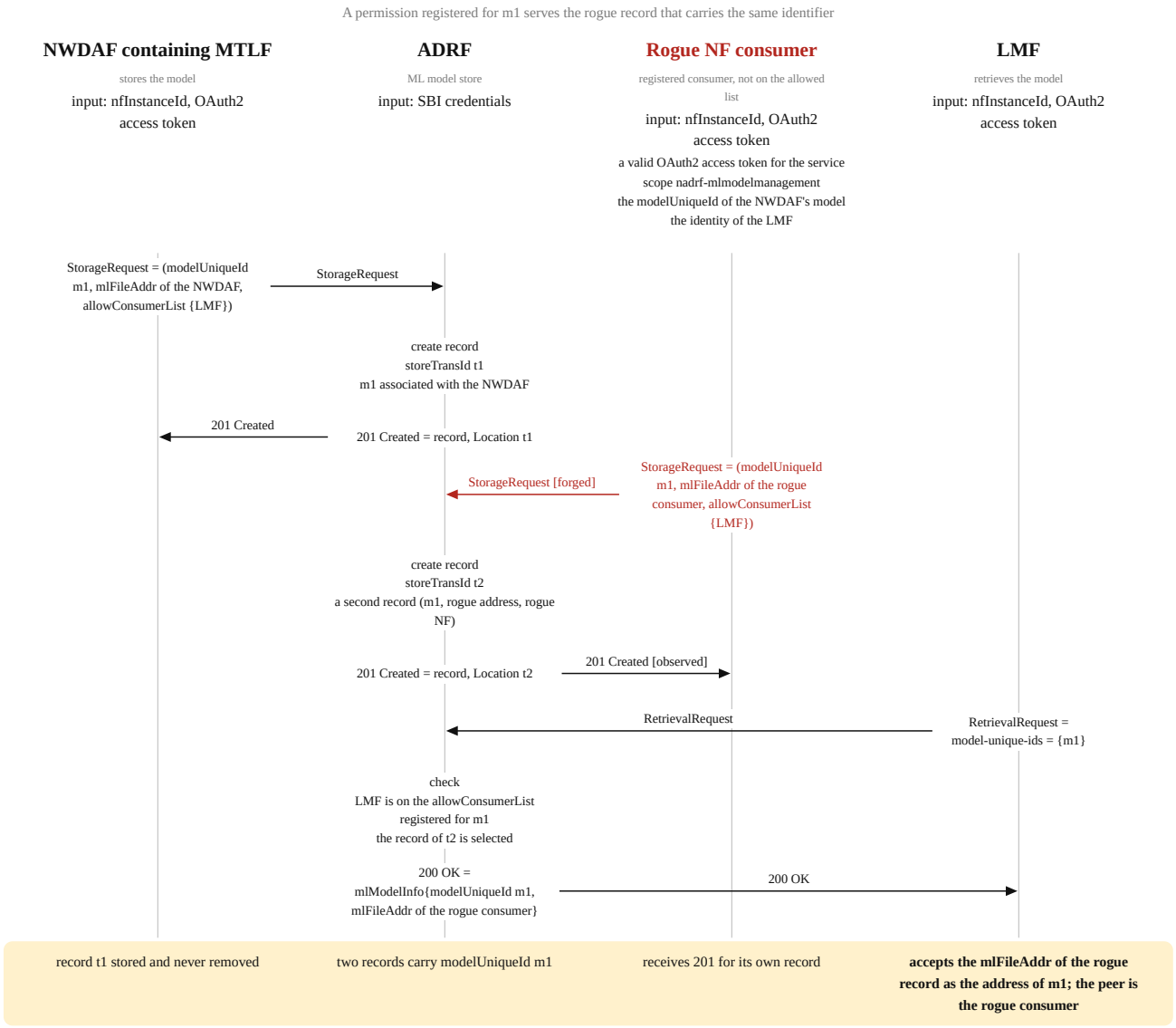


Figure 3. A permission registered for m1 serves the rogue record that carries the same identifier. Property Z03; attack query a08_addr.

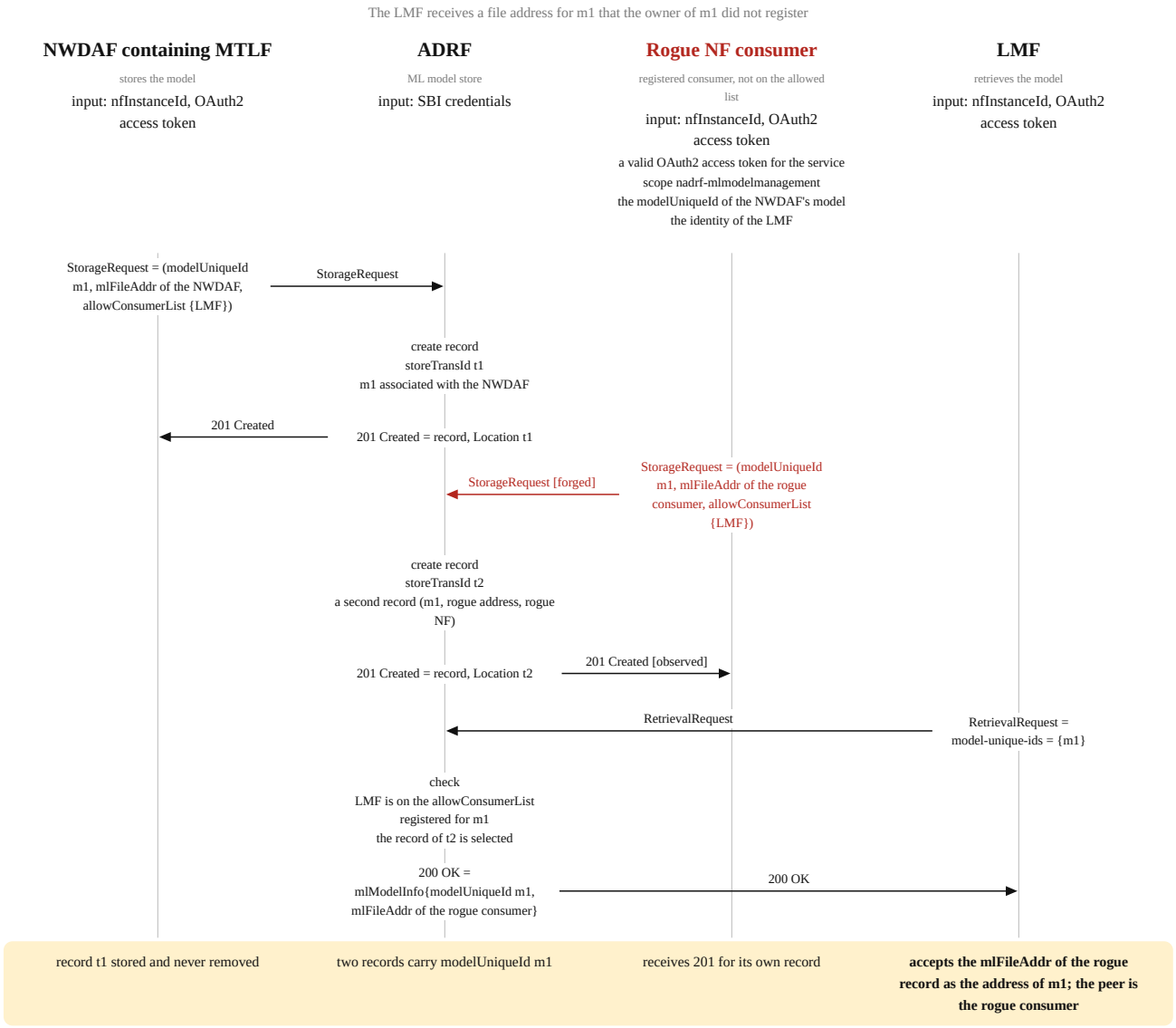


Figure 4. The LMF receives a file address for m1 that the owner of m1 did not register. Property A08; attack query a08_addr.

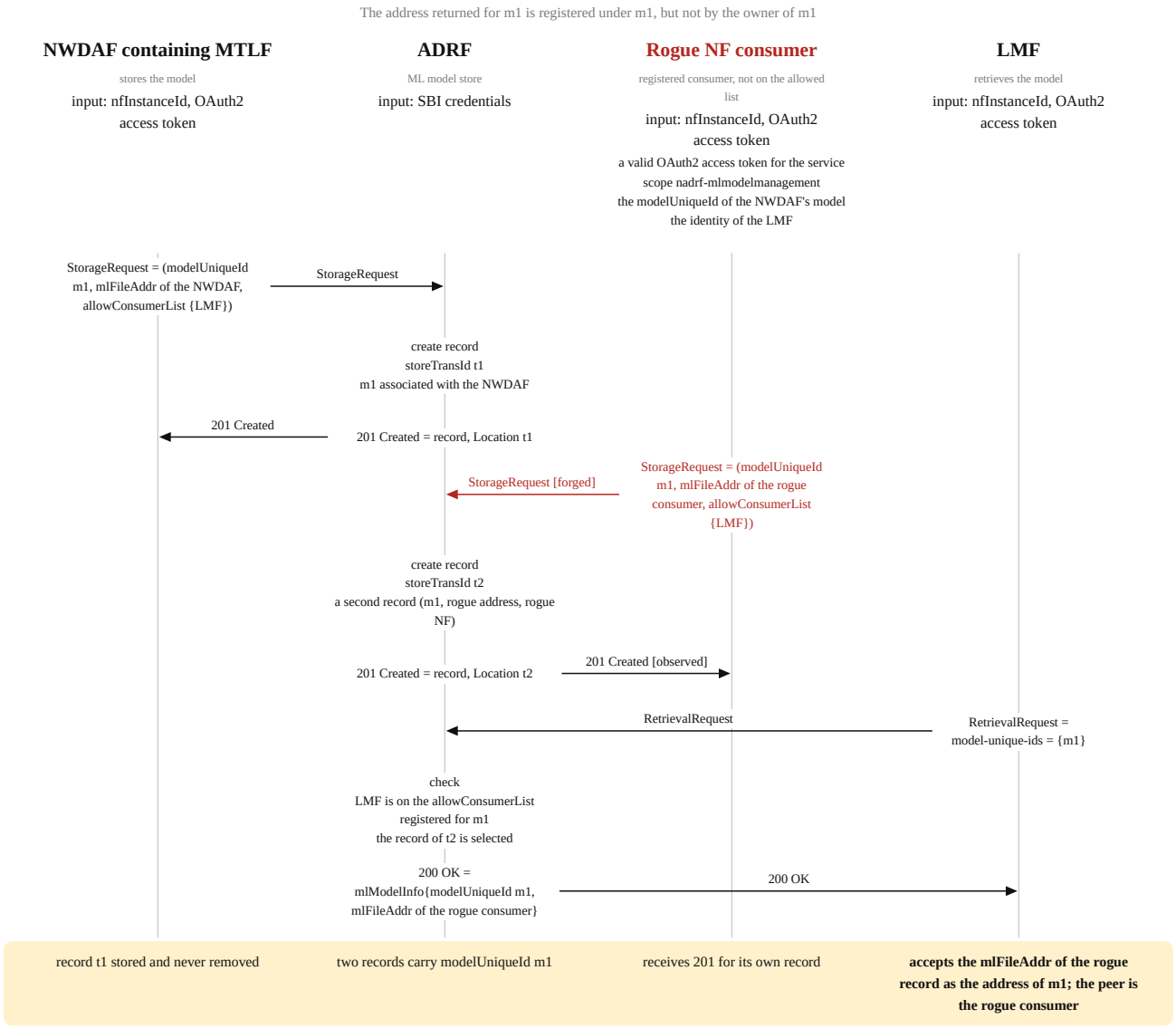


Figure 5. The address returned for m1 is registered under m1, but not by the owner of m1. Property A08.8; attack query a08_addr.

Proposed change. The ADRF shall keep one association per modelUniqueId, as TS 23.288 line 5102 already requires, and reject a storage request whose modelUniqueId is already associated with a different NF instance with 409 Conflict instead of creating a second record for it; a retrieval by modelUniqueId then returns the record of that one association. The check of the existing association and the creation of the new one must be one atomic step.

Check of the change with ProVerif. Not checked: The failure depends on which of the two storage requests the ADRF records first: a check that rejects a duplicate identifier only moves the attack to the run in which the rogue request is recorded first, since the protocol does not order them. Removing it needs the identifier to be bound to a single NF at allocation, and the consumer supplies modelUniqueId itself (s0255), so that change lies outside the message flow this model represents.

3 Properties examined

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

ID	Property	Compromise assumed	Result	Value	In the documents
Z01	**Retrieval only for authorised consumers.** When the ADRF returns 200 OK with a model record (including modelUniqueId and mlFileAddr), the requester must be on that model's allowed consumer list, or be the same NF as the NWDAF (including MTLF) that stored the model, and must hold an authorisation covering the operation.	no compromise	Holds	medium	yes
Z03	**Authorisation bound to model and requester.** The authorisation used for a retrieval must be bound to the requested model identifier (the record pointed to by modelUniqueId or storeTransId), to the requester's identity and to the retrieval action; a permission for one model or one consumer must not be used for another model or another consumer.	no compromise	Fails (1 attack)	high	unclear
A03	**201 answers the consumer's own storage request.** When a consumer receives 201 Created, the record it returns and the storeTransId in the Location correspond to the storage request it made itself; when a consumer receives 200 OK, the modelUniqueId and mlFileAddr of the returned record correspond to the store-trans-id or model-unique-ids it requested.	no compromise	Holds	medium	yes
A08	**Retrieved record answers the request.** Every model entry in a retrieval response corresponds to one of the store-trans-ids or model-unique-ids given in the request, and mlFileAddr is the address registered for that modelUniqueId at storage time; the Location of a creation response points to the record created by this request.	no compromise	Fails (1 attack)	high	no
D01	**Messages cannot be substituted in transit.** The content of a storage request the ADRF accepts is the content the consumer actually sent; the content of a retrieval response the consumer accepts is the content the ADRF actually returned, and has not been replaced by a man in the middle.	no compromise	Holds	low	unclear
K01	**Access token stays secret.** An attacker cannot obtain the consumer's OAuth2 access token or the client credentials used to obtain a token.	no compromise	Holds	low	unclear
A03.7	**Store confirmation binds requester and storeTransId.** When a consumer receives 201 Created, the returned record and the storeTransId in the Location correspond to the storage request it made itself, and the nfInstanceId/nfSetId in the record is that of the requester.	no compromise	Holds	low	yes

ID	Property	Compromise assumed	Result	Value	In the documents
A08.8	**Retrieval entries match requested identifiers.** Every model entry in a retrieval response corresponds to one of the store-trans-ids or model-unique-ids given in the request, and mlFileAddr is the address registered for that modelUniqueId at storage time.	no compromise	Fails (1 attack)	high	no
Z01.9	**Unlisted consumer gets no stored record.** Under a fixed allowed consumer list and storer identity, a consumer that is not on the list and is not the storing NWDAF cannot obtain a 200 OK model record.	no compromise	Holds	medium	yes
X01	**Model identity bound to its storer.** When a consumer retrieves by modelUniqueId, the mlFileAddr it receives is the address registered by the NF that owns that modelUniqueId; another NF service consumer cannot register a second record carrying the same modelUniqueId whose mlFileAddr is then returned to consumers.	A valid OAuth2 access token for the single service scope nadrf-mlmodelmanagement, i.e. a registered NF service consumer, which is not in the allowConsumerList of the model it wants; it can send its own storage and retrieval requests and can choose the modelUniqueId, mlFileAddr and allowConsumerList of its own storage request.	Fails (1 attack)	high	no

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

The questions asked, and the analyst's answers:

Question	Answer	Rests on
Q1: Store confirmation binds the requester and storeTransId	Yes: the 201 Created a consumer accepts, and the storeTransId in its Location, always answer that consumer's own storage request, and the record carries the storing NF instance as its nfInstanceId.	A03, A03.7
Q2: Retrieval results bound to the requested modelUniqueId/transaction and to the returned address	Only partly: the entry returned is always for the requested modelUniqueId, but its mlFileAddr is not necessarily the address the owner of that model registered, because a second record can carry the same identifier and the ADRF may answer with it.	A08, A08.8, X01
Q3: Against a fixed authorisation snapshot, can an unauthorised consumer obtain a record	Against the record it is not associated with, an unlisted consumer is refused; but a consumer that is not on the list and did not store the model can still obtain a record carrying that modelUniqueId by registering one itself, so the protection does not extend to the identifier.	Z01.9, Z01, X01

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 3GPP service based interface is represented as one private authenticated channel per registered NF. Message integrity, sender authentication and the OAuth2 token are assumed, not derived; the documents delegate them to TS 29.500 and TS 33.501.
- Only retrieval by modelUniqueId is modelled. Retrieval by store-trans-id, and a record holding several ML models each with its own allowed consumer list, are not represented.

- The 403 wording of s0083 ('and/or') is modelled as: allowed when the consumer is on the allowed list OR is the storing NWDAF. The AND reading, which would deny a listed consumer, is not analysed.
- One rogue consumer and one retrieving consumer are modelled; the record store is a monotone table, so deletion, update and revocation of permissions are outside the model, as the scope states.
- Partial failure of a storage or retrieval request and the EnModelMgmt feature, including modelStoreResult, are not modelled.
- Token issuance by the NRF, its scope, lifetime and revocation are outside the model; a token is a credential that authenticates an NF instance.
- The ADRF is honest in the model: a dishonest repository is not analysed, and no property of its behaviour can be stated against it.

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

Appendix A: Reading of each property

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

Z01: Retrieval only for authorised consumers

When the ADRF returns 200 OK with a model record (including modelUniqueId and mlFileAddr), the requester must be on that model's allowed consumer list, or be the same NF as the NWDAF (including MTLF) that stored the model, and must hold an authorisation covering the operation.

The ADRF hands the stored record, with the model file address, only to the NWDAF that stored the model or to a consumer the storing NWDAF put on the model's allowed consumer list, and only to a requester that presented a valid access token. Both conditions carry the property: dropping the allowed-list test or the token test makes the ADRF serve the wrong party. The denial for a consumer with no token at all is unconditional.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: s0083 and s0286 require the 403 RETRIEVAL_ML_MODEL_NOT_ALLOWED for a consumer that is not allowed and not the storing NF, and s0292-s0295 require an OAuth2 token for the API; the analysis confirms these rules hold when they are implemented.
- Stated in the documents: yes. Where the documents say it: main:1358, s0083, s0286, s0292, s0295
- Rests on: the 3GPP service based interface is represented as one private, authenticated channel per registered NF, so the sender identity and message integrity are assumed rather than derived (TS 29.500, TS 33.501, outside the selected text); s0083's 403 condition is modelled with the reading that a consumer is allowed when it is on the allowed consumer list OR is the storing NWDAF; the AND reading would deny a listed consumer and is not modelled; the allowed consumer list is held per modelUniqueId; a record with several ML models, each with its own list, is not modelled; token issuance, scope and expiry by the NRF are not modelled; a token is a credential that authenticates an NF instance
- Open: the policy reading of the and/or wording of s0083; whether a token issued for the single service scope can be restricted to one operation by the NRF

Z03: Authorisation bound to model and requester

The authorisation used for a retrieval must be bound to the requested model identifier (the record pointed to by modelUniqueId or storeTransId), to the requester's identity and to the retrieval action; a permission for one model or one consumer must not be used for another model or another consumer.

The identity the ADRF decides a retrieval on is the identity that sent the request, so a consumer cannot ask for a model under another NF's identity and a stolen token makes the thief indistinguishable from the NF it stole from. The second half of the claim does not hold: the permission registered for a modelUniqueId is applied to whichever record carries that identifier, so a permission given for the NWDAF's model is used to serve a record another consumer registered under the same identifier.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value high: the identity binding is what s0083 and s0292-s0295 require and is confirmed here; the reuse of a permission across two records that carry the same modelUniqueId is not stated anywhere in the documents, and TS 23.288 line 5102 only says that the ADRF maintains the association between the model identifier, the storing NF and the allowed list.

- Stated in the documents: unclear. Source text it rests on: s0083, s0255, s0295
- Rests on: sender identity comes from the per-NF channel (TS 29.500/33.501 protection assumed); one record holds one ML model in the model; the per-model allowed list of a multi-model record is not represented; the acquisition and the content of the OAuth2 token are not modelled
- Open: whether the ADRF is required to reject a second storage request for a modelUniqueId already associated with another NF instance

A03: 201 answers the consumer's own storage request

When a consumer receives 201 Created, the record representation it returns corresponds to the storage request it originated itself, identified by the storeTransId in Location. When a consumer receives 200 OK, the modelUniqueId and mlFileAddr of the returned record correspond to the store-trans-id or model-unique-ids it requested.

The 201 Created a consumer accepts always answers a storage request that this consumer sent, and the storeTransId it carries was assigned to that request. A response of another transaction is not accepted, because the record representation is checked against the requested model.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: s0049 and s0165 require the 201 to carry the created record and a Location with the storeTransId, and the analysis confirms the client-side binding holds.
- Stated in the documents: yes. Where the documents say it: s0042, s0049, s0165
- Rests on: responses arrive on the consumer's own authenticated channel (SBI protection assumed); one ML model per record; the record representation is reduced to its modelUniqueId and file address; no concurrency check: a stale 201 for the same modelUniqueId would be accepted for a later request to the same model
- Open: behaviour when the same modelUniqueId is stored twice by the same NF and the older 201 is replayed

A08: Retrieved record answers the request

Every model entry in a retrieval response corresponds to one of the store-trans-id or model-unique-ids given in the request, and mlFileAddr is the address registered for that modelUniqueId at storage time; the Location of a creation response points to the record created by this request.

A consumer receives an entry for the modelUniqueId it asked for, so the response is not mixed up between identifiers. The address it receives is not necessarily the one the owner of that model registered: if another NF consumer has registered a second record with the same modelUniqueId, the ADRF may answer with that record and the consumer downloads the attacker's file.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value high: searched the security clause (s0291-s0295), the storage and retrieval clauses (s0042-s0084) and TS 23.288 clauses 6.2B.5 and 6.2B.7 for a rule on a repeated modelUniqueId: the documents state that a new record and a new storeTransId are created for a model that is already stored (s0048) and that the ADRF keeps the association between the identifier and the storing NF (TS 23.288 line 5102), but they never say that a second consumer may not register the same identifier, nor which record a retrieval then returns.
- Stated in the documents: no. Source text it rests on: s0048, s0080, s0081, s0255
- Rests on: one ML model per record and retrieval by modelUniqueId only; retrieval by storeTransId is not modelled; the ADRF is modelled as returning one matching record; the text does not say which record is returned when several carry the same identifier; the allowed consumer list used for the check is the one registered for the modelUniqueId
- Open: which record the ADRF returns when several carry the requested modelUniqueId

D01: Messages cannot be substituted in transit

The content of a storage request the ADRF accepts matches what the consumer actually sent; the content of a retrieval response the consumer accepts matches what the ADRF actually returned, with no substitution by a man in the middle.

No party on the path can substitute a model identifier or a file address: the ADRF only ever handles the identifiers and addresses the consumers sent, and the consumers only accept responses produced by the ADRF. This result rests entirely on the assumption that the service based interface is authenticated and integrity protected, which the documents delegate to TS 29.500 and TS 33.501.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: the documents state the message contents (s0042, s0049, s0081) and delegate the protection of the interface to other specifications (s0292); they do not state the integrity property themselves.
- Stated in the documents: unclear. Source text it rests on: s0042, s0049, s0081, s0292
- Rests on: the SBI is modelled as one private authenticated channel per NF; without that assumption every one of these properties fails trivially and any failure would be an artefact of the model, not of the protocol; no message is modelled as replayable across connections
- Open: integrity of the SBI as specified in TS 29.500 and TS 33.501

K01: Access token stays secret

The attacker cannot obtain the consumer's OAuth2 access token or the client credentials used to obtain the token.

The access token of the NWDAF is never exposed by the modelled exchange; nobody on the network obtains it. The result says little, because the token is only used as the credential that authenticates the consumer and its issuance by the NRF is outside the model.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: s0292-s0293 say the consumer obtains a token from the NRF before consuming the service but state no secrecy goal for it; the token handling itself is in TS 29.510 and TS 33.501.
- Stated in the documents: unclear. Source text it rests on: s0292, s0293
- Rests on: the token is represented by a private credential of each NF, never sent; token format, lifetime and the client credentials exchange with the NRF are not modelled
- Open: protection of the token in the NRF exchange

A03.7: Store confirmation binds requester and storeTransId

When a consumer receives 201 Created, the returned record and the storeTransId in Location correspond to the storage request it originated itself, and the nfInstanceId/nfSetId in the record is that of the requester.

The nfInstanceId the ADRF records for a stored model is the NF instance that sent the storage request, so the record and its Location belong to the requester and not to a third party.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value low: s0252 and s0049 define the record and its return in the 201; the analysis confirms the field is filled from the requester.
- Stated in the documents: yes. Where the documents say it: s0252, s0049, s0165
- Rests on: sender identity comes from the per-NF channel; the record is reduced to the storing NF, the modelUniqueId and the file address; nfSetId and the other attributes are not represented

- Open: nfSetId as the only identifier (NOTE 1 of s0252), where the storing instance cannot be told apart within a set

A08.8: Retrieval entries match requested identifiers

Every model entry in a retrieval response corresponds to one of the store-trans-id or model-unique-ids given in the request, and mlFileAddr is the address registered for that modelUniqueId at storage time.

Every entry a consumer accepts is for a modelUniqueId it asked for. The address inside that entry is only guaranteed to be one that some consumer registered under that identifier, not the one the owner of the model registered, so the second half of the claim fails for the same reason as in the retrieval target.

- Result: Fails (1 attack); answers what is asked: partly; usable with caveats
- Value high: searched the storage, retrieval and security clauses and TS 23.288 6.2B.5/6.2B.7: the documents do not say which record is returned when several carry the requested modelUniqueId, nor that a duplicate identifier is refused.
- Stated in the documents: no. Source text it rests on: s0048, s0080, s0081
- Rests on: one model per record, retrieval by modelUniqueId, one matching record returned; the ADRF keeps all records created for the same identifier (s0048)
- Open: the rule for duplicate modelUniqueId values

Z01.9: Unlisted consumer gets no stored record

Under a fixed allowed consumer list and a fixed storer identity, a consumer that is not on the list and is different from the storing NWDAF cannot obtain the model record in a 200 OK.

For a record it is not associated with, a consumer that is neither on the allowed list nor the storing NF is refused, and the rogue consumer in the model receives only the record it stored itself. The claim holds for records; read per modelUniqueId it does not, because a rogue consumer can obtain a record carrying another NF's modelUniqueId, which is the attack reported under the model identity target.

- Result: Holds; answers what is asked: yes; usable with caveats
- Value medium: s0083 and s0286 state the denial for a consumer that is not allowed and not the storing NF; the model confirms it for a record.
- Stated in the documents: yes. Where the documents say it: s0083, s0286
- Rests on: one record per modelUniqueId in the snapshot; the duplicate case is the attack reported separately; the allowed consumer list is fixed by the storing consumer's request
- Open: the per-modelUniqueId reading, answered by the model identity target

X01: Model identity bound to its storer

When a consumer retrieves by modelUniqueId, the mlFileAddr it receives is the address registered by the NF that owns that modelUniqueId; another NF service consumer cannot register a second record carrying the same modelUniqueId whose mlFileAddr is then returned to consumers.

The claim fails. A registered NF consumer with a token for the service stores a second record that carries the NWDAF's modelUniqueId but its own file address and an allowed consumer list naming the LMF; the ADRF keeps both records and serves the rogue one, so the LMF downloads the rogue file while believing it is the NWDAF's model.

- Result: Fails (1 attack); answers what is asked: yes; usable with caveats
- Value high: no clause of the selected text and no clause of TS 23.288 6.2B.5/6.2B.7 states that a modelUniqueId is bound to the NF that first registered it or that a second registration is refused; s0048 says the opposite for records, and TS 23.288 line 5102 states only that the ADRF maintains the association between the identifier, the storing NF and the allowed list.
- Stated in the documents: no. Source text it rests on: s0048, s0080, s0081, s0255
- Rests on: a registered NF consumer can obtain a token for the single service scope and choose the modelUniqueId, file address and allowed list of its own request; one model per record; the ADRF returns one of the matching records; the SBI itself is assumed protected; the attack uses no interception
- Open: whether an operator can keep modelUniqueId values from being guessed or chosen by other NFs

Appendix C: Claims and prover answers

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

target	level	bound	lemma	control	attacks	claims
Z01	proved		z01_auth	checked		3 proved
Z03	proved		z03_bind	checked	a08_addr (holds_in_source)	1 proved, 1 attack
A03	proved		a03_bind	checked		1 proved
A08	proved		a08_req	checked	a08_addr (holds_in_source)	1 proved, 1 attack
D01	proved		d01_get	checked		2 proved
K01	proved		k01_tok	checked		1 proved
A03.7	proved		a037_nf	checked		1 proved
A08.8	proved		a08_req	checked	a08_addr (holds_in_source)	1 proved, 1 attack
Z01.9	proved		z019_auth	checked		1 proved
X01	none				a08_addr (holds_in_source)	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
Z01	The ADRF sends the 200 OK record carrying the mlFileAddr that the storing NWDAF registered only to that NWDAF or to a consumer on the allowConsumerList of the model.	`stolen_token`	proved	z01_auth	checked
Z01	A 200 OK is sent only to a requester that presented a valid access token for its identity.	`stolen_token`	proved	z01_tok	checked
Z01	A requester that holds no access token at all is never sent the record by the ADRF.	`stolen_token`	proved	z01_anon	checked
Z03	The identity the ADRF decides a retrieval on is the identity that sent the request; a consumer cannot ask under another NF's identity.	`stolen_token`	proved	z03_bind	checked
Z03	A permission held for a modelUniqueId is not used to serve another record carrying that identifier: the consumer receives the mlFileAddr registered by the owner of the model, not one registered by a different NF for the same modelUniqueId.	`rogue_consumer`	attack (holds_in_source)	a08_addr	
A03	The 201 Created a consumer accepts corresponds to the storage request it sent: the record representation and the Location storeTransId answer its own POST.	`rogue_consumer`	proved	a03_bind	checked
A08	The mlFileAddr a consumer receives for a modelUniqueId is the address registered at storage by the NF that owns that model.	`rogue_consumer`	attack (holds_in_source)	a08_addr	
A08	Every model entry a consumer accepts corresponds to the modelUniqueId it asked for.	`rogue_consumer`	proved	a08_req	checked
D01	The modelUniqueId of a retrieval request that reaches the ADRF is the one the consumer sent.	`rogue_consumer`	proved	d01_get	checked

target	claim	attacker	answer	query	control
D01	The content of a storage request that reaches the ADRF is the content the consumer sent.	`rogue_consumer`	proved	d01_store	checked
K01	The attacker does not obtain the access token or client credential of the NF service consumer.	`network`	proved	k01_tok	checked
A03.7	The nfInstanceId/nfSetId of the record the ADRF creates is the NF instance that sent the storage request.	`rogue_consumer`	proved	a037_nf	checked
A08.8	The mlFileAddr returned for a requested modelUniqueId is the address registered at storage by the NF that owns that model.	`rogue_consumer`	attack (holds_in_source)	a08_addr	
A08.8	Every model entry a consumer accepts corresponds to the modelUniqueId it asked for.	`rogue_consumer`	proved	a08_req	checked
Z01.9	A consumer that is neither in the allowConsumerList of the stored model nor the NF that stored it is not sent a 200 OK record of that model.	`rogue_consumer`	proved	z019_auth	checked
X01	A consumer that retrieves a modelUniqueId receives the mlFileAddr registered by the NF that owns that model; no other NF can have its own address returned for that identifier.	`rogue_consumer`	attack (holds_in_source)	a08_addr	

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-20261114T183856, 2026-11-14. Status: finished. Model: deepseek-v4-1-flash-261017. Steps: 50, prover 7 s, wall 15 min, cost 0.104 USD.

ProVerif: 128 proofs run, 128 decided, 0 without an answer, 5 s, 42 answers recorded.

What was analysed

Protocol	ADRF model repository: storage confirmation and retrieval
Version	19.7.0
Analysed	One fixed storage transaction and an immutable model record; the model is already local/download complete; checks the creation confirmation and retrieval by storeTransId/modelUniqueId. Update, deletion and malicious file download are left for later.
Not analysed	403 conditions exist and/or wording, which needs to be checked against Stage 2/data types and the adopted policy interpretation recorded; confidentiality of the returned download address is not the same as confidentiality of the model file; continuous update, deletion and revocation of permissions are not conclusions of the first-round symbolic model. Only properties under the explicit model and attacker conditions are proved; missing checks, a fully secure channel or all-honest NFs cannot be substituted for the conclusions to be proved.
Source text	295 numbered passages (54906 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	4.3	Nadrf_ MLModelManagement Service
main	4.3.1	Service Description
main	4.3.1.1	Overview
main	4.3.1.2	Service Architecture
main	4.3.1.3	Network Functions
main	4.3.1.3.1	Analytics Data Repository Function (ADRF)
main	4.3.1.3.2	NF Service Consumers
main	4.3.2	Service Operations
main	4.3.2.1	Introduction
main	4.3.2.2	Nadrf_MLModelManagement_StorageRequest service operation
main	4.3.2.2.1	General
main	4.3.2.2.2	Request Storage of ML model(s)
main	4.3.2.2.3	Update Storage of ML model(s)
main	4.3.2.3	Nadrf_MLModelManagement_RetrievalRequest service operation
main	4.3.2.3.1	General

Document	Section	Title
main	4.3.2.3.2	Request and get stored ML model(s) from ADRF ML Model Store
main	4.3.2.4	Nadrf_MLModelManagement_Delete service operation
main	4.3.2.4.1	General
main	4.3.2.4.2	Requesting removal of stored ML model(s)
main	4.3.2.4.3	Requesting removal of stored ML model(s) using unique ML model identifier
main	5.2	Nadrf_MLModelManagement Service API
main	5.2.1	Introduction
main	5.2.2	Usage of HTTP
main	5.2.2.1	General
main	5.2.2.2	HTTP standard headers
main	5.2.2.2.1	General
main	5.2.2.2.2	Content type
main	5.2.2.3	HTTP custom headers
main	5.2.3	Resources
main	5.2.3.1	Overview
main	5.2.3.2	Resource: ADRF ML Model Store Records
main	5.2.3.2.1	Description
main	5.2.3.2.2	Resource Definition
main	5.2.3.2.3	Resource Standard Methods
main	5.2.3.2.3.1	POST
main	5.2.3.2.3.2	GET
main	5.2.3.2.4	Resource Custom Operations
main	5.2.3.3	Resource: Individual ADRF ML Model Store Record
main	5.2.3.3.1	Description
main	5.2.3.3.2	Resource Definition
main	5.2.3.3.3	Resource Standard Methods
main	5.2.3.3.3.1	DELETE
main	5.2.3.3.3.2	PUT
main	5.2.3.3.4	Resource Custom Operations
main	5.2.4	Custom Operations without associated resources
main	5.2.4.1	Overview
main	5.2.4.4	Operation: remove-stored-mlmodel
main	5.2.4.4.1	Description
main	5.2.4.4.2	Operation Definition
main	5.2.5	Notifications
main	5.2.6	Data Model

Document	Section	Title
main	5.2.6.1	General
main	5.2.6.2	Structured data types
main	5.2.6.2.1	Introduction
main	5.2.6.2.2	Type: NadrfMLModelStoreRecord
main	5.2.6.2.3	Type: MLModelInfo
main	5.2.6.2.4	Type: MLModel
main	5.2.6.2.5	Type: MLModelDelResult
main	5.2.6.2.6	Type: AllowedConsumer
main	5.2.6.2.7	Type: ModelStoreResult
main	5.2.6.3	Simple data types and enumerations
main	5.2.6.3.2	Enumeration: StoreResult
main	5.2.6.4	Data types describing alternative data types or combinations of data types
main	5.2.7	Error Handling
main	5.2.7.1	General
main	5.2.7.2	Protocol Errors
main	5.2.7.3	Application Errors
main	5.2.8	Feature negotiation
main	5.2.9	Security

The purpose of the protocol as the source step located it in the documents: fix one storage transaction and an immutable model record, the model already being local/download complete; check the creation confirmation and the retrieval by storeTransId/modelUniqueId. Update, deletion and malicious file download are left to later.

The properties against the documents

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

ID	Name	About	What the documents say	Where
Z01	Retrieval only for authorised consumers	The ADRF returns the NadrfMLModelStoreRecord (modelUniqueId, mlFileAddr) only to an NF service consumer that is in the allowConsumerList of the ML model or is the NF of the NWDAF containing MTLF that stored it, and that holds an access token for the operation.	s0083 requires 403 RETRIEVAL_ML_MODEL_NOT_ALLOWED when the consumer is not in the allowed NF consumer list and/or is not the NF that stored the model; s0286 names the error; s0292-s0293 require an OAuth2 access token for the API; s0295 gives one scope for the whole service.	s0083, s0286, s0292, s0293, s0295

ID	Name	About	What the documents say	Where
Z03	Authorisation bound to model and requester	The permission checked for a retrieval is the permission for the model identifier actually requested, held by the identity that actually sent the request, and for the retrieval action only; the ADRF does not use the permission of one model or one consumer for another.	The GET carries store-trans-id or model-unique-ids (s0078, s0169), the 403 test is per ML model (s0083), and allowConsumerList is a per-model attribute (s0255); s0295 gives only one service-wide scope, so nothing below the service level is bound by the token itself.	s0078, s0083, s0255, s0264, s0295
A03	201 answers the consumer's own storage request	The 201 Created a consumer accepts, with the representation of the record and the Location storeTransId, corresponds to the storage request that this consumer sent.	s0042 and s0045: the consumer POSTs a NadrfMLModelStoreRecord and the ADRF assigns a storeTransId; s0049 and s0165: the 201 carries the record representation and the Location with the storeTransId.	s0042, s0045, s0049, s0165
A08	Retrieved record answers the request	Each model entry in the 200 OK corresponds to the store-trans-id or model-unique-ids of the request, and its mlFileAddr is the address registered at storage for that modelUniqueId by the NF that owns it.	s0078: the GET identifies the records by store-trans-id or model-unique-ids; s0081: the 200 body carries MLModelInfo with modelUniqueId and the address of the ML model file stored in the ADRF.	s0078, s0081, s0169
D01	Messages cannot be substituted in transit	The content the ADRF handles is the content the consumer sent, and the content the consumer accepts is the content the ADRF returned; no other party substitutes a model identifier or a file address in transit.	s0042, s0049 and s0081 describe the request and response bodies; the protection of the SBI itself is delegated to TS 29.500 and TS 33.501 (s0292), which are outside the selected text.	s0042, s0049, s0081, s0292
K01	Access token stays secret	The OAuth2 access token (client credential) of an NF service consumer is not obtained by the attacker.	s0292-s0293: access is authorised by OAuth2 with the client credentials grant, the NRF acting as authorisation server; the consumer obtains a token before consuming the service.	s0292, s0293
A03.7	Store confirmation binds requester and storeTransId	The record returned in the 201 and its Location storeTransId belong to the storage request of this consumer, and the nfInstanceId/nfSetId in the record is this requester.	s0252: the NadrfMLModelStoreRecord carries nfInstanceId or nfSetId of the NWDAF containing MTLF; s0049 and s0165: the 201 returns the record and the Location with the storeTransId.	s0252, s0049, s0165
A08.8	Retrieval entries match requested identifiers	Every model entry of a retrieval response corresponds to a store-trans-id or model-unique-ids value of the request, and its mlFileAddr is the address registered at storage for that modelUniqueId by the NF that owns it.	s0078, s0081 and s0169 define the request identifiers and the returned MLModelInfo with modelUniqueId and mlFileAddr.	s0078, s0081, s0169
Z01.9	Unlisted consumer gets no stored record	With a fixed allowConsumerList and a fixed storing NWDAF, a consumer that is neither in the list nor the storer obtains no 200 OK record of that model.	s0083 and s0286: 403 RETRIEVAL_ML_MODEL_NOT_ALLOWED for a consumer that is not allowed and not the storing NF.	s0083, s0286
X01	Model identity bound to its storer	Which mlFileAddr a consumer receives when it retrieves a modelUniqueId that more than one NF service consumer has stored.	s0255 calls modelUniqueId the unique ML Model identifier; s0080 requires the ADRF to find the ML model(s) according to the requested parameters; s0048 states that the ADRF still creates a new record and assigns a new storeTransId for a model that is already stored.	s0048, s0080, s0255

Added from the documents by the analyst: X01.

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
K02	Payload and end-to-end confidentiality	Z01	
K03	Strong confidentiality / indistinguishability of secret values	Not applicable	no secret value whose indistinguishability is at stake is exchanged; the record fields are identifiers, and the token is covered by K01.
K04	Key purpose, direction and session separation	Not applicable	the service has no key hierarchy, direction or session derivation.
A01	Peer participation (aliveness)	A03	
A02	Weak consistency authentication	A03	
A04	Injective consistency authentication	A03	
A05	Implicit key authentication	Not applicable	the service establishes no session key.
A06	Explicit key confirmation	Not applicable	201 Created confirms the creation of a resource, not the possession of a key.
A07	Identity–credential–key binding	Z03	
R01	Authentication / challenge freshness	A03	
R02	Message-level anti-replay	A03	
R03	At-most-once business operation	Not applicable	s0048 states that the ADRF still creates a new record and assigns a new storeTransId for a model that is already stored, so the service makes no at-most-once claim.
R04	Message acceptance order / window safety	Not applicable	the service defines no sequence numbers, windows or ordering rules.
R05	Anti-downgrade / security policy preservation	Not applicable	the only negotiated feature (EnModelMgmt, s0290) adds an allowConsumerList and multiple results; it is not a security level that can be lowered.
C01	Forward secrecy (FS)	Not applicable	the service establishes no session key.
C02	Key-compromise impersonation resistance (KCI)	Not applicable	the service establishes no session key.
C03	Known-key security / compromise isolation	Not applicable	there is no multi-session key derivation or key hierarchy.
C04	Ephemeral secret / partial state compromise resilience	Not applicable	the documents describe no ephemeral secret or partial state leak for this service.
C05	Post-compromise security recovery (PCS)	Not applicable	the service has no post-compromise recovery mechanism.
C06	Enhanced authentication after registration credential compromise	Not applicable	the service is not a PAKE and holds no registration record of that kind.
C07	Offline dictionary attack resistance	Not applicable	no password or other low-entropy secret is used; the token is obtained from the NRF (s0293).
C08	Goal preservation under failure of a specified primitive	Not applicable	the documents state no goal under the failure of a primitive; the transport protection is delegated to TS 29.500/33.501 (s0292).

Entry	Property of the catalogue	Examined as	Why not
Z02	No expansion of delegated authority	Not applicable	the service describes no delegation chain or proxy that could widen a permission.
Z04	Revocation and expiry take effect	Outside the scope	deletion and revocation of permissions are explicitly left to a later round by the task statement.
Z05	Cross-tenant / cross-service security isolation	Z01	
S01	Encryption nonce uniqueness	Not applicable	the service uses no encryption nonce.
S02	Retransmission / key reinstallation safety	Not applicable	the service installs no key and defines no rekeying.
S03	Rollback resistance and state monotonicity	Not applicable	the service defines no version or epoch whose monotonicity could be rolled back.
S04	Cross-phase security context consistency	Not applicable	the service spans a single storage and retrieval phase and inherits no security context across phases.
S05	Termination and invalidation of old state	Outside the scope	deletion and revocation are left to a later round by the task statement.
P01	Identity anonymity	Not applicable	the parties are named NF instances whose identifiers are carried in the record; the documents state no anonymity requirement.
P02	Session unlinkability	Not applicable	the documents state no unlinkability requirement and the service uses fixed NF instance identifiers.
P03	User / credential existence hiding	Not applicable	the documents state no existence-hiding requirement; s0084 answers 204 No Content when a record does not exist.
V01	Conditional progress / error recovery	The prover cannot be asked	progress and error recovery are liveness properties that a symbolic trace model cannot be asked.
V02	No unintended deadlock	The prover cannot be asked	absence of deadlock is a liveness property that a symbolic trace model cannot be asked.
V03	Bounded resource abuse / amplification resistance	The prover cannot be asked	resource consumption and amplification are outside what counts of events can express.

The parties and the protected values, one by one

The host listed the 17 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: 9. Without: 8. The list follows from the properties; what was made of each entry is the analyst's judgement.

ID	Party or value	Kind	Attacker	Why none
S01	NF service consumer	party	`rogue_consumer`	
S02	ADRF	party	Everything lost by definition	a dishonest ADRF holds every record and answers retrieval itself; no property of its own behaviour can be asked, and the documents ask none under that assumption.
S03	NRF	party	Everything lost by definition	a compromised authorisation server mints a token for any NF identity, so every identity-based authorisation asked here is lost by definition; the documents say nothing of this case.
S04	Unauthorised NF service consumer	party	`rogue_consumer`	

ID	Party or value	Kind	Attacker	Why none
S05	ML model storage record	value	`rogue_consumer`	
S06	modelUniqueId	value	Nothing gained	modelUniqueId is an identifier chosen by the storing consumer and carried in requests and responses; knowing it gives no capability that the network attacker lacks.
S07	mlFileAddr	value	`rogue_consumer`	
S08	allowConsumerList	value	Nothing gained	allowConsumerList is supplied by the storing consumer in its own request and returned in the record to authorised consumers; holding it does not by itself authorise a retrieval, which Z01 answers.
S09	OAuth2 access token	value	`stolen_token`	
S10	storeTransId	value	Nothing gained	storeTransId is returned to the consumer that stored the record and appears in the Location header; it is not a credential, and Z01 checks the requester independently of it.
S11	Access token	value	`stolen_token`	
S12	NadrfMLModelStoreRecord	value	`rogue_consumer`	
S13	store-trans-id	value	Nothing gained	store-trans-id is a request parameter naming a record; knowing one gives no retrieval right, which Z01 checks.
S14	model-unique-ids	value	Nothing gained	model-unique-ids is a request parameter; knowing a modelUniqueId gives no retrieval right, which Z01 checks.
S15	Location	value	Nothing gained	the Location value is the record URI handed to the consumer that created the record; it carries no authorisation.
S16	Client credentials	value	`stolen_token`	
S17	NadrfMLModelStoreRecord.nfInstanceId/nfSetId	value	`rogue_consumer`	

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		
`rogue_consumer`	A valid OAuth2 access token for the single service scope nadrf-mlmodelmanagement, i.e. a registered NF service consumer, which is not in the allowConsumerList of the model it wants; it can send its own storage and retrieval requests and can choose the modelUniqueId, mlFileAddr and allowConsumerList of its own storage request.	s0295 defines a single scope for the entire service with no additional scopes at resource or operation level, so every consumer that obtains a token may invoke every operation; s0083 restricts retrieval only by the allowConsumerList and the storing NF; s0252/s0255 let the storing consumer supply modelUniqueId, mlFileAddr and allowConsumerList.	s0252, s0255, s0083, s0295

Attacker	Has	Basis	Where
`stolen_token`	The OAuth2 access token and client credential of the NF service consumer (NWDAF containing MTLF) that stored the model, obtained before the exchange; it can present that token to the ADRF over its own connection.	s0292-s0293: the ADRF authorises access by means of an OAuth2 bearer token issued by the NRF, so whoever holds the token is authorised as that consumer; the documents describe no token binding beyond the client credentials grant.	s0292, s0293

ID	`network`	`rogue_consumer`	`stolen_token`
Z01	Holds	Holds	Holds
Z03	Holds	Fails	Holds
A03	Holds	Holds	
A08	Holds	Fails	
D01	Holds	Holds	
K01	Holds		
A03.7	Holds	Holds	
A08.8	Holds	Fails	
Z01.9	Holds	Holds	
X01		Fails	

Source and targets

Source: 295 units, 54906 bytes, frozen in source-delivery-01.json. Targets: 6 approved (Z01, Z03, A03, A08, D01, K01); 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: Approve source package protocol-4d0742600750ac68a8578cc80b88

This step as it reports itself: 1 requests (1 failed), 0.000 USD, 42181 tokens in and 3875 out, 97 s.

Notes on the prover and the model

What the analyst named for targets:

step	targets	features	reason
21	Z01, Z03, A03, A08, D01, K01, A03.7, A08.8, Z01.9, X01	many_sessions, other	The properties are asked for any number of storage and retrieval sessions, with the ADRF record store modelled as a monotone table and the rogue consumer as a process with its own credential. No cryptographic primitive is modelled: the SBI is represented by one private channel per registered NF (the TLS/OAuth2 protection of TS 29.500/33.501), so no algebraic law is simplified and the answers do not rest on exclusive-or, Diffie-Hellman or a counter. No target rests on a value a party keeps and changes, so bounded.pv is not needed.

The analyst's closing note

Modelled the ADRF ML model store for one storage transaction and its retrieval (TS 29.520 4.3.2.2.2, 4.3.2.3.2, 5.2.3.2.3.1/2, 5.2.6.2.2/2.3, 5.2.9) and answered all ten targets, each claim under the attacker the target names. Proved with controls: the 201 answers the consumer's own storage request and carries the requester as nfInstanceId; a 200 goes only to the storing NWDAF or a consumer on the model's allowed list and only to a holder of a valid

token, and a client with no token is never served; the identity decided on is the identity that sent the request; retrieved entries match the requested modelUniqueId; the access token is not exposed. Found one weakness the documents do not state: any registered NF consumer with a token for the single service scope can register a second record under an existing modelUniqueId with its own mlFileAddr and an allowed list naming the target consumer, and the ADRF keeps both records and may serve the rogue one, so an authorised consumer downloads a file of the attacker's choosing (s0048 keeps a new record for an already stored model; nothing in the text binds the identifier to its storer). The reading, limits and an attack figure are recorded; the recommendation (one association per modelUniqueId, 409 Conflict, atomic check) is given but not checked with check_fix, because the failure depends on the arrival order and the sound change lies outside the modelled message flow. The results rest on the assumption that the 3GPP service based interface is authenticated and integrity protected (TS 29.500/33.501, outside the selected text) and on reading the 403 condition of s0083 as 'listed or storing NF'; retrieval by store-trans-id, multi-model records, partial failure, update and delete were not modelled.