

Software Testing

Petrolo Melissa- 0286160

Report di testing dei progetti BOOKKEEPER e SYNCOPE

Setup dei progetti

BookKeeper: Per prima cosa ho eseguito un fork da GitHub del progetto, clonandolo sul mio sistema ed eliminando tutti i file di build '.yml' e tutte le cartelle di test, come consigliato. Dopo aver eseguito la build tramite mvn clean package con successo, ho configurato:

- il profile di Jacoco che viene richiamato nel file jacoco.yml, necessario per eseguire la build su GitHub Actions inviando l'analisi dell'esecuzione dei test su SonarCloud; il report aggregate viene prodotto nel modulo di test-jacoco;
- il profile di Pit per il mutation testing, specificando le classi di test, che viene richiamato nel file di build pit.yml;
- il profile di ba-dua.

Syncope: Anche per questo progetto, ho eseguito il fork e il clone del repository in locale eliminando tutti i file di build '.yml' e tutti i file di test. In questo caso, ho dovuto modificare alcuni pom per la corretta compilazione del comando mvn clean install. Ho configurato:

- il profile di Jacoco con il file di build jacoco.yml come in Bookkeeper inviando l'analisi dei test su SonarCloud;
- il profile di Pit, specificando le classi da testare, che viene anch'esso richiamato dal file di build pit.yml;
- per quanto riguarda ba-dua, per questo progetto non sono riuscita a configurarlo a causa dell'incompatibilità di versione java tra questo e syncope.

Scelta delle classi

In entrambi i progetti, ho escluso le classi di test. Per la scelta delle classi ho considerato la metrica numFix del dataset ottenuta dalla feature selection, ma questa non è stata sufficiente per selezionare le classi da testare e per questo motivo ho deciso di associare questa metrica anche a sizeLoc e locTouched del codice. Inoltre, ho escluso le classi che non hanno metodi con almeno un parametro.

BookKeeper: ho scelto come classe org.apache.bookkeeper.client.BookKeeperAdmin.java, perché nonostante la dimensione del codice e il numero di loc toccati siano alti, il numero di fix in alcune release risulta essere minimo. Mentre, per org.apache.bookkeeper.bookie.BufferedChannel.java perché nell'ultima release i loc toccati rispetto alla dimensione del codice è molto alta ma non risulta nessun fix associata a quella classe.

Syncope: ho scelto come classe org.apache.syncope.core.spring.Encryptor.java poiché è una classe che non è mai stata toccata e non ci sono fix, e la classe org.apache.syncope.core.spring.AuthDataAccessor.java perché in alcuni casi ci sono molti loc toccati rispetto alla dimensione del codice con un numero basso di fix.

Implementazione dell'insieme di test basata su category partition

BookKeeper è un servizio che implementa il salvataggio persistente di flussi di "log entries" in sequenze di entries chiamate ledgers. Queste entries vengono replicate in più server (bookie) per fare in modo che i dati siano resistenti a una grande varietà di guasti (crash, corruzioni, ecc...).

Apache Syncope è un sistema Open Source per la gestione delle identità digitali negli ambienti aziendali.

BookKeeper

Le classi scelti per BookKeeper sono: BufferedChannel.java e BookKeeperAdmin.java.

BufferedChannel

La classe BufferedChannel.java fornisce un buffering per un FileChannel; questa classe non presenta molti metodi da implementare, per questo motivo mi sono soffermata su:

- public void write (ByteBuffer src) throws IOException.
- public synchronized int read (ByteBuffer dest, long pos, int length) throws IOException;

CLASSE: org.apache.bookkeeper.bookie.BufferedChannel.java

METODO: public void write (ByteBuffer src)

Questo metodo scrive tutti i dati in un src per un FileChannel. L'unico parametro di questo metodo è ByteBuf che è un oggetto che astrae una sequenza di zero o più byte e di conseguenza ho individuato le tre partizioni {buffer vuoto}, {buffer non vuoto} e {NULL}.



Come prima cosa, ho creato un file temporaneo che verrà eliminato al termine dell'esecuzione, e successivamente ritorno un FileChannel associato a questo file per poter eseguire operazioni di scrittura e lettura. Di seguito, ho istanziato un BufferedChannel, e scritto dei byte random all'interno del canale; per fare ciò è stato necessario introdurre nuovi parametri nel costruttore poiché BufferedChannel richiede una capacità e un *unpersistedBytesBound* che serve per stabilire se effettuare un'operazione di flush quando il suo valore è non zero ed è minore di *unpersistedBytes*:

Parametro	Descrizione	Classi di equivalenza
src	Buffer contenente i dati che devono essere scritti	{NULL}, {VALID}, {EMPTY}
capacity	Capacità del buffer di scrittura in fase di creazione di BufferedChannel	{>capacità src}, {=capacità src}, {< capacità src}
unpersistedBytesBound	Utilizzato per richiamare il metodo flush()	{<0L, =>0L}

Basandomi su queste osservazioni, per il metodo write() ho deciso di progettare i seguenti casi di test:

SRC	CAPACITY	UNPERSISTEDBYTESBOUND	
NULL	0	0L	NullPointerException poiché non è possibile ottenere il numero di bytes scritti se il buffer è nullo
VALID	20	0L	Si ha una capacità del BufferedChannel minore della capacità del ByteBuf e mi aspetto di avere 10 come bytes rimanenti nel buffer dato che non viene effettuato nessuna operazione di flush
VALID	20	2L	Si ha una capacità del BufferedChannel minore della capacità del ByteBuf e mi aspetto di avere 0 come risultato dato che viene effettuato l'operazione di flush sul buffer
VALID	100	0L	Si ha una capacità del BufferedChannel maggiore della capacità del ByteBuf e mi aspetto di avere 50 come risultato dato che non viene effettuato l'operazione di flush sul buffer
VALID	100	2L	Si ha una capacità del BufferedChannel maggiore della capacità del ByteBuf e mi aspetto di avere 0 come risultato dato che viene effettuato l'operazione di flush sul buffer
VALID	50	0L	Si ha una capacità del BufferedChannel uguale alla capacità del ByteBuf e mi aspetto di avere 0 dato che vengono scritti esattamente 50 bytes nel buffer; in questo caso non viene neanche effettuata l'operazione di flush
VALID	50	2L	Si ha una capacità del BufferedChannel uguale alla capacità del ByteBuf e mi aspetto di avere 0 dato che vengono scritti esattamente 50 bytes nel buffer
EMPTY	50	0L	Con un ByteBuf vuoto mi aspetto di avere 0 bytes rimanenti nel write buffer

Le metriche utilizzate sono *STATEMENT COVERAGE* e la *CONDITION COVERAGE*, entrambe fornite da JaCoCo. Implementando i casi di test progettati finora, ottengo dal report di JaCoCo i seguenti valori per le due metriche:

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
● write(ByteBuf)		100%		90%	1	6	0	21	0	1

Ottingo una copertura SC pari a 20/21 (95%) e una CC pari a 9/10 (90%). Per quanto riguarda PIT, il report mostra che solo 9/13 (69%) mutanti sono stati uccisi.

Da una osservazione white-box del codice, ho cercato di implementare altri casi di test considerando il parametro di `unpersistedBytesBound` in relazione al valore di `unpersistedBytes`, aggiungendo come classe di equivalenza `{> unpersistedBytes}`, `{= unpersistedBytes}`, `{< unpersistedBytes}`.

SRC	CAPACITY	UNPERSISTEDBYTESBOUND	
VALID	100	100L	Mi aspetto di avere 50 bytes rimanenti nel writeBuffer in quanto vengono scritti solamente i primi 50 bytes ma dato che <code>unpersistedBytes</code> è minore di <code>unpersistedBytesBound</code> non viene effettuata l'operazione di flush
VALID	100	50L	Mi aspetto di avere 0 bytes rimanenti nel writeBuffer in quanto vengono scritti i primi 50 bytes, viene effettuata l'operazione di flush poiché <code>unpersistedBytesBound</code> è uguale a <code>unpersistedBytes</code> e successivamente vengono scritti gli ultimi 50 bytes
VALID	100	100L	Mi aspetto di avere 50 bytes rimanenti nel writeBuffer in quanto vengono scritti solamente i primi 50 bytes ma dato che <code>unpersistedBytes</code> è minore di <code>unpersistedBytesBound</code> non viene effettuata l'operazione di flush

In questo modo, aggiungendo questi tre casi di test, ottengo dal report di JaCoCo una copertura SC e una copertura CC pari al 100%; mentre sul report di PIT ottengo che 11/13 (84%) mutanti sono stati uccisi. È possibile osservare anche il report di Ba-dua Figure 6 da cui è possibile notare che sono state coperte la maggior parte delle coppie def-use.

CLASSE: org.apache.bookkeeper.bookie.BufferedChannel.java
METODO: public void read (ByteBuf dest, long pos, int length)

Questo metodo legge tanti byte nel ByteBuf in base alla sua capacità a partire dalla posizione nel FileChannel.

Parametro	Descrizione	Classi di equivalenza
dest	Buffer di destinazione	{VALID}, {EMPTY}, {NULL}
pos	Posizione da cui iniziare la lettura	{<capacità write buffer}, {>capacità write buffer}, {=capacità write buffer}
length	Numero di byte che devono essere letti	{< capacità write buffer - pos}, {< capacità write buffer - pos}, {= capacità write buffer - pos}

Un oggetto `ByteBuf` astrae una sequenza di zero o più byte e di conseguenza ho individuato le tre partizioni {VALID}, {EMPTY}, {NULL}. Il parametro "pos" indica la posizione da cui partire per leggere i dati con una classe di equivalenza `{<capacità dest}`, `{=>capacità dest}` aspettandomi una eccezione nel momento in cui cerco di accedere ad una posizione maggiore della capacità del `ByteBuf` di scrittura. Il parametro "length", considerando solo il nome del parametro senza conoscere l'implementazione del codice all'interno del metodo, potrebbe rappresentare il numero di byte che devono essere letti ma successivamente nel momento in cui osservo il codice in maniera white-box noto che il parametro serve per verificare se è strettamente maggiore di zero e viene decrementato da un valore pari al numero di byte che sono stati copiati nell'iterazione, quindi, posso considerare questo parametro come il minimo tra il numero di byte ancora scrivibili all'interno del buffer

utilizzato per la lettura e il numero di byte all'interno del write buffer a partire dalla posizione specificata fino al valore del writerIndex del write buffer stesso avendo come classe di equivalenza $\{ \leq \text{capacità dest} - \text{pos} \}$, $\{ < \text{capacità dest} - \text{pos} \}$ aspettandomi una eccezione nel momento in cui si cerca di leggere un numero di bytes maggiore rispetto alla differenza tra la capacità del ByteBuf e la posizione in cui si vuole iniziare la lettura.

Come prima cosa, ho creato un file temporaneo che verrà eliminato al termine dell'esecuzione, e successivamente ritorno un FileChannel associato a questo file per poter eseguire operazioni di scrittura e lettura. Di seguito, ho istanziato un BufferedChannel, e scritto dei byte random all'interno del canale.

Basandomi su queste osservazioni, per il metodo read() ho deciso di progettare i seguenti casi di test definendo un parametro $NUM_{BYTES} = 80$ per la capacità del ByteBuf di scrittura e di destinazione:

DEST	POS	LENGTH	
NULL	0L	2	Exception, in quanto non posso leggere un ByteBuf nullo;
EMPTY	0L	0	Mi aspetto un valore pari a 0 dato che il ByteBuf è vuoto e non ha nessuna capacità da riempire
VALID	$NUM_{BYTES} - 10$	5	Mi aspetto un valore di ritorno pari a 10 data dalla differenza del write buffer e la posizione in cui si intende iniziare la lettura che è maggiore rispetto a length
VALID	$NUM_{BYTES} - 1$	$NUM_{BYTES} + 1$	Exception, in quanto la differenza tra la capacità del ByteBuf e la posizione in cui si intende leggere è minore rispetto a length
VALID	$NUM_{BYTES} - 1$	1	Mi aspetto un valore di ritorno pari a 1, infatti il minimo tra la differenza della capacità del write buffer e la posizione che è uguale al valore di length è esattamente al valore di quest'ultimo
VALID	$NUM_{BYTES} + 1$	10	Exception, in quanto la posizione a cui si intende accedere è maggiore rispetto alla capacità del ByteBuf di scrittura
VALID	NUM_{BYTES}	0	Mi aspetto un valore di ritorno pari a 0, in quanto la posizione in cui si intende leggere è esattamente uguale alla capacità del buffer di scrittura

Implementando questi casi di test, ottengo dal report di JaCoCo:

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
● read(ByteBuf, long, int)		73%		66%	5	10	7	26	0	1

Ottenendo una copertura SC pari al 14/25 (56%) e una copertura CC pari a 10/18 (55%); mentre, sul report di PIT noto che solo 15/26 (57%) mutanti sono stati uccisi. È possibile osservare il metodo sul report JaCoCo Figure 3 e sul report Pit Figure 4.

Quindi, osservando questi report ho deciso di aggiungere altri casi di test tenendo in considerazione il parametro `writeBufferStartPosition` che risulta essere pari a 0 in un caso normale e di utilizzare Mockito per mockare:

- la classe **ByteBufAllocator**, in questo modo riesco a restituire un writeBuffer nullo attraverso `Mockito.when(byteBufAllocator.directBuffer()).thenReturn(null)` utilizzato come parametro nell'istanziamento di BufferedChannel;
- a classe **FileChannel**, per ritornare una posizione diversa da zero attraverso `Mockito.when(fileChannel.position()).thenReturn(WRITE_BUFFER_START_POSITION)` in modo tale da avere un `writeBufferStartPosition` pari a 2.

Definisco una costante *WRITE_BUFFER_START_POSITION* = 2:

DEST	POS	LENGTH	FILE CHANNEL	
<i>VALID</i>	<i>-NUM_BYTES</i>	10	<i>FILE_CHANNEL</i>	Exception, quanto si intende accedere ad una posizione negativa e in questo caso il valore pos è inferiore al <i>writeBufferStartPosition</i>
<i>VALID</i>	0L	10	<i>FILE_CHANNEL</i>	Mi aspetto un valore di ritorno pari a 80 considerando un valore pos esattamente uguale a <i>writeBufferStartPosition</i>
<i>VALID</i>	<i>WRITE_BUFFER_START_POSITION</i> + 1	1	<i>MOCK_FILE_CHANNEL</i>	Mi aspetto un valore di ritorno pari a 79, in quanto la posizione nel buffer è data dalla differenza del valore pos e del valore <i>writeBufferStartPosition</i> e il minimo valore tra la differenza del <i>writeIndex</i> e la posizione nel buffer con i bytes scrivibili nel buffer di destinazione è esattamente 79
<i>VALID</i>	<i>WRITE_BUFFER_START_POSITION</i> - 1	1	<i>MOCK_FILE_CHANNEL</i>	Mi aspetto un valore di ritorno pari a 80, in quanto la differenza tra pos e <i>writeBufferStartPosition</i> è -1 e quindi il minimo valore tra la differenza del <i>writeIndex</i> e la posizione nel buffer con i bytes scrivibili nel buffer di destinazione è 80
<i>VALID</i>	<i>WRITE_BUFFER_START_POSITION</i>	1	<i>MOCK_FILE_CHANNEL</i>	Mi aspetto un valore di ritorno pari a 80, in quanto la differenza tra pos e <i>writeBufferStartPosition</i> è 0 e quindi il minimo valore tra la differenza del <i>writeIndex</i> e la posizione nel buffer con i bytes scrivibili nel buffer di destinazione è 80
<i>MOCK_BUFFER</i>	<i>WRITE_BUFFER_START_POSITION</i> + 1	1	<i>MOCK_FILE_CHANNEL</i>	Mi Aspetto un valore di ritorno pari a 0, considerando il valore di <i>writeBufferStartPosition</i> minore del valore pos e con un mock indicando un <i>writeBuffer</i> nullo
<i>MOCK_BUFFER</i>	<i>WRITE_BUFFER_START_POSITION</i> - 1	1	<i>MOCK_FILE_CHANNEL</i>	Exception, in quanto il <i>ByteBuf</i> è nullo

MOCK_BUFFER WRITE_BUFFER_START_POSITION

1

MOCK_FILE_CHANNEL

Mi Aspetto un valore di ritorno pari a 0, considerando il valore di *writeBufferStartPosition* uguale al valore pos e con un mock indicando un *writeBuffer* nullo

In questo modo, aggiungendo questi casi di test ottengo una copertura SC pari a 22/25 (88%) e una copertura CC pari a 16/18 (88%); mentre sul report di PIT, ottengo 19/26 (73%) mutanti uccisi.

Non sono riuscita ad individuare altri casi di test che mi permettessero di aumentare le percentuali con JaCoCo, con Badua e con PIT. È possibile notare anche il report di Ba-dua Figure 7 da cui è possibile osservare una buona copertura delle coppie def-use.

BookKeeperAdmin

BookKeeperAdmin è una classe che fornisce metodi per amministrare il cluster di server bookies. I metodi implementati sono:

- *public static boolean initBookie(ServerConfiguration conf)*
- *public static boolean areEntriesOfLedgerStoredInTheBookie(long ledgerId, BookieId bookieAddress, LedgerMetadata ledgerMetadata)*

CLASSE: org.apache.bookkeeper.client.BookKeeperAdmin.java

METODO: public static boolean initBookie(ServerConfiguration conf)

Parametro	Descrizione	Classi di equivalenza
conf	Configurazione che gestisce le impostazioni lato server.	{null}, {new ServerConfiguration}

Questo metodo inizializza un bookie, assicurandosi che *journalDir*, *ledgerDir* e *indexDirs* siano vuoti e che non ci siano Bookie registrati con questo BookieId. *JournalDir* è la directory in cui Bookkeeper invia il suo log write-ahead, idealmente su un dispositivo dedicato; il valore predefinito è *"/tmp/bk-txn"*. *LedgerDir* è la directory del ledger e della directory del journal. *IndexDir* è la directory per memorizzare i file di indice; se non specificato, bookie utilizzerà *ledgerDirectories* per archiviare i file di indice.

Il parametro "conf" gestisce la configurazione delle impostazioni lato server e ha una classe di equivalenza {null}, {new ServerConfiguration}, aspettandoci che con un *ServerConfiguration* nullo ritorni una eccezione. Inoltre, è un tipo di dato complesso in quanto estende la classe *AbstractConfiguration<ServerConfiguration>*, quindi è ragionevole considerare questa classe di equivalenza per le fasi di testing ed eventualmente estenderla per migliorare la coverage. Solo questo parametro non è sufficiente per creare casi di test; di conseguenza, ho utilizzato come altri parametri una stringa che indica l'id da assegnare a BookieId, e dei valori booleani per coprire diversi casi di test per verificare che *journalDir*, *ledgerDir* e *indexDirs* siano effettivamente vuoti.


Parametro	Descrizione	Classi di equivalenza
conf	Configurazione che gestisce le impostazioni lato server.	{NULL}, {new ServerConfiguration}
bookieID	Id da assegnare al nuovo BookieId	{NULL}, {stringa vuota}, {stringa non vuota}
addJournal	Booleano per creare cartella per Journal	{true}, {false}
addLedger	Booleano per creare cartella per Ledger	{true}, {false}
addIndexConf	Booleano per inserire il nome alla cartella Index	{true}, {false}
addIndex	Booleano per creare cartella per Index	{true}, {false}

Nella documentazione, viene esplicitato che bisogna istanziare il cluster dei metadati gestito da Zookeeper per mantenere i metadati dei relativi bookies. Per la configurazione di ServerConfiguration ho utilizzato la classe TestBKConfiguration.

Di seguito, ho implementato i miei casi di test definendo una costante *BOOKIE_ID* = "*BookieId*":

CONF	BOOKIE ID	ADD JOURNAL	ADD LEDGER	ADD INDEX CONF	ADD INDEX	
<i>NULL</i>	<i>BOOKIE</i>	False	False	False	True	Exception, in quanto non esiste una configurazione per ServerConfiguration;
new ServerConfiguration	<i>NULL</i>	False	False	False	True	Exception, l'id del bookie non può essere <i>NULL</i> ;
new ServerConfiguration	""	False	False	False	True	Exception, l'id del bookie non può essere vuota
new ServerConfiguration	<i>BOOKIE</i>	True	False	False	True	False, poiché esiste una cartella in journalDir
new ServerConfiguration	<i>BOOKIE</i>	False	True	False	True	False, poiché esiste una cartella in LedgerDir;
new ServerConfiguration	<i>BOOKIE</i>	False	False	True	False	True, non esiste nessuna cartella
new ServerConfiguration	<i>BOOKIE</i>	False	False	True	True	False, poiché esiste una cartella in IndexDir.
new ServerConfiguration	<i>BOOKIE</i>	False	False	False	True	True, non esiste nessuna cartella

Possiamo vedere il report di JaCoCo:

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed
initBookie(ServerConfiguration)		100%		100%	0	5	0	11	0

Con solo questi casi di test ho ottenuto una copertura SC pari a 14/22 (64%) e una copertura CC pari a 9/10 (80%); mentre, considerando il report di PIT ho ottenuto che 9/13 (69%) di mutanti sono stati uccisi; risultano dei valori differenti tra la figura sopra e le percentuali appena descritte perché il report non considera le linee di return. È possibile notare anche il report di Ba-dua Figure 9 da cui è possibile osservare una buona copertura delle coppie def-use.

CLASSE: org.apache.bookkeeper.client.BookKeeperAdmin.java

METODO: public static boolean areEntriesOfLedgerStoredInTheBookie(long ledgerId, BookieId bookieAddress, LedgerMetadata ledgerMetadata)

Parametro	Descrizione	Classi di equivalenza
ledgerId	L'ID del ledger in cui è stata scritta l'entry	{null}, {istanza valida}, {istanza non valida}
bookieAddress	Identificativo per il bookieId	{null}, {new BookieId}
ledgerMetadata	Rappresenta i metadati lato client di un ledger	{null}, {new LedgerMetadata}

Non c'è documentazione su questo metodo e osservando il codice, mi accorgo che in realtà il parametro ledgerId non viene utilizzata ma è comunque necessaria per la creazione di oggetti *LedgerMetadata* attraverso *LedgerMetadataBuilder*. Per prima cosa, creo quattro diversi *BookieId* attraverso la classe

BookieSocketAddress nel formato “*hostname:port*” e creo due diverse liste di BookieId aggiungendo alla prima i primi tre BookieId creati e al secondo gli ultimi tre. Costruisco anche un *LedgerMetadata* con *ledgerID*, *ensembleSize*, *writeQuorumSize* che deve essere maggiore o uguale a *ensembleSize*, *ackQuorumSize* che deve essere minore o uguale a *writeQuorumSize*, *digestType*, due *ensembleEntry*, che deve essere uguale alla dimensione della lista BookieId, assegnando alla prima come *firstEntry* 0 e la prima lista e al secondo come *firstEntry* l'ultimo entry assegnato + 1 specificando quest'ultimo come *lastEntry*. È possibile osservare questa configurazione nella Figure 1.

BOOKIEID	LEDGER METADATA	
bookieAddress	ledgerMetadata	True, le prime entries risultano essere nel segment registrati nel Bookie;
bookieAddress	<i>NULL</i>	Exception, non è possibile prendere tutti i bookie in quanto non esiste un <i>LedgerMetadata</i> ;
bookie1	ledgerMetadata	True, le prime entries risultano essere nel segment registrati nel Bookie;
<i>NULL</i>	ledgerMetadata	False, non esiste nessun bookie;
bookie3	ledgerMetadata	False, poiché bookie3 fa parte del secondo insieme di BookieId e non risulta essere memorizzato nel Bookie.

Possiamo vedere il report di JaCoCo:

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed
areEntriesOfLedgerStoredInTheBookie(long, BookieId, LedgerMetadata)		100%		100%	0	4	0	10	0

Considerando questi casi di test sono riuscita a raggiungere una copertura SC pari a 10/10 (100%) e una copertura CC pari a 6/6 (100%); mentre, osservando il report di PIT ottengo che 6/6 (1005) mutanti sono stati uccisi. È possibile notare anche il report di Ba-dua Figure 8 da cui è possibile osservare una buona copertura delle coppie def-use.

Syncope

Le classi scelte per Syncope sono: *Encryptor.java* e *AuthDataAccessor.java*.

Encryptor

La classe *Encryptor.java* permette di codificare, verificare e decodificare attraverso algoritmi di cifratura. Dato che i metodi sono molto piccoli ho deciso di sceglierne tre:

- *public String encode(final String value, final CipherAlgorithm cipherAlgorithm)*
- *public boolean verify (final String value, final CipherAlgorithm cipherAlgorithm, final String encoded)*
- *public String decode(final String encoded, final CipherAlgorithm cipherAlgorithm)*

CLASSE: org.apache.syncope.core.spring.security.Encryptor.java

METODO: public String encode(final String value, final CipherAlgorithm)

Parametro	Descrizione	Classi di equivalenza
value	Stringa utilizzata per codificare	{NULL}, {stringa vuota}, {stringa non vuota}
cipherAlgorithm	Algoritmo di cifratura	{NULL}, {SHA}, {SHA1}, {SHA256}, {SHA512}, {AES}, {SMD5}, {SSHA}, {SSHA1}, {SSHA256}, {SSHA512}, {BCRYPT}



Il parametro “value” è una stringa che deve essere cifrata da un algoritmo di cifratura aventi come classe di equivalenza {NULL}, {stringa vuota}, {stringa non vuota}. Il parametro “cipherAlgorithm” è un algoritmo di cifratura che può assumere valori {NULL}, {new CipherAlgorithm}, ma nel nostro caso consideriamo una classe

di equivalenza {NULL}, {SHA}, {SHA1}, {SHA256}, {SHA512}, {AES}, {SMD5}, {SSHA}, {SSHA1}, {SSHA256}, {SSHA512}, {BCRYPT}.

Di seguito, ho implementato questi casi di test:

VALUE	CIPHER ALGORITHM	
<i>NULL</i>	<i>NULL</i>	False, non viene effettuata nessuna cifrata poiché il valore stringa è nulla
<i>NULL</i>	<i>AES</i>	False, non viene effettuata nessuna cifrata poiché il valore stringa è nulla
<i>NULL</i>	<i>BCRYPT</i>	False, non viene effettuata nessuna cifrata poiché il valore stringa è nulla
<i>""</i>	<i>BCRYPT</i>	True, la stringa viene cifrata con l'algoritmo
<i>""</i>	<i>NULL</i>	True, la stringa viene cifrata anche senza algoritmo di cifratura
"Test"	<i>BCRYPT</i>	True, la stringa viene cifrata con l'algoritmo
"Test"	<i>NULL</i>	True, la stringa viene cifrata con l'algoritmo
"Test"	<i>SHA</i>	True, la stringa viene cifrata con l'algoritmo
"Test"	<i>SHA1</i>	True, la stringa viene cifrata con l'algoritmo
"Test"	<i>SHA256</i>	True, la stringa viene cifrata con l'algoritmo
"Test"	<i>SHA512</i>	True, la stringa viene cifrata con l'algoritmo
"Test"	<i>SSHA1</i>	Exception, si verifica nel momento in cerca di ottenere il bean in <i>ApplicationContextProvider</i>
"Test"	<i>SSHA256</i>	Exception, si verifica nel momento in cerca di ottenere il bean in <i>ApplicationContextProvider</i>
"Test"	<i>SSHA512</i>	Exception, si verifica nel momento in cerca di ottenere il bean in <i>ApplicationContextProvider</i>
"Test"	<i>SMD5</i>	Exception, si verifica nel momento in cerca di ottenere il bean in <i>ApplicationContextProvider</i>

Anche per il progetto di Syncope ho utilizzato le metriche *STATEMENT COVERAGE* e la *CONDITION COVERAGE*; di seguito il report di JaCoCo:

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Cxty	Missed Lines	Missed Methods
● encode(String, CipherAlgorithm)		100%		100%	0	5	0

Con questi casi di test ottengo una copertura SC pari a 9/9 (100%) e una copertura CC pari 8/8 (100%); mentre, nel report di PIT ottengo il 100% dei mutanti uccisi.

CLASSE: org.apache.syncope.core.spring.security.Encryptor.java

METODO: public boolean verify(final String value, final CipherAlgorithm cipherAlgorithm, final String encoded)

Parametro	Descrizione	Classi di equivalenza
value	Stringa utilizzata per codificare	{null}, {stringa vuota}, {stringa non vuota}
cipherAlgorithm	Algoritmo di cifratura	{NULL}, {SHA}, {SHA1}, {SHA256}, {SHA512}, {AES}, {SMD5}, {SSHA}, {SSHA1}, {SSHA256}, {SSHA512}, {BCRYPT}



encoded	Stringa cifrata	{==value, ==cipherAlgorithm}, {==value, != cipherAlgorithm}, {!=value, ==cipherAlgorithm}, {!=value, !=cipherAlgorithm}
---------	-----------------	---

Questi parametri corrispondono esattamente al caso descritto nel metodo encode. Il funzionamento in questo caso è simile solo che viene verificata che la stringa value nel momento in cui sia cifrata corrisponde esattamente alla stringa cifrata.

VALUE	CIPHER ALGORITHM	ENCODED	
<i>NULL</i>	<i>AES</i>	{ <i>NULL</i> , <i>AES</i> }	False, dato che una stringa nulla non può essere cifrata;
"Test"	<i>AES</i>	{"Test", <i>AES</i> }	True, la cifratura viene verificata
" "	<i>BCRYPT</i>	{" ", <i>BCRYPT</i> }	True, la cifratura viene verificata
" "	<i>BCRYPT</i>	{" ", <i>BCRYPT</i> }	True, la cifratura viene verificata
"Test"	<i>BCRYPT</i>	{"Test", <i>BCRYPT</i> }	True, la cifratura viene verificata
"Test"	<i>AES</i>	{"Test", <i>BCRYPT</i> }	False, dato che l'algoritmo di cifratura è diverso da quello specificato dal parametro ci aspettiamo infatti che non corrispondano;
<i>NULL</i>	<i>BCRYPT</i>	{"Test", <i>AES</i> }	False, dato che l'algoritmo di cifratura è diverso da quello specificato dal parametro ci aspettiamo infatti che non corrispondano;
"Test"	<i>NULL</i>	{"Test", <i>AES</i> }	True, la cifratura viene verificata
"Test"	<i>SHA</i>	{"Test", <i>SHA</i> }	True, la cifratura viene verificata
"Test"	<i>SHA1</i>	{"Test", <i>SHA1</i> }	True, la cifratura viene verificata
"Test"	<i>SHA256</i>	{"Test", <i>SHA256</i> }	True, la cifratura viene verificata
"Test"	<i>SHA512</i>	{"Test", <i>SHA512</i> }	True, la cifratura viene verificata
"Test"	<i>SSHA1</i>	{"Test", <i>SSHA1</i> }	False, si verifica nel momento in cerca di ottenere il bean in <i>ApplicationContextProvider</i> e restituisce un errore poiché non riesce a verificare il valore codificato
"Test"	<i>SSHA256</i>	{"Test", <i>SSHA256</i> }	False, si verifica nel momento in cerca di ottenere il bean in <i>ApplicationContextProvider</i> e restituisce un errore poiché non riesce a verificare il valore codificato
"Test"	<i>SSHA512</i>	{"Test", <i>SSHA512</i> }	False, si verifica nel momento in cerca di ottenere il bean in <i>ApplicationContextProvider</i> e restituisce un errore poiché non riesce a verificare il valore codificato
"Test"	<i>SMD5</i>	{"Test", <i>SMD5</i> }	False, si verifica nel momento in cerca di ottenere il bean in <i>ApplicationContextProvider</i> e restituisce un errore poiché non riesce a verificare il valore codificato

"Test"	SSHA	{"Test", SSHA}	False, si verifica nel momento in cerca di ottenere il bean in <i>ApplicationContextProvider</i> e restituisce un errore poiché non riesce a verificare il valore codificato
--------	------	----------------	--

Di seguito il report di JaCoCo:

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Cxty	Missed Lines	Missed Methods
verify(String CipherAlgorithm, String)		87%		100%	0	5	2

Con questi casi di test ottengo una copertura SC pari a 8/10 (80%) e una copertura CC pari 8/8 (100%); mentre, nel report di PIT ottengo il 100% dei mutanti uccisi.

CLASSE: org.apache.syncope.core.spring.security.Encryptor.java

METODO: public String decode(final String encoded, final CipherAlgorithm cipherAlgorithm)



Parametro	Descrizione	Classi di equivalenza
encoded	Stringa cifrata	{==value, ==cipherAlgorithm}, {!=value, !=cipherAlgorithm}, {!=value, ==cipherAlgorithm}, {!=value, !=cipherAlgorithm}
cipherAlgorithm	Algoritmo di cifratura	{NULL}, {SHA}, {SHA1}, {SHA256}, {SHA512}, {AES}, {SMD5}, {SSHA}, {SSHA1}, {SSHA256}, {SSHA512}, {BCRYPT}

Questi parametri corrispondono esattamente al caso descritto nel metodo encode. Questo metodo permette di decodificare una stringa cifrata attraverso algoritmi di cifratura che sono gli stessi nei casi precedenti. Come stringa cifrata si prende in considerazione il metodo encode, e si verifica che la stringa decodifica sia uguale alla stringa che abbiamo definito come stringa predefinita, *PASSWORD_VALUE* = "password".

ENCODED	CIPHER ALGORITHM	
{NULL, AES}	AES	False, la decodifica non avviene poiché il valore codificato non corrisponde dato che value è differente
{"", AES}	AES	False la decodifica non avviene poiché il valore codificato non corrisponde dato che value è vuota
{"password", AES}	AES	True, ci aspettiamo un valore decodificato
{"9Pav + xl + UyHt02H9ZBytiA ==", AES}	AES	False, la decodifica non avviene poiché il valore codificato non corrisponde dato che value è differente
{"password", AES}	BCRYPT	False, come previsto dato che gli algoritmi non corrispondono il valore codificato non è uguale al valore decodificato
{"password", AES}	NULL	False, come previsto dato che gli algoritmi non corrispondono il valore codificato non è uguale al valore decodificato
{"password", BCRYPT}	BCRYPT	False, poiché l'unico algoritmo implementato nel metodo è AES;
{"passwordSbagliata", BCRYPT}	BCRYPT	False, poiché l'unico algoritmo implementato nel metodo è AES;
NULL	BCRYPT	False, poiché l'unico algoritmo implementato nel metodo è AES;

<code>{"password",SHA}</code>	<i>SHA</i>	False, poiché l'unico algoritmo implementato nel metodo è AES;
<code>{"password",SHA1}</code>	<i>SHA1</i>	False, poiché l'unico algoritmo implementato nel metodo è AES;
<code>{"password",SHA256}</code>	<i>SHA256</i>	False, poiché l'unico algoritmo implementato nel metodo è AES;
<code>{"password",SHA512}</code>	<i>SHA512</i>	False, poiché l'unico algoritmo implementato nel metodo è AES;
<code>{"password",SSHA}</code>	<i>SSHA</i>	False, poiché l'unico algoritmo implementato nel metodo è AES;
<code>{"password",SSHA1}</code>	<i>SSHA1</i>	False, poiché l'unico algoritmo implementato nel metodo è AES;
<code>{"password",SHA256}</code>	<i>SHA256</i>	False, poiché l'unico algoritmo implementato nel metodo è AES;
<code>{"password",SHA512}</code>	<i>SHA512</i>	False, poiché l'unico algoritmo implementato nel metodo è AES;

Di seguito il report di JaCoCo:

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Cxty	Missed Lines	Missed Methods
● <code>decode(String, CipherAlgorithm)</code>		100%		100%	0 3	0 6	0 1

Con questi casi di test ottengo una copertura SC pari a 6/6 (100%) e una copertura CC pari 4/4 (100%); mentre, nel report di PIT ottengo il 100% dei mutanti uccisi.

AuthDataAccessor

I metodi implementati della classe AuthDataAccessor sono:

- `public Tuple<User, Boolean, String> authenticate (final String domain, final Authentication authentication)`
- `public Set<SyncopeGrantedAuthority> getAuthorities(final String username, final String delegationKey)`

CLASSE: org.apache.syncope.core.spring.security.AuthDataAccessor.java

METODO: `public Tuple<User, Boolean, String> authenticate (final String domain, final Authentication authentication)`

Parametro	Descrizione	Classi di equivalenza
domain	dominio	{ <i>NULL</i> }, {stringa vuota}, {stringa non vuota}
authentication	autenticazione	{ <i>NULL</i> }, {new Authentication}

Questo metodo tenta di autenticare le credenziali fornite rispetto all'archiviazione interna e alle risorse pass-through.

Prima di procedere all'implementazione dei casi di test, ho creato una classe entity *MyUser* contenenti tutte le informazioni necessarie per l'autenticazione, quali: username, password e ho utilizzato Mockito per creare un mock su UserDao in modo tale da restituirmi l'utente tramite `Mockito.when(userDao.findByUsername(any())).thenReturn(user)`.

Utilizzando questi due parametri, ho implementato questi casi di test:

DOMAIN	AUTHENTICATION
--------	----------------

<i>NULL</i>	<i>new Authentication</i>	Exception, dato che i parametri di configurazione risultano nulli;
"Test"	<i>new Authentication</i>	L'utente riesce ad autenticarsi con username e password;
"Test"	<i>NULL</i>	Exception, poiché l'autenticazione è nulla
""	<i>new Authentication</i>	L'utente riesce ad autenticarsi con username e password;

Con questi casi di test, ottengo dal report di JaCoCo una copertura SC pari 8/39 (20%) e una copertura CC pari a 8/24 (33%); mentre dal report di PIT ottengo che 8/21 (38%) di mutanti sono stati uccisi.

Da una osservazione white-box, ho cercato di implementare nuovi casi di test aggiungendo alla classe *MyUser* ulteriori parametri come:

- *CipherAlgorithm*;
- *failedLogins* cioè il numero di volte che l'autenticazione è fallita prima di poter avere un esito positivo;
- *status*;
- *lastModifier* cioè l'ultima modifica che l'utente ha modificato username o password;
- *lastLoginData* cioè l'ultimo accesso.

Ho implementato il mock su:

- ***RealDAO***;
- ***AnySearchDAO*** in modo tale da restituire il numero di utenti registrati superiori ad uno, quindi, restituendo una lista inserendo altri utenti tramite *Mockito.when(anySearchDAO.search(any(SearchCond.class), any(AnyTypeKind.class))).thenReturn(userList)*;
- ***ConfParamOps*** ritornando questi tre valori tramite *Mockito.when(confParam.Ops.get(anyString(), eq("authentication.attributes"), any(), any()).thenReturn(new String[]{username})* per quanto riguarda gli attributi dell'autenticazione, *Mockito.when(confParam.Ops.get(any(), eq("authentication.status"), any(), any()).thenReturn(new String[]{"ACTIVE", SUSPENDED, NULL})* per ottenere i diversi stati dell'utente, *Mockito.when(confParam.Ops.get(anyString(), eq("log.lastlogindata"), any(), any()).thenReturn(true)* per ottenere un valore true in modo tale da definire il nuovo valore di *lastLoginDate*.

Tutti questi mock sono stati specificati all'interno del costruttore di *AuthDataAccessor*. È possibile vedere la classe *AuthDataAccessorMock* nella Figure 10. – AuthDataAccessor – Classe Mock

Adesso, invece di considerare solamente i due parametri del metodo ne ho aggiunti altri in modo tale da aumentare le percentuali dai report di JaCoCo e Pit:

Parametro	Descrizione	Classi di equivalenza
domain	dominio	{ <i>NULL</i> }, {stringa vuota}, {stringa non vuota}
authentication	Il tipo di autenticazione, in modo tale che ci siano diversi casi di errore o corretta autenticazione	{ <i>NULL</i> }, { <i>ACTIVE</i> }, { <i>ACTIVE_PASSWORD_WRONG</i> }, { <i>ACTIVE_PASSWORD_WRONG</i> }, { <i>NO_USER</i> }, { <i>NO_AUTHENTICATION</i> }, { <i>IS_SUSPENDED</i> }, { <i>IS_FAILED_LOGINS</i> }, { <i>STATUS</i> }
<i>confParaOps</i>	per coprire la linea di codice 214 in modo tale da restituire un valore di configurazione "username" diversa da quella del metodo	{ <i>="username"</i> }, { <i>!= "username"</i> }
<i>numUsers</i>	Il numero di utenti da considerare nel mock <i>anySeacrhDAO</i> in modo tale da restituire una dimensione della lista maggiore o uguale a 1.	{ <i>= 1</i> }, { <i>> 1</i> }

Considero i nuovi casi di test definendo una costante *DOMAIN = "Master"*:

DOMAIN	AUTHENTICATION TYPE	CONFPARAMOPS	NUM USERS
--------	---------------------	--------------	-----------

DOMAIN	NO_USER	"username"	1	In questo caso ho definito che dato che l'utente non esiste allora anche l'autenticazione non avviene;
DOMAIN	NO_AUTHENTICATION	"username"	1	Exception, poiché l'autenticazione è <i>NULL</i>
DOMAIN	ACTIVE_PASSWORD_WRONG	"username"	1	L'autenticazione non avviene dato che la password non è corretta
DOMAIN	ACTIVE_USERNAME_WRONG	"username"	1	L'autenticazione è positiva, anche se dato che l'username non è corretto dovrebbe dare una eccezione. Quindi penso che ci sia un errore perché l'autenticazione in realtà non dovrebbe avvenire
DOMAIN	IS_SUSPENDED	"username"	1	L'autenticazione non avviene
DOMAIN	IS_FAILED_LOGINS	"username"	1	Dopo un numero di tentativi di accesso, l'autenticazione ha esito positivo
DOMAIN	STATUS	"username"	1	Exception, l'utente non ha il permesso di autenticarsi
DOMAIN	ACTIVE	"username"	2	L'autenticazione avviene
DOMAIN	ACTIVE	"different-username"	2	Exception, il parametro di configurazione è diverso da "username" quindi l'errore è dovuto al fatto che username e questo valore non corrispondano

Aggiungendo questi casi di test, ottengo dal report di JaCoCo una copertura SC pari a 34/39 (87%) e una copertura CC pari 21/24 (87%); mentre, dal report di PIT ottengo che 16/21 (76%) mutanti vengono uccisi.

CLASSE: org.apache.syncope.core.spring.security.AuthDataAccessor.java

METODO: public Set<SyncopeGrantedAuthority> getAuthorities(final String username, final String delegationKey)

Parametro	Descrizione	Classi di equivalenza
username	Username dell'utente	{ <i>NULL</i> }, {stringa vuota}, {stringa non vuota}
delegationKey	Chiave di delegazione	{ <i>NULL</i> }, {stringa vuota}, {stringa non vuota}

Definisco due costanti *USERNAME* = "username" e *DELEGATION_KEY* = "delegationKey":

USERNAME	DELEGATION KEY	
<i>NULL</i>	<i>NULL</i>	Exception, username utente non esiste e delegazione nulla
<i>USERNAME</i>	<i>NULL</i>	L'utente ottiene l'autorità

NULL

DELEGATION_KEY L'utente ottiene l'autorità

Con solo questi casi di test, ottengo dal report di JaCoCo una percentuale di copertura SC e copertura CC pari allo 0%; quindi, ho deciso di osservare il metodo attraverso una modalità white-box e di considerare ulteriori parametri per poter aumentare questa percentuale dai report.

Parametro	Descrizione	Classi di equivalenza
username	Username dell'utente	{NULL}, {stringa vuota}, {stringa non vuota}
delegationKey	Chiave di delegazione	{NULL}, {stringa vuota}, {stringa non vuota}
anonymousUser	Utente anonimo	{==username}, {!=username}
adminUser	Utente amministratore	{==username}, {!=username}
isDelegationFound	Permette di specificare se la delegazione esiste	{true}, {false}
isFoundUser	Permette di specificare se l'utente esiste	{true}, {false}
isEmptyRole	Permette di specificare se l'utente ha un ruolo	{true}, {false}

Ho implementato due ulteriori mock:

- **SecurityProperties** che mi restituisce una stringa per quanto riguarda l'utente anonimo tramite `Mockito.when(securityProperties.getAnonymousUser()).thenReturn(anonymousUser)` e una stringa per quanto riguarda l'utente amministratore tramite `Mockito.when(securityProperties.getAdminUser()).thenReturn(adminUser)`.
- Ho creato un'altra classe entity `MyDelegation` con parametro `User` utilizzato successivamente dal mock **DelegationDAO** che mi restituisce la delegazione trovata o una eccezione in base al parametro booleano che ho inserito come parametro nel costruttore, quindi se true `Mockito.when(delegationDAO.find(delegationKey)).thenReturn(myDelegation)` altrimenti `Mockito.when(delegationDAO.find(delegationKey)).thenThrow(new UsernameNotFoundException());`
- Ho creato uno `spy(myDelegation)`, creando un mock anche su **Role** e successivamente `Mockito.doReturn(roleSet).when(delegationMock).getRoles()` e `Mockito.when(delegationDAO.find(delegationKey)).thenReturn(delegationMock)`.

È possibile vedere la classe `AuthDataAccessorMock` nella Figure 10. – AuthDataAccessor – Classe Mock.

Di seguito, ho implementato i nuovi casi di test considerando questi ulteriori parametri:

USERNAME	DELEGATION KEY	ANONYMOUS USER	ADMIN USER	IS DELEGATION FOUND	IS FOUND USER	IS EMPTY ROLE	
USERNAME	DELEGATION_K.	USERNAME	USERNAME	False	True	False	Exception che si verifica quando non trova l'utente
USERNAME	DELEGATION_K.	USERNAME	USERNAME	True	True	False	L'utente ottiene l'autorità in modo anonimo
USERNAME	DELEGATION_K.	USERNAME	USERNAME	False	True	False	Exception, l'autorità è nulla
USERNAME	DELEGATION_K.	USERNAME + "a"	USERNAME	True	True	False	L'utente ottiene l'autorità attraverso

							autorità amministrative
<i>USERNAME</i>	<i>DELEGATION_K.</i>	<i>USERNAME</i> + "a"	<i>USERNAME</i> + b"	True	True	False	L'utente ottiene autorità
<i>USERNAME</i>	<i>DELEGATION_K.</i>	<i>USERNAME</i> + "a"	<i>USERNAME</i> + b	False	True	False	Exception, utente non ha il permesso di ottenere l'autorità poiché non è stato trovato. Considero anonymousUser e adminUser diverso da username e delegationKey non nulla
<i>USERNAME</i>	<i>DELEGATION_K.</i>	<i>USERNAME</i> + "a"	<i>USERNAME</i> + b	True	True	True	Exception, poiché il valore "username" non corrisponde al valore di USERNAME dell'utente
<i>USERNAME</i>	<i>DELEGATION_K.</i>	<i>USERNAME</i> + "a"	<i>USERNAME</i> + b	True	True	False	L'utente ottiene autorità
<i>USERNAME</i>	<i>NULL</i>	<i>USERNAME</i> + "a"	<i>USERNAME</i> + b	True	False	True	Exception, poiché non si riesce a trovare l'utente associato. Considero anonymousUser e adminUser diverso da username e delegationKey nulla
<i>USERNAME</i>	<i>NULL</i>	<i>USERNAME</i> + "a"	<i>USERNAME</i> + b	True	True	False	Exception, GroupDAO risulta essere <i>NULL</i>

Con questi nuovi casi di test ottengo dal report di JaCoCo una copertura SC pari a 11/15 (73%) e una copertura CC è 8/8 (100%); mentre, dal report di PIT ottengo che 4/7 (57%) dei mutanti sono stati uccisi.

Stima della reliability

Per la stima della reliability ho pensato che i set di parametri dei test case per ogni metodo siano gli unici profili operazioni e che ogni parametro dei casi di test abbia una probabilità uniforme di essere dato in input al metodo.

Per determinare la reliability, ho semplicemente calcolato la media delle reliability dei metodi testati.

BookKeeper

- BufferedChannel
 - write(): solo un caso di test su 11 si aspetta un errore, quindi il metodo ha una reliability del 90%;
 - read(): dei 15 casi di test si attendono 5 eccezioni, quindi la reliability è pari al 67%.

Per la classe BufferedChannel, la reliability è del 78,5%.

- BookKeeperAdmin
 - initBookie(): ci sono solo 8 casi di test con due eccezioni, quindi la reliability è del 75%;

- areEntriesOfLedgerStoredInTheBookie(): solo una eccezione su 5 casi di test con una reliability del 80%.

Per la classe BookKeeperAdmin, la reliability è del 77,5%.

Syncopé

- Encryptor
 - encode(): in questi casi di test solo 4 su 15 ha una eccezione, quindi la reliability è del 73%;
 - verify(): dei 17 casi di test in 5 si attende un errore, quindi la reliability è del 70%;
 - decode(): nessuno dei 17 casi di test si attende un errore e il test non invoca nessuna eccezione, quindi la reliability è del 100%;

Per la classe Encryptor, la reliability è del 81%.

- AuthDataAccessor
 - authenticate(): dei 13 casi di test, 4 causano una eccezione e un errore; ma in particolare, in questo metodo mi sono resa conto che nel momento in cui si cerca di autenticarsi con un username non corretto, l'autenticazione avviene comunque, quindi il valore atteso non è quello che si aspettiamo ma posso supporre che ci sia magari una caratteristica che permetta di effettuare l'autenticazione con un username non corretto o questo sia effettivamente un bug. In questo caso, secondo le mie supposizioni considero questo caso di test come un errore, quindi ottengo una reliability del 69%;
 - getAuthorities(): 7 casi di test su 13 si attendono una eccezione, quindi la reliability è del 46%.

Per la classe AuthDataAccessor, la reliability è del 57,5%.

Link

GITHUB BOOKKEEPER	https://github.com/emelis-ptr/software-testing-bookkeeper
GITHUB SYNCOPÉ	https://github.com/emelis-ptr/software-testing-syncopé
SONARCLOUD BOOKKEEPER	https://sonarcloud.io/project/overview?id=emelis-ptr_software-testing-bookkeeper
SONARCLOUD SYNCOPÉ	https://sonarcloud.io/project/overview?id=emelis-ptr_software-testing-syncopé

Figure

BookKeeper

```
43 @Parameterized.Parameters
44 public static Collection<?> getParameter() throws Exception {
45     BookKeeper.DigestType digestType = BookKeeper.DigestType.CRC32;
46     String PASSWORD = "testPasswd";
47
48     long ledgerID = 100L;
49     int lastEntryId = 10;
50
51     BookId bookieAddress = new BookieSocketAddress("bookie0:3181").toBookId();
52     BookId bookie1 = new BookieSocketAddress("bookie1:3181").toBookId();
53     BookId bookie2 = new BookieSocketAddress("bookie2:3181").toBookId();
54     BookId bookie3 = new BookieSocketAddress("bookie3:3181").toBookId();
55
56     List<BookId> ensembleOfSegment1 = new ArrayList<>();
57     ensembleOfSegment1.add(bookieAddress);
58     ensembleOfSegment1.add(bookie1);
59     ensembleOfSegment1.add(bookie2);
60
61     List<BookId> ensembleOfSegment2 = new ArrayList<>();
62     ensembleOfSegment2.add(bookie3);
63     ensembleOfSegment2.add(bookie1);
64     ensembleOfSegment2.add(bookie2);
65
66     LedgerMetadataBuilder builder = LedgerMetadataBuilder.create()
67         .withId(ledgerID)
68         .withEnsembleSize(3)
69         .withWriteQuorumSize(3)
70         .withAckQuorumSize(2)
71         .withDigestType(digestType.toApiDigestType())
72         .withPassword(PASSWORD.getBytes())
73         .newEnsembleEntry(0, ensembleOfSegment1)
74         .newEnsembleEntry(lastEntryId + 1, ensembleOfSegment2)
75         .withLastEntryId(lastEntryId).withLength(65576)
76         .withClosedState();
77 }
```

Figure 1 - Configurazione dei parametri iniziali per il metodo `areEntriesOfLedgerStoredInTheBookie()` della classe `BookKeeperAdmin`.

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
2	25% 127/505	27% 67/244	80% 67/84

Breakdown by Package

Name	Number of Classes	Line Coverage	Mutation Coverage	Test Strength
org.apache.bookkeeper.bookie	1	85% 80/94	66% 39/59	75% 39/52
org.apache.bookkeeper.client	1	11% 47/411	15% 28/185	88% 28/32

Figure 2 – BookKeeper copertura con mutazione finale

```

244.  @Override
245.  public synchronized int read(ByteBuf dest, long pos, int length) throws IOException {
246.      long prevPos = pos;
247.      while (length > 0) {
248.          // check if it is in the write buffer
249.          if (writeBuffer != null && writeBufferStartPosition.get() <= pos) {
250.              int positionInBuffer = (int) (pos - writeBufferStartPosition.get());
251.              int bytesToCopy = Math.min(writeBuffer.writerIndex() - positionInBuffer, dest.writableBytes());
252.
253.              if (bytesToCopy == 0) {
254.                  throw new IOException("Read past EOF");
255.              }
256.
257.              dest.writeBytes(writeBuffer, positionInBuffer, bytesToCopy);
258.              pos += bytesToCopy;
259.              length -= bytesToCopy;
260.          } else if (writeBuffer == null && writeBufferStartPosition.get() <= pos) {
261.              // here we reach the end
262.              break;
263.              // first check if there is anything we can grab from the readBuffer
264.          } else if (readBufferStartPosition <= pos && pos < readBufferStartPosition + readBuffer.writerIndex()) {
265.              int positionInBuffer = (int) (pos - readBufferStartPosition);
266.              int bytesToCopy = Math.min(readBuffer.writerIndex() - positionInBuffer, dest.writableBytes());
267.              dest.writeBytes(readBuffer, positionInBuffer, bytesToCopy);
268.              pos += bytesToCopy;
269.              length -= bytesToCopy;
270.              // let's read it
271.          } else {
272.              readBufferStartPosition = pos;
273.
274.              int readBytes = fileChannel.read(readBuffer.internalNioBuffer(0, readCapacity),
275.                  readBufferStartPosition);
276.              if (readBytes <= 0) {
277.                  throw new IOException("Reading from filechannel returned a non-positive value. Short read.");
278.              }
279.              readBuffer.writerIndex(readBytes);
280.          }
281.      }
282.      return (int) (pos - prevPos);
283.  }

```

Figure 3 – Jacoco BufferedChannel – Esempio di copertura prima dei nuovi casi di test

```

244  @Override
245  public synchronized int read(ByteBuf dest, long pos, int length) throws IOException {
246      long prevPos = pos;
247      while (length > 0) {
248          // check if it is in the write buffer
249          if (writeBuffer != null && writeBufferStartPosition.get() <= pos) {
250              int positionInBuffer = (int) (pos - writeBufferStartPosition.get());
251              int bytesToCopy = Math.min(writeBuffer.writerIndex() - positionInBuffer, dest.writableBytes());
252
253              if (bytesToCopy == 0) {
254                  throw new IOException("Read past EOF");
255              }
256
257              dest.writeBytes(writeBuffer, positionInBuffer, bytesToCopy);
258              pos += bytesToCopy;
259              length -= bytesToCopy;
260          } else if (writeBuffer == null && writeBufferStartPosition.get() <= pos) {
261              // here we reach the end
262              break;
263              // first check if there is anything we can grab from the readBuffer
264          } else if (readBufferStartPosition <= pos && pos < readBufferStartPosition + readBuffer.writerIndex()) {
265              int positionInBuffer = (int) (pos - readBufferStartPosition);
266              int bytesToCopy = Math.min(readBuffer.writerIndex() - positionInBuffer, dest.writableBytes());
267              dest.writeBytes(readBuffer, positionInBuffer, bytesToCopy);
268              pos += bytesToCopy;
269              length -= bytesToCopy;
270              // let's read it
271          } else {
272              readBufferStartPosition = pos;
273
274              int readBytes = fileChannel.read(readBuffer.internalNioBuffer(0, readCapacity),
275                  readBufferStartPosition);
276              if (readBytes <= 0) {
277                  throw new IOException("Reading from filechannel returned a non-positive value. Short read.");
278              }
279              readBuffer.writerIndex(readBytes);
280          }
281      }
282      return (int) (pos - prevPos);
283  }

```

Figure 4 – PIT BufferedChannel – Esempio di mutazione con mutanti ancora non uccisi prima di inserire nuovi casi di test

```

1359 public static boolean initBookie(ServerConfiguration conf) throws Exception {
1360     /*
1361      * make sure that journalDirs, ledgerDirs and indexDirs are empty
1362      */
1363     File[] journalDirs = conf.getJournalDirs();
1364     if (!validateDirectoriesAreEmpty(journalDirs, "JournalDir")) {
1365         return false;
1366     }
1367     File[] ledgerDirs = conf.getLedgerDirs();
1368     if (!validateDirectoriesAreEmpty(ledgerDirs, "LedgerDir")) {
1369         return false;
1370     }
1371     File[] indexDirs = conf.getIndexDirs();
1372     if (indexDirs != null) {
1373         if (!validateDirectoriesAreEmpty(indexDirs, "IndexDir")) {
1374             return false;
1375         }
1376     }
1377     return runFunctionWithRegistrationManager(conf, rm -> {
1378         try {
1379             /*
1380              * make sure that there is no bookie registered with the same
1381              * bookieid and the cookie for the same bookieid is not existing.
1382              */
1383             BookieId bookieId = BookieImpl.getBookieId(conf);
1384             if (rm.isBookieRegistered(bookieId)) {
1385                 LOG.error("Bookie with bookieid: {} is still registered, "
1386                     + "If this node is running bookie process, try stopping it first.", bookieId);
1387                 return false;
1388             }
1389             try {
1390                 rm.readCookie(bookieId);
1391                 LOG.error("Cookie still exists in the ZK for this bookie: {}, try formatting the bookie", bookieId);
1392                 return false;
1393             } catch (BookieException.CookieNotFoundException nfe) {
1394                 // it is expected for readCookie to fail with
1395                 // BookieException.CookieNotFoundException
1396             }
1397             return true;
1398         } catch (Exception e) {
1399             throw new UncheckedExecutionException(e.getMessage(), e);
1400         }
1401     });
1402 }

```

Figure 5 - PIT BookKeeperAdminI – Esempio di mutazione con mutanti ancora non uccisi prima di inserire nuovi casi di test

```

<method name="write" desc="(Lio/netty/buffer/ByteBuf;)V">
  <du var="this" def="118" use="133" covered="1"/>
  <du var="this" def="118" use="134" target="135" covered="1"/>
  <du var="this" def="118" use="134" target="141" covered="1"/>
  <du var="this" def="118" use="143" covered="1"/>
  <du var="this" def="118" use="135" covered="1"/>
  <du var="this" def="118" use="136" target="137" covered="1"/>
  <du var="this" def="118" use="136" target="141" covered="1"/>
  <du var="this" def="118" use="137" covered="1"/>
  <du var="this" def="118" use="123" covered="1"/>
  <du var="this" def="118" use="124" covered="1"/>
  <du var="this" def="118" use="129" target="130" covered="1"/>
  <du var="this" def="118" use="129" target="132" covered="1"/>
  <du var="this" def="118" use="130" covered="1"/>
  <du var="src" def="118" use="123" covered="1"/>
  <du var="src" def="118" use="124" covered="1"/>
  <du var="this.writeBuffer" def="118" use="123" covered="1"/>
  <du var="this.writeBuffer" def="118" use="124" covered="1"/>
  <du var="this.writeBuffer" def="118" use="129" target="130" covered="1"/>
  <du var="this.writeBuffer" def="118" use="129" target="132" covered="1"/>
  <du var="this.position" def="118" use="133" covered="1"/>
  <du var="this.doRegularFlushes" def="118" use="134" target="135" covered="1"/>
  <du var="this.doRegularFlushes" def="118" use="134" target="141" covered="1"/>
  <du var="this.unpersistedBytes" def="118" use="135" covered="1"/>
  <du var="this.unpersistedBytes" def="118" use="136" target="137" covered="1"/>
  <du var="this.unpersistedBytes" def="118" use="136" target="141" covered="1"/>
  <du var="this.unpersistedBytesBound" def="118" use="136" target="137" covered="1"/>
  <du var="this.unpersistedBytesBound" def="118" use="136" target="141" covered="1"/>
  <du var="copied" def="118" use="122" target="123" covered="1"/>
  <du var="copied" def="118" use="122" target="133" covered="1"/>
  <du var="copied" def="118" use="133" covered="1"/>
  <du var="copied" def="118" use="135" covered="0"/>
  <du var="copied" def="118" use="123" covered="1"/>
  <du var="copied" def="118" use="124" covered="1"/>
  <du var="copied" def="118" use="125" covered="1"/>
  <du var="shouldForceWrite" def="119" use="142" target="143" covered="0"/>
  <du var="shouldForceWrite" def="119" use="142" target="145" covered="1"/>
  <du var="len" def="121" use="122" target="123" covered="1"/>
  <du var="len" def="121" use="122" target="133" covered="1"/>

```

```

<du var="copied" def="125" use="122" target="123" covered="1"/>
<du var="copied" def="125" use="122" target="133" covered="1"/>
<du var="copied" def="125" use="133" covered="1"/>
<du var="copied" def="125" use="135" covered="1"/>
<du var="copied" def="125" use="123" covered="1"/>
<du var="copied" def="125" use="124" covered="1"/>
<du var="copied" def="125" use="125" covered="1"/>
<du var="shouldForceWrite" def="138" use="142" target="143" covered="1"/>
<du var="shouldForceWrite" def="138" use="142" target="145" covered="0"/>
<counter type="DU" missed="3" covered="45"/>
<counter type="METHOD" missed="0" covered="1"/>

```

Figure 6 - BufferedChannel - Ba-dua - Metodo write()

```

▼<method name="read" desc="(Lio/netty/buffer/ByteBuf;JI)I">
  <du var="this" def="246" use="249" target="249" covered="1"/>
  <du var="this" def="246" use="249" target="260" covered="1"/>
  <du var="this" def="246" use="260" target="260" covered="1"/>
  <du var="this" def="246" use="260" target="264" covered="1"/>
  <du var="this" def="246" use="264" target="264" covered="1"/>
  <du var="this" def="246" use="264" target="272" covered="0"/>
  <du var="this" def="246" use="272" covered="1"/>
  <du var="this" def="246" use="274" covered="1"/>
  <du var="this" def="246" use="279" covered="1"/>
  <du var="this" def="246" use="264" target="265" covered="1"/>
  <du var="this" def="246" use="264" target="272" covered="1"/>
  <du var="this" def="246" use="265" covered="1"/>
  <du var="this" def="246" use="266" covered="1"/>
  <du var="this" def="246" use="267" covered="1"/>
  <du var="this" def="246" use="260" target="262" covered="1"/>
  <du var="this" def="246" use="260" target="264" covered="1"/>
  <du var="this" def="246" use="249" target="250" covered="1"/>
  <du var="this" def="246" use="249" target="260" covered="1"/>
  <du var="this" def="246" use="250" covered="1"/>
  <du var="this" def="246" use="251" covered="1"/>
  <du var="this" def="246" use="257" covered="1"/>
  <du var="dest" def="246" use="266" covered="1"/>
  <du var="dest" def="246" use="267" covered="1"/>
  <du var="dest" def="246" use="251" covered="1"/>
  <du var="dest" def="246" use="257" covered="1"/>
  <du var="pos" def="246" use="282" covered="1"/>
  <du var="pos" def="246" use="264" target="264" covered="1"/>
  <du var="pos" def="246" use="264" target="272" covered="0"/>
  <du var="pos" def="246" use="272" covered="1"/>
  <du var="pos" def="246" use="264" target="265" covered="1"/>
  <du var="pos" def="246" use="264" target="272" covered="1"/>
  <du var="pos" def="246" use="265" covered="1"/>
  <du var="pos" def="246" use="268" covered="1"/>
  <du var="pos" def="246" use="260" target="262" covered="1"/>
  <du var="pos" def="246" use="260" target="264" covered="1"/>
  <du var="pos" def="246" use="249" target="250" covered="1"/>
  <du var="pos" def="246" use="249" target="260" covered="1"/>
  <du var="pos" def="246" use="250" covered="1"/>
  <du var="pos" def="246" use="258" covered="1"/>
  <du var="length" def="246" use="247" target="249" covered="1"/>
  <du var="length" def="246" use="247" target="282" covered="1"/>
  <du var="length" def="246" use="269" covered="1"/>
  <du var="length" def="246" use="259" covered="1"/>
  <du var="this.writeBuffer" def="246" use="249" target="249" covered="1"/>
  <du var="this.writeBuffer" def="246" use="249" target="260" covered="1"/>
  <du var="this.writeBuffer" def="246" use="260" target="260" covered="1"/>
  <du var="this.writeBuffer" def="246" use="260" target="264" covered="1"/>
  <du var="this.writeBuffer" def="246" use="251" covered="1"/>
  <du var="this.writeBuffer" def="246" use="257" covered="1"/>
  <du var="this.writeBufferStartPosition" def="246" use="260" target="262" covered="1"/>
  <du var="this.writeBufferStartPosition" def="246" use="260" target="264" covered="1"/>
  <du var="this.writeBufferStartPosition" def="246" use="249" target="250" covered="1"/>
  <du var="this.writeBufferStartPosition" def="246" use="249" target="260" covered="1"/>
  <du var="this.writeBufferStartPosition" def="246" use="250" covered="1"/>
  <du var="this.readBufferStartPosition" def="246" use="264" target="264" covered="1"/>
  <du var="this.readBufferStartPosition" def="246" use="264" target="272" covered="0"/>
  <du var="this.readBufferStartPosition" def="246" use="264" target="265" covered="0"/>
  <du var="this.readBufferStartPosition" def="246" use="264" target="272" covered="1"/>
  <du var="this.readBufferStartPosition" def="246" use="265" covered="0"/>
  <du var="this.readBuffer" def="246" use="274" covered="1"/>
  <du var="this.readBuffer" def="246" use="279" covered="1"/>
  <du var="this.readBuffer" def="246" use="264" target="265" covered="1"/>
  <du var="this.readBuffer" def="246" use="264" target="272" covered="1"/>

```

```

<du var="this.readBuffer" def="246" use="266" covered="1"/>
<du var="this.readBuffer" def="246" use="267" covered="1"/>
<du var="this.fileChannel" def="246" use="274" covered="1"/>
<du var="this.readCapacity" def="246" use="274" covered="1"/>
<du var="prevPos" def="246" use="282" covered="1"/>
<du var="positionInBuffer" def="250" use="257" covered="1"/>
<du var="bytesToCopy" def="251" use="253" target="254" covered="1"/>
<du var="bytesToCopy" def="251" use="253" target="257" covered="1"/>
<du var="bytesToCopy" def="251" use="257" covered="1"/>
<du var="bytesToCopy" def="251" use="258" covered="1"/>
<du var="bytesToCopy" def="251" use="259" covered="1"/>
<du var="pos" def="258" use="282" covered="1"/>
<du var="pos" def="258" use="264" target="264" covered="0"/>
<du var="pos" def="258" use="264" target="272" covered="0"/>
<du var="pos" def="258" use="272" covered="0"/>
<du var="pos" def="258" use="264" target="265" covered="0"/>
<du var="pos" def="258" use="264" target="272" covered="0"/>
<du var="pos" def="258" use="265" covered="0"/>
<du var="pos" def="258" use="268" covered="0"/>
<du var="pos" def="258" use="260" target="262" covered="0"/>
<du var="pos" def="258" use="260" target="264" covered="0"/>
<du var="pos" def="258" use="249" target="250" covered="1"/>
<du var="pos" def="258" use="249" target="260" covered="0"/>
<du var="pos" def="258" use="250" covered="1"/>
<du var="pos" def="258" use="258" covered="0"/>
<du var="length" def="259" use="247" target="249" covered="1"/>
<du var="length" def="259" use="247" target="282" covered="1"/>
<du var="length" def="259" use="269" covered="0"/>
<du var="length" def="259" use="259" covered="0"/>
<du var="pos" def="268" use="282" covered="1"/>
<du var="pos" def="268" use="264" target="264" covered="0"/>
<du var="pos" def="268" use="264" target="272" covered="0"/>
<du var="pos" def="268" use="272" covered="0"/>
<du var="pos" def="268" use="264" target="265" covered="0"/>
<du var="pos" def="268" use="264" target="272" covered="0"/>
<du var="pos" def="268" use="272" covered="0"/>
<du var="pos" def="268" use="264" target="265" covered="0"/>
<du var="pos" def="268" use="264" target="272" covered="0"/>
<du var="pos" def="268" use="265" covered="0"/>
<du var="pos" def="268" use="268" covered="0"/>
<du var="pos" def="268" use="260" target="262" covered="0"/>
<du var="pos" def="268" use="260" target="264" covered="0"/>
<du var="pos" def="268" use="249" target="250" covered="0"/>
<du var="pos" def="268" use="249" target="260" covered="0"/>
<du var="pos" def="268" use="250" covered="0"/>
<du var="pos" def="268" use="258" covered="0"/>
<du var="length" def="269" use="247" target="249" covered="0"/>
<du var="length" def="269" use="247" target="282" covered="1"/>
<du var="length" def="269" use="269" covered="0"/>
<du var="length" def="269" use="259" covered="0"/>
<du var="this.readBufferStartPosition" def="272" use="264" target="264" covered="1"/>
<du var="this.readBufferStartPosition" def="272" use="264" target="272" covered="0"/>
<du var="this.readBufferStartPosition" def="272" use="264" target="265" covered="1"/>
<du var="this.readBufferStartPosition" def="272" use="264" target="272" covered="0"/>
<du var="this.readBufferStartPosition" def="272" use="265" covered="1"/>
<du var="readBytes" def="274" use="276" target="277" covered="0"/>
<du var="readBytes" def="274" use="276" target="279" covered="1"/>
<du var="readBytes" def="274" use="279" covered="1"/>
<counter type="DU" missed="37" covered="81"/>
<counter type="METHOD" missed="0" covered="1"/>

```

Figure 7 - BufferedChannel - Ba-dua- Metodo read()

```

<method name="areEntriesOfLedgerStoredInTheBookie" desc="(JLorg/apache/bookkeeper/net/BookieId;Lorg/apache/bookkeeper/client/api/LedgerMetadata;)Z">
  <du var="bookieAddress" def="1675" use="1681" target="1682" covered="1"/>
  <du var="bookieAddress" def="1675" use="1681" target="1686" covered="1"/>
  <du var="bookieAddress" def="1675" use="1682" target="1683" covered="1"/>
  <du var="bookieAddress" def="1675" use="1682" target="1686" covered="1"/>
  <du var="ledgerMetadata" def="1675" use="1682" target="1683" covered="1"/>
  <du var="ledgerMetadata" def="1675" use="1682" target="1686" covered="1"/>
  <du var="ensemblesOfSegmentsIterator" def="1676" use="1679" target="1680" covered="1"/>
  <du var="ensemblesOfSegmentsIterator" def="1676" use="1679" target="1688" covered="1"/>
  <du var="ensemblesOfSegmentsIterator" def="1676" use="1680" covered="1"/>
  <du var="segmentNo" def="1678" use="1686" covered="1"/>
  <du var="segmentNo" def="1678" use="1682" target="1683" covered="1"/>
  <du var="segmentNo" def="1678" use="1682" target="1686" covered="0"/>
  <du var="ensemble" def="1680" use="1681" target="1682" covered="1"/>
  <du var="ensemble" def="1680" use="1681" target="1686" covered="1"/>
  <du var="segmentNo" def="1686" use="1686" covered="1"/>
  <du var="segmentNo" def="1686" use="1682" target="1683" covered="0"/>
  <du var="segmentNo" def="1686" use="1682" target="1686" covered="1"/>
  <counter type="DU" missed="2" covered="15"/>
  <counter type="METHOD" missed="0" covered="1"/>

```

Figure 8 - BookKeeperAdmin - Ba-dua - Metodo areEntriesOfLedgerStoredInTheBookie

```

▼ <method name="lambda$initBookie$5" desc="(Lorg/apache/bookkeeper/conf/ServerConfiguration;Lorg/apache/bookkeeper/discover/RegistrationManager;)Ljava/lang/Boolean;">
  <du var="rm" def="1386" use="1387" target="1388" covered="0"/>
  <du var="rm" def="1386" use="1387" target="1394" covered="1"/>
  <du var="rm" def="1386" use="1394" covered="1"/>
  <du var="LOG" def="1386" use="1395" covered="1"/>
  <du var="LOG" def="1386" use="1388" covered="0"/>
  <du var="bookieId" def="1386" use="1387" target="1388" covered="0"/>
  <du var="bookieId" def="1386" use="1387" target="1394" covered="1"/>
  <du var="bookieId" def="1386" use="1394" covered="1"/>
  <du var="bookieId" def="1386" use="1395" covered="1"/>
  <du var="bookieId" def="1386" use="1388" covered="0"/>
  <counter type="DU" missed="4" covered="6"/>
  <counter type="METHOD" missed="0" covered="1"/>

```

Figure 9 - BookKeeperAdmin - Ba-dua - Metodo initBookie()

Syncope

```
36 public class AuthDataAccessorMock {
37     @Mock
38     private RealmDAO realmDAO;
39     @Mock
40     private UserDao userDao;
41     @Mock
42     private AnySearchDAO anySearchDAO;
43     @Mock
44     private DelegationDAO delegationDAO;
45
46     /**
47      * Mock Authentication
48      *
49      * @param domain:
50      * @param username:
51      * @param password:
52      * @return Authentication
53      */
54     protected Authentication authentication(String domain, String username, String password) {
55         Authentication auth = Mockito.mock(Authentication.class);
56         Mockito.when(auth.getName()).thenReturn(username);
57         Mockito.when(auth.getCredentials()).thenReturn(password);
58         Mockito.when(auth.getDetails()).thenReturn(new SyncopeAuthenticationDetails(domain, null));
59         return auth;
60     }
61
62     /**
63      * Mock ConfParamOps
64      *
65      * @param username:
66      * @return ConfParamOps
67      */
68     protected ConfParamOps confParam(String username) {
69         ConfParamOps confParamOps = Mockito.mock(ConfParamOps.class);
70         Mockito.when(confParamOps.get(anyString(), eq("authentication.attributes"), any(), any())).thenReturn(new String[]{username});
71         Mockito.when(confParamOps.get(any(), eq("authentication.statuses"), any(), any())).thenReturn(new String[]{"ACTIVE", "SUSPENDED", null});
72         Mockito.when(confParamOps.get(any(), eq("log.lastlogindate"), any(), any())).thenReturn(true);
73         return confParamOps;
74     }
75
76     /**
77      * Mock RealmDAO
78      *
79      * @return RealmDAO
80      */
81     protected RealmDAO mockRealmDAO() {
82         realmDAO = Mockito.mock(RealmDAO.class);
83         Realm mockRealm = Mockito.mock(Realm.class);
84         Mockito.when(realmDAO.findAncestors(any())).thenReturn(Collections.singletonList(mockRealm));
85         return realmDAO;
86     }
87
88     /**
89      * Mock UserDao
90      *
91      * @param user:
92      * @return UserDao
93      */
94     protected UserDao mockUserDAO(User user) {
95         userDao = Mockito.mock(UserDAO.class);
96         Mockito.when(userDAO.findByUsername(any())).thenReturn(user);
97         return userDao;
98     }
99
100     /**
101      * Mock UserDao
102      *
103      * @param user:
104      * @return UserDao
105      */
106     protected UserDao mockUserDAO(User user, boolean isFoundUser) {
107         userDao = Mockito.mock(UserDAO.class);
108         if (isFoundUser) {
109             Mockito.when(userDAO.findByUsername(any())).thenReturn(user);
110         } else {
111             Mockito.when(userDAO.findByUsername(any())).thenThrow(new UsernameNotFoundException("Could not find any user with username " + user.getUsername()));
112         }
113         return userDao;
114     }
115 }
```



```

116     /**
117     * User
118     *
119     * @param username:
120     * @param password:
121     * @return User
122     */
123     protected User getUser(String username, String password) {
124         User user = new MyUser();
125         user.setUsername(username);
126         try {
127             user.setPassword(Encryptor.getInstance().encode(password, CipherAlgorithm.AES));
128         } catch (UnsupportedEncodingException | NoSuchAlgorithmException | NoSuchPaddingException |
129             InvalidKeyException | IllegalBlockSizeException | BadPaddingException e) {
130             throw new RuntimeException(e);
131         }
132         return user;
133     }
134
135     /**
136     * Mock SecurityProperties
137     *
138     * @param anonymousUser:
139     * @param adminUser:
140     * @return SecurityProperties
141     */
142     protected static SecurityProperties mockSecurityProperties(String anonymousUser, String adminUser) {
143         SecurityProperties securityProperties = mock(SecurityProperties.class);
144         Mockito.when(securityProperties.getAnonymousUser()).thenReturn(anonymousUser);
145         Mockito.when(securityProperties.getAdminUser()).thenReturn(adminUser);
146         return securityProperties;
147     }
148
149     /**
150     * Mock AnySearchDAO
151     *
152     * @param user:
153     * @param numUsers:
154     * @return AnySearchDAO
155     */
156     protected AnySearchDAO mockAnySearchDAO(User user, int numUsers) {
157         List<Any?> userList = new ArrayList<>();
158
159         if (numUsers == 1) {
160             userList.add(user);
161         } else {
162             userList.add(user);
163             for (int i = 1; i < numUsers; i++) {
164                 User newUser = new MyUser();
165                 newUser.setUsername(user.getUsername() + i);
166                 newUser.setPassword(user.getPassword() + i);
167                 userList.add(newUser);
168             }
169         }
170
171         anySearchDAO = mock(AnySearchDAO.class);
172         Mockito.when(anySearchDAO.search(any(SearchCond.class), any(AnyTypeKind.class))).thenReturn(userList);
173         return anySearchDAO;
174     }
175
176     /**
177     * Mock DelegationDAO
178     *
179     * @param user:
180     * @param delegationKey:
181     * @param findDelegation:
182     * @return DelegationDAO
183     */
184     public DelegationDAO mockDelegationDAO(User user, String delegationKey, boolean findDelegation, boolean isEmptyRole) {
185         MyDelegation myDelegation = new MyDelegation(user);
186         delegationDAO = mock(DelegationDAO.class);
187         if (findDelegation) {
188             Mockito.when(delegationDAO.find(delegationKey)).thenReturn(myDelegation);
189         } else {
190             Mockito.when(delegationDAO.find(delegationKey)).thenThrow(new UsernameNotFoundException("Could not find delegation " + delegationKey));
191         }
192
193         MyDelegation delegationMock = Mockito.spy(myDelegation);
194         Set<Role> roleSet = new HashSet<>();
195         Role role = mock(Role.class);
196         if (!isEmptyRole) {
197             roleSet.add(role);
198         }
199         Mockito.doReturn(roleSet).when(delegationMock).getRoles();
200         Mockito.when(delegationDAO.find(delegationKey)).thenReturn(delegationMock);
201         return delegationDAO;
202     }
203
204 }

```

Figure 10 – AuthDataAccessor – Classe Mock

Pit Test Coverage Report

Package Summary

org.apache.syncope.core.spring.security

Number of Classes	Line Coverage	Mutation Coverage	Test Strength
2	51% 156/304	35% 47/133	64% 47/73

Breakdown by Class

Name	Line Coverage	Mutation Coverage	Test Strength
AuthDataAccessor.java	45% 103/230	28% 27/97	61% 27/44
Encryptor.java	72% 53/74	56% 20/36	69% 20/29

Figure 11 – Syncope copertura con mutazione finale

```
@Transactional(noRollbackFor = DisabledException.class)
public Triple<User, Boolean, String> authenticate(final String domain, final Authentication authentication) {
    User user = null;

    String[] authAttrValues = confParamOps.get(
        domain, "authentication.attributes", new String[] { "username" }, String[].class);
    for (int i = 0; user == null && i < authAttrValues.length; i++) {
        if ("username".equals(authAttrValues[i])) {
            user = userDao.findByUsername(authentication.getName());
        } else {
            AttrCond attrCond = new AttrCond(AttrCond.Type.EQ);
            attrCond.setSchema(authAttrValues[i]);
            attrCond.setExpression(authentication.getName());
            try {
                List<User> users = anySearchDAO.search(SearchCond.getLeaf(attrCond), AnyTypeKind.USER);
                if (users.size() == 1) {
                    user = users.get(0);
                } else {
                    LOG.warn("Search condition {} does not uniquely match a user", attrCond);
                }
            } catch (IllegalArgumentException e) {
                LOG.error("While searching user for authentication via {}", attrCond, e);
            }
        }
    }

    Boolean authenticated = null;
    String delegationKey = null;
    if (user != null) {
        authenticated = false;

        if (user.isSuspended() != null && user.isSuspended()) {
            throw new DisabledException("User " + user.getUsername() + " is suspended");
        }

        String[] authStatuses = confParamOps.get(
            domain, "authentication.statuses", new String[] {}, String[].class);
        if (!ArrayUtils.contains(authStatuses, user.getStatus())) {
            throw new DisabledException("User " + user.getUsername() + " not allowed to authenticate");
        }

        boolean userModified = false;
        authenticated = authenticate(user, authentication.getCredentials().toString());
        if (authenticated) {
            delegationKey = getDelegationKey(
                SyncopeAuthenticationDetails.class.cast(authentication.getDetails()), user.getKey());

            if (confParamOps.get(domain, "log.lastlogindate", true, Boolean.class)) {
                user.setLastLoginDate(OffsetDateTime.now());
                userModified = true;
            }

            if (user.getFailedLogins() != 0) {
                user.setFailedLogins(0);
                userModified = true;
            }
        } else {
            user.setFailedLogins(user.getFailedLogins() + 1);
            userModified = true;
        }

        if (userModified) {
            userDao.save(user);
        }
    }

    return Triple.of(user, authenticated, delegationKey);
}
```

Figure 12 – Jacoco AuthDataAccessor – Esempio di copertura prima di aggiungere nuovi casi di test.

```

@Transactional
public Set<SyncopGrantedAuthority> getAuthorities(final String username, final String delegationKey) {
    Set<SyncopGrantedAuthority> authorities;

    if (securityProperties.getAnonymousUser().equals(username)) {
        authorities = ANONYMOUS_AUTHORITIES;
    } else if (securityProperties.getAdminUser().equals(username)) {
        authorities = getAdminAuthorities();
    } else if (delegationKey != null) {
        Delegation delegation = Optional.ofNullable(delegationDAO.find(delegationKey)).
            orElseThrow(() -> new UsernameNotFoundException(
                "Could not find delegation " + delegationKey));

        authorities = delegation.getRoles().isEmpty()
            ? getUserAuthorities(delegation.getDelegating())
            : getDelegatedAuthorities(delegation);
    } else {
        User user = Optional.ofNullable(userDAO.findByUsername(username)).
            orElseThrow(() -> new UsernameNotFoundException(
                "Could not find any user with username " + username));

        authorities = getUserAuthorities(user);
    }

    return authorities;
}

```

Figure 13 – Jacoco AuthDataAccessor – Esempio di copertura prima di aggiungere nuovi casi di test.