Property	Comments	Adversary Assumption	Notable property Assumption	Insight	Result
service request-related properties					
1.1	For a service to be granted, consumer's serving slice info must be allowed by the producer.	OAMs present, No adversary NF update, adversaryDiscoveryRequest enabled.			Confused Producer Attack
1.2	Implementation idea: same as P1.1. the above counterexample is for specific nf type. So, other attacks might exist for specific producer type ATReq.	OAMs present, No adversary NF update, adversaryDiscoveryRequest enabled.	access token request not for specificNFtype		No counterexample
1.3	Same as P1.1. However, enabling benign update in producer's profile might bring about new attack.	OAMs present, No adversary NF update.	benign update = 3 of producer enabled for atmost once		Access Token Reuse attack, for specific nf type
1.4	Same as P1.3. However, the counterexample of P1.3 is for ATReq for specific nf type only. Toggling this might give new attacks.	OAMs present, No adversary NF update.	Toggled the reqForSpecificProducer control variable to off, benign update of producer enabled for atmost once.		Revoked Accesss Token Reuse Attack for specific nf instance.
1.5	Query: For a service to be granted, consumer's serving slice info must be allowed by the producer.	OAMs present, uConsumer = 3 once, adversaryDiscoveryRequest adversaryATRequest adversaryServiceRequest enabled.	Toggled the reqForSpecificProducer control variable to off, benign update of producer enabled for atmost once.		Confused Producer Attack
1.6	Query: For a service to be granted, consumer's serving slice info must be allowed by the producer.	OAMs present, uConsumer = 3 once, adversaryDiscoveryRequest adversaryATRequest adversaryServiceRequest enabled.	reqForSpecificProducer enabled		No counterexample
1.7	Query: For a service to be granted, consumer's serving slice info must be allowed by the producer.	OAMs present, uConsumer = 3 or 0 always, adversaryDiscoveryRequest adversaryATRequest adversaryServiceRequest enabled.	reqForSpecificProducer enabled		No counterexample
1.8		OAMs present, uConsumer = 3 or 2 or 0 once, adversaryDiscoveryRequest adversaryATRequest adversaryServiceRequest enabled.	reqForSpecificProducer enabled		Variation of attack 1 for AT req for specific nf type
1.9		OAMs present, uConsumer = 3 or 2 or 0 once, adversaryDiscoveryRequest adversaryATRequest adversaryServiceRequest enabled.	reqForSpecificProducer enabled		Access Token Reuse attack (Due to consumer's NF update)
1.10		OAMs present, uConsumer = 3 or 2 or 0 once, adversaryDiscoveryRequest adversaryATRequest adversaryServiceRequest disabled.	reqForSpecificProducer enabled		Access Token Reuse attack (Due to consumer's NF update)
1.11	For a nf service request to be granted, consumer's serving sNssais must be allowed in allowedSnssais attribute of the the producer's nfservice.	OAMs present, No adversary NF update			Confused producer attack in combination with Default Overprivilege Attack when consumer.sNssais is empty
1.12	For a nf service request to be granted, consumer's serving sNssais must be allowed in allowedSnssais attribute of the the producer's nfservice.	OAMs present, No adversary NF update	Both consumer's sNssais should not be none initally.		Confused producer attack
1.13	For a nf service request to be granted, consumer's serving sNssais must be allowed in allowedSnssais attribute of the the producer's nfservice.	OAMs present, No adversary NF update	Both consumer's sNssais should not be none initally, reqForSpecificProducer.		No counterexample
NF Discovery related properties					
2.1	For a nf service to be discovered, consumer's serving slice info must be allowed by the producer.	OAMs present, No adversary NF update.			No counterexample.
2.2	For a nf service to be discovered, consumer's serving slice info must be allowed by the producer.	OAMs present, No adversary NF Update, adversaryDiscoveryRequest enabled		P2.1 does not provide a counterexample probably because adversary discovery request update is disabled.	Default Overprivilege Attack(when requesterSnssai = none and consumer sNssais = none.)
2.3	For a nf service to be discovered, consumer's serving slice info must be allowed by the producer.	OAMs present, No adversary NF Update, adversaryDiscoveryRequest enabled	consumer sNssais != none at least initially	Let's consider consumer sNssais != none at least initially	NFDiscovery Bypass attack (requesterSnssais != none.)
2.4	For a nf service to be discovered, consumer's serving slice info must be allowed by the producer.	OAMs present, No adversary NF update, adversaryDiscoveryRequest enabled	consumer sNssais != none at least initially. requesterSnssais = none always.	Let's enforce this check even when requesterSnssais = none	NFDiscovery Bypass attack (even when requesterSnssais = none).
2.5	For a nf service to be discovered, consumer's serving slice info must be allowed by the producer.	OAMs present, No adversary NF update.	consumer sNssais != none at least initially. requesterSnssais = none always.	check if the counterexample is found even if adversaryDiscoveryRequest is disabled	No counterexample.
2.6	For a nf service to be discovered, consumer's serving slice info must be allowed by the producer.	OAMs present, allow adversary consumer update = 3 for at most once	consumer sNssais != none at least initially. requesterSnssais = none always.		No counterexample.
2.7	For a nf service to be discovered, consumer's serving slice info must be allowed by the producer.	OAMs present, allow adversary consumer update = 3.	consumer sNssais != none at least initially. requesterSnssais = none always.		No counterexample.
2.8	For a nf service to be discovered, consumer's serving slice info must be allowed by the producer	OAMs present, allow adversary consumer update = 2 once.	consumer sNssais != none at least initially. requesterSnssais = none always.		No counterexample.

2.9	For a nf service to be discovered, consumer's serving slice info must be allowed by the producer	OAMs present, allow adversary consumer update = 2.	consumer sNssais != none at least initially. requesterSnssais = none always.		No counterexample.
2.10	For a nf service to be discovered, consumer's serving sNssais must be allowed in allowedSnssais attribute of the the producer's nfservice provided that the allowedsNssais of the producer's service is not none.	OAMs present, No adversary NF update			Default Overprivilege Attack when consumer.sNssais is empty
2.11	For a nf service to be discovered, consumer's serving sNssais must be allowed in allowedSnssais attribute of the the producer's nfservice provided that the allowedsNssais of the producer's service is not none.	OAMs present, No adversary NF update, adversary discovery update enabled.	Consumer's sNssais should not be none anymore.	In P2.10, conusmer's sNssais are set None, so, let's see what happens if sNssais are not all none. also adversary discovery update is enabled.	NFDiscovery Bypass attack when requesterSnssais != none
2.12	For a nf service to be discovered, consumer's serving sNssais must be allowed in allowedSnssais attribute of the the producer's nfservice provided that the allowedsNssais of the producer's service is not none.				No counterexample.
2.13	For a nf service to be discovered, consumer's serving sNssais must be allowed in allowedSnssais attribute of the the producer's nfservice provided that the allowedsNssais of the producer's service is not none.	OAMs present, consumer update = 3, adversary discovery update enabled	Both consumer's sNssais should not be none anymore. consumer update = 3 (only auth attrib update) at most once.	Let's allow consumer update = 3 (only auth attrib update) at most once	No counterexample.
2.14	For a nf service to be discovered, consumer's serving sNssais must be allowed in allowedSnssais attribute of the the producer's nfservice provided that the allowedsNssais of the producer's service is not none.	OAMs present, consumer update = 2.	consumer update = 2 (some charactertic attrib and auth attib update e.g. sNssais, etc.) at most once.	Let's allow consumer update = 2 (only auth attrib update) at most once	No counterexample.
2.15	For a nf service to be discovered, consumer's serving sNssais must be allowed in allowedSnssais attribute of the the producer's nfservice provided that the allowedsNssais of the producer's service is not none.	OAMs present, consumer update = 2, adversary discovery update enabled.	Both consumer's sNssais should not be none anymore. requesterSnssais should not be empty anymore. consumer update = 2 (some charactertic attrib and auth attib update e.g. sNssais, etc.) at most once. adversarial nfdiscovery request update enabled	Let's allow adversarial nfdiscovery request update.	variant of NFDiscovery Bypass Attack (requesterSNssais != none)
2.16	For a nf service to be discovered, consumer's serving sNssais must be allowed in allowedSnssais attribute of the the producer's nfservice provided that the allowedsNssais of the producer's service is not none.	OAMs present, consumer update = 2, adversary discovery update enabled.	Both consumer's sNssais should not be none anymore initially. requesterSnssais should not be empty anymore. consumer update = 2 (some charactertic attrib and auth attib update e.g. sNssais, etc.) at most once. adversarial nfdiscovery request update enabled set requesterSNssais = sNssais of the consumer.	P2.15 manipultate requesterSNssais. Let's stop this.	Default overprivilege Attack (Only removal of sNssais from consumer's profile is required)
2.17	For a nf service to be discovered, consumer's serving sNssais must be allowed in allowedSnssais attribute of the the producer's nfservice provided that the allowedsNssais of the producer's service is not none.	OAMs present, No adversary NF update.	Both consumer's sNssais should not be none anymore initially. consumer update = 2 (some charactertic attrib and auth attib update e.g. sNssais, etc.) at most once.	Revert some changes. no adversarial disc req update anymore.	Default overprivilege Attack (Similar to P2. 16)
2.18	For a nf service to be discovered, consumer's serving sNssais must be allowed in allowedSnssais attribute of the the producer's nfservice provided that the allowedsNssais of the producer's service is not none.	OAMs present, No adversary NF update.	requesterSnssais should not be empty anymore. consumer update = 2 (some charactertic attrib and auth attib update e.g. sNssais, etc.) at most once. adversarial nfdiscovery request update enabled set requesterSNssais = sNssais of the consumer. consumer.sNssais != none at least once before nfdiscovery	As P2.17 makes consumer. sNssais = none again, let's consider the case when it is not none at least once.	Default overprivilege Attack (Similar to P2. 16)
2.19	if consumer initial slice = 1 and producer does not allow slice 1, producer will never be discovered.	OAMs present, No adversary NF update			NFDiscovery Bypass Attack when requesterSnssais = none (similar to P2.11)
2.20	if consumer initial slice = 1 and producer does not allow slice 1, producer will never be discovered.	OAMs present, No adversary NF update	requesterSNssais in NFDiscReq must not be none if consumer's sNssais != none	in P2.19, set requesterSNssais can be set empty by the consumer. let, enforce that it cannot be empty	No counterexample.
2.21	For a nf service to be discovered, consumer's serving sNssais must be allowed in allowedSnssais attribute of the the producer's nfservice provided that the allowedsNssais of the producer's service is not none.	OAMs present, No adversary NF update	requesterSnssais should not be empty anymore. consumer update = 2 (some charactertic attrib and auth attib update e.g. sNssais, etc.) at most once. adversarial nfdiscovery request update enabled set requesterSNssais = sNssais of the consumer. consumer.sNssais != none at least once before nfdiscovery	As P2.17 makes consumer. sNssais = none again, let's consider the case when it is not none at least once.	Default Overprivilege attack when requesterSnssais = none (similar to P2.11)
Access token request related properties					
3.1	For an accessTokenRequest to be granted, consumer's serving slice info must be allowed by the producer. .	OAMs present, No adversary NF update.			No counterexample
3.2	For a nf service to be discovered, consumer's serving slice info must be allowed by the producer	OAMs present, No adversary NF Update, adversaryATRequest enabled.			No counterexample
3.3	For an accessTokenRequest to be granted, consumer's serving slice info must be allowed by the producer. .	OAMs present, No adversary NF Update, adversaryATRequest enabled.	consumer sNssais != none at least initially.		No counterexample
3.4	For a nf service to be discovered, consumer's serving slice info must be allowed by the producer	OAMs present, No adversary NF update, adversaryATRequest enabled.	consumer sNssais != none at least initially. requesterSnssais = none always.		No counterexample
3.5	For an accessTokenRequest to be granted, consumer's serving slice info must be allowed by the producer. .	OAMs present, No adversary NF update.	consumer sNssais != none at least initially. requesterSnssais = none always.		No counterexample

3.6	For an accessTokenRequest to be granted, consumer's serving slice info must be allowed by the producer. .	OAMs present, allow adversary consumer update = 3 for at most once.	consumer sNssais != none at least initially. requesterSnssais = none always.		No counterexample
3.7	For an accessTokenRequest to be granted, consumer's serving slice info must be allowed by the producer. .	OAMs present, allow adversary consumer update = 3.	consumer sNssais != none at least initially. requesterSnssais = none always.		No counterexample
3.8	For an accessTokenRequest to be granted, consumer's serving slice info must be allowed by the producer.	OAMs present, allow adversary consumer update = 2 once, adversaryATRequest enabled.	consumer sNssais != none at least initially. requesterSnssais = none always.		No counterexample
3.9	For an accessTokenRequest to be granted, consumer's serving slice info must be allowed by the producer.	OAMs present, allow adversary consumer update = 2, adversaryATRequest enabled.	consumer sNssais != none at least initially. requesterSnssais = consumer's sNssai always.		No counterexample
3.10	For an accessTokenRequest to be granted, consumer's serving sNssais must be allowed in allowedSnssais attribute of the the producer's nfservice provided that the allowedsNssais of the producer's service is not none.	OAMs present, No adversary NF update			Default Overprivilege Attack when consumer.sNssais is empty
3.11	For a nf service to be discovered, consumer's serving sNssais must be allowed in allowedSnssais attribute of the the producer's nfservice provided that the allowedsNssais of the producer's service is not none.	OAMs present, No adversary NF update	Both consumer's sNssais should not be none initally.		No counterexample
3.12	For a nf service to be discovered, consumer's serving sNssais must be allowed in allowedSnssais attribute of the the producer's nfservice provided that the allowedsNssais of the producer's service is not none.	OAMs present, No adversary NF update	Both consumer's sNssais is none initally, requesterSnssais not not be empty.		No counterexample
3.13	For a nf service to be discovered, consumer's serving sNssais must be allowed in allowedSnssais attribute of the the producer's nfservice provided that the allowedsNssais of the producer's service is not none.	OAMs present, No adversary NF update, consumer update = 3 (only auth attrib update) at most once.	Both consumer's sNssais is none initally, requesterSnssais not not be empty.		No counterexample
3.14	For a nf service to be discovered, consumer's serving sNssais must be allowed in allowedSnssais attribute of the the producer's nfservice provided that the allowedsNssais of the producer's service is not none.	OAMs present, No adversary NF update, consumer update = 2 (only auth attrib update) at most once.			No counterexample.
3.15	For a nf service to be discovered, consumer's serving sNssais must be allowed in allowedSnssais attribute of the the producer's nfservice provided that the allowedsNssais of the producer's service is not none, adversaryATRequest enabled.	OAMs present, adversarial consumer update = 2 (only auth attrib update) at most once	Both consumer's sNssais is none initally, requesterSnssais not not be empty.		No counterexample
2-consumer-related properties					
4.1	Resources r meant to be exclusive to a specific nfinstance C1, should not be authorized to access by other nf instances e.g. C2.	OAMs present, No adversary NF update	diff consumers ask for access to the same operation		Parameter Misuse Attack (combined with default overprivilege attack)
4.2	Resources r meant to be exclusive to a specific nfinstance C1, should not be authorized to access by other nf instances e.g. C2.	OAMs present, No adversary NF update	diff consumers ask for access to the same operation, consumer's snssai cannot be none	let's suppress previous counterexample	Parameter Misuse Attack
4.3	Sensitive resource r meant to be exclusive to a specific nfinstance C1, should not be authorized to access by other nf instances e.g. C2.	OAMs present, No adversary NF update	Diff consumers may ask for access to the same/diff operations, consumer's snssai cannot be none	let's remove same operation restriction for the service requests	Parameter Misuse Attack (same as P1.2)
4.4	Sensitive resource r meant to be exclusive to a specific nfinstance C1, should not be authorized to access by other nf instances e.g. C2.	OAMs present, No adversary NF update	Diff consumers ask for access to the diff operations, consumer's snssai cannot be none. make servReqs to go for diff operations explicitly	explicitly make the service req different	Course Scope Attack
4.5	Sensitive resource r meant to be exclusive to a specific nfinstance C1, should not be authorized to access by other nf instances e.g. C2.				Course Scope Attack
4.6	Sensitive resource r meant to be exclusive to a specific nfinstance C1, should not be authorized to access by other nf instances e.g. C2.				Course Scope Attack