# RIASSUNTO ScasDK

## 1.0 Introduction

La tecnologia 5G offre prestazioni di connettività, bassa latenza e velocità senza precedenti. La novità rispetto alle generazioni precedenti è l'infrastruttura software-driven che permette maggiore flessibilità e scalabilità. Questo ecosistema dinamico è sostenuto da network software-defined (SDN), cloud computing e virtualized network functions (VFN). Assieme a questi vantaggi ci sono, però, anche delle sfide per quanto riguarda la sicurezza. Infatti, le caratteristiche specifiche delle reti cellulari hanno introdotto nuove vulnerabilità e vettori di attacco.
Per contrastare queste minacce e vulnerabilità, la Third Generation Partnership Project (3GPP), organizzazione che sovrintende la standardizzazione dei sistemi cellulari, più di 10 anni fa ha iniziato a sviluppare il Security Assurance Specification (SCAS). Quest'ultimo comprende una serie di test creati per valutare la sicurezza di vari componenti e funzioni della rete 5G, tramite induzione di comportamenti anormali per verificare la conformità ai protocolli nei casi limite. SCAS, la cui importanza è stata solo di contorno per le reti 4G, è diventato centrale con l'espansione del 5G.

## 1.1 Motivation

Finora, i test SCAS sono stati utilizzati principalmente per certificazioni come il NESAS (Network Equipment Security Assurance Scheme), che garantisce che apparecchiature e software soddisfino i requisiti di sicurezza del 3GPP. Questi test sono condotti offline in laboratori competenti e sono uno step fondamentale prima del rilascio di componenti o funzioni per l'utilizzo pubblico. Ora però, la natura software e cloud del 5G mette in discussione la validità dei test così svolti. Per questo motivo hanno preso sempre più piede metodi di testing basati su software e cloud computing, come DevSecOps, che incorporano test di sicurezza continui nel ciclo di vita dello sviluppo software. In più, DevSecOps permette parallelamente l'inclusione dei gestori telefonici nel processo di testing di sicurezza.
L'obbiettivo finale sarebbe di perpetrare test a runtime, anche in un ambiente di produzione, in modo da identificare e risolvere immediatamente problemi di sicurezza o funzionalità e garantire un monitoraggio costante dell'infrastruttura. Purtroppo questo approccio è fortemente in contrasto con le tecniche attuali, che isolano le singole Network Function (NF) per testarle.

## 1.2 Contribution

In questo saggio viene presentata una versione avanzata del framework ScasDK, che ora presenta nuove capacità di test e una miglior adattabilità a scenari realistici.
ScasDK include primitive avanzate che consentono ai tester di programmare e personalizzare i propri scenari di test. Questa capacità supporta l'implementazione della maggior parte dei test SCAS 3GPP, facilita la creazione di nuovi test e l'integrazione di diverse metodologie di test. In più, il framework astrae la complessità dell'infrastruttura di test, rendendola adatta a vari ambienti: dai tradizionali test offline a DevSecOps e persino ai test online.
Le novità più importanti sono:

- un miglior supporto di scenari reali grazie alla virtualizzazione
- una nuova infrastruttura scalabile del controller, capace di gestire più istanze di test
- interfaccia che permette a tester terzi di scrivere i propri test

- nuovi test SCAS, con annessa validazione
- convalida del modello di test ScasDK in ambienti DevSecOps attraverso una proof of concept completa
- valutazione dell'utilizzo della piattaforma per test online
- analisi e valutazione della fattibilità dell'utilizzo della piattaforma per i test online

## 2.0 Background

### 2.1 Architettura della rete 5G

La rete 5G core è composta da numerose funzioni di rete. Queste possono essere suddivise in un piano di controllo, responsabile della gestione degli utenti e delle sessioni, e un piano utente, responsabile del trasferimento dei dati. Solo il tipo UPF appartiene al piano utente, mentre la maggior parte delle NF appartiene al piano di controllo. Un'importante innovazione è l'architettura basata sui servizi, che consente alle varie funzionalità di rete del piano di controllo di operare in modo autonomo e di comunicare tra loro utilizzando un set di API standard basato su HTTP. Tuttavia le interfacce tra i piani di controllo e utente sono ancora punto-a-punto e impiegano protocolli specifici. In particolare, gNB e AMF comunicano tramite il protocollo NGAP per la supervisione della mobilità, delle sessioni e della gestione delle comunicazioni, mentre il protocollo NAS governa le comunicazioni di segnalazione tra UE e AMF. PFCP è il protocollo di controllo per stabilire, modificare e rilasciare sessioni di unità dati di protocollo (PDU), nonché per allocare e deallocare dinamicamente le risorse per i flussi di dati del piano utente. Infine, il protocollo GTP si occupa di incapsulare i dati utente durante il trasporto dal gNB all'UPF e viceversa.

### 2.2 Garanzie di sicurezza nel 5G

Grazie alla collaborazione tra 3GPP e GSMA è stata possibile la concretizzazione di Security Assurance Specification (SCAS) per le funzioni di rete 5G. 3GPP si è occupata di sviluppare le specifiche di test, mentre GSMA ha sviluppato il framework NESAS che offre una valutazione completa della sicurezza delle apparecchiature di rete.

### 2.3 Metodi di test

I test offline e online rappresentano due principi complementari nel testing: i primi rappresentano l'approccio standard, in cui il testing viene condotto in un ambiente controllato e isolato, consentendo una validazione completa delle funzionalità, la regressione e i test delle prestazioni senza influire sulla rete; ma senza rispecchiare a pieno le condizioni reali. Proprio per questo, i test online hanno recentemente acquisito rilevanza: questo approccio di testing si svolge direttamente nell'ambiente di produzione, fornendo un feedback immediato sul comportamento del sistema in condizioni reali.
Una tecnica di testing ortogonale è DevSecOps, che può essere applicata sia ai test offline che a quelli online. Si tratta di un approccio che integra la sicurezza nell'intero ciclo di vita IT; integrando le pratiche di sicurezza dall'inizio del processo di sviluppo, le organizzazioni possono identificare e mitigare proattivamente le vulnerabilità.
Nella rete cellulare 5G, ogni funzione di rete dispone di una propria pipeline di test in un'infrastruttura di integrazione/distribuzione continua (CI/CD). Quando il fornitore sviluppa e rilascia nuove funzionalità, l'infrastruttura compila il componente aggiornato, poi lo istanzia in un'infrastruttura di test che riproduce fedelmente lo scenario di produzione. In questa fase,

vengono eseguiti tutti i test relativi al componente. Nel caso di DevSecOps, questi test vengono estesi con routine ad hoc che verificano i requisiti di sicurezza del componente in esame. Se tutti i test hanno esito positivo, è consentito il rilascio e l'istanziazione del componente nell'ambiente di produzione.

## 2.4 Related Work

La sicurezza nelle reti cellulari è un campo critico, con un'ampia ricerca sulla sicurezza LTE. DoLTEst è un framework per il fuzzing del protocollo nel 4G, che si rivolge principalmente alle apparecchiature utente (UE). Richiede ai tester di definire manualmente i casi di test in base alle specifiche tecniche. ScasDK, invece, è specificamente progettato per il Core Network testing, facilitando le interazioni tra più NF.
Differenze tra ScasDK e DoLTEst:

- DoLTEst è a sorgente singola mentre ScasDK multipla, ciò accelera lo sviluppo dei test grazie anche ad una automatizzazione della loro esecuzione
- DoLTEst valuta la sicurezza attraverso anomalie di protocollo mentre ScasDK valuta anche aspetti non funzionali come vulnerabilità di temporizzazione

Molte ricerche sono state fatte sul fuzzing delle reti 5G, al contrario un numero molto limitato è orientato ai test SCAS, due esempi degni di nota sono le piattaforme sviluppate dalle aziende statunitensi Valid84 e Keysight Technologies.
Altri lavori si concentrano sulla generazione dei test piuttosto che sull'esecuzione, questi strumenti integrano ScasDK, che fornisce un ambiente di esecuzione automatizzato per tali casi di test.
Un'altra direzione di ricerca si concentra sulla formazione sulla sicurezza informatica per le reti 5G. Tramite strumenti didattici per creare scenari di attacco controllati, come VET5G, gli studenti possono esercitarsi a sfruttare le vulnerabilità della rete.
Dato che la rete core 5G condivide caratteristiche con le applicazioni basate su microservizi, l'adozione di pratiche CI/CD è stata ampiamente valutata in letteratura. Ciò è in linea con le tendenze del settore, dove i principali fornitori di reti stanno incorporando sempre più strategie DevOps per accelerare il time-to-market e migliorare l'efficienza operativa.

## 3.0 Requisiti di test e approccio

Basandoci su requisiti fondamentali estratti da3GPP, si può definire una strategia di test strutturata e versatile volta a fornire valutazioni di sicurezza complete. Questa strategia supporta i test SCAS e si estende anche per incorporare altre metodologie di test avanzate. I test SCAS per le NF delle reti core 5G consistono in una serie di operazioni di alto livello che devono essere eseguite dal tester.
Ci sono tre categorie di requisiti principali per l’automazione dei test:

- Attori coinvolti: ogni test coinvolge attori, controllati dal tester, che interagiscono attivamente con la NF testata, eseguendo attività specifiche come la modifica dei messaggi o il controllo delle risposte. Alcuni test SCAS coinvolgono un singolo attore, mentre altri ne richiedono più.
- Tipi di azioni: le azioni sono di tipo funzionale, che attivano interazioni tra i componenti, come l'avvio di una procedura o l'invio di un messaggio inaspettato, o non funzionale, che influiscono sulla qualità della comunicazione, come l'aggiunta di ritardi. Un test può richiedere sia azioni funzionali che non funzionali.
- Tipi di verifiche: le verifiche sono classificate come di occorrenza (verificano se un evento si è verificato) o temporali (verificano se gli eventi avvengono nell'ordine corretto).

## 4.0 Architettura di riferimento

I componenti principali di ScasDK includono:

- Proxy che catturano e modificano il traffico di rete per simulare scenari limite richiesti dai test.
- Emulatori che generano comportamenti di rete di base, come i pattern di traffico, per imitare le condizioni del mondo reale.
- Manager che supervisionano le entità di test come proxy ed emulatori, garantendone il corretto funzionamento durante l'esecuzione dei test.
- Il Controller che configura l'infrastruttura di test e coordina l'esecuzione dei casi di test.

La principale innovazione in questa versione aggiornata di ScasDK risiede nella riorganizzazione del componente controller che è stato suddiviso in due entità distinte:

- Infrastructure Controller: gestisce l'architettura di test nel suo complesso. È responsabile di attività come l'individuazione dei componenti di test, la configurazione della rete di test e la supervisione dell'infrastruttura.
- Test Controller: focalizzato sull'esecuzione della logica di test, questo controller orchestra le procedure di test, definisce i passaggi da eseguire e ne convalida i risultati.

La separazione delle responsabilità tra questi due controller consente ruoli più chiari e una migliore manutenibilità. Questa architettura semplifica anche il processo per i tester di terze parti, che non devono più gestire le complessità dell'infrastruttura, ma possono concentrarsi esclusivamente sullo sviluppo della logica di test.

## 4.1 Proxy

Al centro dell'architettura, il proxy funge da unità operativa primaria. Il suo ruolo fondamentale è quello di inoltrare messaggi tra due o più componenti e consente la registrazione di azioni di elaborazione, denominate hook. Gli hook sono le entità responsabili della gestione dell'elaborazione del traffico. Esistono due tipi distinti di hook: write e read. I primi comprendono azioni in grado di alterare la struttura del messaggio o del flusso di comunicazione, come l'introduzione di ritardi o l'eliminazione del messaggio stesso. In questo caso, il proxy attende una risposta prima di inoltrare il messaggio al destinatario. Gli hook di read, invece, includono operazioni che non influiscono sulla struttura del messaggio o sul flusso di comunicazione, come la registrazione e l'analisi delle comunicazioni. Qui, il proxy inoltra il messaggio sia all'hook che al destinatario simultaneamente.

Alla ricezione di un messaggio, il proxy elabora in sequenza gli hook di scrittura. Dopo l'elaborazione degli hook di scrittura, il proxy inoltra il messaggio al destinatario previsto e a tutti gli hook di lettura.

I proxy dovrebbero gestire i diversi tipi di protocolli utilizzati nelle reti core 5G. Questi includono:

- Protocollo NGAP/NAS: per le comunicazioni tra UE/gNB e AMF;
- Protocollo HTTP: per la comunicazione tra le funzioni di rete del piano di controllo, all'interno dell'architettura basata sui servizi;
- Protocollo PFCP: per la comunicazione tra SMF e UPF.

I tester possono sviluppare hook personalizzati all'interno del framework e, a seconda dei requisiti, possono implementare una logica stateless o stateful.

## 4.2 Emulators

Gli emulatori sono componenti che consentono la generazione di eventi standard all'interno della rete 5G, come la registrazione, l'autenticazione e altro ancora. Gli emulatori avviano un flusso di comunicazione standardizzato, che può poi essere modificato dai proxy per creare eventi specifici in base alle specifiche di test. Sebbene componenti come gNB e UE fungano da emulatori primari, è possibile sviluppare e integrare anche componenti aggiuntivi.

### 4.3 Manager

Il "manager" estende le capacità di proxy e emulatori, consentendo la gestione remota di operazioni come avvio, arresto e generazione di eventi, separando così il piano di controllo dell'infrastruttura dal piano di esecuzione dei test. Questo approccio migliora la manutenibilità e la riutilizzabilità del codice di test. In ScasDK, il manager non è più un'entità indipendente, ma stabilisce una connessione persistente con un "Controller" esterno. Il Controller invia configurazioni e operazioni al manager, semplificando la gestione delle risorse. Questo modello è ispirato a sistemi come il proxy HTTP Envoy e la rete 5G Aether, dove la configurazione avviene tramite un controller esterno.Tutto ciò migliora l'efficienza del processo.

### 4.4 Controller

Il Controller assume un ruolo fondamentale all'interno della piattaforma di test: il suo obiettivo principale comprende la gestione completa delle risorse di test, le impostazioni di configurazione, l'esecuzione dei test e altri aspetti correlati. Per ogni test condotto, il Controller inizialmente identifica e configura le risorse necessarie. Successivamente, il cattura il traffico attraverso gli hook e, se necessario, lo modifica di conseguenza per generare lo scenario richiesto. Dopo l'esecuzione del test, il valuta l'esito analizzando il traffico acquisito e presenta i risultati al tester.

#### 4.4.1 Infrastructure Controller

L'obiettivo principale del controller dell'infrastruttura è ottimizzare gli aspetti implementativi della configurazione dell'ambiente di test e della generazione di eventi. Attraverso un'interfaccia fornita dal controller, gli utenti specificano i propri requisiti e il controller li traduce in operazioni eseguibili dai componenti. Questa astrazione protegge il tester dai dettagli implementativi, semplificando l'intero processo di test.
L'interazione tra il controller e componenti gestiti segue uno schema di pubblicazione/sottoscrizione. Durante l'avvio, i componenti stabiliscono un contatto con il controller di riferimento e si sottoscrivono agli eventi a loro pertinenti, identificandosi ciascuno con un ID univoco. Ogni volta che il controller riceve una richiesta dall'utente, la invia al componente corrispondente per applicarla. Questo nuovo approccio offre diversi vantaggi come: identificazione e comunicazione con componenti più semplice, maggior scalabilità e interfaccia più intuitiva.

#### 4.4.2 Test Controller

Il test controller comprende tutta la logica di test all'interno della piattaforma. La sua responsabilità principale è configurare la piattaforma di test e interfacciarsi con il controller dell'infrastruttura. Nello specifico, imposta i parametri di rete e registra gli hook necessari per l'esecuzione del test. All’avvio del test vengono generati eventi e poi, tramite hook, monitorato il traffico, modificandolo se richiesto dai requisiti.

## 5. Implementazione della proof of concept

Attraverso questa proof of concept, vengono illustrate le funzionalità e le capacità principali della piattaforma di test, gettando le basi per una sua più ampia applicabilità e il potenziale per una perfetta integrazione nelle infrastrutture di test esistenti.

### 5.1. Tecnologie di deployment

Il framework di test utilizza container e Kubernetes per il deployment delle funzioni di rete 5G e dei componenti di controllo, seguendo un approccio cloud-native coerente con le tendenze attuali nelle reti core 5G industriali. Kubernetes gestisce l'infrastruttura, garantendo connettività, affidabilità e configurazione di rete. La piattaforma supporta anche ambienti alternativi (bare metal o macchine virtuali) e include punti di ingresso per permettere la comunicazione tra componenti esterni e il sistema di test. Helm permette di gestire tutte le risorse necessarie (deployment, servizi, configurazioni) in modo coerente, flessibile e riproducibile.

### 5.2 Proxy

In questa proof of concept sono stati implementati due dei tre proxy previsti dal framework: NGAP/NAS e HTTP, mentre il proxy PFCP non è ancora incluso. Tuttavia, la sua assenza non compromette i test SCAS per UPF e SMF, poiché solo 5 su 13 test lo richiedono.
Il proxy NGAP/NAS ha un'architettura modulare che facilita l'adattamento a scenari PFCP e supporta il passaggio da SCTP a UDP. Sono stati apportati miglioramenti nella gestione degli hook e dei socket, e ora il proxy può gestire più connessioni gNB contemporaneamente, abilitando test come l'handover.
Il proxy HTTP è basato su Envoy, un proxy open source altamente scalabile e adatto ad ambienti cloud-native. Envoy supporta HTTP/1.1, HTTP/2 e gRPC, e offre funzionalità avanzate come bilanciamento del carico, routing, e osservabilità. Grazie al supporto per processori esterni e all'uso di un'API gRPC, Envoy consente la modifica in tempo reale dei messaggi. Il controller gestisce il servizio gRPC necessario per l'elaborazione dinamica delle comunicazioni proxy.

### 5.3 Emulators

La piattaforma di test include tre emulatori principali per simulare componenti chiave di una rete 5G: funzioni di rete, gNB (gNodeB) e UE (User Equipment).

- Per le funzioni di rete, è stato sviluppato un emulatore HTTP client leggero, utile per test semplici e interazioni con singole funzioni.
- Per gNB e UE, viene utilizzato UERANSIM, strumento conforme alle specifiche 3GPP, che simula lo stack RAN usando UDP, eliminando la necessità di hardware RF.
- UERANSIM consente test flessibili, l'ispezione dello stato dei componenti e l'esecuzione di azioni come deregistrazione o creazione di sessioni PDU.

Questo approccio migliora l'efficienza, riduce i costi e amplia le possibilità di test in scenari 5G complessi.

### 5.4 Manager

Nella versione iniziale della piattaforma di test sono stati introdotti manager per i componenti proxy, gNB e UE. Mentre Envoy dispone già di un sistema di gestione centralizzato remoto per il proxy HTTP, i manager per gli altri componenti sono stati sviluppati da zero. I manager operano in due modalità:

1. Standalone:
    - Il manager funziona in modo autonomo.
    - Espone un'interfaccia gRPC per ricevere configurazioni e comandi da entità esterne.
    - Chi vuole interagire con il manager deve conoscere in anticipo il suo indirizzo.
2. Gestione remota:
    - Il manager non espone interfacce esterne.
    - Si connette automaticamente a un controller esterno all'avvio, mantenendo una connessione persistente.
    - Il controller invia configurazioni e comandi direttamente al manager.

## 5.5 Infrastructure Controller

Il controller dell'infrastruttura, pur essendo concettualmente un unico componente, è implementato in forma modulare, con vari moduli che usano tecnologie e interfacce differenti. Questa struttura facilita lo sviluppo e l'adattamento ai diversi requisiti dei componenti da gestire. Nella proof of concept, sono stati creati due controller distinti:

1. Controller per i proxy HTTP (Envoy):
    - È gestito tramite Istio, un service mesh avanzato per microservizi.
    - Istio consente un controllo dettagliato del traffico (routing, bilanciamento, filtraggio).
    - Usa Custom Resource Definitions (CRD) in Kubernetes per configurare le risorse.
    - La configurazione dell'Envoy External Processor API permette l'elaborazione remota dei messaggi (es. logging o modifica) secondo le specifiche di test.
2. Controller per UE, gNB e il proxy NGAP/NAS:
    - Sviluppato internamente, è composto da moduli dedicati alla gestione di ciascun componente.
    - Espone due interfacce:
        - Configurazione: usa anch'essa CRD su Kubernetes, allineandosi all'approccio di Istio e seguendo il modello CRUD (Create, Read, Update, Delete).
        - Comandi: è imperativa (richiesta → risposta immediata), quindi non è integrata con Kubernetes ma è realizzata con una interfaccia gRPC dedicata, simile a quella dei manager in modalità standalone.
    - I comandi vengono inviati attraverso la connessione persistente stabilita dal manager all'avvio, consentendo l'esecuzione diretta da parte del componente designato.

## 5.6 Test Controller

Il Test Controller della piattaforma di test si basa sul Robot Framework, un potente strumento pensato per l'automazione di test e processi. In questa piattaforma, è stato esteso con funzionalità mirate a supportare le esigenze particolari dei test SCAS, rendendo possibile la gestione completa

dei test all'interno di un’unica infrastruttura. Attraverso l'integrazione di librerie personalizzate, il test controller è in grado di interagire con tutti i componenti coinvolti nel processo di test: proxy, UE e gNB. Queste librerie fungono da interfaccia semplificata verso le API della piattaforma, nascondendo i dettagli tecnici e permettendo ai tester di concentrarsi sulla logica del test senza doversi preoccupare dell’implementazione sottostante. Il processo di test viene strutturato in tre fasi fondamentali:

- configurazione dell’ambiente, questa fase è responsabile della configurazione dei componenti di test, inclusi i parametri di rete e la creazione degli hook. Gli hook funzionano come moduli di codice che elaborano i messaggi e tracciano gli eventi di test rilevanti. Il tester definisce le condizioni per l'invocazione di questo gestore all'interno della funzione di elaborazione dei messaggi. La libreria degli hook gestisce autonomamente i trigger degli eventi, registrando ogni evento insieme a un timestamp per garantire che sia il suo verificarsi che la sua tempistica siano registrati accuratamente per la verifica del test,
- esecuzione delle azioni previste, prevede l'attivazione di azioni specifiche all'interno della rete in fase di test,
- verifica degli eventi generati, il sistema verifica che gli eventi previsti si siano verificati e lo abbiano fatto nel corretto ordine

Queste fasi però non sono necessariamente sequenziali: il modello adottato è infatti flessibile e iterativo, consentendo di ripetere o sovrapporre le fasi in base alle necessità specifiche del test. Oltre all’automazione e alla semplicità di scrittura dei test, un altro punto di forza del Robot Framework è la reportistica dettagliata che fornisce al termine dell’esecuzione. I report includono informazioni sul successo o il fallimento dei test, sulle prestazioni e su eventuali errori, offrendo così una visione chiara del comportamento del sistema testato e facilitando l’individuazione di problemi o discrepanze.

## 6. Risultati

Con i test della proof of concept è stato possibile valutare la sicurezza delle implementazioni, l’efficacia dell’approccio, l’usabilità del framework da parte di tester terzi e l’adattabilità della piattaforma.
Abbiamo sviluppato un totale di 17 casi di test. Questi rappresentano un set completo di test SCAS, ognuno dei quali soddisfa almeno uno dei requisiti di test. I casi di test sono classificati in base alla NF in fase di test.
Per convalidare il nostro approccio e valutare la sicurezza di diverse implementazioni di reti core 5G, abbiamo implementato la piattaforma di test su tre reti core open source: Open5GS v2.7.0, free5GC v3.3.0 e OAI v2.1.0. Abbiamo eseguito tutti i 17 test su queste implementazioni.
Per garantire la correttezza dell’implementazione, è stato acquisito il traffico di rete durante l'esecuzione e analizzato se le interazioni tra i componenti di rete fossero in linea con le azioni previste dal caso di test.

### Gestione degli errori di sincronizzazione - Caso B

Il test verifica che l’AMF, in caso di errore di sincronizzazione segnalato dall’UE durante l’autenticazione, non invii nuove richieste all’UE se l’AUSF non risponde entro il timeout previsto. Lo scenario simula questo comportamento bloccando intenzionalmente la risposta dell’AUSF. Gli hook configurati nei proxy NGAP e HTTP manipolano e monitorano i messaggi per forzare l’errore, ritardare la risposta e registrare gli eventi. Il test si articola in tre fasi (configurazione, azione,

verifica) e conferma che l’AMF rispetti il timeout prima di inviare una nuova richiesta, garantendo il corretto comportamento della rete in condizioni di errore.

#### 6.1.1. Risultati dei test

La maggior parte dei test SCAS fallisce su tutte e tre le implementazioni di rete core. Free5GC è la più conforme, superando 13 test su 17, seguita da Open5GS (9 su 17) e OAI (4 su 17). Nessun core soddisfa completamente i requisiti di sicurezza, evidenziando l'importanza di integrare questi test nel ciclo di sviluppo. L’analisi dei tempi di esecuzione mostra che i test su NRF e UDM sono più rapidi, mentre quelli su AMF richiedono più tempo a causa della maggiore complessità e dell’attesa di eventi o timeout. I test con ritardi artificiali prolungano ulteriormente i tempi.

### 6.2 Integrazione nell’ambiente DevSecOps

Il progetto è stato replicato su un repository GitLab, sfruttando le sue funzionalità CI/CD per automatizzare build, test e rilascio delle funzioni di rete. È stato configurato un cluster Kubernetes per ospitare sia le funzioni core di free5GC sia gli elementi ScasDK.
Per ogni funzione di rete (AMF, UDM, NRF) è stata creata una pipeline CI/CD dedicata, composta da tre fasi: build (creazione dell’immagine containerizzata), test (esecuzione dei test ScasDK tramite il test controller) e rilascio. I test generano report dettagliati per la verifica e la diagnostica. La separazione tra infrastruttura CI/CD e ScasDK facilita l'integrazione con altri strumenti DevSecOps.

### 6.3 Evoluzione delle performance

La sezione valuta l’impatto dei proxy sulle prestazioni di connessione UE alla rete core, confrontando scenari con e senza proxy in un ambiente Kubernetes. I test sono stati condotti su un’infrastruttura con free5GC e ScasDK, utilizzando pod Kubernetes e proxy gestiti da Istio sul piano di controllo.
Sono stati analizzati quattro scenari (nessun proxy, solo NGAP, solo HTTP, entrambi i proxy), ciascuno con almeno 50 cicli di connessione. I risultati mostrano che Istio offre buone prestazioni grazie alla sua service mesh, che riduce la latenza tramite connessioni persistenti. Il proxy NGAP, invece, ha mostrato performance inferiori, come previsto per una semplice proof-of-concept non ottimizzata. I test confermano che l’uso dei proxy non compromette significativamente l’esecuzione dei test online.
Nel complesso, le prestazioni del proxy NGAP sono soddisfacenti, anche se migliorabili. Nel test con implementazione combinata dei proxy, i risultati sono paragonabili allo scenario base, confermando la fattibilità di scenari di test con proxy attivi. Le variazioni nei dati sono più marcate con cinque UE, a causa del campione ridotto, mentre con venti UE i tempi di connessione si stabilizzano dopo una fase iniziale più irregolare.

## 7.0 Conclusione

L’articolo presenta lo sviluppo e la valutazione di ScasDK, un framework completo per il testing della sicurezza nelle reti 5G. Sono stati implementati 17 casi di test e validati su tre Core Network evidenziando che nessuna implementazione supera tutti i test. Il framework si dimostra versatile, accessibile e integrabile in contesti DevSecOps, con prestazioni compatibili con ambienti di produzione.