UNIONE EUROPEA

Ministero dello Sviluppo Economico

Regione Puglia
Dipartimento Sviluppo Economico, Innovazione, Istruzione, Formazione e Lavoro

PUGLIA FESR·FSE 2014/2020
Il futuro alla portata di tutti

BANDO INNOLABS

"Sostegno alla creazione di soluzioni innovative finalizzate a specifici problemi di rilevanza sociale"

Y95B457 Progetto

easyPAL

Deliverable D.D2 – Report: Descrizione dei risultati del testing della piattaforma EasyPAL

Data 02/12/2019

Versione V1.0

NOME DEL DOCUMENTO

Deliverable D.D2 – Report: Descrizione dei risultati del testing della piattaforma EasyPAL

INFORMAZIONI GENERALI

Nome progetto	EasyPAL “Ecosistemi e Servizi Digitali in Cloud per i Cittadini e la PA Locale”
Ambito	BANDO INNOLABS “Sostegno alla creazione di soluzioni innovative finalizzate a specifici problemi di rilevanza sociale”
Riservatezza	Riservatezza ai sensi dell'art. 14 dell’atto di costituzione dell’ATS denominata “Easy PAL” registrata a Lecce in data 01/06/2018 al n. 5694/1T da Notaio Pellegrino.

RESPONSABILITÀ

Funzione	Nome	Data
Redatto da	Unisalento	02/12/2019
Contributi di	Unisalento, Servizi Locali	
Controllato e approvato da	Unisalento	02/12/2019

INDICE

1. Premessa

Il presente deliverable riporta i risultati della fase di testing svolta durante ed a valle dello sviluppo della piattaforma EasyPAL. In particolare, il deliverable è focalizzato sulla parte di testing riguardante la verifica dei vari moduli software che ne costituiscono l'architettura. Infatti, riportando quanto definito in *Deliverable D.D1 – Report: Piano della sperimentazione*, il testing della piattaforma si è articolato in due fasi distinte, la prima appunto oggetto del presente documento, la seconda relativa alla verifica delle funzionalità dei prototipi sviluppati dal punto di vista della User Experience, la quale sarà oggetto del *Deliverable D.D3 D.D4 D.D5 – Descrizione dei risultati del testing dei servizi applicativi*.

Le tipologie di test applicate alla piattaforma sono state: Testing Strutturale, Unit Testing, Penetration Test.

Il *Testing Strutturale*, o *White Box*, verte alla verifica del corretto funzionamento dei macro componenti del sistema e si basa sull'implementazione di una serie di test tale che sia garantito che ogni istruzione all'interno del software sia eseguita e testata almeno una volta (criterio di copertura delle istruzioni o statement coverage).

Lo Unit Testing, o Black Box Testing, è un testing a granularità fine, il cui scopo è verificare i dettagli relativi alle singole unità di codice, come ad esempio le Classi.

In tal senso, anche grazie agli strumenti messi a disposizione dall'IDE Visual Studio 2017 Enterprise Edition utilizzato, la strategia è stata quella di eseguire il *Testing Strutturale* congiuntamente allo *Unit Testing*, verificando, cioè, che lo statement coverage dei test di unità fosse adeguato e soddisfacente.

La fase di *Penetration Test* ha avuto l'obiettivo di verificare la robustezza delle policy di sicurezza implementate per i servizi RESTful erogati dalla piattaforma ed è stato eseguito con strumenti automatici, in particolare con il tool Vooki.

2. Unit Test e Test Strutturale

2.1 Modalità di testing

La fase di testing strutturale è stata condotta unitamente a quella di unit testing, basandosi sugli strumenti messi a disposizione dall'IDE Visual Studio 2017 Enterprise Edition, utilizzato nello sviluppo della piattaforma EasyPAL. Tali strumenti, infatti, consentono non solo di eseguire gli unit test durante la fase di *build&run* della piattaforma, ma anche di dare una precisa evidenza di quella che è la copertura dei blocchi coinvolta nel testing. Per maggiori dettagli, si faccia riferimento al *Deliverable D.D1 – Report: Piano della sperimentazione*.

L'analisi dei moduli da testare si è basata sulla struttura delle componenti che strutturano la piattaforma, mostrata in Figura 1:

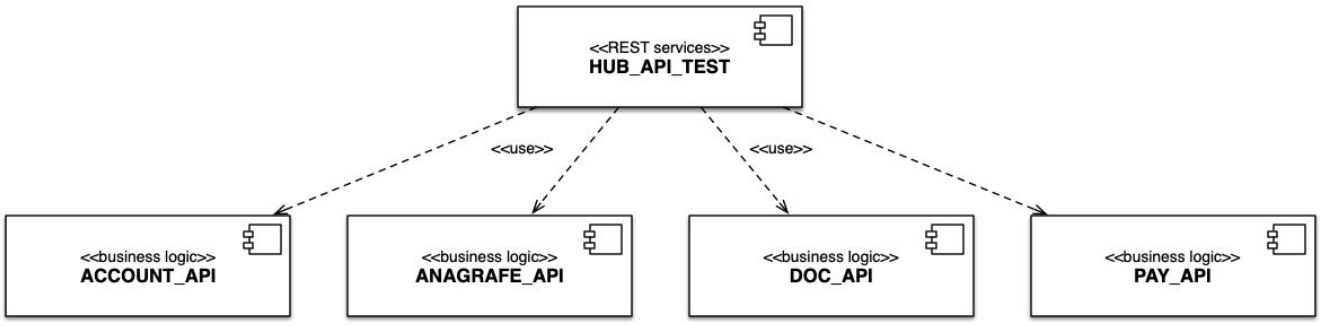


Figura 1 - Diagramma dei componenti relativo ai web services RESTful

Come si evince, i servizi esposti dalla piattaforma e consumati dai client mobile e web sono implementati nel modulo *HUB_API_TEST*. Il modulo è composto da quattro classi Controller principali che implementano i metodi di gestione dei servizi REST:

- AccountController
- ANPRController
- DocController
- PayController

La composizione del modulo è mostrata in Figura 2:

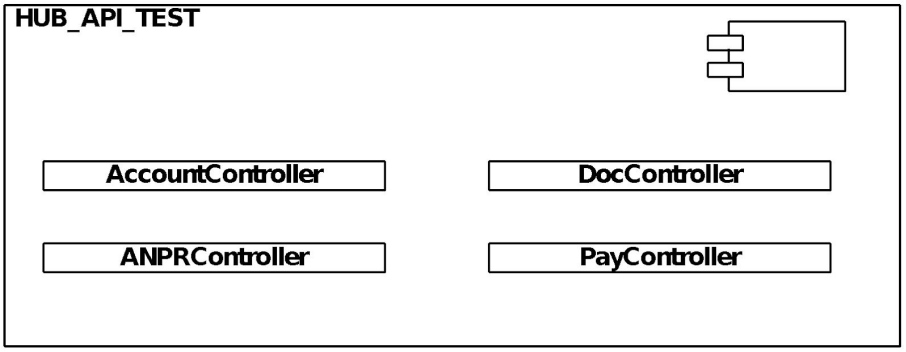


Figura 2 - Diagramma delle classi in the large del modulo HUB_API_TEST

Come si evince dal nome associato ad ognuno, i Controller riprendono le quattro funzionalità principali della piattaforma, cioè rispettivamente: gestione dell'autenticazione, anagrafica nazionale, documentazione comunale e meccanismi di pagamento alla Pubblica Amministrazione.

La business logic sulla quale si basa il funzionamento del modulo è implementata in quattro moduli dedicati, come visualizzato in Figura 1.

Per ognuno di essi, in Figura 3, Figura 4, Figura 5 e Figura 6 vengono rappresentati i diagrammi delle classi in-the-large.

Come evidenziato dalle suddette figure, i quattro moduli condividono la stessa struttura:

- Un'interfaccia espone i metodi da implementare, specifici per il modulo;
- Sono presenti due implementazioni dell'interfaccia, rispettivamente per l'esecuzione in produzione e per quella in test. In quest'ultima, viene completamente disaccoppiato lo strato di persistenza ed i dati sono simulati;
- Un Controller utilizza l'implementazione dell'interfaccia ed espone le funzionalità del modulo.

Ai fini del presente Deliverable, l'architettura è stata testata basandosi sull'implementazione mock delle interfacce dei Controller di business logic. Questa strategia ha consentito di eseguire i test in maniera agnostica rispetto ad uno strato di persistenza reale, come da best practice.

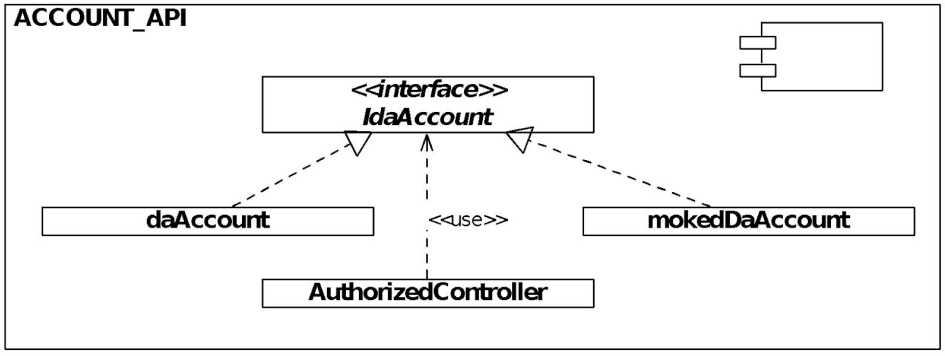


Figura 3 - Diagramma delle classi in-the-large del modulo ACCOUNT_API

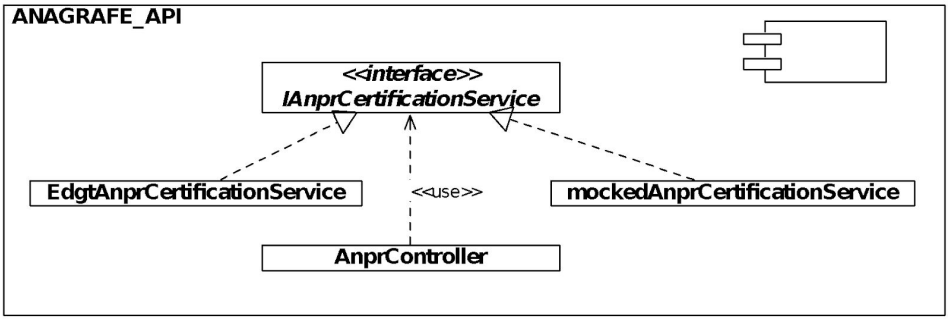


Figura 4 - Diagramma delle classi in the large del modulo ANAGRAFE_API

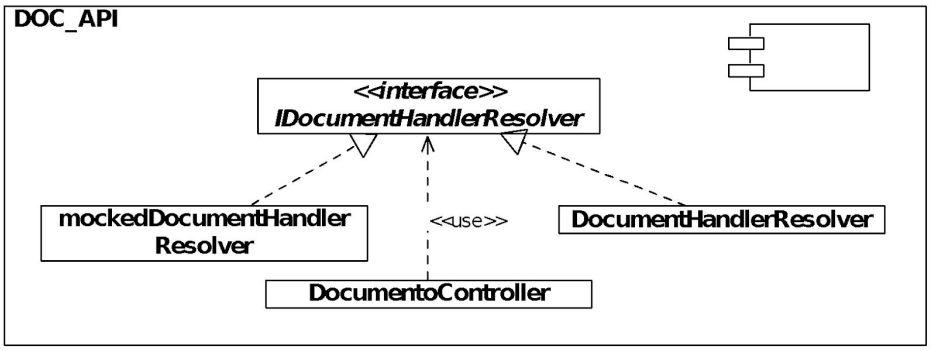


Figura 5 - Diagramma delle classi in the large del modulo DOC_API

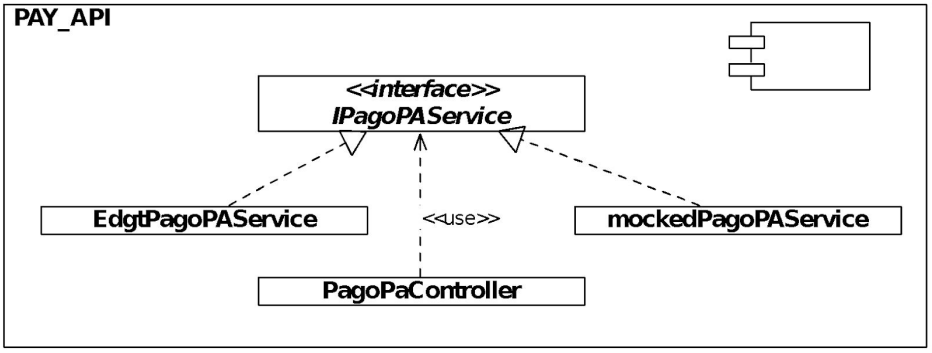


Figura 6 - Diagramma delle classi in the large del modulo PAY_API

2.2 Risultati del test

Come anticipato nel precedente paragrafo, l'implementazione sia dei test strutturali sia degli unit test si è basata interamente sugli strumenti forniti dall'IDE Visual Studio 2017 Enterprise Edition.

In particolare, il testing ha riguardato i moduli software mostrati in Figura 1, nel dettaglio:

- HUB_API_TEST
- ACCOUNT_API
- ANAGRAFE_API
- DOC_API
- PAY_API

2.2.1 Risultati del test strutturali

I risultati dei test strutturali per i 5 moduli elencati nel precedente paragrafo, sono mostrati rispettivamente in Figura 7, Figura 8, Figura 9, Figura 10, Figura 11.

Risultati code coverage

slongodev_SER-PC01 2019-11-26 17_30_59.c

Gerarchia	Blocchi non analizzati	% blocchi non analizzati	Blocchi analizzati	% blocchi analizzati
EasyPAL_HUB_API.Controllers	347	22,14%	1220	77,86%
PayController	43	21,61%	156	78,39%
getUrlEseguiPagamento(EasyPAL_PresentationDLL.API.PagoPA.Param...	14	21,88%	50	78,13%
getEasyPAL_PAY_API_address()	4	23,53%	13	76,47%
TrackException(System.Exception)	3	100,00%	0	0,00%
PayController()	0	0,00%	2	100,00%
Log(string)	0	0,00%	12	100,00%
EsitoPagamento(string, string)	22	21,78%	79	78,22%
DOCController	34	12,36%	241	87,64%
resetCacheConfig()	0	0,00%	16	100,00%
getPagamentiDovuti(Newtonsoft.Json.Linq.JObject)	10	12,66%	69	87,34%
getEasyPAL_DOC_API_address()	4	23,53%	13	76,47%
getAllDocumenti(Newtonsoft.Json.Linq.JObject)	14	19,18%	59	80,82%
downloadFile(Newtonsoft.Json.Linq.JObject)	6	8,22%	67	91,78%
TrackException(System.Exception)	0	0,00%	3	100,00%
Log(string)	0	0,00%	12	100,00%
DOCController()	0	0,00%	2	100,00%
AccountController	133	23,62%	430	76,38%
getEasyPAL_ACCOUNT_API_address()	4	23,53%	13	76,47%
TrackException(System.Exception)	0	0,00%	3	100,00%
PrenotaAutorizzazione(long, string)	4	11,43%	31	88,57%
Logout(Newtonsoft.Json.Linq.JObject)	5	9,62%	47	90,38%
LoginSPID(string, string, string)	3	9,68%	28	90,32%
LoginOperatore(Newtonsoft.Json.Linq.JObject)	0	0,00%	134	100,00%
Log(string)	0	0,00%	12	100,00%
EsitoLoginSPID(string, string, string, string, string, string, string)	95	61,29%	60	38,71%
EsitoLoginSPID(Newtonsoft.Json.Linq.JObject)	2	10,00%	18	90,00%
ControllaTokenInterno(string, string)	14	17,95%	64	82,05%
ControllaToken(Newtonsoft.Json.Linq.JObject)	6	25,00%	18	75,00%
AccountController()	0	0,00%	2	100,00%
ANPRController	137	25,85%	393	74,15%
myTrackException(System.Exception)	0	0,00%	3	100,00%
myTrace(string)	0	0,00%	12	100,00%
getSchedaSoggetto(Newtonsoft.Json.Linq.JObject)	24	23,30%	79	76,70%
getElencoComuniANPR(Newtonsoft.Json.Linq.JObject)	21	27,27%	56	72,73%
getEasyPAL_DOC_API_address()	17	100,00%	0	0,00%
getEasyPAL_ANAGRAFE_API_address()	4	25,00%	12	75,00%
getComponentiFamigliaConvivenza(Newtonsoft.Json.Linq.JObject)	22	20,56%	85	79,44%
getCertificazioneElencoTipiCertificato(Newtonsoft.Json.Linq.JObject)	21	26,58%	58	73,42%
getANPRCertificatiEmessi(Newtonsoft.Json.Linq.JObject)	14	24,56%	43	75,44%
downloadCertificatoANPR(Newtonsoft.Json.Linq.JObject)	14	24,56%	43	75,44%
ANPRController()	0	0,00%	2	100,00%

Figura 7 - Risultati del test strutturale per il modulo HUB_API_TEST

Risultati code coverage

slongodev_SER-PC01 2019-12-10 18_53_52.c

Gerarchia	Blocchi non analizzati	% blocchi non analizzati	Blocchi analizzati	% blocchi analizzati
EasyPAL_ACCOUNT_API.Controllers	75	21,87%	268	78,13%
AuthorizeController	59	18,04%	268	81,96%
AuthorizeController()	0	0,00%	2	100,00%
AuthorizeController(EasyPAL_ConfDa...	0	0,00%	2	100,00%
ConfermaAutorizzazione(long, string)	2	22,22%	7	77,78%
ControllaToken(Newtonsoft.Json.Linq...	10	20,00%	40	80,00%
GenerateToken(string, string)	2	6,67%	28	93,33%
GetHashedString(string)	2	12,50%	14	87,50%
Login()	32	22,22%	112	77,78%
Logout()	2	12,50%	14	87,50%
Post()	0	0,00%	4	100,00%
PrenotaAccesso()	4	18,18%	18	81,82%
PrenotaAutorizzazione()	2	11,76%	15	88,24%
myTrace(string)	0	0,00%	12	100,00%
myTrackException(System.Exception)	3	100,00%	0	0,00%

Figura 8 - Risultati del test strutturale per il modulo ACCOUNT_API

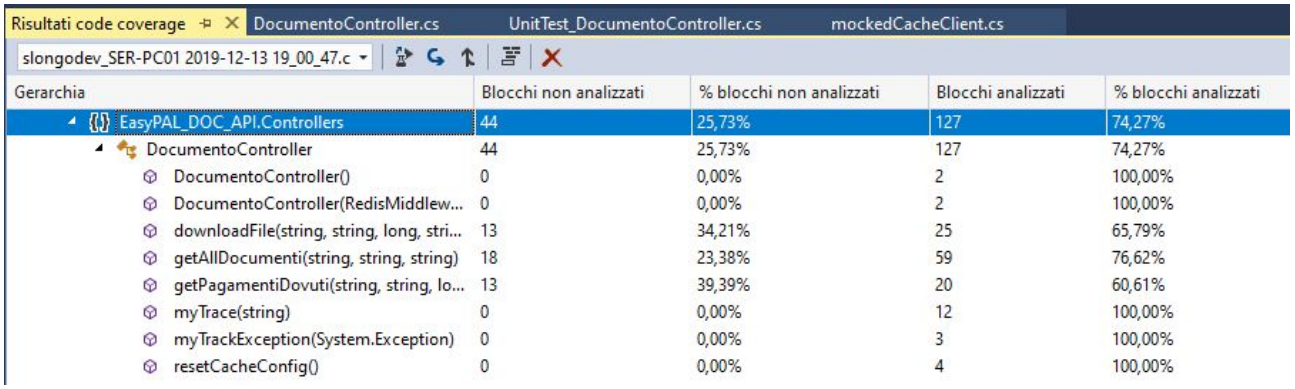
Risultati code coverage

mockedAnprCertificationService.cs UnitTest_AnprController.cs UnitTest_DocumentoController.cs

slongodev_SER-PC01 2019-12-17 13_33_37.c

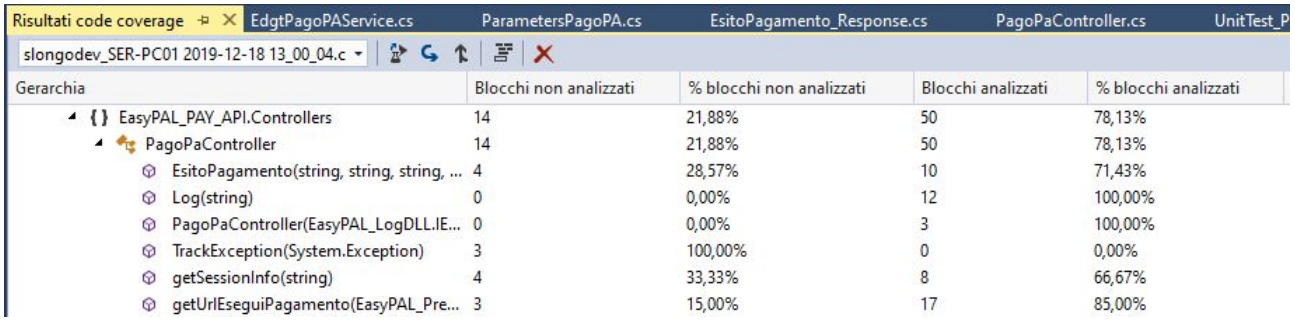
Gerarchia	Blocchi non analizzati	% blocchi non analizzati	Blocchi analizzati	% blocchi analizzati
EasyPAL_ANAGRAFE_API.Controllers	94	30,13%	218	69,87%
AnprController	94	30,13%	218	69,87%
AnprController(EasyPAL_LogDLL.IEPL...	0	0,00%	3	100,00%
downloadCertificatoANPR(Newtonso...	11	36,67%	19	63,33%
getANPRCertificatiEmessi(Newtonsof...	11	20,75%	42	79,25%
getCertificatoANPRTimbratoByQrCod...	10	47,62%	11	52,38%
getCertificazioneElencoMotiviEsenzio...	11	50,00%	11	50,00%
getCertificazioneElencoTipiCertificato...	11	50,00%	11	50,00%
getComponentiFamigliaConvivenza(...	11	42,31%	15	57,69%
getElencoComuniANPR(Newtonsoft....	11	50,00%	11	50,00%
getNewANPRCertificato(Newtonsoft....	7	9,46%	67	90,54%
getSchedaSoggetto(Newtonsoft.Json....	11	45,83%	13	54,17%
myTrace(string)	0	0,00%	12	100,00%
myTrackException(System.Exception)	0	0,00%	3	100,00%

Figura 9 - Risultati del test strutturale per il modulo ANAGRAFE_API



Gerarchia	Blocchi non analizzati	% blocchi non analizzati	Blocchi analizzati	% blocchi analizzati
EasyPAL_DOC_API.Controllers	44	25,73%	127	74,27%
DocumentoController	44	25,73%	127	74,27%
DocumentoController()	0	0,00%	2	100,00%
DocumentoController(RedisMiddlew...	0	0,00%	2	100,00%
downloadFile(string, string, long, stri...	13	34,21%	25	65,79%
getAllDocumenti(string, string, string)	18	23,38%	59	76,62%
getPagamentiDovuti(string, string, lo...	13	39,39%	20	60,61%
myTrace(string)	0	0,00%	12	100,00%
myTrackException(System.Exception)	0	0,00%	3	100,00%
resetCacheConfig()	0	0,00%	4	100,00%

Figura 10 - Risultati del test strutturale per il modulo DOC_API



Gerarchia	Blocchi non analizzati	% blocchi non analizzati	Blocchi analizzati	% blocchi analizzati
EasyPAL_PAY_API.Controllers	14	21,88%	50	78,13%
PagoPaController	14	21,88%	50	78,13%
EsitoPagamento(string, string, string, ...	4	28,57%	10	71,43%
Log(string)	0	0,00%	12	100,00%
PagoPaController(EasyPAL_LogDLL.IE...	0	0,00%	3	100,00%
TrackException(System.Exception)	3	100,00%	0	0,00%
getSessionInfo(string)	4	33,33%	8	66,67%
getUrlEseguiPagamento(EasyPAL_Pre...	3	15,00%	17	85,00%

Figura 11 - Risultati del test strutturale per il modulo PAY_API

I risultati mostrano un code coverage generale di grado soddisfacente. In particolare, di seguito viene elencato il block coverage medio per ognuno dei componenti software considerati nel test [1]:

Tabella 1 - Block coverage dei moduli testati

MODULO	CLASSI	BLOCK COVERAGE
HUB_API_TEST		77,86%
	AccountController	76,38%
	ANPRController	74,15%
	DOCController	87,64%
	PayController	78,39%
ACCOUNT_API		78,13%
	AuthorizeController	81,96%
ANAGRAFE_API		69,87%
	AnprController	69,87%
DOC_API		74,27%
	DocumentController	74,27%
PAY_API		78,13%
	PagoPaController	78,13%

2.2.2 Risultati degli unit test

In Figura 12 e Figura 13 viene evidenziata l'esecuzione automatica degli unit test implementati per i moduli di interesse. Il framework utilizzato per l'implementazione degli unit test è NUnit, valido per il testing in ambiente .Net.

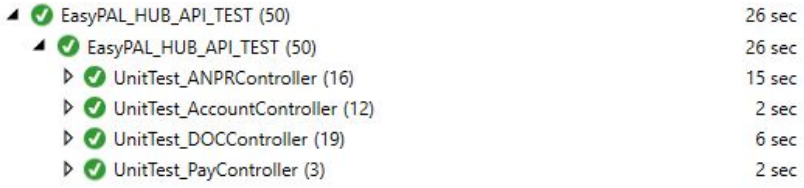


Figura 12 - Unit Test per il modulo HUB_API_TEST

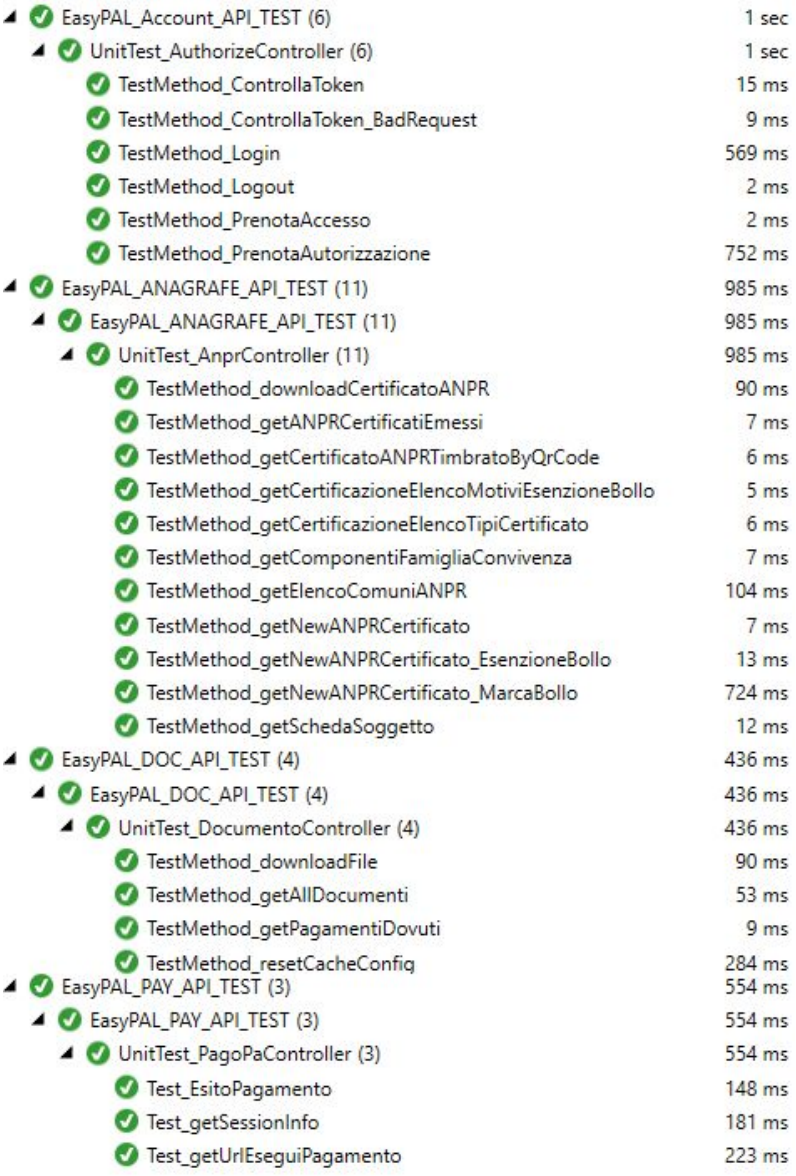


Figura 13 - Unit Test per i moduli ACCOUNT_API, ANAGRAFE_API, DOC_API, PAY_API

3. Penetration Test

3.1 Modalità di testing

Per i penetration test della piattaforma EasyPAL è stato utilizzato il tool Vooki, fornito da VegaBird Technologies. Vooki è conforme agli standard OWASP TOP 10, la serie di direttive utili all'implementazione di valide application security policy stilata dall'organizzazione di riferimento, Open Web Application Security Project (OWASP).

I test condotti sono di tipo esterno, avendo come obiettivo le risorse presenti nell'area DMZ, in particolare le API esposte dalla piattaforma. Per ulteriori dettagli, si faccia riferimento al *Deliverable D.D1 – Report: Piano della sperimentazione*.

3.2 Endpoint

Le API esposte dalla piattaforma sono implementate nel modulo *HUB_API_TEST*. La collezione di API è data da:

1. LoginOperatore
2. getNewANPRCertificato
3. getANPRCertificatiEmessi
4. downloadCertificatoANPR
5. getAllDocumenti
6. getPagamentiDovuti
7. downloadFile
8. ControllaToken
9. Logout
10. getElencoComuniANPR
11. getSchedaSoggetto

I dettagli delle chiamate sono esposti nelle seguenti figure 14 - 24:

POST HUB_API - LoginOperatore

https://sperimentazione.easypal.it/EasyPAL_HUB_API/api/Account/LoginOperatore

HEADERS

Content-Type

application/json

BODY raw


```
{
    "username": "pippo",
    "password": "pluto",
    "client_id": "4439AB6D-7006-4A03-A51C-AA0925C47153",
    "scope": "",
    "session_id": "efxu3svpegxzpfdqs1auflr3"
}
```


Figura 14 - Dettagli API LoginOperatore

POST HUB_API - getNewANPRCertificato - Operatore senza i parametri della marca da bollo

https://sperimentazione.easypal.it/EasyPAL_HUB_API/api/Anpr/getNewANPRCertificato

HEADERS

Content-Type

application/json

BODY raw


```
{
    "codiceComune": "A225",
    "token" : "Nzk3M0g2SFMvVXRoWDdmTldSWjk3a0dieGVFb1lRWnVUUkltMFpqMGw0Yz06cGlwcG86NjM3MDk2OTEzODMyNTI0MD
    "CodiceFiscaleRichiedente": "rssmra80a01E506t",
    "ListaTipiCertificati": "8",
    "CodiceFiscaleIntestatario": "MRRGLM80C13A225N",
    "tipoLogin": "operatore"
}
```


Figura 15 - Dettagli API getNewANPRCertificato

POST HUB_API - getANPRCertificatiEmessi

https://sperimentazione.easypal.it/EasyPAL_HUB_API/api/Anpr/getANPRCertificatiEmessi

HEADERS

Content-Type

application/json

BODY raw


```
{
    "codiceComune": "A225",
    "token": "Nzk3M0g2SFMvVXRoWDdmTldSWjk3a0dieGVFb1lRWnVUUkltMFpqMGw0Yz06cGlwcG86NjM3MDk2OTEzODMyNTI0MDkw",
    "CodiceFiscaleRichiedente": "rssmra80a01E506t",
    "CodiceFiscaleIntestatario": "MRRGLM80C13A225N",
    "DataOraRichiestaDal": "",
    "DataOraRichiestaAl": "",
    "Diritti": "",
    "Bollo": ""
}
```


Figura 16 - Dettagli API getANPRCertificatiEmessi

POST HUB_API - downloadCertificatoANPR

https://sperimentazione.easypal.it/EasyPAL_HUB_API/api/Anpr/downloadCertificatoANPR

HEADERS

Content-Type

application/json

BODY raw


```
{
    "codiceComune": "A225",
    "token": "Nzk3M0g2SFMvVXRoWDdmTldSWjk3a0dieGVFb1lRWnVUUkltMFpqMGw0Yz06cGlwcG86NjM3MDk2OTEzODMyNTI0MDkw",
    "CodiceFiscaleRichiedente": "rssmra80a01E506t",
    "idOperazioneApplicativo": "1099478",
    "ProtocolloANPR": "146306474",
    "flagFirmatoOTimbrato": 1
}
```


Figura 17 - Dettagli API getANPRCertificatiEmessi

POST HUB_API - getAllDocumenti

https://sperimentazione.easypal.it/EasyPAL_HUB_API/api/Doc/getAllDocumenti

HEADERS

Content-Type

application/json

BODY raw


```
{
    "cf": "rssmra80a01E506t",
    "token": "QlhubTRJcUtHNWdTM2cxd3ZtYmsrRlM4QUdsMUJXOGVJNnltY3lxcEZXMD06cGlwcG86NjM3MTAwNDQyOTg1MTkzMTI1",
    "cfInVisualizzazione": "TRSRRT63P26C034K"
}
```


Figura 18 - Dettagli API getAllDocumenti

POST HUB_API - getPagamentiDovuti

https://sperimentazione.easypal.it/EasyPAL_HUB_API/api/Doc/getPagamentiDovuti

HEADERS

Content-Type

application/json

BODY raw


```
{
    "tipoDocumento": "VOTIVA - AVVISO DI PAGAMENTO",
    "pkDocumento": "2",
    "pkConfigurazione": 37,
    "token": "QlhubTRJcUtHNWdTM2cxd3ZtYmsrRlM4QUdsMUJXOGVJNnltY3lxcEZXMD06cGlwcG86NjM3MTAwNDQyOTg1MTkzMTI1",
    "cfToken": "rssmra80a01E506t"
}
```


Figura 19 - Dettagli API getPagamentiDovuti

POST HUB_API - downloadFile

https://sperimentazione.easypal.it/EasyPAL_HUB_API/api/Doc/downloadFile

HEADERS

Content-Type

application/json

BODY raw


```
{
    "tipoDocumento": "VOTIVA - AVVISO DI PAGAMENTO",
    "pkDocumento": "2",
    "pkConfigurazione": 37,
    "nomefile": "Fattura N.2.pdf",
    "token": "QlhubTRJcUtHNWdTM2cxd3ZtYmsrRlM4QUdsMUJXOGVJNnltY3lxcEZXMD06cGlwcG86NjM3MTAwNDQyOTg1MTkzMTI1",
    "cfToken": "rssmra80a01E506t"
}
```


Figura 20 - Dettagli API downloadFile

POST HUB_API - ControllaToken

https://sperimentazione.easypal.it/EasyPAL_HUB_API/api/Account/ControllaToken

HEADERS

Content-Type

application/json

BODY raw


```
{
    "token": "QlhubTRJcUtHNWdTM2cxd3ZtYmsrRlM4QUdsMUJXOGVJNnltY3lxcEZXMD06cGlwcG86NjM3MTAwNDQyOTg1MTkzMTI1",
    "cf": "rssmra80a01E506t"
}
```


Figura 21 - Dettagli API controllaToken

POST HUB_API - Logout

https://sperimentazione.easypal.it/EasyPAL_HUB_API/api/Account/Logout

HEADERS

Content-Type

application/json

BODY raw


```
{
    "codiceFiscale": "rssmra80a01U678t",
    "token": "QzdSTWJvNHp1d1RMUzBkejh5ZHZ5eFpVbjVxZVUvL011RGdZVU5XbThhOD06cGlwcG86NjM2OTg4Nzc1Mjc3Mjk2OTgy"
}
```


Figura 22 - Dettagli API Logout

POST HUB_API - getElencoComuniANPR

https://sperimentazione.easypal.it/EasyPAL_HUB_API/api/ANPR/getElencoComuniANPR

HEADERS

Content-Type

application/json

BODY raw


```
{
    "token": "QlhubTRJcUtHNWdTM2cxd3ZtYmsrRlM4QUdsMUJXOGVJNnltY3lxcEZXMD06cGlwcG86NjM3MTAwNDQyOTg1MTkzMTI1",
    "cf": "rssmra80a01E506t",
    "codiceComune": "A225"
}
```


Figura 23 - Dettagli API getElencoComuniANPR

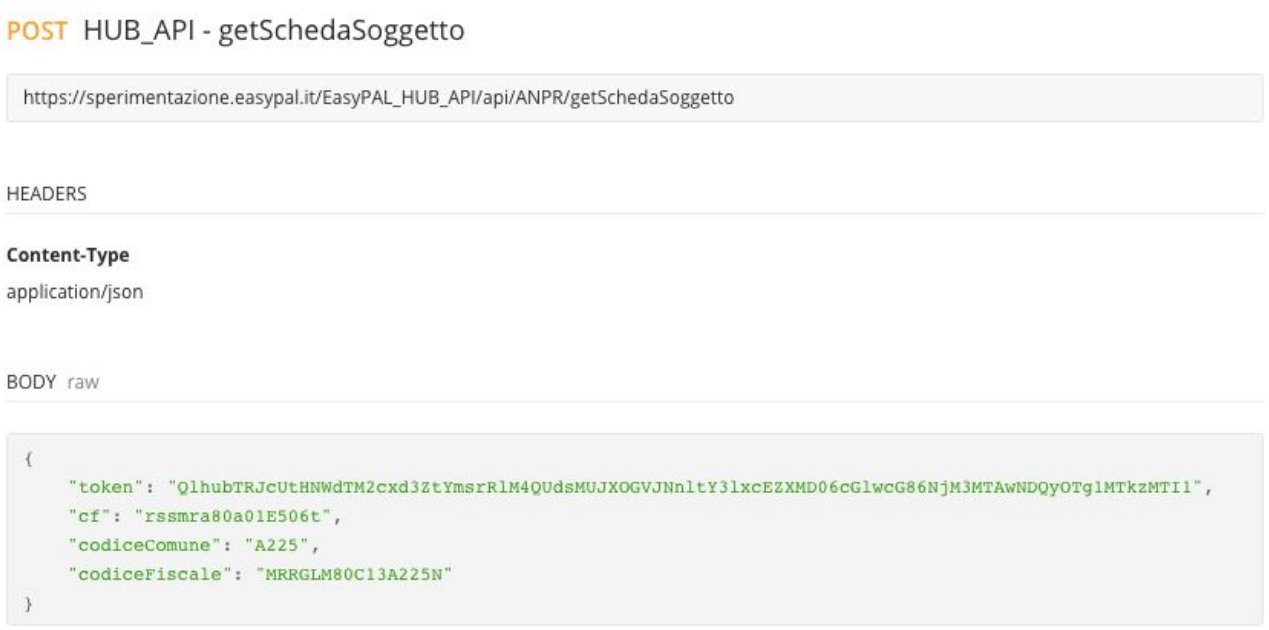


Figura 24 - Dettagli API getSchedaSoggetto

3.3 Risultati

I penetration test condotti hanno rilevato 5 tipologie differenti di vulnerabilità negli undici endpoint esposti dalla piattaforma. Di queste, una risulta essere di elevata criticità e ad alto impatto sulla sicurezza del sistema, mentre le restanti quattro hanno un livello di criticità medio. I risultati sommari sono riportati in Tabella 2.

Tabella 2 – Riepilogo sommario dei penetration test

Risk	Count
High	1
Medium	4
Low	0
Total	5

I dettagli relativi ad ogni criticità sono riportati in Tabella 3. Per ognuna di essere, in particolare, è riportato l'indice *Common Vulnerability Scoring System (CVSS)*, una metrica comunemente utilizzata per acquisire le principali caratteristiche vulnerabilità e produrre un punteggio numerico che ne rifletta la gravità.

La metrica è gestita da FIRST.Org, Inc., un'organizzazione no profit la cui mission è aiutare le security task force mondiali nella gestione degli attacchi di sicurezza informatica. Attualmente è alla versione 3.1 [2].

I risultati CVSS sono riportati su una scala da 0 a 10 e solitamente tradotti in un *4 point-Likert scale* per avere dei valori facilmente interpretabili [3], [4].

Tabella 3 - Riepilogo dettagliato dei penetration test

Vulnerability Name	Risk	Severity	Cvss score	Occurrences
Cross-Site Scripting	High	High	7.1	1
Improper Exception Handling	Medium	Medium	5.0	2
Divulgazione di informazioni sensibili nei response header	Medium	Medium	5.0	11
Response headers di sicurezza non presenti	Medium	Medium	5.0	11
HTTP Strict Transport Security assente	Medium	Medium	5.0	9

3.3.1 Test Cross-Site Scripting (XSS)

Gli attacchi di tipo *Cross-Site Scripting (XSS)* sono del genere "injection", in cui cioè script malevoli vengono iniettati forzatamente nei siti Web. Gli attacchi XSS sono una vulnerabilità che affligge siti web dinamici che impiegano un insufficiente controllo dell'input nei form. Un XSS permette a un cracker di inserire o eseguire codice lato client al fine di attuare un insieme variegato di attacchi quali, ad esempio, raccolta, manipolazione e reindirizzamento di informazioni riservate, visualizzazione e modifica di dati presenti sui server, alterazione del comportamento dinamico delle pagine web, ecc [5].

I test hanno evidenziato una vulnerabilità agli attacchi XSS solo in corrispondenza dell'endpoint *LoginOperatore*: inviando uno script Javascript iniettato nella richiesta all'API, lo stesso script viene ritornato nella response lato server. L'evidenza dell'attacco è mostrata in Tabella 4.

Tabella 4 – Evidenza di XSS endpoint LoginOperatore

Request	Response

{
"username":"pippo",
"password":"pluto",
"client_id":"4439AB6D-7006-4A03-A51C-AA0925C4 7153",
"scope":"",
"session_id":"**<script>alert(123)</script>**"
}
	{
	"idAccount":"1",
	"access_token":"dk1VRjZLeX.....",
	"scope":null,
	"cognome":"rossi",
	"nome":"mario",
	"username":"pippo",
	"messaggioErrore":"",
	"codicefiscale":"rssmra80a01E506t",
	"session_id":"<script>alert(123)</script>",
	"tipoUtenza":"2",
	"Error":
	{"status":200,
	"userMessage":"OK",
	"internalMessage":"OK"}
	}

3.3.2 Improper Exception Handling

La gestione impropria degli errori può comportare una varietà di problemi di sicurezza per un sito Web. Il problema più comune sussiste nel momento in cui l'utente ha la possibilità di visualizzare messaggi di errore interni dettagliati, come ad esempio stacktrace, dump del database e codici di errore, rivelando dettagli implementativi delicati dal punto di vista della sicurezza, poiché potrebbero rivelare importanti indizi su potenziali difetti nel sito.

I test hanno evidenziato una vulnerabilità riguardante un'errata gestione delle eccezioni in corrispondenza dell'endpoint *getElencoComuniANPR*: in corrispondenza dell'invio di un token errato la response lato server riporta un log estremamente dettagliato dell'errore.

Tabella 5 – Evidenza di Improper Exception in getElencoComuniANPR

Request	**Response**

{
{"token":"
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaa","
cf":"rssmra80a01E506t","codiceCom
une":"A225"}
 	
}
	{
	500 - {"Message":"Si è verificato un errore.","ExceptionMessage":"Riferimento a un oggetto non impostato su un'istanza di oggetto.","ExceptionType":"System.NullReferenceException","StackTrace":" in EasyPAL_HUB_API.Controllers.ANPRController.getElencoComuniANPR(JObject jsonData) in C:\\Workspaces\\EasyPAL\\EasyPAL_Solution\\EasyPAL_HUB_API\\Controllers\\ANPRController.cs:riga 846\r\n in lambda_method(Closure , Object , Object[])\r\n in System.Web.Http.Controllers.ReflectedHttpActionDescriptor.ActionExecutor.<>c__DisplayClass6_2.<GetExecutor>b__2(Object instance, Object[] methodParameters)\r\n in System.Web.Http.Controllers.ReflectedHttpActionDescriptor.ExecuteAsync(HttpControllerContext controllerContext, IDictionary`2 arguments, CancellationToken cancellationToken)\r\n--- Fine traccia dello stack da posizione precedente dove è stata generata l'eccezione ---\r\n in System.Runtime.ExceptionServices.ExceptionDispatchInfo.Throw()\r\n in System.Runtime.CompilerServices.TaskAwaiter.HandleNonSuccessAndDebuggerNotification(Task task)\r\n in System.Web.Http.Controllers.ApiControllerActionInvoker.<InvokeActionAsyncCore>d__1.MoveNext()\r\n--- Fine traccia dello stack da posizione precedente dove è stata generata l'eccezione ---\r\n in System.Runtime.ExceptionServices.ExceptionDispatchInfo.Throw()\r\n in System.Runtime.CompilerServices.TaskAwaiter.HandleNonSuccessAndDebuggerNotification(Task task)\r\n in System.Web.Http.Controllers.ActionFilterResult.<ExecuteAsync>d__5.MoveNext()\r\n--- Fine traccia dello stack da posizione precedente dove è stata generata l'eccezione ---\r\n in System.Runtime.ExceptionServices.ExceptionDispatchInfo.Throw()\r\n in System.Runtime.CompilerServices.TaskAwaiter.HandleNonSuccessAndDebuggerNotification(Task task)\r\n in System.Web.Http.Dispatcher.HttpControllerDispatcher.<SendAsync>d__15.MoveNext()"}
	}

3.3.3 Divulgazione di informazioni sensibili nei response header

Diffusamente ricorrenti sono le richieste mirate ai sistemi server con l'obiettivo di ottenere informazioni sulla configurazione degli stessi sistemi. Se il sistema non è ben configurato, potrebbe esporre informazioni come ad esempio la versione e le configurazioni del sistema e delle librerie. Questo tipo di problemi non sono sfruttabili nella maggior parte dei casi ma sono considerati problemi di sicurezza perché consentono ad eventuali aggressori di radunare informazioni che possono essere utilizzate in seguito nel ciclo di vita di attacchi di hacking.

I test hanno evidenziato un'errata divulgazione di informazioni sensibili nei response header in tutti gli undici endpoint della piattaforma, poiché in ogni response sono presenti informazioni di configurazione del sistema server troppo dettagliate, benchè poco utili ai fini di un'elaborazione ordinaria della risposta. In Tabella 6 viene riportata un'evidenza di esempio, valida per tutti gli endpoint esposti.

Tabella 6 - Evidenza di divulgazione di informazioni sensibili LoginOperatore

Request	**Response**
Method: POST Content-Type: application/json	access-control-expose-headers: Request-Context cache-control: no-cache content-length: 273 content-type: application/json; charset=utf-8 date: Mon, 25 Nov 2019 10:44:53 GMT expires: -1 pragma: no-cache request-context: appId=cid-v1:29e3b8fa-e5ae-41a8-951f-0975f72be016 server: Microsoft-IIS/10.0 status code: 200 x-aspnet-version: 4.0.30319 x-powered-by: ASP.NET

3.3.4 Response headers di sicurezza non presenti

Dualmente alle indicazioni di vulnerabilità descritte nel precedente paragrafo è buona norma riportare nelle response server-side degli header utili a migliorare i meccanismi di sicurezza implementabili: una volta impostate, tali header HTTP possono limitare l'implementazione di vulnerabilità facilmente prevenibili nei moderni browser. Dai test risulta che per gli endpoint della piattaforma sarebbe necessario aggiungere i seguenti header di sicurezza:

1. X-XSS-Protection: X-XSS-Protection: 1; mode=block
2. X-Frame-Options : X-Frame-Options: deny
3. X-Content-Type-Options : X-Content-Type-Options: nosniff

Di seguito i dettagli:

X-XSS-Protection header è una funzionalità di Internet Explorer, Chrome e Safari che impedisce il caricamento di pagine web quando vengono rivelati attacchi di tipo *Cross-site Scripting Reflected (XSS)*. Anche se queste protezioni sono largamente non necessarie nei browser moderni, quando i siti web implementano una *Content-Security-Policy (CSP)* restrittiva che disabilita l'uso di JavaScript inline ('unsafe-inline'), possono ancora fornire protezione agli utenti di browser datati che non supportano CSP.

X-Frame-Options header può essere utilizzato per indicare se un browser è autorizzato ad eseguire il rendering di una pagina embedded in *<frame>, <iframe>, <embed> o <object>*. È utile per evitare attacchi di *clickjacking*, garantendo quindi che il contenuto di un sito web non sia incorporato in altri siti malevoli.

X-Content-Type-Options header è usato per indicare che il tipo MIME descritto dagli header di tipo Content-Type dovrebbe esser rispettato senza effettuare modifiche. Tale opzione non permette ai client di effettuare lo *sniffing del MIME type*, pratica spesso utilizzata dai comuni browser quando il MIME type non è specificato. Può presentare sia problemi di sicurezza da un lato sia una caratterizzazione poco dettagliata delle response server-side dall'altro.

3.3.5 HTTP Strict Transport Security assente

HTTP Strict Transport Security (HSTS) è un miglioramento della sicurezza opt-in specificato da un'applicazione Web mediante l'uso di un header di risposta specifico. I browser che ricevono tale header impongono che tutte le comunicazioni con il dominio specificato siano effettuate mediante protocollo HTTPS, evitando quindi il ben noto attacco *man in the middle*.

Dai test effettuati, è stato rilevato che in 9 casi su 11 l'header in oggetto non è presente nelle response server-side, in particolare per tutti gli endpoint in esame meno *getElencoComuniANPR* e *getSchedaSoggetto*.

Riferimenti

[1] J. R. Horgan, S. London, and M. R. Lyu, "Achieving Software Quality with Testing Coverage Measures," *Computer (Long. Beach. Calif).*, vol. 27, no. 9, pp. 60–69, 1994.

[2] "CVSS v3.1 Specification Document." [Online]. Available: https://www.first.org/cvss/v3.1/specification-document. [Accessed: 05-Jan-2020].

[3] "A software application to analyze the effects of temporal and environmental metrics on overall CVSS v2 score - IEEE Conference Publication." [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5749893. [Accessed: 05-Jan-2020].

[4] "On prioritization of vulnerability categories based on CVSS scores - IEEE Conference Publication." [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6316705. [Accessed: 05-Jan-2020].

[5] J. Grossman, *XSS attacks : cross-site scripting exploits and defense*. Syngress, 2007.