



ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	ΑΥΤΟΜΑΤΟΠΟΙΗΜΕΝΗ ΑΝΑΚΑΛΥΨΗ ΧΡΗΣΤΩΝ ΣΤΟ WORDPRESS AUTOMATED USER DISCOVERY IN WORDPRESS
Ονοματεπώνυμο Φοιτητή	Νικόλαος Μπετίνης
Πατρώνυμο	Κωνσταντίνος
Αριθμός Μητρώου	ΜΠΠΛ/ 20052
Επιβλέπων	Κωνσταντίνος Πατσάκης, Καθηγητής

Ημερομηνία Παράδοσης **Ιούνιος 2026**

Τριμελής Εξεταστική Επιτροπή

Κωνσταντίνος Πατσάκης
Καθηγητής

Ευθύμιος Αλέπης
Καθηγητής

Παναγιώτης Κοτζανικολάου
Καθηγητής

Περίληψη

Η ευρεία υιοθέτηση των Συστημάτων Διαχείρισης Περιεχομένου (Content Management Systems – CMS) έχει καταστήσει τις πλατφόρμες αυτές βασικό πυλώνα της σύγχρονης διαδικτυακής παρουσίας, αλλά και ελκυστικό στόχο για επιθέσεις στον κυβερνοχώρο. Μεταξύ των συχνότερων και συχνά υποτιμημένων απειλών συγκαταλέγεται η απαρίθμηση χρηστών (user enumeration), η οποία επιτρέπει την αναγνώριση έγκυρων λογαριασμών και λειτουργεί ως αρχικό στάδιο για πιο σύνθετες επιθέσεις, όπως brute force και credential stuffing. Η παρούσα εργασία επικεντρώνεται στο WordPress, το δημοφιλέστερο CMS παγκοσμίως, εξετάζοντας τόσο μία παλαιότερη έκδοση (WordPress 4.7.1) όσο και μία σύγχρονη έκδοση (WordPress 6.9), με στόχο την ανάλυση και σύγκριση των διαθέσιμων μηχανισμών απαρίθμησης χρηστών.

Για την επίτευξη του στόχου αναπτύχθηκε ελεγχόμενο εργαστηριακό περιβάλλον εικονικών μηχανών, στο οποίο υλοποιήθηκαν διαφορετικοί ρόλοι χρηστών και πραγματοποιήθηκαν συστηματικές δοκιμές μέσω REST API endpoints, μεταδεδομένων περιεχομένου, μηνυμάτων αυθεντικοποίησης, HTML στοιχείων συγγραφέα και παραμέτρων URL. Τα αποτελέσματα καταδεικνύουν ότι, παρά τις βελτιώσεις στις νεότερες εκδόσεις, η απαρίθμηση χρηστών εξακολουθεί να είναι εφικτή μέσω άμεσων ή έμμεσων μηχανισμών, γεγονός που αποδίδεται σε συνδυασμό σχεδιαστικών επιλογών σε διαφορετικά επίπεδα της αρχιτεκτονικής του συστήματος. Στο πλαίσιο της μελέτης εφαρμόστηκαν τεχνικές μετριασμού τόσο στο επίπεδο του web server (Nginx) όσο και στο επίπεδο της εφαρμογής WordPress, οι οποίες αξιολογήθηκαν πειραματικά και οδήγησαν σε ουσιαστική μείωση της επιφάνειας επίθεσης, χωρίς αρνητικές επιπτώσεις στη λειτουργικότητα του συστήματος.

Abstract

The widespread adoption of Content Management Systems (CMS) has made these platforms a fundamental component of modern web presence, while also turning them into attractive targets for cyberattacks. Among the most common and often underestimated threats is user enumeration, which enables the identification of valid user accounts and serves as a preparatory step for more advanced attacks such as brute force and credential stuffing. This thesis focuses on WordPress, the most widely used CMS worldwide, examining both an older version (WordPress 4.7.1) and a modern version (WordPress 6.9), with the aim of analyzing and comparing the available user enumeration vectors.

To achieve this goal, a controlled laboratory environment based on virtual machines was developed, in which multiple user roles were implemented and systematic tests were conducted through REST API endpoints, content metadata, authentication error messages, HTML author attribution, and URL parameters. The results demonstrate that, despite the security improvements introduced in newer versions, user enumeration remains possible through direct or indirect mechanisms, a fact attributed to a combination of design choices at different architectural layers of the system. Within the scope of this study, mitigation techniques were applied at both the web server level (Nginx) and the WordPress application level. These techniques were experimentally evaluated and led to a substantial reduction of the attack surface without negatively affecting system functionality.

Κεφάλαιο 1 — Εισαγωγή

1.1 Εξάπλωση των CMS και Πλεονεκτήματα Χρήσης

Τα Συστήματα Διαχείρισης Περιεχομένου (Content Management Systems – CMS) αποτελούν σήμερα μία από τις πιο διαδεδομένες τεχνολογικές λύσεις για τη δημιουργία, τη διαχείριση και τη συντήρηση ιστοσελίδων. Η διάδοσή τους οφείλεται κυρίως στο γεγονός ότι επιτρέπουν σε χρήστες χωρίς εξειδικευμένες γνώσεις προγραμματισμού να διαχειρίζονται περιεχόμενο μέσω γραφικών διεπαφών (web-based interfaces), μειώνοντας σημαντικά τον χρόνο ανάπτυξης και το κόστος υλοποίησης.¹¹

Η χρήση CMS προσφέρει σημαντικά πλεονεκτήματα τόσο σε επίπεδο λειτουργικότητας όσο και σε επίπεδο διαχείρισης. Ενδεικτικά, τα CMS επιτρέπουν:

- Ταχεία ανάπτυξη ιστοσελίδων με επαγγελματική εμφάνιση, χωρίς αρχικό κόστος (ανοικτός κώδικας) και την ανάγκη ανάπτυξης από μηδενική βάση.
- Ευκολία διαχείρισης περιεχομένου, με δυνατότητα προσθήκης/τροποποίησης σελίδων και άρθρων από χρήστες χωρίς ιδιαίτερες τεχνικές γνώσεις.
- Διαχωρισμό περιεχομένου και σχεδίασης, μέσω themes και templating μηχανισμών.
- Επεκτασιμότητα μέσω πρόσθετων (plugins), τα οποία προσθέτουν λειτουργίες όπως e-commerce, φόρμες, SEO, στατιστικά, κ.ά.
- Σύστημα ρόλων και δικαιωμάτων, το οποίο επιτρέπει την οριοθέτηση αρμοδιοτήτων (π.χ. διαχειριστές, συντάκτες, απλοί χρήστες).

Content Management Systems

Most popular content management systems

© W3Techs.com	usage	change since 1 December 2025	market share	change since 1 December 2025
1. WordPress	42.8%	-0.4%	60.0%	-0.4%
2. Shopify	5.1%	+0.2%	7.1%	+0.3%
3. Wix	4.2%	+0.1%	5.9%	+0.1%
4. Squarespace	2.4%		3.4%	
5. Joomla	1.3%	-0.1%	1.8%	-0.1%

percentages of sites

Fastest growing content management systems since 1 December 2025

© W3Techs.com	sites
1. Shopify	37.7
2. Wix	14.4
3. Elementor	5.9

daily increase of number of sites
per million

Εικόνα 1. Στατιστικά CMS

Η σημασία και η καθολική διάδοση των CMS επιβεβαιώνεται και από τα στατιστικά χρήσης τεχνολογιών στον Παγκόσμιο Ιστό. Σύμφωνα με τα δεδομένα του w3techs, ένα σημαντικό ποσοστό ιστοσελίδων, είτε χρησιμοποιεί κάποιο CMS, είτε ανήκει σε κατηγορία ιστοσελίδων όπου το CMS δεν μπορεί να προσδιοριστεί. Συγκεκριμένα, το w3techs αναφέρει ότι το 28,7% των ιστοσελίδων δεν

χρησιμοποιεί κάποιο από τα CMS που παρακολουθεί, ενώ το Wordpress χρησιμοποιείται από το 42,8% όλων των ιστοσελίδων, αντιπροσωπεύοντας το 60,0% της αγοράς CMS.¹

Παράλληλα, το w3techs κατατάσσει ως επόμενες δημοφιλείς πλατφόρμες CMS τις:

- Shopify (~5,1% όλων των ιστοσελίδων),
- Wix (~4,2%),
- Squarespace (~2,4%),

ενώ άλλες πιο “παραδοσιακές” πλατφόρμες όπως Joomla και Drupal εμφανίζουν σημαντικά χαμηλότερα ποσοστά χρήσης.

Η κυριαρχία του WordPress είναι εμφανής όχι μόνο ως προς τη συνολική του παρουσία στον Ιστό, αλλά και ως προς τη διάδοση των σύγχρονων εκδόσεών του. Σύμφωνα με το w3techs, η έκδοση 6 χρησιμοποιείται από το 91,2% των ιστοσελίδων που χρησιμοποιούν WordPress, γεγονός που υποδεικνύει ότι ένα μεγάλο μέρος του οικοσυστήματος έχει υιοθετήσει τις πιο πρόσφατες γενιές του συστήματος.¹

Η ευρεία χρήση των CMS, και ειδικότερα του WordPress, συνεπάγεται ότι αποτελούν το Νο1 στόχο για κυβερνοεπιθέσεις. Η υψηλή “επιφάνεια επίθεσης” (attack surface) ενός CMS οφείλεται όχι μόνο στον πυρήνα της πλατφόρμας (core), αλλά και στο εκτεταμένο οικοσύστημα θεμάτων και προσθέτων (themes and plugins) που χρησιμοποιεί ένα wordpress site, καθώς και στις προεπιλεγμένες λειτουργίες που είναι συχνά ενεργοποιημένες. Ως αποτέλεσμα, ακόμη και μικρές αδυναμίες αποκάλυψης πληροφοριών, όπως η απαρίθμηση χρηστών (**user enumeration**), μπορούν να αξιοποιηθούν μέσω αυτοματοποιημένων επιθέσεων.¹⁵

1.2 Σύνδεση των CMS με την Ασφάλεια και η Έννοια της Απαρίθμησης Χρηστών (User Enumeration)

Σε εφαρμογές που βασίζονται σε μηχανισμούς σύνδεσης και λογαριασμούς χρηστών, ένα από τα πιο σημαντικά βήματα για έναν επιτιθέμενο είναι ο εντοπισμός έγκυρων χρηστών του συστήματος. Η διαδικασία αυτή είναι γνωστή ως απαρίθμηση χρηστών (**user enumeration**) και συνιστά μία μορφή αποκάλυψης πληροφοριών (information disclosure). Ένας επιτιθέμενος μπορεί να εκμεταλλευτεί διαφορές στις αποκρίσεις ενός συστήματος (π.χ. διαφορετικά μηνύματα σφάλματος, redirects, δημόσια endpoints ή μεταδεδομένα σελίδων) προκειμένου να συμπεράνει, εάν ένας χρήστης υπάρχει.⁵

Η απαρίθμηση χρηστών θεωρείται σημαντική διότι δεν αποτελεί απλώς μια “ανεπαίσθητη” διαρροή πληροφοριών, αλλά δύναται να λειτουργήσει ως πολλαπλασιαστής αποτελεσματικότητας για επιθέσεις που ακολουθούν, όπως:

- password spraying,
- επιθέσεις brute force,
- στοχευμένο phishing,
- credential stuffing (επιβεβαίωση υπαρκτών λογαριασμών).

Σε CMS όπως το WordPress, η απαρίθμηση χρηστών μπορεί να προκύψει μέσω λειτουργιών που είναι απαραίτητες για την καθημερινή χρήση της πλατφόρμας, όπως οι σελίδες συγγραφέων (author archives), τα REST APIs, οι σελίδες σύνδεσης, ή άλλα endpoints (π.χ. XML-RPC σε παλαιότερες εκδόσεις).²

1.3 Διατύπωση Προβλήματος

Παρότι το WordPress έχει ενσωματώσει διάφορες βελτιώσεις ασφάλειας με την πάροδο του χρόνου, παλαιότερες λειτουργίες και προεπιλεγμένα χαρακτηριστικά ενδέχεται να συνεχίσουν να δημιουργούν κινδύνους απαρίθμησης χρηστών. Επιπλέον, η πραγματική έκθεση σε μηχανισμούς απαρίθμησης μπορεί να εξαρτάται από την έκδοση της πλατφόρμας, τις παραμέτρους ρύθμισης και τη δημόσια προβολή του περιεχομένου. Αυτό εγείρει το ερώτημα κατά πόσο οι σύγχρονες εκδόσεις του WordPress περιορίζουν ουσιαστικά τις δυνατότητες απαρίθμησης σε σχέση με παλαιότερες εκδόσεις, καθώς και κατά πόσο οι τεχνικές μετριασμού είναι αποτελεσματικές σε ρεαλιστικά σενάρια εγκατάστασης.¹

Η παρούσα εργασία εξετάζει το ακόλουθο πρόβλημα: **οι εγκαταστάσεις WordPress ενδέχεται, χωρίς πρόθεση, να επιτρέπουν σε μη αυθεντικοποιημένους επιτιθέμενους να εντοπίζουν έγκυρους λογαριασμούς χρηστών μέσω πολλαπλών μηχανισμών απαρίθμησης, αυξάνοντας την πιθανότητα και την αποτελεσματικότητα επιθέσεων που ακολουθούν.**¹

1.4 Στόχοι Έρευνας

Ο κύριος στόχος της διπλωματικής εργασίας είναι η ανάλυση και η σύγκριση της συμπεριφοράς απαρίθμησης χρηστών σε δύο εκδόσεις WordPress υπό ελεγχόμενες συνθήκες. Οι επιμέρους στόχοι περιλαμβάνουν:

1. Εντοπισμός και δοκιμή μηχανισμών απαρίθμησης χρηστών σε εγκαταστάσεις WordPress, όπως:
 - απαρίθμηση μέσω author archive,
 - διαρροή δεδομένων μέσω μεταδεδομένων και δημοσιευμένου περιεχομένου,
 - συμπεριφορά REST API,
 - διαφορές στη διαδικασία σύνδεσης (login feedback),
 - έκθεση της διεπαφής XML-RPC.
2. Σύγκριση παλαιότερης και σύγχρονης έκδοσης WordPress για να προσδιοριστεί:
 - ποιοι μηχανισμοί υπάρχουν σε κάθε έκδοση,
 - πόσα και ποια στοιχεία χρήστη διαρρέουν,
 - εάν η απαρίθμηση εξαρτάται από το ρόλο ή το δημόσιο περιεχόμενο.
3. Αξιολόγηση τεχνικών μετριασμού (mitigation) και προσδιορισμός κατά πόσο:
 - αποτρέπουν πλήρως την απαρίθμηση,
 - μειώνουν την αποτελεσματικότητα/ταχύτητα της απαρίθμησης,
 - ή απλώς αποκρύπτουν συμπτώματα αφήνοντας έμμεση διαρροή.
4. Παροχή πρακτικών συστάσεων προς διαχειριστές συστημάτων και προγραμματιστές για τη μείωση του κινδύνου απαρίθμησης χρηστών σε περιβάλλοντα WordPress.

1.5 Έκταση και Περιορισμοί

Η έρευνα πραγματοποιείται σε ελεγχόμενο εικονικό εργαστηριακό περιβάλλον, ώστε να εξασφαλιστεί η δυνατότητα επανάληψης των δοκιμών. Η έκταση της εργασίας περιορίζεται στην ανάλυση μη **αυθεντικοποιημένης απαρίθμησης χρηστών** σε εγκαταστάσεις WordPress και δεν περιλαμβάνει περαιτέρω εκμετάλλευση πέρα από την ανακάλυψη λογαριασμών (π.χ. cracking κωδικών ή κλιμάκωση προνομίων). Επιπλέον, η εργασία επικεντρώνεται στη συμπεριφορά του WordPress πυρήνα (core) και δεν αξιολογεί την επίδραση πρόσθετων ασφαλείας (security plugins), καθώς η λειτουργικότητα των plugins διαφέρει σημαντικά και μπορεί να εισάγει πρόσθετες μεταβλητές.

Το εργαστήριο χρησιμοποιεί HTTP αντί για HTTPS. Παρότι η κρυπτογράφηση επηρεάζει την εμπιστευτικότητα της επικοινωνίας σε πραγματικές εγκαταστάσεις, ο κύριος στόχος της παρούσας εργασίας είναι η ανάλυση της αποκάλυψης πληροφοριών σε επίπεδο εφαρμογής (endpoints, απαντήσεις, μεταδεδομένα), η οποία παραμένει σημαντική ανεξάρτητα από τη χρήση κρυπτογραφημένου καναλιού.^{9,10}

1.6 Δομή της Διπλωματικής Εργασίας

Η διπλωματική εργασία οργανώνεται ως εξής:

- Κεφάλαιο 1: Παρουσιάζει το θέμα, το κίνητρο, τους στόχους και τα ερευνητικά ερωτήματα.
- Κεφάλαιο 2: Παρουσιάζει το τεχνικό υπόβαθρο σχετικά με την αρχιτεκτονική του WordPress, τους ρόλους χρηστών και τις βασικές έννοιες που σχετίζονται με την απαρίθμηση χρηστών.
- Κεφάλαιο 3: Περιγράφει την εργαστηριακή υποδομή και τις λεπτομέρειες υλοποίησης.
- Κεφάλαιο 4: Παρουσιάζει και αναλύει τα πειραματικά αποτελέσματα για κάθε μηχανισμό απαρίθμησης και για τις δύο εκδόσεις WordPress.
- Κεφάλαιο 5: Αξιολογεί τεχνικές μετριάσμού και συγκρίνει την έκθεση πριν και μετά την εφαρμογή μέτρων σκλήρυνσης.
- Κεφάλαιο 6: Παραθέτει τα συμπεράσματα και προτείνει μελλοντική κατεύθυνση έρευνας.
- Βιβλιογραφία

Κεφάλαιο 2 — Υπόβαθρο και Θεωρητικό Πλαίσιο

2.1 Επισκόπηση WordPress

Το WordPress αποτελεί μία από τις πιο διαδεδομένες πλατφόρμες CMS και χρησιμοποιείται σε πολλές κατηγορίες ιστοσελίδων, όπως ιστολόγια, εταιρικούς ιστότοπους και εφαρμογές ηλεκτρονικού εμπορίου. Είναι υλοποιημένο σε PHP και συνήθως εγκαθίσταται μαζί με έναν web server (π.χ. Nginx ή Apache) και μία σχεσιακή βάση δεδομένων (π.χ. MySQL ή MariaDB).^{1,7,8}

Από αρχιτεκτονική άποψη, το WordPress περιλαμβάνει τα εξής βασικά στοιχεία:

- **Πυρήνας (Core):** Το κεντρικό σύνολο κώδικα του WordPress που διαχειρίζεται τη δρομολόγηση (routing), την αυθεντικοποίηση, την απόδοση σελίδων (rendering) και τη διαχείριση περιεχομένου.
- **Θέματα (Themes):** Πρότυπα παρουσίασης που καθορίζουν τη δομή, το layout και την εμφάνιση των σελίδων. Εγκαθίστανται στο φάκελο wp-content/themes.
- **Πρόσθετα (Plugins):** Επεκτάσεις που προσθέτουν λειτουργίες όπως εργαλεία SEO, φόρμες επικοινωνίας, sitemap, cookie compliance, caching, στατιστικά ή μηχανισμούς ασφάλειας. Εγκαθίστανται στο φάκελο wp-content/plugins.
- **Επίπεδο βάσης δεδομένων (Database layer):** Η σχεσιακή βάση δεδομένων που αποθηκεύει χρήστες, άρθρα, μεταδεδομένα, ρυθμίσεις, καθώς και δεδομένα πρόσθετων.

Η συνολική ασφάλεια μιας εγκατάστασης WordPress εξαρτάται από πολλούς παράγοντες, όπως η έκδοση του WordPress, η παραμετροποίηση της εγκατάστασης, η ενεργοποίηση ή όχι συγκεκριμένων χαρακτηριστικών (REST API, XML-RPC), τα πρόσθετα και θέματα, καθώς και τα μέτρα προστασίας σε επίπεδο server.^{1,2}

2.2 Μοντέλο Χρηστών και Έλεγχος Πρόσβασης με Ρόλους (RBAC)

Το WordPress διαθέτει ενσωματωμένο σύστημα διαχείρισης χρηστών που υποστηρίζει αυθεντικοποίηση (authentication) και εξουσιοδότηση (authorization) μέσω ενός μοντέλου ελέγχου πρόσβασης βασισμένου σε ρόλους (Role-Based Access Control – RBAC). Αντί να εκχωρούνται δικαιώματα σε κάθε χρήστη ξεχωριστά, το WordPress αντιστοιχίζει τους χρήστες σε ρόλους που περιλαμβάνουν σύνολα δυνατοτήτων (capabilities). Με αυτόν τον τρόπο υποστηρίζονται ρεαλιστικά σενάρια λειτουργίας, όπου διαφορετικοί χρήστες συμβάλλουν στη δημιουργία και διαχείριση περιεχομένου με διαφορετικά επίπεδα προνομίων.

Σε μία τυπική εγκατάσταση WordPress, οι πιο συνηθισμένοι ρόλοι είναι:

- **Administrator (Διαχειριστής):** Πλήρης πρόσβαση σε όλες τις λειτουργίες, συμπεριλαμβανομένης της διαχείρισης χρηστών, της εγκατάστασης πρόσθετων, της διαχείρισης θεμάτων και των κεντρικών ρυθμίσεων.
- **Editor (Συντάκτης):** Δυνατότητα δημοσίευσης, επεξεργασίας και διαγραφής άρθρων και σελίδων, ακόμη και αυτών που δημιουργήθηκαν από άλλους χρήστες.
- **Author (Συγγραφέας):** Δυνατότητα δημοσίευσης και διαχείρισης των δικών του άρθρων.
- **Contributor (Συνεργάτης):** Δυνατότητα δημιουργίας άρθρων χωρίς δικαίωμα δημοσίευσης (συνήθως απαιτείται έγκριση από Editor ή Administrator).
- **Subscriber (Συνδρομητής):** Ελάχιστα προνόμια, συνήθως περιοριζόμενα στη διαχείριση του προφίλ του και στην πρόσβαση σε περιεχόμενο εφόσον υπάρχει σχετική ρύθμιση.

Από πλευράς ασφάλειας, οι ρόλοι επηρεάζουν την απαρίθμηση χρηστών με δύο βασικούς τρόπους. Πρώτον, καθορίζουν εάν ένας χρήστης μπορεί να παράγει δημόσια ορατό περιεχόμενο (δημοσιευμένα άρθρα), το οποίο ενδέχεται να εκθέτει πληροφορίες ταυτότητας. Δεύτερον, διαφορετικοί ρόλοι έχουν διαφορετική αξία για τον επιτιθέμενο: η παραβίαση λογαριασμού Administrator έχει σαφώς μεγαλύτερο αντίκτυπο σε σχέση με έναν Subscriber.

2.3 Αποκάλυψη Πληροφορίας και Απαρίθμηση Χρηστών

Οι ευπάθειες αποκάλυψης πληροφορίας (information disclosure vulnerabilities) προκύπτουν όταν ένα σύστημα αποκαλύπτει, χωρίς πρόθεση, ευαίσθητες πληροφορίες μέσω των απαντήσεων, των μεταδεδομένων ή της συμπεριφοράς του. Οι πληροφορίες αυτές μπορεί να μην αποτελούν απευθείας διαπιστευτήρια (credentials), ωστόσο μπορούν να χρησιμοποιηθούν από έναν επιτιθέμενο για την οργάνωση ή την εκτέλεση περαιτέρω επιθέσεων.⁶

Η απαρίθμηση χρηστών (user enumeration) αποτελεί ειδική μορφή αποκάλυψης πληροφορίας, κατά την οποία ένας επιτιθέμενος μπορεί να συμπεράνει εάν ένας λογαριασμός χρήστη υπάρχει. Αυτό επιτυγχάνεται συνήθως μέσω διαφορών στη συμπεριφορά του συστήματος όπως:^{5,15}

- διαφορετικά μηνύματα σφάλματος (π.χ. “άγνωστος χρήστης” έναντι “λανθασμένος κωδικός”)
- διαφορετικά μοτίβα HTTP redirects
- διαφορετικοί κωδικοί HTTP απόκρισης (200 έναντι 404)
- διαφορές στον χρόνο απόκρισης από τον server (timing differences)
- δημόσια APIs που επιστρέφουν δεδομένα χρήστη
- μεταδεδομένα σε HTML σελίδες ή feeds

Σε συστήματα που βασίζονται σε αυθεντικοποίηση, η ύπαρξη έγκυρων usernames μειώνει σημαντικά την αβεβαιότητα για έναν επιτιθέμενο. Έτσι, αντί να δοκιμάζει τυχαίους συνδυασμούς ονόματος χρήστη και κωδικού, μπορεί να δημιουργήσει λίστες επιβεβαιωμένων λογαριασμών και να πραγματοποιήσει πιο στοχευμένες επιθέσεις.

Παρότι η απαρίθμηση χρηστών από μόνη της δεν συνεπάγεται άμεση μη εξουσιοδοτημένη πρόσβαση, μειώνει την προσπάθεια που απαιτείται για την παραβίαση λογαριασμών και αυξάνει την αποτελεσματικότητα αυτοματοποιημένων επιθέσεων.

2.4 Μοντελοποίηση Απειλών και ο Ρόλος της Απαρίθμησης σε Αλυσίδες Επίθεσης

Η μοντελοποίηση απειλών (threat modeling) χρησιμοποιείται για τον καθορισμό των δυνατοτήτων, των στόχων και των πιθανών μεθόδων ενός επιτιθέμενου. Στο πλαίσιο αυτής της διατριβής, θεωρείται ότι ο επιτιθέμενος δεν διαθέτει αρχικά διαπιστευτήρια και μπορεί να στείλει HTTP αιτήματα προς το server του WordPress site.^{9,10}

Η απαρίθμηση χρηστών λειτουργεί συχνά ως ενδιάμεσο στάδιο στην ακόλουθη αλυσίδα επίθεσης:

1. **Αναγνώριση** (Reconnaissance): Εντοπισμός τεχνολογιών και endpoints (CMS, έκδοση, πρόσθετα).
2. **Απαρίθμηση χρηστών**: Εντοπισμός έγκυρων usernames και σχετικών αναγνωριστικών.
3. **Επιθέσεις διαπιστευτηρίων**: Brute force, password spraying ή credential stuffing.
4. **Κλιμάκωση προνομίων ή διατήρηση πρόσβασης**: Απόπειρα απόκτησης υψηλότερων δικαιωμάτων ή μόνιμης πρόσβασης.
5. **Τελικές ενέργειες**: Τροποποίηση περιεχομένου, παραμόρφωση ιστοσελίδας (defacement) και εξαγωγή δεδομένων.

Η απαρίθμηση χρηστών αυξάνει σημαντικά την πιθανότητα επιτυχίας στο στάδιο των επιθέσεων διαπιστευτηρίων, καθώς επιτρέπει στον επιτιθέμενο να επικεντρωθεί σε λογαριασμούς που είναι επιβεβαιωμένα υπαρκτοί.

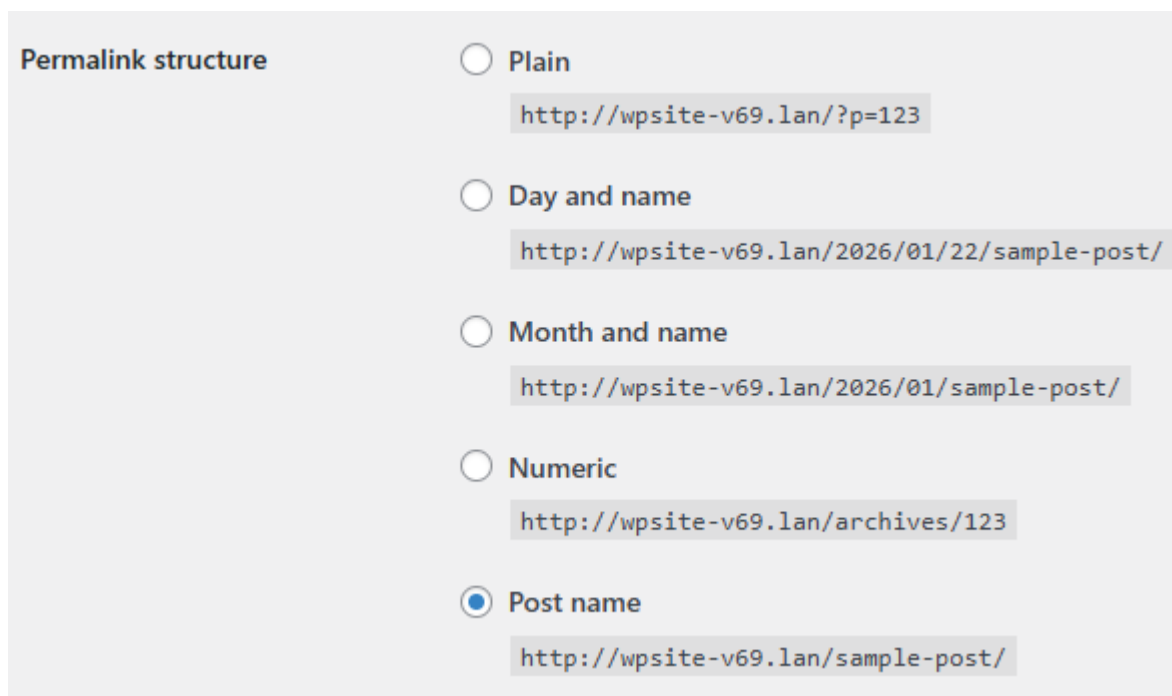
2.5 Μηχανισμοί Απαρίθμησης στο WordPress

Το WordPress εκθέτει διάφορες λειτουργίες που μπορούν να επιτρέψουν απαρίθμηση χρηστών με διαφορετικούς τρόπους. Ορισμένοι μηχανισμοί απαρίθμησης είναι άμεσοι (επιστρέφουν εμφανώς δεδομένα χρήστη), ενώ άλλα είναι έμμεσοι (π.χ. αποκαλύπτουν ID ή author slug που μπορεί να συσχετιστεί με κάποιο χρήστη).

2.5.1 Απαρίθμηση μέσω Author Archive

Το WordPress παρέχει σελίδες “author archive”, οι οποίες εμφανίζουν τα δημοσιευμένα άρθρα ενός συγγραφέα. Όταν είναι ενεργοποιημένα τα permalinks (στα Settings το permalink structure να είναι “Post name”), οι σελίδες συγγραφέων είναι συνήθως διαθέσιμες μέσω URLs όπως:

`/author/<author-slug>/`



The image shows the 'Permalink structure' settings in WordPress. There are five radio button options: 'Plain', 'Day and name', 'Month and name', 'Numeric', and 'Post name'. The 'Post name' option is selected, indicated by a blue dot. Below each option is a preview of the resulting URL. The preview for 'Post name' is `http://wpsite-v69.lan/sample-post/`.

Permalink structure	Example URL
☐ Plain	<code>http://wpsite-v69.lan/?p=123</code>
☐ Day and name	<code>http://wpsite-v69.lan/2026/01/22/sample-post/</code>
☐ Month and name	<code>http://wpsite-v69.lan/2026/01/sample-post/</code>
☐ Numeric	<code>http://wpsite-v69.lan/archives/123</code>
☒ Post name	<code>http://wpsite-v69.lan/sample-post/</code>

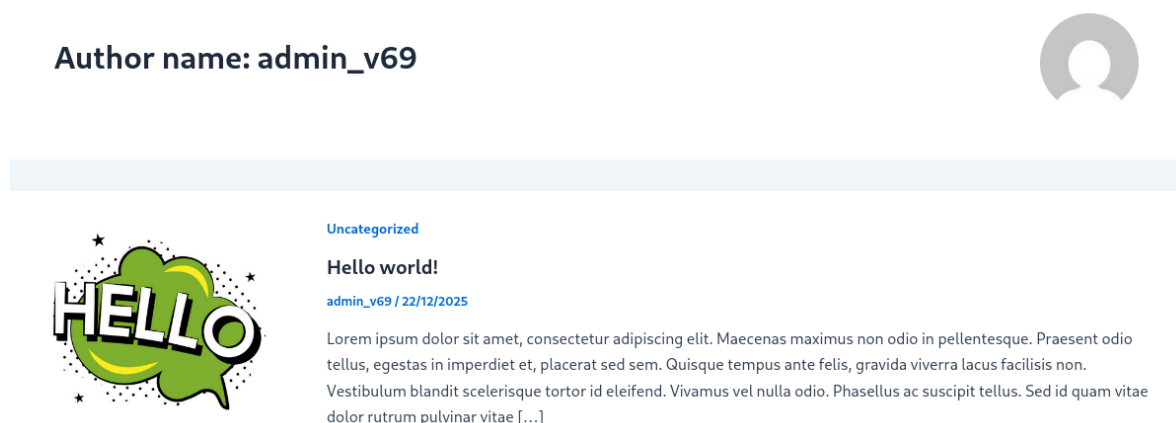
Εικόνα 2. Permalink Settings

Ιστορικά, το WordPress παρείχε επίσης δυνατότητα αναγνώρισης των συγγραφέων μέσω αριθμητικών IDs μετά από κάποιο query. Όταν το σύστημα δέχεται έναν αριθμητικό author ID και κάνει redirect στην αντίστοιχη σελίδα συγγραφέα, ένας επιτιθέμενος μπορεί να δοκιμάσει διαδοχικά IDs και να εντοπίσει έγκυρους χρήστες. Αυτή η συμπεριφορά μπορεί να αποκαλύψει usernames ή “nicenames”, ανάλογα με τη ρύθμιση.

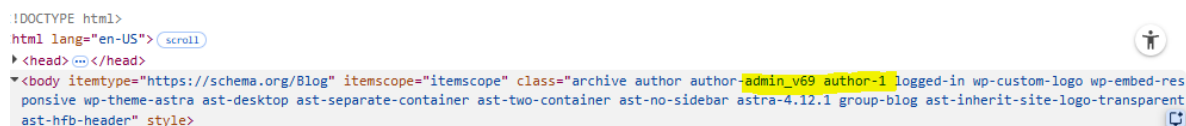
Η απαρίθμηση μέσω author ID είναι ιδιαίτερα αποτελεσματική, καθώς τα αριθμητικά IDs έχουν χαμηλή εντροπία και μπορούν να δοκιμαστούν εύκολα μαζί.

2.5.2 Παθητική Απαρίθμηση μέσω Δημόσιου Περιεχομένου και Μεταδεδομένων

Ακόμη και αν η άμεση απαρίθμηση μέσω author ID αποτραπεί, η ταυτότητα χρηστών μπορεί να εξαχθεί παθητικά από δημόσια προσβάσιμες σελίδες. Τα άρθρα στο WordPress συχνά εμφανίζουν πληροφορίες συγγραφέα στο front-end, όπως το εμφανιζόμενο όνομα (display name) και έναν σύνδεσμο προς το author archive. Επιπλέον, ο HTML πηγαίος κώδικας μπορεί να περιλαμβάνει στοιχεία που σχετίζονται με τον χρήστη (π.χ. CSS classes ή δομημένα μεταδεδομένα), επιτρέποντας έμμεση αποκάλυψη IDs ή slugs.



Εικόνα 3. Front-end page

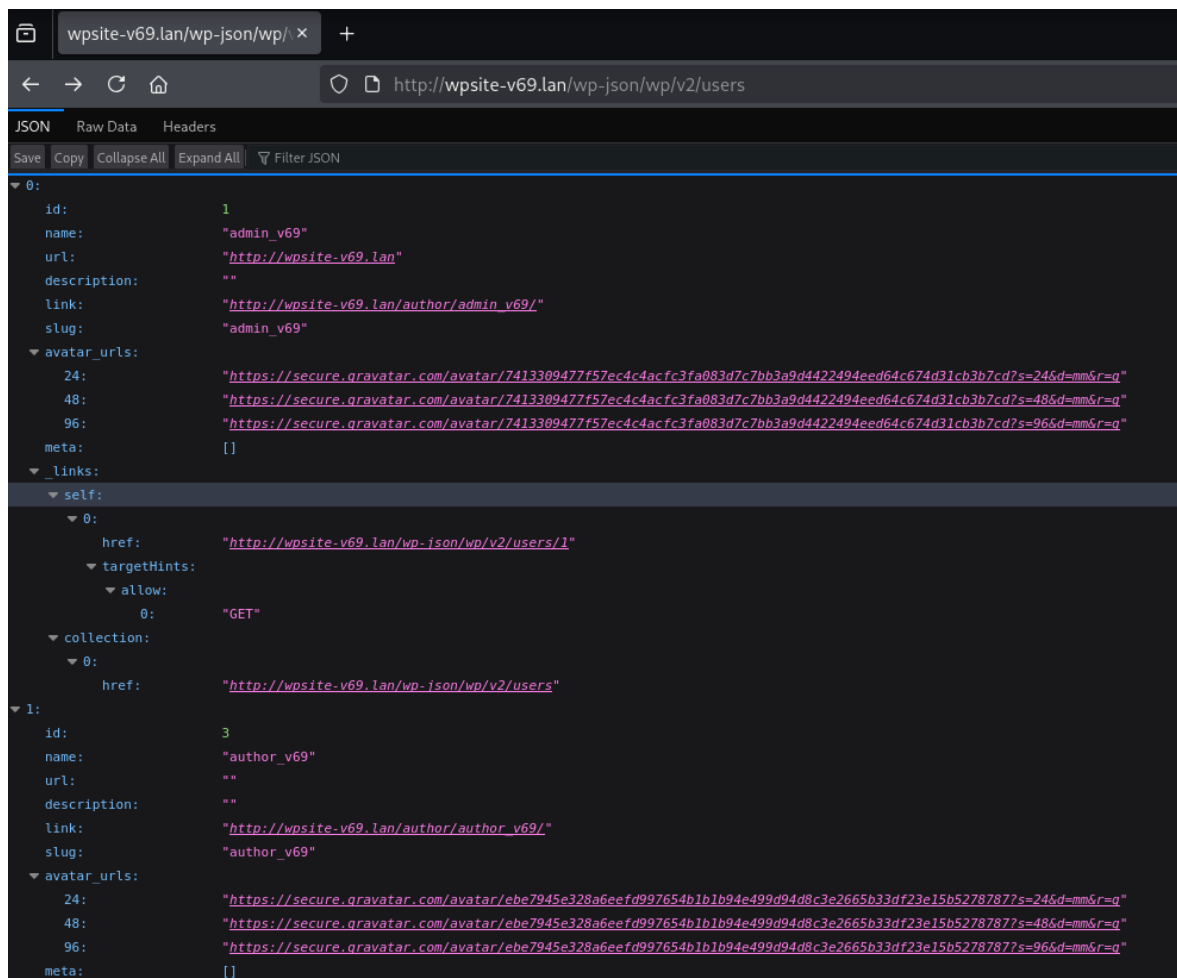


Εικόνα 4. HTML source code

Η παθητική απαρίθμηση είναι ιδιαίτερα σημαντική διότι συχνά δεν θεωρείται “ευπάθεια” από τους διαχειριστές, καθώς σχετίζεται με τη φυσιολογική λειτουργία δημοσίευσης περιεχομένου. Παρ’ όλα αυτά, παρέχει στον επιτιθέμενο πολύτιμες πληροφορίες για λίστες λογαριασμών-στόχων.

2.5.3 Απαρίθμηση μέσω REST API

Το WordPress ενσωματώνει REST API, προσβάσιμο μέσω του path **/wp-json/**. Το REST API υποστηρίζει σύγχρονες εφαρμογές και διασυνδέσεις με τρίτα συστήματα, παρέχοντας προγραμματιστική πρόσβαση σε πόρους όπως άρθρα, κατηγορίες και σχόλια.^{1,2}



Εικόνα 5. REST API

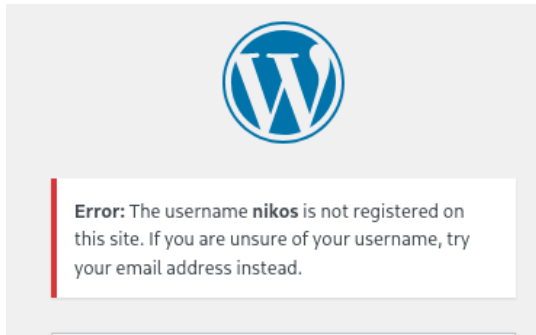
Ανάλογα με τη ρύθμιση και την έκδοση, συγκεκριμένα REST endpoints μπορεί να εκθέτουν δεδομένα χρηστών ή μεταδεδομένα που σχετίζονται με λογαριασμούς. Ακόμη και όταν η άμεση εμφάνιση λίστας χρηστών είναι περιορισμένη, μπορεί να προκύψει έμμεση απαρίθμηση μέσω αντικειμένων περιεχομένου που περιλαμβάνουν author IDs. Ένας επιτιθέμενος μπορεί να συσχετίσει αυτά τα IDs με δημόσιες σελίδες συγγραφέα και να ανασυνθέσει λίστα χρηστών.

2.5.4 Απαρίθμηση μέσω Ανατροφοδότησης στη Σύνδεση (Login Feedback)

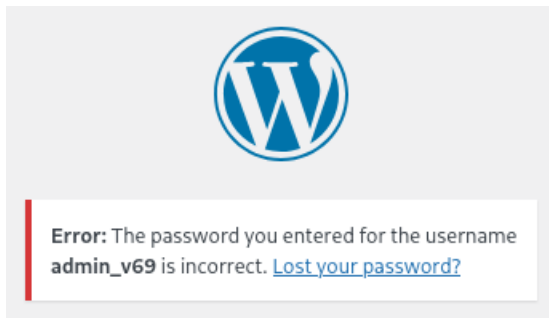
Η σελίδα σύνδεσης του WordPress είναι συνήθως διαθέσιμη μέσω `/wp-login.php`. Η απαρίθμηση μπορεί να προκύψει όταν ο μηχανισμός σύνδεσης αποκαλύπτει διαφορετικές αποκρίσεις σε περιπτώσεις μη έγκυρου username σε σχέση με έγκυρο username αλλά λάθος κωδικό. Διαρροή πληροφορίας μπορεί να περιλαμβάνει:

- διαφορετικά μηνύματα σφάλματος

- διαφορά στο μέγεθος της απόκρισης
- διαφορά στον χρόνο επεξεργασίας (response time)



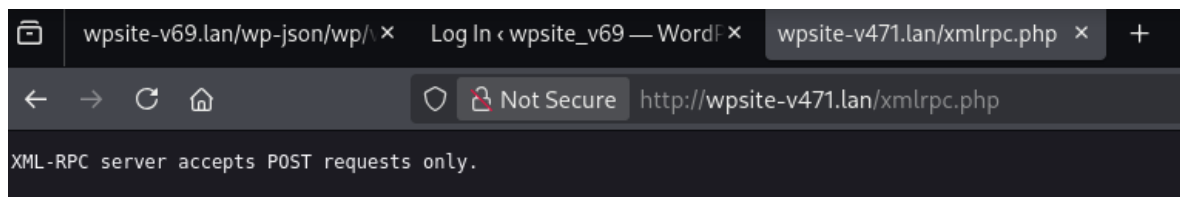
Εικόνα 6-7. Διαφορετικά μηνύματα σφάλματος



Τέτοιες διαφορές μπορούν να αξιοποιηθούν από αυτοματοποιημένα εργαλεία για την επικύρωση usernames. Ακόμη και όταν τα μηνύματα σφάλματος γίνονται τα ίδια, μπορεί να παραμένουν ανεπαίσθητες αλλά μετρήσιμες διαφορές στα HTTP responses.^{9,10}

2.5.5 Έκθεση μέσω XML-RPC

Το XML-RPC αποτελεί μία διεπαφή που επιτρέπει απομακρυσμένες ενέργειες μέσω XML μηνυμάτων πάνω σε HTTP, και χρησιμοποιήθηκε ιστορικά για απομακρυσμένη δημοσίευση και legacy ενσωματώσεις. Συνήθως είναι προσβάσιμο μέσω **/xmlrpc.php**.



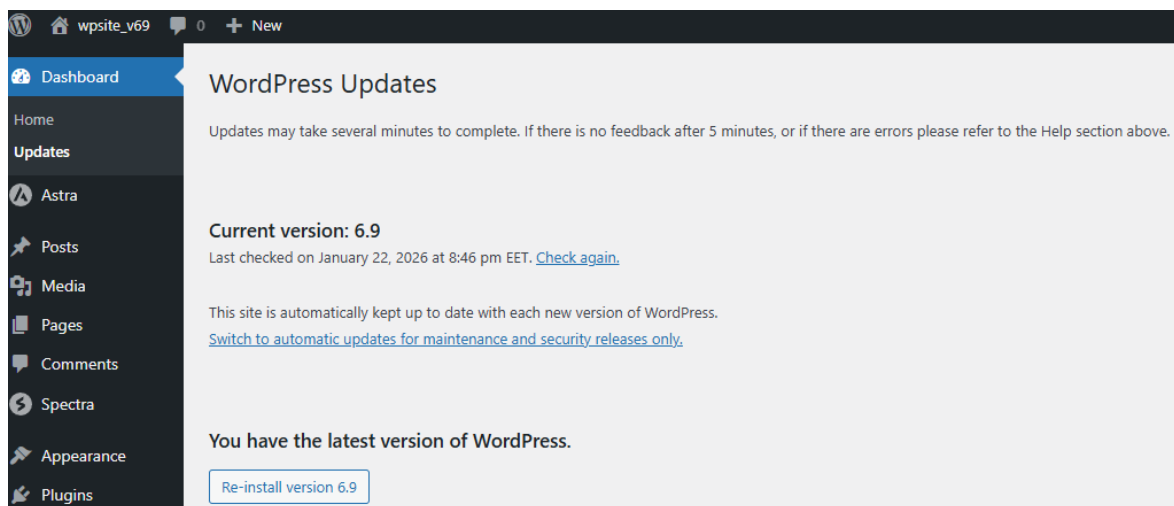
Εικόνα 8. XML-RPC

Από πλευράς ασφάλειας, το XML-RPC μπορεί να αυξήσει την επιφάνεια επίθεσης, καθώς μπορεί να επιτρέψει λειτουργίες αυθεντικοποίησης και ομαδοποιημένα αιτήματα (batch requests).

Παρότι δεν αποτελεί απαραίτητα μηχανισμό απαρίθμησης από μόνο του, η ύπαρξη και η συμπεριφορά του μπορεί να υποστηρίξει αυτοματοποιημένες απόπειρες σύνδεσης ή αναγνώριση λειτουργικότητας, ειδικά σε παλαιότερες εκδόσεις WordPress.

2.6 Νέες Εκδόσεις WordPress και Εξέλιξη της Ασφάλειας

Ανά τακτά χρονικά διαστήματα δημοσιεύονται νέες εκδόσεις για τον πυρήνα του WordPress, οι οποίες έχουν σαν στόχο την ασφάλεια ή και τη βελτίωση / αναβάθμιση λειτουργιών. Αυτές εμφανίζονται στο υπομενου updates του Dashboard όπου αναφέρεται και η έκδοση του WordPress που τρέχει αυτήν τη στιγμή το site.



Εικόνα 9. WordPress version

Η εξέλιξη του WordPress περιλαμβάνει αλλαγές σε προεπιλεγμένες συμπεριφορές, έκθεση endpoints και μηχανισμούς που σχετίζονται με αυθεντικοποίηση. Βασικό στοιχείο αυτής της διατριβής είναι η σύγκριση μεταξύ μιας παλαιότερης έκδοσης (WordPress 4.7.1) και μιας σύγχρονης έκδοσης (WordPress 6.9). Η σύγκριση αυτή είναι σημαντική διότι:

- οι παλαιότερες εκδόσεις ενδέχεται να εκθέτουν λειτουργίες που έχουν μετριαστεί σε νεότερες
- οι σύγχρονες εκδόσεις συχνά εφαρμόζουν πιο περιοριστικές προεπιλογές, ειδικά σε θέματα REST και δεδομένων χρηστών
- οι παλαιότερες εγκαταστάσεις παραμένουν ρεαλιστικό σενάριο, λόγω ασυμβατοτήτων, έλλειψης συντήρησης ή λειτουργικών περιορισμών

Στην πράξη, η ασφάλεια δεν εξαρτάται μόνο από την έκδοση του WordPress, αλλά και από τις ρυθμίσεις και τα πρόσθετα/θέματα που έχουν εγκατασταθεί. Ως εκ τούτου, η αξιολόγηση σε ελεγχόμενο περιβάλλον επιτρέπει την εξαγωγή συμπερασμάτων για το επίπεδο έκθεσης της πλατφόρμας.

2.7 Αρχές Μετριασμού (Mitigation) για την Απαρίθμηση Χρηστών

Οι τεχνικές μετριασμού για την απαρίθμηση χρηστών στοχεύουν στη μείωση της δυνατότητας ενός επιτιθέμενου να διακρίνει έγκυρους από μη έγκυρους λογαριασμούς. Ενδεικτικά περιλαμβάνουν:

- **Μείωση της δημόσιας έκθεσης αναγνωριστικών χρηστών**, ειδικά σε URLs και APIs
- **Ομογενοποίηση μηνυμάτων σφάλματος και αποκρίσεων**, ώστε να μην αποκαλύπτεται η ύπαρξη λογαριασμών
- **Ενημερώσεις** τόσο του πυρήνα του WordPress όσο και των themes-plugins.
- **Περιορισμός σε endpoints αυθεντικοποίησης**, ώστε να αποτρέπονται αυτοματοποιημένες προσπάθειες
- **Περιορισμός ή απενεργοποίηση μη απαραίτητων λειτουργιών**, όπως το XML-RPC
- **Εγκατάσταση προσθέτων για ασφάλεια**, όπως το Wordfence Security
- **Ενέργειες σε επίπεδο server** (π.χ. κανόνες στο web server ή WAF) για φιλτράρισμα γνωστών μοτίβων enumeration

Ένα κρίσιμο σημείο είναι ότι ορισμένες μορφές αποκάλυψης σχετίζονται με τη φυσιολογική λειτουργικότητα δημοσίευσης περιεχομένου. Για παράδειγμα, η εμφάνιση ονόματος συγγραφέα σε ένα blog post αποτελεί βασικό χαρακτηριστικό. Ως αποτέλεσμα, ο μετριασμός συνήθως επικεντρώνεται στην αποτροπή μαζικής/εύκολης απαρίθμησης, παρά στην πλήρη εξάλειψη κάθε δημόσιας πληροφορίας συγγραφέα.

Κεφάλαιο 3 — Μεθοδολογία και Εργαστηριακή Υποδομή

3.1 Επισκόπηση

Το παρόν κεφάλαιο περιγράφει τη μεθοδολογία έρευνας και το εργαστηριακό περιβάλλον που ακολουθήθηκε για την αξιολόγηση της απαρίθμησης χρηστών (user enumeration) σε εγκαταστάσεις WordPress. Στόχος της σχεδίασμού είναι να διασφαλιστεί ότι όλα τα τεστ μπορούν να επαναληφθούν (reproducibility), ότι τα αποτελέσματα μπορούν να συγκριθούν μεταξύ διαφορετικών εκδόσεων WordPress υπό ελεγχόμενες συνθήκες, και ότι η παρατηρούμενη συμπεριφορά απαρίθμησης μπορεί να ποσοτικοποιηθεί χρησιμοποιώντας σταθερές μετρήσεις.^{1,5,15}

Η έρευνα βασίζεται σε συγκριτική πειραματική προσέγγιση, μέσω δοκιμών συγκεκριμένων μηχανισμών απαρίθμησης σε δύο εκδόσεις WordPress: μία παλαιότερη (WordPress 4.7.1) και μία σύγχρονη (WordPress 6.9). Όλες οι δοκιμές πραγματοποιήθηκαν στο ελεγχόμενο εργαστηριακό περιβάλλον, προκειμένου να αποφευχθούν ηθικά και νομικά ζητήματα που θα προέκυπταν σε πραγματικά συστήματα.

3.2 Εργαστηριακό Περιβάλλον και Ρυθμίσεις Συστήματος

3.2.1 Εικονική Υποδομή

Όλα τα πειράματα διεξήχθησαν σε ιδιωτικό εργαστήριο που περιλάμβανε δύο εικονικές μηχανές (VMs) σε VMware Workstation:

1. VM-Στόχος (Ubuntu Server)

- Πόροι: 2 vCPUs, 4 GB RAM
- Δίκτυο: στατική IP 192.168.178.4
- Web server: Nginx
- Εφαρμογές: δύο εγκαταστάσεις WordPress

2. VM-Επιτιθέμενος (Kali Linux)

- Δίκτυο: στατική IP 192.168.178.5
- Χρησιμοποιήθηκε για όλες τις ενέργειες απαρίθμησης και μετρήσεις

Το σύστημα επιτιθέμενου και το σύστημα στόχου τοποθετήθηκαν στο ίδιο τοπικό δίκτυο (ίδιο subnet), ώστε να εξασφαλιστεί σταθερή συνδεσιμότητα και σταθερό request latency.

3.2.2 Εγκαταστάσεις WordPress και Ρύθμιση Domain

Ο Ubuntu server φιλοξενούσε δύο ξεχωριστές εγκαταστάσεις WordPress, καθεμία αντιστοιχισμένη σε διαφορετικό hostname μέσω Nginx server block configuration:^{7,8}

- wpsite-v471.lan → WordPress 4.7.1
- wpsite-v69.lan → WordPress 6.9

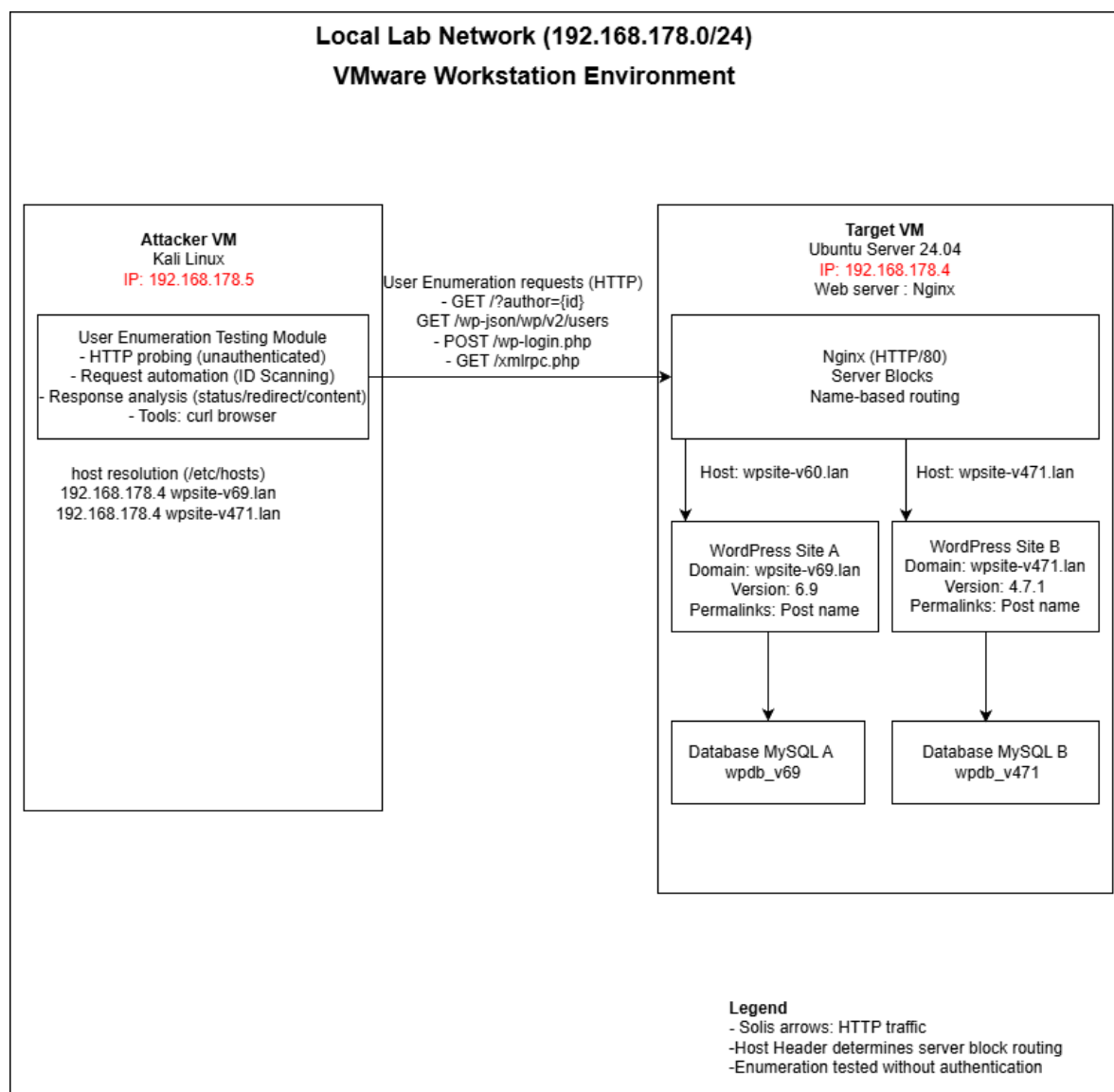
Η επίλυση ονομάτων (name resolution) των εσωτερικών domains υλοποιήθηκε μέσω τοπικών καταχωρήσεων στο αρχείο hosts στον επιτιθέμενο (/etc/hosts). Με αυτόν τον τρόπο

επιτεύχθηκε σταθερή και επαναλήψιμη πρόσβαση στα συστήματα χωρίς την ανάγκη εξωτερικής DNS υποδομής.

Και οι δύο εγκαταστάσεις WordPress ήταν προσβάσιμες μέσω HTTP. Παρότι το HTTPS αποτελεί συνήθη πρακτική σε παραγωγικά περιβάλλοντα (production), το αντικείμενο της έρευνας επικεντρώνεται στη συμπεριφορά απαρίθμησης σε επίπεδο εφαρμογής (application layer), η οποία παραμένει ορατή ανεξαρτήτως του αν το κανάλι μεταφοράς είναι κρυπτογραφημένο.^{1,9,10}

3.3.3 Τοπολογία Δικτύου

Η λογική τοπολογία του εργαστηρίου συνοψίζεται ως εξής:



Εικόνα 10. Τοπολογία δικτύου

3.3.4 Βασικές Παραδοχές (Baseline Configuration)

Για να περιοριστούν οι εξωτερικές μεταβλητές και να αναδειχθούν καθαρά οι διαφορές μεταξύ των εκδόσεων WordPress, εφαρμόστηκαν οι ακόλουθες βασικές παραδοχές:

- Δεν εγκαταστάθηκαν πρόσθετα ασφάλειας (π.χ. WAF plugins, login protection plugins, security plugins).
- Τα προεπιλεγμένα endpoints αυθεντικοποίησης ήταν προσβάσιμα (π.χ. /wp-login.php).
- Τα permalinks ρυθμίστηκαν σε "Post name", επιτρέποντας φιλικά URLs και υποστηρίζοντας author archive URLs.
- Και οι δύο ιστότοποι ήταν προσβάσιμοι από τον επιτιθέμενο χωρίς προηγούμενη αυθεντικοποίηση.

Η επιλογή αυτή αναπαριστά ένα συχνό σενάριο βασικής εγκατάστασης WordPress και προσφέρει καθαρό πλαίσιο για σύγκριση της συμπεριφοράς ανά έκδοση.

3.4 Αρχιτεκτονική Εξυπηρετητή

3.4.1 Δρομολόγηση μέσω Nginx Server blocks

Ο Nginx χρησιμοποιήθηκε για τη φιλοξενία και των δύο WordPress sites στην ίδια IP μέσω name-based virtual hosting. Τα εισερχόμενα αιτήματα δρομολογούνται με βάση το HTTP Host header, επιτρέποντας τη συνύπαρξη των δύο εγκαταστάσεων κάτω από διαφορετικά domain names.^{7,8,9,10}

Κάθε WordPress instance ρυθμίστηκε με ξεχωριστό Nginx server block, το οποίο περιλαμβάνει:

- listen 80; για HTTP πρόσβαση
- μοναδικό server name (wpsite-v471.lan, wpsite-v69.lan)
- ανεξάρτητο document root κάτω από /var/www/
- κανόνες δρομολόγησης που υποστηρίζουν permalinks ("pretty URLs")

```
server {
    listen 80;
    server_name wpsite-v69.lan;
    root /var/www/wpsite-v69.lan;

    index index.php index.html;

    location / {
        try_files $uri $uri/ /index.php?$args;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/run/php/php8.3-fpm.sock;
    }

    location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg)$ {
        expires max;
        log_not_found off;
    }
}
```

Εικόνα 11. wpsite-v69.lan

```
server {
    listen 80;
    server_name wpsite-v471.lan;
    root /var/www/wpsite-v471.lan;

    index index.php index.html;

    location / {
        try_files $uri $uri/ /index.php?$args;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/run/php/php7.2-fpm.sock;
    }

    location ~* \.(js|css|png|jpg|jpeg|gif|ico|svg)$ {
        expires max;
        log_not_found off;
    }
}
```

Εικόνα 12. wpsite-v471.lan

Με αυτή τη ρύθμιση διασφαλίζεται ότι ένα αίτημα προς το ένα site δεν θα απαντηθεί από την άλλη εγκατάσταση.

3.4.2 Απομόνωση Μεταξύ των WordPress Instances

Παρότι οι δύο εκδόσεις WordPress φιλοξενούνται στο ίδιο λειτουργικό σύστημα, επιτεύχθηκε λογική απομόνωση μέσω:

1. Ξεχωριστών web roots

Κάθε instance διαθέτει δικό του φάκελο κάτω από /var/www/, αποφεύγοντας αλληλεπικαλύψεις αρχείων.

2. Ανεξάρτητων βάσεων δεδομένων

Κάθε WordPress εγκατάσταση συνδέεται με ξεχωριστή βάση δεδομένων (ή schema), ώστε να μην μοιράζονται λογαριασμούς, άρθρα ή ρυθμίσεις.

3. Διακριτών Nginx virtual hosts

Η δρομολόγηση βάσει domain υποστηρίζει καθαρά πειράματα και επαναληψιμότητα.

Αυτός ο σχεδιασμός επιτρέπει άμεση σύγκριση μεταξύ των δύο WordPress εκδόσεων, με ελαχιστοποιημένες παρεμβολές από άλλους παράγοντες.

3.5 Λογαριασμοί Χρηστών και Ρύθμιση Ρόλων

3.5.1 Επιλογή Ρόλων

Το WordPress διαθέτει προεπιλεγμένο σύστημα ελέγχου πρόσβασης βασισμένο σε ρόλους (RBAC). Για τη διερεύνηση του κατά πόσο τα επίπεδα προνομίων επηρεάζουν την έκθεση απαρίθμησης, δημιουργήθηκαν πέντε λογαριασμοί χρηστών σε κάθε εγκατάσταση:

1. Administrator
2. Editor
3. Author
4. Contributor
5. Subscriber

Ο σχεδιασμός αυτός επιτρέπει τη μελέτη του αν οι μηχανισμοί απαρίθμησης εκθέτουν όλους τους χρήστες ισότιμα ή κυρίως λογαριασμούς που συνδέονται με δημόσια προσβάσιμο περιεχόμενο.

3.5.2 Έλεγχος Δημόσιας Ορατότητας Περιεχομένου

Για να αποφευχθεί μεροληψία από διαφορές στο δημοσιευμένο περιεχόμενο, η δημόσια ορατότητα ρυθμίστηκε ως εξής σε κάθε ιστότοπο:

- Οι λογαριασμοί Administrator, Editor και Author συνδέθηκαν με τουλάχιστον ένα δημοσιευμένο άρθρο.
- Ο λογαριασμός Contributor δημιούργησε άρθρο σε μορφή draft (μη δημοσιευμένο).
- Ο λογαριασμός Subscriber δεν δημιούργησε περιεχόμενο.

Η ρύθμιση αυτή επιτρέπει την υπόθεση ότι ορισμένοι μηχανισμοί απαρίθμησης αποκαλύπτουν μόνο λογαριασμούς που είναι δημόσια συνδεδεμένοι με περιεχόμενο, ενώ λογαριασμοί χωρίς δημόσια δραστηριότητα παραμένουν λιγότερο εκτεθειμένοι.

3.6 Μοντέλο Απειλής (Threat Model)

3.6.1 Προφίλ Επιτιθέμενου και Δυνατότητες

Ο επιτιθέμενος μοντελοποιείται ως μία μη αυθεντικοποιημένη οντότητα με δικτυακή πρόσβαση στις web εφαρμογές στόχους. Οι δυνατότητες του επιτιθέμενου περιλαμβάνουν:

- αποστολή αυθαίρετων HTTP αιτημάτων προς τους ιστότοπους
- επιθεώρηση HTTP status codes, headers και περιεχομένου
- αυτοματοποίηση αιτημάτων και μοτίβων σκαναρίσματος (π.χ. δοκιμή διαδοχικών IDs)
- παθητική ανάλυση δημόσιου περιεχομένου (π.χ. εξαγωγή μεταδεδομένων)

Ο επιτιθέμενος θεωρείται ότι δεν διαθέτει αρχικά διαπιστευτήρια, δεν έχει πρόσβαση σε διαχειριστικές λειτουργίες και δεν έχει πρόσβαση σε server logs ή εσωτερικές ρυθμίσεις.

3.6.2 Στόχοι Επιτιθέμενου

Ο κύριος στόχος του επιτιθέμενου είναι η ανακάλυψη έγκυρων λογαριασμών χρηστών και αναγνωριστικών που μπορούν να χρησιμοποιηθούν σε επόμενα στάδια επιθέσεων. Οι στόχοι περιλαμβάνουν:

- εντοπισμό έγκυρων usernames, author slugs, display names ή user IDs
- συσχέτιση αναγνωριστικών με δημόσια προφίλ ή author archive σελίδες
- μείωση του εύρους αναζήτησης για επιθέσεις διαπιστευτηρίων όπως password spraying

Ο επιτιθέμενος δεν επιδιώκει εκμετάλλευση ευπαθειών πέρα από την απαρίθμηση (π.χ. απομακρυσμένη εκτέλεση κώδικα), καθώς το αντικείμενο της διατριβής περιορίζεται στους μηχανισμούς αποκάλυψης χρήστη.

3.6.3 Παραδοχές και Περιορισμοί Εμβέλειας

Το μοντέλο απειλής βασίζεται στις ακόλουθες παραδοχές:

- ο επιτιθέμενος δεν μπορεί να τροποποιήσει κώδικα ή ρυθμίσεις του server
- ο επιτιθέμενος δεν παρακάμπτει την αυθεντικοποίηση μέσω exploits
- ο επιτιθέμενος δεν χρησιμοποιεί εξωτερικές πηγές δεδομένων (π.χ. διαρροές credentials)
- οι δοκιμές περιορίζονται στις δύο εκδόσεις WordPress και στο συγκεκριμένο configuration

3.7 Μηχανισμοί Απαρίθμησης και Περιπτώσεις Δοκιμών (Test Cases)

Για συστηματική αξιολόγηση της απαρίθμησης χρηστών, ορίστηκε ένα σύνολο μηχανισμών που βασίζονται σε συχνά σημεία έκθεσης του WordPress και σε γνωστές συμπεριφορές CMS. Κάθε μηχανισμός δοκιμάστηκε με την ίδια διαδικασία και στις δύο εκδόσεις.¹

3.7.1 Μηχανισμός A — Απαρίθμηση Author μέσω Query Parameter

Η απαρίθμηση author δοκιμάστηκε μέσω αριθμητικών author IDs χρησιμοποιώντας αιτήματα της μορφής: `/?author={id}`. Η δοκιμή μετρά εάν το WordPress αποκρίνεται με redirect προς author archive URL όπως: `/author/<author-slug>/`

Δεδομένα που καταγράφηκαν:

- HTTP response code (π.χ. 200, 301, 302, 404)
- ύπαρξη Location header (redirect)
- εξαγόμενο αναγνωριστικό χρήστη από το redirect
- εύρος IDs που οδηγούν σε επιτυχή αντιστοίχιση author

Συμπέρασμα:

Τα αριθμητικά IDs έχουν χαμηλή εντροπία και δίνουν την δυνατότητα αυτοματοποιημένης απαρίθμησης.

3.7.2 Μηχανισμός B — Παθητική Απαρίθμηση μέσω Δημόσιου Περιεχομένου

Η παθητική απαρίθμηση αξιολογήθηκε μέσω επιθεώρησης δημοσίως διαθέσιμων άρθρων και σελίδων για στοιχεία ταυτότητας όπως:

- εμφάνιση author name στο περιεχόμενο της σελίδας
- author archive links στον HTML κώδικα
- user-related CSS classes ή δομημένα μεταδεδομένα

Δεδομένα που καταγράφηκαν:

- εάν το author link αποκαλύπτει μοναδικό slug
- εάν το εμφανιζόμενο author name ταυτίζεται με το login username
- ύπαρξη author metadata σε πολλαπλά άρθρα

Συμπέρασμα:

Πολλοί ιστότοποι WordPress εμφανίζουν σκόπιμα το όνομα συγγραφέα, γεγονός που μπορεί να δώσει στον επιτιθέμενο έγκυρα αναγνωριστικά χωρίς ενεργή σάρωση.

3.7.3 Μηχανισμός C — Απαρίθμηση μέσω REST API

Το WordPress REST API δοκιμάστηκε μέσω αιτημάτων προς endpoints όπως:

- /wp-json/
- /wp-json/wp/v2/users
- /wp-json/wp/v2/posts

Η δοκιμή εξετάζει εάν είναι δυνατή η επιστροφή λίστας χρηστών ή η έμμεση εξαγωγή αναγνωριστικών.

Δεδομένα που καταγράφηκαν:

- HTTP status codes (200, 401, 403, 404)
- επιστρεφόμενα user-related objects
- fields που εκτίθενται (π.χ. id, slug, name, link)
- διαφορές συμπεριφοράς μεταξύ εκδόσεων

Συμπέρασμα:

Το REST API είναι ευρέως χρησιμοποιούμενο και μπορεί να αποκαλύπτει user identifiers άμεσα ή έμμεσα.

3.7.4 Μηχανισμός D — Απαρίθμηση μέσω Ανατροφοδότησης Σύνδεσης (Login Feedback)

Η απαρίθμηση μέσω login feedback αξιολογήθηκε μέσω αποστολής προσπαθειών σύνδεσης στο **/wp-login.php** με ελεγχόμενα credentials όπου έγιναν δοκιμές με ανύπαρκτο username και δοκιμές με έγκυρο username και λάθος κωδικό⁶

Δεδομένα που καταγράφηκαν:

- διαφορές στα μηνύματα σφάλματος
- διαφορές στα HTTP responses (μέγεθος, redirect patterns)
- διαφορές χρόνου μεταξύ valid και invalid usernames

Συμπέρασμα:

Ακόμη και μικρές διαφορές μπορούν να αξιοποιηθούν από αυτοματοποιημένα εργαλεία για επικύρωση usernames.

3.7.5 Μηχανισμός E — Έκθεση μέσω XML-RPC

Η ύπαρξη και συμπεριφορά του XML-RPC αξιολογήθηκε μέσω αιτημάτων προς **/xmlrpc.php**

Δεδομένα που καταγράφηκαν:

- εάν το endpoint είναι ενεργό και προσβάσιμο

- χαρακτηριστικά της απόκρισης που υποδηλώνουν διαθεσιμότητα υπηρεσίας

Συμπέρασμα:

Το XML-RPC αυξάνει την επιφάνεια επίθεσης και είναι ιδιαίτερα σημαντικό ειδικά σε παλαιότερες εγκαταστάσεις WordPress.

3.8 Κριτήρια Μέτρησης και Αξιολόγηση των Metrics

3.8.1 Κατηγοριοποίηση Επιτυχίας Απαρίθμησης

Για κάθε μηχανισμό, η επιτυχία απαρίθμησης ταξινομήθηκε σε τρεις κατηγορίες:

- Καμία : Δεν υπάρχει παρατηρήσιμη διαφορά ή κάποια διαρροή στοιχείων χρήστη.
- Έμμεση : Δεν αποκαλύπτεται άμεσα username, αλλά εμφανίζεται διακριτή συμπεριφορά (π.χ. timing ή metadata) ώστε να εξαχθούν συμπεράσματα.
- Άμεση : Αποκαλύπτεται απευθείας αναγνωριστικό χρήστη, όπως author slug, username ή user object επιστρεφόμενα μέσω API.

Η κλίμακα αυτή επιτρέπει ποσοτική σύγκριση μεταξύ μηχανισμών και εκδόσεων.

3.8.2 Τύποι Διαρροής Πληροφορίας

Όταν η απαρίθμηση ήταν επιτυχής, οι πληροφορίες που διέρρευσαν κατηγοριοποιήθηκαν ως εξής:

- Username (αναγνωριστικό σύνδεσης)
- Author slug / nicename (αναγνωριστικό με βάση το URL)
- Αριθμητικό user ID
- Display name (εμφανιζόμενο όνομα)
- Έμμεση ένδειξη ρόλου (εφόσον μπορεί να συναχθεί)

3.8.3 Προσπάθειες και Μέτρηση Κόστους

Κάθε μέθοδος απαρίθμησης αξιολογήθηκε και ως προς το πρακτικό της κόστος

- Αριθμός αιτημάτων για την ανακάλυψη τουλάχιστον ενός έγκυρου χρήστη
- Κλιμάκωση (scalability): δυνατότητα μαζικής απαρίθμησης
- Προϋποθέσεις: απαιτεί δημοσιευμένο περιεχόμενο, απαιτεί αυθεντικοποίηση κ.λπ.

3.8.4 Ανίχνευση επιτιθέμενου

Παρότι η ανίχνευση των επιθέσεων δεν αποτελεί το κύριο στόχο, καταγράφηκαν ενδεικτικά στοιχεία όπως:

- αν παράγονται μοτίβα μη φυσιολογικής μετακίνησης δεδομένων (traffic patterns)
- αν απαιτούνται επαναλαμβανόμενα αιτήματα σε ένα endpoint (εύκολη ανίχνευση)
- αν μπορεί να γίνει παθητικά (δυσκολότερη ανίχνευση)

3.9 Εργαλεία και Εκτέλεση Πειραμάτων

3.9.1 Δημιουργία Αιτημάτων

Τα τεστ απαρίθμησης εκτελέστηκαν μέσω command-line HTTP εργαλείων από το Kali VM. Το κύριο εργαλείο ήταν το **curl**, καθώς επιτρέπει:^{9,10}

- προβολή HTTP status και headers
- πλήρη έλεγχο της δομής αιτήματος
- σταθερή επαναληψιμότητα για τεκμηρίωση

Για την αυτοματοποίηση δοκιμών (π.χ. πολλαπλά author IDs) χρησιμοποιήθηκαν απλά scripts.

3.9.2 Καταγραφή και Τεκμηρίωση Αποτελεσμάτων

Για κάθε δοκιμή, τα αποτελέσματα καταγράφηκαν σε δομημένους πίνακες που περιλάμβαναν:

- έκδοση WordPress και αντίστοιχο domain
- τύπο αιτήματος και παραμέτρους
- HTTP αποκρίσεις (status / redirects)
- αναγνωριστικά που εξήχθησαν
- σημειώσεις για ιδιαίτερη συμπεριφορά ανά έκδοση

Παράλληλα πάρθηκαν στιγμιότυπα (screenshots) και αποσπάσματα από raw HTTP responses, ώστε τα αποτελέσματα να είναι επαληθεύσιμα και επαναλήψιμα.

3.9.3 Αυτοματοποιημένο Εργαλείο Απαρίθμησης

Για την εξασφάλιση της επαναληψιμότητας και της συνέπειας των πειραμάτων, αναπτύχθηκε ένα προσαρμοσμένο εργαλείο απαρίθμησης σε Python (wp_enum_lab_v5.py). Το script ενοποιεί πολλαπλές τεχνικές απαρίθμησης χρηστών σε μία αυτοματοποιημένη διαδικασία, συμπεριλαμβανομένων των:

- Απαρίθμηση Author ID μέσω /?author=ID
- Απαρίθμηση χρηστών μέσω REST API (/wp-json/wp/v2/users)
- Εξαγωγή Author IDs μέσω REST posts (/wp-json/wp/v2/posts)
- Παθητική αναγνώριση χρηστών μέσω HTML author links
- Έλεγχος έκθεσης του XML-RPC endpoint

Το εργαλείο λειτουργεί αποκλειστικά σε δημόσια διαθέσιμα endpoints χωρίς αυθεντικοποίηση και χρησιμοποιήθηκε σε ελεγχόμενο εργαστηριακό περιβάλλον. Με τον τρόπο αυτό διασφαλίζεται η συμμόρφωση με τις ηθικές αρχές, ενώ επιτρέπεται η συστηματική σύγκριση μεταξύ διαφορετικών εκδόσεων WordPress.

Ο πλήρης κώδικας του εργαλείου παρατίθεται στο Παράρτημα Α.

Το script εκτελέστηκε και στα δύο περιβάλλοντα ως εξής:

```
python3 wp_enum_lab_v5.py http://wpsite-v471.lan/  
python3 wp_enum_lab_v5.py http://wpsite-v69.lan/
```

3.10 Ηθικές Παρατηρήσεις

Όλα τα πειράματα πραγματοποιήθηκαν αποκλειστικά σε κλειστό εργαστηριακό περιβάλλον. Δεν πραγματοποιήθηκε σάρωση ή δοκιμή σε δημόσιους ιστότοπους ή υποδομές τρίτων. Η εμβέλεια της έρευνας περιορίστηκε αποκλειστικά στη συμπεριφορά απαρίθμησης χρηστών και δεν περιλαμβάνει cracking κωδικών, εκμετάλλευση (exploitation), διατήρηση πρόσβασης ή συνθήκες άρνησης υπηρεσίας (DoS).

Η προσέγγιση αυτή διασφαλίζει τη συμμόρφωση με αρχές ηθικής έρευνας και αποτρέπει μη εξουσιοδοτημένες ενέργειες.

Κεφάλαιο 4 — Πειραματικά Αποτελέσματα και Ανάλυση

4.1 Επισκόπηση

Το παρόν κεφάλαιο παρουσιάζει τα πειραματικά αποτελέσματα που προέκυψαν από τη δοκιμή των μηχανισμών απαρίθμησης χρηστών (user enumeration) στις δύο εγκαταστάσεις WordPress που φιλοξενήθηκαν σε ελεγχόμενο εργαστηριακό περιβάλλον: WordPress 4.7.1 (wpsite-v471.lan) και WordPress 6.9 (wpsite-v69.lan). Η ανάλυση επικεντρώνεται σε μη αυθεντικοποιημένη απαρίθμηση και αξιολογεί εάν ταυτότητες χρηστών ή σχετικά αναγνωριστικά εκτίθενται μέσω κοινών WordPress endpoints.

Οι μηχανισμοί που αξιολογήθηκαν είναι:

- απαρίθμηση μέσω author archives με αριθμητικά author IDs (/?author={id})
- Παθητική απαρίθμηση μέσω δημοσίων άρθρων (HTML metadata leakage)
- απαρίθμηση μέσω WordPress REST API (/wp-json/wp/v2/users)
- Έμμεση απαρίθμηση μέσω REST endpoint /wp-json/wp/v2/posts
- απαρίθμηση μέσω ανατροφοδότησης στη σελίδα σύνδεσης (/wp-login.php)
- έκθεση της διεπαφής XML-RPC (/xmlrpc.php)

Για κάθε μηχανισμό παρουσιάζονται τα αποτελέσματα ανά έκδοση και ακολουθεί αξιολόγηση του πρακτικού αντίκτυπου στην ασφάλεια.

4.2 Μηχανισμός Απαρίθμησης A — Απαρίθμηση Author ID (/?author={id})

4.2.1 Περιγραφή δοκιμής

Το διάνυσμα απαρίθμησης συγγραφέα (author enumeration) αξιολογήθηκε μέσω μη αυθεντικοποιημένων αιτημάτων που χρησιμοποιούν αριθμητικά author IDs:

- `http://<target>/?author=1`
- `http://<target>/?author=2`
- ...
- `http://<target>/?author=10`

Βασικός δείκτης επιτυχίας θεωρείται ένα HTTP redirect που αποκαλύπτει το URL author archive στη μορφή: `/author/<author-slug>/`. Η συγκεκριμένη τεχνική είναι ιδιαίτερα σημαντική διότι η σειριακή δοκιμή αριθμητικών IDs δεν έχει κάποιο κόστος και μπορεί να αυτοματοποιηθεί εύκολα με τη χρήση script:

```
for i in $(seq 1 10); do
    echo "--- author=$i ---"

    curl -s -o /dev/null -D - "http://<target>?author=$i" | egrep -i "HTTP/|Location:"
done
```

4.2.2 Αποτελέσματα — WordPress 4.7.1 (wpsite-v471.lan)

Αυτοματοποιημένη ανακάλυψη χρηστών στο Wordpress

```
(nikos@kali)-[~]
$ echo "=== v471 author id test ==="
for i in $(seq 1 10); do
  echo "— author=$i —"
  curl -s -o /dev/null -D - "http://wpsite-v471.lan/?author=$i" | egrep -i "HTTP/|Location:"
done

=== v471 author id test ===
— author=1 —
HTTP/1.1 301 Moved Permanently
Location: http://wpsite-v471.lan/author/admin_v471/
— author=2 —
HTTP/1.1 301 Moved Permanently
Location: http://wpsite-v471.lan/author/author_v471/
— author=3 —
HTTP/1.1 200 OK
— author=4 —
HTTP/1.1 301 Moved Permanently
Location: http://wpsite-v471.lan/author/editor_v471/
— author=5 —
HTTP/1.1 200 OK
— author=6 —
HTTP/1.1 404 Not Found
— author=7 —
HTTP/1.1 404 Not Found
— author=8 —
HTTP/1.1 404 Not Found
— author=9 —
HTTP/1.1 404 Not Found
— author=10 —
HTTP/1.1 404 Not Found
```

Εικόνα 13. Author enumeration v471

Η παλαιότερη εγκατάσταση WordPress αποκάλυψε έγκυρες ταυτότητες συγγραφέα μέσω redirects:

/?author=1 → 301 Moved Permanently → /author/admin_v471/

/?author=2 → 301 Moved Permanently → /author/author_v471/

/?author=4 → 301 Moved Permanently → /author/editor_v471/

Επιπλέον παρατηρήθηκαν οι ακόλουθες αποκρίσεις:

/?author=3 → 200 OK

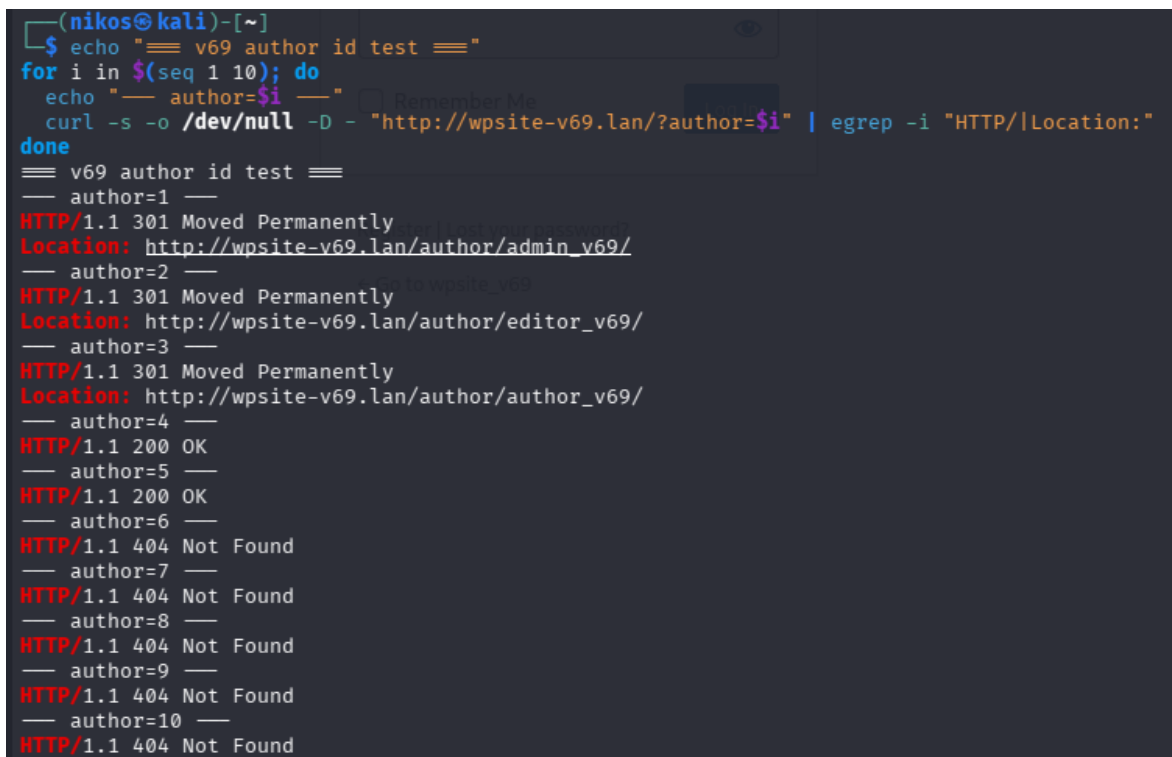
/?author=5 → 200 OK

/?author=6–10 → 404 Not Found

Τα redirects για τα IDs 1,2 και 4 αποτελούν άμεση απαρίθμηση χρηστών (direct user enumeration), καθώς αποκαλύπτονται author slugs χωρίς αυθεντικοποίηση.

Αξιοσημείωτο είναι ότι ορισμένα IDs επέστρεψαν 200 OK χωρίς εμφανές redirect στα headers. Αυτό υποδηλώνει ότι ο server επέστρεψε έγκυρη σελίδα για τα αιτήματα αυτά, ωστόσο απαιτείται πρόσθετος έλεγχος του περιεχομένου της απόκρισης ώστε να επιβεβαιωθεί εάν υπήρξε διαρροή στοιχείων χρήστη ή εάν επέστρεψε γενικευμένη απάντηση.

4.2.3 Αποτελέσματα — WordPress 6.9 (wpsite-v69.lan)



```
(nikos@kali)-[~]
$ echo "=== v69 author id test ==="
for i in $(seq 1 10); do
  echo "— author=$i —"
  curl -s -o /dev/null -D - "http://wpsite-v69.lan/?author=$i" | egrep -i "HTTP/|Location:"
done
=== v69 author id test ===
— author=1 —
HTTP/1.1 301 Moved Permanently
Location: http://wpsite-v69.lan/author/admin_v69/
— author=2 —
HTTP/1.1 301 Moved Permanently
Location: http://wpsite-v69.lan/author/editor_v69/
— author=3 —
HTTP/1.1 301 Moved Permanently
Location: http://wpsite-v69.lan/author/author_v69/
— author=4 —
HTTP/1.1 200 OK
— author=5 —
HTTP/1.1 200 OK
— author=6 —
HTTP/1.1 404 Not Found
— author=7 —
HTTP/1.1 404 Not Found
— author=8 —
HTTP/1.1 404 Not Found
— author=9 —
HTTP/1.1 404 Not Found
— author=10 —
HTTP/1.1 404 Not Found
```

Εικόνα 14. Author enumeration v69

Η σύγχρονη εγκατάσταση WordPress παρουσίασε επίσης άμεση απαρίθμηση μέσω redirects:

`/?author=1` → 301 Moved Permanently → `/author/admin_v69/`

`/?author=2` → 301 Moved Permanently → `/author/editor_v69/`

`/?author=3` → 301 Moved Permanently → `/author/author_v69/`

Επιπλέον παρατηρήθηκαν:

`/?author=4` → 200 OK

`/?author=5` → 200 OK

`/?author=6–10` → 404 Not Found

Το αποτέλεσμα αυτό δείχνει ότι το WordPress 6.9 εξακολουθεί να επιτρέπει απαρίθμηση μέσω author ID probing κάτω από baseline ρυθμίσεις.

4.2.4 Συγκριτική ανάλυση

Και οι δύο εκδόσεις WordPress εμφάνισαν άμεση απαρίθμηση μέσω redirects: Το WordPress 4.7.1 αποκάλυψε 3 author slugs με IDs 1,2 και 4 και το WordPress 6.9 αποκάλυψε 3 author slugs με IDs 1, 2 και 3.

Η παρουσία αποκρίσεων 404 Not Found για τα IDs 6–10 υποδηλώνει ότι ο χώρος έγκυρων user IDs είναι μικρός και μπορεί να διερευνηθεί πλήρως με περιορισμένο αριθμό αιτημάτων. Συνεπώς, ο μηχανισμός αυτός μπορεί να εκτελεστεί γρήγορα και αξιόπιστα, με ελάχιστη δικτυακή δραστηριότητα.

4.2.5 Αντίκτυπος στην ασφάλεια

Η απαρίθμηση μέσω author IDs αποτελεί υψηλού αντίκτυπου μηχανισμό διότι:

- δεν απαιτεί αυθεντικοποίηση
- μπορεί να αυτοματοποιηθεί εύκολα
- αποκαλύπτει σταθερά αναγνωριστικά μέσω προβλέψιμων URL δομών
- μειώνει σημαντικά την προσπάθεια για επόμενες επιθέσεις τύπου credential-based

Ακόμη και αν το εμφανιζόμενο author slug δεν ταυτίζεται πλήρως με το login username, μπορεί να χρησιμοποιηθεί ως αναγνωριστικό λογαριασμού που συσχετίζεται με άλλα σημεία έκθεσης, όπως το REST API.²

4.3 Μηχανισμός απαρίθμησης B — Παθητική Απαρίθμηση μέσω Δημοσίων Άρθρων (HTML Metadata Leakage)

4.3.1 Περιγραφή δοκιμής

Η παθητική απαρίθμηση δοκιμάστηκε μέσω λήψης δημοσίων σελίδων άρθρων και εξαγωγής πληροφοριών ταυτότητας χρήστη από τον HTML κώδικα. Σε αντίθεση με την ενεργή απαρίθμηση (π.χ. `/?author={id}` ή REST `/users`), η παθητική προσέγγιση δεν απαιτεί δοκιμή εξειδικευμένων endpoints και μοιάζει με φυσιολογική πλοήγηση χρήστη. Ο επιτιθέμενος αναζήτησε ένα δημοσιευμένο post και εξήγαγε στοιχεία σχετικά με το συγγραφέα όπως:

- author archive links (`/author/<slug>/`)
- ορατή αναφορά συγγραφέα (byline)
- semantic identifiers όπως author, vcard
- metadata όπως `rel="author"` και schema.org annotations

Για την εξαγωγή των παραπάνω στοιχείων χρησιμοποιήθηκαν τα commands από τον επιτιθέμενο:

- `curl -s "http://target/<POST-URL>" | grep -Eo 'href="[^\"]*/author/[^\"]*" | head -n 10`
- `curl -s "http://target/<POST-URL>" | grep -iE "author|rel=\"author\"|/author/" | head -n 40`

4.3.2 Αποτελέσματα — WordPress 4.7.1 (wpsite-v471.lan)

Η δημόσια σελίδα άρθρου <http://wpsite-v471.lan/author-post/> περιείχε απευθείας σύνδεσμο προς τον συγγραφέα: [href=http://wpsite-v471.lan/author/author_v471/](http://wpsite-v471.lan/author/author_v471/). Αυτό καταδεικνύει το γεγονός ότι το WordPress 4.7.1 εκθέτει το author slug (author_v471) μέσω της κανονικής απόδοσης δημοσιεύσεων, επιτρέποντας την παθητική ανακάλυψη ενός έγκυρου αναγνωριστικού χρήστη χωρίς να απαιτείται έλεγχο ταυτότητας.

Εκτός από την αποκάλυψη του URL του author archive, η σελίδα της ανάρτησης περιέχει επίσης ρητά μεταδεδομένα απόδοσης συγγραφέα (author attribution) μέσα στη δομή HTML. Η ταυτότητα του συγγραφέα ενσωματώνεται στην ενότητα entry-meta ως μέρος της γραμμής “byline” και επισημαίνεται με σημασιολογικές κλάσεις όπως author vcard. Αυτό επιτρέπει σε επιτιθέμενους να εξαγάγουν αυτόματα έγκυρα αναγνωριστικά συγγραφέα μέσω scraping δημόσιων σελίδων, χωρίς να απαιτείται probing σε ειδικά endpoints. Επιπλέον, η ίδια σελίδα περιλαμβάνει άμεσο σύνδεσμο προς το endpoint σύνδεσης του WordPress (/wp-login.php) μέσω του μηνύματος “You must be logged in”, γεγονός που διευκολύνει την ανακάλυψη endpoints αυθεντικοποίησης μέσω παθητικής παρατήρησης.

```
(nikos@kali)-[~]
$ curl -s "http://wpsite-v471.lan/author-post/" | grep -iE "author|rel=\"author\"|/author/" | head -n 40
<title>Author Post 6#8211; wpsite_v471</title>
<link rel="alternate" type="application/rss+xml" title="wpsite_v471 6raquo; Author Post Comments Feed" href="http://wpsite-v471.lan/author-post/feed/" />
<link rel="canonical" href="http://wpsite-v471.lan/author-post/" />
<link rel="alternate" type="application/json+oembed" href="http://wpsite-v471.lan/wp-json/oembed/1.0/embed?url=http%3A%2F%2Fwpsite-v471.lan%2Fauthor-post%2F" />
<div class="entry-meta"><span class="posted-on"><span class="screen-reader-text">Posted on</span> <a href="http://wpsite-v471.lan/author-post/" rel="bookmark"><time class="entry-date published updated" datetime="2026-01-26T19:19:28+02:00">26/01/2026</time></a></span><span class="byline"> by <span class="author vcard"><a class="url fn n" href="http://wpsite-v471.lan/author/author_v471/">author_v471</a></span></span></div>
<div class="entry-meta"><h1 class="entry-title">Author Post</h1>
</div>
<div class="entry-header">
<h3 id="reply-title" class="comment-reply-title">Leave a Reply <small><a rel="nofollow" id="cancel-comment-reply-link" href="/author-post/#respond" style="display:none;">Cancel reply</a></small></h3>
<p class="must-log-in">You must be <a href="http://wpsite-v471.lan/wp-login.php?redirect_to=http%3A%2F%2Fwpsite-v471.lan%2Fauthor-post%2F">logged in</a> to post a comment.</p>
</div>
</div>
</div>
```

Εικόνα 15. Στοιχεία Συγγραφέα v471

4.3.3 Αποτελέσματα — WordPress 6.9 (wpsite-v69.lan)

Η αντίστοιχη σελίδα άρθρου <http://wpsite-v69.lan/author-post/> αποκάλυψε επίσης το σύνδεσμο προς το συγγραφέα [href=http://wpsite-v69.lan/author/author_v69/](http://wpsite-v69.lan/author/author_v69/). Επομένως, το WordPress 6.9 διαρρέει επίσης αναγνωριστικά συντάκτη μέσω δημόσιων σελίδων ανάρτησης, επιτρέποντας την παθητική απαρίθμηση.

Επιπλέον, επειδή στην εγκατάσταση WordPress 6.9 χρησιμοποιήθηκε το “Astra” theme στην τελευταία έκδοσή του, τα HTML semantics identifiers περιλάμβαναν rel="author" και schema.org labeling όπου ο συγγραφέας αναφέρεται ως schema.org οντότητα:

- itemtype="https://schema.org/Person"
- itemprop="author" και itemprop="name"

Αυτό υποδεικνύει ότι τα σύγχρονα θέματα (themes) του WordPress ενδέχεται να εκθέτουν την ταυτότητα του συγγραφέα χρησιμοποιώντας δομημένα μεταδεδομένα (structured metadata), τα οποία έχουν σχεδιαστεί για σκοπούς SEO και σημασιολογικής ορθότητας. Από την οπτική ενός

επιπιθέμενου, αυτά τα χαρακτηριστικά αυξάνουν την αξιοπιστία της αυτοματοποιημένης εξαγωγής (scraping), επειδή η πληροφορία του συγγραφέα επισημαίνεται ξεκάθαρα και εμφανίζεται με συνεπή τρόπο μέσα στη σελίδα. Όπως και στην παλαιότερη έκδοση, η σελίδα αποκαλύπτει επίσης το endpoint σύνδεσης μέσω του μηνύματος που απαιτεί σύνδεση για την υποβολή σχολίου:

http://wpsite-v69.lan/wp-login.php?redirect_to=...

4.3.4 Αντίκτυπος στην ασφάλεια

Η παθητική απαρίθμηση είναι ιδιαίτερα πρακτική διότι:

- δεν απαιτεί endpoint probing
- μοιάζει με φυσιολογική περιήγηση
- διαρρέει σταθερά author slugs
- παρέχει υποψήφια usernames για επιθέσεις credentials

Επειδή η απόδοση συγγραφέα (author attribution) αποτελεί βασική λειτουργία σε CMS ιστοσελίδες, η πλήρης εξάλειψη της διαρροής είναι δύσκολη χωρίς μείωση λειτουργικότητας.

4.4 Μηχανισμός Απαρίθμησης C — Απαρίθμηση μέσω REST API (/wp-json/wp/v2/users)

4.4.1 Περιγραφή δοκιμής

Για να αξιολογηθεί αν το WordPress εκθέτει λίστα χρηστών μέσω REST API, εκτελέστηκε το ακόλουθο μη αυθεντικοποιημένο αίτημα **GET /wp-json/wp/v2/users?per_page=100**

Η απόκριση αναλύθηκε ως προς τη συμπεριφορά ελέγχου πρόσβασης (HTTP status), την επιστροφή λίστας χρηστών και την έκθεση πεδίων όπως id, slug, name. Για την εξαγωγή των παραπάνω στοιχείων χρησιμοποιήθηκε το command:^{9,10}

```
curl -i "http://target/wp-json/wp/v2/users?per_page=100" | head -n 60
```

4.4.2 Αποτελέσματα — WordPress 4.7.1 (wpsite-v471.lan)

Η δοκιμή στο endpoint χρηστών του WordPress REST API επέστρεψε επιτυχή απόκριση: HTTP status: 200 OK και οι συνολικοί χρήστες που επιστράφηκαν είναι 3 (X-WP-Total: 3).

```
(nikos@kali)-[~]
└─$ echo "=== v471 REST users ==="
curl -i "http://wpsite-v471.lan/wp-json/wp/v2/users?per_page=100" | head -n 60
=== v471 REST users ===
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %             0      Dload  Upload  Total  Spent    Left   Speed
100 1735    0 1735    0      0 6811    0 --:--:-- --:--:-- --:--:-- 6803
HTTP/1.1 200 OK
Server: nginx/1.28.0
Date: Tue, 27 Jan 2026 11:43:05 GMT
Content-Type: application/json; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
X-Robots-Tag: noindex
Link: <http://wpsite-v471.lan/wp-json/>; rel="https://api.w.org/"
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages
Access-Control-Allow-Headers: Authorization, Content-Type
X-WP-Total: 3
X-WP-TotalPages: 1
Allow: GET

[{"id":1,"name":"admin_v471","url":"","description":"","link":"http://wpsite-v471.lan/author/admin_v471/","slug":"admin_471","avatar_urls":{"24":"http://0.gravatar.com/avatar/f7148d72a30389130f3299c2b075adaf?s=24&mm6r=g","48":"http://0.gravatar.com/avatar/f7148d72a30389130f3299c2b075adaf?s=48&mm6r=g","96":"http://0.gravatar.com/avatar/f7148d72a30389130f3299c2b075adaf?s=96&mm6r=g"},"meta":{"_links":{"self":{"href":"http://wpsite-v471.lan/wp-json/wp/v2/users/1"},"collection":{"href":"http://wpsite-v471.lan/wp-json/wp/v2/users"}}}},{"id":2,"name":"author_v471","url":"","description":"","link":"http://wpsite-v471.lan/author/author_v471/","slug":"author_v471","avatar_urls":{"24":"http://0.gravatar.com/avatar/3b3658b504c4ad9f017104c20b87b74a?s=24&mm6r=g","48":"http://0.gravatar.com/avatar/3b3658b504c4ad9f017104c20b87b74a?s=48&mm6r=g","96":"http://0.gravatar.com/avatar/3b3658b504c4ad9f017104c20b87b74a?s=96&mm6r=g"},"meta":{"_links":{"self":{"href":"http://wpsite-v471.lan/wp-json/wp/v2/users/2"},"collection":{"href":"http://wpsite-v471.lan/wp-json/wp/v2/users"}}}},{"id":4,"name":"editor_v471","url":"","description":"","link":"http://wpsite-v471.lan/author/editor_v471/","slug":"editor_v471","avatar_urls":{"24":"http://0.gravatar.com/avatar/ca35f5ab088451c40b255bae24a33617?s=24&mm6r=g","48":"http://0.gravatar.com/avatar/ca35f5ab088451c40b255bae24a33617?s=48&mm6r=g","96":"http://0.gravatar.com/avatar/ca35f5ab088451c40b255bae24a33617?s=96&mm6r=g"},"meta":{"_links":{"self":{"href":"http://wpsite-v471.lan/wp-json/wp/v2/users/4"},"collection":{"href":"http://wpsite-v471.lan/wp-json/wp/v2/users"}}}]}
```

Εικόνα 16. User API response v471

Η JSON απόκριση αποκάλυψε δομημένα user objects που περιλάμβαναν πεδία όπως:

- id (αριθμητικό αναγνωριστικό χρήστη)
- name (εμφανιζόμενο όνομα)
- slug (αναγνωριστικό χρήστη σε URLs)
- link (σύνδεσμος προς author archive)
- avatar_urls (URLs avatar/Gravatar)
- _links (αναφορές προς REST πόρους χρηστών)

Οι χρήστες που εκτέθηκαν χωρίς αυθεντικοποίηση ήταν:

- id: 1, slug: admin_v471, link: /author/admin_v471/
- id: 2, slug: author_v471, link: /author/author_v471/
- id: 4, slug: editor_v471, link: /author/editor_v471/

Το αποτέλεσμα επιβεβαιώνει ότι στο WordPress 4.7.1 το endpoint /wp-json/wp/v2/users επιτρέπει άμεση απαρίθμηση χρηστών (direct user enumeration), καθώς αποκαλύπτει αναγνωριστικά χρηστών σε μία μόνο κλήση χωρίς να απαιτείται authentication.

4.4.3 Αποτελέσματα — WordPress 6.9 (wpsite-v69.lan)

Η σύγχρονη εγκατάσταση επέστρεψε επίσης επιτυχή απόκριση HTTP status: 200 OK και επίσης αριθμό συνολικών χρηστών: 3 (X-WP-Total: 3)

```

[nikos@kali]~$ echo "=== v471 REST users ==="
curl -i "http://wpsite-v69.lan/wp-json/wp/v2/users?per_page=100" | head -n 60

=== v471 REST users ===
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %             0         0     0    2272      0 --:--:-- --:--:-- --:--:-- 2271
HTTP/1.1 200 OK
Server: nginx/1.28.0
Date: Tue, 27 Jan 2026 12:00:05 GMT
Content-Type: application/json; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
X-Robots-Tag: noindex
Link: <http://wpsite-v69.lan/wp-json/>; rel="https://api.w.org/"
X-Content-Type-Options: nosniff
Access-Control-Expose-Headers: X-WP-Total, X-WP-TotalPages, Link
Access-Control-Allow-Headers: Authorization, X-WP-Nonce, Content-Disposition, Content-MD5, Content-Type
X-WP-Total: 3
X-WP-TotalPages: 1
Allow: GET
Vary: Origin

[{"id":1,"name":"admin_v69","url":"http://wpsite-v69.lan","description":"","link":"http://wpsite-v69.lan/author/admin_v69/","slug":"admin_v69","avatar_urls":{"24":"https://secure.gravatar.com/avatar/7413309477f57ec4c4acfc3fa083d7c7bb3a9d422494eed64c674d31cb3b7cd?s=24&mm6r=g","48":"https://secure.gravatar.com/avatar/7413309477f57ec4c4acfc3fa083d7c7bb3a9d422494eed64c674d31cb3b7cd?s=48&mm6r=g","96":"https://secure.gravatar.com/avatar/7413309477f57ec4c4acfc3fa083d7c7bb3a9d422494eed64c674d31cb3b7cd?s=96&mm6r=g"},"meta":{"_links":{"self":{"href":"http://wpsite-v69.lan/wp-json/wp/v2/users/1","targetHints":{"allow":["GET"]}}},"collection":{"href":"http://wpsite-v69.lan/wp-json/wp/v2/users/"}}},"id":3,"name":"author_v69","url":"","description":"","link":"http://wpsite-v69.lan/author/author_v69/","slug":"author_v69","avatar_urls":{"24":"https://secure.gravatar.com/avatar/ebe7945e328a6eef997654b1b1b94e99d94d8c3e2665b33df23e15b5278787?s=24&mm6r=g","48":"https://secure.gravatar.com/avatar/ebe7945e328a6eef997654b1b1b94e99d94d8c3e2665b33df23e15b5278787?s=48&mm6r=g","96":"https://secure.gravatar.com/avatar/ebe7945e328a6eef997654b1b1b94e99d94d8c3e2665b33df23e15b5278787?s=96&mm6r=g"},"meta":{"_links":{"self":{"href":"http://wpsite-v69.lan/wp-json/wp/v2/users/"}}},"collection":{"href":"http://wpsite-v69.lan/wp-json/wp/v2/users/"}}},"id":2,"name":"editor_v69","url":"","description":"","link":"http://wpsite-v69.lan/author/editor_v69/","slug":"editor_v69","avatar_urls":{"24":"https://secure.gravatar.com/avatar/1ec290616e6e3dc043e286f62b87c0f992c4889ead141b3e3dfc70d018a983e0?s=24&mm6r=g","48":"https://secure.gravatar.com/avatar/1ec290616e6e3dc043e286f62b87c0f992c4889ead141b3e3dfc70d018a983e0?s=48&mm6r=g","96":"https://secure.gravatar.com/avatar/1ec290616e6e3dc043e286f62b87c0f992c4889ead141b3e3dfc70d018a983e0?s=96&mm6r=g"},"meta":{"_links":{"self":{"href":"http://wpsite-v69.lan/wp-json/wp/v2/users/2","targetHints":{"allow":["GET"]}}},"collection":{"href":"http://wpsite-v69.lan/wp-json/wp/v2/users/"}}}]

```

Εικόνα 17. User API response v69

Η απόκριση περιείχε αντίστοιχα user objects, αποκαλύπτοντας τα id, slug, name, link, avatar urls. Οι χρήστες που εκτέθηκαν ήταν:

- id: 1, slug: admin_v69, link: /author/admin_v69/
- id: 2, slug: editor_v69, link: /author/editor_v69/
- id: 3, slug: author v69, link: /author/author v69/

Το αποτέλεσμα δείχνει ότι, υπό baseline ρυθμίσεις, το WordPress 6.9 επίσης διαρρέει λίστα χρηστών μέσω REST API χωρίς αυθεντικοποίηση.

4.4.4 Συσχέτιση με Author ID Enumeration

Κρίσιμο εύρημα αποτελεί η ισχυρή συσχέτιση μεταξύ των αποτελεσμάτων REST API και της απαρίθμησης μέσω author IDs. Στο WordPress 4.7.1, το REST API αποκάλυψε τους χρήστες 1 2 και 4, δηλαδή τα ίδια IDs που εμφάνισαν redirects στο /?author={id}. Στο WordPress 6.9, το REST API αποκάλυψε τους χρήστες 1, 2, και 3, τα οποία επίσης εμφάνισαν redirects. Έτσι, διαφορετικά διανύσματα απαρίθμησης συγκλίνουν στο ίδιο σύνολο χρηστών, αυξάνοντας την αξιοπιστία της απαρίθμησης από την πλευρά του επιτιθέμενου.

4.4.5 Αντίκτυπος στην ασφάλεια

Η απαρίθμηση μέσω REST API θεωρείται ιδιαίτερα επικίνδυνη επειδή:

- επιστρέφει αναγνωριστικά χρηστών σε δομημένη μορφή (machine-readable)
- μπορεί να πραγματοποιηθεί με ένα μόνο αίτημα
- κλιμακώνεται πολύ εύκολα σε αυτοματοποιημένες επιθέσεις
- παρέχει πολλαπλά πεδία χρήσιμα για στοχευμένη επίθεση (IDs, slugs, links, avatars)

Η δυνατότητα αυτή αποτελεί κρίσιμο κίνδυνο ανακάλυψης λογαριασμών, ειδικά σε περιπτώσεις όπου στόχος είναι password spraying ή brute-force προσπάθειες.

4.5 Μηχανισμός D — Έμμεση Απαρίθμηση μέσω REST /wp-json/wp/v2/posts

4.5.1 Περιγραφή δοκιμής

Για την αξιολόγηση έμμεσης απαρίθμησης, ο επιτιθέμενος πραγματοποίησε αίτημα στο endpoint των άρθρων: **GET /wp-json/wp/v2/posts?per_page=1**. Στόχος ήταν να διαπιστωθεί εάν η απόκριση διαρρέει αριθμητικά author IDs ("author": <id>) και άμεσες αναφορές σε author endpoints μέσω _links. Για την εξαγωγή των παραπάνω στοιχείων χρησιμοποιήθηκε το command:

```
curl -s "http://target/wp-json/wp/v2/posts?per_page=1" | head -n 40
```

4.5.2 Αποτελέσματα — WordPress 4.7.1 (wpsite-v471.lan)

Η απόκριση περιλάμβανε δημοσιευμένο άρθρο με:

- "slug": "author-post"
- "link": "http://wpsite-v471.lan/author-post/"
- "author": 2

[illegible]

Εικόνα 18. Posts API response v471

Ένα κρίσιμο σημείο είναι ότι στο `_links` υπήρχε αναφορά προς το αντίστοιχο user object: `_links → author → href: /wp-json/wp/v2/users/2`. Το αποτέλεσμα επιβεβαιώνει ότι user IDs μπορούν να εξαχθούν έμμεσα από δημόσια REST endpoints περιεχομένου, ακόμη κι αν η λίστα χρηστών στο `/users` περιοριστεί.

4.5.3 Αποτελέσματα — WordPress 6.9 (wpsite-v69.1an)

Η σύγχρονη εγκατάσταση επέστρεψε άρθρο με:

- "slug": "editor-post"
- "link": "http://wpsite-v69.lan/editor-post/"
- "author": 2

Στο `_links` υπήρχε επίσης άμεση αναφορά στον χρήστη: `_links → author → href: /wp-json/wp/v2/users/2`. Επιπλέον εύρημα στο WordPress 6.9 είναι ότι η απόκριση περιείχε πληροφορίες συγγραφέα σε επιπλέον metadata πεδία, όπως: `"uagb_author_info": { "display_name": "editor_v69", "author_link": ".../author/editor_v69/" }`. Το στοιχείο αυτό αποτελεί άμεση διαρροή ταυτότητας ενσωματωμένη στο post object, η οποία αυξάνει την ισχύ της απαρίθμησης μέσω περιεχομένου.

4.5.4 Αντίκτυπος στην ασφάλεια

Ο μηχανισμός αυτός αποδεικνύει ότι ακόμη και εάν οι διαχειριστές περιορίσουν το endpoint `/wp/v2/users`, η απαρίθμηση χρηστών μπορεί να παραμείνει εφικτή μέσω των posts, καθώς αποκαλύπτονται numeric author IDs και άμεσες συνδέσεις προς REST user objects.

Ειδικά στο WordPress 6.9, η ύπαρξη author-related πληροφοριών μέσα σε επιπλέον metadata (π.χ. πεδία θεμάτων ή plugins) αυξάνει περαιτέρω τον κίνδυνο, επειδή εκθέτει display names και author URLs μέσω ενός endpoint περιεχομένου.

4.6 Μηχανισμός Απαρίθμησης Ε — Απαρίθμηση μέσω Login Feedback (/wp-login.php)

4.6.1 Περιγραφή δοκιμής

Η απαρίθμηση μέσω login feedback δοκιμάστηκε συγκρίνοντας τα μηνύματα σφάλματος σύνδεσης μεταξύ:

1. Ανύπαρκτου username και λανθασμένου κωδικού
2. Έγκυρου username με λανθασμένο κωδικό

Αν τα μηνύματα διαφέρουν, τότε η login σελίδα επιτρέπει επιβεβαίωση ύπαρξης username.

4.6.2 Αποτελέσματα — WordPress 4.7.1 (wpsite-v471.lan)

Η εγκατάσταση WordPress 4.7.1 επέστρεψε διαφορετικά μηνύματα σφάλματος. Στην 1^η περίπτωση του ανύπαρκτου username η σελίδα επέστρεψε:

“ERROR: Invalid username. Lost your password?”

Ενώ στη 2^η περίπτωση του έγκυρου username (admin_v471) με λάθος κωδικό το μήνυμα ήταν:

“ERROR: The password you entered for the username admin_v471 is incorrect. Lost your password?”

Η διαφορά αυτή επιτρέπει στον επιτιθέμενο να επιβεβαιώσει εάν ένας λογαριασμός υπάρχει, άρα η σελίδα σύνδεσης λειτουργεί ως μηχανισμός άμεσης επικύρωσης usernames.

4.6.3 Αποτελέσματα — WordPress 6.9 (wpsite-v69.lan)

Η εγκατάσταση WordPress 6.9 επέστρεψε επίσης διακριτά μηνύματα, όπου στην 1^η περίπτωση του ανύπαρκτου username το μήνυμα ήταν:

“Error: The username <username> is not registered on this site. If you are unsure of your username, try your email address instead.”

ενώ στη 2^η περίπτωση του έγκυρου username με λάθος κωδικό:

“Error: The password you entered for the username admin_v69 is incorrect. Lost your password?”

Η σαφής διατύπωση αποκαλύπτει άμεσα την εγκυρότητα του username. Παρότι το κείμενο διαφέρει μεταξύ εκδόσεων, και οι δύο συμπεριφορές επιτρέπουν άμεση απαρίθμηση.

4.6.4 Αντίκτυπος στην ασφάλεια

Η απαρίθμηση μέσω login feedback αποτελεί ιδιαίτερα αποτελεσματικό μηχανισμό διότι:

- είναι ακριβής και αξιόπιστος
- υποστηρίζει αυτοματοποιημένη επικύρωση μεγάλων λιστών usernames
- μειώνει την αβεβαιότητα για επιθέσεις διαπιστευτηρίων
- επιβεβαιώνει έγκυρους λογαριασμούς ακόμη και αν άλλες τεχνικές έχουν περιοριστεί

Στην πράξη, μπορεί να χρησιμοποιηθεί για επιβεβαίωση usernames πριν την εκτέλεση password spraying.

4.7 Μηχανισμός Απαρίθμησης F — Έκθεση XML-RPC (/xmlrpc.php)

4.7.1 Περιγραφή δοκιμής

Για την αξιολόγηση της έκθεσης της διεπαφής XML-RPC, εκτελέστηκε το αίτημα: **GET /xmlrpc.php** με τη χρήση του ακόλουθου command από τον επιτιθέμενο:

```
curl -i "http://target/xmlrpc.php" | head -n 40
```

Η απόκριση αναλύθηκε ως προς τους κωδικούς κατάστασης και ενδείξεις ενεργοποίησης του XML-RPC.

4.7.2 Αποτελέσματα και στις δύο εκδόσεις

Και οι δυο εγκαταστάσεις του WordPress επέστρεψαν:

- HTTP status: 405 Method Not Allowed
- Header: Allow: POST
- Body: "XML-RPC server accepts POST requests only."

```
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
             Dload  Upload    Total      Spent      Left     Speed
100    42    0    42    0    0    157      0  --:--:-- --:--:-- --:--:--    157
HTTP/1.1 405 Method Not Allowed
Server: nginx/1.28.0
Date: Tue, 27 Jan 2026 17:05:37 GMT
Content-Type: text/plain; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Allow: POST

XML-RPC server accepts POST requests only.
```

Εικόνα 19. Έκθεση xmlrpc.php

Η συμπεριφορά αυτή υποδεικνύει ότι το endpoint υπάρχει και είναι ενεργό και προσβάσιμο, παρότι δεν αποδέχεται GET αιτήματα.

4.7.3 Αντίκτυπος και ερμηνεία

Το XML-RPC δεν αποκάλυψε άμεσα usernames μέσω ενός GET αιτήματος. Ωστόσο θεωρείται σημαντικό από πλευράς ασφάλειας διότι:

- επιβεβαιώνει ότι μια legacy διεπαφή είναι ενεργοποιημένη
- αυξάνει την επιφάνεια επίθεσης με επιπλέον endpoints
- ενδέχεται να υποστηρίξει επιθέσεις επόμενων σταδίων μέσω POST αιτημάτων, ανάλογα με τη ρύθμιση

Για τους σκοπούς της εργασίας, το XML-RPC ταξινομείται κυρίως ως “εκτεθειμένη διεπαφή” (exposed interface) και όχι ως άμεσος μηχανισμός απαρίθμησης.

4.8 Σύνοψη Ευρημάτων

4.8.1 Συνολική εικόνα επιτυχίας απαρίθμησης

Τα πειράματα έδειξαν ότι πολλαπλοί τρόποι απαρίθμησης είναι επιτυχείς και στις δύο εκδόσεις υπό baseline συνθήκες. Ο Πίνακας 4.1 συνοψίζει τα αποτελέσματα.

Μηχανισμός Απαρίθμησης	WP 4.7.1	WP 6.9	Κατηγορία	Διαρροή πληροφορίας
<code>/?author={id}</code>	✓	✓	Άμεση	Author slugs + έγκυρα IDs
Παθητικό HTML άρθρων	✓	✓	Άμεση	Author slug + author archive link
REST <code>/wp/v2/users</code>	✓	✓	Άμεση	IDs, slugs, names, links, avatars
REST <code>/wp/v2/posts</code>	✓	✓	Έμμεση	Numeric author ID + links προς users (+ extra metadata στο v6.9)
Login feedback	✓	✓	Άμεση	επικύρωση usernames
XML-RPC exposure	✓	✓	Έκθεση	XML-RPC ενεργό (Allow: POST)

4.8.2 Ορατότητα ρόλων και μερική έκθεση

Κάθε WordPress site περιλάμβανε πέντε ρόλους (Administrator, Editor, Author, Contributor, Subscriber). Ωστόσο, τα αποτελέσματα της απαρίθμησης δεν αποκάλυψαν και τους πέντε λογαριασμούς. Και στις δύο εκδόσεις του WordPress, εκτέθηκαν μόνο 3 χρήστες μέσω REST API και redirects.

Το εύρημα αυτό υποδεικνύει ότι, υπό τις συγκεκριμένες ρυθμίσεις, ορισμένοι ρόλοι (όπως Contributor και Subscriber) ενδέχεται να μην εκτίθενται μέσω author enumeration ή REST API listing εάν δεν σχετίζονται με δημόσιο περιεχόμενο. Αυτό συμφωνεί με τη γενικότερη παρατήρηση ότι το WordPress εκθέτει “authors” συχνότερα από λογαριασμούς χαμηλών δικαιωμάτων.^{5,15}

Ωστόσο, η απαρίθμηση μέσω login feedback δεν εξαρτάται από δημοσιευμένο περιεχόμενο και θεωρητικά μπορεί να επαληθεύσει οποιονδήποτε λογαριασμό, εφόσον ο επιτιθέμενος διαθέτει υποψήφια usernames.

4.8.3 Συνολικό συμπέρασμα σύγκρισης

Το βασικό συμπέρασμα του κεφαλαίου είναι ότι τόσο η παλαιότερη όσο και η σύγχρονη εγκατάσταση WordPress παρουσίασαν άμεση απαρίθμηση χρηστών υπό baseline συνθήκες. Παρότι το WordPress 6.9 είναι πολύ νεότερο από το 4.7.1, τα πειράματα έδειξαν ότι βασικές συμπεριφορές απαρίθμησης εξακολουθούν να υφίστανται, κυρίως μέσω:

- redirects σε author archive URLs με αριθμητικά IDs
- επιστροφής λίστας χρηστών μέσω REST API
- διακριτών login error μηνυμάτων που αποκαλύπτουν την εγκυρότητα username

Συνεπώς, και οι δύο εκδόσεις παρέχουν αξιόπιστους τρόπους στον επιτιθέμενο για την ανακάλυψη έγκυρων αναγνωριστικών χρηστών, γεγονός που μπορεί να αυξήσει σημαντικά την αποτελεσματικότητα επόμενων επιθέσεων τύπου credential-based.⁶

Κεφάλαιο 5 — Τεχνικές Μετριασμού και Αξιολόγηση Αποτελεσματικότητας

5.1 Εισαγωγή

Σε αντίθεση με το Κεφάλαιο 4, το οποίο παρουσιάζει την baseline συμπεριφορά των εκδόσεων WordPress, το παρόν κεφάλαιο επικεντρώνεται στην πρακτική εφαρμογή τεχνικών μετριασμού και στην εμπειρική αξιολόγηση της αποτελεσματικότητάς τους μέσω αυτοματοποιημένου εργαλείου απαρίθμησης.

Η πειραματική ανάλυση που παρουσιάστηκε στο προηγούμενο κεφάλαιο κατέδειξε ότι η απαρίθμηση χρηστών στο WordPress μπορεί να επιτευχθεί μέσω πολλαπλών ανεξάρτητων μηχανισμών, συμπεριλαμβανομένων της χειραγώγησης παραμέτρων URL, των REST API endpoints, της παθητικής διαρροής μεταδεδομένων HTML και των μηχανισμών ανατροφοδότησης κατά την αυθεντικοποίηση. Αν και αυτές οι συμπεριφορές δεν αποτελούν απαραίτητα σφάλματα υλοποίησης, ο συνδυασμός τους μειώνει σημαντικά το απαιτούμενο επίπεδο προσπάθειας για την ανακάλυψη λογαριασμών και την εκτέλεση επακόλουθων επιθέσεων.

Το παρόν κεφάλαιο παρουσιάζει συγκεκριμένες τεχνικές μετριασμού (mitigation technics) που εφαρμόστηκαν τόσο στο επίπεδο του web server (Nginx) όσο και στο επίπεδο της εφαρμογής (WordPress). Κάθε μέτρο αντιστοιχίζεται άμεσα σε ένα αναγνωρισμένο μηχανισμό απαρίθμησης και αξιολογείται μέσω ελεγχόμενης επαναδοκιμής στο ίδιο εργαστηριακό περιβάλλον.^{7,8}

5.2 Μετριασμός σε Επίπεδο Διακομιστή με Παραμετροποίηση του Nginx

5.2.1 Αποκλεισμός Απαρίθμησης Author ID μέσω Παραμέτρων URL

Ένας από τους πιο αξιόπιστους μηχανισμούς απαρίθμησης που παρατηρήθηκαν ήταν η χρήση της παραμέτρου `?author={id}`. Για τον μετριασμό αυτής της συμπεριφοράς στο πρωιμότερο δυνατό στάδιο, εφαρμόστηκε φιλτράρισμα αιτημάτων στο επίπεδο του Nginx. Ο ακόλουθος κανόνας προστέθηκε στη ρύθμιση του virtual host του WordPress:

```
if ($args ~* "(^|&)author=\d+(&|$)") {  
    return 404;  
}
```

Ο κανόνας αυτός αποτρέπει την επεξεργασία αιτημάτων που περιέχουν αριθμητικά αναγνωριστικά συγγραφέα, επιστρέφοντας ομοιόμορφη απόκριση ανεξαρτήτως του αν το αναγνωριστικό αντιστοιχεί σε έγκυρο χρήστη. Ως αποτέλεσμα, οι επιτιθέμενοι δεν μπορούν πλέον να συναγάγουν την ύπαρξη χρηστών μέσω HTTP status codes ή redirects.^{9,10}

5.2.2 Περιορισμός Απαρίθμησης Χρηστών μέσω REST API

Η άμεση απαρίθμηση χρηστών μέσω του REST API endpoint `/wp-json/wp/v2/users` ήταν πλήρως επιτυχής και στις δύο εκδόσεις του WordPress πριν τον μετριασμό. Για την εξάλειψη αυτού του φαινομένου, εφαρμόστηκε ο ακόλουθος περιορισμός στο virtual host του Nginx:

```
location ~* ^/wp-json/wp/v2/users(/|$) {  
    return 403;  
}
```

Η ρύθμιση αυτή αποκλείει την μη αυθεντικοποιημένη πρόσβαση τόσο στη λίστα χρηστών όσο και σε μεμονωμένους πόρους χρηστών, αποτρέποντας την αποκάλυψη αναγνωριστικών χρηστών, slugs και μεταδεδομένων προφίλ στο επίπεδο μεταφοράς.

5.2.3 Απενεργοποίηση XML-RPC

Παρότι το XML-RPC δεν αποκάλυψε άμεσα πληροφορίες χρηστών κατά τη διάρκεια των δοκιμών, η παρουσία του αυξάνει τη συνολική επιφάνεια επίθεσης και ιστορικά έχει καταχραστεί για επιθέσεις αυθεντικοποίησης. Για τον λόγο αυτό, το XML-RPC απενεργοποιήθηκε πλήρως στο επίπεδο του web server με το πρόσθεση του παρακάτω configuration στο virtual host:

```
location = /xmlrpc.php {  
    return 403;  
}
```

Η ρύθμιση αυτή διασφαλίζει ότι τα αιτήματα XML-RPC απορρίπτονται πριν φτάσουν στην εφαρμογή WordPress.

5.2.4 Περιορισμός του Ρυθμού Προσπαθειών Αυθεντικοποίησης

Για τον μετριασμό της πρακτικής εκμετάλλευσης των απαριθμημένων usernames, εφαρμόστηκε περιορισμός ρυθμού (rate limiting) στο endpoint σύνδεσης του WordPress. Το μέτρο αυτό περιορίζει τον αριθμό προσπαθειών αυθεντικοποίησης που μπορεί να πραγματοποιήσει ένας επιτιθέμενος από μία συγκεκριμένη IP διεύθυνση. Ο ακόλουθος κανόνας ορίστηκε στη γενική ρύθμιση του Nginx (`/etc/nginx/nginx.conf`) στο πλαίσιο του http:

```
limit_req_zone $binary_remote_addr zone=login_zone:10m rate=10r/m;
```

Στη συνέχεια, το endpoint σύνδεσης προστατεύτηκε με τη προσθήκη του παρακάτω κανόνα στο virtual host:

```
location = /wp-login.php {
```

```
limit_req zone=login_zone burst=20 nodelay;
include snippets/fastcgi-php.conf;
fastcgi_pass unix:/run/php/php8.3-fpm.sock;
}
```

Όταν το καθορισμένο όριο ξεπεραστεί, ο Nginx επιστρέφει HTTP 503 Service Temporarily Unavailable, που αποτελεί το προεπιλεγμένο status απόρριψης του module limit_req. Η συμπεριφορά αυτή παρατηρήθηκε με συνέπεια κατά την επαναδοκιμή.

[illegible]

Εικόνα 20. Rate-limit authentication

5.3 Μετριασμός σε Επίπεδο Εφαρμογής με Χρήση WordPress

5.3.1 Περιορισμός REST API Endpoints Χρηστών

Για την ενίσχυση της άμυνας σε βάθος (defense in depth), η πρόσβαση στα REST API endpoints χρηστών περιορίστηκε και στο επίπεδο της εφαρμογής WordPress μέσω must-use plugin. Δημιουργήθηκε το αρχείο **hardening.php** στο directory **/wp-content/mu-plugins/** με τον εξής κώδικα:

```
add_filter('rest_authentication_errors', function ($result) {  
    if (!empty($result)) return $result;  
  
    $uri = $_SERVER['REQUEST_URI'] ?? '';  
    if (preg_match('#^/wp-json/wp/v2/users/(/|\\?|\\$)#i', $uri)) {  
        if (!is_user_logged_in() || !current_user_can('list_users')) {  
            return new WP_Error('rest_forbidden', 'Forbidden', ['status' => 403]);  
        }  
    }  
    return $result;  
});
```

Η προσέγγιση αυτή διασφαλίζει ότι, ακόμη και αν παρακαμφθεί ή τροποποιηθεί η ρύθμιση του Nginx, οι μη αυθεντικοποιημένοι χρήστες δεν μπορούν να ανακτήσουν αντικείμενα χρηστών μέσω του REST API.

5.3.2 Ενοποίηση Μηνυμάτων Σφαλμάτων Αυθεντικοποίησης

Τα μηνύματα σφάλματος κατά τη σύνδεση διέφεραν ανάλογα με την ύπαρξη του username, επιτρέποντας την επιβεβαίωση έγκυρων λογαριασμών. Για τον μετριασμό αυτού του φαινομένου, τα μηνύματα ενοποιήθηκαν. Στο **hardening.php** προστέθηκε ο κώδικας:

```
add_filter('login_errors', function () {  
    return 'Error: Invalid login credentials.';  
});
```

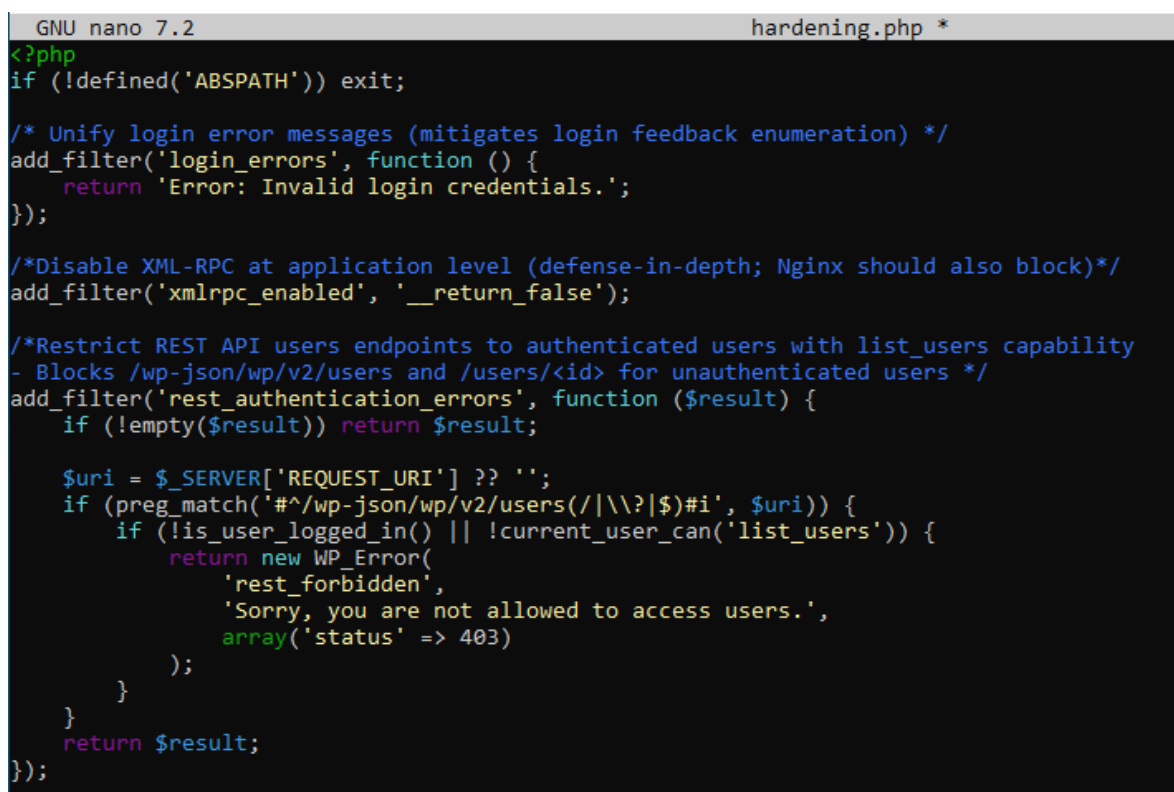
Η αλλαγή αυτή αποτρέπει τους επιτιθέμενους από το να διακρίνουν μεταξύ ανύπαρκτων usernames και λανθασμένων κωδικών πρόσβασης.

5.3.3 Απενεργοποίηση XML-RPC σε Επίπεδο Εφαρμογής

Ως επιπλέον μέτρο προστασίας, το XML-RPC απενεργοποιήθηκε και στο επίπεδο της εφαρμογής WordPress με τη προσθήκη του εξής κώδικα στο hardening.php:

```
add_filter('xmlrpc_enabled', '__return_false');
```

Η πλεοναστική αυτή προσέγγιση διασφαλίζει ότι η λειτουργικότητα XML-RPC παραμένει απενεργοποιημένη ακόμη και αν μεταβληθούν οι κανόνες του web server.



```
GNU nano 7.2                                     hardening.php *
<?php
if (!defined('ABSPATH')) exit;

/* Unify login error messages (mitigates login feedback enumeration) */
add_filter('login_errors', function () {
    return 'Error: Invalid login credentials.';
});

/*Disable XML-RPC at application level (defense-in-depth; Nginx should also block)*/
add_filter('xmlrpc_enabled', '__return_false');

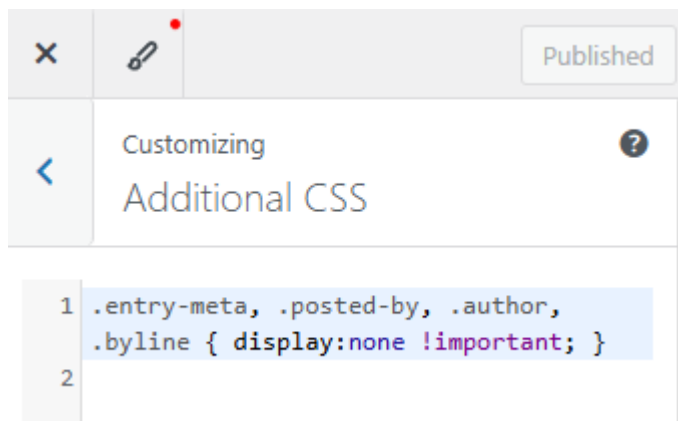
/*Restrict REST API users endpoints to authenticated users with list_users capability
- Blocks /wp-json/wp/v2/users and /users/<id> for unauthenticated users */
add_filter('rest_authentication_errors', function ($result) {
    if (!empty($result)) return $result;

    $uri = $_SERVER['REQUEST_URI'] ?? '';
    if (preg_match('#^/wp-json/wp/v2/users(/|\?|)$#i', $uri)) {
        if (!is_user_logged_in() || !current_user_can('list_users')) {
            return new WP_Error(
                'rest_forbidden',
                'Sorry, you are not allowed to access users.',
                array('status' => 403)
            );
        }
    }
    return $result;
});
```

Εικόνα 21. hardening.php

5.4 Μετριάσμός Παθητικής Απαρίθμησης μέσω Δημόσιου Περιεχομένου

Η παθητική απαρίθμηση μέσω απόδοσης συγγραφέα στις δημοσιευμένες αναρτήσεις μετριάστηκε με την απόκρυψη μεταδεδομένων συγγραφέα σε επίπεδο θέματος (theme), προσθέτοντας κώδικα CSS στο appearance customizer. Η προσέγγιση αυτή μειώνει την έκθεση author slugs, author archive URLs και δομημένων μεταδεδομένων (π.χ. schema.org/Person) που διευκολύνουν την αυτοματοποιημένη εξαγωγή δεδομένων.



Εικόνα 22. Theme Customizer

Παρότι το μέτρο αυτό δεν εξαλείφει πλήρως την απαρίθμηση χρηστών, αυξάνει σημαντικά το απαιτούμενο επίπεδο προσπάθειας για την παθητική ανακάλυψη αναγνωριστικών.

5.5 Αξιολόγηση Αποτελεσματικότητας

5.5.1 Μεθοδολογία

Για την αξιολόγηση της αποτελεσματικότητας των μέτρων, όλες οι δοκιμές απαρίθμησης που πραγματοποιήθηκαν στη βασική αξιολόγηση επαναλήφθηκαν χρησιμοποιώντας το αυτοματοποιημένο εργαλείο `wp_enum_lab_v5.py` από το virtual machine του επιτιθέμενου, το οποίο εκτελεί συνδυασμένες τεχνικές απαρίθμησης. Το εργαλείο ελέγχει:

- Author ID enumeration
- REST API (users)
- REST API (posts)
- Passive HTML enumeration
- XML-RPC

Η εκτέλεση του εργαλείου πριν και μετά την εφαρμογή των mitigation μέτρων επέτρεψε την άμεση σύγκριση της εκτεθειμένης πληροφορίας μεταξύ των δύο καταστάσεων. Αναλύθηκαν οι κωδικοί απόκρισης, οι χρόνοι αιτημάτων και τα αρχεία καταγραφής του διακομιστή. Τα μέτρα σκλήρυνσης εφαρμοστήκαν μόνο στην έκδοση v69, ώστε να εντοπιστεί η αποτελεσματικότητα τους.

5.5.2 Αποτελέσματα

WordPress 4.7.1 (Μη προστατευμένο)

Η εγκατάσταση WordPress 4.7.1 παρουσίασε πλήρη έκθεση σε όλους τους εξεταζόμενους μηχανισμούς απαρίθμησης. Η απαρίθμηση Author ID επέστρεφε HTTP 301 redirects, αποκαλύπτοντας άμεσα τα usernames. Το REST API endpoint `/wp-json/wp/v2/users` ανταποκρινόταν με HTTP 200 και παρείχε πλήρη λίστα χρηστών. Επιπλέον, το XML-RPC παρέμενε

ενεργό, γεγονός που υποδηλώνει πιθανή περαιτέρω επιφάνεια επίθεσης. Η παθητική ανάλυση HTML επιβεβαίωσε ότι όλα τα usernames ήταν διαθέσιμα μέσω author links στις δημόσιες σελίδες άρθρων.

```
===== SUMMARY =====
Author ID redirects:
  id=1 slug=admin_v471 location=http://wpsite-v471.lan/author/admin_v471/
  id=2 slug=author_v471 location=http://wpsite-v471.lan/author/author_v471/
  id=4 slug=editor_v471 location=http://wpsite-v471.lan/author/editor_v471/

REST users:
  id=1 slug=admin_v471 name=admin_v471 link=http://wpsite-v471.lan/author/admin_v471/
  id=2 slug=author_v471 name=author_v471 link=http://wpsite-v471.lan/author/author_v471/
  id=4 slug=editor_v471 name=editor_v471 link=http://wpsite-v471.lan/author/editor_v471/

Author IDs from REST posts:
  author_id=1
  author_id=2
  author_id=4

Passive HTML slugs:
  slug=admin_v471
  slug=author_v471
  slug=editor_v471
```

Εικόνα 23. Wordpress v471 output

WordPress 6.9 (Προστατευμένο σύστημα)

Η εγκατάσταση WordPress 6.9 παρουσίασε σημαντική μείωση των άμεσων μηχανισμών απαρίθμησης μετά την εφαρμογή μέτρων σκλήρυνσης. Η απαρίθμηση Author ID επέστρεφε ομοιόμορφα HTTP 404, αποτρέποντας την αναγνώριση έγκυρων χρηστών. Το REST API /users ήταν περιορισμένο, επιστρέφοντας HTTP 403 Forbidden, ενώ και το XML-RPC είχε απενεργοποιηθεί. Ωστόσο, παρέμεινε έμμεση διαρροή πληροφορίας. Το REST endpoint /posts συνέχιζε να αποκαλύπτει numeric author IDs, ενώ η παθητική ανάλυση HTML αποκάλυπτε author slugs μέσω δημόσιου περιεχομένου.

```

===== SUMMARY =====
Author ID redirects:

REST users:

Author IDs from REST posts:
  author_id=1
  author_id=2
  author_id=3

Passive HTML slugs:
  slug=admin_v69
  slug=author_v69
  slug=editor_v69

```

Εικόνα 24. Wordpress v69 output

Ο Πίνακας 4.2 συνοψίζει τα αποτελέσματα Enumeration πριν και μετά τις τεχνικές μετριασμού

Μηχανισμός Απαρίθμησης	WP 4.7.1 (baseline)	WP 6.9 (hardened)
<code>/?author=</code>	301 → usernames	404 (μπλοκαρισμένο)
Παθητικό HTML άρθρων	usernames εκτεθειμένα	usernames εκτεθειμένα
REST <code>/wp/v2/users</code>	200 → πλήρη δεδομένα	403 (μπλοκαρισμένο)
REST <code>/wp/v2/posts</code>	author IDs εκτεθειμένα	author IDs εκτεθειμένα
XML-RPC exposure	ενεργό (405)	μπλοκαρισμένο (403)

5.5.3 Συζήτηση

Συνοψίζοντας μετά την εφαρμογή των μέτρων μετριασμού

- Τα αιτήματα `/?author={id}` επέστρεφαν σταθερά 404
- Η πρόσβαση στο `/wp-json/wp/v2/users` επέστρεφε 403 Forbidden
- Τα αιτήματα XML-RPC απορρίπτονταν πριν την επεξεργασία από την εφαρμογή
- Οι υψηλής συχνότητας προσπάθειες σύνδεσης περιορίζονταν, με τα υπεράριθμα requests να επιστρέφουν 503
- Τα μηνύματα σφάλματος σύνδεσης δεν αποκάλυπταν πλέον την εγκυρότητα usernames

Τα αρχεία καταγραφής του Nginx επιβεβαίωσαν την ενεργή επιβολή του rate limiting μέσω σχετικών εγγραφών.^{7,8}

```
2026/01/28 12:52:46 [error] 1428#1428: *220 limiting requests, excess: 20.736 by zone "login_zone", client: 192.168.178.5, server: wpsite-v69.lan, request: "POST /wp-login.php HTTP/1.1", host: "wpsite-v69.lan"
2026/01/28 12:52:46 [error] 1428#1428: *221 limiting requests, excess: 20.735 by zone "login_zone", client: 192.168.178.5, server: wpsite-v69.lan, request: "POST /wp-login.php HTTP/1.1", host: "wpsite-v69.lan"
2026/01/28 12:52:46 [error] 1428#1428: *222 limiting requests, excess: 20.734 by zone "login_zone", client: 192.168.178.5, server: wpsite-v69.lan, request: "POST /wp-login.php HTTP/1.1", host: "wpsite-v69.lan"
2026/01/28 12:52:46 [error] 1428#1428: *223 limiting requests, excess: 20.733 by zone "login_zone", client: 192.168.178.5, server: wpsite-v69.lan, request: "POST /wp-login.php HTTP/1.1", host: "wpsite-v69.lan"
2026/01/28 12:52:46 [error] 1428#1428: *224 limiting requests, excess: 20.731 by zone "login_zone", client: 192.168.178.5, server: wpsite-v69.lan, request: "POST /wp-login.php HTTP/1.1", host: "wpsite-v69.lan"
```

Εικόνα 25. Nginx error.log

Τα εφαρμοσμένα μέτρα μείωσαν σημαντικά τόσο τις άμεσες όσο και τις έμμεσες δυνατότητες απαρίθμησης χρηστών. Παρότι η έμμεση διαρροή μέσω REST endpoints αναρτήσεων είναι δύσκολο να εξαιρεθεί χωρίς να επηρεαστεί η βασική λειτουργικότητα, τα πολυεπίπεδα μέτρα αυξάνουν ουσιαστικά το κόστος και την πολυπλοκότητα για έναν επιτιθέμενο.

Κεφάλαιο 6 — Συμπεράσματα και Μελλοντική Έρευνα

6.1 Συμπεράσματα

Η παρούσα διπλωματική εργασία εξέτασε το ζήτημα της απαρίθμησης χρηστών (user enumeration) σε Συστήματα Διαχείρισης Περιεχομένου (Content Management Systems – CMS), με ιδιαίτερη έμφαση στο WordPress. Μέσω της δημιουργίας ενός ελεγχόμενου εργαστηριακού περιβάλλοντος και της ανάλυσης τόσο μιας παλαιότερης έκδοσης (WordPress 4.7.1) όσο και μιας σύγχρονης έκδοσης (WordPress 6.9), αποδείχθηκε ότι η απαρίθμηση χρηστών παραμένει ένα επίκαιρο ζήτημα ασφάλειας ανεξαρτήτως της γενιάς της πλατφόρμας.^{11,15}

Τα πειραματικά αποτελέσματα έδειξαν ότι πολλαπλοί μηχανισμοί απαρίθμησης είναι ενεργοί από προεπιλογή, συμπεριλαμβανομένης της άμεσης απαρίθμησης μέσω των REST API endpoints χρηστών, της έμμεσης απαρίθμησης μέσω μεταδεδομένων δημοσιεύσεων, των παλαιότερων τεχνικών μέσω παραμέτρων URL, της παθητικής αποκάλυψης ταυτότητας μέσω HTML στοιχείων συγγραφέα, καθώς και της ανατροφοδότησης κατά τη διαδικασία αυθεντικοποίησης. Παρότι αρκετοί από αυτούς τους μηχανισμούς έχουν σχεδιαστεί για λόγους χρηστικότητας, διαλειτουργικότητας ή βελτιστοποίησης για μηχανές αναζήτησης (SEO), ο συνδυασμός τους μειώνει σημαντικά την απαιτούμενη προσπάθεια για τον εντοπισμό έγκυρων λογαριασμών χρηστών από έναν επιτιθέμενο.²

Ένα βασικό συμπέρασμα της παρούσας μελέτης είναι ότι η απαρίθμηση χρηστών δεν αποτελεί αποτέλεσμα ενός μεμονωμένου ευπαθούς σημείου, αλλά προκύπτει από αλληλεπικαλυπτόμενες σχεδιαστικές επιλογές σε διαφορετικά αρχιτεκτονικά επίπεδα. Το χαρακτηριστικό αυτό καθιστά την πλήρη εξάλειψη της απαρίθμησης ιδιαίτερα δύσκολη και υπογραμμίζει τη σημασία της πολυεπίπεδης άμυνας.

6.2 Αποτελεσματικότητα Μέτρων Σκλήρυνσης (Hardening)

Η εφαρμογή τεχνικών μετριασμού τόσο στο επίπεδο του web server όσο και στο επίπεδο της εφαρμογής μείωσε σημαντικά την επιφάνεια επίθεσης. Οι μηχανισμοί ελέγχου που υλοποιήθηκαν στον Nginx απέτρεψαν επιτυχώς παλαιότερες τεχνικές απαρίθμησης, περιόρισαν την πρόσβαση σε ευαίσθητα REST API endpoints και επέβαλαν περιορισμούς στον ρυθμό προσπαθειών αυθεντικοποίησης. Τα μέτρα σκλήρυνσης στο WordPress λειτούργησαν συμπληρωματικά, ενοποιώντας τα μηνύματα σφαλμάτων σύνδεσης, περιορίζοντας την πρόσβαση στο REST API βάσει δικαιωμάτων χρήστη και απενεργοποιώντας μη απαραίτητες λειτουργίες, όπως το XML-RPC.

Η αποτελεσματικότητα των μέτρων αξιολογήθηκε μέσω συστηματικής επανεκτέλεσης των ίδιων δοκιμών και εργαλείων. Τα αποτελέσματα κατέδειξαν σταθερό αποκλεισμό τεχνικών απαρίθμησης που προηγουμένως ήταν επιτυχείς, καθώς και σημαντική αύξηση του χρόνου και της προσπάθειας που απαιτείται για αυτοματοποιημένες επιθέσεις. Αξίζει να σημειωθεί ότι οι βελτιώσεις αυτές επιτεύχθηκαν χωρίς να επηρεαστεί αρνητικά η κανονική λειτουργία για τους νόμιμους χρήστες.

6.3 Περιορισμοί

Παρά την αποτελεσματικότητα των εφαρμοσμένων μέτρων, ορισμένοι περιορισμοί εξακολουθούν να υφίστανται. Η έμμεση απαρίθμηση χρηστών μέσω endpoints που σχετίζονται με περιεχόμενο, όπως το `/wp-json/wp/v2/posts`, συνεχίζει να αποκαλύπτει αριθμητικά αναγνωριστικά συγγραφέων ως μέρος του μοντέλου περιεχομένου του WordPress. Η πλήρης εξάλειψη αυτής της διαρροής θα απαιτούσε πιο παρεμβατικές αλλαγές, οι οποίες ενδέχεται να επηρεάσουν αρνητικά τη βασική λειτουργικότητα ή τη συμβατότητα με τρίτες εφαρμογές.

Επιπλέον, το πειραματικό περιβάλλον περιορίστηκε σε μία πλατφόρμα CMS και σε μία συγκεκριμένη ρύθμιση web server. Τα αποτελέσματα ενδέχεται να διαφέρουν σε περιβάλλοντα που χρησιμοποιούν διαφορετικούς web servers, παρόχου φιλοξενίας ή πρόσθετα ασφάλειας. Τέλος, η παρούσα μελέτη δεν εξέτασε μηχανισμούς απαρίθμησης που εισάγονται από πρόσθετα (plugins) ή προσαρμοσμένα θέματα (themes), τα οποία μπορεί να μεταβάλλουν σημαντικά την επιφάνεια επίθεσης.

6.4 Μελλοντική Έρευνα

Αρκετές κατευθύνσεις για μελλοντική έρευνα προκύπτουν από την παρούσα εργασία. Αρχικά, παρόμοια πειράματα θα μπορούσαν να διεξαχθούν σε άλλα δημοφιλή CMS ώστε να πραγματοποιηθεί συγκριτική ανάλυση της συμπεριφοράς απαρίθμησης και της αποτελεσματικότητας των μέτρων σκλήρυνσης.

Επιπλέον, θα μπορούσε να αξιολογηθεί συστηματικά ο ρόλος δημοφιλών πρόσθετων ασφάλειας ως προς την παρεχόμενη προστασία έναντι της απαρίθμησης χρηστών. Μία ακόμη κατεύθυνση αφορά τη διερεύνηση τεχνικών αυτόματης ανίχνευσης προσπαθειών απαρίθμησης μέσω ανάλυσης συμπεριφοράς και συσχέτισης καταγραφών (log correlation).

Τέλος, μελλοντική έρευνα θα μπορούσε να εξετάσει αλλαγές σε επίπεδο σχεδιασμού του CMS, όπως ο διαχωρισμός της δημόσιας παρουσίασης περιεχομένου από την ταυτότητα του συγγραφέα, με στόχο τη μείωση του κινδύνου απαρίθμησης χωρίς υποβάθμιση της χρηστικότητας ή των απαιτήσεων SEO.

6.5 Τελικές Παρατηρήσεις

Η απαρίθμηση χρηστών συχνά αντιμετωπίζεται ως ζήτημα χαμηλής σοβαρότητας· ωστόσο, η παρούσα εργασία καταδεικνύει ότι διαδραματίζει κρίσιμο ρόλο στη διευκόλυνση σοβαρότερων επιθέσεων, όπως το credential stuffing και οι στοχευμένες επιθέσεις brute force. Καθώς το WordPress εξακολουθεί να κυριαρχεί στο οικοσύστημα των CMS, η κατανόηση και η αντιμετώπιση των κινδύνων απαρίθμησης παραμένουν ζωτικής σημασίας.⁶

Τα ευρήματα αυτής της μεταπτυχιακής διατριβής αναδεικνύουν τη σημασία της πολυεπίπεδης άμυνας και αποδεικνύουν ότι ουσιαστικές βελτιώσεις ασφάλειας μπορούν να επιτευχθούν μέσω προσεκτικής παραμετροποίησης και συνδυαστικών ελέγχων, ακόμη και σε πλατφόρμες ευρείας χρήσης και υψηλής λειτουργικότητας.

Βιβλιογραφία

- [1] WordPress Foundation, WordPress Documentation,
<https://wordpress.org/support/>
- [2] WordPress Foundation, WordPress REST API Handbook,
<https://developer.wordpress.org/rest-api/>
- [3] WordPress Foundation, Roles and Capabilities,
<https://wordpress.org/documentation/article/roles-and-capabilities/>
- [4] OWASP Foundation, OWASP Top 10 – 2021,
<https://owasp.org/www-project-top-ten/>
- [5] OWASP Foundation, OWASP Web Security Testing Guide,
<https://owasp.org/www-project-web-security-testing-guide/>
- [6] OWASP Foundation, Credential Stuffing,
https://owasp.org/www-community/attacks/Credential_stuffing
- [7] Nginx Inc., Nginx Rate Limiting,
https://nginx.org/en/docs/http/nginx_http_limit_req_module.html
- [8] Nginx Inc., Nginx HTTP Access Module,
https://nginx.org/en/docs/http/nginx_http_access_module.html
- [9] Fielding, R. et al., HTTP Semantics (RFC 9110), IETF, 2022,
<https://www.rfc-editor.org/rfc/rfc9110.html>
- [10] Fielding, R. et al., Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content (RFC 7231), IETF, 2014,
<https://www.rfc-editor.org/rfc/rfc7231.html>
- [11] W3Techs, Usage Statistics of Content Management Systems,
https://w3techs.com/technologies/overview/content_management
- [12] Behl, A. and Behl, K., Cyberwar: The Next Threat to National Security and What to Do About It, Oxford University Press, 2017.
- [13] Scarfone, K. and Mell, P., Guide to Intrusion Detection and Prevention Systems (IDPS), NIST SP 800-94, 2007.
- [14] NIST, Digital Identity Guidelines, NIST SP 800-63B, 2017.

- [15] Kumar, S. and Kalita, K., “User Enumeration Attacks on Web Applications,” International Journal of Computer Applications, vol. 123, no. 5, 2015.
- [16] Stoneburner, G., Goguen, A., and Feringa, A., Risk Management Guide for Information Technology Systems, NIST SP 800-30, 2012.
- [17] Akamai, State of the Internet / Security Report, 2023.
- [18] Imperva, Bad Bot Report, 2023.
- [19] Grimes, R., Hacking the Hacker: Learn from the Experts Who Take Down Hackers, Wiley, 2017.
- [20] Kahn Academy Labs, Web Application Security Fundamentals, 2020.

Παράρτημα

Στο παράρτημα παρατίθεται ο πλήρης κώδικας του εργαλείου που χρησιμοποιήθηκε για την αυτοματοποίηση των δοκιμών.

```
#!/usr/bin/env python3
import argparse
import requests
import re
from urllib.parse import urljoin, urlparse
from bs4 import BeautifulSoup

HEADERS = {"User-Agent": "WP-Enumeration-Lab/1.1 Authorized Testing"}

def norm(base):
    return base.rstrip("/") + "/"

def same_host(url, base):
    return urlparse(url).netloc == urlparse(base).netloc

def author_id_enum(base, max_id):
    print("\n[1] Author ID enumeration: ?author=ID")
    found = []
    for i in range(1, max_id + 1):
        url = f"{base}?author={i}"
        r = requests.get(url, headers=HEADERS, allow_redirects=False,
timeout=8)
        loc = r.headers.get("Location", "")
        print(f"ID {i}: {r.status_code} {loc}")
        m = re.search(r"/author/([^\s]+)/?", loc, re.I)
        if m:
            found.append((i, m.group(1), loc))
    return found

def rest_users(base):
    print("\n[2] REST API users endpoint")
    url = urljoin(base, "wp-json/wp/v2/users?per_page=100")
    r = requests.get(url, headers=HEADERS, timeout=8)
    print(f"Status: {r.status_code}")

    users = []
    try:
        data = r.json()
        if isinstance(data, list):
```

```
        for u in data:
            item = {
                "id": u.get("id"),
                "name": u.get("name"),
                "slug": u.get("slug"),
                "link": u.get("link")
            }
            users.append(item)
            print(item)
        else:
            print(data)
    except Exception:
        print(r.text[:300])

    return users

def rest_posts(base):
    print("\n[3] REST API posts endpoint")
    url = urljoin(base, "wp-json/wp/v2/posts?per_page=10")
    r = requests.get(url, headers=HEADERS, timeout=8)
    print(f"Status: {r.status_code}")

    authors = set()
    post_urls = []

    try:
        data = r.json()
        if isinstance(data, list):
            for post in data:
                author = post.get("author")
                link = post.get("link")

                if author:
                    authors.add(author)
                if link:
                    post_urls.append(link)

            print(f"post={post.get('id')} author={author} link={link}")

            uagb = post.get("uagb_author_info")
            if isinstance(uagb, dict):
                print(f" uagb_author_info={uagb}")
        else:
```

```
        print(data)
    except Exception:
        print(r.text[:300])

    return authors, post_urls

def discover_post_urls_from_homepage(base):
    print("\n[4] Discovering post URLs from homepage")

    urls = set()
    r = requests.get(base, headers=HEADERS, timeout=10,
allow_redirects=True)
    print(f"Homepage status: {r.status_code}")

    soup = BeautifulSoup(r.text, "html.parser")

    for a in soup.find_all("a", href=True):
        href = a["href"].strip()
        full = urljoin(base, href)

        if not same_host(full, base):
            continue

        low = full.lower()
        skip_parts = [
            "/wp-admin", "/wp-login", "/wp-content", "/wp-json",
            "/feed", "/comments", "/category/", "/tag/", "/author/"
        ]
        if any(x in low for x in skip_parts):
            continue

        path = urlparse(full).path.strip("/")
        if path and "." not in path:
            urls.add(full)

    for url in sorted(urls):
        print(f"candidate_post={url}")

    return sorted(urls)

def scan_author_links_from_url(url):
    found = set()
```

```
try:
    r = requests.get(url, headers=HEADERS, timeout=10,
allow_redirects=True)
except Exception as e:
    print(f"Error fetching {url}: {e}")
    return found

print(f"\nURL: {url} status={r.status_code}")

html = r.text

# Version 4-style broad regex: catches /author/<slug>/ anywhere in HTML
patterns = [
    r"/author/([^/\"'\\s<>]+)/?",
    r"\\author\\([^/\"'\\s<>+)\\/?",
]

for pattern in patterns:
    for slug in re.findall(pattern, html, re.I):
        slug = slug.strip("/")
        if slug:
            found.add(slug)
            print(f"regex_author_slug={slug}")

soup = BeautifulSoup(html, "html.parser")

for a in soup.find_all("a", href=True):
    href = a["href"]
    text = a.get_text(" ", strip=True)

    if "/author/" in href.lower():
        full = urljoin(url, href)
        print(f"author_link text='{text}' href='{full}'")

        m = re.search(r"/author/([^/]+)/?", full, re.I)
        if m:
            found.add(m.group(1))

    rel = a.get("rel")
    if rel:
        rel_text = " ".join(rel) if isinstance(rel, list) else str(rel)
        if "author" in rel_text.lower():
            print(f"rel_author text='{text}' href='{href}'")
```

```
        return found

def passive_html(base, manual_paths, discovered_urls):
    print("\n[5] Passive HTML author links")
    found = set()
    urls = set()

    for path in manual_paths:
        urls.add(base.rstrip("/") + "/" + path.lstrip("/"))

    for url in discovered_urls:
        urls.add(url)

    if not urls:
        urls.add(base)

    for url in sorted(urls):
        found.update(scan_author_links_from_url(url))

    return found

def xmlrpc_check(base):
    print("\n[6] XML-RPC exposure")
    url = urljoin(base, "xmlrpc.php")
    r = requests.get(url, headers=HEADERS, timeout=8)
    print(f"Status: {r.status_code}")
    print(f"Allow: {r.headers.get('Allow')}")
    print(r.text[:200])

def main():
    parser = argparse.ArgumentParser(
        description="Authorized WordPress user enumeration lab script"
    )
    parser.add_argument("base_url", help="Example: http://wpsite-v69.lan/")
    parser.add_argument("--max-id", type=int, default=10)
    parser.add_argument(
        "--post-path",
        action="append",
        default=[],
        help="Optional public post path, e.g. /author-post/. Can be repeated."
    )
```

```
args = parser.parse_args()
base = norm(args.base_url)

author_results = author_id_enum(base, args.max_id)
rest_user_results = rest_users(base)
post_author_ids, rest_post_urls = rest_posts(base)
homepage_post_urls = discover_post_urls_from_homepage(base)

all_post_urls = sorted(set(rest_post_urls + homepage_post_urls))
passive_slugs = passive_html(base, args.post_path, all_post_urls)

xmlrpc_check(base)

print("\n===== SUMMARY =====")

print("Author ID redirects:")
for item in author_results:
    print(f"  id={item[0]} slug={item[1]} location={item[2]}")

print("\nREST users:")
for u in rest_user_results:
    print(f"  id={u['id']} slug={u['slug']} name={u['name']}
link={u['link']}")

print("\nAuthor IDs from REST posts:")
for aid in sorted(post_author_ids):
    print(f"  author_id={aid}")

print("\nPassive HTML slugs:")
for slug in sorted(passive_slugs):
    print(f"  slug={slug}")

if __name__ == "__main__":
    main()
```