TINEXTA CYBER

dinova

WAPT – Municipium Cloud Backoffice

Maggioli

GRUPPO Maggioli

Tabella Revisioni

Versione	Autore	Revisore	Data	Pagine	Commenti
1.0	Marianna Fucci	Marco Prati	23/07/2024	28	-

Note Legali

Copyright Tinexta Cyber S.p.A. 2024.

Tutti i diritti riservati. Il presente documento è di classificazione TLP:AMBER ed è destinato all'uso esclusivo da parte di Maggioli per gli scopi da esso riportati in oggetto. Sono espressamente vietati l'uso, la riproduzione e la diffusione non autorizzata di questo documento.

Limitazioni all'Uso del Documento

Questa relazione contiene informazioni riguardanti incidenti e minacce riscontrate all'interno della rete di Maggioli e riporta dettagli riguardanti vittime e tecniche di attacco. Tinexta Cyber S.p.A. raccomanda di prendere particolari precauzioni per proteggere la riservatezza di questo documento e le informazioni in esso contenute.

Tinexta Cyber S.p.A.
Società soggetta alla Direzione e Coordinamento di Tinexta S.p.A
www.tinextacyber.com - info@tinextacyber.com

Sede Legale
Piazza Sallustio, 9, 0018, Roma

Cap.Soc. €1.000.000, P.IVA/C.F. 15971011000, REA RM – 1626691

Indice

1 Introduzione 6
1.1 Contesto 6
1.2 Scopo 6
1.3 Contenuto 6
2 Metodologia di testing 7
2.1 Metodologia OWASP-based 7
2.2 Classificazione dei risultati delle vulnerabilità 8
3 Superficie di Test 10
3.1 Sistemi in Analisi 10
3.2 Utenze utilizzate 10
3.3 Materiale Fornito 10
3.4 Out of scope, vincoli e limiti operativi 10
4 Executive Summary 11
4.1 Piano di mitigazione proposto 13
4.2 Security Posture 13
5 Dettaglio Risultati 14
5.1 Vulnerabilities 14
V-001 – Insecure Storage of Sensitive Information 14
V-002 – Cross Site Scripting (XSS) 16
V-003 – Unrestricted File Upload 18
V-004 – Insecure Direct Object Reference (IDOR) 20
V-005 – Broken Access Control 23
V-006 – Information Disclosure 25
V-007 – Improper Session Handling 27
5.2 Altre evidenze 28
5.2.1 Security Headers Misconfiguration 28

Indice delle Figure

Figura 4-1 Vulnerabilità rilevate classificate per severità 11
Figura 4-2 Vulnerabilità rilevate classificate secondo OWASP Top 10 12
Figura 5-1 - Informazioni sensibili in chiaro all'interno della risposta HTTP 15
Figura 5-2 - Inserimento di codice javascript 17
Figura 5-3 - Esecuzione del codice javascript nel browser 17
Figura 5-4 - Evidenza caricamento di un file contenente un payload xss 19
Figura 5-5 - File caricato con successo con esecuzione del codice javascript inserito all'interno 19
Figura 5-6 - Richiesta HTTP per effettualre l'enumerazione dei profili degli utenti 20
Figura 5-7 - Risposta contenente le informazioni degli utenti 21
Figura 5-8 - Richiesta HTTP per effettualre l'enumerazione dei contatti 21
Figura 5-9 - Risposta contenente le informazioni dei contatti 22
Figura 5-10 - Richiesta HTTP per effettuare l'enumerazione degli avvisi 22
Figura 5-11 - Risposta contente le informazioni di file associate a bucket Amazon s3 22
Figura 5-12 - Vista dell'interfaccia amministrativa richiamata da utente con privilegi normali - l'opzione Administrators -> Administrative list 23
Figura 5-13 - Vista della sezione Attachment da utente base con ruolo Author 24
Figura 5-14 - Versione di nginx 25
Figura 5-15 - Errori applicativi prolissi 26
Figura 5-16 - Information disclosure di dati interni all'infrastruttura 26
Figura 5-17 - Headers di sicurezza mancanti 28

Indice delle Tabelle

Tabella 2-1 Livelli e punteggi della metrica Severità 8
Tabella 2-2 Tipologie di Skills dell'Attaccante 8
Tabella 2-3 Tipologie di Condizioni al contorno 9
Tabella 2-4 Tipologie di Impatto 9
Tabella 3-1 Sistemi in Analisi 10
Tabella 3-2 Utenze utilizzate per i test 10
Tabella 4-1 Vulnerabilità riscontrate 11
Tabella 4-2 Criticità rilevate in riferimento a OWASP Top 10 12
Tabella 4-3 Top 5 Vulnerabilità "Quick Win" 13

1 Introduzione

Su richiesta di Maggioli, Tinexta Cyber S.p.A. ha effettuato la valutazione della sicurezza delle infrastrutture in uso presso l'organizzazione del Cliente. Lo scopo di questa valutazione è l'identificazione di problemi di sicurezza e conseguenti vulnerabilità che interessano server ed applicazioni che gestiscono dati dell'organizzazione.

1.1 Contesto

Questa relazione presenta i risultati della valutazione della vulnerabilità dell'asset oggetto di test. Per valutare la sicurezza della rete e delle applicazioni, Tinexta Cyber S.p.A. tenta di eseguire transazioni non autorizzate, ottenere informazioni riservate e determinare la sicurezza generale delle applicazioni eseguendo una grande varietà di controlli di vulnerabilità. Il test comprende anche i server, sistemi operativi e dispositivi di rete associati all'organizzazione del Cliente e previsti negli accordi riguardanti gli asset sottoposti a verifica.
I risultati di questa relazione riflettono le condizioni trovate durante il periodo di svolgimento delle attività e non riflettono necessariamente le condizioni attuali.

1.2 Scopo

Lo scopo di questa valutazione è quello di individuare problemi di sicurezza a livello di rete ed applicativo che potrebbero pregiudicare la confidenzialità, l'integrità e la disponibilità di dati e servizi gestiti dal Cliente e delle sue infrastrutture di rete. Le attività di verifica svolte sugli asset informatici del Cliente hanno lo scopo di delineare i livelli globali di sicurezza delle applicazioni e delle infrastrutture in uso. Nel dettaglio i test richiesti hanno lo scopo di:

- Valutare lo stato delle infrastrutture informatiche sotto il profilo della sicurezza con enfasi su vulnerabilità applicative dell'asset analizzato;
- Misurare il reale livello di rischio delle piattaforme attualmente in uso nei confronti di un attaccante posto esternamente alla rete;
- Determinare le possibili conseguenze dovute allo sfruttamento delle vulnerabilità rilevate e relativi impatti sul business dell'organizzazione del Cliente;
- Fornire indicazioni su azioni da intraprendere per la minimizzazione dei rischi associati alle problematiche individuate.

1.3 Contenuto

Questa relazione presenta i risultati del web application penetration test così organizzati:

- Panoramica delle principali vulnerabilità identificate;
- Elenco di tutte le vulnerabilità in ordine di criticità e spiegazione dettagliata delle vulnerabilità principali;
- Descrizione di possibili vettori di attacco che un attaccante può seguire sfruttando le vulnerabilità rilevate.

2 Metodologia di testing

Tinexta Cyber S.p.A. utilizza una combinazione delle metodologie delineate come “best-in-class”, come *OpenSource Security Testing Methodology Manual* (**OSSTMM**) v3 di ISECOM e *Open Web Application Security Project* (**OWASP**) *Testing Guide* v4.
I domini funzionali di OSSTMM in linea con l'attività di verifica svolte sono elencati di seguito:

- Raccolta informazioni su obiettivi, rilevamento tecnologie ed enumerazione risorse applicative;
- Verifica e scansione dei servizi;
- Ricerca vulnerabilità note delle piattaforme tecnologiche in uso, utilizzo di strumenti di scansione automatici e testing manuale;
- Analisi vulnerabilità, determinazione rischi associati e validazione in caso di basso rischio di disservizio.

2.1 Metodologia OWASP-based

La metodologia OWASP viene utilizzata principalmente per testare le applicazioni Web classiche, come i siti Web o le backend API. L'elenco che segue mostra i principali problemi di sicurezza segnalati dalla *OWASP Top Ten* inclusa nella metodologia di test di Tinexta Cyber S.p.A.:

- A01 – Broken Access Control;
- A02 – Cryptographic Failures;
- A03 – Injection;
- A04 – Insecure Design;
- A05 – Security Misconfiguration;
- A06 – Vulnerable and Outdated Components;
- A07 – Identification and Authentication Failures;
- A08 – Software and Data Integrity;
- A09 – Security Logging and Monitoring Failures;
- A10 – Server Side Request Forgery (SSRF).

2.2 Classificazione dei risultati delle vulnerabilità

Ogni problema di sicurezza individuato da Tinexta Cyber S.p.A. è descritto con il rispettivo rilevamento, la severità del problema, quanto sarebbe facile per un utente malintenzionato lo sfruttamento della vulnerabilità e una raccomandazione/suggerimento per la risoluzione della problematica. Ogni problema di sicurezza è valutato sulla base di una serie di fattori, ciascuno dei quali è descritto nelle seguenti sezioni.
La classificazione delle vulnerabilità avviene secondo la metrica di **Common Vulnerability Scoring System Version 3.1** (**CVSSv3.1**), il quale fornisce un open framework per la comunicazione delle caratteristiche e degli impatti delle vulnerabilità IT. Il suo modello garantisce misurazioni accurate ed è utilizzabile come standard per industrie, organizzazioni e governi che hanno la necessità di disporre di punteggi di impatto di vulnerabilità accurati e coerenti.
Secondo questa metrica ogni vulnerabilità viene catalogata attraverso un Vettore che riassume la totalità dei molteplici elementi di una vulnerabilità, i quali contribuiscono a formare un punteggio che definisce il livello di severità della vulnerabilità.

A seguito di un confronto con il Cliente, vengono specificati i principali elementi che compongono la metrica **CVSSv3.1**, per i quali è interessante dare una definizione più specifica in modo da fornire un quadro di comprensione maggiore.
Di seguito sono quindi riportati gli elementi elencati all'interno delle tabelle delle vulnerabilità rilevate per scenario, mentre le specifiche complete delle restanti voci della metrica si rimanda al link: https://www.first.org/cvss/specification-document

Severità: metrica che riassume, attraverso un punteggio e un livello elencati di seguito, la gravità della vulnerabilità rilevata, come si può vedere in Tabella 2-1:

Livello	Punteggio CVSSv3.1
None	0.0
Low	0.1 – 3.9
Medium	4.0 – 6.9
High	7.0 – 8.9
Critical	9.0 – 10.0

Tabella 2-1 Livelli e punteggi della metrica Severità

Complessità di Accesso: metrica che descrive e misura la quantità di conoscenza, di capacità e di interazione richiesta ad un ipotetico attaccante al fine di sfruttare con successo la vulnerabilità rilevata. La complessità di accesso viene mostrata all'interno della sezione che dettaglia i risultati ottenuti sotto forma di un valore ***Alta***, ***Media*** o ***Bassa***, ma è determinata a partire da alcune componenti, tra le quali:

- **Skills dell'Attaccante**, ossia le abilità tecniche che un attaccante deve possedere per poter eseguire correttamente un attacco, suddivise in basse, medie e alte come indicato in Tabella 2-2.

Skills	Descrizione
Basse	L'abilità richiesta all'attaccante è bassa o nulla, l'attacco può essere eseguito utilizzando tool già pronti o seguendo brevi tutorial disponibili in rete.
Medie	Le abilità necessarie possono essere acquisite tramite autoapprendimento nell'arco di alcune settimane o pochi mesi, senza richiedere complesse conoscenze pregresse.
Alte	Sono necessarie abilità complesse, non ottenibili in poche settimane o mesi, e che richiedono un background di conoscenze specifico.

Tabella 2-2 Tipologie di Skills dell'Attaccante

- **Condizioni al contorno**, ossia quelle condizioni aggiuntive necessarie per classificare in modo più preciso la probabilità che un attacco sia a portato a termine, in funzione dell'ambiente presente e di tutte le ulteriori condizioni collaterali, come si può vedere in Tabella 2-3.

Condizioni al contorno	Descrizione
Basse	Lo sfruttamento della vulnerabilità rilevata non richiede particolari skill e allo stesso tempo esistono condizioni molto favorevoli che permettono di raggiungere l'obiettivo facilmente. Per esempio, non sono presenti particolari barriere, non è richiesto alcun privilegio, non vengono richieste interazioni da parte della vittima, non viene richiesta la scrittura di tool ad-hoc ma sono sufficienti software open source reperibili online per portare a termine l'attacco.
Medie	Esistono condizioni mediamente favorevoli per permettere ad un attaccante la possibilità di sfruttare una vulnerabilità. Per esempio, potrebbe essere necessario un accesso più diretto alle infrastrutture del Cliente, potrebbe essere richiesto un più alto livello di privilegi assieme ad una eventuale interazione parziale dell'utente vittima, non sono sufficienti strumenti software facilmente reperibili ma spesso ci si deve affidare ad una combinazione di strumenti software e hardware configurati ad-hoc per portare a termine l'attacco.
Alte	Esistono condizioni ostili per permettere ad un attaccante di sfruttare una vulnerabilità. Per esempio, potrebbero essere presenti grosse barriere di accesso alle infrastrutture del Cliente, potrebbe essere richiesto un livello di privilegi molto alto assieme ad una eventuale continua e completa interazione della vittima, potrebbero non essere sufficienti strumenti software e hardware configurati ad-hoc ma occorrerebbe crearne di nuovi per l'occasione. Potrebbero essere richieste eventuali attività di movimento laterale e/o persistenza obbligatorie per il raggiungimento dell'obiettivo.

Tabella 2-3 Tipologie di Condizioni al contorno

Impatto: metrica che specifica l'impatto che una vulnerabilità ha sui componenti di Confidenzialità, Integrità e Disponibilità (CIA), come si può vedere in Tabella 2-4. Questi componenti, parte integrante della metrica di *CVSSv3.1*, saranno condensati all'interno della sezione che dettaglia i risultati ottenuti sotto forma di un valore ***Alto***, ***Medio*** o ***Basso***.

Metrica	Descrizione
Confidenzialità	Misura l'impatto sulla riservatezza delle risorse informative gestite da un componente software a causa di una vulnerabilità sfruttata con successo. La riservatezza si riferisce alla limitazione dell'accesso e della divulgazione delle informazioni solo agli utenti autorizzati, nonché alla prevenzione dell'accesso o della divulgazione a persone non autorizzate.
Integrità	Misura l'impatto sull'integrità di una vulnerabilità sfruttata con successo. L'integrità si riferisce all'affidabilità e alla veridicità delle informazioni.
Disponibilità	Misura l'impatto sulla disponibilità del componente interessato risultante da una vulnerabilità sfruttata con successo. Si riferisce alla perdita di disponibilità del componente interessato stesso, come il servizio di rete (es. Web, database, e-mail). Poiché la disponibilità si riferisce all'accessibilità delle risorse informative, gli attacchi che consumano la larghezza di banda della rete, i cicli del processore o lo spazio su disco hanno un impatto sulla disponibilità di un componente interessato.

Tabella 2-4 Tipologie di Impatto

3 Superficie di Test

La valutazione è stata effettuata su una parte delle infrastrutture del Cliente, come da accordi di servizio. Gli specifici sistemi e sottoreti testate sono indicati nella sezione Sistemi in Analisi. I test effettuati sono da intendere come una verifica globale dello stato della rete del Cliente ed i sistemi e le sottoreti riportate in seguito ricadono negli scopi e negli obiettivi delle attività svolte. Inoltre, i risultati riportati in questa relazione riflettono le condizioni riscontrate durante i test, e non riflettono necessariamente le condizioni attuali.

3.1 Sistemi in Analisi

Nel dettaglio di Tabella 3-1 sono stati oggetto di verifica i seguenti host:

ID	Asset	Tipo	Versione	Note
T-1	https://staging.municipiumapp.it/#/passport/login	TEST		

Tabella 3-1 Sistemi in Analisi

3.2 Utenze utilizzate

Nel dettaglio di Tabella 3-2 sono riportate le utenze fornite ed utilizzate per effettuare i test.

ID	Utenza	Note
U-1	cla_r2@libero.it	Utenza con privilegi limitati
U-2	cla_r3@libero.it	Utenza con privilegi limitati
U-3	cla_r@libero.it	Utenza con privilegi amministrativi

Tabella 3-2 Utenze utilizzate per i test

3.3 Materiale Fornito

Non è stato fornito alcun materiale aggiuntivo relativo agli asset oggetto dei test, oltre a quelli sopra citati.

3.4 Out of scope, vincoli e limiti operativi

Le attività sono state condotte in ambiente di test, a partire da IP precedentemente comunicati al cliente.

4 Executive Summary

Durante il periodo di test Tinexta Cyber S.p.A. ha analizzato un sistema appartenente all'organizzazione del Cliente. Le vulnerabilità e percorsi di attacco rilevati comprendono:

- **0** Vulnerabilità di severità **critica**;
- **1** Vulnerabilità di severità **alta**;
- **4** Vulnerabilità di severità **media**;
- **2** Vulnerabilità di severità **bassa**.

Il grafico in Figura 4-1 mostra la ripartizione delle vulnerabilità individuate. I colori rosso, arancio, giallo e verde indicano la presenza di vulnerabilità di severità critica, alta, media e bassa all'interno degli asset analizzati, fornendo un'importante indicazione sulla loro postura di sicurezza attuale.

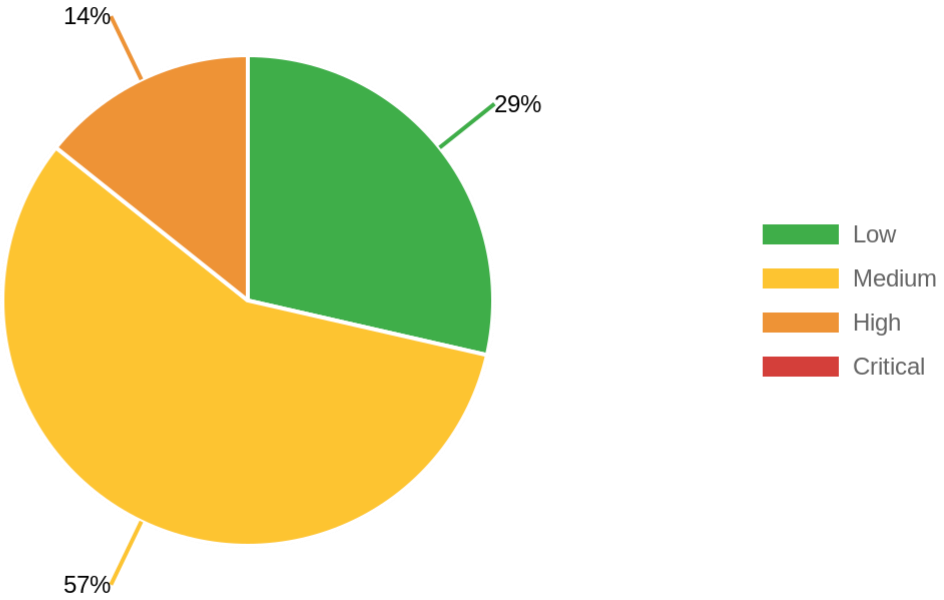

Figura 4-1 Vulnerabilità rilevate classificate per severità

La tabella Tabella 4-1 mostra tutte le vulnerabilità in ordine di cvss rinvenute durante l'attività.

ID	Titolo	CVSS Score
V-001	Insecure Storage of Sensitive Information	**7.9**
V-002	Cross Site Scripting (XSS)	**5.0**
V-003	Unrestricted File Upload	**4.6**
V-004	Insecure Direct Object Reference (IDOR)	**4.3**
V-005	Broken Access Control	**4.0**
V-006	Information Disclosure	**3.8**
V-007	Improper Session Handling	**3.2**

Tabella 4-1 Vulnerabilità riscontrate

Con riferimento alla OWASP top 10, le criticità rilevate sono riassunte in Tabella 4-2:

Asset ID	OWASP ID										Other Vulns
	A01	A02	A03	A04	A05	A06	A07	A08	A09	A10	OV
T-1	2	1	1	1	1	0	1	0	0	0	0

Tabella 4-2 Criticità rilevate in riferimento a OWASP Top 10

Le vulnerabilità riscontrate rientrano nelle differenti categorie OWASP di:

- A01 - Broken Access Control
- A02 - Cryptographic Failures
- A03 - Injection
- A04 - Insecure Design
- A05 - Security Misconfiguration
- A07 - Identification and Authentication Failures

Il grafico in Figura 4-2 mostra la ripartizione delle vulnerabilità individuate in base alla loro severità e categoria **OWASP** appartenente.

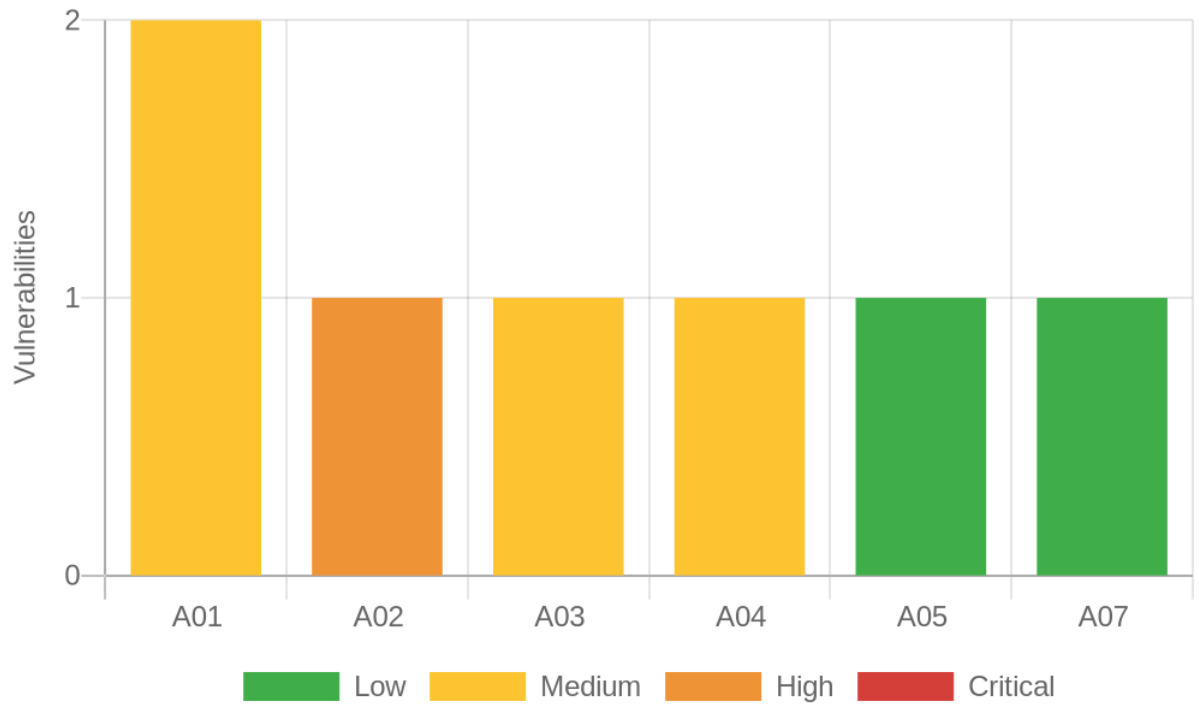

Figura 4-2 Vulnerabilità rilevate classificate secondo OWASP Top 10

4.1 Piano di mitigazione proposto

Le vulnerabilità rilevate e sopra riassunte hanno differenti tipologie ed effort di mitigazione. Alcuni rilevamenti potrebbero addirittura "amplificare" l'impatto di altre vulnerabilità, di fatto creando le basi di una catena di sfruttamento di vulnerabilità a maggior impatto rispetto alle singole componenti.

Questa sezione vuole identificare le vulnerabilità che, se mitigate per prime, permettono di ridurre in maniera significativa la superficie d'attacco. L'elenco di queste vulnerabilità prende in considerazione l'effort **tecnico** necessario a mitigarle, secondo una stima dell'analista che ha eseguito l'attività. Questo effort **stimato** non tiene in considerazione eventuali limiti o vincoli tecnologici e/o di business, e che comunque non sono in possesso dell'analista.

La Tabella 4-3 riassume i cinque "Quick Win": vulnerabilità ordinate per impatto sulla security posture, che possono – con un effort tecnico **stimato** medio/basso – garantire una veloce copertura sulle maggiori aree di esposizione.

ID	Top 5 – Quick Win	CVSS Score	Effort
V-001	Insecure Storage of Sensitive Information	7.9	**Medio**
V-002	Cross Site Scripting (XSS)	5.0	**Medio**
V-003	Unrestricted File Upload	4.6	**Medio**
V-004	Insecure Direct Object Reference (IDOR)	4.3	**Medio**
V-005	Broken Access Control	4.0	**Medio**

Tabella 4-3 Top 5 Vulnerabilità "Quick Win"

Per ridurre il perimetro d'attacco si riassumono le principali raccomandazioni atte a mitigare i rischi indicati in Tabella 4-3:

- Evitare di conservare e scambiare **informazioni sensibili in maniera non sicura** (V-001)
- **Eseguire** una **corretta sanitizzazione** dei **dati controllabili dagli utenti** sia in input che in output (V-002)
- Implementare un meccanismo di controllo per il contenuto dei file caricati dagli utenti (V-003)
- Identificare oggetti e ruoli tramite identificati randomici, per **evitarne la predicibilità** (V-004)
- **Applicare controlli adeguati** al fine di gestire le **autorizzazioni** dei singoli utenti in tutte le aree del portale (V-005)

4.2 Security Posture

A seguito della verifica di sicurezza condotta, è stato riscontrato che la postura di sicurezza del portale Municipium Cloud Backoffice si attesta su un **livello medio/basso**.
In particolare, sono emersi diversi punti d'attenzione i cui rischi possono essere riassunti come segue:

- All'interno delle risposte del server sono state identificate:

Versioni di tecnologie usate dall'applicazione e risorse interne all'infrastruttura del cliente
Presenza di credenziali in chiaro

- É stato possibile iniettare codice html e/o javascript all'interno dell'applicativo, in quanto la mancata sanitizzazione dei dati inseriti dall'utente ha permesso di condurre attacchi di tipo Cross Site Scripting.
- Risulta possibile effettuare l'upload di file arbitrari, in quanto manca la validazione sul tipo/contenuto del file
- I riferimenti di oggetti/utenti/entità all'interno dell'applicativo è effettuato tramite ID incrementale e non randomico, permettendo ad un attaccante di ottenere informazioni di altri oggetti a visibilità di altri utenti.
- Risulta una mancata gestione della sessione dell'utente in quanto a seguito del logout degli utenti, il token JWT di sessione, risulta riutilizzabile.

5 Dettaglio Risultati

Di seguito sono riportati, nell'ordine:

- Vulnerabilità rilevate, con dettagli sintetici su descrizione e risoluzione;
- Scenari di attacco, che concatenano le vulnerabilità rilevate con l'obiettivo di raggiungere privilegi e/o funzionalità non previste;
- Altre evidenze, che includono bug e/o malfunzionamenti non strettamente legati ad ambiti di sicurezza.

5.1 Vulnerabilities

Insecure Storage of Sensitive Information				
V-001	SEVERITY	CVSS SCORE	ACCESS COMPLEXITY	IMPACT
	High	7.9	Low	High
	OWASP CATEGORY		OWASP CONTROL	
	A02 - Cryptographic Failures		WSTG-CRYP-04	
	CVSS 3.1 VECTOR			
	AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N/E:F/RL:W/RC:C/CR:H/IR:L/AR:L/MAV:N/MAC:L/MPR:L/MUI:N/MS:U/MC:H/MI:N/MA:N			

Asset

https://staging.municipiumapp.it/api/v1/admin/municipalityimpostazione/8093

Descrizione

La vulnerabilità di insecure storage of sensitive information si verifica quando le informazioni sensibili non vengono correttamente protette durante la loro memorizzazione, ad esempio quando vengono memorizzate in chiaro o in formati considerati non sicuri.
Questo può consentire ai potenziali attaccanti malevoli di accedere e leggere le informazioni. Anche se le informazioni sono codificate in un modo non leggibile dall'uomo, alcune tecniche potrebbero determinare quale codifica viene utilizzata, quindi decodificare le informazioni.
A volte i termini "cleartext" e "plaintext" vengono utilizzati per indicare la mancanza di crittografia, ma è importante sottolineare che all'interno della crittografia questi termini hanno significati più precisi. Il "plaintext" rappresenta l'informazione appena prima di essere inserita in un algoritmo crittografico, incluso il testo già crittografato, mentre il termine "cleartext" si riferisce a qualsiasi informazione non crittografata, anche se potrebbe essere in una forma codificata che non è facilmente leggibile dall'uomo (come la codifica base64).

Evidenza

Durante l'analisi di sicurezza dell'applicazione, è stata rilevata una vulnerabilità di "insecure storage of sensitive information". In particolare, le informazioni sensibili [come credenziali, dati personali ecc] sono state memorizzate in chiaro. Questo può consentire a potenziali attaccanti malevoli di accedere e leggere queste informazioni, mettendo a rischio la sicurezza degli utenti e della applicazione stessa.
Nel dettaglio, All'interno dell'applicativo, è stato possibile identificare la presenza di username e password in chiaro, al richiamo dell'API */api/v1/admin/municipalityimpostazione/####*

Si riporta di seguito la richiesta GET utilizzata:

```
GET /api/v1/admin/municipalityimpostazione/8093 HTTP/2
Host: staging.municipiumapp.it
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: application/json, text/plain, */*
Accept-Language: en-US
Accept-Encoding: gzip, deflate
Referer: https://staging.municipiumapp.it/
Authorization: Bearer [Token]
```

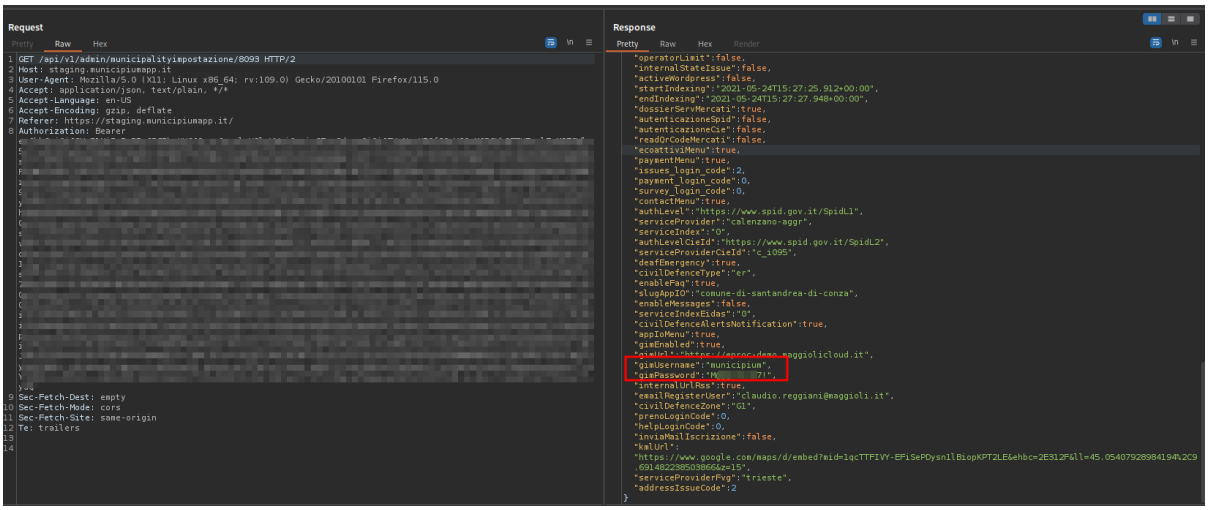

Figura 5-1 - Informazioni sensibili in chiaro all'interno della risposta HTTP

Mitigazione

Per garantire la sicurezza delle informazioni sensibili, è necessario applicare la crittografia in fase di archiviazione. Ciò implica l'utilizzo di algoritmi di crittografia affidabili e l'adozione di una gestione delle chiavi sicura. Inoltre, è importante considerare le specifiche esigenze di sicurezza dell'applicazione e adottare le misure adeguate alla protezione dei dati, come la definizione di politiche di accesso e l'utilizzo di controlli di sicurezza supplementari.

Riferimenti

- https://cwe.mitre.org/data/definitions/312.html
- https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure

Cross Site Scripting (XSS)				
V-002	SEVERITY	CVSS SCORE	ACCESS COMPLEXITY	IMPACT
	Medium	5.0	Low	Medium
	OWASP CATEGORY		OWASP CONTROL	
	A03 - Injection		WSTG-INPV-01,WSTG-INPV-02	
	CVSS 3.1 VECTOR			
	AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N/E:P/RL:W/RC:C/CR:X/IR:X/AR:X/MAV:N/MAC:L/MPR:L/MUI:R/MS:C/MC:L/MI:L/MA:N			

Asset

https://staging.municipiumapp.it/#/garbage/calendar_types/8093

Descrizione

Gli script lato client sono utilizzati ampiamente dalle moderne applicazioni web. Essi eseguono da semplici funzioni, come la formattazione del testo, fino alla manipolazione completa del DOM come nelle single page application.
Il cross site scripting (XSS) permette ad un utente malevolo di iniettare codice JavaScript in una richiesta e far sì che questo venga eseguito nel contesto del browser dell'utente vittima. Questo accade perché l'applicazione non sanifica correttamente i dati controllabili da un utente e li ripropone senza un corretto encoding durante la renderizzazione.
Se il codice iniettato viene restituito immediatamente, questo è noto come XSS riflesso, altrimenti se viene memorizzato dal server e restituito a qualsiasi utente che visiti la pagina compromessa, allora questo è noto come XSS persistente (anche XSS memorizzato).
Questa seconda tipologia è più pericolosa in quanto non obbliga l'attaccante a dover stabilire un contatto più diretto con la vittima.

Evidenza

Durante le analisi è stato riscontrato che il portale risulta afflitto da istanze di Stored Cross-Site Scripting.
L'applicativo presenta, infatti, delle problematiche nella sanitizzazione ed encoding dell'input utente, permettendo ad un utente di inserire codice javascript e/o html arbitrario all'interno delle response.
Nel dettaglio, all'interno della funzionalità *Garbage Utilities* è stato possibile iniettare codice javascript (*zzz<img src=1 onerror=confirm('MaggioliXSS')>*) che verrà eseguito poi dal browser della vittima nel momento in cui l'utente visita la pagina.

Di seguito la richiesta effettuata con l'inserimento del payload xss:

```
POST /api/v1/admin/calendartype/ HTTP/2
Host: staging.municipiumapp.it
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: application/json, text/plain, */*
Accept-Language: en-US
Accept-Encoding: gzip, deflate
Referer: https://staging.municipiumapp.it/
Authorization: Token
Content-Type: application/json
Content-Length: 146
Origin: https://staging.municipiumapp.it
{
"translations":{
      "en":"zzz<img src=1 onerror=confirm('MaggioliXSS')>",
      "it":"zzz<img src=1 onerror=confirm('MaggioliXSS')>"
      },
      "municipalityId":8093
}
```

Inoltre, si segnala che attraverso la funzione di **Upload Documents** della funzionalità **Attachments** è possibile caricare file inserendo un payload XSS. Maggiori dettagli sono descritti nella Vulnerabilità Unrestricted File Upload.

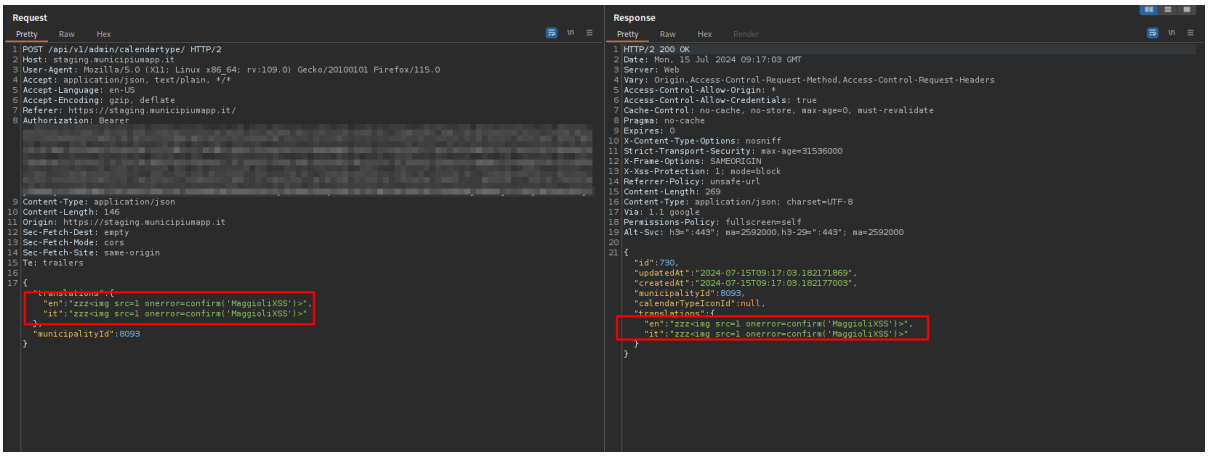

Figura 5-2 - Inserimento di codice javascript

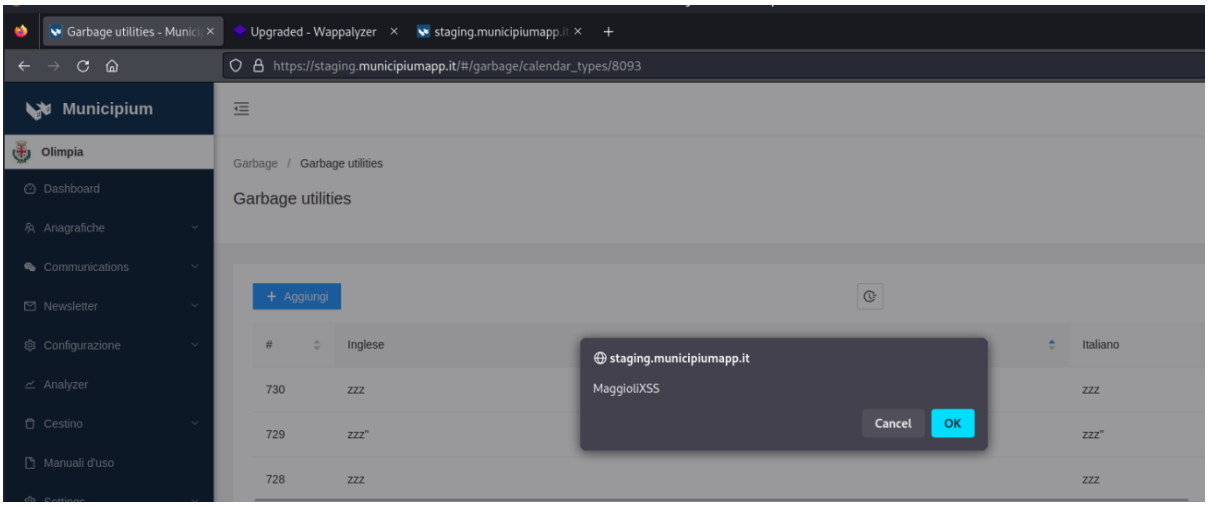

Figura 5-3 - Esecuzione del codice javascript nel browser

Mitigazione

Per rimediare alle vulnerabilità XSS, è importante non usare mai dati non fidati o non filtrati all'interno del codice di una pagina HTML. I dati non attendibili possono provenire non solo dal client, ma potenzialmente da una terza parte o da un file caricato in precedenza, ecc. Il filtraggio dei dati non attendibili tipicamente comporta la conversione dei caratteri speciali nelle loro controparti codificate come entità HTML (tuttavia, esistono altri metodi, vedi riferimenti). Questi caratteri speciali includono:
& < > ' /
Un esempio di codifica di entità HTML è la conversione di < in <. Sebbene sia possibile filtrare l'input non fidato, ci sono cinque posizioni all'interno di una pagina HTML dove l'input non fidato (anche se è stato filtrato) non dovrebbe mai essere inserito:

- Direttamente in uno script.
- All'interno di un commento HTML.
- Nel nome di un attributo.
- In un nome di tag.
- Direttamente nel CSS.

Ognuna di queste posizioni ha la propria forma di escape e di filtraggio. Poiché molti browser tentano di implementare la protezione XSS, qualsiasi verifica manuale di questa scoperta dovrebbe essere condotta utilizzando più browser e versioni diverse.

Riferimenti

- http://projects.webappsec.org/w/page/13246920/Cross%20Site%20Scripting
- https://www.owasp.org/index.php/XSS_%28Cross_Site_Scripting%29_Prevention_Cheat_Sheet

Unrestricted File Upload				
V-003	SEVERITY	CVSS SCORE	ACCESS COMPLEXITY	IMPACT
	Medium	4.6	Medium	Low
	OWASP CATEGORY		OWASP CONTROL	
	A04 - Insecure Design		WSTG-BUSL-08,WSTG-BUSL-09	
	CVSS 3.1 VECTOR			
	AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:L/A:N/E:H/RL:O/RC:C/CR:L/IR:L/AR:L/MAV:N/MAC:L/MPR:L/MUI:N/MS:U/MC:L/MI:L/MA:L			

Asset

https://staging.municipiumapp.it/api/s3/uploadallegato/8093/?path=8093/allegati

Descrizione

Le vulnerabilità di "Unrestricted File Upload" sorgono quando un'applicazione implementa un meccanismo di controllo inadeguato per la funzionalità di caricamento dei file. In particolare, il sistema non esegue controlli sufficienti sulla tipologia, dimensione e contenuto del file prima di consentirne il caricamento.
Tale carenza di controlli costituisce un problema di sicurezza, poiché potrebbe consentire ad un attaccante di caricare sul server remoto file dannosi per il sistema o per altri utenti, quali ad esempio file eseguibili, script o malware.

Evidenza

Durante l'analisi dell'applicazione, è stata individuata una vulnerabilità di Unrestricted File Upload nella funzionalità dell'API */api/s3/uploadallegato/8092/?path=8092/allegati/contatto/262463/privato*. L'applicazione non esegue adeguati controlli sulle proprietà dei file caricati, tra cui tipologia, dimensione e contenuto, il che potrebbe consentire ad un attaccante di caricare file dannosi sul server remoto.

Si riporta la richiesta effettuata per sfruttare la vulnerabilità:

```
POST /api/s3/uploadallegato/8093/?path=8093/allegati HTTP/2
Host: staging.municipiumapp.it
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: application/json, text/plain, */*
Accept-Language: en-US
Accept-Encoding: gzip, deflate
Referer: https://staging.municipiumapp.it/
X-Requested-With: XMLHttpRequest
Authorization: [Token]
Content-Type: multipart/form-data;
boundary=---------------------------218446534225774722932475565780
Content-Length: 245
Origin: https://staging.municipiumapp.it

---------------------------218446534225774722932475565780
Content-Disposition: form-data; name="file"; filename="js2.html"
Content-Type: text/html
<script>alert(1)</script>
---------------------------218446534225774722932475565780--
```

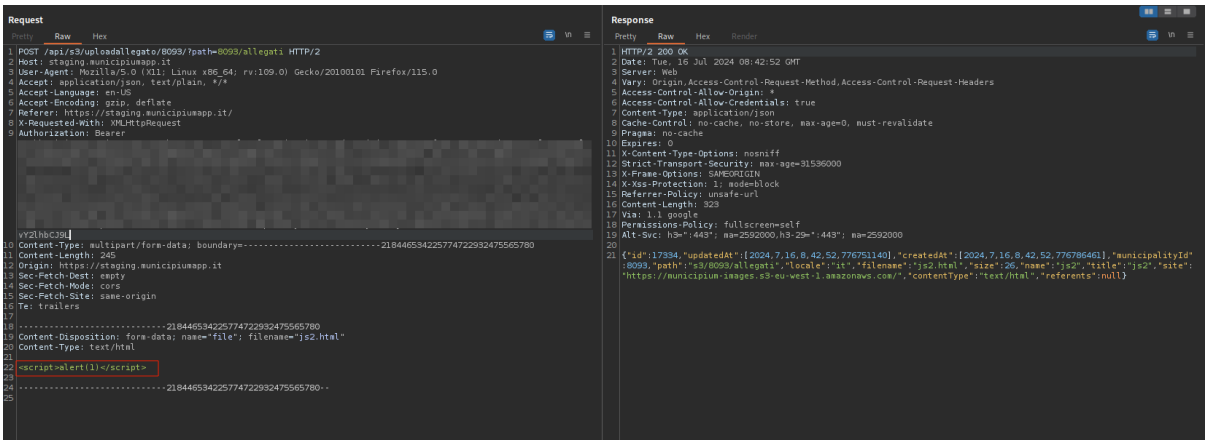

Figura 5-4 - Evidenza caricamento di un file contenente un payload xss

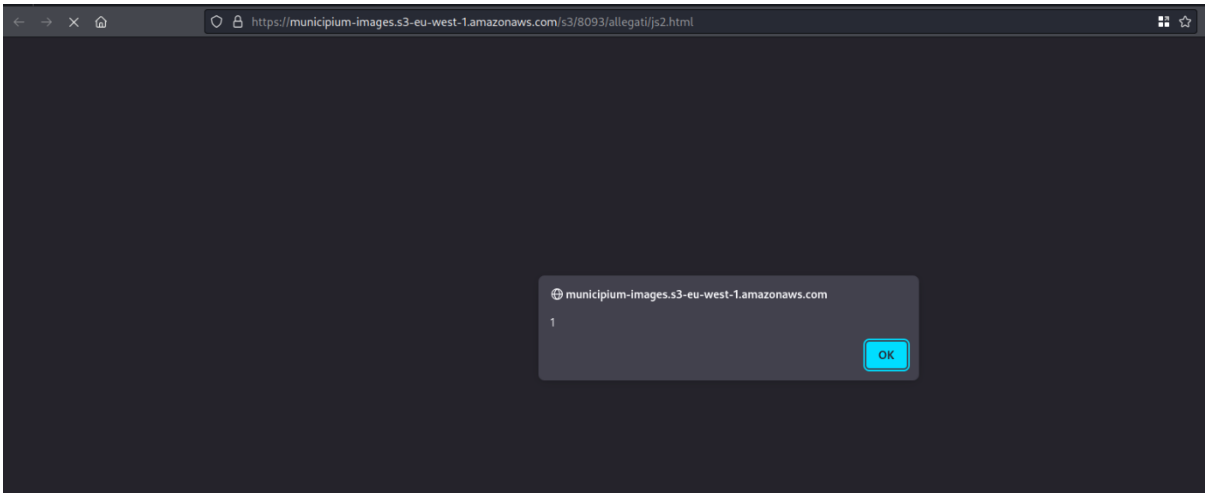

Figura 5-5 - File caricato con successo con esecuzione del codice javascript inserito all'interno

Mitigazione

Si raccomanda di implementare una serie di controlli e restrizioni lato server sulla funzionalità di file-upload, in modo da prevenire il caricamento di file dannosi o indesiderati sul sistema. Queste raccomandazioni comuni includono:

- Implementare un elenco limitato di estensioni accettabili per i file caricabili (whitelist) e assicurarsi che l'applicazione esegua anche un controllo sul contenuto dei file caricati.
- Verificare che i file caricati abbiano un'unica estensione e bloccare tutti i tentativi di caricamento di file con più estensioni (ad esempio "malicious_file.exe.png").
- Limitare l'intervallo di caratteri utilizzabili per comporre il nome del file da caricare (es. whitelist con i soli caratteri alfanumerici "a-zA-zA-Z0-9" e il carattere a punti ".").
- Fissare una lunghezza massima accettabile per il nome del file.
- Implementare controlli sui nomi di file che non sono sensibili alle maiuscole e minuscole, se il server web supporta nomi di file non sensibili alle maiuscole e minuscole.
- Controllare e bloccare i nomi di file incompleti, con la sola estensione (ad esempio ".htaccess"), o senza estensione (ad esempio "malicious_file").
- Assicurarsi che la cartella in cui vengono caricati i file sia configurata in modo da non concedere i permessi per l'esecuzione dei file di cui sopra.
- Rinominare i file che vengono caricati con nomi non facilmente predicibili.
- Salvare i file caricati su un percorso esterno al documento web di root, meglio ancora se in un server dedicato.
- Fissare un limite massimo consentito per la dimensione del file e la quantità di file caricabili per richiesta e implementare i relativi controlli.
- Installare un sistema antivirus sul server remoto per controllare tutti i file caricati nella cartella condivisa, se non è già presente

Riferimenti

- https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload

Insecure Direct Object Reference (IDOR)				
V-004	SEVERITY	CVSS SCORE	ACCESS COMPLEXITY	IMPACT
	Medium	4.3	Low	Medium
	OWASP CATEGORY		OWASP CONTROL	
	A01 - Broken Access Control		WSTG-ATHZ-01,WSTG-ATHZ-02,WSTG-ATHZ-03,WSTG-ATHZ-04	
	CVSS 3.1 VECTOR			
	AV:N/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N/E:P/RL:W/RC:C/CR:L/IR:X/AR:X/MAV:N/MAC:L/MPR:L/MUI:N/MS:U/MC:H/MI:N/MA:N			

Asset

https://staging.municipiumapp.it/api/v1/admin/amministratore/***
https://staging.municipiumapp.it/api/v1/admin/avvisoattachment/?avvisoId=*
https://staging.municipiumapp.it/api/v1/admin/gastonecontact/***

Descrizione

L'Insecure Direct Object Reference (IDOR) è una vulnerabilità di sicurezza appartenente alla categoria Broken Access Control. Essa si manifesta quando un'applicazione rivela direttamente all'utente un riferimento interno a un oggetto, come un ID numerico o un percorso di file. Questa esposizione consente a utenti malintenzionati di accedere a risorse non autorizzate manipolando semplicemente i riferimenti agli oggetti.
Il controllo degli accessi (Access Control) di un applicazione è responsabile di gestire i permessi sui dati a cui un utente può accedere o operazioni che un utente può effettuare. La vulnerabilità Broken Access Control porta alla divulgazione non autorizzata di informazioni, alla modifica/cancellazione di dati o all'esecuzione di una funzione al di fuori dei limiti operativi dell'utente.
Attraverso queste vulnerabilità è possibile, ad esempio:

- Accedere ad informazioni anche potenzialmente sensibili destinate ad un altro utente
- Modificare dati legati ad un altro utente, manipolando parametri che fanno riferimento ad ID o

Evidenza

All'interno dell'applicativo è stato possibile identificare la referenziazione di oggetti tramite ID incrementale.
Attraverso questa vulnerabilità è stato possibile determinare la presenza di diversi elementi, ripetendo la richiesta con ID incremenato (in quanto predicibile).
Dal momento che tale criticità risulta essere presente in molte degli endpoint consultabili all'interno del menù presente sulla parte sinistra dell'applicativo, è doveroso sottolineare che la presente lista non sarà esaustiva. Si consiglia a tal proposito di fare riferimento all'applicativo ed agli URL raggiungibili dai vari menù, per porne rimedio.
Nel dettaglio è stato possibile enumerare:

- le informazioni sul profilo degli altri utenti attraverso l'endpoint *https://staging.municipiumapp.it/api/v1/admin/amministratore/****

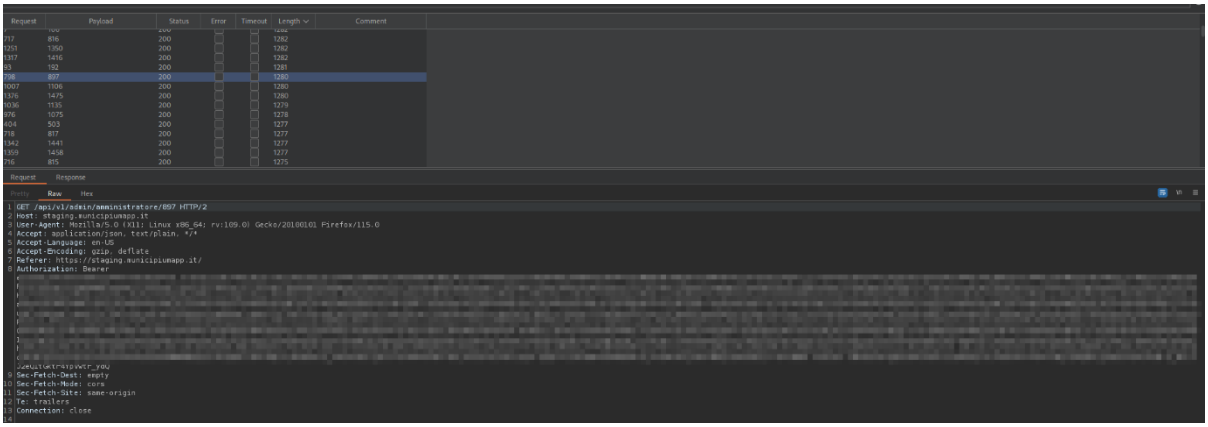

Figura 5-6 - Richiesta HTTP per effettualre l'enumerazione dei profili degli utenti

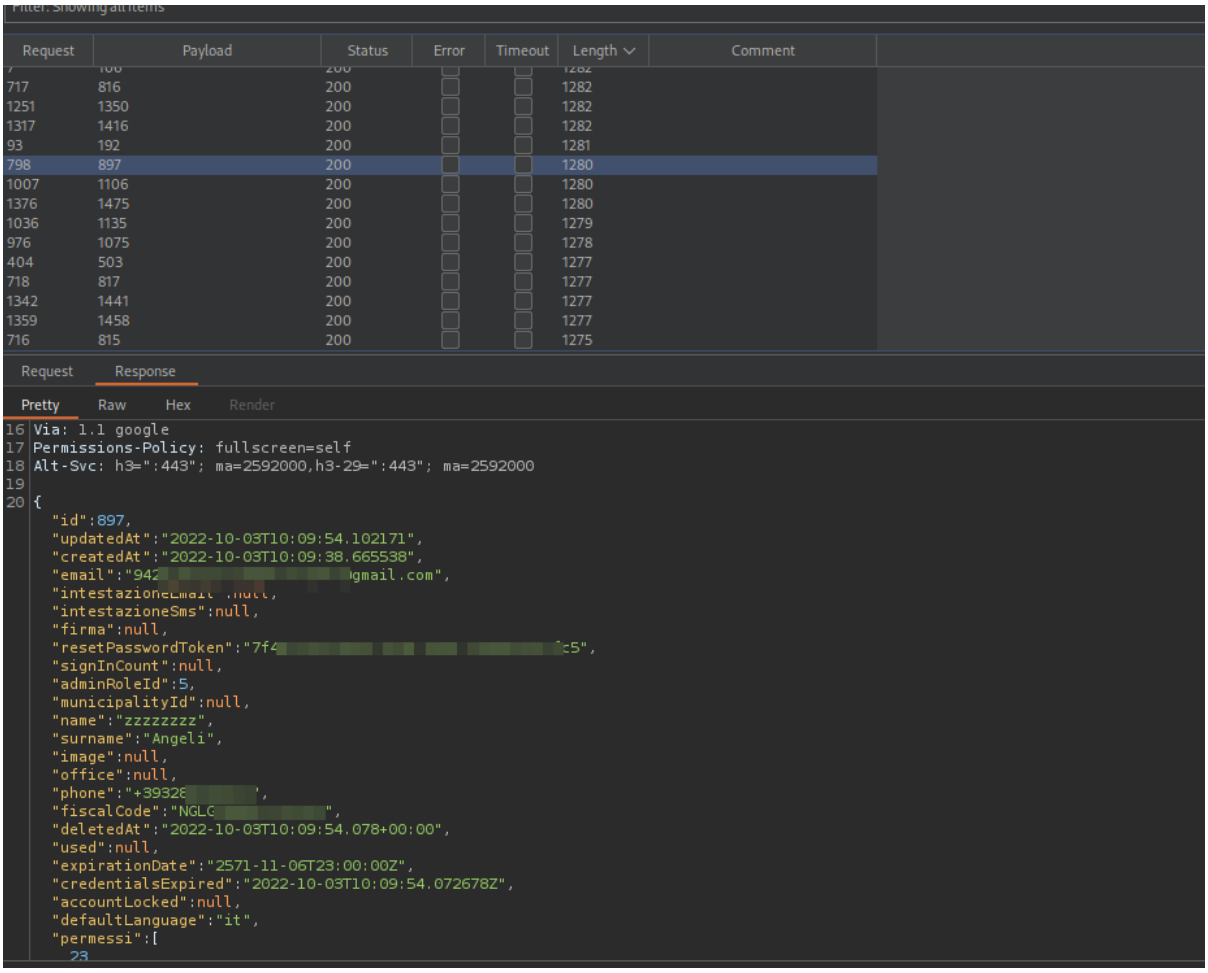

Figura 5-7 - Risposta contenente le informazioni degli utenti

- i contatti attraverso l'endpoint *https://staging.municipiumapp.it/api/v1/admin/gastonecontact/****

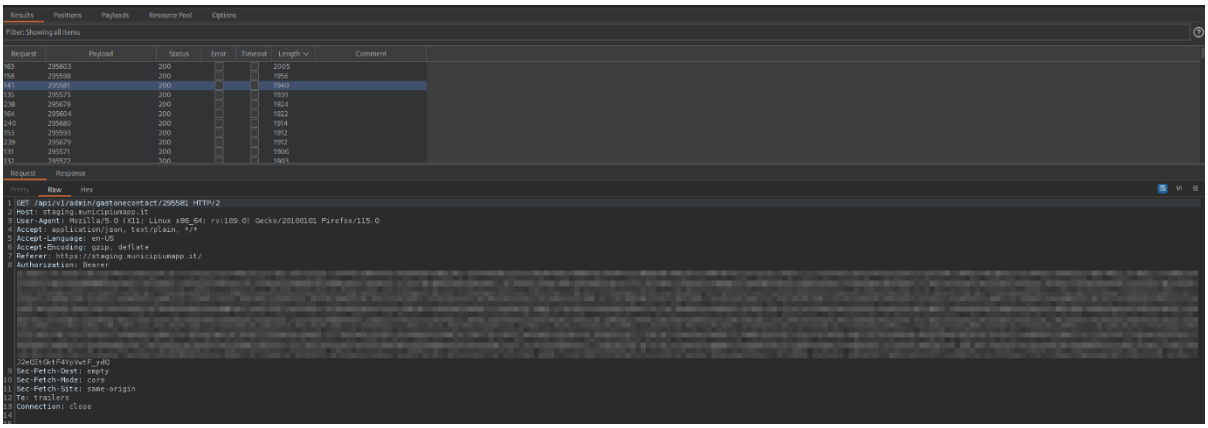

Figura 5-8 - Richiesta HTTP per effettualre l'enumerazione dei contatti

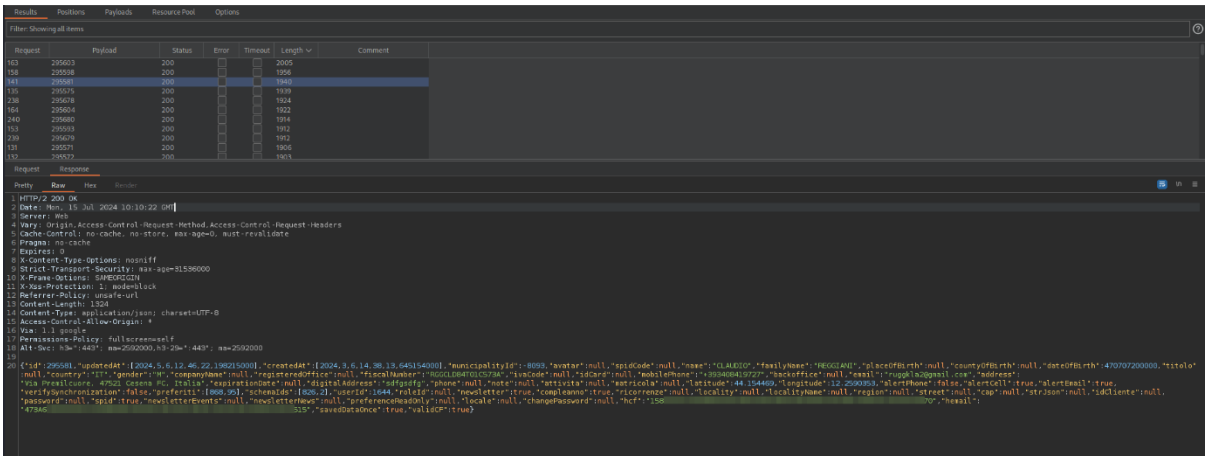

Figura 5-9 - Risposta contenente le informazioni dei contatti

- la lista di alcuni file associati a bucket Amazon s3 attraverso l'endpoint https://staging.municipiumapp.it/api/v1/admin/avvisoattachment/?avvisoId=*

Figura 5-10 - Richiesta HTTP per effettuare l'enumerazione degli avvisi

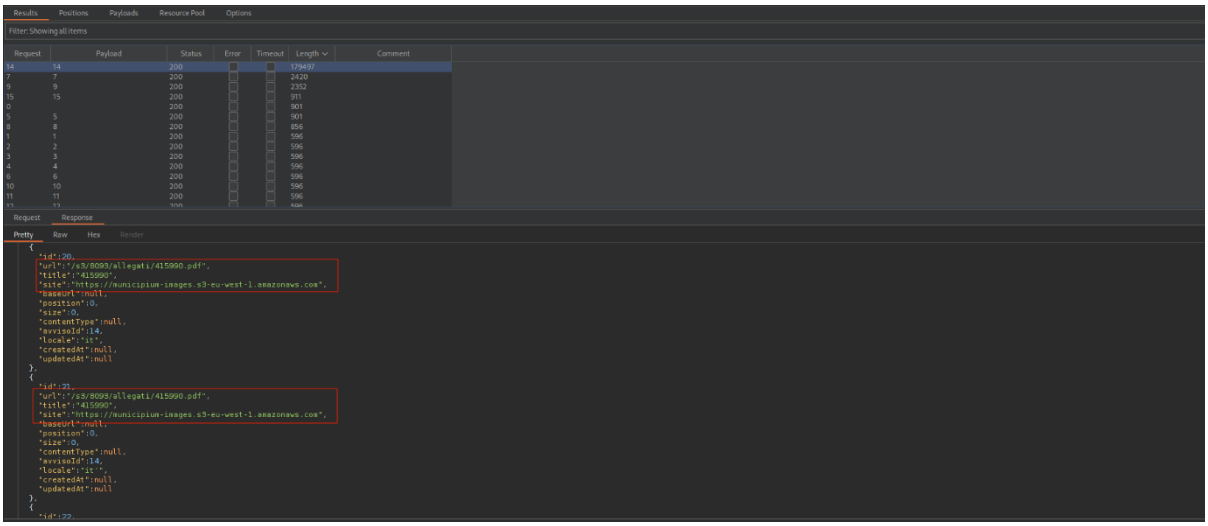

Figura 5-11 - Risposta contente le informazioni di file associate a bucket Amazon s3

Mitigazione

Si consiglia di non utilizzare identificativi facilmente prevedibili (es. numeri in progressione) per referenziare gli oggetti (es. file e/o utenti) all'interno dell'applicativo.

Riferimenti

- https://owasp.org/www-project-top-ten/2017/A5_2017-Broken_Access_Control

Broken Access Control				
V-005	SEVERITY	CVSS SCORE	ACCESS COMPLEXITY	IMPACT
	Medium	4.0	Low	Medium
	OWASP CATEGORY		OWASP CONTROL	
	A01 - Broken Access Control		WSTG-ATHZ-01,WSTG-ATHZ-02,WSTG-ATHZ-03,WSTG-ATHZ-04	
	CVSS 3.1 VECTOR			
	AV:N/AC:L/PR:L/UI:N/S:U/C:L/I:N/A:N/E:P/RL:W/RC:C/CR:M/IR:X/AR:X/MAV:N/MAC:L/MPR:L/MUI:N/MS:U/MC:L/MI:N/MA:N			

Asset

https://staging.municipiumapp.it/#/crm/admins/8093

Descrizione

Il controllo degli accessi di un'applicazione è responsabile di gestire i permessi sui dati a cui un utente può accedere o operazioni che un utente può effettuare. La vulnerabilità Broken Access Control porta alla divulgazione non autorizzata di informazioni, alla modifica/cancellazione di dati o all'esecuzione di una funzione al di fuori dei limiti operativi dell'utente.
Attraverso queste vulnerabilità è possibile, ad esempio:

- Accedere ad informazioni anche potenzialmente sensibili destinate ad un altro utente;
- Modificare dati legati ad un altro utente, manipolando parametri che fanno riferimento ad ID o nomi utente;
- Cancellare file fuori dalla propria competenza, sia destinati ad altri utenti, sia fuori dal contesto applicativo.

Evidenza

Durante l'attività eseguita è stato possibile notare che l'applicativo Municipium Cloud Backoffice risulta vulnerabile a Broken Access Control.
Tramite questa problematica, un utente normale è in grado di interagire con le risorse dell'utente Amministratore.
Nel dettaglio, un utente base può richiamare con successo delle API che dovrebbero essere a sola visibilità dell'Amministratore.
NB: Le informazioni sul profilo degli altri utenti possono essere enumerate attraverso l'endpoint "https://staging.municipiumapp.it//api/v1/admin/amministratore/001" come descritto all'interno del paragrafo relativo alla vulnerabilità Insecure Drect Object Reference (IDOR).

L'utente base Claudio Yarix può richiamare la funzionalità Administrators list che dovrebbe essere visibile solo all'utente Yarix Test (utente admin).

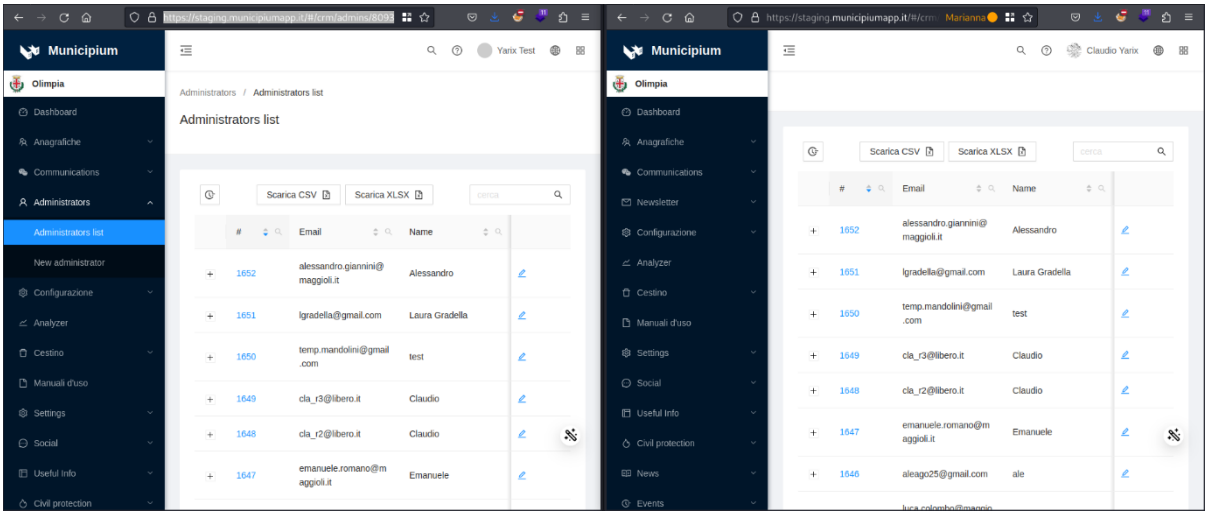

Figura 5-12 - Vista dell'interfaccia amministrativa richiamata da utente con privilegi normali - l'opzione Administrators -> Administrative list

Inoltre, l'utente Claudio Yarix (cla_r2@libero.it) può visualizzare la sezione Attachment previa conoscenza dell'endopoint.

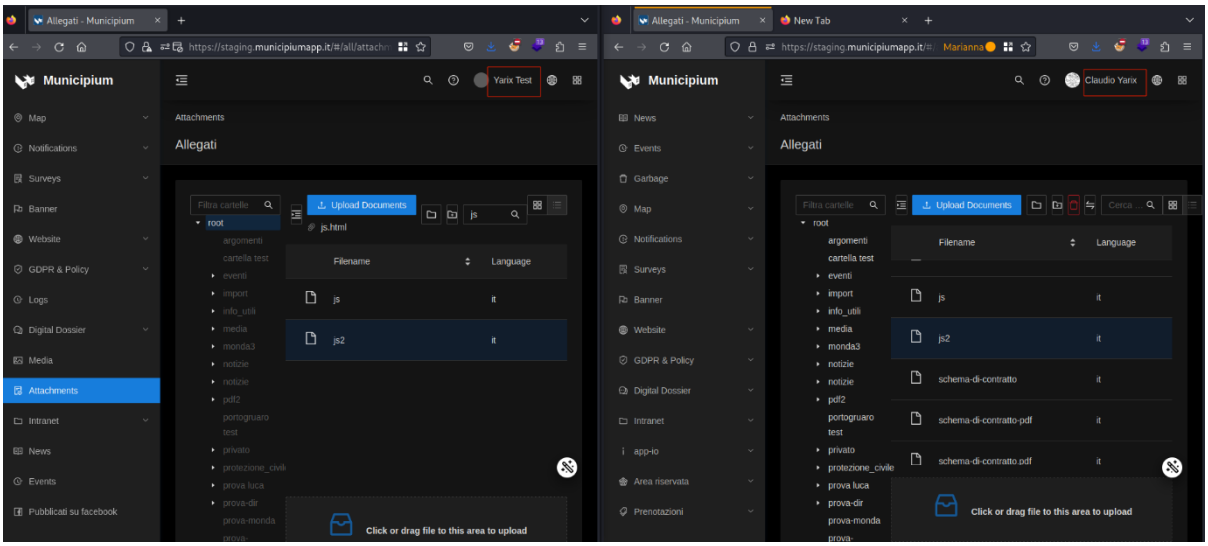

Figura 5-13 - Vista della sezione Attachment da utente base con ruolo Author

Mitigazione

Assicurarsi di gestire le autorizzazioni che regolano l'accesso ai servizi in modo che il potenziale attaccante non possa modificarne il comportamento.
È necessario implementare codice che non possa essere manipolato da eventuali input gestibili dall'utente.
Le linee guida consigliano di:

- Negare l'accesso ad un servizio o ad una funzionalità di default. Questo approccio whitelist risulta molto comodo soprattutto per la gestione dei permessi nel tempo.
- Utilizzare meccanismo basati su ruoli.
- Non limitarsi a nascondere servizi o funzionalità: un attaccante con sufficiente tempo e motivazione potrebbe riuscire a risalire ad eventuali endpoint utilizzando tecniche di fuzzing ed enumeration.

Riferimenti

- https://owasp.org/Top10/A01_2021-Broken_Access_Control/

Information Disclosure				
V-006	SEVERITY	CVSS SCORE	ACCESS COMPLEXITY	IMPACT
	Low	3.8	Low	Low
	OWASP CATEGORY		OWASP CONTROL	
	A05 - Security Misconfiguration		WSTG-INFO-02,WSTG-INFO-05,WSTG-INFO-08,WSTG-INFO-09,WSTG-ERRH-01,WSTG-ERRH-02	
	CVSS 3.1 VECTOR			
	AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N/E:P/RL:O/RC:U/CR:L/IR:X/AR:X/MAV:N/MAC:L/MPR:N/MUI:N/MS:U/MC:X/MI:X/MA:X			

Asset

https://staging.municipiumapp.it/assets
https://staging.municipiumapp.it/api/v1/admin/gastonetask/anteprima/

Descrizione

Information Disclosure è una criticità delle applicazioni che consiste nella esposizione di dati sensibili come dettagli tecnici dell'applicazione web, informazioni sull'ambiente in utilizzo e sui dati dell'utente. Questa fuga di informazioni può fornire indizi su eventuali vulnerabilità note, aumentando così la possibilità di attacchi successivi.
Le cause più comuni di Information Disclosure includono la presenza di commenti HTML o script contenenti informazioni riservate, configurazioni inappropriate dell'applicazione o del server e una gestione scorretta degli errori dell'applicazione.

Evidenza

Durante i test è stato possibile identificare:

- la versione del web server nginx/1.26.1
- l'esposizione di risorse interne dell'infrastruttura.

È stato inoltre possibile, attraverso degli errori applicativi, risalire ad alcune informazioni tecniche riguardo l'applicativo, ad esempio la tipologia di framework con cui è stato scritto l'applicativo.

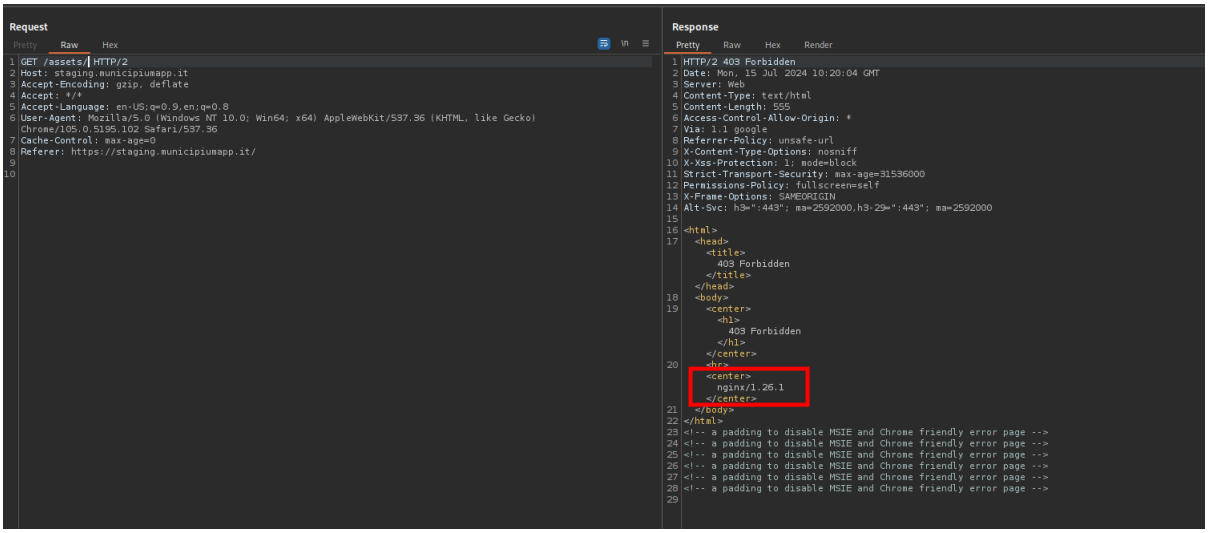

Figura 5-14 - Versione di nginx

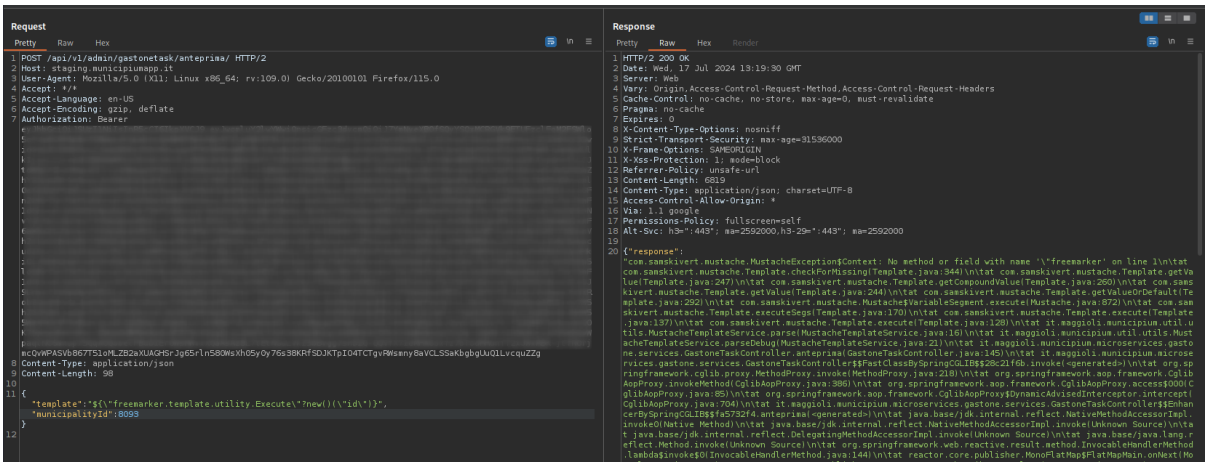

Figura 5-15 - Errori applicativi prolissi

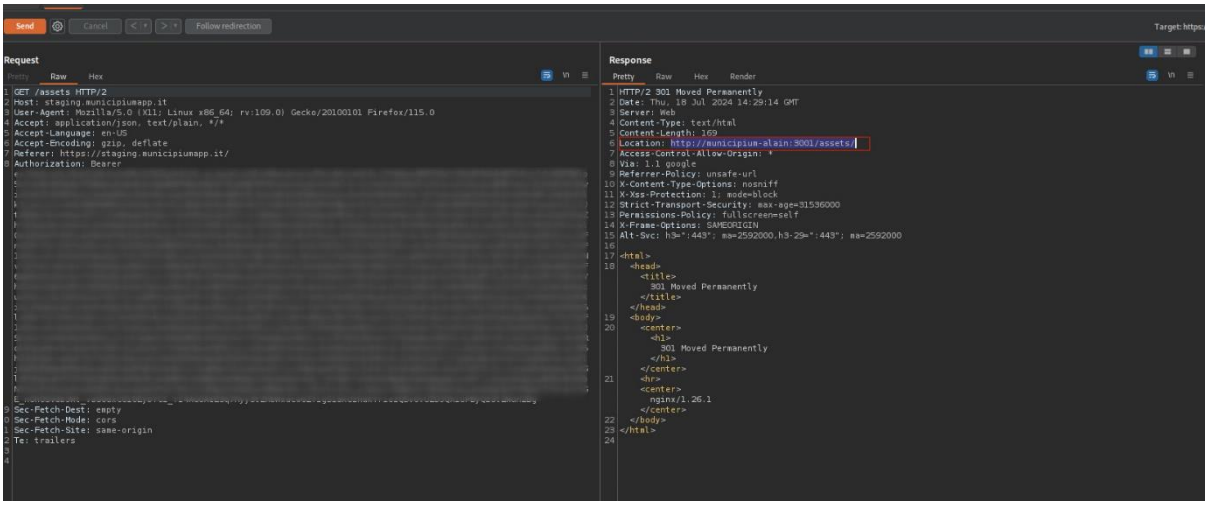

Figura 5-16 - Information disclosure di dati interni all'infrastruttura

Mitigazione

Assicurarsi di implementare pratiche e controlli di sicurezza adeguati per prevenire la fuoriuscita di informazioni sensibili. Di seguito alcune azioni che possono essere intraprese per correggere questa vulnerabilità:

- Rimuovere o mascherare tutti i commenti HTML o script che contengono informazioni sensibili;
- Configurare correttamente l'applicazione e il server per impedire la fuoriuscita di informazioni;
- Gestire correttamente gli errori dell'applicazione, in modo che non vengano esposte informazioni riservate;
- Verificare che i dati sensibili vengano trattati in modo sicuro e che non vengano registrati nei log di sistema;
- Implementare pratiche di sicurezza rigorose, come la crittografia dei dati sensibili e la limitazione dell'accesso solo ai soggetti autorizzati.

Riferimenti

- http://projects.webappsec.org/w/page/13246936/Information%20Leakage

Improper Session Handling				
V-007	SEVERITY	CVSS SCORE	ACCESS COMPLEXITY	IMPACT
	Low	3.2	Medium	Medium
	OWASP CATEGORY		OWASP CONTROL	
	A07 - Identification and Authentication Failures		WSTG-SESS-01	
	CVSS 3.1 VECTOR			
	AV:N/AC:L/PR:L/UI:R/S:U/C:L/I:N/A:N/E:P/RL:O/RC:C/CR:X/IR:X/AR:X/MAV:N/MAC:L/MPR:L/MUI:R/MS:U/MC:L/MI:X/MA:X			

Asset

https://staging.municipiumapp.it/#/passport/login

Descrizione

Questo tipo di problematica indica un'errata o non ottimale gestione della sessione che identifica l'utente, come ad esempio:

- Mancato utilizzo di un valore casuale crittograficamente forte come identificatore di sessione
- Mancata tutela della riservatezza del cookie identificativo di sessione.
- Mancato rinnovo del cookie di sessione al login
- Mancata invalidazione della sessione al logout
- Mancata chiusura automatica della sessione sul server dopo un periodo di inattività predefinito.
- Mancata invalidazione della sessione dopo la chiusura del browser

Evidenza

Durante le verifiche sulla sessione dell'utente, è stato possibile determinare che a seguito dell'operazione di logout, la riproduzione di chiamate API tramite lo stesso bearer continuano ad andare a buon fine.

Mitigazione

Utilizzare lato server un sistema che si occupi di generare nuovi ID di sessione ad alta entropia dopo il login dell'utente.
La sessione non deve avere riferimenti presenti nell'URL o in altri parametri considerati insicuri.
L'identificatore di sessione deve essere archiviato in modo sicuro e invalidato correttamente dopo il logout, la chiusura del browser o un tempo di inattività stabilito.

Riferimenti

- https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/

5.2 Altre evidenze

I seguenti sotto paragrafi elencano ulteriori rilevamenti interessanti, non riconducibili direttamente a scenari di attacco completi e/o vulnerabilità.

5.2.1 Security Headers Misconfiguration

I Security Header fanno riferimento ad un sottoinsieme di header HTTP che si ricevono in risposta dal server a seguito delle richieste di pagine del server web da parte di un browser. Il loro scopo è quello di indicare al browser come comportarsi in ambito di gestione di contenuto del sito web. La loro corretta implementazione, attraverso l'utilizzo delle migliori best practices, può portare al miglioramento della postura di sicurezza di una applicazione web, proteggendo essa da alcune vulnerabilità e attacchi conosciuti.
Non è correttamente settato il seguente headers:
Content-Security-Policy
Si tratta di un response header che permette di definire una serie di policy che identificano quali risorse il browser ha il permesso di caricare, eseguire o contattare. Viene utilizzato per prevenire attacchi di tipo XSS, e altri attacchi di code injection ed è ampiamente supportato dai browser più utilizzati.

```
kali in ~/Desktop/script_utili/testssl.sh λ shcheck.py https://staging.municipiumapp.it/

 > shcheck.py - santoru ..............................
 Simple tool to check security headers on a webserver

[*] Analyzing headers of https://staging.municipiumapp.it/
[*] Effective URL: https://staging.municipiumapp.it/
[*] Header X-XSS-Protection is present! (Value: 1; mode=block)
[*] Header X-Frame-Options is present! (Value: SAMEORIGIN)
[*] Header X-Content-Type-Options is present! (Value: nosniff)
[*] Header Strict-Transport-Security is present! (Value: max-age=31536000)
[!] Missing security header: Content-Security-Policy
[*] Header Referrer-Policy is present! (Value: unsafe-url)
[!] Missing security header: Expect-CT
[*] Header Permissions-Policy is present! (Value: fullscreen=self)
[!] Missing security header: Cross-Origin-Embedder-Policy
[!] Missing security header: Cross-Origin-Resource-Policy
[!] Missing security header: Cross-Origin-Opener-Policy
[!] Headers analyzed for https://staging.municipiumapp.it/
[+] There are 6 security headers
[-] There are not 5 security headers
```

Figura 5-17 - Headers di sicurezza mancanti

Si consiglia di configurare il server web o l'applicativo per restituire gli header indicati. Si prega di consultare il riferimento per ogni header specifico.