ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Διπλωματική Εργασία

των

Κορδώνη Μαρία

Παπαθανασίου Άγγελου

Ανάλυση ασφάλειας και δοκιμή διείσδυσης σε πλαίσια ανάπτυξης εφαρμογών ιστού

(Security analysis and Penetration Testing of Web Development Frameworks)

Επιβλέπων: Κρητικός Κυριάκος (Αναπληρωτής Καθηγητής)

Μέλη Εξεταστικής Επιτροπής: Κοκολάκης Σπύρος (Καθηγητής) , Καρύδα Μαρία (Καθηγήτρια)

Σάμος, Φεβρουάριος, 2024

[Κενή σελίδα]

Πρόλογος και Ευχαριστίες

Πρόλογος

Η παρούσα διπλωματική εργασία αντιπροσωπεύει το αποτέλεσμα αφοσιωμένης έρευνας και προσπάθειας στον τομέα της ασφάλειας εφαρμογών ιστού. Μέσα από αυτήν την εργασία, επιδιώξαμε να εξερευνήσουμε τις ευπαθείς πτυχές των εφαρμογών που αναπτύσσονται με τη χρήση πλαισίων ανάπτυξης εφαρμογών ιστού ανοικτού κώδικα, εστιάζοντας ειδικά στη γλώσσα προγραμματισμού Java.

Μέσα από τη θεωρητική ανάλυση των χαρακτηριστικών ασφάλειας και την πρακτική αξιολόγηση των επιλεγμένων πλαισίων, ελπίζουμε να προσφέρουμε νέες εισηγήσεις και πρακτικές συστάσεις που θα συμβάλουν στην ενίσχυση της ασφάλειας των εφαρμογών ιστού.

Ευχαριστίες

Θέλουμε να εκφράσουμε τις ειλικρινείς μας ευχαριστίες προς τον Επιβλέποντα καθηγητή, κ. Κυριάκο Κρητικό για την καθοδήγηση, τη συμβουλή και την υποστήριξή του καθ' όλη τη διάρκεια αυτής της πορείας. Επίσης, εκφράζουμε την ευγνωμοσύνη μας προς τους συναδέλφους που μοιράστηκαν τις γνώσεις τους και προσέφεραν εποικοδομητική ανταλλαγή ιδεών.

Στα πλαίσια ολοκλήρωσής της φοίτησής μας στο Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων του Πανεπιστημίου Αιγαίου, αλλά και με την εκπόνηση της διπλωματικής μας εργασίας, είναι προνόμιο να έχουμε συνεργαστεί με τόσους υπέροχους ανθρώπους και ελπίζουμε ότι αυτή η εργασία θα συμβάλει στον ευρύτερο κοινωνικό και επαγγελματικό χώρο.

Τέλος, ευχαριστούμε τους φίλους και την οικογένειά μας για τη στήριξη και την κατανόησή τους κατά τη διάρκεια αυτής της πορείας. Η αγάπη και η υποστήριξή τους ήταν ανεκτίμητες.

[Κενή σελίδα]

Περίληψη

Η παρούσα διπλωματική εργασία διερευνά την ασφάλεια των εφαρμογών ιστού που αναπτύσσονται με χρήση πλαισίων ανάπτυξης εφαρμογών ιστού ανοικτού κώδικα, εστιάζοντας στη γλώσσα προγραμματισμού Java. Η έρευνα ξεκινά με την επιλογή κατάλληλων πλαισίων και στη συνέχεια προχωρά σε μια θεωρητική ανάλυση των χαρακτηριστικών ασφαλείας που παρέχουν αυτά τα πλαίσια.

Στη συνέχεια, πραγματοποιείται ανάλυση των τρωτοτήτων με τη χρήση ανιχνευτών τρωτοτήτων, προκειμένου να εντοπιστούν πιθανές ευπάθειες των πλαισίων. Τα αποτελέσματα αυτής της ανάλυσης χρησιμοποιούνται για να συγκριθούν τα επιλεγμένα πλαίσια όσον αφορά τις τρωτότητες και το επίπεδο ρίσκου που ενδέχεται να παρουσιάζουν.

Τέλος, με βάση τα ευρήματα της ανάλυσης τρωτοτήτων, διενεργείται δοκιμή διείσδυσης σε εφαρμογές ιστού που έχουν υλοποιηθεί με χρήση των επιλεγμένων πλαισίων. Αυτό παρέχει πρακτική επιβεβαίωση των ευπαθειών και επιτρέπει τη βελτίωση της ασφάλειας της εφαρμογής.

Συνολικά, η εργασία αποτελεί μια ολοκληρωμένη προσέγγιση για την αξιολόγηση και βελτίωση της ασφάλειας σε εφαρμογές ιστού που χρησιμοποιούν πλαίσια ανοικτού κώδικα στο περιβάλλον της γλώσσας προγραμματισμού Java.

Λέξεις Κλειδιά: Ασφάλεια Εφαρμογών Ιστού, Πλαίσια Ανάπτυξης Εφαρμογών Ιστού, Java, Ανάλυση Χαρακτηριστικών Ασφάλειας, Ανίχνευση Τρωτοτήτων, Δοκιμή Διείσδυσης.

Abstract

This Thesis explores the security of web applications developed using open-source web application frameworks, focusing on the Java programming language. The research begins with the selection of suitable frameworks and then proceeds to a theoretical analysis of the security features provided by these frameworks.

Subsequently, a vulnerability analysis is conducted using vulnerability scanners to identify potential weaknesses in the frameworks. The results of this analysis are then used to compare the selected frameworks in terms of vulnerabilities and the corresponding level of risk they may pose.

Finally, based on the findings of the vulnerability analysis, penetration testing is performed on web applications implemented using the chosen frameworks. This provides practical confirmation of vulnerabilities and allows for the improvement of the application's security.

Overall, this thesis represents a comprehensive approach to assessing and enhancing security in web applications utilizing open-source frameworks in the Java programming language.

Key words: Web Application Security, Open Source Web Development Frameworks, Java, Security Features Analysis, Vulnerability Scanning, Penetration Testing

Πίνακας Περιεχομένων

Περιεχόμενα

1.Εισαγωγή 15
1.1 Γενικά 15
1.2 Δομή Εργασίας 17
2. Πλαίσια ανάπτυξης εφαρμογών ιστού για τη γλώσσα Java 19
2.1 Τι είναι ένα web application framework; 19
2.2 Ανάλυση των πλαισίων 20
3. Θεωρητική Ανάλυση Ασφάλειας Πλαισίων Ιστού 25
3.1 Κριτήρια Ανάλυσης 25
3.1.1 Αυθεντικοποίηση (Authentication) 25
3.1.2 Εξουσιοδότηση (Authorization) 27
3.1.3 Προστασία από Cross-Site Request Forgery (CSRF) επιθέσεις 28
3.1.4 Προστασία από Cross-site scripting (XSS) επιθέσεις 29
3.1.5 Επιθέσεις Έκχυσης SQL 30
3.1.6 Διαχείριση συνόδων (Session management) 31
3.1.7 Κρυπτογράφηση (Encryption) 32
3.2 Ανάλυση 34
3.2.1 Ανάλυση Πλαισίων 34
3.2.2. Καθολική Ανάλυση 41
4. Ανάλυση Τρωτοτήτων 43
4.1 Επιλεγμένα Εργαλεία Ανίχνευσης Τρωτοτήτων 44
4.1.1 Codacy 44
4.1.2 Snyk 49
4.1.3 Sonarqube 52
4.2 Ανάλυση εντοπισμένων Τρωτοτήτων 53
4.2.1 Πίνακες Τρωτοτήτων ανά πλαίσιο 53
4.2.2 Ανάλυση Τρωτοτήτων ανά πλαίσιο 57
4.3 Συμπεράσματα 117
5. Δοκιμή διείσδυσης (Penetration Testing) 120
5.1 Ορισμός και Βήματα Διείσδυσης 120
5.2 Εργαλεία Διείσδυσης 121

5.2.1 WMap 121
5.2.2 Nikto 123
5.2.3 Wappalyzer 125
5.3 Μεθοδολογία Διείσδυσης 126
5.4 Δοκιμές Διείσδυσης 127
5.4.1 Δοκιμή διείσδυσης στο Vaadin 127
5.4.2 Δοκιμή διείσδυσης στο Spring 133
5.4.3 Δοκιμή διείσδυσης στο Tapestry 140
5.4.4 Δοκιμή διείσδυσης για το Jmix 146
5.4.5 Δοκιμή διείσδυσης στο Struts 150
5.4.6 Δοκιμή διείσδυσης στο JHipster 156
5.5 Συμπεράσματα Δοκιμών 161
6. Συμπεράσματα και Μελλοντικά βήματα 162
6.1 Συμπεράσματα 162
6.2 Μελλοντικά Βήματα 164
7. Βιβλιογραφία 165

Λίστα Εικόνων

Εικόνα 1 Στιγμιότυπο από github λογαριασμό 46
Εικόνα 2 Στιγμιότυπο από codacy που εμφανίζονται τα αποθετήριά μας 47
Εικόνα 3 Στιγμιότυπο από codacy όπου εμφανίζονται τα αποθετήρια μας που σαρώθηκαν και το ποσοστό προβλημάτων που εντοπίστηκαν. 48
Εικόνα 4 Στιγμιότυπο από codacy που εμφανίζονται και τα commits του κώδικα αποθετηρίου 49
Εικόνα 5 Στιγμιότυπο από snyk που εμφανίζονται τα αποθετήρια μας 51
Εικόνα 6 Τρωτότητες του Vaadin από το codacy 57
Εικόνα 7 Στιγμιότυπο τρωτότητας 58
Εικόνα 8 Στιγμιότυπο τρωτότητας 60
Εικόνα 9 Τρωτότητες Spring Boot από το codacy 61
Εικόνα 10 Στιγμιότυπο Τρωτότητας 62
Εικόνα 11 Τρωτότητες Tapestry από το codacy 63
Εικόνα 12 Στιγμιότυπο τρωτότητας 64
Εικόνα 13 Τρωτότητες Struts από το codacy 65
Εικόνα 14 Στιγμιότυπο τρωτότητας 66
Εικόνα 15 Τρωτότητες JMix από το codacy 67
Εικόνα 16 Τρωτότητες JHipster από το codacy 68
Εικόνα 17 Τρωτότητες Tapestry από το Snyk 69
Εικόνα 18 Στιγμιότυπο τρωτότητας 71
Εικόνα 19 Στιγμιότυπο τρωτότητας 73
Εικόνα 20 Τρωτότητες Struts από το codacy 75
Εικόνα 21 Στιγμιότυπο τρωτότητας 76
Εικόνα 22 Στιγμιότυπο Τρωτότητας 77
Εικόνα 23 Στιγμιότυπο Τρωτότητας 78
Εικόνα 24 Στιγμιότυπο Τρωτότητας 79
Εικόνα 25 Τρωτότητες Vaadin από το Snyk 80
Εικόνα 26 Στιγμιότυπο Τρωτότητας 82
Εικόνα 27 Στιγμιότυπο Τρωτότητας 84
Εικόνα 28 Στιγμιότυπο Τρωτότητας 85
Εικόνα 29 Στιγμιότυπο Τρωτότητας 86
Εικόνα 30 Στιγμιότυπο Τρωτότητας 87
Εικόνα 31 Στιγμιότυπο Τρωτότητας 87
Εικόνα 32 Τρωτότητες του Struts από το Snyk 88
Εικόνα 33 Στιγμιότυπο Τρωτότητας 89
Εικόνα 34 Στιγμιότυπο Τρωτότητας 90
Εικόνα 35 Τρωτότητες Spring Boot από το Snyk 91
Εικόνα 36 Στιγμιότυπο Τρωτότητας 92
Εικόνα 37 Αποτελέσματα ανάλυσης του Jmix με Snyk 93
Εικόνα 38 Αποτελέσματα ανάλυσης του JHipster με Snyk 94
Εικόνα 39 Ευπάθειες του Vaadin από το SonarQube 95
Εικόνα 40 Ευπάθειες του Spring-Boot από το SonarQube 96
Εικόνα 41 Έλλειψη επικύρωσης hostname απομακρυσμένου εξυπηρετητή 97
Εικόνα 42 Παράκαμψη επικύρωσης πιστοποιητικού του εξυπηρετητή 99

Εικόνα 43 Διαρροή διαπιστευτηρίων για τον RabbitMQ μέσω του πηγαίου κώδικα 100
Εικόνα 44 Παράδειγμα σωστής εντολής (αντιπαραβολή με Εικόνα 43) 101
Εικόνα 45 Διαρροή διαπιστευτηρίων για τον εξυπηρετητή Redis 102
Εικόνα 46 Παράδειγμα χρήσης μεταβλητών συστήματος 102
Εικόνα 47 Απενεργοποίηση πρόσβασης σε εξωτερικές οντότητες της XML 103
Εικόνα 48 Κώδικας απενεργοποίησης επεξεργασίας εξωτερικής οντότητας 104
Εικόνα 49 Ευπάθειες του JHipster από το sonarqube 105
Εικόνα 50 Ευπάθειες του JMix από το Sonarqube 106
Εικόνα 51 Απενεργοποίηση πρόσβασης σε εξωτερικές οντότητες της XML. 107
Εικόνα 52 Χρήση μη ασφαλούς IV 108
Εικόνα 53 Σωστή δημιουργία IV 109
Εικόνα 54 Ανάλυση του Struts από το Sonarqube 110
Εικόνα 55 Διαχείριση εξαιρέσεων από την συνάρτηση include 111
Εικόνα 56 Απενεργοποίηση πρόσβασης σε εξωτερικές οντότητες XML 113
Εικόνα 57 Ανάλυση του Tapestry από το Sonarqube 114
Εικόνα 58 Απενεργοποίηση πρόσβασης σε εξωτερικές οντότητες της XML. 115
Εικόνα 59 Απενεργοποίηση πρόσβασης σε εξωτερικές οντότητες XML 115
Εικόνα 60 Ιστοσελίδα που έχει αναπτυχθεί με Vaadin 127
Εικόνα 61 Αποτελέσματα WMap για Vaadin 128
Εικόνα 62 Αποτελέσματα WMap για Vaadin 129
Εικόνα 63 Αποτελέσματα ευπαθειών με WMap 129
Εικόνα 64 Αναγνώριση καταλόγων Seminargo 130
Εικόνα 65 Αποτελέσματα ευπαθειών με Nikto 131
Εικόνα 66 Υλοποίηση εφαρμογής κατοικίδιων με το Spring 133
Εικόνα 67 Εντολές για την εκτέλεση της εφαρμογής Petclinic 133
Εικόνα 68 Αποτέλεσμα WMAP για Spring 134
Εικόνα 69 Εύρεση πιθανών ευπαθειών 135
Εικόνα 70 Χρήση ευπάθειας multi/http/lens_php_exec 137
Εικόνα 71 Αποτελέσματα Nikto για Spring 138
Εικόνα 72 Απόδειξη πρόκειται για τον Tomcat web server 139
Εικόνα 73 Ιστοσελίδα με χρήση Tapestry 140
Εικόνα 74 Ιστοσελίδα με χρήση Tapestry 141
Εικόνα 75 Αποτελέσματα χρήσης WMAP 142
Εικόνα 76 Αποτελέσματα χρήσης WMAP 142
Εικόνα 77 Στιγμιότυπο από Wappalyzer που εμφανίζει τις τεχνολογίες που χρησιμοποιούνται 143
Εικόνα 78 Αποτελέσματα Nikto για Tapestry 144
Εικόνα 79 Εφαρμογή με χρήση Jmix 146
Εικόνα 80 Περιεχόμενο που αφορά κλινική κατοικιδίων 146
Εικόνα 81 Αποτελέσματα WMAP 147
Εικόνα 82 Αποτελέσματα του Nikto 148
Εικόνα 83 Ιστοσελίδα TWF 150
Εικόνα 84 Αποτελέσματα WMAP 151
Εικόνα 85 Αποτελέσματα WMAP 152
Εικόνα 86 Αποτελέσματα WMAP 153

Εικόνα 87 Αποτελέσματα χρήσης exploits..... 154
Εικόνα 88 Αποτελέσματα χρήσης Nikto..... 155
Εικόνα 89 Η Ιστοσελίδα υλοποιημένη με JHipster που ελέγχεται για ευπάθειες 156
Εικόνα 90 Ιστοσελίδα υλοποιημένη με JHipster..... 157
Εικόνα 91 Αποτελέσματα χρήσης WMAP..... 158
Εικόνα 92 Αποτελέσματα χρήσης WMAP..... 159
Εικόνα 93 Αποτελέσματα χρήσης Nikto..... 160

Λίστα Πινάκων

Πίνακας 1 Ανάλυση πλαισίων ως προς την Αυθεντικοποίηση 34
Πίνακας 2 Ανάλυση Πλαισίων ως προς την Εξουσιοδότηση 35
Πίνακας 3 Ανάλυση ως προς το CSRF 36
Πίνακας 4 Ανάλυση ως προς το XSS 37
Πίνακας 5 Ανάλυση ως προς το SQL Injection 38
Πίνακας 6 Ανάλυση ως προς τη Διαχείριση Συνόδων 39
Πίνακας 7 Ανάλυση ως προς την Κρυπτογράφηση 40
Πίνακας 8 Καθολική Ανάλυση 41
Πίνακας 9 Τρωτότητες Vaadin 53
Πίνακας 10 Τρωτότητες Spring-Boot 54
Πίνακας 11 Τρωτότητες Tapestry 54
Πίνακας 12 Τρωτότητες Struts 55
Πίνακας 13 Τρωτότητες JHipster 56
Πίνακας 14 Τρωτότητες JMix 56
Πίνακας 15 Συμπεράσματα Codacy 117
Πίνακας 16 Συμπεράσματα Snyk 118
Πίνακας 17 Συμπεράσματα Sonarqube 118
Πίνακας 18 Καθολικά Συμπεράσματα 119
Πίνακας 19 Συμπεράσματα Δοκιμών Διείσδυσης 161
Πίνακας 20 Καθολικά Συμπεράσματα Δοκιμών Διείσδυσης 161

Ακρωνύμια

JDK	Java Development Kit
IDE	Integrated Development Environments
JDBC	Java Database Connectivity
JAAS	Java Authentication and Authorization Service
POM	Project Object Model
DSL	Domain Specific Language
SaaS	Sscurity as a Service
HTTP	Hypertext Transfer Protocol
HTML	HyperText Markup Language
MVC	Model-View-Controller
J2EE	Java 2 Platform, Enterprise Edition
POJO	Plain Old Java Object
JSF	JavaServer Faces
JPA	Java Persistence API
UI	user-interface
CSS	Cascading Style Sheets
GWT	Google Web Toolkit
API	Application Programming Interface
PHP	Hypertext Preprocessor
JSP	JavaServer Pages
OAuth	Open Authentication
CA	Certificate Authority
LDAP	Lightweight Directory Access Protocol
SSO	Single Sign-On
RBAC	Role-Based Access Control
ABAC	Attribute-Based Access Control
IP	Internet Protocol
CSRF	Cross-Site Request Forgery
URL	Uniform Resource Locator
DSC	Double Submit Cookie
XSS	Cross-site scripting
CSP	Content Security Policy
SOP	Same Origin Policy
SQL	Structured Query Language
Regex	Regular Expression
HTTPS	Hypertext Transfer Protocol Secure
ID	Identifier
AES	Advanced Encryption Standard
RSA	Rivest-Shamir-Adleman
CI/CD	Continuous Integration/Continuous Delivery
OWASP	Open Web Application Security Project

RCE	Remote Code Execution
XXE	External Entity
XML	Extensible Markup Language
SQLi	SQL Injection
IMAP	Internet Message Access Protocol
CWD	Current Working Directory
ENV	Environment Variables
CMD	Windows Command Prompt
REST	Representational State Transfer
JSON	JavaScript Object Notation
DoS	Denial of Service
EJB	Enterprise JavaBeans
RegExp	Regular Expression
ReDoS	Regular Expression Denial of Service
JVM	Java Virtual Machine
JAR	Java Archive
XSLT	XSL Transformations
URI	Uniform Resource Identifier
SSL	Secure Sockets Layer
TLS	Transport Layer Security
MitM	Man-in-the-Middle
CA	Certificate Authority
SSRF	Server-Side Request Forgery
IV	Initialization Vector
PHP	Hypertext Preprocessor
CGI	Common Gateway Interface
CSV	Comma-Separated Values
EOL	End-of-Life
SAML	Security Assertion Markup Language
HSTS	HTTP Strict-Transport-Security

1.Εισαγωγή

1.1 Γενικά

Η Java είναι μια δημοφιλής γλώσσα προγραμματισμού που χρησιμοποιείται ευρέως για την ανάπτυξη εφαρμογών ιστού (web applications), ιδιαίτερα ανοικτού κώδικα (open-source), λόγω των αρκετών πλεονεκτημάτων που προσφέρει. Ορισμένα από αυτά τα πλεονεκτήματα είναι:

Πολυπλατφορμικότητα: Η Java είναι μια πολυπλατφορμική γλώσσα προγραμματισμού. Αυτό σημαίνει πως οι εφαρμογές που αναπτύσσονται σε αυτήν μπορούν να τρέξουν σε διαφορετικά λειτουργικά συστήματα, όπως Windows, Linux, MacOS κ.λ.π.

Μεγάλη κοινότητα: Η Java διαθέτει μεγάλη κοινότητα προγραμματιστών και υποστηρικτών, οι οποίοι παρέχουν πλούσια τεκμηρίωση, βιβλιοθήκες και πλαίσια ανάπτυξης, καθώς και συχνές ενημερώσεις και βελτιώσεις της γλώσσας και του περιβάλλοντος ανάπτυξης. Ειδικότερα, εκτός από το Java Development Kit (JDK) και τις βασικές βιβλιοθήκες της Java, έχουν υλοποιηθεί εργαλεία τύπου Integrated Development Environments (IDEs) (Ενοποιημένα Περιβάλλοντα Ανάπτυξης) που υποστηρίζουν ανάπτυξη κώδικα Java (Eclipse, IntelliJ, NetBeans). Επιπλέον, η Java σχετίζεται με πλαίσια (frameworks), όπως το Spring Framework και το Hibernate, τα οποία παρέχουν μια σειρά από βιβλιοθήκες και εργαλεία για την ανάπτυξη διαφόρων ειδών εφαρμογών σε Java καθώς και την διασύνδεση με βάσεις δεδομένων όπως MySQL, Oracle, και PostgreSQL κυρίως μέσω της χρήσης του JDBC.

Ασφάλεια και Αξιοπιστία:
Η Java είναι μια σταθερή, αξιόπιστη και ασφαλής γλώσσα προγραμματισμού, με πολλές δυνατότητες για τον έλεγχο των σφαλμάτων και τη διαχείριση των εξαιρέσεων. Διαθέτει ένα αξιόπιστο σύστημα διαχείρισης μνήμης, το οποίο βοηθά στην πρόληψη ευπάθειας του συστήματος σε περιπτώσεις υπερχείλισης μνήμης (memory overflow) ή διαρροής μνήμης (memory leak).

Η Java διαθέτει αρκετά ενσωματωμένα μέτρα ασφάλειας (κρυπτογραφία, αυθεντικοποίηση και εξουσιοδότηση), τα οποία μπορούν να προστατεύσουν τις εφαρμογές από επιθέσεις κακόβουλου λογισμικού. Αυτό αποδεικνύεται στην επίσημη ιστοσελίδα της Oracle για την ασφάλεια της Java[1].

Επιπλέον, η Java χρησιμοποιεί μια αρκετά συνηθισμένη μέθοδο για την αυθεντικοποίηση και εξουσιοδότηση των χρηστών, γνωστή ως "Java Authentication and Authorization Service" (JAAS), η οποία χρησιμοποιεί μηχανισμούς όπως κωδικούς πρόσβασης, κλειδιά ασφαλείας και πιστοποιητικά. Στη συνέχεια, η JAAS εξουσιοδοτεί το χρήστη να προσπελάσει συγκεκριμένους πόρους, χρησιμοποιώντας διαφορετικούς μηχανισμούς εξουσιοδότησης, όπως αντικείμενα "αδειών" και ρόλους χρηστών. Οι αδειοδοτήσεις αντιπροσωπεύουν τη δικαιοδοσία για συγκεκριμένες ενέργειες σε συγκεκριμένους πόρους, ενώ οι ρόλοι χρηστών παρέχουν δικαιώματα πρόσβασης σε ομάδες χρηστών.

[1] https://www.oracle.com/java/technologies/java-security.html

Ένα σημαντικό πλεονέκτημα της JAAS είναι ότι επιτρέπει στους προγραμματιστές να ορίζουν πολύπλοκες πολιτικές ασφαλείας για τις εφαρμογές τους. Οι πολιτικές ασφαλείας μπορούν να οριστούν σε ένα αρχείο παραμέτρων και να προσαρμοστούν ανάλογα με τις ανάγκες της εφαρμογής.

Υπάρχουν επίσης πολλά εργαλεία ανάλυσης ασφαλείας που είναι διαθέσιμα για τους προγραμματιστές της Java, όπως το FindBugs και το PMD, τα οποία πραγματοποιούν στατική ανάλυση εντοπίζοντας προγραμματιστικά σφάλματα και θέματα ασφάλειας, τα οποία έπειτα μπορούν να επιλυθούν από τους προγραμματιστές ώστε να παράγουν αξιόπιστο και ασφαλή κώδικα.

Τέλος, η Java διαθέτει μια εκτεταμένη κοινότητα που επικεντρώνεται στην ασφάλεια, με τη δυνατότητα ανάπτυξης βιβλιοθηκών και εργαλείων ασφαλείας που μπορούν να χρησιμοποιηθούν σε συνδυασμό με τα πλαίσια ανάπτυξης εφαρμογών ιστού (web frameworks) που βασίζονται σε αυτήν. Αυτό μπορεί να αυξήσει το επίπεδο ασφάλειας των παραγόμενων εφαρμογών ιστού.
Ως εκ τούτου, με βάση τα παραπάνω, αν και καμία γλώσσα προγραμματισμού δεν είναι 100% ασφαλής, η Java θεωρείται μια καλή επιλογή ως προς την ασφάλεια για τη δημιουργία πλαισίων ανάπτυξης (εφαρμογών) ιστού (web frameworks).

Τα δύο βασικά εργαλεία για τη διαχείριση κατασκευής ενός έργου λογισμικού σε Java είναι το Maven και το Gradle. Και τα δύο παρέχουν λύσεις για την αυτοματοποίηση της διαδικασίας κατασκευής, εκτέλεσης δοκιμών, και διαχείρισης εξαρτήσεων.

Maven:

Το Maven είναι ένα εργαλείο διαχείρισης έργων που βασίζεται σε ένα πρότυπο οργάνωσης έργου, το "Project Object Model" (POM). Κάθε έργο Maven έχει ένα αρχείο POM που περιέχει πληροφορίες σχετικά με το έργο, τις εξαρτήσεις του, τις διαδικασίες κατασκευής, και άλλες ρυθμίσεις.

Πλεονεκτήματα του Maven:

1. Αυτοματοποίηση Κατασκευής**:** Το Maven διευκολύνει τη διαδικασία κατασκευής και τη διαχείριση των εξαρτήσεων.
2. Κοινότητα: Έχει μεγάλη κοινότητα που συνεισφέρει σε πρόσθετα plugins και υποστηρίζει πολλά έργα.
3. Πρότυπο Οργάνωσης: Η χρήση του POM προσφέρει συνέπεια και ευκολία στη διαχείριση έργων.

Gradle:

Το Gradle είναι ένα ευέλικτο εργαλείο διαχείρισης κατασκευής που χρησιμοποιεί τη γλώσσα DSL (Domain Specific Language) για τον έλεγχο των διαδικασιών κατασκευής. Είναι βασισμένο σε ένα γράφο εξαρτήσεων, προσφέροντας ευελιξία και αποδοτικότητα.

Πλεονεκτήματα του Gradle:

1. Ευελιξία: Οι χρήστες μπορούν να ορίσουν τις διαδικασίες κατασκευής με βάση τις ανάγκες τους χρησιμοποιώντας DSL.
2. Αποδοτικότητα: Το Gradle είναι σχεδιασμένο για να είναι γρήγορο και αποδοτικό.
3. Συμβατότητα με Maven: Μπορεί να χρησιμοποιηθεί με έργα που ήδη χρησιμοποιούν το Maven.

Και τα δύο αυτά εργαλεία είναι δημοφιλή στην κοινότητα των προγραμματιστών Java, και η επιλογή μεταξύ τους εξαρτάται συχνά από τις ανάγκες και τις προτιμήσεις της ομάδας ανάπτυξης.

Συμπεράσματα: Με βάση την παραπάνω ανάλυση, η γλώσσα Java είναι αρκετά αξιόπιστη, ασφαλής και πλούσια σε χαρακτηριστικά για αυτό και είναι αρκετά δημοφιλής στους προγραμματιστές. Επιπλέον, με την έλευση του υπολογιστικού νέφους, όλες οι εφαρμογές περνάνε από το παραδοσιακό μοντέλο διάθεσης στην μορφή SaaS όπου και διατίθενται ως εφαρμογές ιστού. Αυτό παρέχει πολλαπλά οφέλη για τον καταναλωτή όπως μείωση κόστους, μειωμένο/μηδαμινό διαχειριστικό κόστος και δυναμική κλιμάκωση των εφαρμογών.

Υπάρχει πληθώρα από πλαίσια για την ανάπτυξη εφαρμογών ιστού και είναι δύσκολο κάποιος να επιλέξει κάποια από αυτά, ειδικά αν έχει γνώμονα την ασφάλεια, εκτός από τα δυνητικά πλούσια χαρακτηριστικά ανάπτυξης που προσφέρουν.

Με βάση το παραπάνω ζητούμενο, η διπλωματική αυτή εργασία παρέχει σημαντική γνώση ως προς την ασφάλεια των πιο δημοφιλών πλαισίων ανάπτυξης που είναι βασισμένα σε Java για εφαρμογές ιστού, βοηθώντας τους προγραμματιστές να επιλέξουν εκείνο το πλαίσιο που ταιριάζει με τις ανάγκες ασφαλείας τους.

1.2 Δομή Εργασίας

Συγκεκριμένα, στην εργασία αναπτύσσονται τα ακόλουθα κεφάλαια:
Στην εισαγωγή, παρουσιάζεται μια ανάλυση της Java ως γλώσσας προγραμματισμού για την ανάπτυξη εφαρμογών ιστού. Επισημαίνονται πλεονεκτήματα και η συνολική αξιοπιστία της Java, επιβεβαιώνοντας την επιλογή μας για ανάλυση των πλαισίων που έχουν υλοποιηθεί πάνω σε αυτή.
Στο Κεφάλαιο 2 πραγματοποιείται μια γενική ανάλυση σχετικά με τα πλαίσια εφαρμογών ιστού, καθώς και μία ειδική ανάλυση των πλαισίων που επιλέξαμε για αξιολόγηση στα πλαίσια αυτής της εργασίας καθώς και των λόγων που τα επιλέξαμε.
Στο κεφάλαιο 3 πραγματοποιείται μια ανάλυση σε θεωρητικό επίπεδο για τα χαρακτηριστικά ασφάλειας που προσφέρονται από κάθε ένα από αυτά τα δημοφιλή πλαίσια. Παρουσιάζονται κάποια κριτήρια με βάση τα οποία επιλέξαμε να γίνει η αξιολόγηση και στο τέλος γίνεται μια καθολική σύγκριση των πλαισίων.
Στο κεφάλαιο 4, η εργασία αυτή εστιάζει στο να ανιχνεύσει τρωτότητες σε αυτά τα πλαίσια με τη χρήση ειδικών εργαλείων ανίχνευσης τρωτοτήτων (vulnerability scanners) ανοικτού κώδικα. Με βάση τα αποτελέσματα της ανίχνευσης, πραγματοποιεί σύγκριση των πλαισίων ως προς τις τρωτότητες που παρουσιάζουν και το αντίστοιχο ρίσκο ασφάλειας τους.

Στο κεφάλαιο 5 υλοποιείται δοκιμή διείσδυσης σε εφαρμογές ιστού που έχουν υλοποιηθεί μέσω αυτών των πλαισίων με χρήση ειδικών εργαλείων και παρουσιάζονται τα αποτελέσματα για κάθε πλαίσιο.
Τέλος στο κεφάλαιο 6 αναφέρονται τα συμπεράσματα της έρευνας.

2. Πλαίσια ανάπτυξης εφαρμογών ιστού για τη γλώσσα Java

2.1 Τι είναι ένα web application framework;

Ένα πλαίσιο ανάπτυξης εφαρμογών ιστού (web application framework) είναι μια συλλογή βιβλιοθηκών, εργαλείων και προτύπων που χρησιμοποιούνται για τη δημιουργία δυναμικών ιστοσελίδων και εφαρμογών ιστού.
Τα πλαίσια αυτά παρέχουν ένα σύνολο από βασικές λειτουργίες και δυνατότητες, όπως διαχείριση δρομολόγησης (routing), αυθεντικοποίηση χρηστών, σύνδεση σε βάσεις δεδομένων, διαχείριση αιτημάτων HTTP, και παραγωγή HTML ή άλλων τύπων περιεχομένου. Η χρήση ενός πλαισίου ανάπτυξης επιτρέπει στους προγραμματιστές να επικεντρωθούν στην υλοποίηση της λειτουργικότητας των εφαρμογών τους, αντί να χάνουν χρόνο στην υλοποίηση των προαναφερόμενων βασικών λειτουργιών που παρέχονται ήδη από το πλαίσιο. Τα πλαίσια ανάπτυξης, επίσης, προάγουν την επαναχρησιμοποίηση κώδικα (code reuse) καθώς και σχετικών λύσεων σε διαφορετικά έργα ανάπτυξης. Αυτό συμβαίνει διότι είναι σχεδιασμένα έτσι ώστε να παρέχουν έτοιμες λύσεις για συχνά επαναλαμβανόμενα προβλήματα στην ανάπτυξη λογισμικού, με στόχο την απλοποίηση της διαδικασίας ανάπτυξης και τη βελτίωση της ποιότητας του κώδικα. Επομένως, παρέχουν βιβλιοθήκες και πακέτα κώδικα που είναι σχεδιασμένα για να επαναχρησιμοποιηθούν σε άλλα έργα ανάπτυξης. Αυτό μειώνει τον χρόνο που απαιτείται για την ανάπτυξη λογισμικού καθώς και την πιθανότητα σφαλμάτων στον κώδικα.

Η αρχιτεκτονική που ακολουθεί ένα πλαίσιο αφορά τη σχέση του με βάσεις δεδομένων, εξυπηρετητές (servers) και εφαρμογές. Καθορίζει το πως η λειτουργικότητα και η λογική ενός συστήματος λογισμικού μοιράζεται μεταξύ του εξυπηρετητή και του πελάτη, ενώ διασφαλίζει ότι οι δύο πλευρές συνεννοούνται σωστά. Επιπλέον, εξασφαλίζει τον έλεγχο των δεδομένων και στις 2 πλευρές ώστε πάντοτε έγκυρα δεδομένα να παραδίδονται τελικά στον εξυπηρετητή. Τα περισσότερα πλαίσια εφαρμογών ακολουθούν την αρχιτεκτονική Model-View-Controller (MVC), η οποία αποτελείται από τα αντίστοιχα 3 μέρη: (α) το μοντέλο (Model) που αντιστοιχεί στην λογική αναπαράστασης των δεδομένων, (β) την Όψη (View) που αφορά την οπτικοποίηση των δεδομένων στον χρήστη/πελάτη και (γ) τον ελεγκτή (Controller) που λαμβάνει τα αιτήματα από τον χρήστη και εκτελεί τις κατάλληλες ενέργειες για να αλληλεπιδράσει με το μοντέλο, ώστε να ενημερώσει τα δεδομένα. Τα ανανεωμένα δεδομένα έπειτα εμφανίζονται στον χρήστη μέσω της Όψης.
Κάθε πλαίσιο ανάπτυξης ιστού έχει τα πλεονεκτήματά του και τα μειονεκτήματά του. Προχωρήσαμε στην επιλογή των πλαισίων, τα οποία αναλύονται παρακάτω, λαμβάνοντας υπόψη κριτήρια, όπως ο σκοπός που θα χρησιμοποιηθούν (για την ανάπτυξη εφαρμογών ιστού), τα χαρακτηριστικά που διαθέτουν, η διαθεσιμότητα (θα πρέπει να είναι ανοικτού κώδικα) & ευκολία χρήσης του κώδικά τους, η απόδοση τους, οι κριτικές σχετικά με την ασφάλειά τους (την οποία καλούμαστε και εμείς να αξιολογήσουμε) καθώς και η συμβατότητά τους με εργαλεία ανάλυσης ασφάλειας και τρωτοτήτων.

2.2 Ανάλυση των πλαισίων

Πραγματοποιώντας μια έρευνα για ανοικτού κώδικα πλαίσια ανάπτυξης εφαρμογών ιστού, τα οποία έχουν υλοποιηθεί με τη χρήση της γλώσσας προγραμματισμού Java, καταλήξαμε στα παρακάτω:

- **Spring:** Αρχικά γράφτηκε από τον Rod Johnson και κυκλοφόρησε για πρώτη φορά με την άδεια Apache 2.0 τον Ιούνιο του 2003.

  Είναι ελαφρύ όσον αφορά το μέγεθος (η βασική έκδοση του πλαισίου Spring είναι περίπου 2MB) και έχει σχεδιαστεί για να παρέχει μια λεπτή στρώση αφαίρεσης (abstraction layer) μεταξύ των διαφόρων τεχνολογιών και του κώδικα της εφαρμογής. Αυτό βοηθά στην ανάπτυξη ευέλικτων και επεκτάσιμων εφαρμογών, καθώς παρέχει στους προγραμματιστές τη δυνατότητα δημιουργίας πιο σύντομου κώδικα με την επαναχρησιμοποίηση λειτουργικότητας από βιβλιοθήκες. Το Spring Framework χρησιμοποιεί τους στολισμούς (annotations) για να δηλώνει τη δομή και τη συμπεριφορά των κλάσεων, των μεθόδων και των άλλων στοιχείων του κώδικα και κυρίως για την εκτέλεση διαφόρων λειτουργιών, όπως η διαχείριση των εξαρτήσεων, η ανίχνευση των συστατικών του πλαισίου, η εξυπηρέτηση αιτημάτων HTTP και άλλα. Οι στολισμοί είναι δηλώσεις που προστίθενται πάνω σε διάφορα στοιχεία του κώδικα, προκειμένου να προσδιορίσουν πληροφορίες σχετικά με το πώς πρέπει να εκτελεστεί ο κώδικας, πώς να αντιμετωπιστούν τα δεδομένα και ποια είναι η σημασία τους. Μπορούν να χρησιμοποιηθούν από τον μεταγλωττιστή (compiler) για την παραγωγή πρόσθετου κώδικα ή τον έλεγχο της ορθότητας του κώδικα, καθώς και από τα εργαλεία ανάλυσης κώδικα για τη δημιουργία τεκμηρίωσης, αυτόματων ελέγχων και άλλων λειτουργιών.

  Τα βασικά χαρακτηριστικά του πλαισίου Spring μπορούν να χρησιμοποιηθούν για την ανάπτυξη οποιασδήποτε εφαρμογής Java, αλλά υπάρχουν επεκτάσεις για τη δημιουργία εφαρμογών ιστού πάνω από την πλατφόρμα Java EE. Το Spring στοχεύει να κάνει την ανάπτυξη J2EE εφαρμογών πιο εύκολη στη χρήση και προωθεί καλές πρακτικές προγραμματισμού, ενεργοποιώντας ένα μοντέλο προγραμματισμού που βασίζεται στο POJO (είναι ένα αντικείμενο της Java που δεν έχει καμία εξάρτηση από άλλες βιβλιοθήκες ή πλαίσια, και ακολουθεί απλές αρχές σχεδίασης αντικειμένων).

  Μερικές από τις βασικές λειτουργίες του πλαισίου Spring περιλαμβάνουν τη διαχείριση των κλάσεων (class) - διαχειρίζονται από ένα σύστημα που ονομάζεται "Spring Container" και είναι υπεύθυνο για την φόρτωση και τη διαχείριση των κλάσεων που χρησιμοποιούνται από την εφαρμογή, την έγχυση εξαρτήσεων (dependency injection), την υποστήριξη για την ανάπτυξη εφαρμογών ιστού και τη πρόσβαση στα δεδομένα (data access). Επιπλέον, το Spring είναι συμβατό με πολλές (γνωστές) τεχνολογίες και πλαίσια, όπως το Hibernate για τη διαχείριση της διασύνδεσης με βάσεις δεδομένων και το JavaServer Faces (JSF) για τη δημιουργία ιστοσελίδων.

  Με το πλαίσιο Spring, οι προγραμματιστές μπορούν να αναπτύξουν εφαρμογές Java με μεγαλύτερη ευκολία, ταχύτητα και αξιοπιστία για διάφορους λόγους, όπως την υποστήριξη της ανάπτυξης βάση των αρχών του αντικειμενοστραφούς

προγραμματισμού και την εύκολη διαχείριση συστατικών (components). Το Spring παρέχει, επίσης, μια ευρεία γκάμα επεκτάσεων (add-ons) και πρόσθετων λειτουργιών (plugins), όπως το Spring Security (επέκταση που παρέχει αυξημένη ασφάλεια μέσω της επιβολής δικαιωμάτων πρόσβασης στις εφαρμογές) και το Spring Data JPA (πρόσθετο που επιτρέπει στους προγραμματιστές να χρησιμοποιούν το Java Persistence API (JPA) για πρόσβαση στη βάση δεδομένων), τα οποία βοηθούν τους προγραμματιστές να προσαρμόσουν και να επεκτείνουν τις λειτουργίες του πλαισίου για τις ανάγκες τους.

Η κοινότητα του Spring είναι ένας από τους παράγοντες που συνέβαλαν στην τεράστια επιτυχία και διάδοση του πλαισίου στην κοινότητα των προγραμματιστών. Το Spring Framework προσφέρει ευέλικτη και ισχυρή υποδομή για την ανάπτυξη εφαρμογών Java και έχει κερδίσει μεγάλη αποδοχή σε πολλούς κλάδους της ανάπτυξης λογισμικού. Η κοινότητα του Spring περιλαμβάνει προγραμματιστές από διάφορες περιοχές, όπως enterprise εφαρμογές, διαδικτυακές εφαρμογές, ανάπτυξη για κινητές συσκευές και cloud computing.

➔ **JHipster:** Το JHipster είναι ένα εργαλείο ανοιχτού κώδικα που χρησιμοποιείται για την ανάπτυξη πλήρως λειτουργικών εφαρμογών ιστού σε Java και JavaScript. Χρησιμοποιείται ευρέως από εταιρείες και προγραμματιστές για τη δημιουργία πολύπλοκων εφαρμογών ιστού. Συνδυάζει τις τεχνολογίες Java, Spring Boot στο νωτιαίο επίπεδο (backend) και Angular.js, React.js στο επίπεδο της διεπαφής χρήστη (user interface - UI), για να παρέχει μια πλατφόρμα ανάπτυξης εφαρμογών που μπορεί να χρησιμοποιηθεί για τη δημιουργία εφαρμογών ιστού με πλούσια λειτουργικότητα.

Το JHipster διευκολύνει τη διαδικασία ανάπτυξης λογισμικού, παρέχοντας ένα σύνολο εργαλείων και βιβλιοθηκών για την αυτοματοποίηση πολλών κοινών διαδικασιών. Αυτό συμπεριλαμβάνει τη δημιουργία μοντέλων δεδομένων, τη διαχείριση της ασφάλειας, τη δημιουργία RESTful APIs (χρήση του πρωτοκόλλου HTTP για ανταλλαγή δεδομένων) που ενσωματώνουν την βασική λειτουργικότητας μιας εφαρμογής (ιστού) και των διασυνδέσεων με τη βάση δεδομένων, καθώς και την ανάπτυξη της διεπαφής χρήστη.

Τέλος, Η κοινότητα του JHipster αποτελείται από προγραμματιστές και χρήστες που ενδιαφέρονται για την ανάπτυξη εφαρμογών που βασίζονται στο πλαίσιο. Πολλοί προγραμματιστές παρέχουν συνεισφορές στον κώδικα του πλαισίου, διορθώνουν σφάλματα (bugs) και βελτιώνουν τις λειτουργίες του. Επίσης, η κοινότητα μοιράζεται γνώσεις, βοηθά νέους προγραμματιστές, και συζητά για βέλτιστες πρακτικές και τις εμπειρίες τους με το πλαίσιο. Έχοντας μια μεγάλη κοινότητα είναι πιο πιθανό να βρείτε τη βοήθεια που χρειάζεστε όταν αντιμετωπίζετε προβλήματα ή αντιμετωπίζετε δυσκολίες με το JHipster. Επίσης, υπάρχουν περισσότεροι πόροι, όπως εκπαιδευτικό υλικό και παραδείγματα κώδικα, που μπορείτε να χρησιμοποιήσετε για να μάθετε και να αναπτύξετε το JHipster πλαίσιο.

➔ **Vaadin:** Το Vaadin είναι ένα ανοιχτού κώδικα πλαίσιο λογισμικού για τη δημιουργία επιχειρησιακών και διαδικτυακών εφαρμογών ιστού. Ο βασικός στόχους του Vaadin είναι να διευκολύνει τη δημιουργία εφαρμογών ιστού χωρίς την ανάγκη σε γνώσεις HTML, CSS και JavaScript.

Η αρχιτεκτονική του Vaadin βασίζεται στην Java και το GWT (Google Web Toolkit) ενώ παρέχει ένα σύνολο από UI συστατικά (components) (δηλ. τα βασικά στοιχεία που χρησιμοποιούνται για το σχεδιασμό και την κατασκευή μιας διεπαφής χρήστη σε μια εφαρμογή ή ιστοσελίδα). Η χρήση του Vaadin μπορεί να επιταχύνει τη διαδικασία ανάπτυξης και να βελτιώσει την αναπαραστατική ικανότητα μιας εφαρμογής, διότι η όλη λειτουργικότητα της εφαρμογής περιλαμβάνεται στον Java κώδικα που αναπτύσσεται και άρα δεν απαιτείται η ανάγκη αλλαγής σε γραφικά στοιχεία και στην δυναμικότητα της αλληλεπίδρασης με αυτά.

Επιπλέον, το Vaadin είναι εύκολο στη συντήρηση και στην επέκταση καθώς παρέχει πολλές επιλογές διαμόρφωσης και επικοινωνίας με νωτιαία (backend) συστήματα. Επίσης, παρέχει ευέλικτες δυνατότητες επαναχρησιμοποίησης κώδικα μέσω της δυνατότητας δημιουργίας προσαρμοσμένων UI συστατικών (components), όπως κουμπιά, πίνακες, και φόρμες, τα οποία μπορούν να επαναχρησιμοποιηθούν σε διάφορα μέρη της εφαρμογής.

Τέλος, η κοινότητα του Vaadin εχεί μέλη από όλον τον κόσμο και περιλαμβάνει προγραμματιστές που έχουν ενδιαφέρον στο να αναπτύξουν προγράμματα χρησιμοποιώντας το Vaadin, καθώς και χρήστες που χρησιμοποιούν το πλαίσιο για τις ανάγκες των εφαρμογών τους. Μια μεγάλη κοινότητα έχει πολλά πλεονεκτήματα, όπως περισσότερες παρεχόμενες προσθήκες, πιο γρήγορη επίλυση προβλημάτων, καλύτερη τεκμηρίωση και πιο διάφορες πηγές μάθησης. Τα μεγάλα κοινοτικά εγχειρίδια και οδηγοί μπορούν να βοηθήσουν τόσο τους νέους όσο και τους έμπειρους προγραμματιστές να επωφεληθούν πλήρως από το πλαίσιο.

➔ **Tapestry:** Το Tapestry είναι ένα ανοιχτού κώδικα πλαίσιο για την ανάπτυξη επαγγελματικών και δυναμικών εφαρμογών ιστού. Η προσέγγιση του Tapestry βασίζεται στην αρχή του συστήματος βασιζόμενο σε συστατικά (component-based system), όπου κάθε στοιχείο της εφαρμογής αντιστοιχεί σε ένα συστατικό (component), το οποίο μπορεί να είναι επαναχρησιμοποιήσιμο άλλα και να συνδυαστεί με άλλα στοιχεία ώστε να δημιουργηθεί ένα πιο σύνθετο συστατικό. Το Tapestry είναι γραμμένο σε Java και βασίζεται στις αρχές του αντικειμενοστραφούς προγραμματισμού και της αρχιτεκτονικής Model-View-Controller (MVC).

Ειδικότερα, τα επαναχρησιμοποιήσιμα συστατικά που προσφέρει το Tapestry για τη δημιουργία εφαρμογών περιλαμβάνουν φόρμες, πίνακες και διαγράμματα, ενώ παράλληλα παρέχεται η δυνατότητα δημιουργίας προσαρμοσμένων συστατικών ανάλογα με τις ανάγκες της εφαρμογής. Επιπλέον, το Tapestry παρέχει μια εξαιρετικά ισχυρή δυνατότητα διαχείρισης και αναζήτησης σφαλμάτων (debugging), καθιστώντας

τη διαδικασία ανάπτυξης και συντήρησης των εφαρμογών ιστού γρήγορη και αποτελεσματική.

Το Tapestry παρόλο που είναι εντελώς ανεξάρτητο από το νωτιαίο επίπεδο (backend), το υποστηρίζει διότι προσφέρει αλληλεπίδραση με τη βάση δεδομένων και την επεξεργασία των δεδομένων μέσω της προγραμματιστικής διεπαφής Hibernate API και του αντικειμένου συνόδου (session) που διατηρείται ανάμεσα στις αιτήσεις HTTP. Επίσης, μπορεί να χρησιμοποιηθεί με οποιαδήποτε νωτιαία τεχνολογία, όπως Java, PHP, και .NET. Αυτό καθιστά το Tapestry μια εξαιρετικά ευέλικτη λύση για την ανάπτυξη εφαρμογών ιστού.

Η τελευταία έκδοση του Tapestry είναι η 5.7.2, η οποία κυκλοφόρησε τον Φεβρουάριο του 2021. Η κοινότητα του Tapestry είναι αρκετά μικρή σε σχέση με άλλα πλαίσια ανάπτυξης, αλλά είναι ακόμα ενεργή και διαθέσιμη για υποστήριξη και βοήθεια στους χρήστες του Tapestry.

➔ **Struts:** Το Struts είναι ένα πλαίσιο ανοικτού κώδικα (open-source framework) για την ανάπτυξη επιχειρησιακών εφαρμογών σε Java. Το Struts βασίζεται στο μοντέλο σχεδίασης MVC και παρέχει μια δομή, δηλαδή έναν υπολογιστικό κορμό (core) για τη διαχείριση της ροής της εφαρμογής, την ανάπτυξη και τη συντήρηση επαναχρησιμοποιήσιμου κώδικα, τη διεπαφή με τον χρήστη, τη σύνδεση με βάσεις δεδομένων και άλλες κοινές λειτουργίες.

Μοντέλο (Model): Ο Model αναπαριστά την επιχειρηματική λογική και τα δεδομένα της εφαρμογής. Αυτό μπορεί να περιλαμβάνει αλγορίθμους, διαδικασίες ή ακόμα και αποθηκευμένα δεδομένα σε μια βάση δεδομένων. Η υπερβολική λογική και οποιαδήποτε αλληλεπίδραση με τη βάση δεδομένων συγκεντρώνονται στο μοντέλο. Η κεντρική ιδέα είναι η χωριστή και ξεκάθαρη διαχείριση των δεδομένων.

Προβολή (View): Η προβολή είναι υπεύθυνη για την παρουσίαση των δεδομένων στους χρήστες. Αντιπροσωπεύει το UI (Διεπαφή χρήστη) της εφαρμογής και μπορεί να είναι σε μορφή HTML, JSP (JavaServer Pages), ή οποιαδήποτε άλλη μορφή παρουσίασης. Η προβολή δεν περιέχει επιχειρηματική λογική, αλλά μόνο κώδικα για την εμφάνιση των δεδομένων που παρέχονται από το Model.

Ελεγκτής (Controller): Ο ελεγκτής είναι υπεύθυνος για τον συντονισμό της επικοινωνίας μεταξύ του χρήστη και της εφαρμογής. Αναλαμβάνει τον ρόλο του μεσολαβητή μεταξύ της προβολής και του μοντέλου. Όταν ο χρήστης δραστηριοποιείται στο UI, ο ελεγκτής λαμβάνει τα αιτήματα (requests), επεξεργάζεται τις ενέργειες που πρέπει να γίνουν και τις δρομολογεί στο κατάλληλο μοντέλο για επεξεργασία. Αφού το μοντέλο ολοκληρώσει τη λογική επεξεργασίας, ο ελεγκτής επιλέγει τη σωστή προβολή για να εμφανίσει τα αποτελέσματα στον χρήστη.

Μέσω της αρχιτεκτονικής MVC, το Struts επιτρέπει τη διαχωριστική συντήρηση των διάφορων συνιστωσών της εφαρμογής, παρέχοντας έτσι μια οργανωμένη προσέγγιση για την ανάπτυξη λογισμικού. Αυτό επιτρέπει την ευκολότερη συντήρηση, την αναπτυξιακή ευελιξία και την επαναχρησιμοποίηση του κώδικα. Επίσης, το Struts παρέχει διάφορα εργαλεία για τη διαχείριση και επικύρωση των δεδομένων, υποστηρίζει διεθνοποίηση και παρέχει εύκολο τρόπο επεκτασιμότητας της εφαρμογής.

Τα τελευταία χρόνια, με την εμφάνιση νέων τεχνολογιών και πλαισίων ανάπτυξης, η δημοτικότητα του Struts μειώθηκε σημαντικά. Ορισμένες άλλες τεχνολογίες, όπως το Spring Framework και το JavaServer Faces (JSF), έγιναν πιο δημοφιλείς στην κοινότητα των προγραμματιστών Java

Το Struts χρησιμοποιείται σε πολλές μεγάλες εταιρείες και οργανισμούς, όπως η IBM, η Cisco, η Sun Microsystems και πολλές άλλες.

- **Jmix:** Το Jmix είναι ένα ανοιχτού κώδικα πλαίσιο για την ανάπτυξη επιχειρησιακών εφαρμογών Java. Πρόκειται για ένα σύγχρονο, ευέλικτο και επεκτάσιμο πλαίσιο που παρέχει μια πλήρη σουίτα εργαλείων για την ανάπτυξη εφαρμογών υψηλής ποιότητας.

  Το Jmix βασίζεται στο Spring Framework ενώ συνδυάζει το μοντέλο αρχιτεκτονικής σχεδίασης MVC με την αρχή Convention over Configuration (μια σειρά από συμβάσεις και κανόνες, οι οποίοι καθορίζουν τον τρόπο λειτουργίας της εφαρμογής). Αυτό προσφέρει δυνατότητες ανάπτυξης εφαρμογών διαφορετικών ειδών (όπως ιστού, κινητές ή παραδοσιακών που τρέχουν σε περιβάλλοντα desktop) , επιτρέποντας στους προγραμματιστές να δημιουργούν ευέλικτες και προσαρμόσιμες εφαρμογές με μια κοινή βάση κώδικα.

  Το Jmix προσφέρει επίσης πλούσια λειτουργικότητα και εργαλεία για την ανάπτυξη εφαρμογών, όπως επεξεργασία δεδομένων, διαχείριση χρηστών και δικαιωμάτων, αυτόματη δημιουργία αναφορών και πολλά άλλα. Επιπλέον, παρέχει εργαλεία για τη διαχείριση του κύκλου ζωής των εφαρμογών, όπως ενημερώσεις, αναβαθμίσεις και διαχείριση εξαρτήσεων. Επίσης, παρέχει δυνατότητες επέκτασης με πρόσθετα (plugins) όπου επιτρέπει στους προγραμματιστές να προσθέτουν τη δική τους λειτουργικότητα σε αυτό .

  Το Jmix επιτρέπει ακόμη τη χρήση διαφόρων παραγωγικών εργαλείων (ειδικότερα IDEs) για την ανάπτυξη εφαρμογών, όπως τα IntelliJ IDEA, Eclipse και NetBeans. Τέλος, το Jmix είχε μια σχετικά μικρή αλλά ενεργή κοινότητα που υποστήριζε και συνέβαλε στην ανάπτυξή του. Ωστόσο, το μέγεθος και η ενεργητικότητα της κοινότητας μπορεί να έχει αυξηθεί από την αρχή του Jmix και μετά, καθώς άλλοι προγραμματιστές και εταιρείες ενδιαφέρονται να χρησιμοποιήσουν το πλαίσιο για την ανάπτυξη των εφαρμογών τους.

3. Θεωρητική Ανάλυση Ασφάλειας Πλαισίων Ιστού

Σε αυτή την ενότητα θα αναλύσουμε σε θεωρητικό επίπεδο τα χαρακτηριστικά ασφαλείας των πλαισίων ανάπτυξης εφαρμογών ιστού που επιλέχθηκαν στο προηγούμενο κεφάλαιο. Η ανάλυση θα πραγματοποιηθεί με βάση κάποια κριτήρια, τα οποία μερικά έχουν εφευρεθεί από εμάς και τα περισσότερα έχουν αντληθεί από την βιβλιογραφία, θα τα ορίσουμε παρακάτω. Για να πραγματοποιηθεί αποτίμηση των κριτηρίων αυτών μας ενδιαφέρει να ελέγξουμε αν ένα πλαίσιο υποστηρίζει πολλαπλούς μηχανισμούς ασφάλειας και όχι μόνο έναν ανά κριτήριο.

3.1 Κριτήρια Ανάλυσης

3.1.1 Αυθεντικοποίηση (Authentication)

Είναι ένα κριτήριο ασφάλειας που αφορά στη διαδικασία επαλήθευσης της ταυτότητας ενός χρήστη προτού αυτός αποκτήσει πρόσβαση σε προστατευόμενους πόρους ή δεδομένα (μιας εφαρμογής). Κατά τη διαδικασία της αυθεντικοποίησης, ο χρήστης παρέχει διαπιστευτήρια, όπως όνομα χρήστη και κωδικό πρόσβασης, τα οποία ελέγχονται έτσι ώστε να επιβεβαιωθεί η ταυτότητά του. Η αυθεντικοποίηση είναι σημαντική για την προστασία από την παράνομη πρόσβαση σε προστατευόμενους πόρους και δεδομένα. Η έλλειψη αυθεντικοποίησης επιτρέπει σε κακόβουλους χρήστες ή κακόβουλο λογισμικό να αποκτήσουν πρόσβαση σε προστατευόμενες πληροφορίες ή να εκτελέσουν επιθέσεις σε ένα σύστημα. Παρακάτω ακολουθεί ανάλυση μηχανισμών ταυτοποίησης που μπορούν να υποστηριχθούν από ένα πλαίσιο:

- *Κλασική ταυτοποίηση*: μέσω διαπιστευτηρίων (όνομα χρήστη και κωδικός πρόσβασης) από την ίδια την εφαρμογή.
- *OAuth (Open Authentication - Ανοικτή Ταυτοποίηση)*: Είναι ένα ανοιχτό πρότυπο πρωτόκολλο που επιτρέπει σε μια εφαρμογή να ζητήσει και να λάβει ασφαλή πρόσβαση σε πόρους εξ' ονόματος ενός χρήστη χωρίς να αποκαλύπτονται οι διαπιστευτήριά του χρήστη στην εφαρμογή. Συνήθως χρησιμοποιείται για την εξουσιοδότηση πρόσβασης τρίτων εφαρμογών σε περιορισμένους πόρους ενός χρήστη, όπως φωτογραφίες σε μια κοινωνική δικτύωση ή ηλεκτρονικό ταχυδρομείο. Ο χρήστης παρέχει συναίνεση στην τρίτη εφαρμογή για να αποκτήσει πρόσβαση σε συγκεκριμένους πόρους, χωρίς να αποκαλύπτει τα πραγματικά του διαπιστευτήρια.
- *Αυθεντικοποίηση μέσω πιστοποιητικού (certificate-based authentication)*: Ένα ψηφιακό πιστοποιητικό είναι ένα αρχείο που περιέχει πληροφορίες για τον κάτοχό του, όπως το όνομα, την διεύθυνση email του και ένα δημόσιο κρυπτογραφικό κλειδί. Η διαδικασία αυθεντικοποίησης με πιστοποιητικό λειτουργεί ως εξής:
    1. Ο χρήστης αιτείται πρόσβαση σε μια προστατευμένη περιοχή ή υπηρεσία.

2. Ο διακομιστής που φιλοξενεί την προστατευμένη περιοχή ή υπηρεσία απαιτεί το πιστοποιητικό από το χρήστη.
3. Ο χρήστης στέλνει το πιστοποιητικό του στον διακομιστή.
4. Ο διακομιστής ελέγχει το πιστοποιητικό με τη βοήθεια της αρχής πιστοποίησης (CA).
5. Αν το πιστοποιητικό είναι έγκυρο και η αρχή πιστοποίησης (CA) είναι αξιόπιστη, ο διακομιστής παρέχει πρόσβαση στην προστατευόμενη περιοχή ή υπηρεσία στον χρήστη.

- *Αυθεντικοποίηση με βάση εξωτερική υπηρεσία LDAP (Lightweight Directory Access Protocol)*: χρησιμοποιείται συνήθως σε περιβάλλοντα εταιρικών δικτύων όπου υπάρχει ανάγκη για αυθεντικοποίηση χρηστών σε πολλές εταιρικές εφαρμογές και συστήματα. Μέσω του LDAP, οι χρήστες μπορούν να αυθεντικοποιηθούν με ασφάλεια και να αποκτήσουν πρόσβαση σε πολλές εφαρμογές μέσω ενός κοινού σημείου ελέγχου πρόσβασης (single access control point). Η διαδικασία αυθεντικοποίησης με χρήση του πρωτοκόλλου LDAP λειτουργεί ως εξής:
    1. Ο χρήστης αιτείται πρόσβαση σε μια προστατευμένη περιοχή ή υπηρεσία.
    2. Ο διακομιστής που φιλοξενεί την προστατευμένη περιοχή ή υπηρεσία απαιτεί τα διαπιστευτήρια του χρήστη (όνομα χρήστη και κωδικό πρόσβασης).
    3. Ο χρήστης εισάγει το όνομα και τον κωδικό του στην υπηρεσία.
    4. Ο διακομιστής στέλνει το αίτημα αυθεντικοποίησης στον εξωτερικό LDAP εξυπηρετητή.
    5. Ο LDAP εξυπηρετητής αναζητά τα στοιχεία του χρήστη στον κατάλογο LDAP.
    6. Αν το όνομα χρήστη και ο κωδικός πρόσβασης του χρήστη είναι έγκυρα, ο LDAP εξυπηρετητής επιστρέφει μια επιτυχή απάντηση στον διακομιστή.
    7. Ο διακομιστής ελέγχει την απάντηση του LDAP εξυπηρετητή και, αν είναι επιτυχής, παρέχει την πρόσβαση στην προστατευμένη περιοχή ή υπηρεσία στο χρήστη.

- *Single Sign-On (SSO)*: είναι ένα πρωτόκολλο που επιτρέπει σε έναν χρήστη να συνδεθεί μια φορά σε ένα κεντρικό σημείο ταυτοποίησης και να αποκτήσει πρόσβαση σε πολλές εφαρμογές ή υπηρεσίες, χωρίς να απαιτείται νέα είσοδος διαπιστευτηρίων. Με άλλα λόγια, ένας χρήστης μπορεί να συνδεθεί και να έχει πρόσβαση σε πολλαπλές εφαρμογές χωρίς να χρειάζεται να παρέχει τα στοιχεία ταυτοποίησης του πολλαπλές φορές.

3.1.2 Εξουσιοδότηση (Authorization)

Είναι ένα κριτήριο ασφάλειας που αναφέρεται στη διαδικασία ελέγχου που προσπαθεί να διασφαλίσει ότι ένας (ταυτοποιημένος) χρήστης έχει το κατάλληλο επίπεδο πρόσβασης σε συγκεκριμένους πόρους ή λειτουργίες σε ένα σύστημα λογισμικού. Ο έλεγχος εξουσιοδότησης βασίζεται στο γεγονός πως κάθε χρήστης πρέπει να έχει πρόσβαση μόνο στα απαραίτητα δεδομένα και λειτουργίες για να μπορεί να εκτελέσει τις εργασίες του ανάλογα με το είδος/ρόλο του. Η εξουσιοδότηση μπορεί να επιτευχθεί με τη χρήση διαφόρων μηχανισμών ασφάλειας όπως τους παρακάτω :

- *Έλεγχος Πρόσβασης με βάση τους Ρόλους (Role-Based Access Control - RBAC)*: Με το RBAC, ανατίθενται συγκεκριμένοι ρόλοι (roles) στους χρήστες της εφαρμογής. Οι ρόλοι αυτοί έχουν πρόσβαση στις λειτουργίες της εφαρμογής με βάση τα δικαιώματα που έχουν οριστεί και αντιστοιχιστεί στους ρόλους αυτούς. Τα επίπεδα πρόσβασης καθορίζονται από τους ρόλους που έχουν οριστεί στο σύστημα. Κάθε λειτουργία της εφαρμογής μπορεί να είναι διαθέσιμη για έναν συγκεκριμένο ρόλο ή για πολλαπλούς ρόλους με πιθανώς διαφορετικά δικαιώματα. Με αυτόν τον τρόπο, μπορούν να καθοριστούν διαφορετικά επίπεδα πρόσβασης για διαφορετικούς ρόλους, προσφέροντας έτσι μια ευέλικτη διαχείριση πρόσβασης.
- *Έλεγχος Πρόσβασης βασισμένος σε Χαρακτηριστικά (Attribute-Based Access Control - ABAC)*: Στο ABAC, καθορίζονται κανόνες πρόσβασης βασισμένοι στις ιδιότητες του χρήστη και του αντικειμένου (προς πρόσβαση), όπως τον ρόλο, τη θέση (η θέση αναφέρεται συνήθως στον ρόλο ή τη θέση που διατελεί κάποιος χρήστης στον οργανισμό ή την εταιρεία. Οι χρήστες που κατέχουν την ίδια θέση συνήθως έχουν παρόμοια δικαιώματα πρόσβασης στους πόρους του συστήματος) και την κατηγορία τους αλλά και σε περιβαλλοντικές ιδιότητες, οι οποίες αναφέρονται στα στοιχεία που αφορούν το περιβάλλον εκτέλεσης, όπως ο τρέχων χρόνος, η τοποθεσία (Η τοποθεσία στον έλεγχο πρόσβασης αναφέρεται συνήθως στη φυσική ή λογική τοποθεσία ενός χρήστη ή συσκευής. Αυτό μπορεί να είναι το γεωγραφικό περιβάλλον, όπως η διεύθυνση IP της συσκευής ή ο φυσικός της τόπος, όπως η τοποθεσία μιας κινητής συσκευής), η κατάσταση του συστήματος, οι δικτυακές συνθήκες κτλ. Αυτές οι ιδιότητες μπορούν να ληφθούν υπόψη κατά τον έλεγχο της πρόσβασης σε πόρους ή λειτουργίες ενός συστήματος μέσω συνθηκών που τις εμπεριέχουν, είτε απλές ή σύνθετες. Επομένως, οι συνθήκες των αντίστοιχων κανόνων ελέγχονται και η πρόσβαση στους αντίστοιχους πόρους ή υπηρεσία επιτρέπεται ή απορρίπτεται με βάση το αποτέλεσμα του ελέγχου αυτού.
- *Ασφάλεια ζώνης (Zone security)*: δημιουργία περιοχών ασφαλείας (ζωνών) που μπορούν να χρησιμοποιηθούν για τη διαχείριση της πρόσβασης σε διαφορετικά μέρη της εφαρμογής. Κάθε ζώνη αποτελεί έναν καθορισμένο χώρο με συγκεκριμένους κανόνες ασφαλείας και ελέγχου πρόσβασης. Ο σκοπός της δημιουργίας ζωνών είναι να περιορίσει τον ελεύθερο χειρισμό. Ένα κλασικό παράδειγμα της ασφάλειας περιοχής είναι ένα κτίριο με πολλαπλές ζώνες ασφαλείας. Σε αυτή την περίπτωση, μπορεί να υπάρχει μια δημόσια προσβάσιμη περιοχή για το κοινό, μια περιοχή μόνο για υπαλλήλους με περιορισμένη πρόσβαση και μια ακόμα περιοχή, η κρισιμότερη, όπου βρίσκονται οι πλέον ευαίσθητες πληροφορίες ή εξοπλισμός, με αυστηρότερα μέτρα ασφαλείας. Με την εφαρμογή της ασφάλειας περιοχής, οι ζώνες είναι

σχεδιασμένες να προστατεύονται από ανοικτή πρόσβαση και εξωτερικές απειλές, ενώ παρέχουν, επίσης, έναν μηχανισμό ελέγχου για την πρόσβαση των εξουσιοδοτημένων χρηστών σε συγκεκριμένες περιοχές. Ένας τρόπος διαχωρισμού της εφαρμογής σε ζώνες ασφαλείας είναι η χρήση ενός λογισμικού επιπέδου εφαρμογής που υποστηρίζει τη διαχείριση της ασφάλειας. Τέτοια λογισμικά είναι τα τείχη προστασίας εφαρμογών (application firewalls) και οι διακομιστές προστασίας εφαρμογών ιστού (web application protection servers). Οι ζώνες ασφαλείας σχετίζονται άμεσα με την πρόσβαση στα μέρη της εφαρμογής που αντιστοιχούν σε κάθε ζώνη.

3.1.3 Προστασία από Cross-Site Request Forgery (CSRF) επιθέσεις

Ο CSRF είναι ένας τύπος επιθέσεων ασφαλείας που εστιάζει σε διαδικτυακές εφαρμογές. Με τη χρήση κακόβουλου λογισμικού στοχεύει στην εκτέλεση εντολών στο όνομα ενός εξουσιοδοτημένου χρήστη χωρίς την συγκατάθεσή του. Η επίθεση αυτή λαμβάνει χώρα όταν ένας κακόβουλος χρήστης δημιουργεί μια παραπλανητική σελίδα που περιέχει κώδικα που εκτελεί αυτόματα εντολές σε μια ιστοσελίδα στόχο, χρησιμοποιώντας τα διαπιστευτήρια του χρήστη. Ο κακόβουλος χρήστης μπορεί να στείλει μέσω ηλεκτρονικού μηνύματος το URL της ιστοσελίδας αυτής και πατώντας το, το θύμα ουσιαστικά επιτρέπει την εκτέλεση κακόβουλων ενεργειών χωρίς τη συγκατάθεσή του. Παραδείγματος χάρη, αν ένας χρήστης είναι συνδεδεμένος στο λογαριασμό του σε μια τράπεζα και επισκέπτεται μια κακόβουλη ιστοσελίδα, η οποία περιέχει κώδικα που ζητά αυτόματα μια μεταφορά χρημάτων σε έναν λογαριασμό που ελέγχεται από τον επιτιθέμενο, τότε ο λογαριασμός του θα μπορούσε να χρησιμοποιηθεί για την εκτέλεση αυτής της μεταφοράς χωρίς την άδειά του. Ο κακόβουλος κώδικας δεν έχει άμεση πρόσβαση στον λογαριασμό του χρήστη ή στις πληροφορίες ταυτοποίησής του. Αντίθετα, εκμεταλλεύεται την εμπιστοσύνη του χρήστη στην εφαρμογή και τη δυνατότητα της εφαρμογής να εκτελεί ενέργειες με τα δικαιώματα του χρήστη χωρίς ενεργή έγκριση. Η κακόβουλη ιστοσελίδα στην ουσία εκμεταλλεύεται τα δικαιώματα που παρέχονται από το λογαριασμό του χρήστη και του επιτρέπουν να πραγματοποιήσει συναλλαγές όπως την προαναφερόμενη. Προστασία από τέτοιου είδους επιθέσεις παρέχουν οι εξής μηχανισμοί ασφάλειας:

- *Built-in CSRF protection*: είναι ένας μηχανισμός που παρέχεται απευθείας από ένα πλαίσιο ανάπτυξης λογισμικού για την προστασία από επιθέσεις Cross-Site Request Forgery (CSRF). Ουσιαστικά ο χρήστης δεν κάνει κάτι και όλες οι ενέργειες πραγματοποιούνται από το πλαίσιο οπότε δυνητικά μπορεί να χρησιμοποιηθεί είτε το CSRF token είτε το DSC από το ίδιο το πλαίσιο ενώ στα παρακάτω το πλαίσιο επιτρέπει στον προγραμματιστή, για παράδειγμα να δημιουργεί ρητά tokens και να τα ενσωματώνει μετά στις φόρμες.
- *Χρήση μίας CSRF ένδειξης (token)*: πρέπει να συμπεριλαμβάνεται σε κάθε αίτημα που προέρχεται από τον χρήστη. Αυτό το token δημιουργείται από τον εξυπηρετητή κατά την είσοδο του χρήστη στο σύστημα και αποθηκεύεται στην σύνοδο του χρήστη. Κάθε φορά που ο χρήστης υποβάλλει ένα αίτημα, το token συμπεριλαμβάνεται στη φόρμα ή το URL του αιτήματος και επικυρώνεται από το σύστημα. Η φόρμα παράγεται από

τον εξυπηρετητή ενσωματώνοντας το token κι αυτό δεν μπορεί να κλαπεί μέσω script άλλης ιστοσελίδας. Αν η αίτηση δεν περιλαμβάνει το σωστό token, τότε απορρίπτεται από τον εξυπηρετητή.

- *Double Submit Cookie (DSC)*: Ο τρόπος λειτουργίας του DSC είναι ο εξής:

1) Όταν ο χρήστης συνδέεται στην εφαρμογή, ο διακομιστής δημιουργεί μια τυχαία τιμή (nonce). Αν είναι naive, δεν απαιτείται αποθήκευση, Αν είναι signed, τότε η τιμή γίνεται bound (δεσμεύεται) στην τρέχουσα σύνοδο.
2) Κατά την κάθε αίτηση προς τον διακομιστή, ο πελάτης συμπεριλαμβάνει το DSC ως header στο αίτημα.
3) Η τιμή του DSC αποθηκεύεται ως cookie (και άρα στην κεφαλίδα) και παρέχεται και ως παράμετρος της αίτησης (request parameter). Αν οι δύο τιμές ταιριάζουν, τότε η αίτηση θεωρείται έγκυρη και επιτρέπεται η εκτέλεσή της. Αν οι τιμές δεν ταιριάζουν, τότε η αίτηση απορρίπτεται ως πιθανή επίθεση CSRF. Ο μηχανισμός DSC επιτρέπει την αποτροπή των επιθέσεων CSRF χωρίς την ανάγκη χρήσης συγκεκριμένων τεχνικών αυθεντικοποίησης, όπως tokens αυθεντικοποίησης.

3.1.4 Προστασία από Cross-site scripting (XSS) επιθέσεις

Είναι μια από τις πιο κοινές ευπάθειες που εντοπίζονται σε εφαρμογές ιστού. Οι επιθέσεις Cross-Site Scripting συμβαίνουν όταν ένας κακόβουλος κώδικας JavaScript εισάγεται αποτελεσματικά από έναν επιτιθέμενο σε μια νομότυπη ιστοσελίδα, ώστε οι χρήστες που αιτούνται την εκφόρτωση της σελίδας να τον εκτελούν όταν αυτή φορτώνεται. Η εκτέλεση κακόβουλου κώδικα JavaScript στη ιστοσελίδα μπορεί να επιτρέψει στον επιτιθέμενο να κλέψει προσωπικά δεδομένα των χρηστών, να παρακολουθήσει τις ενέργειες τους, να αλλοιώσει το περιεχόμενο της ιστοσελίδας ή να υλοποιήσει επιθέσεις στους επισκέπτες της. Οι τρόποι προστασίας που μπορεί να παρέχονται από ένα πλαίσιο περιλαμβάνουν:

- *Thymeleaf*: Το Thymeleaf είναι μια μηχανή προτύπων (template engine), τα οποία βοηθούν στην αποφυγή αυτού του κινδύνου. Το Thymeleaf δεν επιτρέπει σε έναν κακόβουλο χρήστη να εισάγει κώδικα HTML, JavaScript και άλλα επικίνδυνα στοιχεία στο περιεχόμενο ενός ιστοτόπου, ενώ παρέχει μια σειρά από χρήσιμες λειτουργίες για τη διαχείριση των δεδομένων εισαγωγής χρήστη, όπως η δυνατότητα κωδικοποίησης, αποκωδικοποίησης και αντικατάστασης ειδικών χαρακτήρων.
- Το *CSP (Content Security Policy)*: είναι ένας μηχανισμός ασφαλείας που χρησιμοποιείται στο περιβάλλον του Παγκόσμιου Ιστού για την προστασία από επιθέσεις XSS (Cross-Site Scripting). Ο στόχος του CSP είναι να εφαρμόζει περιορισμούς στους πόρους που μπορούν να φορτωθούν στις σελίδες του ιστού. Το CSP σχεδιάστηκε ώστε οι περιηγητές (browsers) που δεν το υποστηρίζουν να εξακολουθούν να λειτουργούν με εξυπηρετητές (servers) που το εφαρμόζουν και αντίστροφα. Οι περιηγητές που δεν υποστηρίζουν τον CSP τον αγνοούν, λειτουργώντας κανονικά και προεπιλέγοντας την πρότυπη πολιτική ίδιας προέλευσης,

(ασφαλούς μηχανισμού περιορισμού στις περιηγήσεις του διαδικτύου). Ο μηχανισμός αυτός εφαρμόζεται από τους περιηγητές (π.χ. Chrome, Firefox, Safari) για να προστατεύσουν τους χρήστες από επιθέσεις και παραβιάσεις από άγνωστες πηγές κώδικα για το περιεχόμενο του ιστού. Εάν ο ιστότοπος δεν προσφέρει την κεφαλίδα CSP, οι περιηγητές χρησιμοποιούν επίσης την πρότυπη πολιτική ίδιας προέλευσης. Σύμφωνα με την πολιτική ίδιας προέλευσης (SOP), ένας ιστότοπος έχει πρόσβαση μόνο στους πόρους (όπως JavaScript, CSS, cookies κλπ.) από άλλες διευθύνσεις URL που έχουν την ίδια προέλευση (scheme, host, port) με τον ιστότοπο. Σε απλά λόγια, αυτό σημαίνει ότι ένας ιστότοπος δεν μπορεί να αναφερθεί σε πόρους που βρίσκονται σε άλλους ιστότοπους, παρά μόνο σε πόρους που προέρχονται από τον ίδιο ιστότοπο.

- *Κωδικοποίηση Εξόδου (Output Encoding)*: Η χρήση του μηχανισμού αυτού είναι ένα από τα σημαντικότερα μέτρα προστασίας από επιθέσεις XSS (Cross-Site Scripting). Με τον μηχανισμό αυτό, τα δεδομένα που εμφανίζονται από έναν ιστότοπο κωδικοποιούνται πριν από την εμφάνισή τους, γεγονός που αποτρέπει την εκτέλεση κακόβουλου κώδικα JavaScript ή HTML μέσω αυτών . Ο κακόβουλος κώδικας XSS συνήθως εισάγεται στην εφαρμογή μέσω μη αξιόπιστων πηγών, όπως της εισόδου του χρήστη (όπως σε μια φόρμα εισαγωγής σχολίων) ή από εξωτερικές πηγές δεδομένων. Όταν τα δεδομένα αυτά εμφανίζονται από την ιστοσελίδα χωρίς κωδικοποίηση, ο περιηγητής του χρήστη μπορεί να εκτελέσει τον κακόβουλο κώδικα ως JavaScript ή HTML, με αποτέλεσμα να πραγματοποιηθούν επιθέσεις XSS.

3.1.5 Επιθέσεις Έκχυσης SQL

Είναι μια μέθοδος εισαγωγής ή επέκτασης SQL κώδικα σε μια εφαρμογή ιστού που επιτρέπει σε έναν επιτιθέμενο να αποκτήσει, να διαγράψει ή να τροποποιήσει ευαίσθητα δεδομένα από μια βάση δεδομένων της εφαρμογής αυτής. Ο εισβολέας επιτυγχάνει αυτήν την επίθεση όταν η εφαρμογή αποδέχεται την είσοδο του ως έχει και στη συνέχεια τη χρησιμοποιεί ως μέρος των δηλώσεων SQL προς την υποκείμενη βάση δεδομένων χωρίς τη κατάλληλη επικύρωση και καθαρισμό. Η είσοδος αυτή επεκτείνει τις δηλώσεις SQL ώστε όταν αυτές εκτελεστούν είτε να επιστρέφονται επιπλέον αποτελέσματα είτε να διαγράφονται/τροποποιούνται δεδομένα με μη εξουσιοδοτημένο τρόπο Η επίθεση έκχυσης SQL μπορεί να οδηγήσει στην παράκαμψη της εξουσιοδότησης, στην απώλεια δεδομένων/εγγράφων, στην απόρριψη πρόσβασης σε νόμιμους χρήστες καθώς και στην κατάρρευση ολόκληρης της βάσης δεδομένων. Η προστασία από τέτοιου είδους επιθέσεις μπορεί να επιτευχθεί με τους εξής μηχανισμούς:

- *Έτοιμες δηλώσεις (prepared statements)*: Ο μηχανισμός των έτοιμων δηλώσεων εκτελεί παραμετροποιημένες SQL επερωτήσεις, οι οποίες είναι προκαθορισμένες και η δομή ή κώδικάς τους δεν μπορεί να αλλάξει, παρά μόνο να αντικατασταθούν συγκεκριμένα μέρη τους με τις τιμές που δίνει η αντίστοιχη εφαρμογή (που μπορούν να παρέχονται από τον χρήστη). Οπότε, αν ο χρήστης δώσει μια τιμή που οδηγεί σε επέκταση της επερώτησης, η επέκταση αυτή θα αγνοείται.
- *Φίλτρα Αποτροπής SQL Έκχυσης (SQL Injection Filters)*: μηχανισμός που περιλαμβάνει φίλτρα, τα οποία εφαρμόζονται στα δεδομένα που λαμβάνονται από

τους χρήστες, πριν από την ενσωμάτωσή τους σε SQL επερωτήσεις προς τη βάση δεδομένων, για την αναγνώριση και απόρριψη επιθέσεων έκχυσης SQL. Τα φίλτρα αυτά ελέγχουν αν υπάρχουν επικίνδυνα στοιχεία SQL (όπως, π.χ., εντολές INSERT, UPDATE, DELETE) στα δεδομένα και τα αφαιρούν προτού ενσωματωθούν στις SQL επερωτήσεις. Ένα από αυτά τα φίλτρα που μπορούν να χρησιμοποιηθούν είναι το εξής:

- *Java Servlet Filters*: Τα Java Servlet Filters είναι κομμάτια κώδικα που εκτελούνται πριν ή μετά από την επεξεργασία μιας αίτησης από ένα Servlet (πχ. μια υπηρεσία ιστού) σε έναν web container (όπως ο Tomcat). Αυτά τα φίλτρα προσφέρουν μια εύκολη και ευέλικτη τεχνική για την επεξεργασία και τη μετατροπή των δεδομένων των αιτήσεων και των αποκρίσεων.
- *Επικύρωση εισόδου (Input Validation)*: τεχνικές επαλήθευσης εισόδου, όπως regex (σύνολο κανόνων που περιγράφουν ένα πρότυπο αναζήτησης σε ένα κείμενο), whitelist ή blacklist για να αποτραπεί η εισαγωγή επικίνδυνων δεδομένων στη βάση δεδομένων. Η λευκή λίστα (whitelist) περιλαμβάνει μια λίστα αποδεκτών στοιχείων ή ενεργειών που επιτρέπονται, ενώ η μαύρη λίστα (blacklist) περιλαμβάνει μια λίστα απαγορευμένων στοιχείων ή ενεργειών που απαγορεύονται.

3.1.6 Διαχείριση συνόδων (Session management)

Οι σύνοδοι είναι σημαντικές στις εφαρμογές ιστού καθώς παρέχουν στους χρήστες πρόσβαση σε προσωπικά δεδομένα και προσωποποιημένες λειτουργίες. Η αδυναμία ασφαλούς διαχείρισης συνόδων μπορεί να οδηγήσει σε σοβαρές απειλές ασφαλείας, όπως η αποκάλυψη προσωπικών δεδομένων ή η υποκλοπή ταυτότητας χρηστών. Παρακάτω αναλύονται μέτρα ασφάλειας για την προστασία των συνόδων:

- *Χρήση Τυχαίων κλειδιών (Random Tokens)*: Η χρήση των τυχαίων κλειδιών επιτρέπει στην εφαρμογή να αναγνωρίζει και να επικυρώνει τους χρήστες χωρίς να αποθηκεύει τα προσωπικά τους δεδομένα, όπως ονόματα χρηστών και κωδικοί πρόσβασης. Αντ' αυτού, κάθε φορά που ο χρήστης συνδέεται στην εφαρμογή, του δίνεται ένα τυχαίο token, το αναγνωριστικό συνόδου, που αντιπροσωπεύει τον λογαριασμό του. Αντί να βασίζεται σε σταθερές ταυτοποιήσεις, όπως τα cookies, τα τυχαία κλειδιά παράγονται με τυχαίο τρόπο και σχετίζονται με τη σύνοδο του χρήστη. Αυτά τα κλειδιά είναι μοναδικά για κάθε σύνοδο και αλλάζουν μετά από κάθε χρήση, δηλαδή αν η τρέχουσα σύνοδος τελειώσει τότε η νέα σύνοδος θα έχει ένα διαφορετικό τυχαίο κλειδί. Με τη χρήση τυχαίων κλειδιών, ακόμη και αν ένας κακόβουλος χρήστης καταφέρει να αποκτήσει πρόσβαση σε μια ενδεχόμενη ένδειξη, δεν μπορεί να την επαναχρησιμοποιήσει σε επόμενες συνόδους, καθώς έχει αλλάξει. Αυτό προσφέρει ένα επιπρόσθετο στρώμα ασφαλείας και δυσκολεύει τους επιτιθέμενους να παρακολουθήσουν, να παρεμβληθούν ή να παραβιάσουν τις συνόδους των χρηστών.
- *Κατάλληλη διαμόρφωση των cookies*: ειδικές σημαίες τύπου Cookie Flags για την επιβολή των απαιτήσεων ασφαλείας σε cookies, όπως η HttpOnly, η οποία αποτρέπει

την πρόσβαση στα cookies από τον πελάτη μέσω JavaScript, και η Secure, η οποία απαιτεί τη χρήση HTTPS για τη μεταφορά των cookies.

- *Διαχείριση Ιδιοτήτων συνόδου (Session Attributes)*: Η διαχείριση των ιδιοτήτων συνόδου (Session Attributes) έχει στενή σχέση με την ασφάλεια του session management και την προστασία των δεδομένων των χρηστών. Οι ιδιότητες συνόδου αναφέρονται στις πληροφορίες που αποθηκεύονται και διατηρουνται κατά τη διάρκεια της συνεδρίας του χρήστη στον διακομιστή. Ένας από τους τρόπους που συμβάλει η διαχείριση των ιδιοτήτων συνόδου στην ασφάλεια του session management είναι η προστασία ευαίσθητων δεδομένων, δηλαδή όταν αποθηκεύονται δεδομένα συνόδου, θα πρέπει να δοθεί ιδιαίτερη προσοχή στα ευαίσθητα δεδομένα όπως κωδικοί πρόσβασης, προσωπικά στοιχεία κ.λπ. Σημαντικό είναι να μην αποθηκεύονται αυτά τα δεδομένα στις ιδιότητες συνόδου, αλλά να αποθηκεύονται στον διακομιστή με ασφαλή τρόπο (όπως στη βάση δεδομένων).
- *Διαχείριση συνόδου με βάση δεδομένων (Session Management with Database Storage):* Σε αυτήν την προσέγγιση, τα ευαίσθητα δεδομένα των συνεδριών αποθηκεύονται σε μια βάση δεδομένων στον διακομιστή. Οι πληροφορίες της συνεδρίας αποθηκεύονται με ασφάλεια στη βάση δεδομένων, και το session ID αποστέλλεται στον πελάτη για να εντοπίσει την αντίστοιχη συνεδρία. Η χρήση μιας αξιόπιστης και ασφαλούς βάσης δεδομένων είναι κρίσιμη για την προστασία των ευαίσθητων δεδομένων.
- *Χρονική Λήξη Συνεδρίας (Session Timeout):* Κατά την ανάπτυξη μιας εφαρμογής, οι χρονικές λήξεις συνεδρίας πρέπει να ορίζονται κατάλληλα για να εξασφαλίσουν την ασφάλεια των δεδομένων και να προστατεύσουν την ιδιωτικότητα των χρηστών. Σκοπός αυτού του μέτρου είναι να λήγει η συνεδρία αν δεν έχει δραστηριότητα για ένα συγκεκριμένο χρονικό διάστημα. Ανάλογα με τον τύπο της εφαρμογής, μπορεί να οριστεί το χρονικό διάστημα που η συνεδρία παραμένει ενεργή χωρίς δραστηριότητα και όταν η περίοδος αυτή λήγει, ο χρήστης αποσυνδέεται και απαιτείται νέα επανασύνδεση. Παράλληλα, Μπορεί να οριστεί ένα συγκεκριμένο μέγιστο χρόνο διάρκειας για τις συνεδρίες, ανεξαρτήτως δραστηριότητας. Όταν περάσει ορισμένος χρόνος από την έναρξη της συνεδρίας, αυτή αυτομάτως λήγει και απαιτείται πάλι νέα επανασύνδεση.

3.1.7 Κρυπτογράφηση (Encryption)

Είναι ένα κριτήριο ασφάλειας που χρησιμοποιείται για την προστασία των δεδομένων από την ανεπιθύμητη πρόσβαση ή αποκάλυψη από τρίτους. Κατά τη μεταφορά δεδομένων από ένα άκρο σε ένα άλλο, η κρυπτογράφηση χρησιμοποιείται για την προστασία των δεδομένων κατά την αποστολή τους μέσω ενός δικτύου όπως το Διαδίκτυο. Η κρυπτογράφηση κατά την αποθήκευση δεδομένων επιτυγχάνεται με τη μετατροπή των δεδομένων σε μια μορφή που είναι δυσνόητη για άλλους (πχ. επιτιθέμενους) εκτός από τον εξουσιοδοτημένο χρήστη (δηλ. τον κάτοχό τους ή οποιοδήποτε άλλο άτομο που έχει δικαίωμα πρόσβασης σε αυτά). Αυτό βοηθά στην προστασία των δεδομένων από την πρόσβαση ή τη διαρροή από άτομα που δεν έχουν την απαιτούμενη εξουσιοδότηση. Παρακάτω παρέχονται μερικοί μηχανισμοί κρυπτογράφησης που συμβάλλουν στην προστασία των εφαρμογών ιστού:

- *Χρήση HTTPS*: Το HTTPS είναι μια πρωτόκολλο επικοινωνίας που χρησιμοποιείται στον Παγκόσμιο Ιστό για την ασφαλή μετάδοση δεδομένων ανάμεσα στον περιηγητή του χρήστη και τον διακομιστή του ιστοτόπου. Αυτό επιτυγχάνεται μέσω κρυπτογράφησης των δεδομένων που ανταλλάσσονται ανάμεσα στον χρήστη και τον διακομιστή, καθιστώντας τα δεδομένα μη αναγνώσιμα και ασφαλή.
- *Συμμετρική κρυπτογράφηση*: Στη συμμετρική κρυπτογράφηση, χρησιμοποιείται το ίδιο κλειδί τόσο για την κρυπτογράφηση όσο και για την αποκρυπτογράφηση των δεδομένων. Οι δύο οντότητες που ανταλλάσσουν δεδομένα πρέπει να συμφωνήσουν εκ των προτέρων για το κλειδί που θα χρησιμοποιηθεί. Η συμμετρική κρυπτογράφηση είναι πιο γρήγορη και αποδοτική και αυτό είναι ιδιαίτερα σημαντικό όταν πρέπει να αποθηκευτούν μεγάλος όγκος δεδομένων αλλά απαιτεί ασφάλεια στη μεταφορά του κλειδιού. Ο AES (Advanced Encryption Standard) είναι ένας αλγόριθμος κρυπτογράφησης συμμετρικού κλειδιού. Αυτό σημαίνει πως χρησιμοποιεί το ίδιο κλειδί τόσο για την κρυπτογράφηση όσο και για την αποκρυπτογράφηση των δεδομένων. Η χρήση συμμετρικών κλειδιών AES είναι ενδεδειγμένη σε περιπτώσεις που η απόδοση της εφαρμογής (ιστού) είναι κρίσιμη και η κρυπτογράφηση δεδομένων πρέπει να γίνεται γρήγορα και αποτελεσματικά. Επιπλέον, ο AES μπορεί να χρησιμοποιηθεί για την κρυπτογράφηση δεδομένων σε βάσεις δεδομένων, τα αρχεία που αποθηκεύονται στο δίσκο ή στη μνήμη, και για την προστασία δεδομένων που μεταφέρονται μεταξύ διάφορων συστημάτων και υπηρεσιών. Είναι πιο κατάλληλος για την ασφαλή ανταλλαγή δεδομένων όταν απαιτείται γρήγορη και αποτελεσματική κρυπτογράφηση μεγάλου όγκου δεδομένων, όπως στις εφαρμογές ιστού
- *Ασύμμετρη κρυπτογράφηση:* Στην ασύμμετρη κρυπτογράφηση, χρησιμοποιούνται δύο διαφορετικά κλειδιά, ένα δημόσιο και ένα ιδιωτικό. Κάθε χρήστης έχει ένα ζεύγος κλειδιών. Το δημόσιο κλειδί χρησιμοποιείται για την κρυπτογράφηση των δεδομένων, ενώ το ιδιωτικό κλειδί χρησιμοποιείται για την αποκρυπτογράφηση τους. Η ασύμμετρη κρυπτογράφηση είναι πολύ ασφαλής, αλλά πιο χρονοβόρα από τη συμμετρική κρυπτογράφηση. Ο RSA (Rivest-Shamir-Adleman) είναι ένα αλγόριθμος κρυπτογράφησης δημόσιου κλειδιού που επιτρέπει ασφαλή επικοινωνία μέσω ενός μη ασφαλούς δικτύου, όπως το Διαδίκτυο. Γενικά, η κρυπτογράφηση RSA περιλαμβάνει τη δημιουργία ενός ζεύγους δημόσιου-ιδιωτικού κλειδιού, όπου το δημόσιο κλειδί χρησιμοποιείται για την κρυπτογράφηση δεδομένων και το ιδιωτικό κλειδί χρησιμοποιείται για την αποκρυπτογράφηση των δεδομένων. Μια από τις βασικές εφαρμογές του RSA είναι η ασφαλής ανταλλαγή κλειδιών για άλλους αλγορίθμους κρυπτογράφησης, όπως η συμμετρική κρυπτογράφηση. Αυτό επιτρέπει την ασφαλή και αξιόπιστη επικοινωνία μεταξύ δύο οντοτήτων χωρίς να χρειάζεται να μοιράζονται ευαίσθητα κλειδιά. Ωστόσο, λόγω της υψηλής υπολογιστικής πολυπλοκότητας της ασύμμετρης κρυπτογράφησης, ο RSA συνήθως δεν χρησιμοποιείται για την κρυπτογράφηση μεγάλου όγκου δεδομένων

3.2 Ανάλυση

3.2.1 Ανάλυση Πλαισίων

Θεωρώντας κάθε μέτρο ασφαλείας ανά πλευρά/κριτήριο ασφάλειας χρήσιμο και πως 2 ή παραπάνω μέτρα ασφάλειας σε μια πλευρά μπορεί να είναι συμπληρωματικά, πιστεύουμε πως η αποτίμηση ασφάλειας ενός πλαισίου θα πρέπει να εστιάσει στο ποια μέτρα ασφάλειας υποστηρίζονται ανά πλευρά ασφάλειας και όχι στο ποιο από αυτά είναι το πιο ισχυρό. Παρακάτω, παραθέτουμε την τοπική αποτίμηση των πλαισίων ανά πλευρά ασφάλειας καθώς και μια καθολική αποτίμηση των πλαισίων με βάση την τοπική τους αποτίμηση.

3.2.1.1 Ανάλυση πλαισίων ως προς την Αυθεντικοποίηση

	κλασική ταυτοποίηση	OAuth	LDAP	SSO	certificate-based authentication
Spring	+	+	+	+	-
Vaadin	+	-	-	-	-
Tapestry	+	-	+	-	-
Struts	+	-	-	+	+
Jmix	+	-	-	-	-
JHipster	+	+	-	-	-

Πίνακας 1 Ανάλυση πλαισίων ως προς την Αυθεντικοποίηση

Επομένως, όσον αφορά το κριτήριο της αυθεντικοποίησης θα μπορούσαμε να καταλήξουμε στο συμπέρασμα ότι το Spring σε σχέση με τα υπόλοιπα πλαίσια παρουσιάζει τα περισσότερα μέτρα ασφαλείας (4/5).Το JMix και Vaadin στην τρέχουσα περίπτωση είναι τα πλαίσια με την χειρότερη απόδοση (1/5). Καθώς και να τονίσουμε πως η κλασική ταυτοποίηση έχει την καλύτερη δυνατή υποστήριξη αφού την χρησιμοποιούν όλα τα πλαίσια, ενώ την χειρότερη η αυθεντικοποίηση μέσω πιστοποιητικών.

3.2.1.2 Ανάλυση πλαισίων ως προς την Εξουσιοδότηση

	Role-Based access	Attribute-Based Access Control	Zone - security
Spring	+	+	-
Vaadin	+	+	-
Tapestry	+	+	-
Struts	+	+	-
Jmix	+	+	+
JHipster	+	+	-

Πίνακας 2 Ανάλυση Πλαισίων ως προς την Εξουσιοδότηση

Όσον αφορά το κριτήριο της εξουσιοδότησης, παρατηρούμε ότι τα περισσότερα μέτρα ασφαλείας έχει το Jmix (3/3). Στην τρέχουσα περίπτωση όλα τα υπόλοιπα πλαίσια έχουν ελαφρώς χειρότερη απόδοση (2/3). Επίσης, πρέπει να τονίσουμε πως τα μέτρα RBAC & ABAC έχουν την καλύτερη δυνατή υποστήριξη αφού τα ενσωματώνουν όλα τα πλαίσια, ενώ την χειρότερη δυνατή το μέτρο Zone security (ενσωματώνεται μόνο από ένα πλαίσιο).

3.2.1.3 Ανάλυση πλαισίων ως προς το Cross-Site Request Forgery (CSRF)

	CSRF token	Double Submit Cookie	built-in CSRF protection
Spring	+	+(απευθειας)	+
Vaadin	+	+	+
Tapestry	+	+	-
Struts	+	+	-
Jmix	+	+	-
JHipster	+	+	-

Πίνακας 3 Ανάλυση ως προς το CSRF

Όσον αφορά το κριτήριο CSRF, τα πλαίσια Spring και Vaadin έχουν την καλύτερη απόδοση, υποστηρίζοντας όλα τα μέτρα ασφαλείας (3/3). Ενώ όλα τα υπόλοιπα πλαίσια έχουν μια ελαφρώς χειρότερη απόδοση (2/3). Ακόμη, πρέπει να τονίσουμε πως το Double Submit Cookie και το CSRF token έχουν την καλύτερη δυνατή υποστήριξη αφού τα ενσωματώνουν όλα τα πλαίσια, ενώ την χειρότερη υποστήριξη έχει η built-in CSRF προστασία.

3.2.1.4 Ανάλυση πλαισίων ως προς την Cross-Site Scripting (XSS)

	Thymeleaf	Content Security Policy	Output Encoding
Spring	+	+	+
Vaadin	-	+	+
Tapestry	-	+	+
Struts	-	+	+
Jmix	-	+	+
JHipster	+	+	+

Πίνακας 4 Ανάλυση ως προς το XSS

Για το κριτήριο XSS, τα πλαίσια Spring και JHipster προσφέρουν όλα τα μέτρα ασφαλείας (3/3), άρα ξεχωρίζουν με την καλύτερη δυνατή απόδοση. Όλα τα υπόλοιπα πλαίσια έχουν μια ελαφρώς χειρότερη απόδοση (2/3). Τέλος, πρέπει να τονίσουμε πως τα μέτρα Content Security Policy και Output Encoding έχουν την καλύτερη δυνατή υποστήριξη αφού τα ενσωματώνουν όλα τα πλαίσια, ενώ την χειρότερη έχει το Thymeleaf (ενσωμάτωση μόνο από δύο πλαίσια).

3.2.1.5 Ανάλυση πλαισίων ως προς την SQL Injection

	Prepared Statements	SQL Injection Filters	Input Validation	Parameterized queries
Spring	+	+	+	-
Vaadin	+	-	-	-
Tapestry	+	-	-	-
Struts	+	-	-	-
Jmix	-	+	+	+
JHipster	+	-	-	+

Πίνακας 5 Ανάλυση ως προς το SQL Injection

Για το παραπάνω κριτήριο παρατηρούμε ότι ποσοτικά τα πλαίσια Spring και Jmix ξεχωρίζουν αρκετά σε σχέση με τα υπόλοιπα ως προς τα μέτρα ασφαλείας που προσφέρουν (3/4). Αντιθέτως, το Vaadin, το Tapestry και το Struts έχουν την χειρότερη απόδοση (1/4). Ακόμη, πρέπει να σημειώσουμε πως το μέτρο Prepared Statements έχει την καλύτερη δυνατή υποστήριξη αφού το ενσωματώνουν όλα τα πλαίσια εκτός του Jmix. Απεναντίας, τα άλλα 2 μέτρα έχουν την χειρότερη υποστήριξη διότι προσφέρονται μόνο από δύο πλαίσια έκαστο.

3.2.1.6 Ανάλυση πλαισίων ως προς την Διαχείριση συνόδων

	Random Tokens	Cookie - Flags	Session Management with Database Storage	Session Attributes	Session Timeout
Spring	+	+	+	+	+
Vaadin	-	-	-	+	+
Tapestry	-	-	-	+	+
Struts	-	-	-	+	+
Jmix	-	-	-	+	+
JHipster	-	-	-	+	+

Πίνακας 6 Ανάλυση ως προς τη Διαχείριση Συνόδων

Σχετικά με το κριτήριο της διαχείρισης συνόδων, παρατηρούμε ότι το Spring προσφέρει τα περισσότερα μέτρα ασφαλείας (5/5), τα υπόλοιπα πλαίσια παρέχουν μόνο δύο μέτρα, έχοντας την χειρότερη απόδοση. Ακόμα, θα πρέπει να τονίσουμε πως το μέτρο Session Attributes και το Session Timeout έχουν την καλύτερη δυνατή υποστήριξη αφού τα προσφέρουν όλα τα πλαίσια. Αντιθέτως, τα άλλα τρία μέτρα έχουν την χειρότερη υποστήριξη διότι προσφέρονται μόνο από δύο πλαίσια έκαστο.

3.2.1.7 Ανάλυση πλαισίων ως προς την Κρυπτογράφηση

	HTTPS πρωτόκολλο	Advanced Encryption Standard (AES)	Ασύμμετρη κρυπτογραφία (RSA)
Spring	+	+	+
Vaadin	+	-	+
Tapestry	+	-	-
Struts	-	+	-
Jmix	-	+	-
JHipster	+	+	-

Πίνακας 7 Ανάλυση ως προς την Κρυπτογράφηση

Τέλος, όσον αφορά το κριτήριο της κρυπτογράφησης, παρατηρούμε ότι το Spring έχει τα περισσότερα μέτρα ασφαλείας από τα υπόλοιπα (3/3). Αντιθέτως, το Tapestry, το Struts και το Jmix έχουν την χειρότερη απόδοση (1/3). Τέλος, θα πρέπει να σημειώσουμε πως το πρωτόκολλο HTTPS και το AES έχουν την καλύτερη δυνατή υποστήριξη αφού τα προσφέρουν όλα τα πλαίσια εκτός από δύο, ενώ την χειρότερη το RSA καθώς υποστηρίζεται μόνο από δύο πλαίσια.

3.2.2. Καθολική Ανάλυση

Παρακάτω παρουσιάζεται ένας συγκεντρωτικός πίνακας με βάση την ανάλυση των κριτηρίων που έγινε προηγουμένως. Δεδομένου ότι δεν υποστηρίζεται ένα μέγιστο επίπεδο ασφάλειας και όλα τα πλαίσια υστερούν σε κάποια κριτήρια, θα θεωρήσουμε ότι στην αποτίμηση μας όποιο πλαίσιο έχει τα περισσότερα χαρακτηριστικά παίρνει την μεγαλύτερη βαθμολογία και αναλόγως με το κατά πόσο υστερούν τα υπόλοιπα σε σχέση με αυτό ορίζουμε τη βαθμολογία τους. Τέλος, υπολογίζουμε ένα μέσο όρο ασφάλειας για κάθε πλαίσιο λαμβάνοντας υπόψη την τιμή αποτίμησης του για κάθε κριτήριο.

	Authentication	**Authorization**	**CSRF**	**XSS**	**SQL Injection**	**Session Management**	**Cryptography**	**Total Score M.O**
Spring	4/5	2/3	3/3	3/3	3/4	5/5	3/3	8.8/10
JHipster	2/5	2/3	2/3	3/3	2/4	2/5	2/3	6.1/10
Jmix	1/5	3/3	2/3	2/3	3/4	2/5	1/3	5.7/10
Vaadin	1/5	2/3	3/3	2/3	1/4	2/5	2/3	5.5/10
Struts	3/5	2/3	2/3	2/3	1/4	2/5	1/3	5.1/10
Tapestry	2/5	2/3	2/3	2/3	1/4	2/5	1/3	4.8/10

Πίνακας 8 Καθολική Ανάλυση

Αξιολογώντας την αποτίμηση που κάναμε, τη μεγαλύτερη τιμή συγκεντρώνει το πλαίσιο Spring έχοντας τις περισσότερες “νίκες” έναντι των υπολοίπων. Το μόνο κριτήριο στο οποίο φαίνεται να μην υπερέχει το Spring είναι το Authorization. Το αποτέλεσμα για τον μέσο όρο ανά πλαίσιο προκύπτει από την πρόσθεση όλων τον βαθμολογιών ανά κριτήριο και την διαίρεση του συνόλου με τον αριθμό των συνολικών κριτηρίων. Κρίνοντας την ασφάλεια με βάση το μέσο όρο το μεγαλύτερο βαθμό συγκεντρώνει ξανά το Spring (8.8/10) και αμέσως επόμενο το JHipster (6.1/10). Συνεπώς, θεωρείται ότι προσφέρει περισσότερους μηχανισμούς/μέτρα ασφάλειας ανά κριτήριο που επιτρέπουν την επίτευξη μεγαλύτερου επιπέδου ασφάλειας για εφαρμογές ιστού σε σχέση με τα υπόλοιπα πλαίσια που εξετάστηκαν. Το χειρότερο πλαίσιο ως προς το πλήθος επιλογών είναι το Tapestry με συνολική βαθμολογία (4.8/10) καθώς υστερεί σε κάποια από τα παραπάνω κριτήρια.

Όσον αφορά τα πλαίσια που ξεχώρισαν λόγω της υψηλής βαθμολογίας τους, θα πρέπει να παραδεχθούμε πως υπάρχουν περιθώρια βελτίωσης. Για το Spring θα μπορούσαμε να πούμε ότι πρέπει να βελτιώσει την ασφάλεια του ως προς τα κριτήρια Authentication, Authorization, SQL Injection και Session management. Περισσότερης βελτίωσης παρόλα αυτά χρήσει το Jmix συγκεντρώνοντας σημαντικά χαμηλότερη βαθμολογία σε σχέση με το Spring ως προς τα κριτήρια Authentication, Session management και Cryptography. Στα υπόλοιπα κριτήρια, εκτός του Authorization με βάση το οποίο κρίθηκε το πιο ασφαλές, παρατηρούμε ότι υπάρχει επίσης περιθώριο για βελτίωση.

4. Ανάλυση Τρωτοτήτων

Στο κεφάλαιο αυτό πραγματοποιούμε στατική ανάλυση του κώδικα των πλαισίων ανάπτυξης εφαρμογών ιστού με σκοπό να εντοπίσουμε σχετικές τρωτότητες ασφαλείας. Για να το επιτύχουμε αυτό θα χρησιμοποιήσουμε ορισμένα εργαλεία, τα οποία ονομάζονται σαρωτές τρωτοτήτων (vulnerability scanners). Ένας σαρωτής τρωτοτήτων είναι ένα λογισμικό ή εργαλείο ασφαλείας που χρησιμοποιείται για την ανίχνευση ευπαθειών σε ένα σύστημα, δίκτυο, ή εφαρμογή. Τα εργαλεία αυτού του είδους επιτρέπουν στους διαχειριστές ασφαλείας ή τους ειδικούς σε κυβερνοασφάλεια να ανιχνεύουν πιθανές ευπάθειες ή αδυναμίες στον κώδικα, τις ρυθμίσεις ή τις διαμορφώσεις του συστήματος τους, που θα μπορούσαν να εκμεταλλευτούν κακόβουλοι χρήστες για να προκαλέσουν προβλήματα ασφαλείας (π.χ. διείσδυση στο σύστημα, αποκάλυψη ευαίσθητων δεδομένων κα.).

Οι σαρωτές τρωτοτήτων λειτουργούν αυτοματοποιημένα, εξετάζοντας τον στόχο τους για γνωστές ευπάθειες, όπως λάθη στο λογισμικό, ανοιχτές πόρτες, και αδύναμες κωδικοποιήσεις. Χρησιμοποιούν τεχνικές είτε στατικής είτε δυναμικής ανάλυσης ώστε να εντοπίσουν τις ευπάθειες. Έπειτα, παράγουν αναφορές εκμεταλλευόμενοι μια βάση δεδομένων τρωτοτήτων ώστε να προσδιορίσουν όλη την απαραίτητη πληροφορία για την κάθε εντοπισμένη ευπάθεια προς αναφορά, συμπεριλαμβανομένου συστάσεων για τον τρόπο επιδιόρθωσης της.

Προκειμένου να επιλέξουμε τον σωστό και πιο αποδοτικό σαρωτή τρωτοτήτων αρχικά εντοπίσαμε τη γλώσσα στην οποία είναι γραμμένο το κάθε πλαίσιο. Για το ζήτημα αυτό, λόγω της εστίασης μας σε πλαίσια ανάπτυξης εφαρμογών ιστού σε Java, τα περισσότερα από τα πλαίσια που επιλέξαμε είναι γραμμένα σε αυτήν την γλώσσα προγραμματισμού εκτός από το JHipster, το οποίο είναι γραμμένο σε πολλές γλώσσες. Παρόλα αυτά το server-side μέρος του JHipster, το οποίο είναι κι αυτό που μας ενδιαφέρει (ως προς την ανάπτυξη των εφαρμογών ιστού) είναι γραμμένο σε Java. Στη συνέχεια κάναμε μια έρευνα στο διαδίκτυο έτσι ώστε να καταλήξουμε στα κατάλληλα εργαλεία ανίχνευσης τρωτοτήτων για τα πλαίσια μας, τα οποία περιγράφονται στην Ενότητα 4.1.

Στις επόμενες ενότητες θα πραγματοποιηθεί μια περιγραφή της διαδικασίας που ακολουθήθηκε για την ανάλυση των τρωτοτήτων. Στην ενότητα 4.1 περιγράφονται τα εργαλεία ανίχνευσης που χρησιμοποιήθηκαν για την συγκέντρωση των τρωτοτήτων. Στην ενότητα 4.2 πραγματοποιείται μια ανάλυση των τρωτοτήτων που ανιχνεύθηκαν στο προηγούμενο κεφάλαιο (4.1) για κάθε εργαλείο. Τέλος, στην ενότητα 4.3 παρουσιάζονται κάποια συμπεράσματα σχετικά με την ανάλυση που πραγματοποιήθηκε.

4.1 Επιλεγμένα Εργαλεία Ανίχνευσης Τρωτοτήτων

Στις επόμενες ενότητες θα αναλύσουμε τα εργαλεία ανίχνευσης τρωτοτήτων που επιλέξαμε στα πλαίσια αυτής της εργασίας. Τα εργαλεία αυτά είναι το Codacy, το οποίο αναλύεται στην ενότητα 4.1.1, το Snyk, το οποίο αναλύεται στην ενότητα 4.1.2, και το Sonarqube που αναλύεται στην ενότητα 4.1.3. Θα αναφερθούμε σε βασικά χαρακτηριστικά των εργαλείων αυτών, που αφορούν την ασφάλεια, τα οποία μας οδήγησαν και στο να τα επιλέξουμε.

4.1.1 Codacy

Είναι ένα αυτοματοποιημένο εργαλείο ανάλυσης κώδικα που βοηθά τους προγραμματιστές και τις ομάδες ανάπτυξης να εντοπίζουν και να διορθώνουν ζητήματα ασφάλειας, σφάλματα καθώς και διάφορα άλλα είδη θεμάτων ποιότητας στον κώδικα τους. Ειδικότερα, παρέχει αυτοματοποιημένες αναφορές σχετικά με πιθανές ευπάθειες και ζητήματα ποιότητας κώδικα ενώ συνιστά σχετικές διορθώσεις και βελτιώσεις.

Τα βασικά χαρακτηριστικά του σαρωτή τρωτοτήτων Codacy είναι τα εξής:

- **Στατική Ανάλυση Κώδικα (Static Code Analysis):**
  Το Codacy προσφέρει στατική ανάλυση του κώδικα για τον εντοπισμό σφαλμάτων, προειδοποιήσεων και άλλων προβλημάτων ποιότητας.

- **Ανίχνευση ζητημάτων ασφαλείας:**
  Το Codacy αναλύει τον κώδικα για ευπάθειες, αδυναμίες και ζητήματα ασφαλείας, συμπεριλαμβανομένων των ευπαθειών σε επιθέσεις, όπως η έγχυση SQL και το cross-site scripting (XSS).

- **Ανίχνευση ζητημάτων ποιότητας (Code quality metrics):**
  Το εργαλείο αναλύει επίσης τον κώδικα για θέματα ποιότητας, ασφάλειας και σημασιολογικά/λογικά σφάλματα, όπως η κακή διαχείριση μνήμης.
- **Συνεργασία και ενσωμάτωση με εργαλεία ανάπτυξης (Integration with development tools):**
  Το Codacy επιτρέπει στις ομάδες ανάπτυξης να συνεργάζονται για την επίλυση προβλημάτων και τη βελτίωση του κώδικα. Μπορεί επίσης να ενσωματωθεί σε τυποποιημένες εργαλειοθήκες ανάπτυξης,ή repository services όπως το GitHub.

- **Προσαρμογή (Adaptation / Adjustment):**
  Οι χρήστες μπορούν να προσαρμόσουν τους κανόνες και τις ρυθμίσεις του Codacy ώστε να ανταποκρίνονται στις ανάγκες και τις απαιτήσεις του έργου τους.

- **Συμμόρφωση με Πρότυπα Κώδικα:**
  Επιτρέπει τον έλεγχο και τη συμμόρφωση του κώδικα με καθορισμένα πρότυπα και κανόνες προγραμματισμού.

- **Συνεχής Έλεγχος (Continuous Integration):**
  Υποστηρίζει την ενσωμάτωση με διαδικασίες CI/CD για αυτόματους ελέγχους κώδικα κατά τη διάρκεια της ανάπτυξης.

- **Παρακολούθηση Εξέλιξης (Code Churn Metrics):**

Παρέχει μετρήσεις/μετρικές σχετικά με την εξέλιξη του κώδικα, τον ρυθμό αλλαγών, και άλλα.

Το Codacy βελτιώνει την ποιότητα του κώδικα, οδηγεί σε μείωση των τρωτών σημείων ασφαλείας και επιταχύνει τον κύκλο ανάπτυξης λογισμικού. Είναι ένα χρήσιμο εργαλείο για τις ομάδες ανάπτυξης που θέλουν να διασφαλίσουν ότι ο κώδικάς τους είναι ασφαλής και υψηλής ποιότητας. Συνεργάζεται με διάφορα εργαλεία και περιβάλλοντα ανάπτυξης, επιτρέποντας την ομαλή ενσωμάτωση στις διαδικασίες ανάπτυξης λογισμικού.

Η διαδικασία εκμετάλλευσης του Codacy περιλαμβάνει τα παρακάτω βήματα:

- **Εγκατάσταση:** Αρχικά, μετά την εγκατάσταση[2] ακολουθεί η ρύθμιση του Codacy, η οποία περιλαμβάνει τη σύνδεση του Codacy με τον κώδικα και την παραμετροποίηση των κανόνων αξιολόγησης που εφαρμόζονται.

- **Ανάλυση Κώδικα:** Μόλις είναι εγκατεστημένο και συνδεδεμένο, το Codacy αναλύει τον κώδικα για την ανίχνευση πιθανών τρωτοτήτων και ευπαθειών.

- **Αναφορά Τρωτοτήτων:** Το Codacy παρέχει αναφορές για τις τρωτότητες που εντοπίζει στον κώδικα. Αυτές οι αναφορές συνήθως περιλαμβάνουν πληροφορίες σχετικά με τη φύση του προβλήματος, τον τρόπο επίλυσής του και τον κώδικα που εμπλέκεται.

Με βάση τις αναφορές που προσφέρει το Codacy, οι χρήστες μπορούν να διορθώσουν τα θέματα που εντοπίζονται και να βελτιώσουν την ποιότητα του κώδικα.

Η χρήση του Codacy είναι καλή ιδέα καθώς προσφέρει πολλά οφέλη όπως τα παρακάτω:

- **Αυτόματη Αξιολόγηση:** Οι χρήστες μπορούν να εξοικονομήσουν χρόνο χάρη στην αυτοματοποιημένη λειτουργία του Codacy.

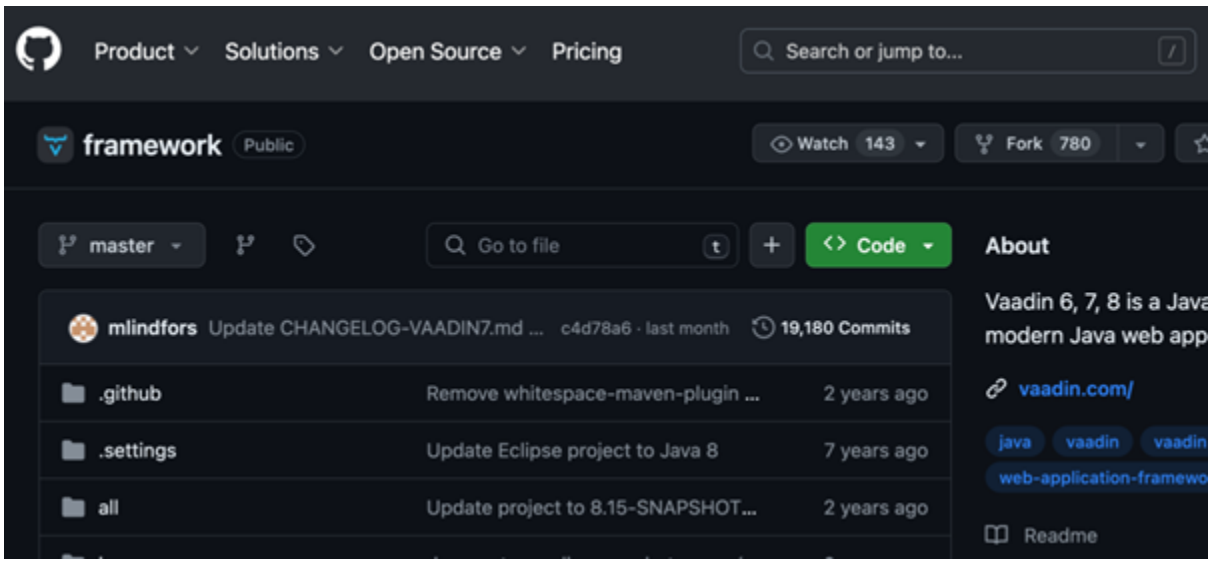
- **Βελτίωση Ποιότητας:** Η αξιολόγηση του κώδικα ενισχύει στον εντοπισμό πιθανών τρωτοτήτων και προβλημάτων του κώδικα, έτσι βελτιώνεται η συνολική ποιότητα του.

- **Αποτελεσματική συνεργασία:** Το Codacy παρέχει ένα συνεργατικό περιβάλλον όπου η ομάδα ανάπτυξης βοηθάει στην ανίχνευση και επίλυση προβλημάτων.

Γενικότερα, το Codacy αποτελεί ένα ισχυρό εργαλείο που συνεισφέρει στην ανάπτυξη υψηλής ποιότητας λογισμικού μέσω της αξιολόγησης του κώδικα και της βελτίωσης ανάπτυξης του.

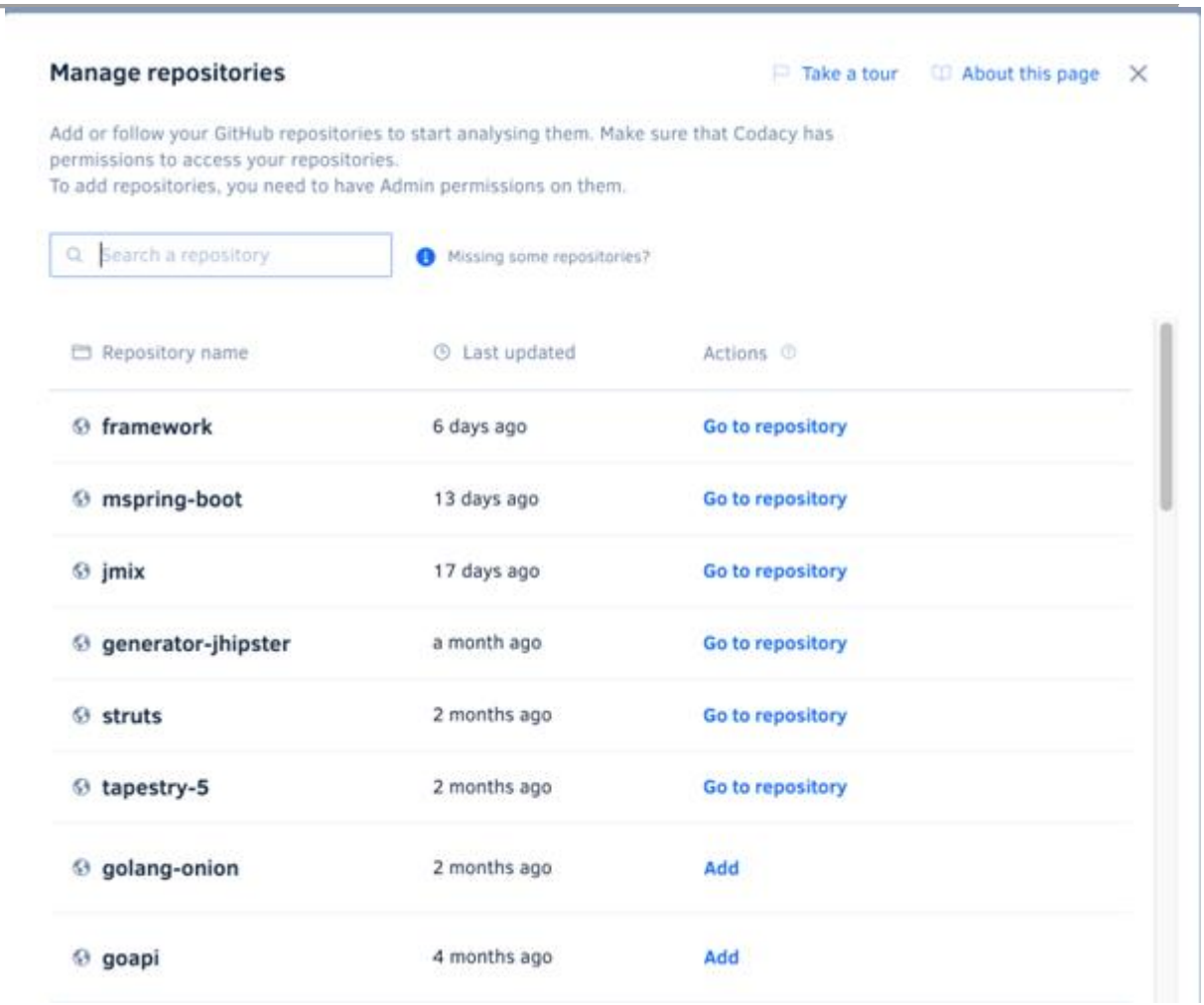
[2] https://docs.codacy.com/chart/

Έχοντας επιλέξει τα αποθετήρια που θέλουμε να ελέγξουμε, μπορούμε να επισκεφτούμε το κάθε ένα ξεχωριστά και να επιλέξουμε να κάνουμε fork το έργο στον λογαριασμό μας. Το fork δημιουργεί ένα πιστό αντίγραφο του έργου στον δικό μας λογαριασμό, συνεπώς μπορούμε να το χρησιμοποιήσουμε για τους μελλοντικούς ελέγχους μας.



Εικόνα 1 Στιγμιότυπο από github λογαριασμό

Για να προσθέσουμε ένα καινούριο αποθετήριο στο εργαλείο Codacy πρέπει να συνδέσουμε τον Git λογαριασμό μας, με επιλογή ενός εκ των Github, Gitlab και BitBucket. Επιλέγοντας τον Github λογαριασμό, μπορούμε να δούμε τα αποθετήρια που έχουμε στον λογαριασμό μας και τα αποθετήρια που έχουμε προσθέσει.



Εικόνα 2 Στιγμιότυπο από codacy που εμφανίζονται τα αποθετήριά μας

Στην συνέχεια, στο βασικό board του Codacy εμφανίζονται όλα τα αποθετήρια που έχουμε προσθέσει με αναλυτικές πληροφορίες για το ποσοστό προβλημάτων που έχει το καθένα. Η πρώτη στήλη εμφανίζει το όνομα του αντίστοιχου αποθετηρίου, ενώ η δεύτερη στήλη εμφανίζει την τελική βαθμολογία του κάθε πλαισίου, βάσει του αριθμού και της κρισιμότητας των τρωτοτήτων που ανακαλύφθηκαν. Η τρίτη στήλη, "Issues", παρουσιάζει το ποσοστό ζητημάτων σε σχέση με την αναμενόμενη βασική γραμμή. Τα όρια για ζητήματα καθορίζονται στις ρυθμίσεις ποιότητας κάθε αποθετηρίου. Η τέταρτη στήλη, "Complexity", αφορά το ποσοστό σύνθετων αρχείων στο αποθετήριο. Τα όρια πολυπλοκότητας καθορίζονται στις ρυθμίσεις ποιότητας κάθε αποθετηρίου. Τέλος, η στήλη "Duplication", παρουσιάζει για κάθε αποθετήριο το ποσοστό διπλότυπων αρχείων. Τα όρια για την αντιγραφή καθορίζονται στις ρυθμίσεις ποιότητας κάθε αποθετηρίου.

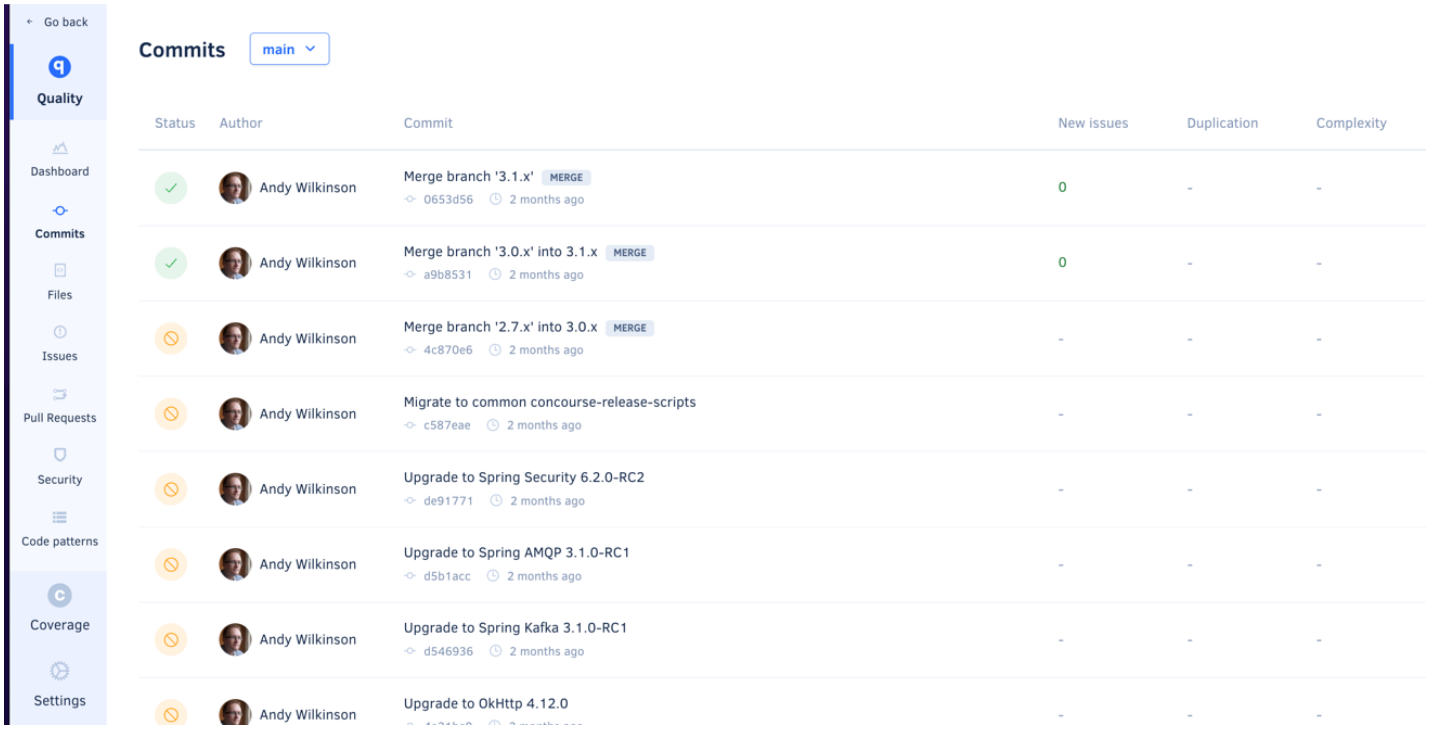
Εικόνα 3 Στιγμιότυπο από codacy όπου εμφανίζονται τα αποθετήρια μας που σαρώθηκαν και το ποσοστό προβλημάτων που εντοπίστηκαν.

Όπως φαίνεται και στην Εικόνα 3, μετά την φόρτωση του κώδικα των πλαισίων από το github, εμφανίζονται πλήρως οι πληροφορίες του πλαισίου, όπως για παράδειγμα όλα τα commits που έχουν γίνει από την έναρξη της δημιουργίας του.

4.1.2 Snyk

Είναι ένα εργαλείο ασφάλειας και ανίχνευσης ευπαθειών που έχει σχεδιαστεί για τον εντοπισμό και την αντιμετώπιση ζητημάτων ασφάλειας στον κώδικα εφαρμογών. Αναλύοντας τον κώδικα, τις εξαρτήσεις και τις βιβλιοθήκες, οι προγραμματιστές μπορούν να εντοπίσουν και να διορθώσουν ευάλωτες τρωτότητες που μπορεί να οδηγήσουν σε πιθανές ευπάθειες.

Είναι γνωστό για τον έλεγχο ανοιχτών πηγών (open source) και εξαρτήσεων των έργων λογισμικού, προκειμένου να εντοπίζει πιθανές αδυναμίες ασφάλειας και να παρέχει συστάσεις για επιδιορθώσεις.



Εικόνα 4 Στιγμιότυπο από codacy που εμφανίζονται και τα commits του κώδικα αποθετηρίου

Τα κύρια χαρακτηριστικά του Snyk Vulnerability Scanner είναι τα εξής:

- **Ανίχνευση τρωτών σημείων σε βιβλιοθήκες:**
  Το Snyk αναλύει εξαρτήσεις κώδικα και εντοπίζει ευάλωτες εκδόσεις βιβλιοθηκών που ενδέχεται να περιέχουν γνωστές ευπάθειες.

- **Ευπάθειες:**
  Το Snyk ενσωματώνει την ασφάλεια στη διαδικασία ανάπτυξης χάρη στη συμβατότητά του με πολλά εργαλεία ανάπτυξης και την ενσωμάτωση του στον κύκλο ζωής ανάπτυξης λογισμικού. Αναλύει τις εξαρτήσεις του λογισμικού και εντοπίζει πιθανές ευπαθείς εκδόσεις ή εξαρτήσεις που περιλαμβάνουν γνωστά προβλήματα ασφαλείας.

- **Αυτόματη αναφορά:**
  Παρέχει λεπτομερείς αναφορές σχετικά με τις ευάλωτες εκδόσεις και προτείνει συγκεκριμένα βήματα για τη διόρθωσή τους.

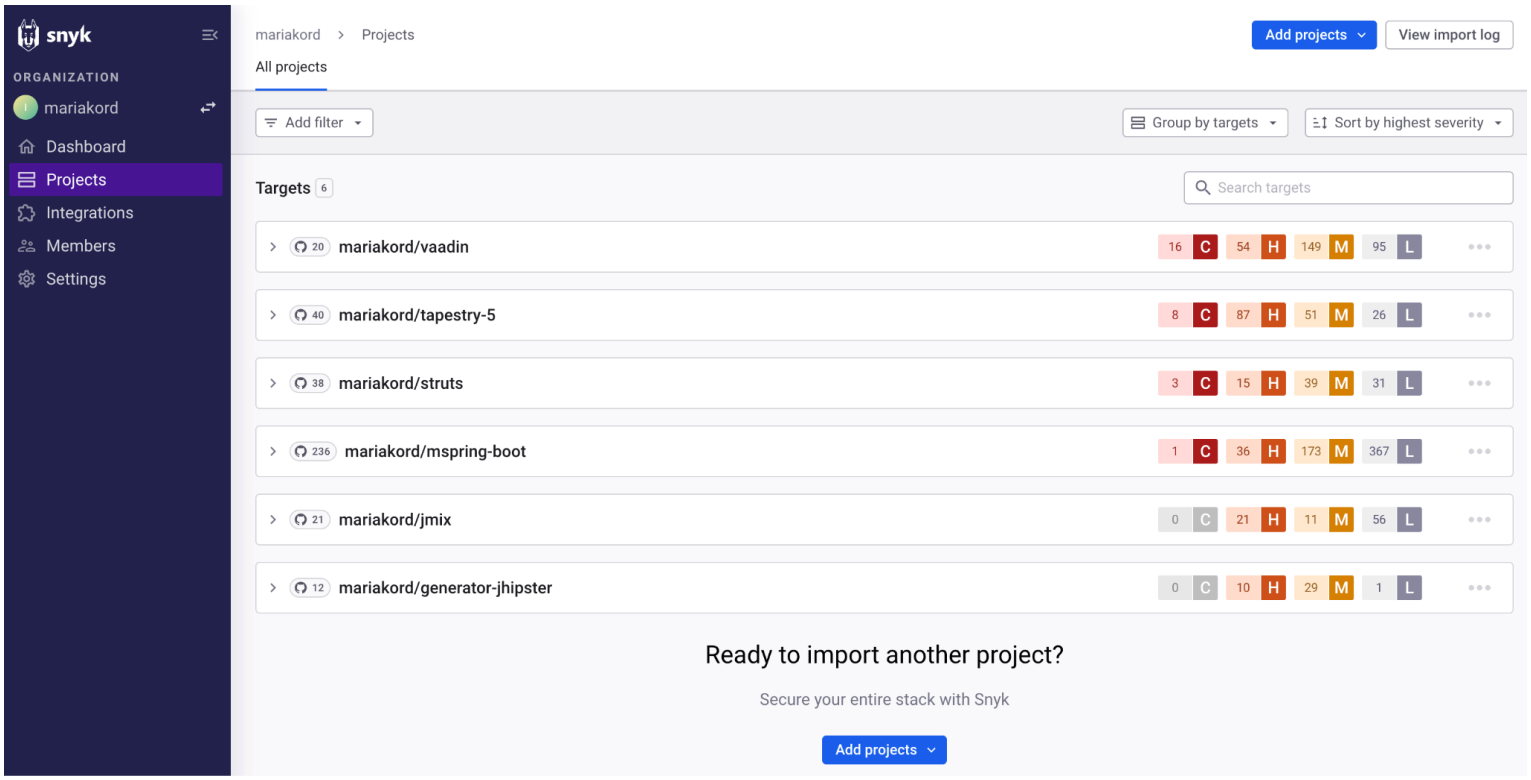
- **Ενισχυμένη Ασφάλεια:**
  Το Snyk συμβάλλει στην ενίσχυση της ασφάλειας των εφαρμογών προστατεύοντας από γνωστές ευπάθειες που ενδέχεται να είναι ευάλωτες σε επιθέσεις.

- **Ενσωμάτωση με CI/CD:**
  Το Snyk μπορεί να ενσωματωθεί σε διαδικασίες CI/CD, επιτρέποντας την αυτόματη εκτέλεση της ανάλυσης ασφαλείας κατά τη διάρκεια της ανάπτυξης.

- **Παρακολούθηση Εξαρτήσεων:** Επιτρέπει στις ομάδες ανάπτυξης να παρακολουθούν την εξέλιξη των εξαρτήσεών τους (λογισμικά/ βιβλιοθήκες) και των πιθανών ευπαθειών.

Συνολικά, το Snyk είναι ένα πολύ χρήσιμο εργαλείο για την ασφάλεια του λογισμικού και βοηθά τις ομάδες ανάπτυξης να παρακολουθούν και να διορθώνουν θέματα ασφάλειας κατά τη διάρκεια της ανάπτυξης και διαχείρισης εφαρμογών.

Στην παρακάτω εικόνα βλέπουμε τα Projects που έχουμε σαν Targets τα οποία περιλαμβάνουν τα framework που μας απασχολούν όπως το Vaadin, το Jmix, το Tapestry κτλ. Στα δεξιά του Framework εμφανίζονται τα Critical, High, Medium, Low ζητήματα καθώς και η ποσότητα τους.



Εικόνα 5 Στιγμιότυπο από snyk που εμφανίζονται τα αποθετήρια μας

4.1.3 Sonarqube[3]

Το SonarQube vulnerability scanner είναι ένα εργαλείο ασφαλείας λογισμικού που επικεντρώνεται στον έλεγχο του κώδικα για εντοπισμό πιθανών ευπαθειών και προβλημάτων ασφάλειας. Με ανάλυση του στατικού κώδικα, εξετάζει τις δομές του προγράμματος για να αναγνωρίσει τυχόν προβλήματα.

Τα κυριότερα χαρακτηριστικά του ως vulnerability scanner περιλαμβάνουν:

Ανίχνευση Ευπαθειών: Το SonarQube ελέγχει τον κώδικα για να εντοπίσει ευπάθειες.

Παροχή Αναφορών Ασφαλείας: Δημιουργεί αναφορές που περιέχουν λεπτομέρειες για τις ευπάθειες που εντοπίζονται, με πληροφορίες για τη σοβαρότητά τους και προτεινόμενες λύσεις.

Συμμόρφωση με Πρότυπα Ασφάλειας: Το SonarQube ελέγχει τον κώδικα έναντι γνωστών προτύπων ασφάλειας όπως το OWASP Top 10 για να εξασφαλίσει τη συμμόρφωση με βασικές αρχές ασφάλειας.

Ενσωμάτωση στη Διαδικασία Ανάπτυξης: Είναι σχεδιασμένο να ενσωματωθεί στη διαδικασία ανάπτυξης λογισμικού, επιτρέποντας στις ομάδες ανάπτυξης να αντιμετωπίζουν ευπάθειες ασφαλείας κατά τη διάρκεια του κύκλου ζωής του λογισμικού.

Οι λόγοι για τους οποίους το επιλέξαμε για την ανάλυση των πλαισίων είναι οι δυνατότητες που προσφέρει για:

Βελτίωση της Ποιότητας του Κώδικα: Το SonarQube βοηθά στην ανίχνευση πιθανών προβλημάτων στον κώδικα, βοηθώντας έτσι στη βελτίωση της ποιότητας του λογισμικού.

Ενοποιημένη Πλατφόρμα: Το SonarQube παρέχει ένα ενιαίο σημείο αναφοράς για την ανάλυση και τη διαχείριση της ποιότητας του κώδικα, επιτρέποντας στις ομάδες ανάπτυξης να εργαστούν αποτελεσματικά.

Αυτοματοποίηση: Το SonarQube προσφέρει αυτοματοποιημένες αναλύσεις και αναφορές, μειώνοντας τον χρόνο και την προσπάθεια που απαιτούνται για την εξέταση του κώδικα χειροκίνητα.

[3] https://docs.sonarsource.com/sonarqube/latest/try-out-sonarqube/

4.2 Ανάλυση εντοπισμένων Τρωτοτήτων

Παρακάτω θα προχωρήσουμε σε μια ανάλυση των εντοπισμένων τρωτοτήτων. Η ανάλυση αυτή θα χωριστεί σε 2 ενότητες. Η πρώτη (4.2.1 Πίνακες Τρωτοτήτων ανά πλαίσιο) θα περιέχει μια αναπαράσταση των τρωτοτήτων κάθε πλαισίου με τη μορφή πίνακα. Κάθε στήλη θα αντιστοιχεί σε διαφορετικό επίπεδο κρισιμότητας, με τιμές επιπέδου, υψηλό, μεσαίο και χαμηλό επίπεδο. Η αναπαράσταση κάθε επιπέδου θα γίνεται με φθίνουσα σειρά. Η δεύτερη ενότητα θα περιέχει μια ανάλυση των τρωτοτήτων που ανιχνεύθηκαν για κάθε πλαίσιο ξεχωριστά ανάλογα με το εργαλείο που χρησιμοποιήθηκε. Για κάθε τρωτότητα θα υπάρχει ανάλυση των επιπτώσεων που μπορεί να προκαλέσει και ένας τρόπος αντιμετώπισης.

4.2.1 Πίνακες Τρωτοτήτων ανά πλαίσιο

Στην παρακάτω ενότητα παρουσιάζονται οι τρωτότητες κάθε πλαισίου με τη μορφή πίνακα.

4.2.1.1 Τρωτότητες Vaadin

Με φθίνουσα σειρά κρισιμότητας.

υψηλού επιπέδου	**μεσαίου επιπέδου**	**χαμηλού επιπέδου**
Improper Input Validation	Subprocess Call - Check For execution of untrusted input	Session Encryption Weakness
Authorization Bypass	Requests call without timeout	Signature Rejection
Remote Code Execution - RCE	Misleading Tokens	URL Parameter Handling
Arbitrary Code Execution	Error Disclosure	Vulnerable Data Input Controllers
-	Unencrypted Data Storage	Hardcoded Secrets

Πίνακας 9 Τρωτότητες Vaadin

4.2.1.2 Τρωτότητες SpringBoot

Με φθίνουσα σειρά κρισιμότητας.

υψηλού επιπέδου	**μεσαίου επιπέδου**	**χαμηλού επιπέδου**
XSS Injection	Password Storage Vulnerabilities	Cross-Site Scripting
External Entity (XXE) Injection	Insecure Direct Object References	Cross-Site Request Forgery
Resource Expansion	Confused Deputy Vulnerabilities	Insecure Image Handling
Command Execution	Information Disclosure Vulnerabilities	External Resource Connection Vulnerabilities
Internal Network Disclosure	Unvalidated Input	Access Control Vulnerabilities

Πίνακας 10 Τρωτότητες Spring-Boot

4.2.1.3 Τρωτότητες Tapestry

Με φθίνουσα σειρα κρισιμότητας.

υψηλού επιπέδου	**μεσαίου επιπέδου**	**χαμηλού επιπέδου**
Encrypted Data Management Vulnerabilities	Shell Command Injection	Data Storage Vulnerabilities
Remote Code Execution	Prototype Pollution	Access Control Vulnerabilities
Arbitrary Code Execution	Sensitive States	Input Data Analysis Vulnerabilities
XML External Entity (XXE) Injection	Software Exposures	Event Vulnerabilities
Deserialization of Untrusted Data	Error Handling	Exception Handling

Πίνακας 11 Τρωτότητες Tapestry

4.2.1.4 Τρωτότητες Struts

Με φθίνουσα σειρα κρισιμότητας.

υψηλού επιπέδου	**μεσαίου επιπέδου**	**χαμηλού επιπέδου**
Command injection	Input Validation	Password Recovery Weaknesses
Remote Code Execution	Authentication Weaknesses	Text Analysis Weaknesses
Cross-Site Scripting	Privacy Protection Weaknesses	Data Validation
Deserialization Weaknesses	SQL injection	Clickjacking Weaknesses
Insecure Access Control	Executable File Control Weaknesses	Content Spoofing

Πίνακας 12 Τρωτότητες Struts

4.2.1.5 Τρωτότητες JHipster

Με φθίνουσα σειρα κρισιμότητας.

υψηλού επιπέδου	**μεσαίου επιπέδου**	**χαμηλού επιπέδου**
-	Cross-Site Scripting (XSS)	Poor Code Security Practices
-	File Path and Directory Security Vulnerabilities	Tool and Library Management Vulnerabilities
-	SQL Injection (SQLi)	Application Storage Management
-	Centralized Control Check Vulnerabilities	Input/Output Dependency Vulnerabilities
-	Input Data Validation Vulnerabilities	Software Development and Management Vulnerabilities

Πίνακας 13 Τρωτότητες JHipster

4.2.1.6 Τρωτότητες JMix

Με φθίνουσα σειρα κρισιμότητας.

υψηλού επιπέδου	**μεσαίου επιπέδου**	**χαμηλού επιπέδου**
IMAP (Internet Message Access Protocol) Vulnerabilities	Input Validation Vulnerabilities	Buffer Overflow
Dependency Management Vulnerabilities	Fault Control	Malware Recognition Vulnerabilities
-	Identity Management Vulnerabilities	Denial of Service
-	Communication Procedure Security Vulnerabilities	Encryption Security Vulnerabilities
-	File Processing Security	Insufficient Connection Security

Πίνακας 14 Τρωτότητες JMix

4.2.2 Ανάλυση Τρωτοτήτων ανά πλαίσιο

Στην παρακάτω ενότητα πραγματοποιείται μια ανάλυση των τρωτοτήτων για κάθε πλαίσιο με τη χρήση των τριών εργαλείων ανίχνευσης. Η πρώτη ανάλυση είναι για το εργαλείο Codacy (4.2.2.1), η δεύτερη για το Snyk (4.2.2.2) και η τρίτη για το Sonarqube (4.2.2.3).

4.2.2.1 Ανάλυση με χρήση του Codacy

Στην παρακάτω ενότητα γίνεται ανάλυση των τρωτοτήτων κάθε πλαίσιο με χρήση του εργαλείου Codacy.

4.2.2.1.1 Ανάλυση τρωτοτήτων για το Vaadin

Το πρώτο framework που εξετάσαμε ήταν το Vaadin. Στην παρακάτω εικόνα εμφανίζονται τα αποτελέσματα της ανάλυσης.

Εικόνα 6 Τρωτότητες του Vaadin από το codacy

Στην παρούσα ανάλυση εξετάζουμε τα θέματα ασφαλείας (δηλ. που ανήκουν στην κατηγορία Security). Παρατηρούμε ότι το Vaadin εμφάνισε 39 θέματα ασφάλειας. Η κρισιμότητά τους χαρακτηρίζεται μεσαίου αλλά και χαμηλού επιπέδου. Πιο συγκεκριμένα από τα 39, τα 29 ήταν μεσαίας κρισιμότητας ζητήματα και τα 10 χαμηλής. Θα παρουσιάσουμε και θα εξηγήσουμε μερικά από τα μεσαίας κρισιμότητας ζητήματα, τα οποία είχαν μεγαλύτερο βαθμό κρισιμότητας σε σχέση με τα υπόλοιπα.

➔ **Subprocess Call - Check For execution of untrusted input** (Κλήση υποδιεργασίας - Έλεγχος για εκτέλεση μη αξιόπιστης εισόδου)

Εικόνα 7 Στιγμιότυπο τρωτότητας

Ανάλυση Τρωτότητας

Η συνάρτηση check_output() ανήκει στο πακέτο subprocess της γλώσσας προγραμματισμού Python[4] και είναι παρόμοια με τη συνάρτηση run(). Είναι ένα από τα βασικά εργαλεία που παρέχει το πλαίσιο της Python για τη διαχείριση εξωτερικών διεργασιών και επικοινωνία με αυτές και πρόκειται για βασικό κώδικα του πλαισίου. Αυτή η συνάρτηση χρησιμοποιείται για να εκτελέσει μια εντολή στο κέλυφος του συστήματος και να αποκτήσει την έξοδο της εντολής**.** Αναφέρεται στον έλεγχο της εξόδου της κλήσης υποδιεργασίας για να διασφαλίσει ότι η εκτέλεση ολοκληρώθηκε με επιτυχία. Η χρήση της check_output με μη αξιόπιστη είσοδο, όπως η είσοδος που παρέχεται από τον χρήστη και συγκεκριμένα η χρήση της παραμέτρου shell είναι ιδιαίτερα επικίνδυνη εάν δεν γίνεται καλά έλεγχος της εισόδου, καθώς μπορεί να οδηγήσει σε εκτέλεση ανεπιθύμητου κώδικα (injection attacks, shell injection).

[4] Παρά το γεγονός πως το Vaadin είναι στο μεγαλύτερο μέρος του γραμμένο σε Java, στο project εμφανίζονται κάποια αρχεία script γραμμένα στην γλώσσα Python. Αν και τα αρχεία αυτά δεν αποτελούν μέρος της λειτουργίας του Vaadin, χρησιμοποιούνται από το επίσημο αποθετήριο για άλλες σημαντικές εργασίες. Πιο συγκεκριμένα, τα script αυτά χρησιμοποιούνται για την αυτοματοποίηση εργασιών, όπως η κατασκευή και επικύρωση διαφόρων λειτουργιών, είναι υπεύθυνα για τον χειρισμό διαφόρων αρχείων παραμετροποίησης και εξαρτήσεων, ενώ δημιουργούν δοχεία Docker και τα διαχειρίζονται για την εκτέλεση της Vaadin εφαρμογής.

Η συνάρτηση check_output παίρνει τα ίδια ορίσματα με τη run(), συμπεριλαμβανομένων των args που καθορίζουν την εντολή που θα εκτελεστεί και άλλα προαιρετικά ορίσματα, όπως τα stdin, stderr, shell, universal_newlines, timeout, text, cwd και env. Αναλύουμε τα περισσότερα από αυτά παρακάτω:

- args: Το πρόγραμμα ή η εντολή που εκτελείται, ως λίστα ή ως συμβολοσειρά.
- stdin, stderr: Προαιρετικά ορίσματα που καθορίζουν τα αντικείμενα εισόδου και εξόδου (σφάλματος) για την εντολή.
- shell: Εάν είναι True (αληθής), τότε η εντολή εκτελείται μέσα από ένα κέλυφος.
- universal_newlines: Εάν είναι True, η έξοδος επιστρέφεται ως συμβολοσειρά (κείμενο).
- timeout: Χρονικό όριο εκτέλεσης της εντολής σε δευτερόλεπτα.
- text: Ίδιο με το universal_newlines, αλλά προτείνεται για νεότερες εκδόσεις Python.
- cwd (current working directory): καθορίζει σε ποιον κατάλογο θα εκτελεστεί η εντολή. Εάν δεν οριστεί, η εντολή εκτελείται στον τρέχοντα κατάλογο εργασίας του προγράμματος που το καλεί.
- env (environment variables): επιτρέπει τον καθορισμό περιβαλλοντικών μεταβλητών για την εκτέλεση της εντολής. Αν δεν οριστεί, η εντολή εκτελείται με το τρέχον περιβάλλον εκτέλεσης του προγράμματος.

Η συνάρτηση εκτελεί τη δοθείσα εντολή και επιστρέφει την έξοδο της ως byte string ή κείμενο, ανάλογα με τη ρύθμιση του universal_newlines ή text. Αν η εντολή αποτύχει, εγείρεται μια εξαίρεση τύπου CalledProcessError.

Απειλές κλήσης των συναρτήσεων:

1. Επιθέσεις Shell Command Injection**:** Εάν το shell=True, αυτό σημαίνει πως η συνάρτηση χρησιμοποιείται χωρίς προσεκτικό χειρισμό των ορισμάτων, οπότε μπορεί να επιτρέψει επιθέσεις τύπου shell command injection (έκχυσης εντολών κελύφους).
2. Path Traversal: Ο κώδικας αναζητά ένα αρχείο .cmd με βάση την τιμή της μεταβλητής εντολής. Εάν η μεταβλητή εντολής βρίσκεται υπό τον έλεγχο ενός επιτιθέμενου και δεν υπάρχει περιορισμός στη διαδρομή των αρχείων, δηλαδή επιτρέπεται η διέλευση καταλόγου, τότε μπορεί ο επιτιθέμενος να έχει μη εξουσιοδοτημένη πρόσβαση σε αρχεία και καταλόγους που δεν πρέπει να είναι προσβάσιμοι από τρίτους. Αυτό θα μπορούσε να του επιτρέψει να προβάλει ή να τροποποιήσει ευαίσθητα αρχεία. Η επιτυχία του Path Traversal εξαρτάται από τον γενικό σχεδιασμό και τις πρακτικές ασφαλείας του συστήματος. Σε περιβάλλοντα με αυξημένα προνόμια, όπως σε διακομιστές, ένας επιτυχημένος Path Traversal μπορεί να έχει πιο σοβαρές συνέπειες, ακόμη και να επιτρέψει στον επιτιθέμενο να εκτελέσει κακόβουλο κώδικα στο σύστημα.
3. Unauthorized Access: αναφέρεται στον τρόπο πρόσβασης σε συστήματα, πόρους ή δεδομένα χωρίς την επίσημη ή έγκυρη άδεια ή εξουσιοδότηση. Εάν η υποδιεργασία (που δημιουργείται από την κλήση ενός εκτελέσιμου αρχείου ή μια εντολής) εκτελείται με αυξημένα προνόμια, θα μπορούσε να οδηγήσει σε μη εξουσιοδοτημένη πρόσβαση στους πόρους του συστήματος, επιτρέποντας ενδεχομένως σε έναν επιτιθέμενο να εκτελέσει ενέργειες, όπως διαρροή ευαίσθητων δεδομένων, αλλοίωση ή καταστροφή δεδομένων, παρακώλυση της λειτουργίας ενός συστήματος ή χρήση του συστήματος για παράνομες ενέργειες.

Τρόπος αντιμετώπισης

Προτείνεται η χρήση του Java ProcessBuilder αντί για το subprocess καθώς επιτρέπει μεγαλύτερο έλεγχο και ασφάλεια κατά την εκτέλεση εξωτερικών εντολών.

➔ **Requests call without timeout (Κλήση σε REST υπηρεσία χωρίς χρονικό όριο)**

Εικόνα 8 Στιγμιότυπο τρωτότητας

Ανάλυση Τρωτότητας

- requests.post(): Δημιουργεί ένα HTTP POST αίτημα.
- headers={"Content-Type": "Application/json"}: Αυτό το όρισμα καθορίζει τις κεφαλίδες (headers) του αιτήματος. Συγκεκριμένα, θέτει μια κεφαλίδα με το όνομα "Content-Type" και την τιμή "Application/json". Αυτό υποδηλώνει ότι το περιεχόμενο του αιτήματος θα είναι σε μορφή JSON.
- **kwargs: Αυτή είναι μια συντομογραφία για "keyword arguments" και χρησιμοποιείται για να περάσετε οποιαδήποτε επιπλέον ορίσματα που υποστηρίζονται από τη συνάρτηση.Τα ορίσματα μπορεί να περιλαμβάνουν το URL, τα δεδομένα (data), τις παραμέτρους (params), την εξουσιοδότηση (auth), το timeout, κ.ά.

Συνοπτικά, αυτό το αίτημα POST εισάγει μια κεφαλίδα "Content-Type" που υποδηλώνει ότι τα δεδομένα που στέλνονται είναι σε μορφή JSON, και στη συνέχεια, περνά οποιαδήποτε άλλα ορίσματα που έχουν καθοριστεί με το **kwargs.

Το πρόβλημα εντοπίζεται στον κύριο κώδικα, εκεί όπου καλείται η συνάρτηση requests.post() χωρίς να ορίζεται χρονικό όριο (timeout), πράγμα που σημαίνει ότι το αίτημα θα μπορούσε ενδεχομένως να περιμένει την ολοκλήρωσή του επ' αόριστον εάν ο διακομιστής δεν ανταποκριθεί ποτέ. Αυτό μπορεί να οδηγήσει σε διάφορα ζητήματα ασφάλειας και απόδοσης, όπως:

1. Denial of Service (DoS) επιθέσεις: Επικίνδυνοι επιτιθέμενοι μπορεί να εκμεταλλευτούν την έλλειψη timeout για να εκτελέσουν επιθέσεις DoS, στέλνοντας αιτήματα που παραμένουν ανοικτά για μεγάλο χρονικό διάστημα, εξαντλώντας τους πόρους του συστήματος.
2. Καθυστέρηση Εφαρμογής: Αν ένα αίτημα HTTP δεν έχει καθορισμένο timeout και ο πόρος που προσπαθείτε να ανακτήσετε είναι μη διαθέσιμος ή καθυστερεί, η εφαρμογή σας μπορεί να εμμένει στην αναμονή για μεγάλο χρονικό διάστημα, προκαλώντας καθυστέρηση στις ανταποκρίσεις.

Τρόπος αντιμετώπισης

Προτείνεται η χρήση του HttpClient της Java, καθώς είναι πιο σύγχρονο και ευέλικτο από το HttpURLConnection.

4.2.2.1.2 Ανάλυση τρωτότητων για το SpringBoot

Για το SpringBoot προέκυψαν 16 ζητήματα ασφαλείας, από τα οποία κανένα δεν ήταν υψηλής σημασίας. Παρακάτω θα αναλύσουμε ένα από τα ζητήματα μεσαίας κρισιμότητας που κρίναμε το πιο σημαντικό και είχε 4 εμφανίσεις.

Εικόνα 9 Τρωτότητες Spring Boot από το codacy

➔ **Generic Object Injection Sink**

Εικόνα 10 Στιγμιότυπο Τρωτότητας

Ανάλυση Τρωτότητας
Σύμφωνα με το Codacy, η παραπάνω γραμμή κώδικα αφορά ευπάθεια τύπου Έγχυσης εντολής (Command injection) και παρουσιάζεται στο συγκεκριμένο σημείο, όπως παρατηρούμε, σε κάποια δομή ελέγχου.

Ενώ το συγκεκριμένο πλαίσιο δεν είναι διαθέσιμο, είναι ζωτικής σημασίας να κατανοήσουμε τη συνολική λειτουργικότητα και τη πιθανή εμπλοκή δεδομένων χρήστη.

- Ποιος είναι ο σκοπός της μεταβλητής eventName;
- Πώς παράγεται ή κατασκευάζεται το eventName;
- Υπάρχει κάποιο δεδομένο χρήστη που θα μπορούσε να επηρεάσει το eventName;

Εάν το eventName επηρεάζεται άμεσα ή έμμεσα από δεδομένα χρήστη χωρίς σωστή επικύρωση, μπορεί να εκμεταλλευτεί για έγχυση εντολών. Ένας επιτιθέμενος θα μπορούσε να δημιουργήσει μια κακόβουλη τιμή για το eventName η οποία, κατά την εκτέλεσή της, θα μπορούσε να οδηγήσει σε μη εξουσιοδοτημένες ενέργειες στο σύστημα.

Τρόπος Αντιμετώπισης

Καλύτερες πρακτικές για την πρόληψη της εισβολής εντολών:

- Επικύρωση όλων των δεδομένων χρήστη: Θα πρέπει να γίνεται χρήση ενσωματωμένων λειτουργιών επικύρωσης ή κανονικές εκφράσεις για να διασφαλιστεί ότι τα δεδομένα συμμορφώνονται με τα αναμενόμενα μοτίβα. Θα πρέπει επίσης να γίνει sanitize τα δεδομένα για να αφαιρεθούν πιθανώς επιβλαβείς χαρακτήρες ή ακολουθίες.
- Χρήση παραμετροποιημένων ερωτημάτων: Όταν εκτελείται αλληλεπίδραση με βάσεις δεδομένων, θα πρέπει να γίνεται χρήση παραμετροποιημένων ερωτημάτων αντί για συνένωση συμβολοσειρών. Αυτό αποτρέπει την εισαγωγή κακόβουλου κώδικα στο ερώτημα.
- Αποφυγή ειδικών χαρακτήρων: Όταν γίνεται χρήση δεδομένων χρήστη σε συμβολοσειρές εντολών, θα πρέπει να γίνεται αποφυγή τυχόν ειδικών χαρακτήρων που θα μπορούσαν να ερμηνευθούν ως μέρος της ίδιας της εντολής. Για παράδειγμα, η χρήση \" για την αποτροπή της ενέργειας των διπλών εισαγωγικών σε Linux ή Windows.
- Εξέταση εναλλακτικών προσεγγίσεων: Εάν είναι δυνατόν, θα πρέπει να γίνει αποφυγή χρήσης δεδομένων χρήστη σε συμβολοσειρές εντολών συνολικά. Θα πρέπει να διερευνηθούν εναλλακτικές προσεγγίσεις που δεν περιλαμβάνουν άμεση εκτέλεση εντολών. Για παράδειγμα, θα μπορούσε να χρησιμοποιηθεί ένα whitelist

επιτρεπόμενων εντολών ή να γίνει παροχή μιας περιορισμένης διασύνδεσης για τους χρήστες να αλληλεπιδράσουν με το σύστημα.

4.2.2.1.3 Ανάλυση τρωτοτήτων για το Tapestry

Για το tapestry η ανίχνευση ευπαθειών εμφάνισε 15 ζητήματα ασφάλειας από τα οποία το 1 μόνο κρίθηκε υψηλής σημασίας, ενώ τα υπόλοιπα 13 χαμηλής.

Εικόνα 11 Τρωτότητες Tapestry από το codacy

The use of java.io.File violates the Enterprise Java Bean specification (Η χρήση του java.io.File παραβιάζει την προδιαγραφή Enterprise Java Bean)

Εικόνα 12 Στιγμιότυπο τρωτότητας

Ανάλυση Τρωτότητας

Το μήνυμα σφάλματος "Η χρήση του java.io.File παραβιάζει την προδιαγραφή Enterprise Java Bean" πιθανότατα σχετίζεται με τη χρήση της κλάσης java.io.File σε ένα πλαίσιο όπου το Enterprise JavaBeans (EJBs) αναμένεται να συμμορφώνεται με ορισμένους περιορισμούς και βέλτιστες πρακτικές.

Σε ένα τυπικό περιβάλλον EJB, υπάρχουν ορισμένοι περιορισμοί που επιβάλλονται από την προδιαγραφή EJB για να διασφαλιστεί ότι τα στοιχεία EJB μπορούν να διαχειρίζονται από τον container (εμπεριέχων) και να εκτελούνται με κατανεμημένο και συναλλακτικό τρόπο. Αυτοί οι περιορισμοί βοηθούν στη διατήρηση της φορητότητας και της επεκτασιμότητας των εφαρμογών EJB.

Η χρήση του java.io.File ενδέχεται να παραβιάζει την προδιαγραφή EJB για διάφορους λόγους:

1. *Προσπέλαση Συστήματος Αρχείων (Accessing the File System)*: Τα EJB συνήθως προορίζονται να εκτελούνται σε ελεγχόμενο και διαχειριζόμενο περιβάλλον, όπως ένας διακομιστής εφαρμογών (application server). Δεν πρέπει να έχουν άμεση πρόσβαση στο σύστημα αρχείων. Αυτό συμβαίνει επειδή τα EJB αναπτύσσονται συχνά με κατανεμημένο τρόπο και η πρόσβαση στο σύστημα αρχείων μπορεί να προκαλέσει προβλήματα φορητότητας και ασφάλειας.

2. *Διαχείριση Δοσοληψιών* (T*ransaction Management)*: Τα EJB βασίζονται στον container που τα φιλοξενεί για τη διαχείριση των συναλλαγών. Η χρήση του java.io.File για αλληλεπίδραση με το σύστημα αρχείων μπορεί να σπάσει την ακεραιότητα των συναλλαγών.

3. Φορητότητα (P*ortability)*: Τα EJB έχουν σχεδιαστεί για να είναι φορητά σε διαφορετικούς EJB containers και διακομιστές εφαρμογών. Η χρήση του java.io.File ενδέχεται να εμποδίζει την φορητότητα, καθώς η πρόσβαση στο σύστημα αρχείων μπορεί να είναι διαφορετική σε διάφορες πλατφόρμες.

Τρόπος αντιμετώπισης

Χρήση Java NIO (java.nio.file) για το χειρισμό αρχείων. Οι κλάσεις αυτές είναι σχεδιασμένες για να προσφέρουν βελτιωμένες λειτουργίες για τον χειρισμό αρχείων σε σχέση με τις παλαιότερες java.io.File και java.io.* κλάσεις.

4.2.2.1.4 Ανάλυση Τρωτοτήτων για το Struts

Εικόνα 13 Τρωτότητες Struts από το codacy

Από την ανάλυση του Struts προέκυψαν 16 ζητήματα ασφάλειας από τα οποία μόνο το 1 αναγνωρίστηκε υψηλής κρισιμότητας ενώ τα υπόλοιπα 15 χαμηλής.

Found RegExp with non literal argument (Βρέθηκε RegExp με μη κυριολεκτικό όρισμα)

Εικόνα 14 Στιγμιότυπο τρωτότητας

Ανάλυση Τρωτότητας

1. RegExp:
   - Η κλάση RegExp στη JavaScript χρησιμοποιείται για τη δημιουργία ενός αντικειμένου που αντιπροσωπεύει μια κανονική έκφραση (regular expression).
2. "^" + ptn + "$":
   - Το ^ στην αρχή της κανονικής έκφρασης σημαίνει ότι η ακολουθία πρέπει να ξεκινάει από την αρχή της συμβολοσειράς.
   - Το $ στο τέλος σημαίνει ότι η ακολουθία πρέπει να τελειώνει στο τέλος της συμβολοσειράς.
   - Το ptn πιθανόν να είναι μια μεταβλητή που περιέχει ένα κομμάτι της κανονικής έκφρασης, αφού προηγουμένως έχει ανατεθεί.
3. Ανάθεση σε μεταβλητή:
   - Η κανονική έκφραση που δημιουργείται από τη σύνθεση των παραπάνω μερών ανατίθεται στη μεταβλητή reg.

Συνολικά, ο κώδικας δημιουργεί μια κανονική έκφραση που ελέγχει αν μια συμβολοσειρά ξεκινάει και τελειώνει με βάση το περιεχόμενο της μεταβλητής ptn.

Το μήνυμα σφάλματος "Found RegExp with non-literal argument" συνήθως υποδεικνύει την προσπάθεια για δημιουργία μιας τυπικής έκφρασης χρησιμοποιώντας μια μη κυριολεκτική συμβολοσειρά ως μοτίβο. Στο JavaScript, όταν δημιουργούμε μια τυπική έκφραση χρησιμοποιώντας τον κατασκευαστή RegExp, το μοτίβο θα πρέπει να περιλαμβάνει μεταβλητή η οποία θα έχει ελεγχθεί πριν τη χρήση της για την αποφυγή προβλημάτων όπως η εκφραστική επίθεση (Regular Expression Denial of Service - ReDoS).

Τρόπος αντιμετώπισης

Σταθερές Τιμές για τις regular expressions όταν είναι δυνατόν, αντί για δυναμικές τιμές που προέρχονται από μεταβλητές.

Χειρισμός Ειδικών Χαρακτήρων: Κατά τη χρήση μεταβλητών στις regular expressions, απαιτείται σωστός χειρισμός των ειδικών χαρακτήρων.

Επιβεβαίωση Εισόδου Χρήστη: οι regular expressions να μην δέχονται ως είσοδο μη αξιόπιστες πηγές, όπως η είσοδος του χρήστη. Επιβεβαίωση εισόδου του χρήστη προτού χρησιμοποιηθεί σε μια regular expression.

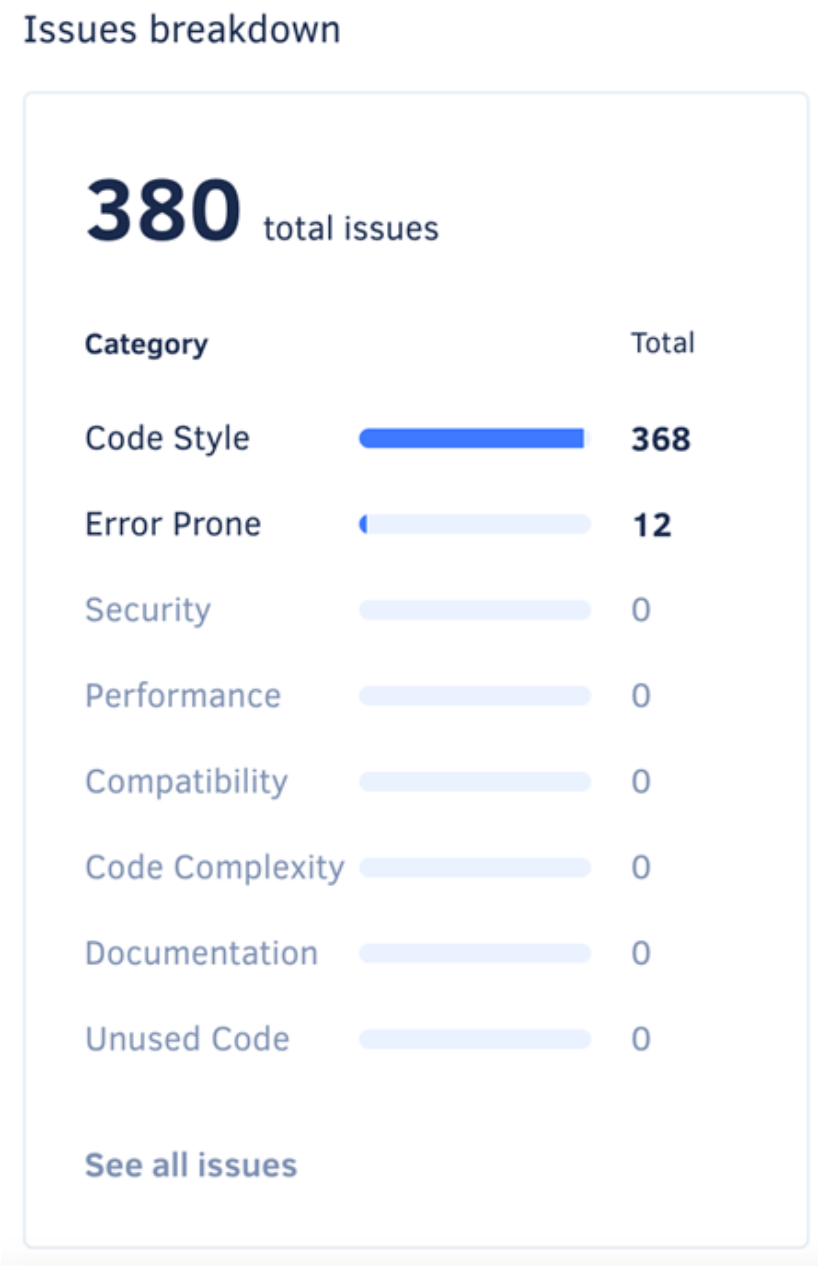
4.2.2.1.5 Ανάλυση τρωτοτήτων για το Jmix

Εικόνα 15 Τρωτότητες JMix από το codacy

Το Jmix εμφάνισε κυρίως μεσαίου αλλά και χαμηλού επιπέδου ζητήματα ασφαλείας. Συγκεκριμένα εμφάνισε μόνο 3 μεσαίου επιπέδου ζητήματα και 32 χαμηλού επιπέδου από τα 35 που είχε συνολικά.

4.2.2.1.6 Ανάλυση τρωτοτήτων για το JHipster

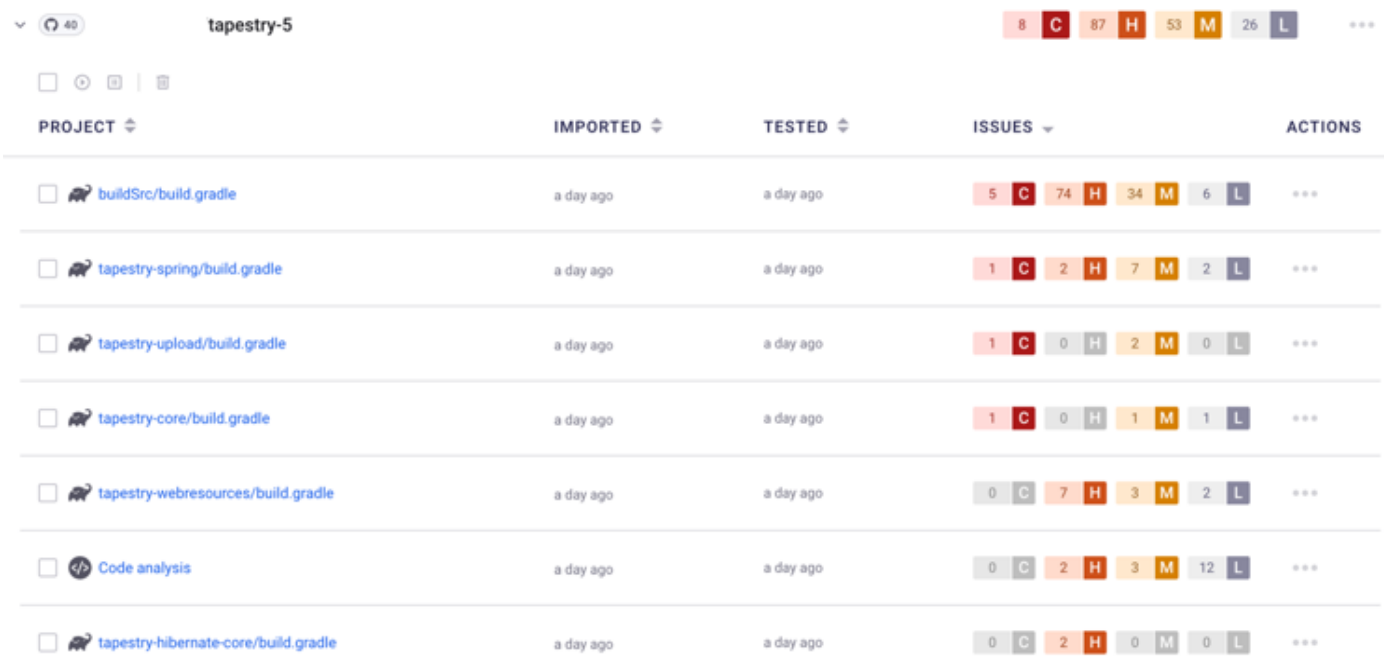
Για το JHipster δεν βρέθηκαν προβλήματα ασφάλειας από το codacy σε κανένα επίπεδο.



Εικόνα 16 Τρωτότητες JHipster από το codacy

4.2.2.2 Ανάλυση με χρήση του Snyk

Στην παρακάτω ενότητα γίνεται ανάλυση των τρωτοτήτων κάθε πλαίσιο με χρήση του εργαλείου Snyk.



Εικόνα 17 Τρωτότητες Tapestry από το Snyk

4.2.2.2.1 Ανάλυση τρωτοτήτων για το Tapestry

Το Snyk αναγνώρισε συνολικά 8 κρίσιμου επιπέδου θέματα ασφάλειας, 87 υψηλού, 53 μεσαίου και 26 χαμηλού.

Οι 5 από τις πιο κρίσιμες οφείλονται στο αρχείο build.gradle του φακέλου buildSrc, το οποίο χρησιμοποιείται για τη διαχείριση εξωτερικών βιβλιοθηκών σε έργα Gradle. Είναι φυσικό οι τρωτότητες να εμφανίζονται σε αυτό το αρχείο διότι αυτό χρησιμοποιείται για την διαμόρφωση των εξαρτήσεων του έργου και ο σαρωτής τρωτοτήτων εξετάζει την ύπαρξη τρωτοτήτων που αφορούν τις εξαρτήσεις ενός έργου.

Το Gradle είναι ένα εργαλείο αυτοματοποίησης της διαδικασίας δόμησης (build automation) για τη δημιουργία και τη διαχείριση έργων λογισμικού. Ανήκει στην κατηγορία των συστημάτων διαχείρισης δόμησης (build systems) και χρησιμοποιείται κυρίως για τη διαχείριση έργων που βασίζονται σε Java, αλλά μπορεί επίσης να χρησιμοποιηθεί για έργα που χρησιμοποιούν άλλες γλώσσες προγραμματισμού. Υιοθετεί ένα σύστημα διαχείρισης εξαρτήσεων που βασίζεται σε αρχεία κώδικα γραμμένα σε Groovy ή Kotlin, που είναι γλώσσες προγραμματισμού που τρέχουν στην Java Virtual Machine (JVM). Χρησιμοποιείται ευρέως στον χώρο της ανάπτυξης λογισμικού για τη διαχείριση εξαρτήσεων, την δημιουργία, τον έλεγχο της έκδοσης (version control), και άλλες εργασίες δόμησης.

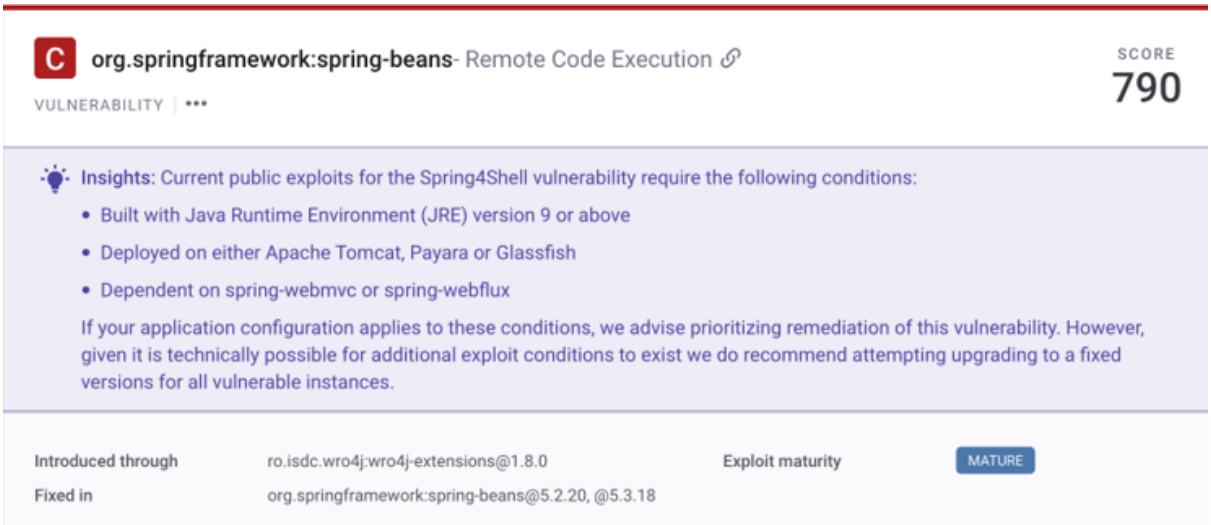
Το Gradle παρέχει ένα ευέλικτο και δυναμικό μοντέλο δόμησης

1. Φάκελος **buildSrc:**
    - Ο κώδικας που περιέχεται στον φάκελο buildSrc χρησιμοποιείται για την προσθήκη πρόσθετων κατασκευαστικών λειτουργιών, προσαρμοσμένων tasks ή plugins που μπορούν να χρησιμοποιηθούν στο κύριο έργο.
2. Αρχείο **build.gradle:**
    - Το αρχείο build.gradle χρησιμοποιείται για τη δήλωση των εξωτερικών βιβλιοθηκών και των εξαρτήσεων που απαιτούνται από το κύριο έργο.
    - Εδώ ορίζονται οι εξαρτήσεις, οι εκδόσεις των βιβλιοθηκών και οποιεσδήποτε άλλες ρυθμίσεις σχετικές με τις εξωτερικές βιβλιοθήκες.

Πατώντας πάνω σε κάθε ευπάθεια που απεικονίζεται στην προηγούμενη εικόνα που εμφανίζει το dashboard που παράγει το εργαλείο Snyk, εμφανίζονται περισσότερες λεπτομέρειες σχετικά με την ευπάθεια αυτή. Παρακάτω θα αναλύσουμε τις 8 κρίσιμες ευπάθειες που προέκυψαν από τον έλεγχο.

1. **Remote Code Execution** (Απομακρυσμένη εκτέλεση κώδικα)

Ανάλυση Τρωτότητας



Εικόνα 18 Στιγμιότυπο τρωτότητας

Το org.springframework:spring-beans είναι ένα πακέτο που αποτελεί τη βάση για το κοντέινερ IoC του Spring Framework. Η διεπαφή BeanFactory παρέχει έναν προηγμένο μηχανισμό διαμόρφωσης ικανό να διαχειρίζεται οποιοδήποτε είδος αντικειμένου.

Οι επηρεαζόμενες εκδόσεις αυτού του πακέτου είναι ευάλωτες στην απομακρυσμένη εκτέλεση κώδικα μέσω χειρισμού του ClassLoader που μπορεί να επιτευχθεί με ένα αίτημα POST HTTP. Αυτό θα μπορούσε να επιτρέψει σε έναν εισβολέα να εκτελέσει ένα τερματικό ιστού (web terminal) στην εφαρμογή ενός θύματος (TomCat) ή να κατεβάσει αυθαίρετα αρχεία από τον διακομιστή (Payara/Glassfish).

Επιπλέον, στο αρχείο δηλώνεται η χρήση του πακέτου wro4j-extensions από την εξωτερική βιβλιοθήκη org.springframework:spring-beans. Το "wro4j-extensions" είναι μια επέκταση για το εργαλείο "wro4j," το οποίο είναι ένα πλαίσιο (framework) για τη διαχείριση και την συμπίεση πόρων στην ανάπτυξη ιστοσελίδων (web development). Το "wro4j" επικεντρώνεται κυρίως στον συνδυασμό, τη συμπίεση και τον διαχωρισμό των προ-επεξεργασμένων πόρων (όπως CSS, JavaScript) προκειμένου να βελτιστοποιήσει την απόδοση των ιστοσελίδων. Οι "επεκτάσεις" (extensions) συνήθως προσφέρουν πρόσθετες δυνατότητες ή επιπρόσθετες λειτουργίες στο βασικό πλαίσιο. Επομένως, το "wro4j-extensions" περιλαμβάνει πρόσθετες δυνατότητες ή εργαλεία (με τη μορφή επεκτάσεων) που μπορούν να χρησιμοποιηθούν μαζί με το "wro4j" για την ανάπτυξη ιστοσελίδων.

```
dependencies {
        implementation ("ro.isdc.wro4j:wro4j-extensions:1.8.0"){
        exclude group: 'org.jruby'
        exclude module: 'spring-web'
        exclude module: 'closure-compiler'
        exclude module: 'gmaven-runtime-1.7'
        exclude module: 'less4j'
        }
        gradleApi()
}
```

Ωστόσο η έκδοση του πακέτου που χρησιμοποιείται είναι η 1.8.0, στην οποία έχει ανακαλυφθεί η ευπάθεια Spring4Shell. Το Spring4Shell, είναι μια κρίσιμη ευπάθεια απομακρυσμένης εκτέλεσης κώδικα (RCE) στο Spring Framework (εκδόσεις 5.3.0 έως 5.3.17, 5.2.0 έως 5.2.19, καθώς και παλαιότερες μη υποστηριζόμενες εκδόσεις).

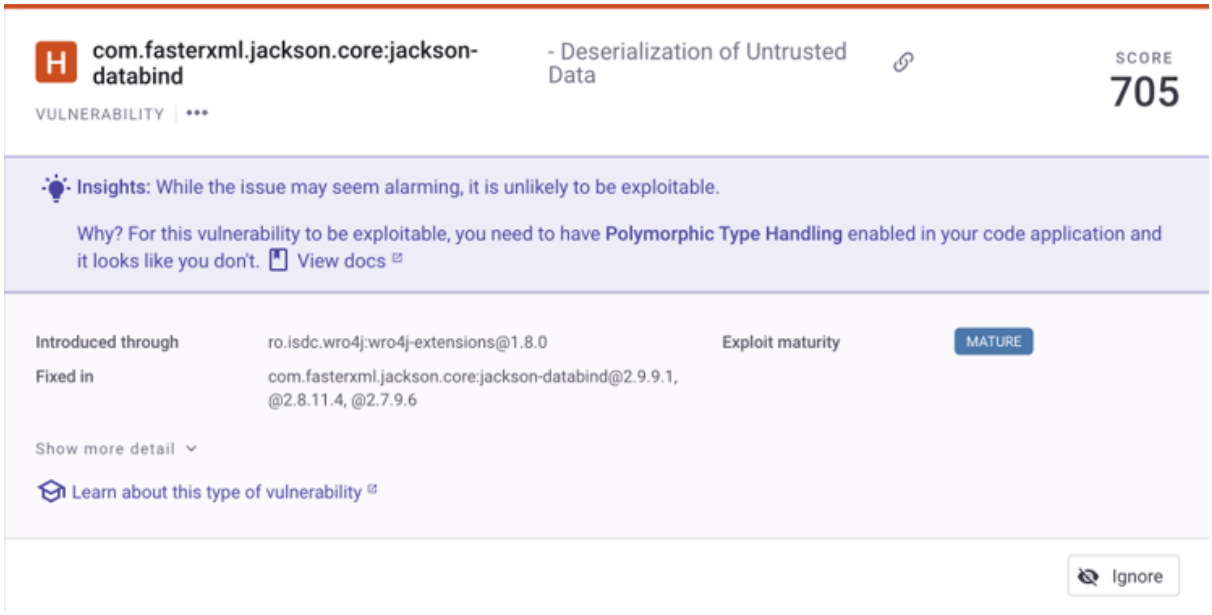
Η ευπάθεια Spring4Shell εντοπίστηκε στις 29 Μαρτίου 2022. Δυστυχώς, οι αποδείξεις των εννοιών που αποδεικνύουν την εκμετάλλευση της ευπάθειας διέρρευσαν στο διαδίκτυο πριν η ομάδα της Spring καταφέρει να κυκλοφορήσει μια ενημέρωση κώδικα, με αποτέλεσμα το Spring4Shell να χαρακτηριστεί ως zero-day ευπάθεια. Οι συντηρητές Spring κυκλοφόρησαν ενημερώσεις κώδικα στις 31 Μαρτίου 2022, μετριάζοντας την ευπάθεια.

Τρόπος αντιμετώπισης

Προτείνεται η ενημέρωση του org.springframework:spring-beans στην έκδοση 5.2.20 ή σε υψηλότερη.

2. **Deserialization of Untrusted Data** (Αποσειριοποίηση Μη Εμπιστευτικών Δεδομένων)

Ανάλυση Τρωτότητας



Εικόνα 19 Στιγμιότυπο τρωτότητας

Πρόκειται για ευπάθεια που αφορά την αποκωδικοποίηση (deserialization) αναξιόπιστων δεδομένων στη βιβλιοθήκη Jackson, συγκεκριμένα στην εξάρτηση com.fasterxml.jackson.core:jackson-databind. Αυτό το είδος ευπάθειας συνήθως σχετίζεται με τη διαχείριση της αποκωδικοποίησης JSON ή άλλων δομών δεδομένων κατά την αποσυσκευασία τους από μια σειρά bytes. Όταν δεδομένα αποκωδικοποιούνται χωρίς την κατάλληλη ασφαλή διαχείριση, μπορεί να υπάρξουν ευπάθειες που επιτρέπουν επιθέσεις, όπως την εκτέλεση κώδικα (code execution) ή άλλες ανεπιθύμητες ενέργειες.

Οφείλεται και πάλι στο πακέτο “wro4j-extensions”. Ωστόσο παρόλο που έχει πολύ υψηλό σκορ και αξιολογείται κρίσιμη, το snyk μας ενημερώνει πως δεν μπορεί κάποιος να την εκμεταλλευτεί άμεσα εφόσον ο Χειρισμός Πολυμορφικών Τύπων (Polymorphic Type Handling) είναι απενεργοποιημένος.

Ο Χειρισμός Πολυμορφικών Τύπων αναφέρεται στον τρόπο με τον οποίο ένα σύστημα αντιμετωπίζει και διαχειρίζεται αντικείμενα που μπορεί να ανήκουν σε διάφορους υποτύπους (υποκλάσεις) ενός κοινού γονικού τύπου.

Ορισμένες γλώσσες προγραμματισμού παρέχουν δυνατότητες πολυμορφισμού οπότε ο χειρισμός πολυμορφικών τύπων συχνά σχετίζεται με τον τρόπο αντιμετώπισης της σειράς τύπων κληρονομικότητας (inheritance). Ένας τρόπος χειρισμού των πολυμορφικών τύπων είναι:

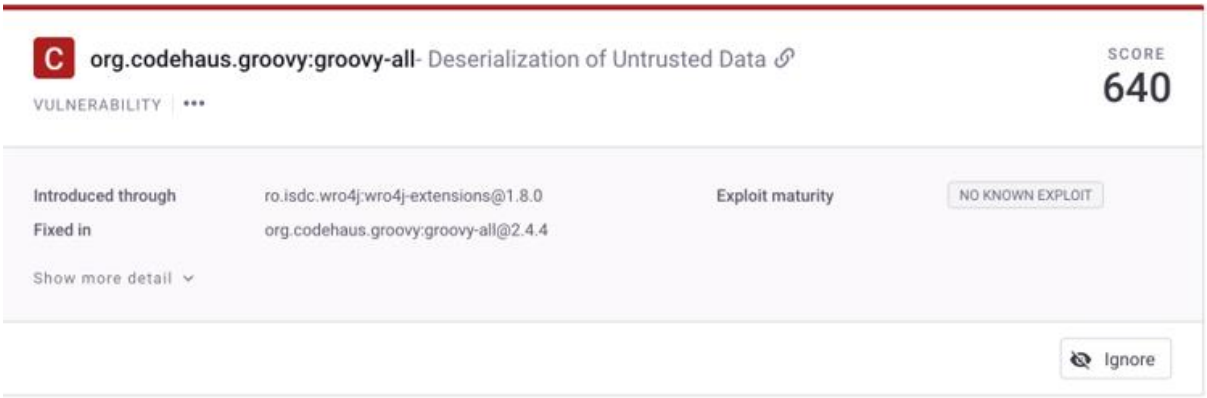
- **Στατική Δέσμευση (Static Binding):** Στη στατική δέσμευση, ο τύπος του αντικειμένου καθορίζεται κατά τον χρόνο συγγραφής του κώδικα. Αυτό σημαίνει ότι ο μεταγλωττιστής γνωρίζει προκαταβολικά τον τύπο του αντικειμένου και εκχωρεί τις κατάλληλες μεθόδους κατά τον χρόνο μεταγλώττισης.

Ο χειρισμός πολυμορφικών τύπων είναι σημαντικός για την επίτευξη ευελιξίας και επαναχρησιμοποίησης κώδικα σε αντικειμενοστρεφείς γλώσσες προγραμματισμού, καθώς επιτρέπει την αντιμετώπιση διαφορετικών τύπων αντικειμένων με έναν ομοιόμορφο τρόπο.

Τρόπος αντιμετώπισης

Προτείνεται η ενημέρωση του com.fasterxml.jackson.core:jackson-databind στην έκδοση 2.9.10.8 ή σε υψηλότερη. Είναι μια βιβλιοθήκη που περιέχει τη λειτουργικότητα δέσμευσης δεδομένων γενικού σκοπού και το μοντέλο δέντρων για τον επεξεργαστή δεδομένων Jackson.

3. Deserialization of Untrusted Data



Εικόνα 20 Τρωτότητες Struts από το codacy

Ανάλυση Τρωτότητας

Η εξάρτηση org.codehaus.groovy:groovy-all αναφέρεται στη βιβλιοθήκη Groovy σε μια συγκεκριμένη έκδοση. Η εξάρτηση αυτή συνήθως χρησιμοποιείται για να προσθέσετε όλα τα απαραίτητα JAR αρχεία της Groovy στο classpath του έργου σας.

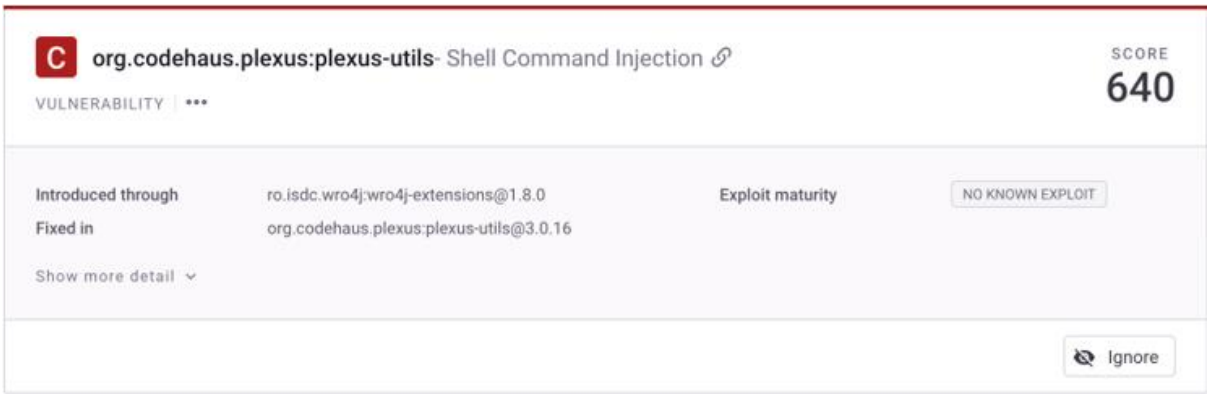
Η Groovy είναι μια γλώσσα προγραμματισμού που τρέχει στην Java Virtual Machine (JVM) και προσφέρει χαρακτηριστικά προγραμματισμού που είναι παρόμοια με αυτά της Java, αλλά με περισσότερη ευελιξία και δυνατότητες που χαρακτηρίζουν τις δυναμικές γλώσσες προγραμματισμού.

Η εξάρτηση org.codehaus.groovy:groovy-all συνήθως περιλαμβάνει όλες τις κλάσεις και τις εξαρτήσεις της Groovy, καθιστώντας την κατάλληλη για περιβάλλοντα όπου θέλετε να χρησιμοποιήσετε όλες τις λειτουργίες της γλώσσας. Η τρωτότητα προκύπτει από πολλούς λόγους, συμπεριλαμβανομένων σφαλμάτων προγραμματισμού, ευπάθειες στον έλεγχο εισόδου και στην διαχείριση αποθηκευμένων δεδομένων.

Τρόπος αντιμετώπισης

Προτείνεται η ενημέρωση του org.codehaus.groovy:groovy στην έκδοση 2.4.7 ή υψηλότερη. Επιπλέον, λόγω της χρήσης του πακέτου wro4j-extensions από το org.codehaus.groovy:groovy-all, όλες οι επηρεαζόμενες εκδόσεις του είναι ευάλωτες σε Deserialization of Untrusted Data. Επομένως, μια ακόμα λύση θα ήταν η χρήση μιας έκδοσης της βιβλιοθήκης που δεν περιλαμβάνει το συγκεκριμένο πακέτο wro4j-extensions.

4. Shell Command Injection (Έκχυση Εντολών Κελύφους)



Εικόνα 21 Στιγμιότυπο τρωτότητας

Ανάλυση Τρωτότητας

Η επίθεση "Shell Command Injection" συνήθως συμβαίνει όταν εισάγονται μη αξιόπιστα δεδομένα ως είσοδος σε συστήματα που εκτελούν εντολές σε κέλυφος, χωρίς να γίνεται επαρκής έλεγχος. Αυτό μπορεί να οδηγήσει σε εκτέλεση ανεπιθύμητων εντολών ή ακόμη και σε ανεπιθύμητη πρόσβαση σε ευαίσθητα δεδομένα.

Η βιβλιοθήκη plexus-utils περιέχει το wro4j-extensions. Πιο συγκεκριμένα, το plexus-utils είναι μέρος του codehaus Plexus, το οποίο είναι μια συλλογή από στοιχεία που χρησιμοποιούνται από το Apache Maven (εργαλείο κατασκευής αντίστοιχου του Gradle). Η plexus-utils παρέχει βοηθητικές μεθόδους και εργαλεία για διάφορες εργασίες, όπως:

1. Εργαλεία εκτέλεσης εντολών και διαχείρισης διεργασιών.
2. Διάφορες χρήσιμες μεθόδους για την εργασία με αρχεία και καταλόγους.
3. Μεθόδους για τη διαχείριση των καταχωρητών (loggers) και των καταχωρητών χρηστών.
4. Χρήσιμες μεθόδους για τη διαχείριση συμβολοσειρών.

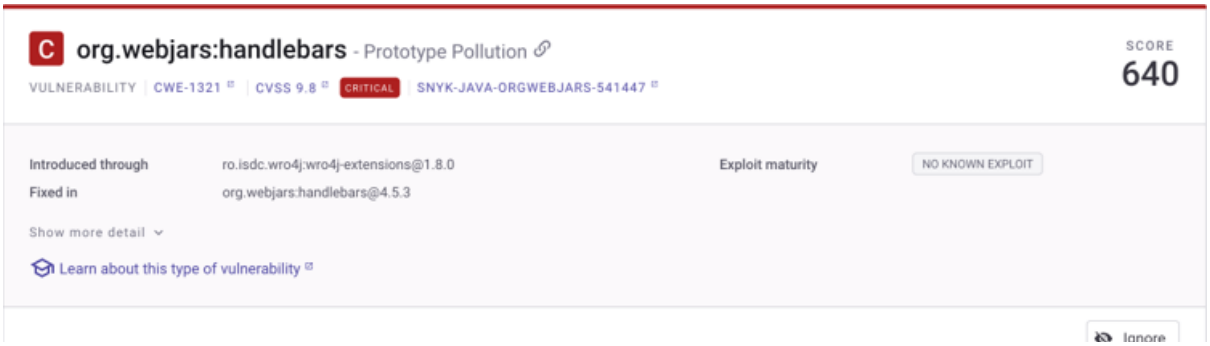
Οι εκδόσεις αυτής της βιβλιοθήκης είναι ευάλωτες στο Shell Command Injection. Η κλάση Commandline στο plexus-utils που χρησιμοποιείται για τη δημιουργία και εκτέλεση εντολών στο κέλυφος (shell) του συστήματος, δεν χειρίζεται σωστά τα ορίσματα εντολής και τα κελύφη εκτέλεσης εντολών ενώ δεν παραθέτει σωστά τα περιεχόμενα των συμβολοσειρών με διπλά εισαγωγικά.

Τρόπος αντιμετώπισης

Προτείνεται η ενημέρωση του Codehaus Plexus στην έκδοση 3.0.16 ή σε υψηλότερη.

5. Prototype Pollution (Ρύπανση Πρωτοτύπου)

Ανάλυση Τρωτότητας



Εικόνα 22 Στιγμιότυπο Τρωτότητας

Το org.webjars:handlebars είναι μια επέκταση στη γλώσσα προτύπων Mustache. Τόσο το Mustache όσο και το Handlebars είναι γλώσσες προτύπων (template languages) που επιτρέπουν την ενσωμάτωση δεδομένων σε πρότυπα κειμένου. Ωστόσο, το Handlebars επεκτείνει τη σύνταξη του Mustache και προσφέρει πρόσθετες δυνατότητες και λειτουργίες.

Οι επιπλέον δυνατότητες του Handlebars περιλαμβάνουν τη δυνατότητα χρήσης πιο σύνθετων δομών δεδομένων, την ενσωμάτωση λογικής χειρισμού (όπως συνθήκες και επαναλήψεις), και άλλες βελτιώσεις στην σύνταξη των προτύπων. Επιπλέον, το Handlebars παρέχει τη δυνατότητα ορισμού προσαρμοσμένων βοηθητικών συναρτήσεων (helpers) που μπορούν να επεκτείνουν τη λειτουργικότητα των προτύπων.

Η βιβλιοθήκη org.webjars:handlebars είναι μια εκδοχή του Handlebars που έχει ενσωματωθεί στο σύστημα εξαρτήσεων Maven ως WebJar. Τα WebJars είναι πακέτα JAR που περιέχουν πόρους του web (όπως JavaScript, CSS, κ.λπ.) που μπορούν να χρησιμοποιηθούν σε μια εφαρμογή ιστού Java.

Χρησιμοποιώντας το org.webjars:handlebars στο Maven, μπορείτε να ενσωματώσετε το Handlebars στο έργο Java σας ως εξάρτηση και να το χρησιμοποιήσετε για τη δημιουργία προτύπων HTML στην πλευρά του πελάτη ή τη διαχείριση προτύπων στην πλευρά του διακομιστή.

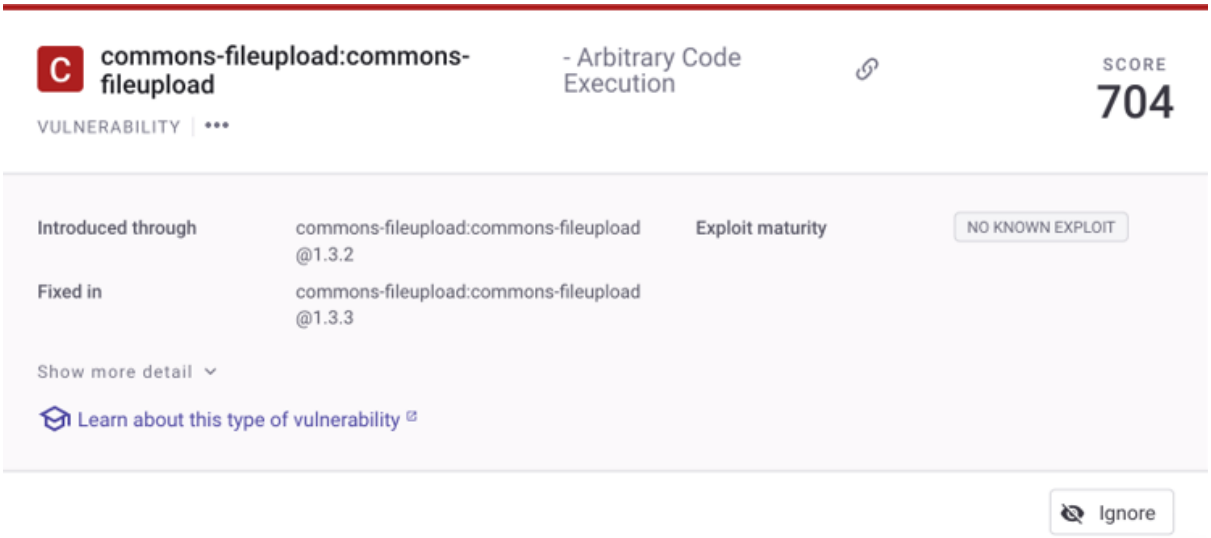
Οι επηρεαζόμενες εκδόσεις αυτού του πακέτου είναι ευάλωτες στη Ρύπανση Πρωτοτύπου (Prototype Pollution). Η Ρύπανση Πρωτοτύπου είναι μια ευπάθεια που σχετίζεται με τον τρόπο που αντιμετωπίζονται και τροποποιούνται τα πρωτότυπα των αντικειμένων σε γλώσσες προγραμματισμού, όπως η JavaScript. Είναι δυνατή η προσθήκη ή τροποποίηση ιδιοτήτων στο πρωτότυπο του αντικειμένου (γενικό αντικείμενο στη Javascript) μέσω ενός κακόβουλου προτύπου. Αυτό μπορεί να επιτρέψει στους εισβολείς να διακόψουν την εφαρμογή ή να εκτελέσουν αυθαίρετο κώδικα υπό συγκεκριμένες συνθήκες.

Τρόπος αντιμετώπισης

Προτείνεται η ενημέρωση του org.webjars:handlebars στην έκδοση 4.5.3 ή σε υψηλότερη.

6. Arbitrary Code Execution (Εκτέλεση αυθαίρετου κώδικα)

Ανάλυση Τρωτότητας



Εικόνα 23 Στιγμιότυπο Τρωτότητας

Η ευπάθεια οφείλεται στη βιβλιοθήκη commons-fileupload:commons-fileupload@1.3.2.

Το Apache Commons FileUpload είναι μια βιβλιοθήκη Java που χρησιμοποιείται για την επεξεργασία αιτημάτων HTTP που περιέχουν μεταφορά αρχείων. Η βιβλιοθήκη FileUpload του Apache Commons περιέχει ένα αντικείμενο Java που, κατά την αποσειριοποίηση, μπορεί να χρησιμοποιηθεί για την εγγραφή ή την αντιγραφή αρχείων σε αυθαίρετες τοποθεσίες. Εάν ενσωματωθεί με το ysoserial, είναι δυνατή η μεταφόρτωση και η εκτέλεση δυαδικών αρχείων σε μία μόνο κλήση αποσειριοποίησης.

Το "ysoserial" είναι ένα εργαλείο για τη δημιουργία και εκτέλεση επιθέσεων αξιοποιώντας ευπάθειες στον μηχανισμό αποσειριοποίησης της Java. Η Αποσειριοποίηση Java (Java Serialization) είναι ένας μηχανισμός που χρησιμοποιείται για τη μετατροπή αντικειμένων σε byte streams και αντιστρόφως. Η ευπάθεια συνήθως παρατηρείται όταν ένα αντικείμενο που περιέχει ευαίσθητες πληροφορίες αποστέλλεται ή λαμβάνεται χωρίς την αναγκαία προσοχή.

Το ysoserial χρησιμοποιείται συχνά σε σενάρια εκμετάλλευσης όπου ο επιτιθέμενος προσπαθεί να εκτελέσει κώδικα αυθαίρετου τύπου (Arbitrary Code Execution) σε μια ευάλωτη εφαρμογή που χρησιμοποιεί Java Serialization.

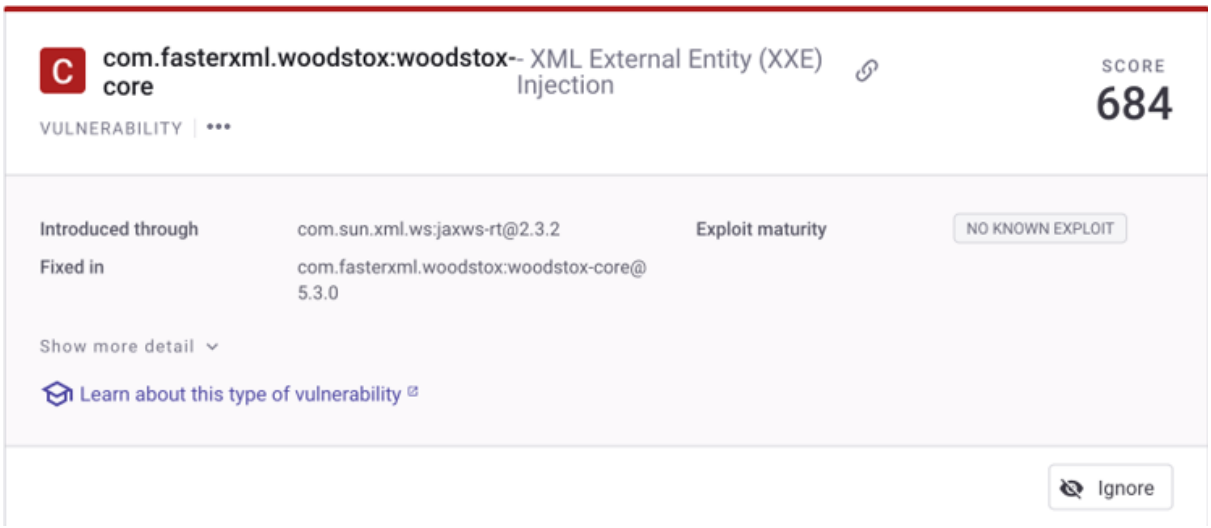
Τρόπος αντιμετώπισης

Προτείνεται η ενημέρωση το commons-fileupload στην έκδοση 1.3.3 ή σε υψηλότερη.

7. XML External Entity (XXE) Injection (Έγχυση εξωτερικής οντότητας XML (XXE))

Ανάλυση Τρωτότητας

Οι επηρεαζόμενες εκδόσεις αυτού του πακέτου είναι ευάλωτες στο XML External Entity (XXE) Injection, λόγω μη ασφαλούς επεξεργασίας και έλλειψης περιορισμού στο χειρισμό των αρχείων XML. Ένας εισβολέας μπορεί να εκμεταλλευτεί αυτή την ευπάθεια στέλνοντας ένα ειδικά δημιουργημένο κακόβουλο αρχείο XML που περιέχει εξωτερικές οντότητες XML με URI που επιλύονται σε έγγραφα εκτός της επιδιωκόμενης σφαίρας ελέγχου. Αν το σύστημα δέχεται την ανάλυση εξωτερικών οντοτήτων και έχει πρόσβαση σε εξωτερικά έγγραφα, ο εισβολέας μπορεί να πετύχει διάφορες επιπτώσεις. Μερικά παραδείγματα περιλαμβάνουν:



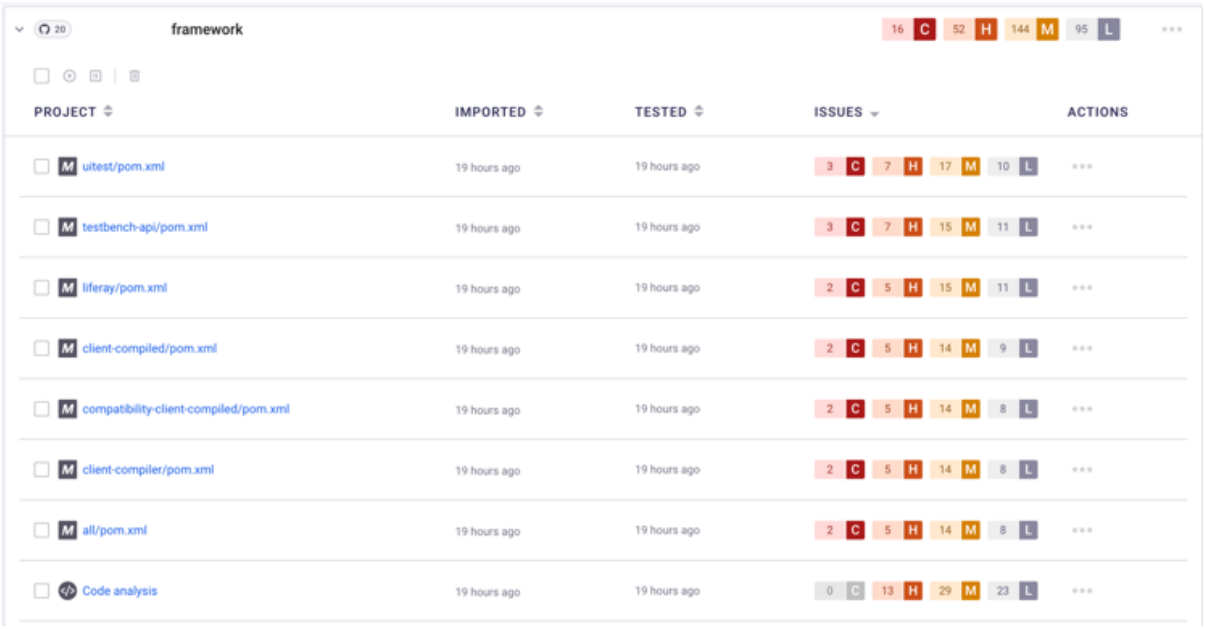
Εικόνα 24 Στιγμιότυπο Τρωτότητας

1. **Δηλητηρίαση Αρχείων Συστήματος (File System Poisoning):** Ο εισβολέας μπορεί να πετύχει την δηλητηρίαση αρχείων στο σύστημα, εκτελώντας κακόβουλο κώδικα.
2. **Αποκάλυψη Ευαίσθητων Πληροφοριών:** Μπορεί να ανακαλύψει ευαίσθητες πληροφορίες από εξωτερικά έγγραφα.
3. **Απομακρυσμένη Εκτέλεση Κώδικα (Remote Code Execution):** Ο εισβολέας μπορεί να προκαλέσει την απομακρυσμένη εκτέλεση κακόβουλου κώδικα εάν εκτελεστεί κακόβουλος κώδικας από τα εξωτερικά έγγραφα.

Τρόπος αντιμετώπισης

Προτείνεται ενημέρωση του com.fasterxml.woodstox:woodstox-core στην έκδοση 5.3.0 ή σε υψηλότερη

4.2.2.2.2 Ανάλυση Τρωτοτήτων για το Vaadin



Εικόνα 25 Τρωτότητες Vaadin από το Snyk

Το Snyk σαρώνοντας το Vaadin εμφάνισε συνολικά 16 κρίσιμου επιπέδου θέματα ασφάλειας, 52 υψηλού, 144 μεσαίου και 95 χαμηλού. Τα επόμενα 3 βρίσκονται στο αρχείο pom.xml στον φάκελο uitest. Επίσης από τα 16 κρίσιμα θέματα ασφάλειας, τα 3 βρίσκονται στο αρχείο pom.xml στον φάκελο testbench-api (περιλαμβάνει δοκιμαστικό κώδικα που ελέγχει τη σωστή λειτουργία του API (Διεπαφή Προγραμματισμού Εφαρμογών)). Τέλος 2 από τις 16 κρίσιμες ευπάθειες βρίσκονται στο αρχείο pom.xml του φακέλου lifera και οι ίδιες δύο βρίσκονται και στους φακέλους client-compiled, compatibility-client-compiled, client-compiler και all.

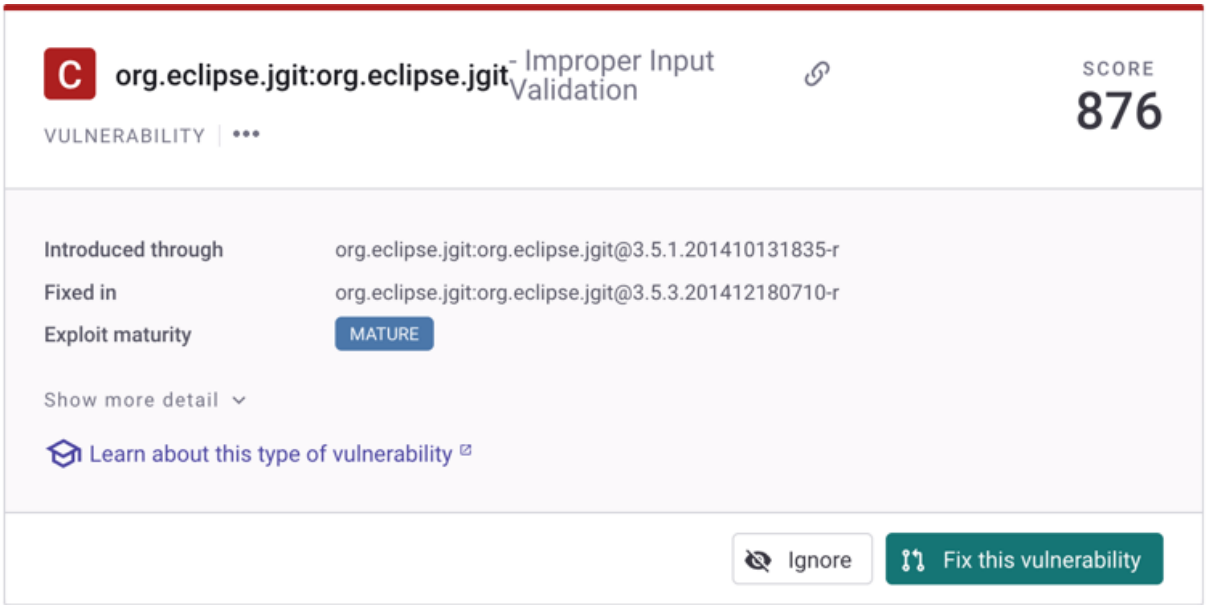
Θα αναλύσουμε αρχικά τις τρωτότητες του φακέλου uitest στο αρχείο pom. Τα αρχεία pom χρησιμοποιούνται στην αρχιτεκτονική maven για την δήλωση συγκεκριμένων στοιχείων του project. Ορισμένα από τα βασικά στοιχεία που μπορούν να δηλωθούν σε ένα αρχείο pom.xml περιλαμβάνουν:

1. **Πληροφορίες Έργου (Project Information):**
   - Όνομα και περιγραφή του έργου.
   - Στοιχεία επαφής για τους δημιουργούς του έργου.
   - URL του έργου.
2. **Διαμόρφωση Έργου (Project Configuration):**
   - Έκδοση της Java που χρησιμοποιείται στο έργο.
   - Ρυθμίσεις μεταγλώττισης και εκτέλεσης του κώδικα.
3. **Εξαρτήσεις (Dependencies):**
   - Δηλώσεις εξαρτήσεων σε εξωτερικές βιβλιοθήκες.
   - Έκδοση των εξωτερικών βιβλιοθηκών που χρησιμοποιούνται.
4. **Διαχείριση Έργου (Build Management):**
   - Οδηγίες για τον κύκλο ζωής του έργου (compile, test, package, etc.).
   - Πλατφόρμες εκτέλεσης (plugins).
5. **Σύνοψη Δομής Καταλόγου (Directory Structure):**
   - Δήλωση των καταλόγων όπου βρίσκονται οι πηγές, τα πειραματικά αρχεία, τα αρχεία πόρων, κλπ.

Μέσω αυτού του αρχείου, το Maven έχει τη δυνατότητα να οδηγήσει την διαδικασία κατασκευής, δοκιμής, και παράδοσης του έργου σας, αλλά και να διαχειρίζεται τις εξαρτήσεις του.

1.Improper Input Validation (Ακατάλληλη Επικύρωση Εισόδου)

Ανάλυση Τρωτότητας



Εικόνα 26 Στιγμιότυπο Τρωτότητας

Η αναφορά "Improper Input Validation" συνήθως υποδηλώνει ένα πρόβλημα ασφαλείας που σχετίζεται με τον τρόπο που μια εφαρμογή ή μια βιβλιοθήκη ελέγχει και επικυρώνει την είσοδο που λαμβάνει από τον χρήστη ή από άλλες πηγές. Η έλλειψη σωστού ελέγχου εισόδου μπορεί να οδηγήσει σε ποικίλα προβλήματα ασφαλείας, όπως προσβολή της ακεραιότητας δεδομένων, εκτέλεση κώδικα από απομακρυσμένες πηγές, έκχυση SQL κ.ά.

JGit: πρόγραμμα πελάτη (client) για το σύστημα διαχείρισης εκδόσεων Git

Το org.eclipse.jgit είναι μια βιβλιοθήκη υλοποίησης προγράμματος πελάτη στην Java για το σύστημα διαχείρισης εκδόσεων Git. Το JGit ανήκει στο πλαίσιο του Eclipse Foundation και παρέχει ένα Java API για την επικοινωνία με αποθετήρια Git, τη δημιουργία και τη διαχείριση αντιγράφων ασφαλείας (commits), και άλλες λειτουργίες που σχετίζονται με το Git.

Το JGit επιτρέπει σε εφαρμογές που χρησιμοποιούν τη γλώσσα προγραμματισμού Java να αλληλεπιδρούν με τα αποθετήρια Git χωρίς την ανάγκη για την εκτέλεση εξωτερικών εντολών Git από το κέλυφος του συστήματος. Επίσης για τη διαχείριση του πηγαίου κώδικα, για παράδειγμα, όταν αναπτύσσεται ή διαχειρίζεται ένα project σε ένα αποθετήριο Git.

Μερικά χαρακτηριστικά του JGit περιλαμβάνουν:

1. **Πρόσβαση σε Αποθετήρια Git:** Παρέχει Java API για να αλληλεπιδρά με τα αποθετήρια Git.
2. **Δημιουργία Commits:** Επιτρέπει τη δημιουργία, τη διαχείριση και την ανάκτηση των commits.
3. **Πλοήγηση στο Ιστορικό:** Παρέχει μέσα για την περιήγηση και την ανάκτηση ιστορικού αλλαγών.
4. **Αλλαγές και Κλάδευση (Branching):** Υποστηρίζει τις λειτουργίες που σχετίζονται με τη διαχείριση των αλλαγών και τη δημιουργία κλαδιών.

5. **Διαχείριση Αρχείων:** Επιτρέπει την εργασία με αρχεία και αλλαγές στα αποθετήρια.

Η παραπάνω ευπάθεια οφείλεται στην έκδοση 3.5.1 του πακέτου org.eclipse.jgit. Οι εκδόσεις αυτού του πακέτου που επηρεάζονται είναι ευάλωτες σε ακατάλληλη επικύρωση εισόδου. Το Git στις εκδόσεις πριν από την 2.2.1 σε Windows και OS X καθώς και παλιότερες εκδόσεις του JGit (πριν από τις 08-12-2014) επιτρέπουν στους απομακρυσμένους διακομιστές Git να εκτελούν αυθαίρετες εντολές μέσω ενός δέντρου που περιέχει ένα δημιουργημένο αρχείο .git/config με

(1) ένα αδιάφορο σημείο κωδικού Unicode,

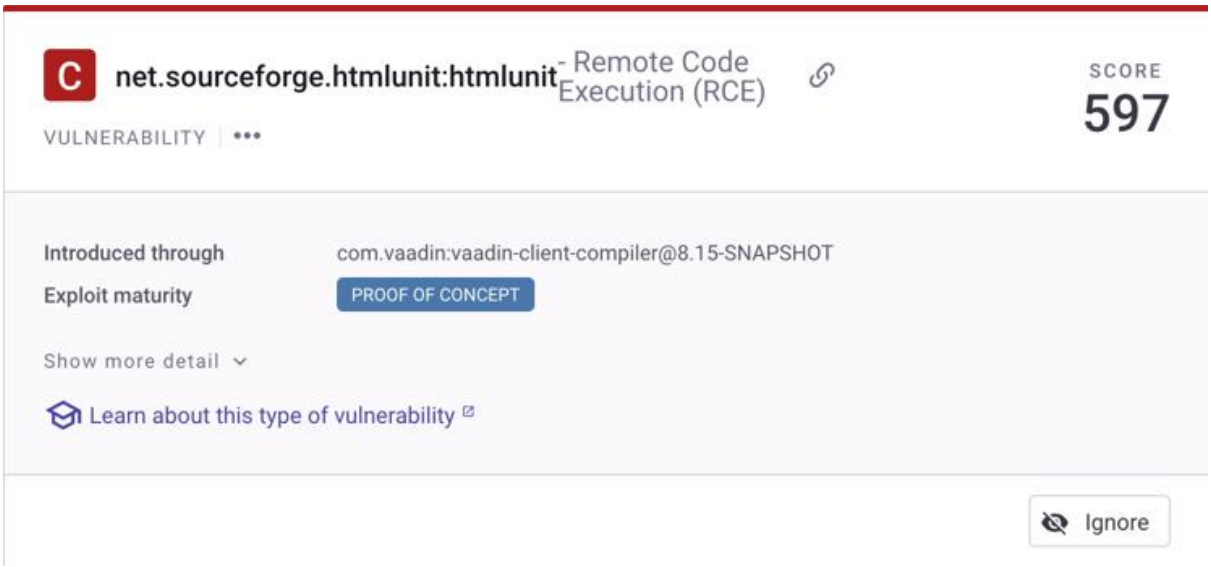
(2) μια μίξη κεφαλαίων-πεζών που δεν αντιμετωπίζετε σωστά σε σύστημα αρχείων χωρίς διάκριση πεζών-κεφαλαίων.

Τρόπος αντιμετώπισης

Προτείνεται η ενημέρωση του org.eclipse.jgit:org.eclipse.jgit στην έκδοση 3.5.3.201412180710-r ή σε υψηλότερη.

2. Remote Code Execution - RCE (Απομακρυσμένη Εκτέλεση Κώδικα)

Ανάλυση Τρωτότητας



Εικόνα 27 Στιγμιότυπο Τρωτότητας

Η αναφορά σε "Remote Code Execution (RCE)" σε σχέση με το net.sourceforge.htmlunit:htmlunit υποδηλώνει ότι υπάρχει μια ευπάθεια ασφάλειας που επιτρέπει σε επιτιθέμενους να εκτελέσουν κακόβουλο κώδικα απομακρυσμένα στο πλαίσιο εφαρμογής που χρησιμοποιεί τη συγκεκριμένη βιβλιοθήκη.

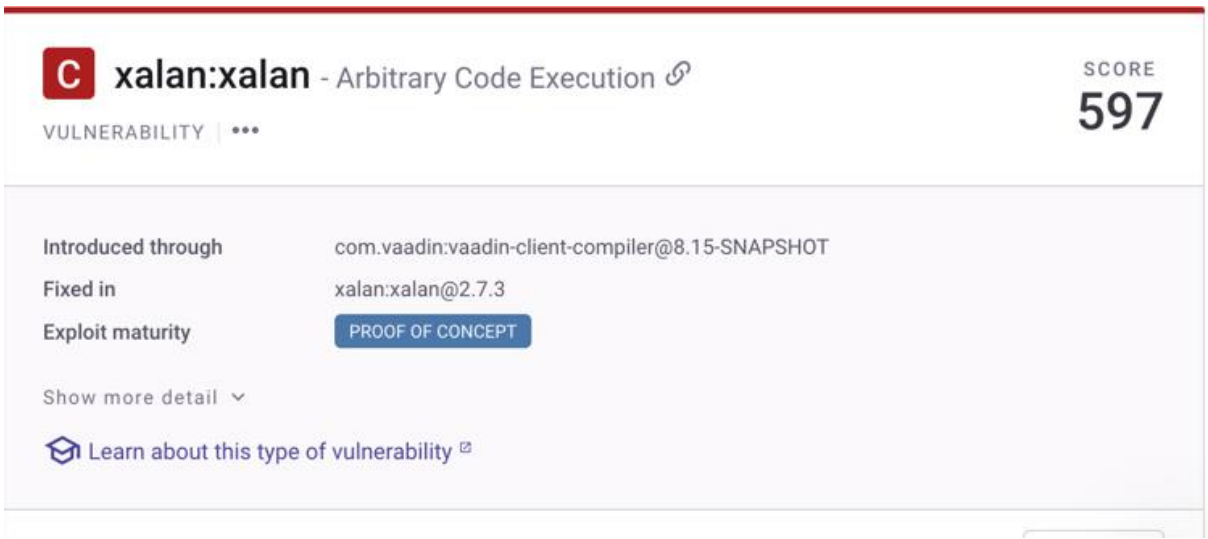
Το πακέτο net.sourceforge.htmlunit:htmlunit ανήκει στη βιβλιοθήκη HTMLUnit. Η HTMLUnit είναι μια βιβλιοθήκη της Java που προσομοιώνει τον περιηγητή (browser) και επιτρέπει την εκτέλεση δοκιμαστικών σεναρίων σε HTML σε περιβάλλον Java, η οποία συνήθως συμβαίνει για τον έλεγχο της συμπεριφορά της ιστοσελίδας ή της εφαρμογής ιστού σε ένα περιβάλλον που προσομοιώνει τον περιηγητή.
Η ευπάθεια αυτή οφείλεται στην έκδοση 2.19 του πακέτου htmlunit. Οι επηρεαζόμενες εκδόσεις αυτού του πακέτου είναι ευάλωτες στην απομακρυσμένη εκτέλεση κώδικα (RCE) μέσω XSLT, κατά την περιήγηση στην ιστοσελίδα του εισβολέα.

Τρόπος αντιμετώπισης

Δεν υπάρχει επιδιορθωμένη έκδοση αυτού του πακέτου. Παρ’ όλα αυτά, προτείνεται η αναβάθμιση σε org.htmlunit:htmlunit v3.0.0 που περιέχει λύση του θέματος.

3. Arbitrary Code Execution (Εκτέλεση Αυθαίρετου Κώδικα)

Ανάλυση Τρωτότητας



Εικόνα 28 Στιγμιότυπο Τρωτότητας

Η αναφορά σε "Arbitrary Code Execution" σε σχέση με το xalan:xalan υποδηλώνει ότι υπάρχει μια ευπάθεια ασφάλειας που επιτρέπει σε επιτιθέμενους να εκτελέσουν αυθαίρετο κώδικα (κακόβουλος κώδικας) στο πλαίσιο του Xalan.Η παραπάνω ευπάθεια οφείλεται στο πακέτο xalan και στην έκδοση 2.7.2. Το xalan:xalan είναι ένας επεξεργαστής XSLT για τη μετατροπή εγγράφων XML σε HTML, κείμενο ή άλλους τύπους εγγράφων XML. Υλοποιεί την XSL Transformations (XSLT) στην έκδοση 1.0 και την XML Path Language (XPath) στην έκδοση 1.0 ενώ μπορεί να χρησιμοποιηθεί από τη γραμμή εντολών, σε μια μικροεφαρμογή ή ένα servlet ή ως λειτουργική μονάδα σε άλλο πρόγραμμα.

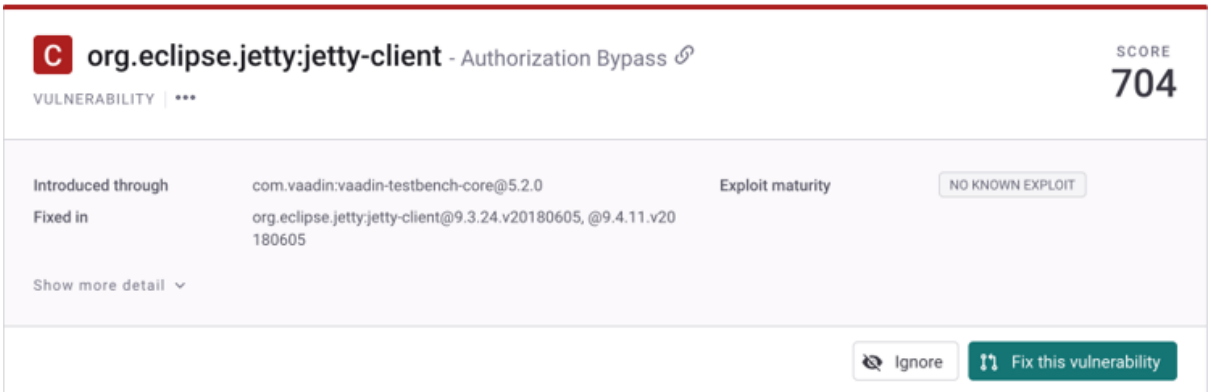
Οι επηρεαζόμενες εκδόσεις αυτού του πακέτου είναι ευάλωτες στην εκτέλεση αυθαίρετου κώδικα κατά την επεξεργασία κακόβουλων φύλλων στυλ XSLT, λόγω προβλήματος περικοπής ακεραίων. Αυτό επιτρέπει στους εισβολείς να καταστρέψουν αρχεία κλάσης Java που δημιουργούνται από τον εσωτερικό μεταγλωττιστή XSLTC και να εκτελέσουν αυθαίρετο bytecode Java.

Τρόπος αντιμετώπισης

Προτείνεται η ενημέρωση του xalan:xalan στην έκδοση 2.7.3 ή σε υψηλότερη.

Στη συνέχεια θα αναλύσουμε τις τρωτότητες που βρίσκονται στο φάκελο testbench-api.

1. **Authorization Bypass** (Παράκαμψη Εξουσιοδότησης)



Εικόνα 29 Στιγμιότυπο Τρωτότητας

Ανάλυση Τρωτότητας

Η παραπάνω ευπάθεια οφείλεται στο πακέτο jetty-client και πιο συγκεκριμένα στην έκδοση 9.4.7. Η έκδοση αυτού του πακέτου είναι ευάλωτη στην Παράκαμψη Εξουσιοδότησης (Authorization Bypass). Η "Παράκαμψη Εξουσιοδότησης" (Authorization Bypass) αναφέρεται σε μια ευπάθεια στο λογισμικό όπου ένας απομακρυσμένος εισβολέας καταφέρνει να αποφύγει τους μηχανισμούς εξουσιοδότησης που προορίζονται να εμποδίσουν μη εξουσιοδοτημένη πρόσβαση σε κρίσιμους πόρους ή λειτουργίες. Ένα μεγάλο chunk (κομμάτι δεδομένων) θα μπορούσε να ερμηνευτεί ως μικρότερο chunk και το περιεχόμενο που αποστέλλεται ως σώμα chunk θα μπορούσε να ερμηνευτεί ως pipelined(αποστολή πολλών μηνυμάτων χωρίς την αναμονή απάντησης) αίτημα. Δηλαδή, το περιεχόμενο που αποστέλλεται ως σώμα chunk μπορεί να ερμηνευτεί ως αντίστοιχο με ένα pipelined αίτημα, δηλαδή, ως πολλά συνεχόμενα αιτήματα που αποστέλλονται χωρίς να αναμένεται η απάντηση του προηγούμενου. Εάν το Jetty (web server) δεν είναι επαρκώς προστατευμένο και είχε αναπτυχθεί πίσω από έναν ενδιάμεσο φορέα που επέβαλε κάποια εξουσιοδότηση και επέτρεπε τη διαβίβαση αυθαίρετα μεγάλων τμημάτων αμετάβλητων, τότε αυτό το ελάττωμα θα μπορούσε να χρησιμοποιηθεί για την παράκαμψη της εξουσιοδότησης που επέβαλε ο ενδιάμεσος φορέας, καθώς το ψεύτικο pipelined αίτημα δεν θα ερμηνευόταν από τον ενδιάμεσο φορέα ως αίτημα.

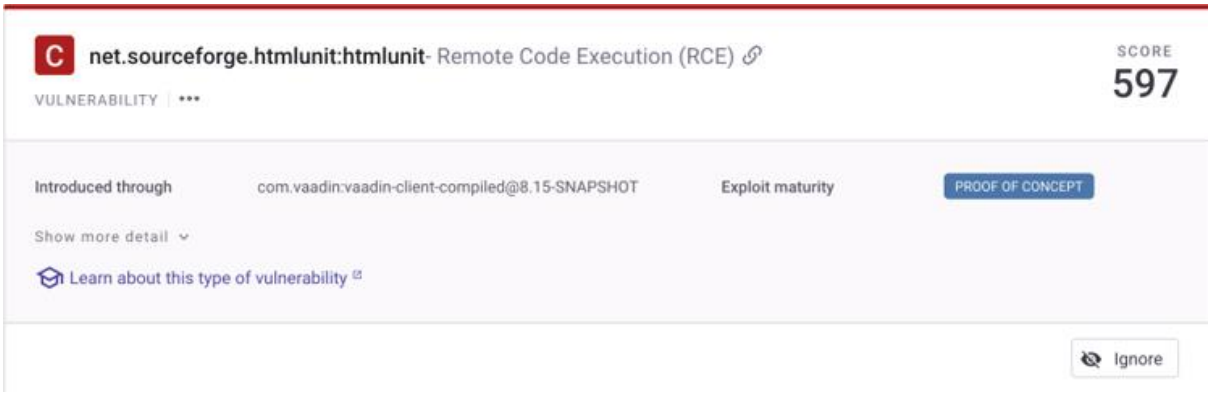
Στην πραγματικότητα, υπάρχουν διάφορες μέθοδοι και τεχνικές που ενδέχεται να επιτρέπουν σε εισβολείς να παρακάμψουν την εξουσιοδότηση, όπως η χρήση κακόβουλου κώδικα, η εκμετάλλευση ελλείψεων στον έλεγχο εξουσιοδότησης, ή η εκμετάλλευση σφαλμάτων στον κώδικα.

Τρόπος αντιμετώπισης

Προτείνεται η ενημέρωση του org.eclipse.jetty:jetty-client σε έκδοση 9.3.24.v20180605, 9.4.11.v20180605ή υψηλότερη.

Στη συνέχεια αναφέρουμε τις ευπάθειες του αρχείου pom.xml στον φάκελο testbench-api που έχουν ήδη αναλυθεί παραπάνω.

1. Remote Code Execution (RCE)



Εικόνα 30 Στιγμιότυπο Τρωτότητας

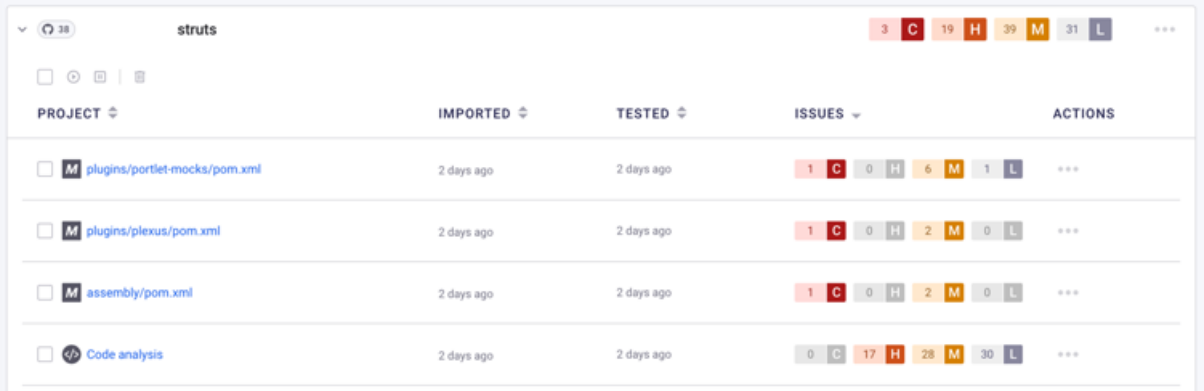
2. Arbitrary Code Execution

Εικόνα 31 Στιγμιότυπο Τρωτότητας

Ομοίως, τα pom αρχεία στους φακέλους client-compiled, compatibility-client-compiled, client-compiler και all, περιέχουν τις 2 παραπάνω ευπάθειες, όπως αυτές καταγράφηκαν.

4.2.2.2.3 Ανάλυση Τρωτοτήτων για το Struts

Για το πλαίσιο Struts ανιχνεύθηκαν συνολικά 3 κρίσιμου επιπέδου θέματα ασφάλειας, 19 υψηλού, 39 μεσαίου και 31 χαμηλού. Παρακάτω θα αναλύσουμε τα 3 κρίσιμα θέματα ασφαλείας που εντοπίστηκαν.

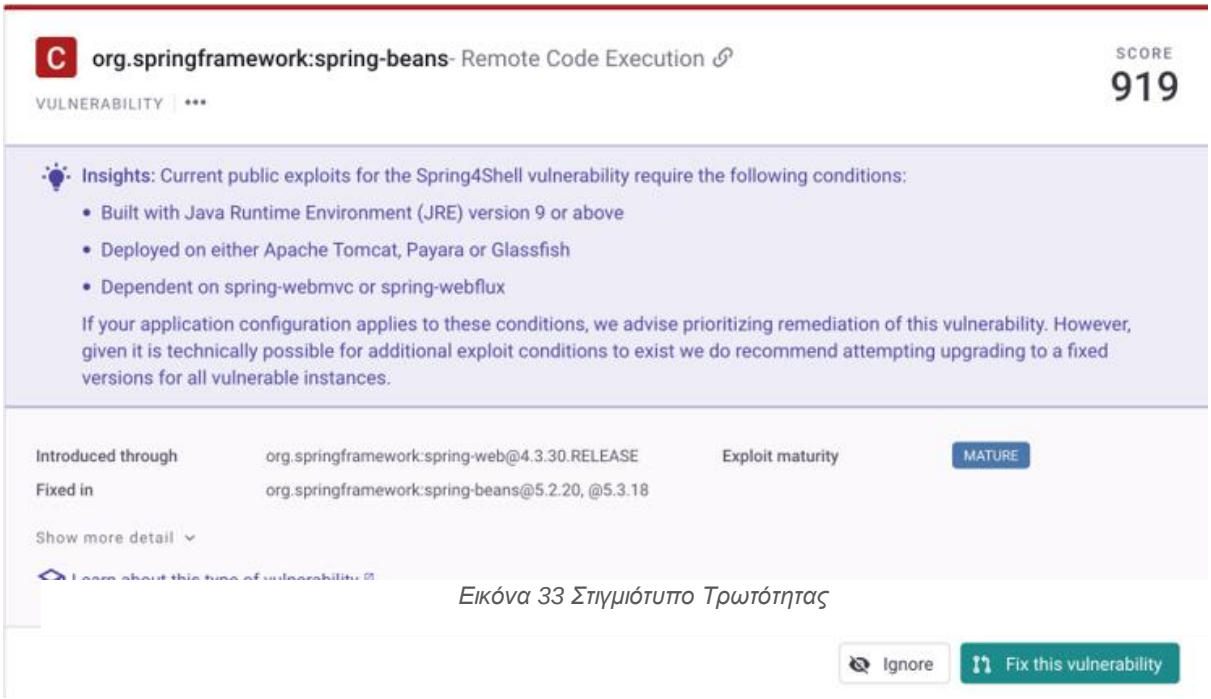


Εικόνα 32 Τρωτότητες του Struts από το Snyk

Οι παρακάτω τρωτότητες έχουν αναλυθεί προηγουμένως και οι τρόποι αντιμετώπισής τους εφαρμόζονται και για το τρέχον πλαίσιο.

1. Remote Code Execution

Ανάλυση Τρωτότητας



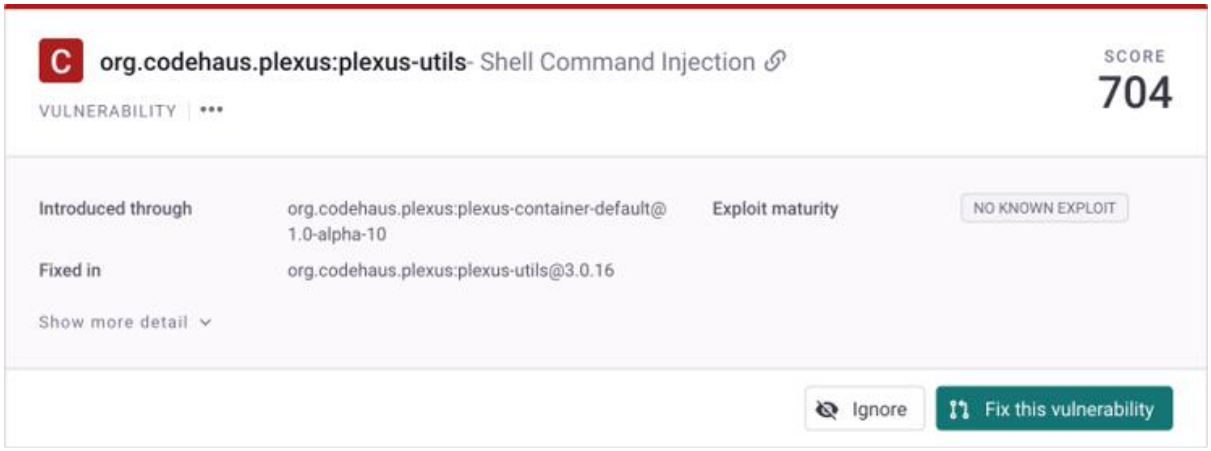
Εικόνα 33 Στιγμιότυπο Τρωτότητας

Η τρωτότητα έχει αναλυθεί προηγουμένως κατά την ανίχνευση του Tapestry με Snyk.

Τρόπος αντιμετώπισης

Έχει αναλυθεί ο τρόπος αντιμετώπισης γι' αυτή την τρωτότητα κατά την ανίχνευση του Tapestry με Snyk.

2. Shell Command Injection



Εικόνα 34 Στιγμιότυπο Τρωτότητας

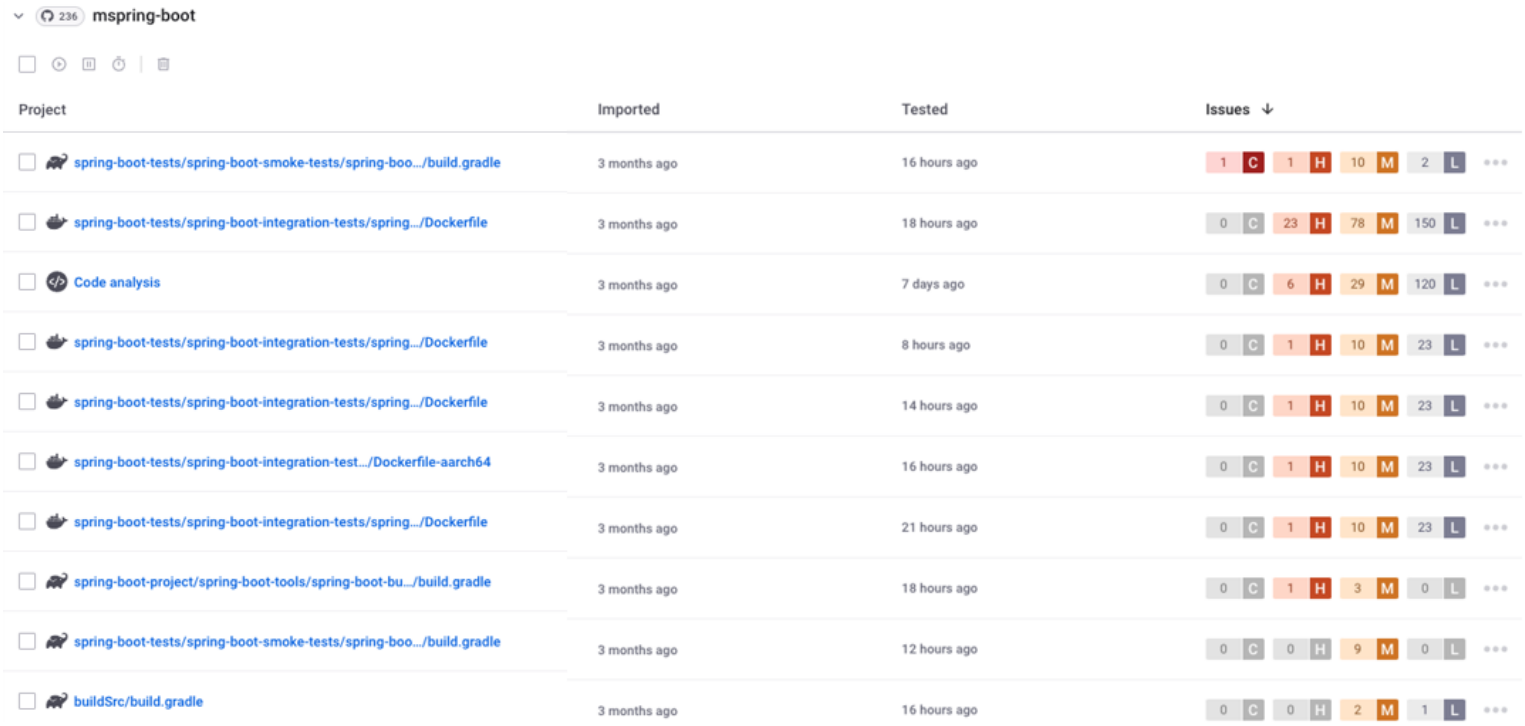
Ανάλυση Τρωτότητας

Η τρωτότητα έχει αναλυθεί προηγουμένως κατά την ανίχνευση του Tapestry με Snyk.

Τρόπος αντιμετώπισης

Έχει αναλυθεί ο τρόπος αντιμετώπισης γι' αυτή την τρωτότητα κατά την ανίχνευση του Tapestry με Snyk.

4.2.2.2.4 Ανάλυση Τρωτοτήτων για το Spring Boot

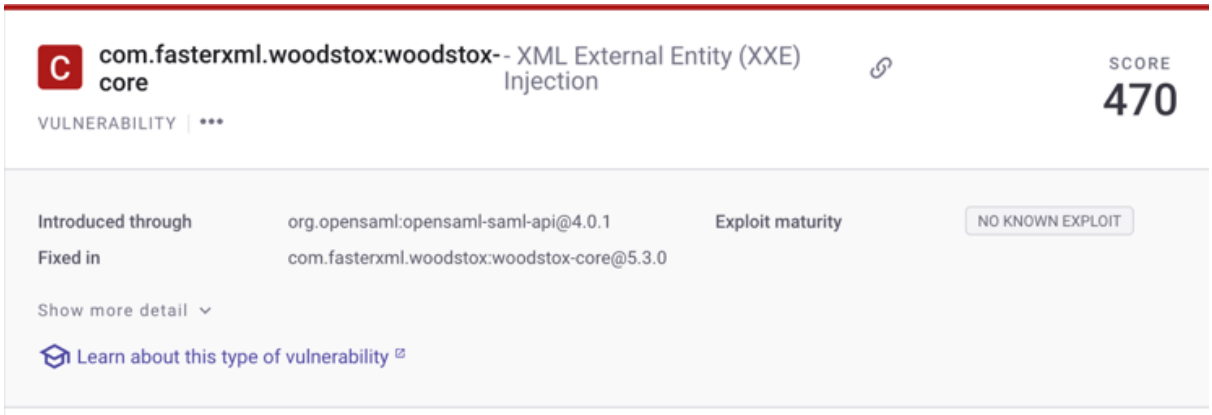


Εικόνα 35 Τρωτότητες Spring Boot από το Snyk

Το Snyk ανίχνευσε για το Spring Boot τα εξής: μόλις μία ευπάθεια κρίσιμου επιπέδου, 45 υψηλού, 131 μεσαίου και 316 χαμηλού επιπέδου. Η ευπάθεια αυτή αναλύεται παρακάτω.

Ανάλυση Τρωτότητας

1. **com.fasterxml.woodstox:woodstox-coreXML: External Entity (XXE) Injection**



Εικόνα 36 Στιγμιότυπο Τρωτότητας

Το Woodstox χρησιμοποιείται ευρέως σε πολλά έργα Java που απαιτούν επεξεργασία XML, ενώ οι εφαρμογές που χρησιμοποιούν το woodstox-core μπορεί να είναι ποικίλες, όπως εφαρμογές web, εφαρμογές επιχειρηματικής λογικής, κ.ά.

Οι επηρεαζόμενες εκδόσεις αυτού του πακέτου είναι ευάλωτες στο XML External Entity (XXE) Injection, λόγω μη ασφαλούς επεξεργασίας και έλλειψης περιορισμού των αρχείων XML. Η XML External Entity (XXE) Injection είναι μια ευπάθεια ασφάλειας που αφορά την επεξεργασία XML δεδομένων. Συμβαίνει όταν η εφαρμογή δέχεται εισόδους XML που περιέχουν εξωτερικές οντότητες (external entities) με αναφορά σε εξωτερικά δεδομένα, αρχεία ή ακόμη και εντολές.Ένας κακόβουλος χρήστης μπορεί να εκμεταλλευτεί αυτή την ευπάθεια στέλνοντας ένα ειδικά δημιουργημένο κακόβουλο αρχείο XML που περιέχει οντότητες XML με URI που επιλύονται σε έγγραφα εκτός της επιδιωκόμενης σφαίρας ελέγχου. Αυτό θα μπορούσε να οδηγήσει σε διάφορα προβλήματα ασφαλείας:

1. **Διαβολή Πόρων (Resource Expansion):** Ο επιτιθέμενος μπορεί να δηλώσει μια εξωτερική οντότητα που αναφέρεται σε ένα εξωτερικό αρχείο, με αποτέλεσμα να φορτωθεί το περιεχόμενο του αρχείου αυτού.
2. **Εκτέλεση Εντολών (Command Execution):** Ο επιτιθέμενος μπορεί να εισάγει εξωτερικές οντότητες που περιέχουν εντολές για εκτέλεση, επιτυγχάνοντας έτσι ανεπιθύμητη εκτέλεση κώδικα στο σύστημα.
3. **Αποκάλυψη Εσωτερικών Δικτύων (Internal Network Disclosure):** Ο επιτιθέμενος μπορεί να προσπαθήσει να ανακαλύψει εσωτερικές διευθύνσεις IP ή άλλες εσωτερικές πληροφορίες μέσω εξωτερικών οντοτήτων.

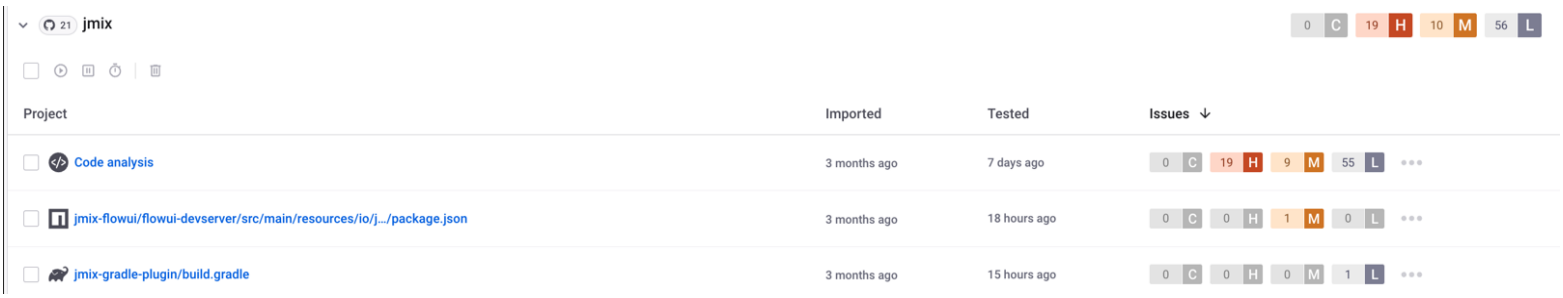
Τρόπος αντιμετώπισης

Προτείνεται η αναβάθμιση του com.fasterxml.woodstox:woodstox-core στην έκδοση 5.3.0 ή σε υψηλότερη.

4.2.2.2.5 Ανάλυση Τρωτοτήτων για το JMix

Το Snyk μετά την ανίχνευση στο πλαίσιο Jmix για ευπάθειες, δεν ανακάλυψε καμία ευπάθεια κρίσιμου επιπέδου, ενώ ανακάλυψε 18 υψηλού, 8 μεσαίου και 35 χαμηλού επιπέδου.

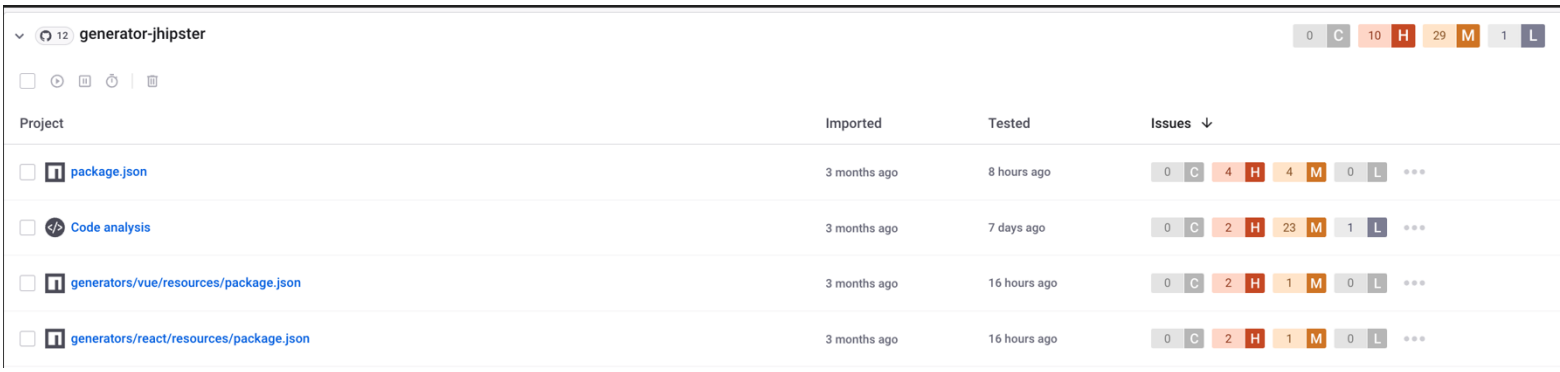
Σημαντικό σε αυτό το σημείο είναι να σημειώσουμε ότι παρόμοια αποτελέσματα είχε δώσει και η ανίχνευση του Codacy.



Εικόνα 37 Αποτελέσματα ανάλυσης του Jmix με Snyk

4.2.2.2.6 Ανάλυση Τρωτοτήτων για το JHipster

Το Snyk μετά την ανίχνευση στο πλαίσιο JHipster για ευπάθειες, δεν ανακάλυψε καμία ευπάθεια κρίσιμου επιπέδου, ενώ ανακάλυψε 4 υψηλού, 26 μεσαίου και 1 χαμηλού επιπέδου.



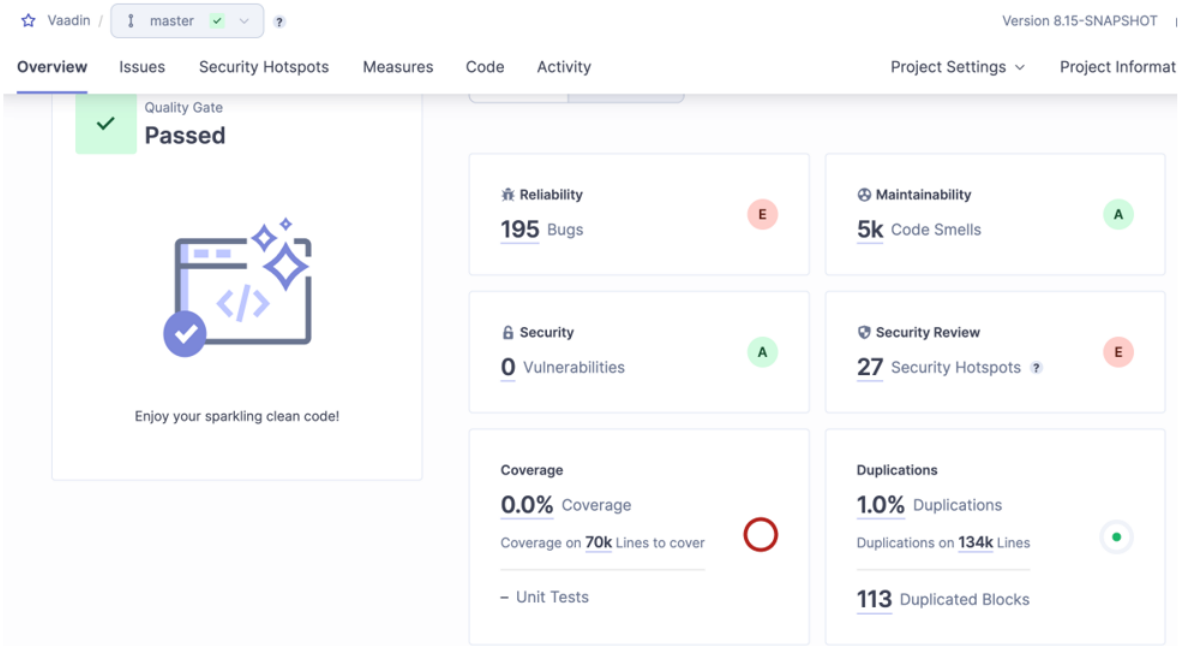
Εικόνα 38 Αποτελέσματα ανάλυσης του JHipster με Snyk

4.2.2.3 Ανάλυση με χρήση του Sonarqube

Στην παρακάτω ενότητα γίνεται ανάλυση των τρωτοτήτων για κάθε πλαίσιο με χρήση του εργαλείου Sonarqube.

4.2.2.3.1 Ανάλυση τρωτοτήτων για το Vaadin

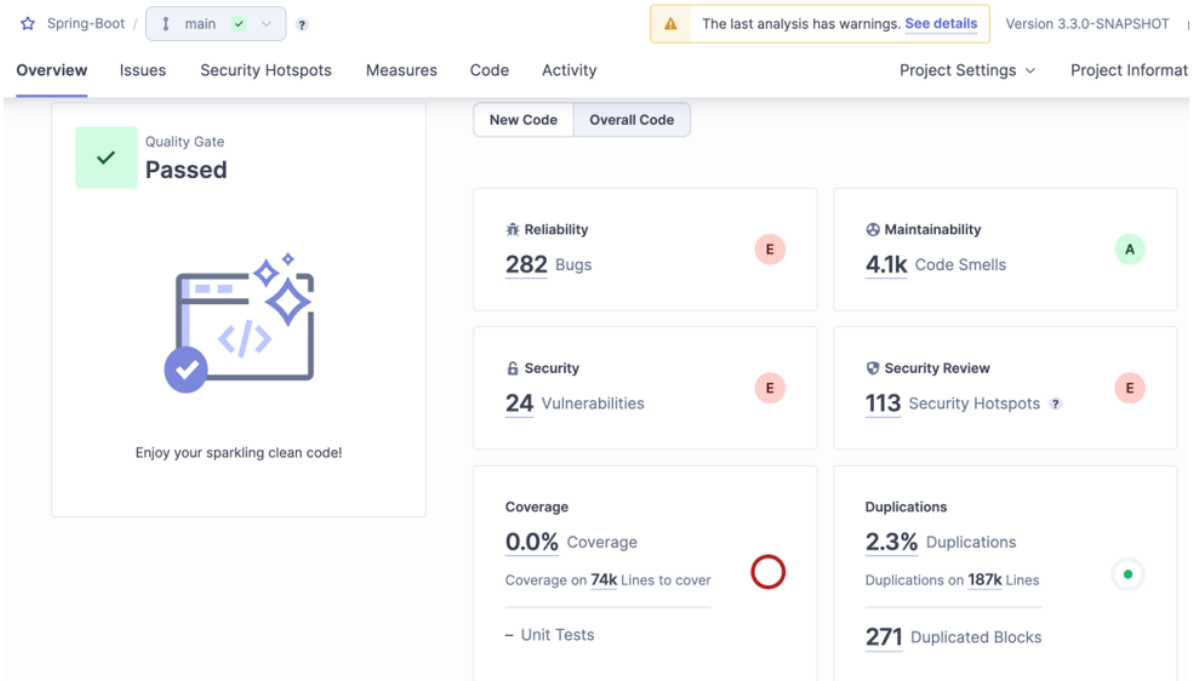
Μετά την επιτυχή σάρωση του Vaadin, η τελευταία έκδοση έδειξε μηδενικό αριθμό ευπαθειών, χαρακτηρίζοντας και το επίπεδο ασφάλειας του πλαισίου με Α. Επίσης ανιχνεύθηκαν 27 security reviews, τα οποία δεν κρίνονται σοβαρές ευπάθειες, ωστόσο απαιτούν την επίβλεψη των μηχανικών ασφαλείας για να αποφασιστεί εάν χρειάζονται βελτίωση ή όχι.



Εικόνα 39 Ευπάθειες του Vaadin από το SonarQube

4.2.2.3.2 Ανάλυση τρωτοτήτων για το Spring-Boot

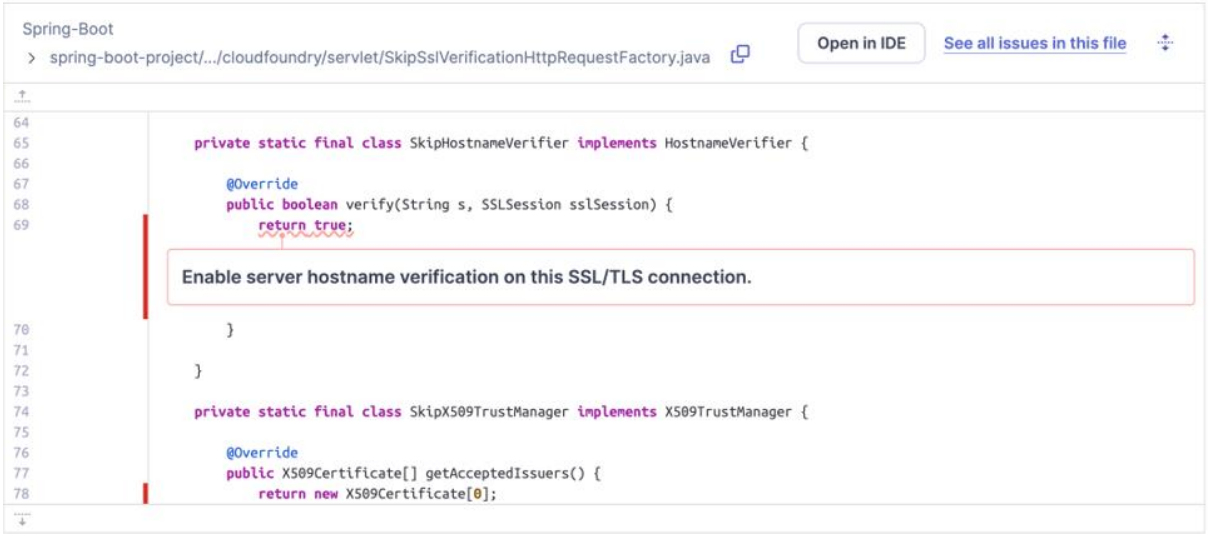
Για το spring-boot οι ευπάθειες που βρέθηκαν είναι 24, ενώ έχουν βρεθεί και 113 security reviews. Το SonarQube αξιολογεί τις ευπάθειες του Spring-Boot με E, καθώς υπάρχουν ευπάθειες που έχουν μεγάλη πιθανότητα να επηρεάσουν τη συμπεριφορά της εφαρμογής στην παραγωγή.



Εικόνα 40 Ευπάθειες του Spring-Boot από το SonarQube

Ανάλυση Τρωτότητας

1. **Enable server hostname verification on this SSL/TLS connection** (Έλλειψη επικύρωσης hostname απομακρυσμένου εξυπηρετητή)



Εικόνα 41 Έλλειψη επικύρωσης hostname απομακρυσμένου εξυπηρετητή

Η πρώτη ευπάθεια αφορά την έλλειψη επαλήθευσης του hostname του εξυπηρετητή στον οποίο εκτελείται σύνδεση. Πιο συγκεκριμένα, το Transport Layer Security (TLS) παρέχει ασφαλή επικοινωνία μεταξύ συστημάτων μέσω Διαδικτύου κρυπτογραφώντας τα δεδομένα που αποστέλλονται μεταξύ τους. Σε αυτή τη διαδικασία, ο ρόλος της επικύρωσης του hostname ενός απομακρυσμένου εξυπηρετητή, σε συνδυασμό με την επικύρωση πιστοποιητικού, αφορά την διασφάλιση ότι ένα σύστημα είναι πράγματι αυτό που ισχυρίζεται ότι είναι, προσθέτοντας ένα επιπλέον επίπεδο εμπιστοσύνης και ασφάλειας.

Όταν η επικύρωση του hostname είναι απενεργοποιημένη, ο πελάτης παραλείπει αυτόν τον κρίσιμο έλεγχο. Αυτό δημιουργεί μια ευκαιρία για τους εισβολείς να παρουσιαστούν ως αξιόπιστη οντότητα και να υποκλέψουν, να χειραγωγήσουν ή να κλέψουν τα δεδομένα που μεταδίδονται. Για να γίνει αυτό, ένας εισβολέας θα αποκτήσει ένα έγκυρο πιστοποιητικό με έλεγχο ταυτότητας example.com, θα το εξυπηρετήσει χρησιμοποιώντας ένα διαφορετικό όνομα κεντρικού υπολογιστή και ο κωδικός της εφαρμογής θα εξακολουθεί να το δέχεται.

Η εδραίωση εμπιστοσύνης με ασφαλή τρόπο είναι ένα μη τετριμμένο έργο. Όταν απενεργοποιείται η επικύρωση του hostname, καταργείται ένας βασικός μηχανισμός που έχει σχεδιαστεί για την οικοδόμηση αυτής της εμπιστοσύνης στην επικοινωνία μέσω Διαδικτύου, ανοίγοντας το σύστημά σας σε έναν αριθμό πιθανών απειλών.

Εάν ένα σύστημα δεν επικυρώνει ονόματα κεντρικών υπολογιστών, δεν μπορεί να επιβεβαιώσει την ταυτότητα του άλλου μέρους που εμπλέκεται στην επικοινωνία. Ένας εισβολέας μπορεί να το εκμεταλλευτεί αυτό δημιουργώντας έναν ψεύτικο διακομιστή και μεταμφιέζοντας τον ως νόμιμο. Για παράδειγμα, μπορεί να δημιουργήσει έναν διακομιστή που μοιάζει με τον διακομιστή της τράπεζάς ενός ανυποψίαστου χρήστη, ξεγελώντας το σύστημά του ώστε να πιστεύει ότι επικοινωνεί με την τράπεζα. Αυτό το σενάριο, που ονομάζεται πλαστογράφηση ταυτότητας, επιτρέπει στον εισβολέα να συλλέγει τυχόν δεδομένα που του στέλνει το σύστημα του θύματος, οδηγώντας ενδεχομένως σε σημαντικές παραβιάσεις δεδομένων.

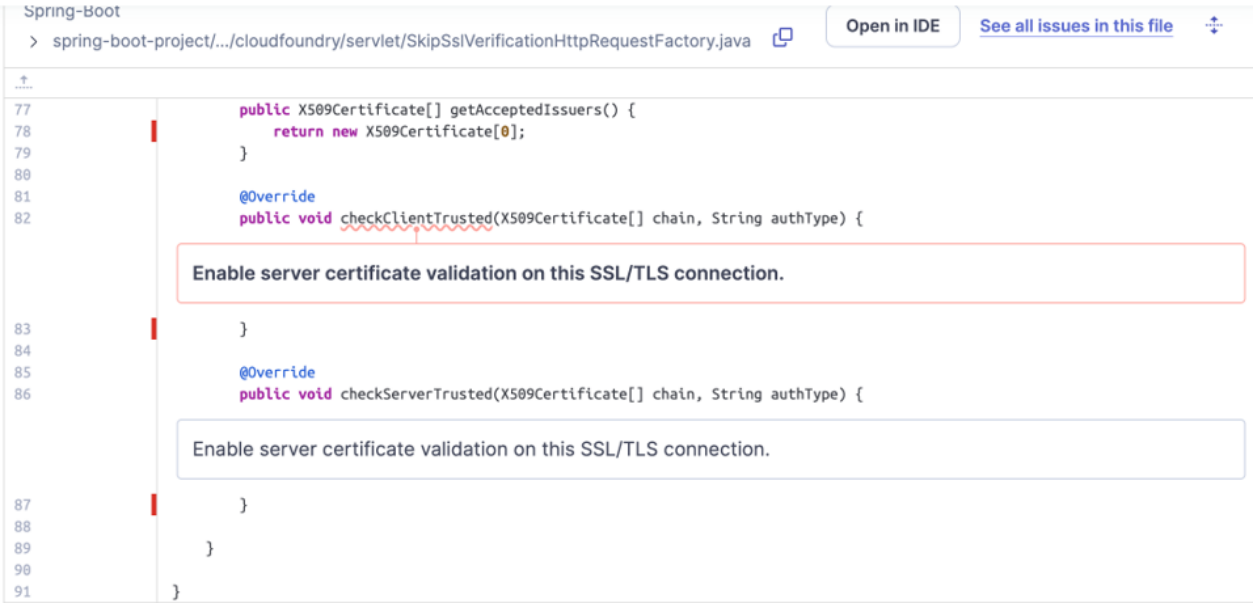
Ο παραπάνω κώδικας *(Εικόνα 42)* ορίζει ένα προσαρμοσμένο HostnameVerifier που επιστρέφει πάντα true, απενεργοποιώντας ουσιαστικά την επαλήθευση hostname για συνδέσεις SSL/TLS. Αυτό δημιουργεί σημαντικό κίνδυνο ασφάλειας, επειδή επιτρέπει στον πελάτη να δέχεται πιστοποιητικά για οποιοδήποτε όνομα κεντρικού υπολογιστή, ανεξάρτητα από την εγκυρότητά του ή αν ταιριάζει με τον προβλεπόμενο διακομιστή. Αυτή την ευπάθεια θα μπορούσε να εκμεταλλευτεί κάποιος εισβολέας για επιθέσεις man-in-the-middle (MitM) ή άλλες κακόβουλες δραστηριότητες.

Τρόπος αντιμετώπισης

Για να μην παραβιάζεται η ασφάλεια του πλαισίου, η συνάρτηση verify δεν θα έπρεπε να επιστρέφει πάντα true, αλλά το αποτέλεσμα της επαλήθευσης του hostname.

2. **Enable server certificate validation on this SSL/TLS connection** (Παράκαμψη επικύρωσης πιστοποιητικού του εξυπηρετητή)

Ανάλυση Τρωτότητας



Εικόνα 42 Παράκαμψη επικύρωσης πιστοποιητικού του εξυπηρετητή

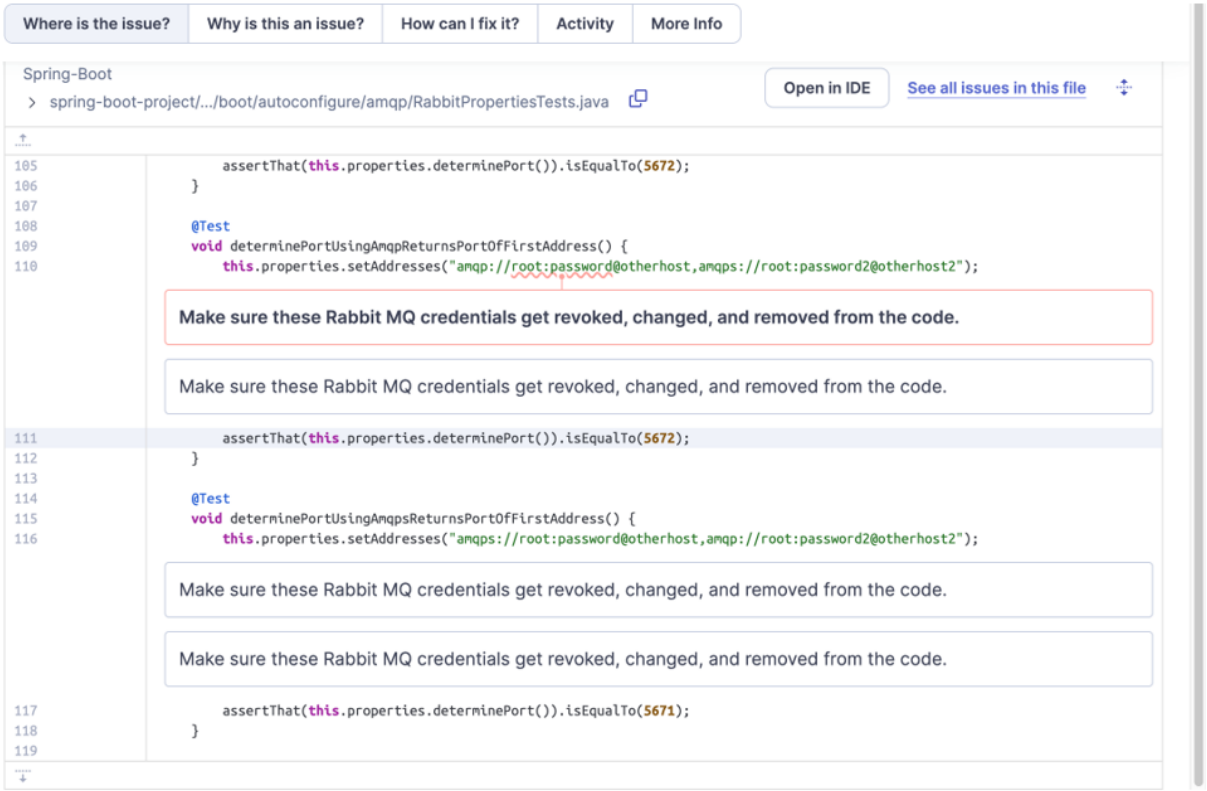
Η δεύτερη ευπάθεια που εμφανίζεται είναι αντίστοιχη με την πρώτη ευπάθεια, με την διαφορά ότι στο παρόν παράδειγμα ο κώδικας παρακάμπτει την επικύρωση του πιστοποιητικού. Επειδή τα αρχεία που εμφανίζονται ως ευπαθή αφορούν αρχεία ελέγχων, είναι λογικά σκόπιμη η χρήση αυτών των μεθόδων. Για παράδειγμα, όπως φαίνεται στην Εικόνα 35, το όνομα του αρχείου προδίδει την πρόθεση των προγραμματιστών να παρακάμψουν την επικύρωση του πιστοποιητικού κατά την εκτέλεση αιτημάτων HTTP.

Όταν η επικύρωση πιστοποιητικού TLS είναι απενεργοποιημένη, η ακεραιότητα των δεδομένων που στέλνονται και λαμβάνονται δεν είναι εγγυημένη. Ένας εισβολέας θα μπορούσε να τροποποιήσει τα δεδομένα κατά τη μεταφορά και να μη γίνει αντιληπτός. Αυτό μπορεί να κυμαίνεται από ανεπαίσθητους χειρισμούς των δεδομένων που λαμβάνονται μέχρι την έγχυση κακόβουλου κώδικα ή κακόβουλου λογισμικού στο σύστημά. Οι συνέπειες τέτοιων παραβιάσεων της ακεραιότητας των δεδομένων μπορεί να είναι σοβαρές, ανάλογα με τη φύση των δεδομένων και του συστήματος.

Τρόπος αντιμετώπισης

Χρήση αξιόπιστων πιστοποιητικών, που έχουν εκδοθεί από μια γνωστή, αξιόπιστη CA για το διακομιστή που χρησιμοποιείται. Τα περισσότερα περιβάλλοντα προγραμματισμού διαθέτουν έναν προκαθορισμένο κατάλογο αξιόπιστων root CA και τα πιστοποιητικά που εκδίδονται από αυτές τις αρχές επικυρώνονται αυτόματα. Αυτή είναι η καλύτερη πρακτική και δεν απαιτεί πρόσθετο κώδικα ή διαμόρφωση.

3. **Make sure these Rabbit MQ credentials get revoked, changed, and removed from the code** (Διαρροή διαπιστευτηρίων για τον RabbitMQ μέσω του πηγαίου κώδικα)



Εικόνα 43 Διαρροή διαπιστευτηρίων για τον RabbitMQ μέσω του πηγαίου κώδικα

Ανάλυση Τρωτότητας

Η επόμενη ευπάθεια αφορά κάποια default στοιχεία πρόσβασης που εμφανίζονται σε αρχεία ελέγχου και αφορούν την πρόσβαση στον RabbitMQ εξυπηρετητή. Εάν η συγκεκριμένη υπηρεσία στο πλαίσιο χρησιμοποιείται για την αποθήκευση ή την επεξεργασία στοιχείων προσωπικής ταυτοποίησης ή άλλων ευαίσθητων δεδομένων, οι εισβολείς που γνωρίζουν ένα μυστικό ελέγχου ταυτότητας θα μπορούσαν να αποκτήσουν εύκολα πρόσβαση σε αυτό.

Ανάλογα με τον τύπο των δεδομένων που παραβιάζονται, θα μπορούσε να οδηγήσει σε παραβιάσεις του απορρήτου, κλοπή ταυτότητας, οικονομική απώλεια ή άλλα αρνητικά αποτελέσματα. Στις περισσότερες περιπτώσεις, μια εταιρεία που υποφέρει από παραβίαση ευαίσθητων δεδομένων θα αντιμετωπίσει απώλεια φήμης όταν δημοσιοποιηθεί το ζήτημα ασφάλειας.

Τρόπος αντιμετώπισης

Για την ασφαλή διαχείριση των διαπιστευτηρίων και για την απόκρυψη τους από τον πηγαίο κώδικα, οι κωδικοί θα έπρεπε να βρίσκονται αποθηκευμένοι σε κάποιο ασφαλές αποθετήριο (πχ. Vault) το οποίο θα κρατάει τους κωδικούς κρυπτογραφημένους. Οι κωδικοί αυτοί στην συνέχεια θα μπορούσαν να χρησιμοποιηθούν μέσω των μεταβλητών συστήματος. Έτσι, για παράδειγμα, η παραπάνω εντολή θα μπορούσε να μετατραπεί ως εξής:

```
this.properties.setAddresses("System.getenv("amqps://"+
System.getenv("AMQP_CREDENTIALS")+"@example.com:8080/example");
```

Εικόνα 44 Παράδειγμα σωστής εντολής (αντιπαραβολή με Εικόνα 43)

4. **Make sure these Redis credentials get revoked, changed, and removed from the code** (Διαρροή διαπιστευτηρίων για τον Redis μέσω του πηγαίου κώδικα)

```
115         this.contextRunner
116             .withPropertyValues("spring.data.redis.host:foo", "spring.data.redis.password:xyz",
117                     "spring.data.redis.port:1000", "spring.data.redis.ssl.enabled:false",
118                     "spring.data.redis.url:rediss://user:password@example:33")
```

Make sure these Redis credentials get revoked, changed, and removed from the code.

```
119             .run((context) -> {
120                 JedisConnectionFactory cf = context.getBean(JedisConnectionFactory.class);
121                 assertThat(cf.getHostName()).isEqualTo("example");
122                 assertThat(cf.getPort()).isEqualTo(33);
123                 assertThat(getUserName(cf)).isEqualTo("user");
124                 assertThat(cf.getPassword()).isEqualTo("password");
125                 assertThat(cf.isUseSsl()).isTrue();
126             });
127     }
```

Εικόνα 45 Διαρροή διαπιστευτηρίων για τον εξυπηρετητή Redis

Ανάλυση Τρωτότητας

Σε παρόμοια κλίμακα, μια άλλη ευπάθεια αφορά την χρήση διαπιστευτηρίων με σκληρή κωδικοποίηση (hardcoded) στον κώδικα, και πιο συγκεκριμένα τα διαπιστευτήρια που αφορούν την σύνδεση στον εξυπηρετητή Redis, όπως παρουσιάζεται και στην Εικόνα 46.

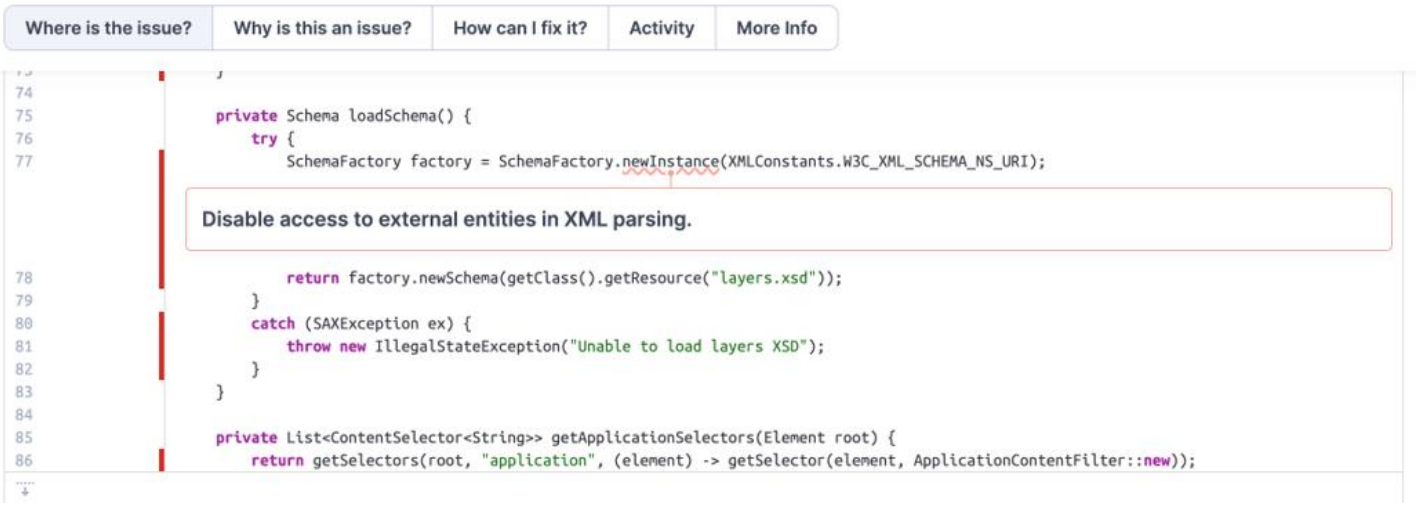
Τρόπος αντιμετώπισης

Αντίστοιχα, η παραπάνω ευπάθεια μπορεί εξαλειφθεί με την χρήση μεταβλητών συστήματος.

```
this.contextRunner.withPropertyValues("spring.data.redis.host:foo",
"spring.data.redis.password:xyz", "spring.data.redis.port:1000",
"spring.data.redis.ssl.enabled:false", System.getenv("REDIS_URL"));
```

Εικόνα 46 Παράδειγμα χρήσης μεταβλητών συστήματος

5. **Disable access to external entities in XML Parsing** (Απενεργοποίηση πρόσβασης σε εξωτερικές οντότητες της XML)



Εικόνα 47 Απενεργοποίηση πρόσβασης σε εξωτερικές οντότητες της XML

Ανάλυση Τρωτότητας

Η τελευταία ευπάθεια του Spring-Boot, αφορά την απενεργοποίηση της πρόσβασης σε εξωτερικές οντότητες στην ανάλυση XML. Η επεξεργασία εξωτερικής οντότητας επιτρέπει την ανάλυση XML με τη συμμετοχή εξωτερικών οντοτήτων. Ωστόσο, όταν αυτή η λειτουργία είναι ενεργοποιημένη χωρίς τις κατάλληλες προφυλάξεις, μπορεί να οδηγήσει σε μια ευπάθεια γνωστή ως επίθεση XML External Entity (XXE).

Ένας σημαντικός κίνδυνος των τρωτών σημείων XXE είναι η πιθανότητα έκθεσης σε ευαίσθητα δεδομένα. Δημιουργώντας κακόβουλα ωφέλιμα φορτία XML, οι εισβολείς μπορούν να αναφέρουν εξωτερικές οντότητες που περιέχουν ευαίσθητες πληροφορίες, όπως αρχεία συστήματος, διαπιστευτήρια βάσης δεδομένων ή αρχεία διαμόρφωσης. Όταν αυτές οι οντότητες υποβάλλονται σε επεξεργασία κατά την ανάλυση XML, ο εισβολέας μπορεί να εξάγει τα περιεχόμενα και να αποκτήσει μη εξουσιοδοτημένη πρόσβαση σε ευαίσθητα δεδομένα. Αυτό αποτελεί σοβαρή απειλή για το απόρρητο των κρίσιμων πληροφοριών.

Τα τρωτά σημεία XXE μπορούν επίσης να ενεργοποιήσουν επιθέσεις πλαστογράφησης αιτημάτων από την πλευρά του διακομιστή (SSRF). Αξιοποιώντας την ικανότητα να περιλαμβάνει εξωτερικές οντότητες, ένας εισβολέας μπορεί να κάνει την ευάλωτη εφαρμογή να στέλνει αυθαίρετα αιτήματα σε άλλα εσωτερικά ή εξωτερικά συστήματα. Αυτό μπορεί να οδηγήσει σε ακούσιες ενέργειες, όπως η ανάκτηση δεδομένων από εσωτερικούς πόρους, η σάρωση εσωτερικών δικτύων ή η επίθεση σε άλλα συστήματα. Οι επιθέσεις SSRF μπορούν να οδηγήσουν σε σοβαρές συνέπειες, συμπεριλαμβανομένης της μη εξουσιοδοτημένης πρόσβασης σε δεδομένα, παραβίασης του συστήματος ή ακόμη και περαιτέρω εκμετάλλευσης εντός της υποδομής του δικτύου.

Τρόπος Αντιμετώπισης

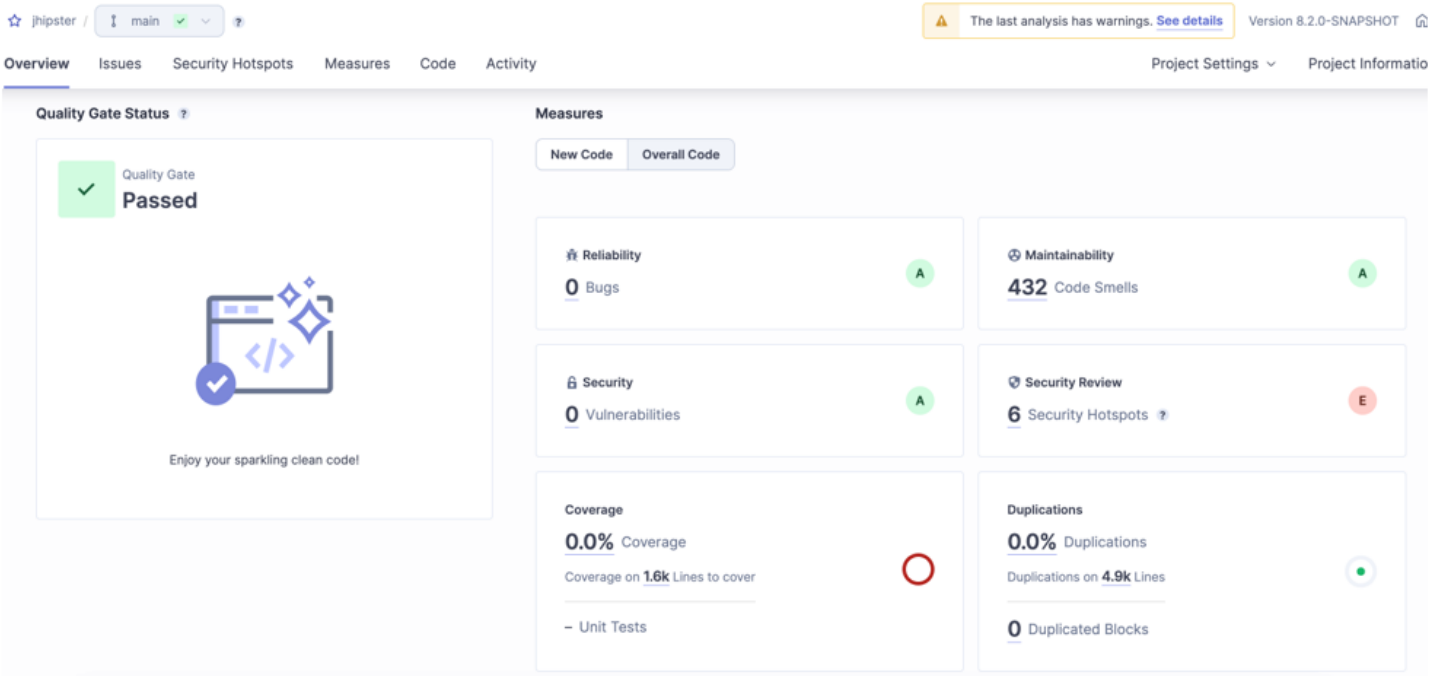
Η πιο αποτελεσματική προσέγγιση για την αποφυγή ευπάθειας XXE είναι η πλήρης απενεργοποίηση της επεξεργασίας εξωτερικής οντότητας, εκτός εάν απαιτείται ρητά για συγκεκριμένες περιπτώσεις χρήσης. Από προεπιλογή, οι αναλυτές XML θα πρέπει να ρυθμιστούν ώστε να απορρίπτουν την επεξεργασία εξωτερικών οντοτήτων. Αυτό μπορεί να επιτευχθεί ορίζοντας τις κατάλληλες ιδιότητες ή επιλογές στη βιβλιοθήκη ή το πλαίσιο ανάλυσης XML, όπως εμφανίζεται στον παρακάτω κώδικα.

```
import org.dom4j.io.SAXReader;

public void decode() {

   SAXReader xmlReader = new SAXReader();

   xmlReader.setFeature("http://apache.org/xml/features/disallow-doctype-decl", true);

}
```

Εικόνα 48 Κώδικας απενεργοποίησης επεξεργασίας εξωτερικής οντότητας

4.2.2.3.3 Ανάλυση τρωτοτήτων για το JHipster

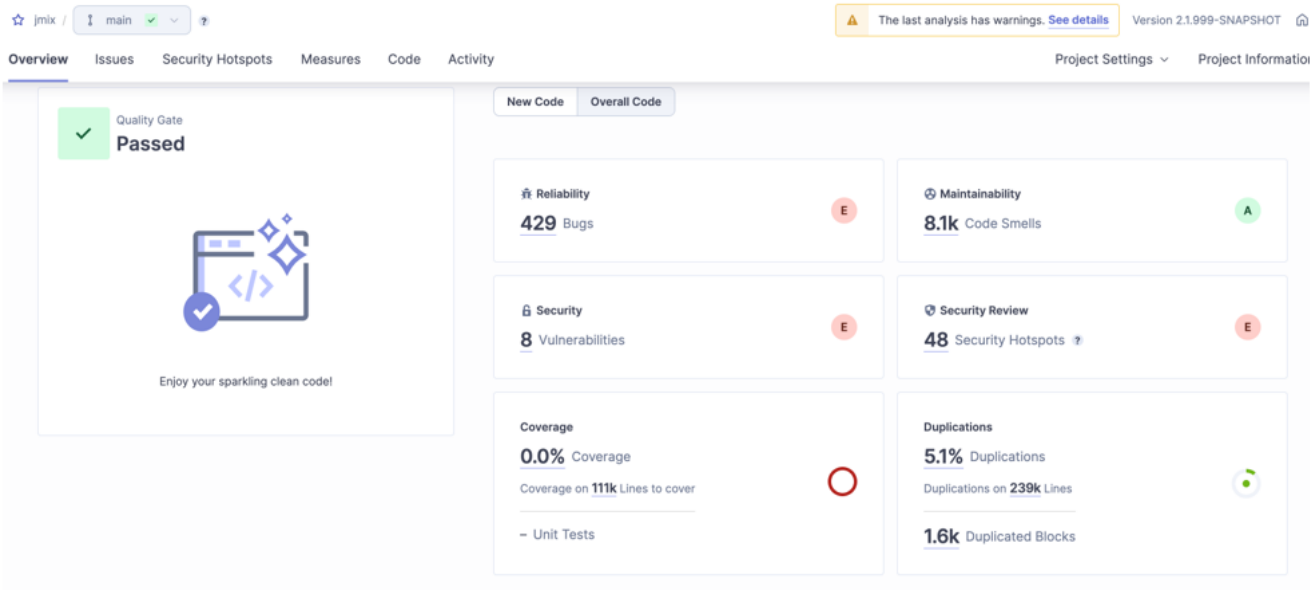
Το JHipster μετά την σάρωσή του από το SonarQube βρέθηκε και αυτό με μηδενικό αριθμό ευπαθειών και με μονοψήφιο αριθμό από security reviews. Για τον λόγο αυτό αξιολογείται με Α από το SonarQube.



Εικόνα 49 Ευπάθειες του JHipster από το sonarqube

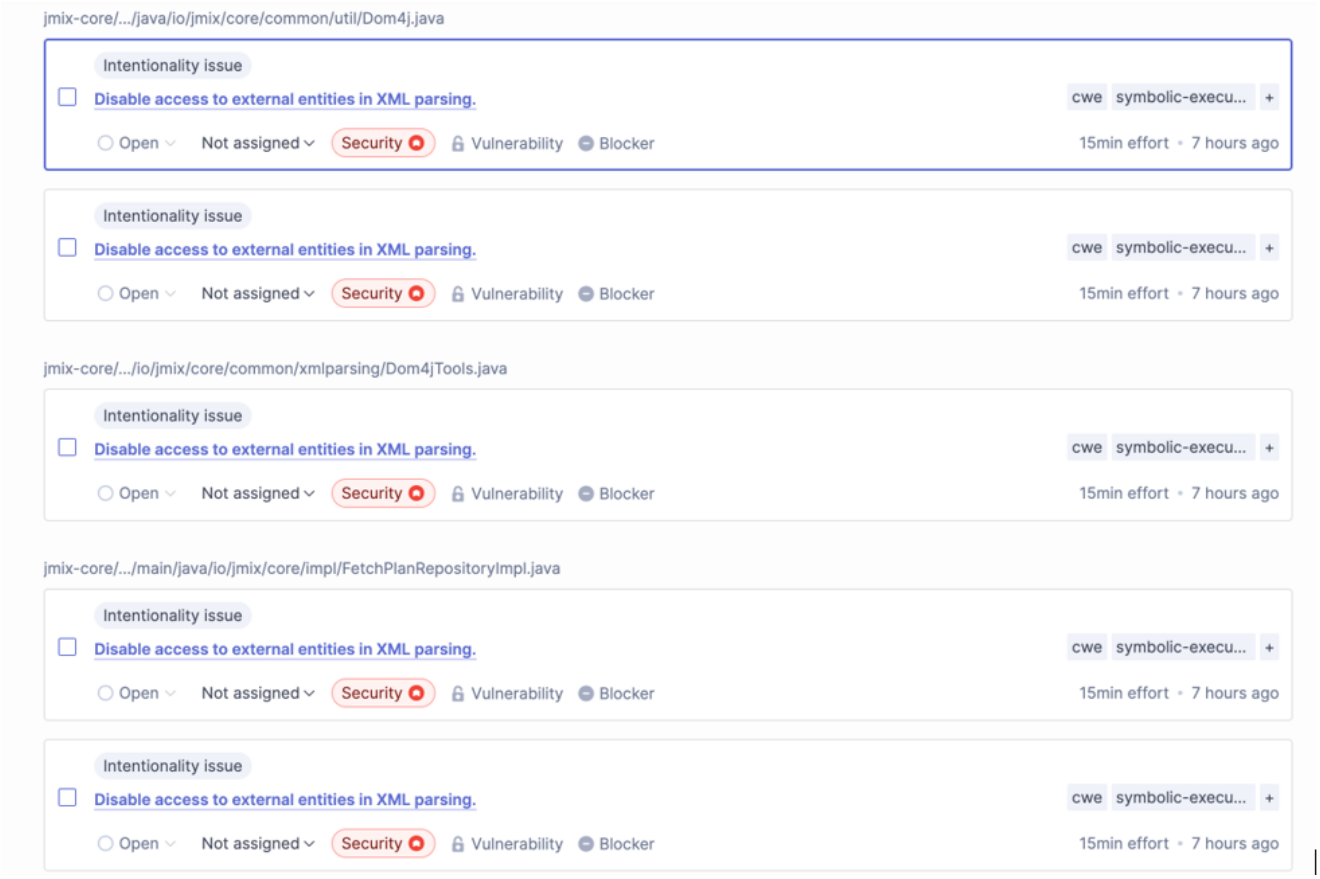
4.2.2.3.4 Ανάλυση τρωτοτήτων για το JMix

Το επόμενο πλαίσιο που αναλύθηκε είναι το JMix το οποίο μετά την σάρωσή του από το SonarQube εμφάνισε 8 ευπάθειες και 48 Security Reviews. Οι 8 ευπάθειες αυτές ωστόσο έχουν βαθμολογηθεί με E καθώς κάποιες από αυτές κρίνονται αρκετά κρίσιμες.



Εικόνα 50 Ευπάθειες του JMix από το Sonarqube

1. **Disable access to external entities in XML Parsing** (Απενεργοποίηση πρόσβασης σε εξωτερικές οντότητες XML.)



Εικόνα 51 Απενεργοποίηση πρόσβασης σε εξωτερικές οντότητες της XML.

Ανάλυση Τρωτότητας

Όπως φαίνεται και στην *Εικόνα 51*, οι πρώτες πέντε ευπάθειες αφορούν την πρόσβαση εξωτερικών XML αρχείων κατά την ανάλυσή τους. Η τρωτότητα αυτή έχει αναλυθεί νωρίτερα για το Spring - Boot.

Τρόπος αντιμετώπισης

Ο τρόπος αντιμετώπισης αυτής της τρωτότητας αυτή έχει αναλυθεί νωρίτερα για το Spring - Boot.

2. **Use a dynamically - generated, random IV** (Χρήση μη ασφαλούς διανύσματος αρχικοποίησης)

Εικόνα 52 Χρήση μη ασφαλούς IV

Ανάλυση Τρωτότητας

Η τελευταία ευπάθεια αφορά την δημιουργία ενός τυχαίου διανύσματος αρχικοποίησης (Initialization Vector - IV) για την χρήση στους αλγόριθμους κρυπτογράφησης. Είναι γνωστό πως εάν το ίδιο IV χρησιμοποιείται για πολλαπλές περιόδους σύνδεσης ή μηνύματα κρυπτογράφησης, κάθε νέα κρυπτογράφηση της ίδιας εισόδου απλού κειμένου θα παράγει πάντα την ίδια έξοδο κρυπτογραφημένου κειμένου. Αυτό μπορεί να επιτρέψει σε έναν εισβολέα να ανιχνεύσει μοτίβα στο κρυπτογραφημένο κείμενο.

Μετά την ανάκτηση κρυπτογραφημένων δεδομένων και την εκτέλεση κρυπτογραφικών επιθέσεων σε αυτά σε ένα δεδομένο χρονικό πλαίσιο, οι εισβολείς μπορούν να ανακτήσουν το απλό κείμενο που υποτίθεται ότι προστατεύει η κρυπτογράφηση. Ανάλογα με τα ανακτημένα δεδομένα, ο αντίκτυπος μπορεί να διαφέρει. Παρακάτω είναι μερικά σενάρια πραγματικού κόσμου που απεικονίζουν τον πιθανό αντίκτυπο ενός εισβολέα που εκμεταλλεύεται την ευπάθεια.

- Πρόσθετη επιφάνεια επίθεσης

Τροποποιώντας το απλό κείμενο του κρυπτογραφημένου μηνύματος, ένας εισβολέας μπορεί να είναι σε θέση να ενεργοποιήσει πρόσθετες ευπάθειες στον κώδικα. Ένας εισβολέας μπορεί να εκμεταλλευτεί περαιτέρω ένα σύστημα για να λάβει περισσότερες πληροφορίες. Οι κρυπτογραφημένες τιμές θεωρούνται συχνά αξιόπιστες επειδή δεν θα ήταν δυνατό για κάποιον τρίτο να τις τροποποιήσει υπό κανονικές συνθήκες.

- Παραβίαση του απορρήτου και του απορρήτου

Όταν τα κρυπτογραφημένα δεδομένα περιέχουν προσωπικές ή ευαίσθητες πληροφορίες, η ανάκτησή τους από έναν εισβολέα μπορεί να οδηγήσει σε παραβιάσεις του απορρήτου, κλοπή

ταυτότητας, οικονομική απώλεια, ζημιά στη φήμη ή μη εξουσιοδοτημένη πρόσβαση σε εμπιστευτικά συστήματα. Σε αυτό το σενάριο, μια εταιρεία, οι εργαζόμενοι, οι χρήστες και οι συνεργάτες της θα μπορούσαν να επηρεαστούν σοβαρά. Ο αντίκτυπος είναι διπλός, καθώς οι παραβιάσεις δεδομένων και η έκθεση κρυπτογραφημένων δεδομένων μπορεί να υπονομεύσει την εμπιστοσύνη στον οργανισμό, καθώς οι πελάτες και οι ενδιαφερόμενοι μπορεί να χάσουν την εμπιστοσύνη τους στην ικανότητα του οργανισμού να προστατεύει τα ευαίσθητα δεδομένα τους.

- Νομικά θέματα και θέματα συμμόρφωσης

Σε πολλές βιομηχανίες και τοποθεσίες, υπάρχουν νομικές απαιτήσεις και απαιτήσεις συμμόρφωσης για την προστασία ευαίσθητων δεδομένων. Εάν τα κρυπτογραφημένα δεδομένα παραβιαστούν και το απλό κείμενο μπορεί να ανακτηθεί, οι εταιρείες αντιμετωπίζουν νομικές συνέπειες, κυρώσεις ή παραβιάσεις της νομοθεσίας περί απορρήτου.

Τρόπος αντιμετώπισης

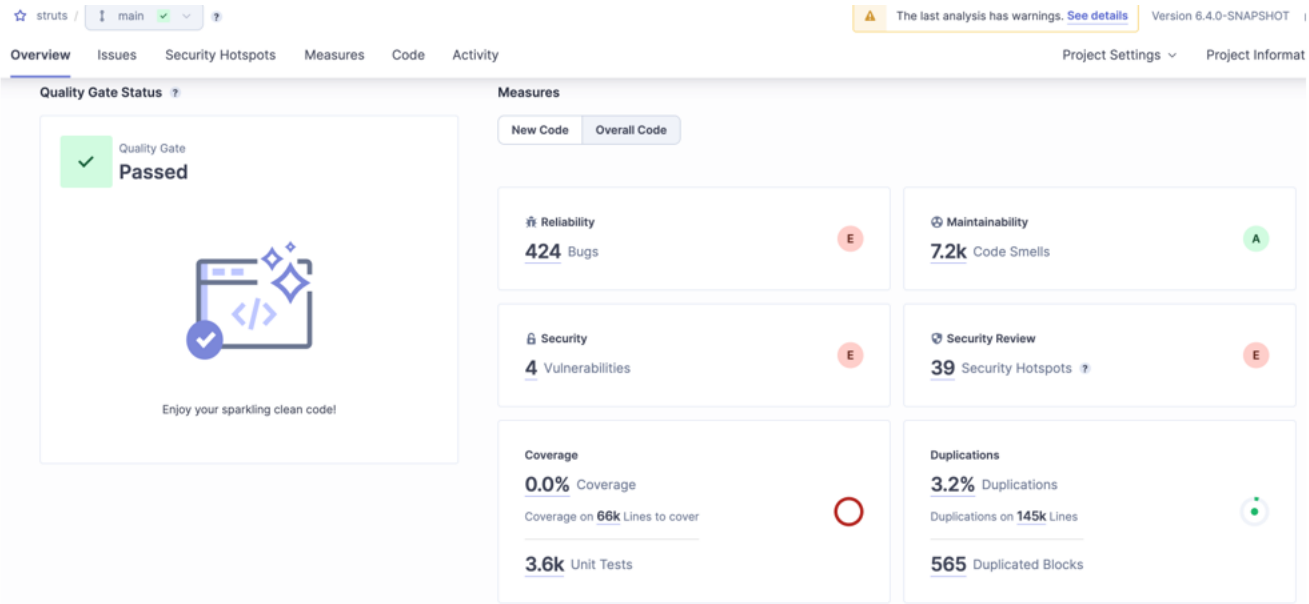
Παρακάτω παρουσιάζεται ένα παράδειγμα για το πως θα ήταν η σωστότερη δημιουργία ενός IV για την αποφυγή των παραπάνω αντίκτυπων.

```
SecureRandom random = new SecureRandom ();

byte [] randomBytes = νέο byte [ 16 ];

random.nextBytes(randomBytes);

GCMParameterSpec iv = new GCMParameterSpec ( 128 , randomBytes);

SecretKeySpec keySpec= new SecretKeySpec(key.getBytes(StandardCharsets.UTF_8),"AES" );
```

Εικόνα 53 Σωστή δημιουργία IV

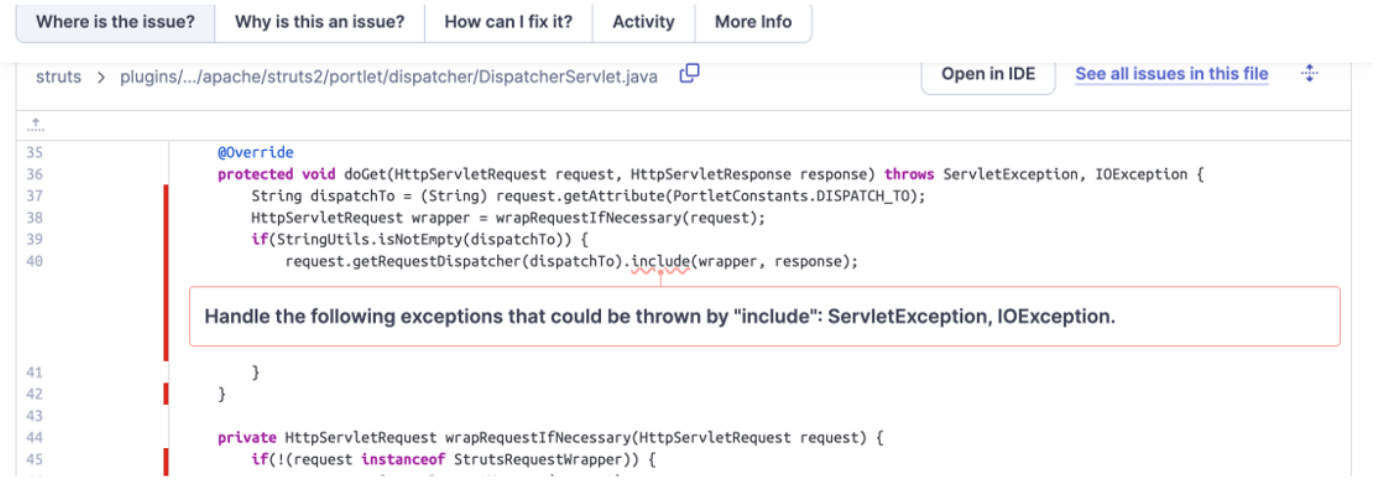
4.2.2.3.5 Ανάλυση τρωτοτήτων για το Struts

Από την ανάλυση του Struts προέκυψαν μόλις 4 ευπάθειες, ωστόσο και αυτές έχουν βαθμολογηθεί με Ε από το SonarQube.



Εικόνα 54 Ανάλυση του Struts από το Sonarqube

1. **Handle the following exceptions that could be thrown by "include":ServletException, IOException.** (Διαχείριση εξαιρέσεων από την συνάρτηση include)



Εικόνα 55 Διαχείριση εξαιρέσεων από την συνάρτηση include

Ανάλυση Τρωτότητας

Η πρώτη ευπάθεια αφορά εξαιρέσεις που θα έπρεπε ο κώδικας να διαχειρίζεται, και ειδικότερα την ServletException και IOException. Τα Servlets είναι στοιχεία στην ανάπτυξη ιστού Java, υπεύθυνα για την επεξεργασία αιτημάτων HTTP και τη δημιουργία απαντήσεων. Σε αυτό το πλαίσιο, οι εξαιρέσεις χρησιμοποιούνται για το χειρισμό και τη διαχείριση απροσδόκητων σφαλμάτων ή εξαιρετικών συνθηκών που μπορεί να προκύψουν κατά την εκτέλεση ενός servlet. Η σύλληψη εξαιρέσεων εντός του servlet επιτρέπει την μετατροπή τους σε ουσιαστικά, φιλικά προς το χρήστη μηνύματα. Διαφορετικά, η αποτυχία λήψης και χειρισμού εξαιρέσεων θα μεταδοθεί στο περιβάλλον φιλοξενίας του servlet (ειδικότερα τον servlet container), όπου ο προεπιλεγμένος μηχανισμός διαχείρισης σφαλμάτων μπορεί να επηρεάσει τη συνολική ασφάλεια και σταθερότητα του διακομιστή.

Πιθανά προβλήματα ασφαλείας είναι:

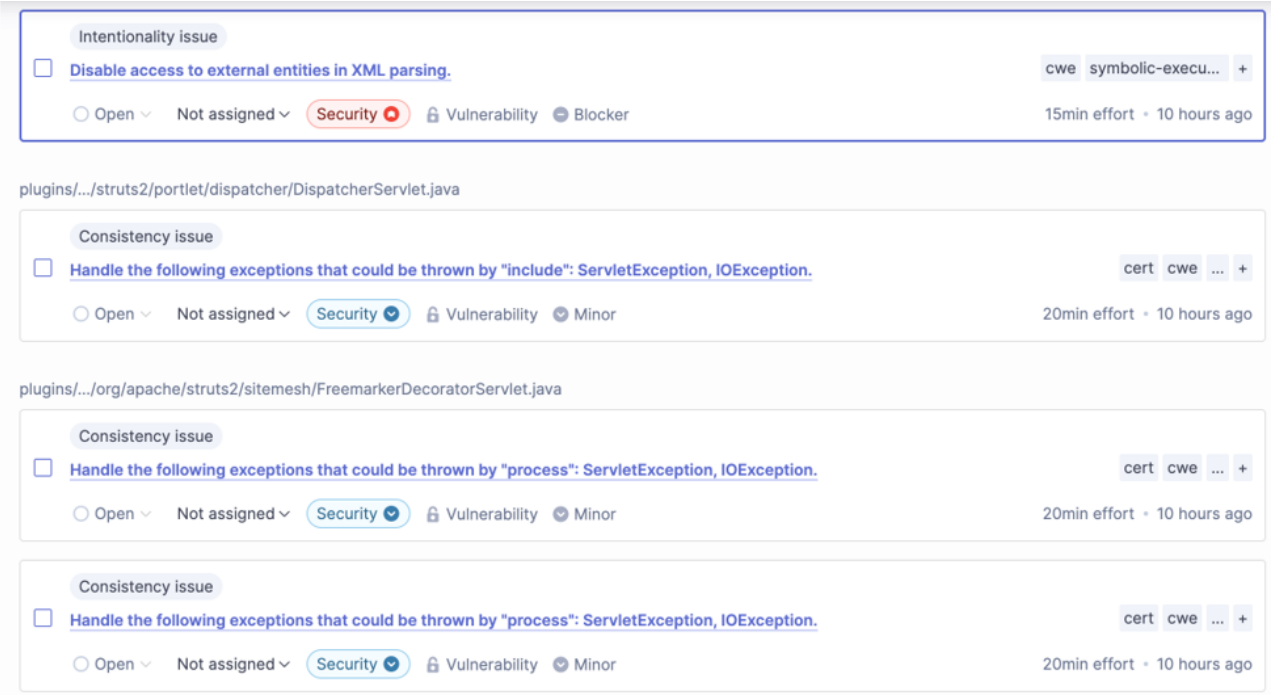
1. **Επιθέσεις άρνησης υπηρεσίας:** Οι εξαιρέσεις που δεν έχουν εντοπιστεί μπορεί να αφήσουν τον servlet container σε ασταθή κατάσταση, γεγονός που μπορεί να εξαντλήσει τους διαθέσιμους πόρους και να καταστήσει το σύστημα μη διαθέσιμο.

2. **Έκθεση ευαίσθητων πληροφοριών:** Οι εξαιρέσεις που χειρίζεται ο servlet container, από προεπιλογή, εκθέτουν στον χρήστη λεπτομερή μηνύματα σφάλματος ή πληροφορίες εντοπισμού σφαλμάτων, τα οποία μπορεί να περιέχουν ευαίσθητα δεδομένα, όπως ίχνη στοίβας, σύνδεση βάσης δεδομένων ή διαμόρφωση συστήματος.

Τρόπος αντιμετώπισης

1. Για την αποφυγή των παραπάνω όλες οι συναρτήσεις τύπου «do» όπως οι doGet και doPost θα έπρεπε να επιλύουν εσωτερικά τις όποιες εξαιρέσεις τους.

2. **Disable access to external entities in XML Parsing** (Απενεργοποίηση πρόσβασης σε εξωτερικές οντότητες XML.)



Εικόνα 56 Απενεργοποίηση πρόσβασης σε εξωτερικές οντότητες XML

Ανάλυση Τρωτότητας

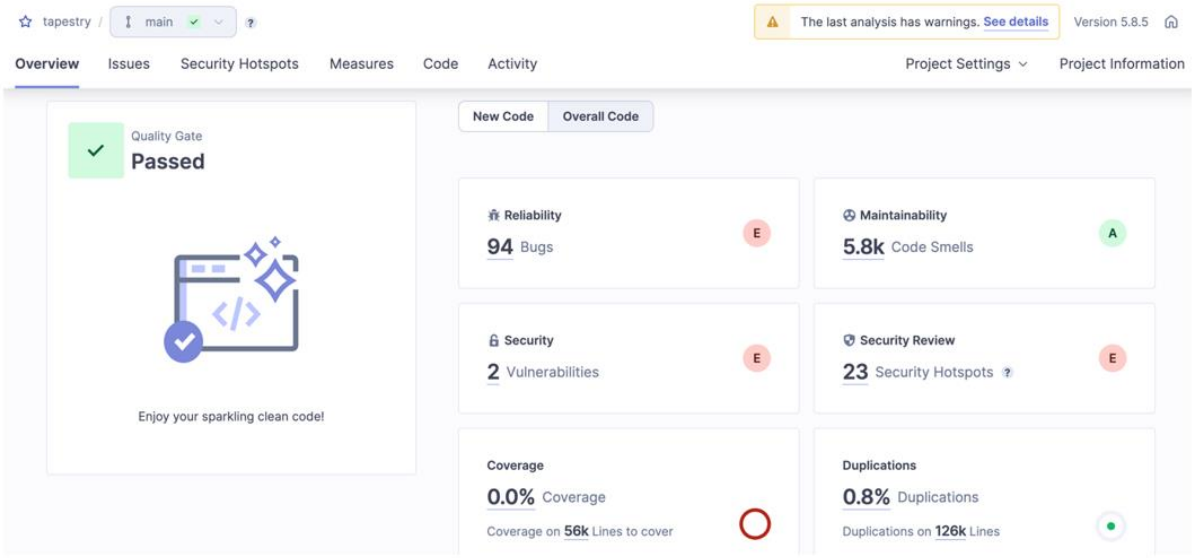
Η τρωτότητα αυτή έχει αναλυθεί νωρίτερα για το Spring - Boot.

Τρόπος αντιμετώπισης

Ο τρόπος αντιμετώπισης αυτής της τρωτότητας έχει αναλυθεί νωρίτερα για το Spring - Boot.

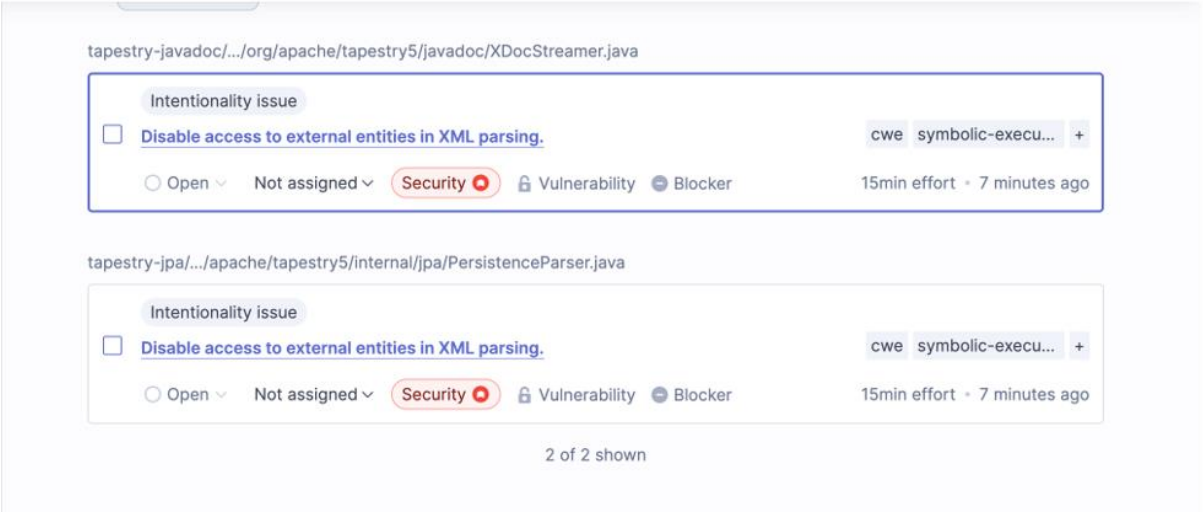
4.2.2.3.6 Ανάλυση τρωτοτήτων για το Tapestry

Το τελευταίο πλαίσιο που αναλύθηκε είναι το Tapestry το οποίο εμφανίζει μόλις 2 ευπάθειες και 23 security reviews, ωστόσο και αυτές έχουν βαθμολογηθεί με Ε από το SonarQube.



Εικόνα 57 Ανάλυση του Tapestry από το Sonarqube

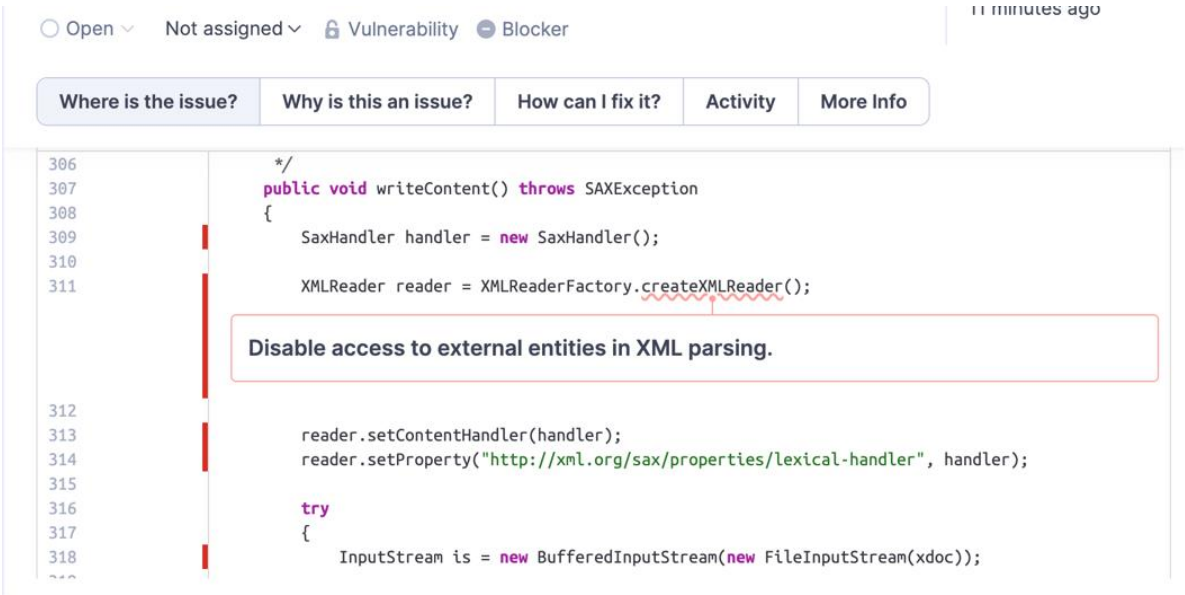
Disable access to external entities in XML Parsing (Απενεργοποίηση πρόσβασης σε εξωτερικές οντότητες XML.)



Εικόνα 58 Απενεργοποίηση πρόσβασης σε εξωτερικές οντότητες της XML.

Ανάλυση Τρωτότητας

Και οι 2 ευπάθειες αφορούν την πρόσβαση που παρέχεται στα XML αρχεία από τους κώδικες ελέγχου.



Εικόνα 59 Απενεργοποίηση πρόσβασης σε εξωτερικές οντότητες XML

Έτσι, μπορούμε να δούμε το σημείο στο οποίο χρησιμοποιείται το αντικείμενο του XMLReader, χωρίς την χρήση κάποιας επιλογής που να προστατεύει την επεξεργασία των XML αρχείων.

Τρόπος αντιμετώπισης

Μία λύση στο παραπάνω ζήτημα ασφαλείας θα ήταν η προσθήκη του παρακάτω κώδικα για την προστασία των XML αρχείων:

```
xmlReader.setFeature("http://apache.org/xml/features/disallow-doctype-decl", true);
```

4.3 Συμπεράσματα

Με βάση την ανάλυση που πραγματοποιήθηκε στις προηγούμενες ενότητες, συγκεντρώσαμε ορισμένα σημαντικά συμπεράσματα στους παρακάτω πίνακες. Ειδικότερα, από τα 3 εργαλεία ανίχνευσης τρωτοτήτων προέκυψαν κάποιες συνολικές τρωτότητες για τα πλαίσια. Η σύγκριση έγινε αρχικά με βάση τις τρωτότητες Υψηλής Κρισιμότητας ενώ σε περίπτωση που δεν υπήρχαν ή υπάρχει ισοβαθμία, η σύγκριση έγινε με βάση τις τρωτότητες μεσαίας κρισιμότητας. Η κατάταξη των πλαισίων έγινε με φθίνουσα σειρά (ως προς τον αριθμό και την κρισιμότητα των τρωτοτήτων) χαρακτηρίζοντας το χαμηλότερο σε επίπεδο πλαίσιο καλύτερο σε ασφάλεια, ενώ το πρώτο σε επίπεδο πλαίσιο μη ασφαλές.
Λόγω του ότι το κάθε εργαλείο χρησιμοποιεί διαφορετικά επίπεδα για να μετρήσει την κρισιμότητα, στους παρακάτω πίνακες θα δηλώσουμε τις κρίσιμες τρωτότητες ως υψηλής κρισιμότητας τρωτότητες και τις τρωτότητες υψηλής κρισιμότητας ως μεσαίας κρισιμότητας για να μπορέσουμε να καταλήξουμε σε καθολικά συμπεράσματα για όλα τα εργαλεία. Αξίζει να σημειωθεί ότι ο αριθμός των τρωτοτήτων για κάθε πλαίσιο που αποτυπώθηκε στον πίνακα αφορά μία μόνο εμφάνιση της κάθε τρωτότητας ακόμα και αν αυτή εμφανίστηκε περισσότερες φορές κατά την ανίχνευση τρωτοτήτων. (π.χ. με τη χρήση του Snyk το Tapestry εμφάνισε 8 κρίσιμου επιπέδου τρωτότητες, αλλά επειδή οι 5 από αυτές αφορούσαν την ίδια τρωτότητα με 5 εμφανίσεις, την προσμετρήσαμε ως 1 και ο αριθμός που αποτυπώθηκε στον πίνακα ήταν 4.

Codacy

Πλαίσια	Υψηλής Κρισιμότητας	Μεσαίας Κρισιμότητας
Struts	**1**	**15**
Tapestry	**1**	**13**
Vaadin	**-**	**29**
Jmix	**-**	**3**
Spring Boot	**-**	**1**
JHipster	**-**	**-**

Πίνακας 15 Συμπεράσματα Codacy

Snyk

Πλαίσια	Υψηλής Κρισιμότητας	Μεσαίας Κρισιμότητας
Tapestry	**4**	**87**
Vaadin	**4**	**52**
Struts	**3**	**19**
Spring-Boot	**1**	**45**
JMix	**-**	**18**
JHipster	**-**	**4**

Πίνακας 16 Συμπεράσματα Snyk

Sonarqube

Πλαίσια	Υψηλής Κρισιμότητας	Μεσαίας Κρισιμότητας
Spring Boot	**5**	**113**
JMix	**2**	**48**
Struts	**2**	**39**
Tapestry	**1**	**23**
Vaadin	**-**	**27**
JHipster	**-**	**6**

Πίνακας 17 Συμπεράσματα Sonarqube

Καθολικά Συμπεράσματα

Πλαίσια	Υψηλής Κρισιμότητας	Μεσαίας Κρισιμότητας
Spring Boot	**6**	**159**
Tapestry	**6**	**123**
Struts	**6**	**73**
Vaadin	**4**	**108**
Jmix	**2**	**69**
JHipster	**-**	**10**

Πίνακας 18 Καθολικά Συμπεράσματα

Όπως αποτυπώνεται και στον παραπάνω πίνακα το πλαίσιο JHipster είναι το πιο ασφαλές καθώς δεν εμφάνισε καμια ευπάθεια σε κανένα από τα εργαλεία ανίχνευσης. Από την άλλη πλευρά, το πλαίσιο Spring-Boot είναι αυτό που εμφάνισε τις περισσότερες ευπάθειες υψηλής και μεσαίας κρισιμότητας οπότε και χαρακτηρίζεται ως το λιγότερο ασφαλές. Με βάση τους παραπάνω πίνακες θα μπορούσε αυτό να θεωρηθεί κάπως οξύμωρο καθώς τα αποτελέσματα του εργαλείου Codacy το έκριναν ως (σχεδόν) τελείως ασφαλές πλαίσιο ενώ τα αποτελέσματα του Snyk το έκριναν ως αρκετά ασφαλές με μια μόνο κρίσιμη τρωτότητα. Ωστόσο, η βαθμολογία του από το Sonarqube ήταν αρκετά χαμηλή (E) και δε μπορεί να θεωρηθεί αμελητέα, επομένως ήταν αυτή που το καθόρισε το τελικό αποτέλεσμα.

5. Δοκιμή διείσδυσης (Penetration Testing)

Στην ενότητα αυτή θα πραγματοποιηθεί μια περιγραφή της διαδικασίας, η οποία ακολουθήθηκε για την δοκιμή διείσδυσης στα πλαίσια. Χωρίζεται σε 5 υποενότητες από τις οποίες η πρώτη περιέχει τον Ορισμό και τα Βήματα της Διείσδυσης, η δεύτερη περιέχει τα εργαλεία που χρησιμοποιήθηκαν για τη διείσδυση, η τρίτη περιέχει τη Μεθοδολογία της διείσδυσης, η τέταρτη περιέχει τα αποτελέσματα της διείσδυσης για κάθε πλαίσιο και η τελευταία τα συμπεράσματα των δοκιμών διείσδυσης.

5.1 Ορισμός και Βήματα Διείσδυσης

Η δοκιμή διείσδυσης, γνωστή και ως ethical hacking ή penetration testing, αποτελεί μια πρακτική ασφάλειας στην οποία εξειδικευμένοι επαγγελματίες, οι οποίοι ονομάζονται ελεγκτές διείσδυσης (penetration testers), αξιολογούν τα υπολογιστικά συστήματα, δίκτυα ή εφαρμογές για ευπάθειες. Σκοπός είναι ο εντοπισμός πιθανών αδυναμιών που θα μπορούσαν να εκμεταλλευτούν επιτιθέμενοι. Ο γενικός στόχος της δοκιμής διείσδυσης είναι να βοηθήσει τους οργανισμούς να ενισχύσουν την ασφάλειά τους αντιμετωπίζοντας αυτές τις ευπάθειες πριν τις εκμεταλλευτούν κακόβουλοι χρήστες.

Βασικά στάδια της δοκιμής διείσδυσης περιλαμβάνουν τον καθορισμό της εμβέλειας της δοκιμής, τη συλλογή πληροφοριών, την ανάλυση ευπαθειών, την εκμετάλλευση πιθανών αδυναμιών, και τον έλεγχο μετά την εκμετάλλευση. Όλες αυτές οι δραστηριότητες πρέπει να πραγματοποιούνται με ηθικούς κανόνες και εντός των προκαθορισμένων ορίων της δοκιμής.

Τα βήματα που ακολουθούνται συνοπτικά είναι τα εξής:

- **Καθορισμός Εμβέλειας (Scope)**: Ορισμός των συστημάτων, των δικτύων ή των εφαρμογών που θα υποβληθούν σε δοκιμή, καθώς και των μεθόδων που θα χρησιμοποιηθούν.
- **Συλλογή Πληροφοριών (Information Gathering):** Ο ελεγκτής συλλέγει πληροφορίες για το περιβάλλον του στόχου, όπως δομή του δικτύου, ρυθμίσεις συστημάτων και πιθανά σημεία εισόδου.
- **Ανάλυση Ευπαθειών (Vulnerability Analysis):** Αναζήτηση και ανάλυση πιθανών ευπαθειών χρησιμοποιώντας αυτόματα εργαλεία σάρωσης και χειροκίνητες επιθετικές δοκιμές.
- **Εκμετάλλευση (Exploitation):** Προσπάθεια εκμετάλλευσης των ευπαθειών για να επιτευχθεί μη εξουσιοδοτημένη πρόσβαση. Πρέπει να γίνεται με ηθικούς κανόνες και μέσα στα όρια της εμβέλειας.
- **Έλεγχος μετά την Εκμετάλλευση (Post-Exploitation Analysis):** Έλεγχος του πώς πρέπει να αντιδρούν τα συστήματα μετά από μια επιτυχημένη εκμετάλλευση καθώς και πως θα πρέπει να αποτρέψουν τις επιθέσεις από το να πραγματοποιηθούν μέσω εφαρμογής προληπτικών μέτρων.

Οι ελεγκτές διείσδυσης παίζουν ένα καίριο ρόλο στη βελτίωση της ασφάλειας ενός οργανισμού, βοηθώντας τους να προλαμβάνουν επιθέσεις και να ενισχύουν τα συστήματά τους.

5.2 Εργαλεία Διείσδυσης

Στην παρακάτω ενότητα θα πραγματοποιηθεί μια ανάλυση των εργαλείων που χρησιμοποιήθηκαν για την απόπειρα διείσδυσης στα πλαίσια. Στην Ενότητα 5.2.1 γίνεται ανάλυση του εργαλείου WMAP, στην Ενότητα 5.2.2 γίνεται ανάλυση του εργαλείου Nikto και στην ενότητα 5.2.3 μια μικρή ανάλυση του εργαλείου Wappalyzer.

5.2.1 WMap

Το WMAP είναι ένας σαρωτής εφαρμογών ιστού που ενσωματώνεται στο Metasploit και έχει σχεδιαστεί ειδικά για εφαρμογές ιστού. Βοηθά στον εντοπισμό ευπαθειών με ενεργή σάρωση και ανάλυση και στην εκμετάλλευση τρωτών σημείων σε εφαρμογές Ιστού. Χρησιμοποιεί μια διαφορετική προσέγγιση σε σύγκριση με άλλες εναλλακτικές λύσεις ανοιχτού κώδικα και εμπορικούς σαρωτές, καθώς το WMAP δεν έχει δημιουργηθεί γύρω από κανένα πρόγραμμα περιήγησης ή spider για τη λήψη και τη διαχείριση δεδομένων.

Επιπρόσθετα, το WMAP είναι σε θέση να σαρώσει ένα ευρύ φάσμα εφαρμογών ιστού, συμπεριλαμβανομένων εκείνων που έχουν κατασκευαστεί με διάφορες τεχνολογίες όπως PHP, ASP.NET και Java. Μπορεί να εντοπίσει μια ποικιλία τρωτών σημείων, συμπεριλαμβανομένου του cross-site scripting (XSS), της SQL injection και της command injection. Το WMAP χρησιμοποιείται συνήθως στις φάσεις αναγνώρισης και απαρίθμησης των δοκιμών διείσδυσης. Βοηθά στον εντοπισμό της δομής της εφαρμογής ιστού και των πιθανών σημείων εισόδου χαρτογραφώντας τη δομή της και στη συνέχεια ελέγχει συστηματικά κάθε στοιχείο για ευπάθειες. Το WMAP μπορεί επίσης να χρησιμοποιηθεί για να ανακαλύψει συγκεκριμένες ευπάθειες στην εφαρμογή-στόχο, οι οποίες στη συνέχεια μπορούν να γίνουν εκμεταλλεύσιμες.

Τέλος, δίνει την δυνατότητα της δημιουργίας προσαρμοσμένων προφίλ σάρωσης για τη στόχευση συγκεκριμένων ευπαθειών ή εφαρμογών.

Το WMAP επιλέγεται επειδή έχει τη δυνατότητα να ανιχνεύσει μια εφαρμογή ιστού, να χαρτογραφήσει τη δομή της και στη συνέχεια να ελέγξει συστηματικά κάθε στοιχείο για ευπάθειες. Η επιλογή του WMAP για μια απόπειρα διείσδυσης σε μια ιστοσελίδα θα μπορούσε να είναι σκόπιμη για πολλούς λόγους:

Αυτοματοποίηση και Ευελιξία: Το WMAP παρέχει ένα πλήρως αυτοματοποιημένο πλαίσιο επίθεσης και ελέγχου ασφαλείας, το οποίο μπορεί να εξετάσει μεγάλους όγκους κώδικα και ιστοσελίδες με αποτελεσματικότητα.

Ευρεία Κάλυψη επιθέσεων: Το WMAP υποστηρίζει μια ευρεία γκάμα επιθέσεων εναντίον ιστοσελίδων, συμπεριλαμβανομένων SQL injection, Cross-Site Scripting (XSS), Command Injection.

Αναφορά Ευρημάτων: Μετά την ολοκλήρωση της δοκιμής, το WMAP παρέχει λεπτομερείς αναφορές σχετικά με τα ευρήματα αδυναμίας ασφαλείας, τα ευπαθή σημεία, και τις πιθανές επιπτώσεις.

Ανοικτού Κώδικα: Το WMAP είναι ένα εργαλείο ανοικτού κώδικα, που σημαίνει ότι μπορεί να προσαρμοστεί και να επεκταθεί ανάλογα με τις ανάγκες του χρήστη.

Κοινότητα και Υποστήριξη: Υπάρχει μια ενεργή κοινότητα πίσω από το WMAP, που παρέχει υποστήριξη, συμβουλές και ανάπτυξη νέων χαρακτηριστικών.

Συνολικά, η επιλογή του WMAP μπορεί να εξυπηρετήσει τις ανάγκες δοκιμής ασφαλείας μιας ιστοσελίδας ή μιας εφαρμογής ιστού με αποτελεσματικότητα, ευελιξία και ευκολία χρήσης.

5.2.2 Nikto

Το Nikto είναι ένας σαρωτής ανοιχτού κώδικα που έχει σχεδιαστεί για ολοκληρωμένες δοκιμές ασφαλείας. Σαρώνει διακομιστές Ιστού για να εντοπίσει ευπάθειες, εσφαλμένες διαμορφώσεις και δυνητικά επικίνδυνα αρχεία ή σενάρια. Οι επαγγελματίες ασφαλείας, οι διαχειριστές συστήματος και οι ελεγκτές διείσδυσης χρησιμοποιούν συνήθως το Nikto για να αξιολογήσουν τη στάση ασφαλείας των διακομιστών ιστού, συμπεριλαμβανομένων πάνω από 6700 πιθανώς επικίνδυνων αρχείων/CGI.

Τα βασικά χαρακτηριστικά του Nikto περιλαμβάνουν την ικανότητά του να σαρώνει για ένα ευρύ φάσμα γνωστών τρωτών σημείων, ξεπερασμένες εκδόσεις λογισμικού, εκτεθειμένους καταλόγους και μη ασφαλείς διαμορφώσεις διακομιστή. Χρησιμοποιεί μια τακτικά ενημερωμένη βάση δεδομένων με γνωστά τρωτά σημεία και επιτρέπει στους χρήστες να προσαρμόζουν τις σαρώσεις με βάση τις συγκεκριμένες ανάγκες τους. Το Nikto παρέχει αποτελέσματα σε διάφορες μορφές, όπως απλό κείμενο, XML, HTML και CSV, διευκολύνοντας την ενσωμάτωση σε διαφορετικά εργαλεία αναφοράς και ανάλυσης.

Το Nikto υποστηρίζει διάφορες μεθόδους HTTP, συμπεριλαμβανομένων των GET, POST και HEAD, κατά τη διάρκεια των σαρώσεων. Επιτρέπει επίσης τη χρήση πρόσθετων για την επέκταση της λειτουργικότητάς του και προσαρμοσμένες δοκιμές και ελέγχους. Λειτουργεί μέσω μιας διεπαφής γραμμής εντολών, καθιστώντας το κατάλληλο για ενσωμάτωση σε σενάρια, αυτοματοποιημένες δοκιμές και μεγαλύτερες ροές εργασιών αξιολόγησης ασφάλειας.

Το Nikto επιλέγεται για την εκτεταμένη βάση δεδομένων του για γνωστές ευπάθειες και κακές διαμορφώσεις. Είναι ιδιαίτερα χρήσιμο για τον εντοπισμό κοινών προβλημάτων, όπως παρωχημένες εκδόσεις λογισμικού, ανασφαλείς διαμορφώσεις διακομιστών και γνωστές ευπάθειες σε διακομιστές και εφαρμογές ιστού.

Βήματα Διαδικασιών Δοκιμής Διείσδυσης: Το Nikto χρησιμοποιείται συνήθως στην αρχική φάση αναγνώρισης των δοκιμών διείσδυσης. Βοηθά στη συγκέντρωση πληροφοριών σχετικά με τον διακομιστή ιστού-στόχο, συμπεριλαμβανομένης της έκδοσης του, του εγκατεστημένου λογισμικού και των πιθανών ευπαθειών. Το Nikto μπορεί να αποκαλύψει "εύκολα θύματα" και να παρέχει πολύτιμες πληροφορίες σχετικά με την κατάσταση ασφαλείας του στόχου, καθοδηγώντας περαιτέρω προσπάθειες εκμετάλλευσης. Η επιλογή του Nikto για μια απόπειρα διείσδυσης σε μια ιστοσελίδα ή εφαρμογή ιστού θα μπορούσε να είναι σκόπιμη για πολλούς λόγους:

Αυτοματοποίηση Ελέγχων Ασφαλείας: Το Nikto παρέχει ένα εκτεταμένο σύνολο εργαλείων για αυτοματοποιημένους ελέγχους ασφαλείας σε ιστοσελίδες, που καλύπτουν μια ευρεία γκάμα πιθανών ευπαθειών και επιθέσεων.

Ευρεία Κάλυψη Ελέγχων: Το Nikto εξετάζει πολλές πιθανές αδυναμίες ασφαλείας, συμπεριλαμβανομένων σφαλμάτων στην ρύθμιση του διακομιστή, ευπάθειες σε εφαρμογές, και ανοιχτές διευθύνσεις URL.

Αναφορές Ασφαλείας: Μετά την ολοκλήρωση της δοκιμής, το Nikto παρέχει λεπτομερείς αναφορές σχετικά με τις ευρήματα αδυναμίας ασφαλείας, τα ευπαθή σημεία, και τις πιθανές επιπτώσεις.

Ευελιξία και Προσαρμοστικότητα: Το Nikto είναι ένα εργαλείο ανοικτού κώδικα, που σημαίνει ότι μπορεί να προσαρμοστεί και να επεκταθεί ανάλογα με τις ανάγκες του χρήστη.

Κοινότητα και Υποστήριξη: Υπάρχει μια ενεργή κοινότητα πίσω από το Nikto, που παρέχει υποστήριξη, συμβουλές και ανάπτυξη νέων χαρακτηριστικών.

Συνολικά, η επιλογή του Nikto μπορεί να εξυπηρετήσει τις ανάγκες δοκιμής ασφαλείας μιας ιστοσελίδας ή εφαρμογής ιστού με αποτελεσματικότητα, ευελιξία και ευκολία χρήσης.

5.2.3 Wappalyzer

Τέλος θα θέλαμε να κάνουμε μια μικρή αναφορά στο εργαλείο Wappalyzer, το οποίο είναι διαθέσιμο ως πρόσθετο (extension) για πολλά δημοφιλή προγράμματα περιήγησης, συμπεριλαμβανομένων του Google Chrome, του Mozilla Firefox, του Microsoft Edge και ανιχνεύει τις τεχνολογίες που χρησιμοποιούνται σε ιστοσελίδες. Μπορεί να αναγνωρίσει διάφορα στοιχεία όπως πλατφόρμες διαχείρισης περιεχομένου, εμπορικές πλατφόρμες, λογισμικό διακομιστή, εργαλεία ανάλυσης και άλλα. Αυτές οι πληροφορίες μπορούν να είναι χρήσιμες για προγραμματιστές, μάρκετινγκ και ερευνητές για να κατανοήσουν την τεχνολογική υποδομή μιας ιστοσελίδας και να λάβουν αποφάσεις σχετικά με ανάπτυξη, μάρκετινγκ και ανάλυση ανταγωνιστών. Το Wappalyzer μπορεί να παρέχει ενδείξεις σχετικά με τα εργαλεία και τις πλατφόρμες που χρησιμοποιούνται διαδεδομένα στον ιστό.

Στην παρούσα εργασία χρησιμοποιήθηκε μόνο για τη δοκιμή διείσδυσης σε ένα πλαίσιο, το Tapestry. Πιο συγκεκριμένα χρησιμοποιήθηκε επιπρόσθετα από το εργαλείο WMAP, αρχικά για τον έλεγχο αν η ιστοσελίδα χρησιμοποιεί το lotus και στη συνέχεια για να ανιχνεύσουμε τις τεχνολογίες που χρησιμοποιούνταν από τη συγκεκριμένη σελίδα που προσπαθούσαμε να διεισδύσουμε.

5.3 Μεθοδολογία Διείσδυσης

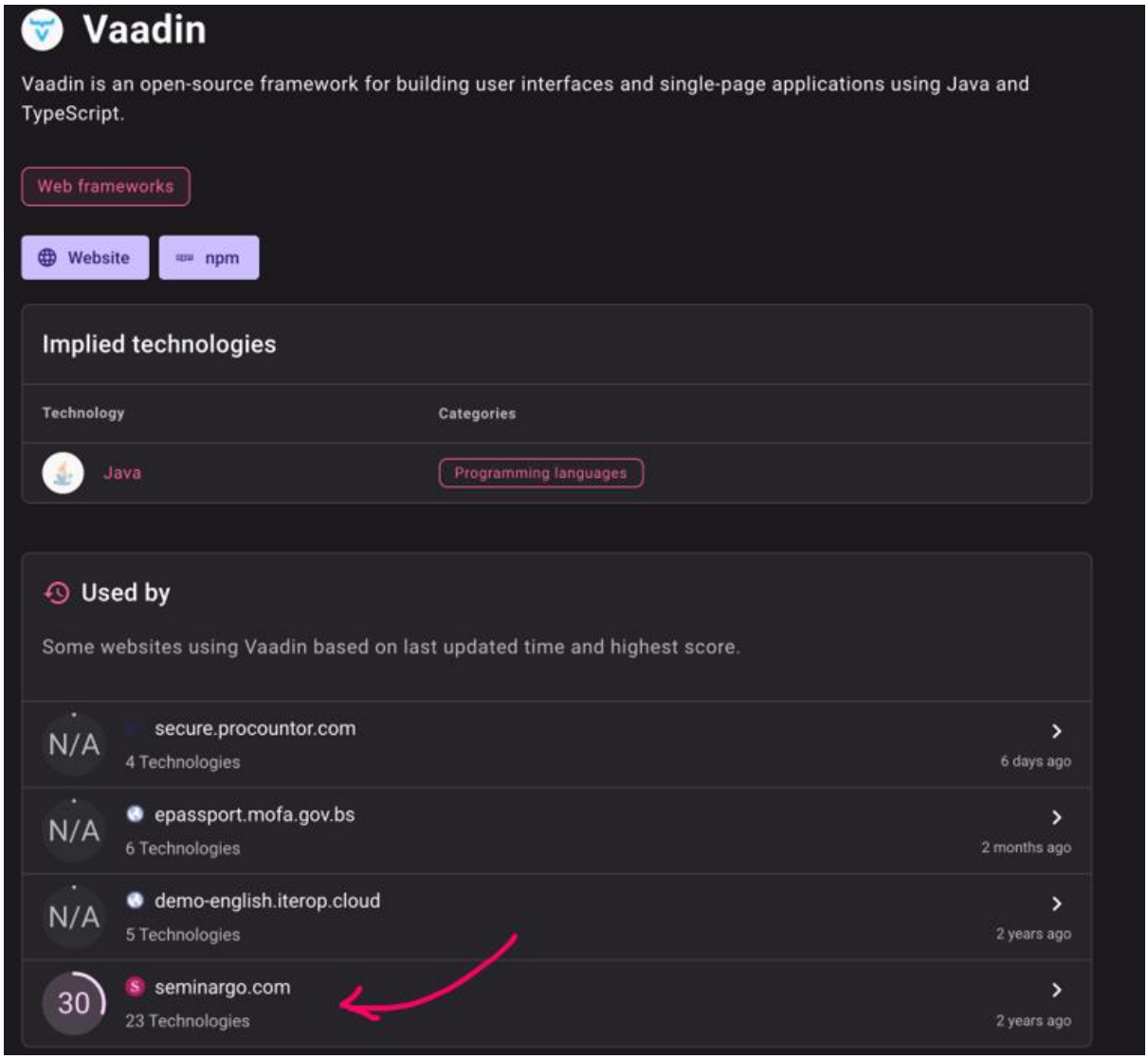
Για την πραγματοποίηση της διείσδυσης, προσπαθήσαμε να εντοπίσουμε πραγματικές εφαρμογές ιστού που έχουν αναπτυχθεί με τα πλαίσια που επιλέξαμε και έπειτα προχωρήσαμε σε αντίστοιχες δοκιμές διείσδυσης για κάθε μία από αυτές. Ο στόχος μας ήταν να εκμεταλλευτούμε τις ευπάθειες που ανιχνεύσαμε στο προηγούμενο κεφάλαιο με τα εργαλεία ανίχνευσης τρωτοτήτων. Οι ευπάθειες αυτές όμως δεν ήταν δυνατό να χρησιμοποιηθούν για δοκιμή διείσδυσης καθώς δεν ήταν εκμεταλλεύσιμες. Επομένως, προσπαθήσαμε να εκμεταλλευτούμε άλλες ευπάθειες που προέκυψαν από τη νέα σάρωση στα πλαίσια της δοκιμής διείσδυσης με τα εργαλεία WMAP και Nikto.

5.4 Δοκιμές Διείσδυσης

Σε αυτή την ενότητα παρουσιάζονται τα αποτελέσματα των δοκιμών διείσδυσης για κάθε πλαίσιο ξεχωριστά και γίνεται μια σύντομη ανάλυσή τους.

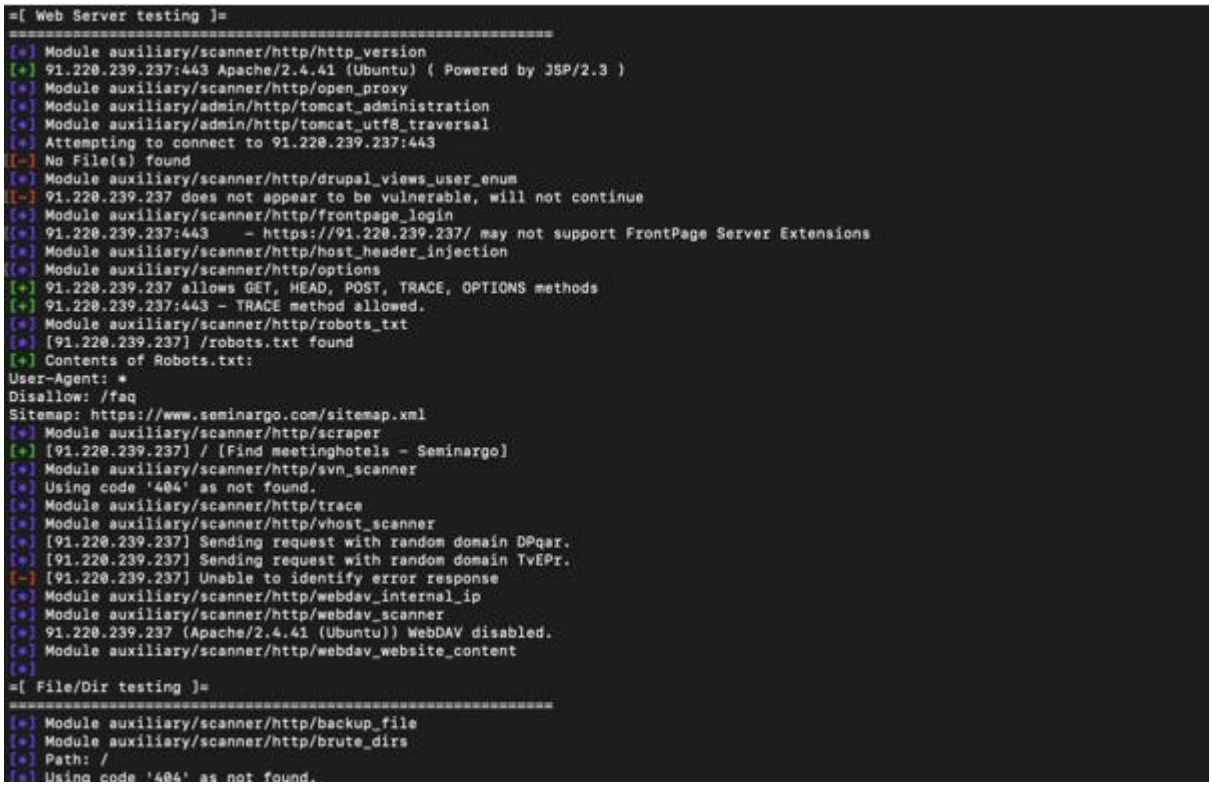
5.4.1 Δοκιμή διείσδυσης στο Vaadin

Η πρώτη δοκιμή διείσδυσης έγινε για το πλαίσιο Vaadin. Επιλέξαμε την ιστοσελίδα https://www.seminargo.com/, η οποία χρησιμοποιεί κατά βάση το Vaadin για την ανάπτυξη των λειτουργιών της.

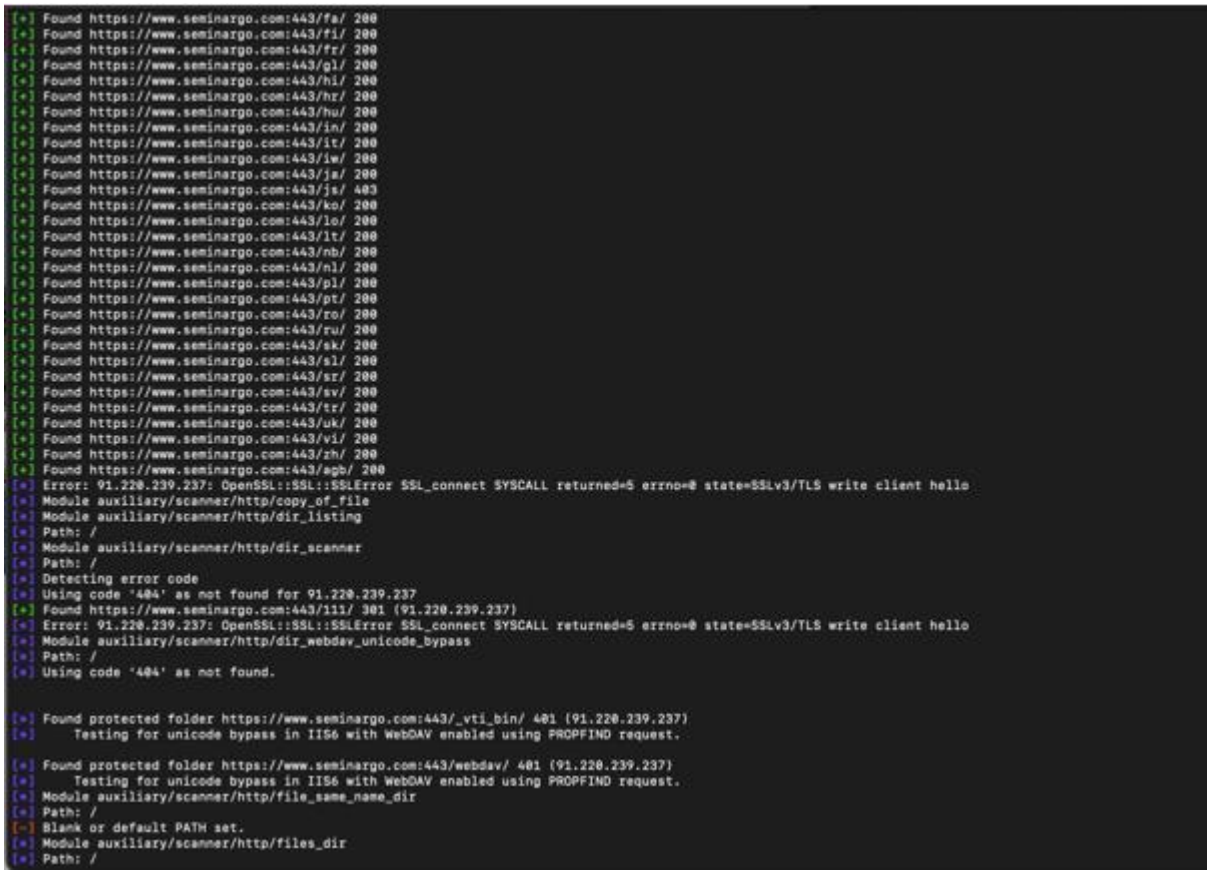


Εικόνα 60 Ιστοσελίδα που έχει αναπτυχθεί με Vaadin

Με τη χρήση του WMap τα αποτελέσματα ήταν τα ακόλουθα:



Εικόνα 61 Αποτελέσματα WMap για Vaadin



Εικόνα 62 Αποτελέσματα WMap για Vaadin

Μετά το επιτυχές σκανάρισμα της web εφαρμογής, το WMap εμφανίζει τις ευπάθειες που βρέθηκαν. Πιο συγκεκριμένα, η εφαρμογή επιτρέπει το HTTP Trace, την αναγνώριση δηλαδή στοιχείων και φακέλων της εφαρμογής από τρίτους.

Εικόνα 63 Αποτελέσματα ευπαθειών με WMap

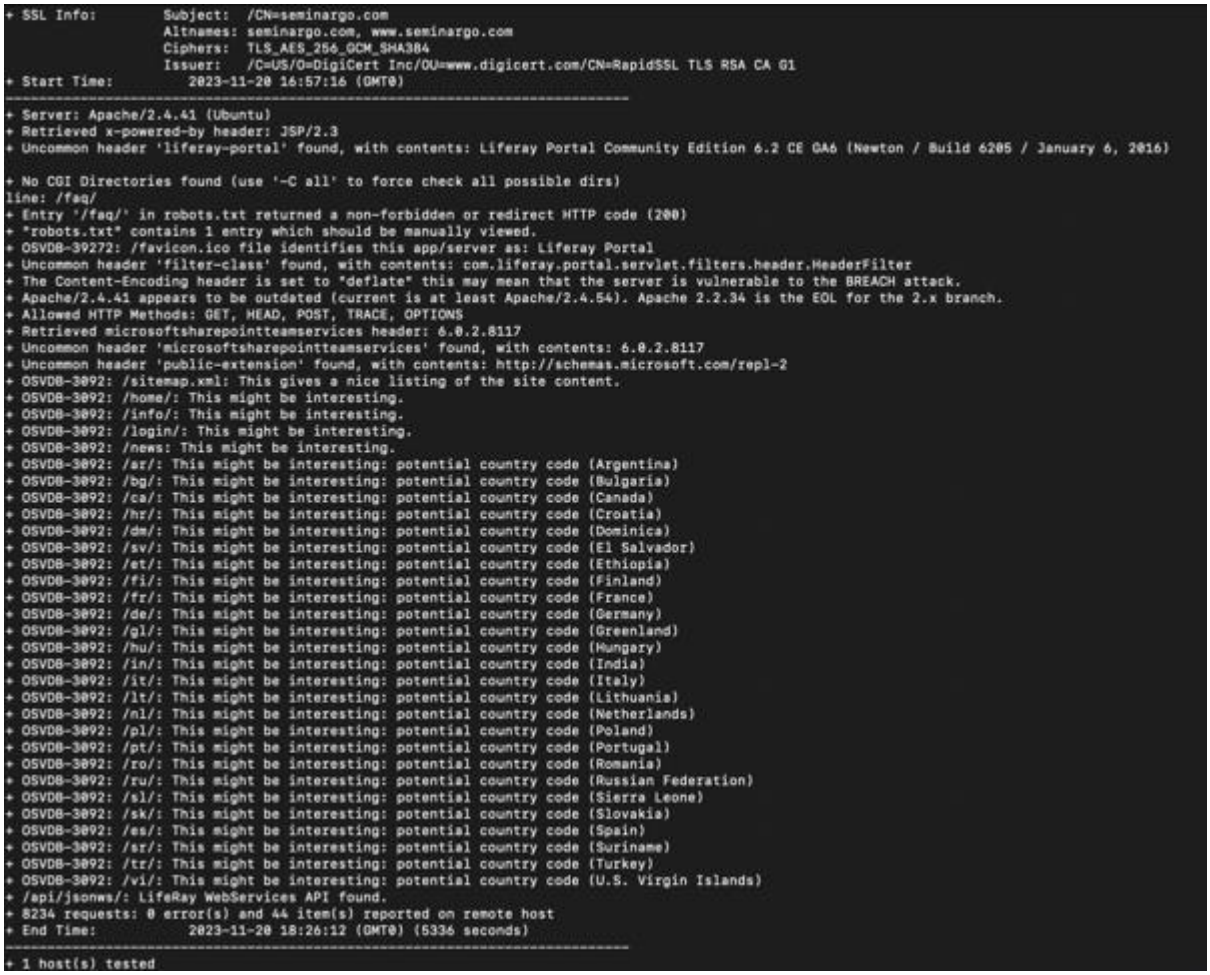
Τα αποτελέσματα της επίθεσης εμφανίζονται παρακάτω:

```
msf6 > wmap_vulns -l
[*] + [34.196.140.85] (34.196.140.85): directory /bks/
[*]     directory Directory found.
[*]     GET Res code: 502
[*] + [34.196.140.85] (34.196.140.85): directory /abv/
[*]     directory Directory found.
[*]     GET Res code: 502
[*] + [91.220.239.237] (91.220.239.237): scraper /
[*]     scraper Scraper
[*]     GET Find meetinghotels - Seminargo
[*] + [91.220.239.237] (91.220.239.237): directory /c/
[*]     directory Directory found.
[*]     GET Res code: 302
[*] + [91.220.239.237] (91.220.239.237): directory /o/
[*]     directory Directory found.
[*]     GET Res code: 503
[*] + [91.220.239.237] (91.220.239.237): directory /ar/
[*]     directory Directory found.
[*]     GET Res code: 200
[*] + [91.220.239.237] (91.220.239.237): directory /bg/
[*]     directory Directory found.
[*]     GET Res code: 200
[*] + [91.220.239.237] (91.220.239.237): directory /ca/
[*]     directory Directory found.
[*]     GET Res code: 200
[*] + [91.220.239.237] (91.220.239.237): directory /cs/
[*]     directory Directory found.
[*]     GET Res code: 200
[*] + [91.220.239.237] (91.220.239.237): directory /da/
[*]     directory Directory found.
[*]     GET Res code: 200
[*] + [91.220.239.237] (91.220.239.237): directory /de/
[*]     directory Directory found.
[*]     GET Res code: 200
[*] + [91.220.239.237] (91.220.239.237): directory /dm/
[*]     directory Directory found.
[*]     GET Res code: 200
[*] + [91.220.239.237] (91.220.239.237): directory /el/
[*]     directory Directory found.
[*]     GET Res code: 200
[*] + [91.220.239.237] (91.220.239.237): directory /en/
[*]     directory Directory found.
[*]     GET Res code: 200
[*] + [91.220.239.237] (91.220.239.237): directory /es/
[*]     directory Directory found.
[*]     GET Res code: 200
[*] + [91.220.239.237] (91.220.239.237): directory /et/
[*]     directory Directory found.
[*]     GET Res code: 200
[*] + [91.220.239.237] (91.220.239.237): directory /eu/
[*]     directory Directory found.
[*]     GET Res code: 200
[*] + [91.220.239.237] (91.220.239.237): directory /fa/
[*]     directory Directory found.
[*]     GET Res code: 200
```

Εικόνα 64 Αναγνώριση καταλόγων Seminargo

Παρατηρούμε ότι όλες οι τρωτότητες προκύπτουν από κακή παραμετροποίηση του εξυπηρετητή της εφαρμογής ιστού.

Κατά τη δοκιμή διείσδυσης με Nikto παρατηρήθηκαν παρόμοια αποτελέσματα όσον αφορά την βαρύτητα των ευπαθειών. Η ανακάλυψη του δέντρου της ιστοσελίδας είναι πρακτικά η μεγαλύτερη ευπάθεια στην ιστοσελίδα, ωστόσο τα αποτελέσματα περιέχουν επιπλέον πληροφορίες σχετικά με πιθανές ευπάθειες ή σημεία στα οποία οι διαχειριστές θα πρέπει να δώσουν σημασία.



Εικόνα 65 Αποτελέσματα ευπαθειών με Nikto

Πιο συγκεκριμένα :

· robots.txt issues:

  o Η καταχώρηση '/faq/' στο robots.txt επέστρεψε κωδικό HTTP μη απαγορευμένο ή ανακατεύθυνσης (200). Αυτό σημαίνει ότι το αρχείο robots.txt δεν έχει διαμορφωθεί σωστά και ενδέχεται να επιτρέπει την πρόσβαση σε περιορισμένους πόρους.

  o Το robots.txt περιέχει μια είσοδο η οποία θα πρέπει να ελεγχθεί. Αυτό σημαίνει ότι το αρχείο robots.txt file μπορεί να περιέχει επιπλέον κανόνες που θα πρέπει να ανασκοπηθούν ώστε να διασφαλιστεί ότι η ευαίσθητη πληροφορία δεν είναι εκτεθειμένη.

· Uncommon headers:

- Βρέθηκε ασυνήθιστη κεφαλίδα 'filter-class' με περιεχόμενο: com.liferay.portal.servlet.filters.header.HeaderFilter. Αυτή η κεφαλίδα ενδέχεται να χρησιμοποιείται από το Liferay Portal για το φιλτράρισμα αιτημάτων.

- Βρέθηκε ασυνήθιστη κεφαλίδα 'microsoftsharepointteamservices' με περιεχόμενο: 6.0.2.8117. Αυτή η κεφαλίδα ενδέχεται να χρησιμοποιείται από τις Υπηρεσίες ομάδας Microsoft SharePoint για τον προσδιορισμό της έκδοσης διακομιστή.

- Βρέθηκε ασυνήθιστη κεφαλίδα 'public-extension' με περιεχόμενο:

  http://schemas.microsoft.com/repl-2. Αυτή η κεφαλίδα ενδέχεται να χρησιμοποιείται από το Microsoft SharePoint για τον προσδιορισμό της έκδοσης διακομιστή.

· Content-Encoding header:

  - Η κεφαλίδα Content-Encoding έχει οριστεί σε "deflate". Αυτό σημαίνει ότι ο διακομιστής χρησιμοποιεί συμπίεση για τη μείωση του μεγέθους της απόκρισης. Αυτό μπορεί να κάνει τον διακομιστή ευάλωτο στην επίθεση BREACH.

· Outdated Apache software:

  - Το Apache/2.4.41 φαίνεται να είναι ξεπερασμένο (το τρέχον είναι τουλάχιστον το Apache/2.4.54). Το Apache 2.2.34 είναι το EOL(End-of-Life) για τον κλάδο 2.x. Αυτό σημαίνει ότι ο διακομιστής δεν εκτελεί την πιο πρόσφατη έκδοση του Apache, η οποία μπορεί να τον εκθέσει σε γνωστές ευπάθειες.

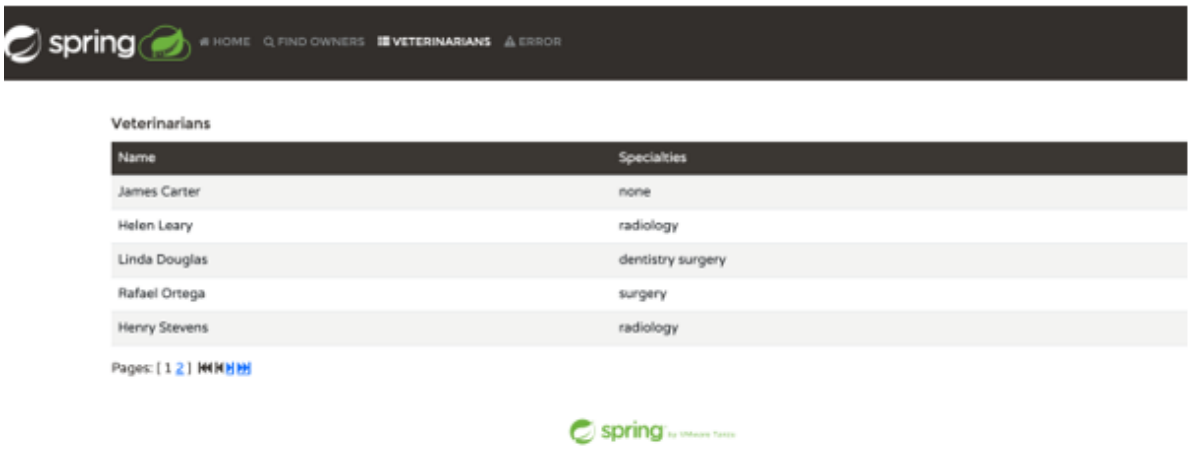
· Allowed HTTP methods:

  - Επιτρεπόμενες μέθοδοι HTTP: GET, HEAD, POST, TRACE, OPTIONS. Αυτό σημαίνει ότι ο διακομιστής επιτρέπει μια ποικιλία μεθόδων HTTP, συμπεριλαμβανομένου του TRACE. Η μέθοδος TRACE μπορεί να χρησιμοποιηθεί για την αποκάλυψη ευαίσθητων πληροφοριών σχετικά με τη διαμόρφωση του διακομιστή.

Στην περίπτωση του Vaadin τα περισσότερα κενά ασφαλείας που παρουσιάζονται δεν αφορούν την ίδια την εφαρμογή που έχει υλοποιηθεί με την χρήση αυτού του πλαισίου αλλά εσφαλμένες παραμετροποιήσεις του εξυπηρετητή ιστού (web server) στον οποίο φιλοξενείται η εφαρμογή αυτή.

5.4.2 Δοκιμή διείσδυσης στο Spring

Στη συνέχεια πραγματοποιήσαμε δοκιμή διείσδυσης στο Spring. Επιλέξαμε μια εφαρμογή[5] που είναι υλοποιημένη με βάση το πλαίσιο αυτό και αφορά την εύρεση ιδιοκτητών για κατοικίδια καθώς και την παροχή λίστας κτηνιάτρων με συγκεκριμένες ειδικότητες. Η εκτέλεση της δοκιμής έγινε τοπικά διότι έγινε εγκατάσταση και εκτέλεση της εφαρμογής στο τοπικό μας σύστημα.



Εικόνα 66 Υλοποίηση εφαρμογής κατοικίδιων με το Spring

Το Spring Petclinic είναι μια Spring εφαρμογή υλοποιημένη μέσω Maven. Για την εκτέλεσή της χρησιμοποιήθηκαν οι εξής εντολές:

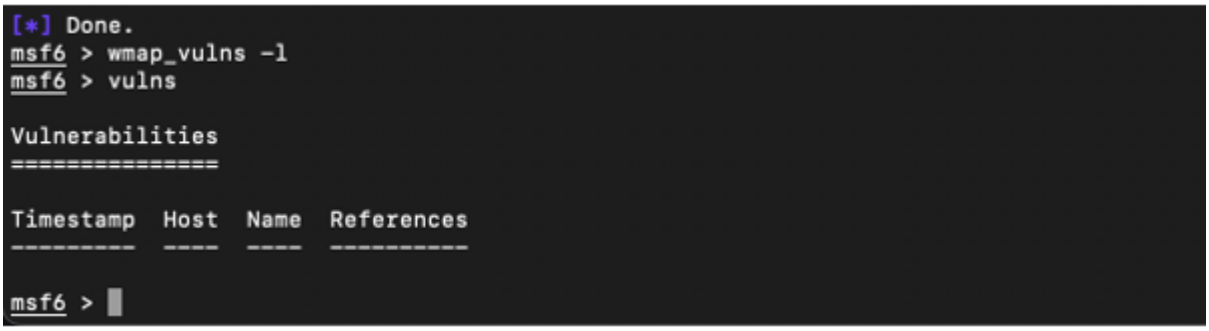
```
git clone https://github.com/spring-projects/spring-petclinic.git
cd spring-petclinic
./mvnw package
java -jar target/*.jar
```

Εικόνα 67 Εντολές για την εκτέλεση της εφαρμογής Petclinic

Στην συνέχεια μέσω της διεύθυνσης “localhost:8000” μπορέσαμε να επισκεφτούμε την εφαρμογή.

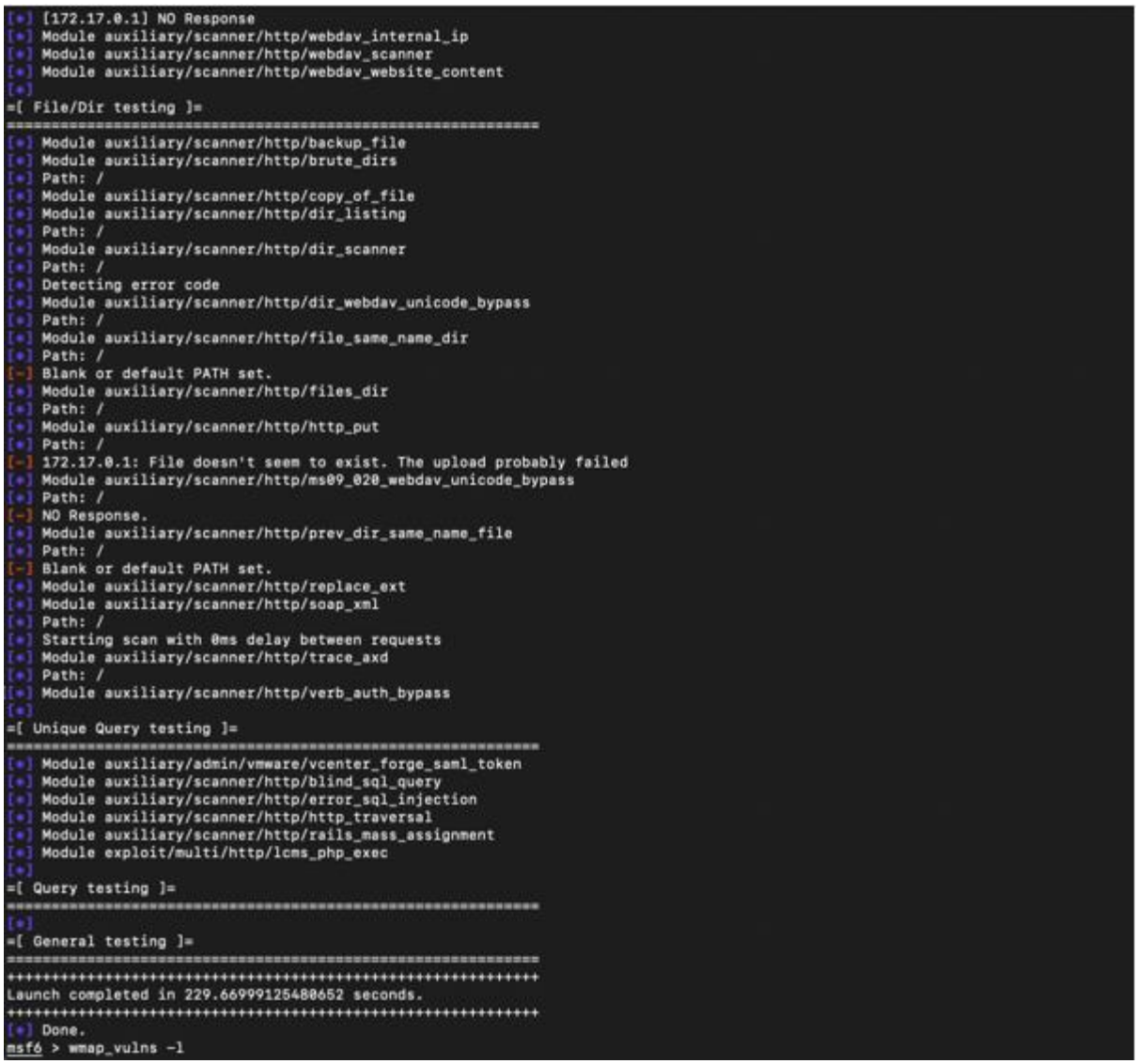
[5] https://github.com/spring-projects/spring-petclinic

Στην προσπάθεια μας να εντοπίσουμε ευπάθειες με χρήση του WMAP δεν καταφέραμε να συγκεντρώσουμε αποτελέσματα σοβαρών ευπαθειών στην εφαρμογή αυτή.



Εικόνα 68 Αποτέλεσμα WMAP για Spring

Το Metasploit σαν αποτέλεσμα μας επιστρέφει ένα κομμάτι που αναφέρεται ως «Unique Query Testing» το οποίο θα χρησιμοποιήσουμε για την ολοκλήρωση των επιθέσεών μας.



Εικόνα 69 Εύρεση πιθανών ευπαθειών

Η μέθοδος «Unique Query Testing» περιλαμβάνει την αποστολή μοναδικών και προσεκτικά σχεδιασμένων ερωτημάτων στο σύστημα στόχο. Αυτά τα ερωτήματα έχουν σκοπό να ενεργοποιήσουν συγκεκριμένες ευπάθειες με βάση τις απαντήσεις τους. Σε αντίθεση με τις γενικές σαρώσεις, η δοκιμή με μοναδικά ερωτήματα επικεντρώνεται στην εκμετάλλευση μεμονωμένων αδυναμιών στο σύστημα.

Όπως φαίνεται στην *εικόνα 69* στον τομέα Unique Query testing, τα πιθανά module που μπορούν να χρησιμοποιηθούν για εκμετάλλευση ευπαθειών αφορούν τα εξής module:

§ Auxiliary/admin/vmware/vcenter_forge_saml_token

§ Auxiliary/scanner/http/blind_sql_query

§ Auxiliary/scanner/http/error_sql_injection

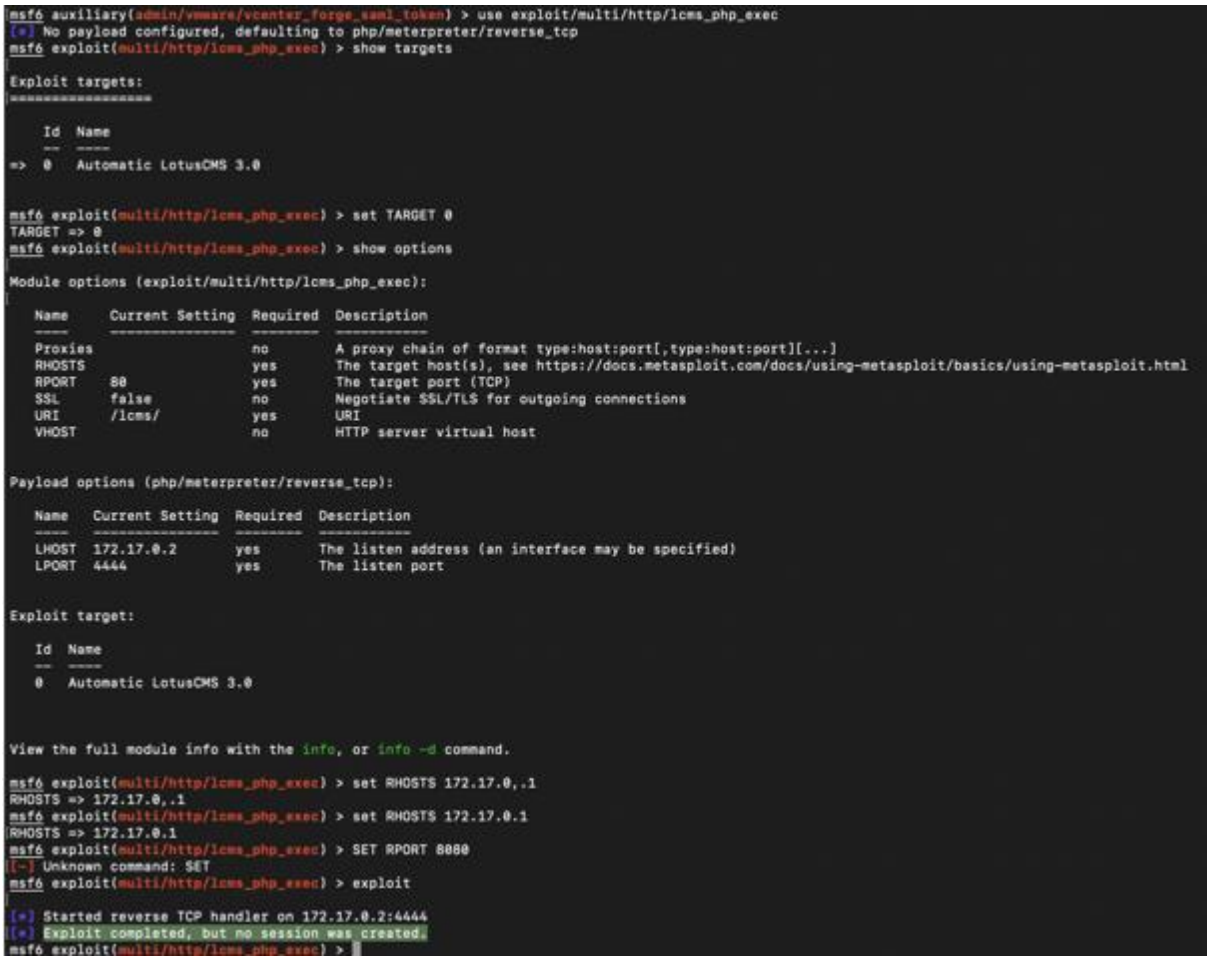
§ Auxiliary/scanner/http/http_traversal

§ Auxiliary/scanner/http/rails_mass_assignment

§ Exploit/multi/http/lcms_php_exec

Ο φάκελος "exploit/multi/http" μέσα στο Metasploit Framework αναφέρεται σε μια συγκεκριμένη κατηγορία exploits που στοχεύουν ευπάθειες σε διαδικτυακές εφαρμογές και διακομιστές web. Συνεπώς, ο φάκελος "exploit/multi/http" περιέχει exploits που αξιοποιούν ευπάθειες στις τεχνολογίες και τα πρωτόκολλα web, επιτρέποντας στους επιτιθέμενους να αποκτήσουν ενδεχομένως μη εξουσιοδοτημένη πρόσβαση, να κλέψουν δεδομένα ή να θέσουν σε κίνδυνο το σύστημα που επηρεάζεται. Σε αντίθεση με τις υπόλοιπα exploits του φακέλου scanner. Ο φάκελος "auxiliary/scanner/http" μέσα στο Metasploit Framework περιέχει modules που σκανάρουν για ευπάθειες και συλλέγουν πληροφορίες σχετικά με διακομιστές web και διαδικτυακές εφαρμογές. Αυτά τα modules δεν προσπαθούν να εκμεταλλευτούν τις εντοπισμένες ευπάθειες, αλλά παρέχουν πληροφορίες στους επαγγελματίες ασφαλείας για την αξιολόγηση πιθανών κινδύνων και τον εντοπισμό τομέων για περαιτέρω έρευνα, και συνεπώς δεν θα βοηθήσουν άμεσα στην εκμετάλλευση των ευπαθειών.

Αφού ρυθμίσαμε τις απαραίτητες παραμέτρους της ενότητας αυτής, δοκιμάσαμε να την εκτελέσουμε. Ωστόσο όπως φαίνεται και στην παρακάτω εικόνα τα αποτελέσματα δεν ήταν τα αναμενόμενα, εφόσον η επίθεση απέτυχε. Πιο συγκεκριμένα, ενώ ολοκληρώθηκε επιτυχώς η εκτέλεση, δεν δημιουργήθηκε κάποια σύνοδος.



Εικόνα 70 Χρήση ευπάθειας multi/http/lens_php_exec

Οι λόγοι που μπορεί να μην πέτυχε η παραπάνω επίθεση είναι :

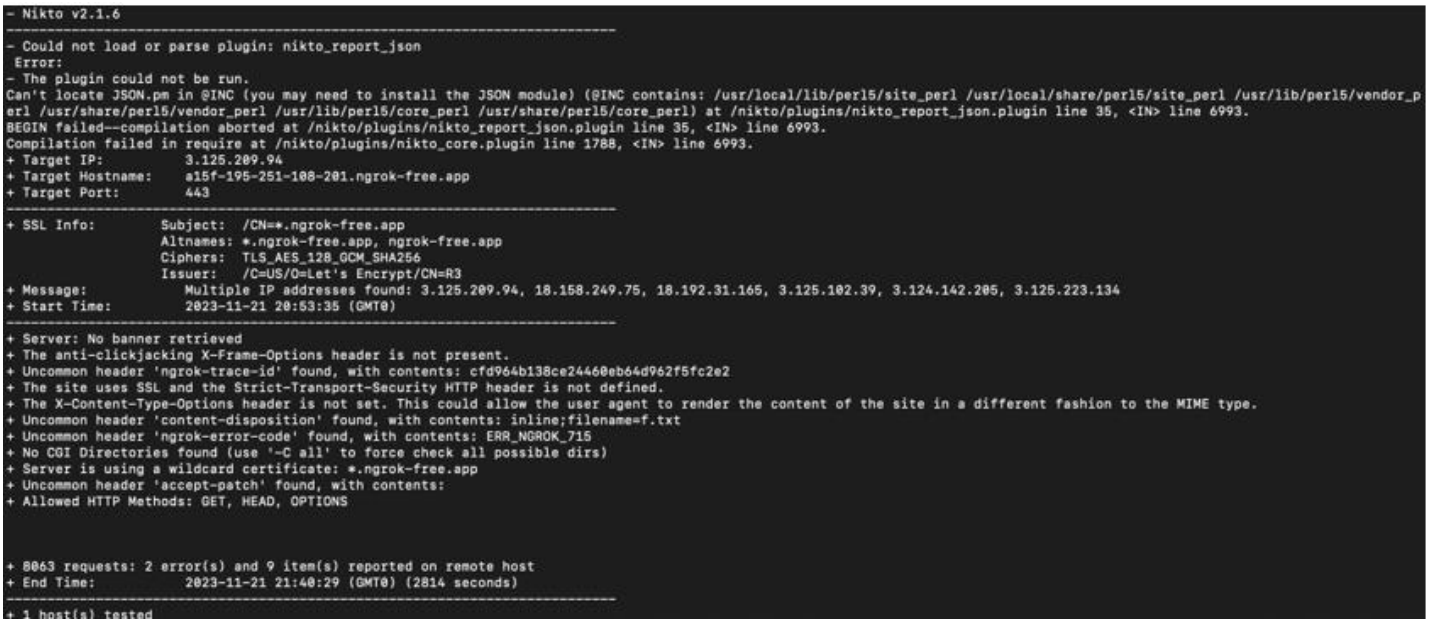
1. Ο Στόχος Έχει Εφαρμόσει Ενημερώσεις:

Το σύστημα στόχος ενδέχεται να έχει εφαρμόσει ενημερώσεις ή μέτρα ασφαλείας που αντιμετωπίζουν τη συγκεκριμένη ευπάθεια που επιχειρείται από το Metasploit.

2. Περιορισμοί δικτύου ή τοίχους προστασίας:

Η ύπαρξη τοίχους προστασίας ή φίλτρων στο δίκτυο μπορεί να εμποδίζει την επιτυχή επικοινωνία μεταξύ του συστήματος του επιτιθέμενου και του στόχου.

Εφόσον οι παράμετροι που θέσαμε είναι σωστοί, μιας και έχουν δοκιμαστεί και σε άλλα συστήματα με επιτυχία και εφόσον δεν έχουμε θέσει στο σύστημα μας (δηλ. το σύστημα που φιλοξενεί την εφαρμογή) τα παραπάνω αντίμετρα προστασίας, καταλήγουμε στο αποτέλεσμα πως ούτε το Spring ούτε η εφαρμογή είχε κάποια σοβαρή ευπάθεια προς εκμετάλλευση.



Εικόνα 71 Αποτελέσματα Nikto για Spring

Στη συνέχεια επιχειρήσαμε διείσδυση στην εφαρμογή του Spring με τη χρήση του Nikto.

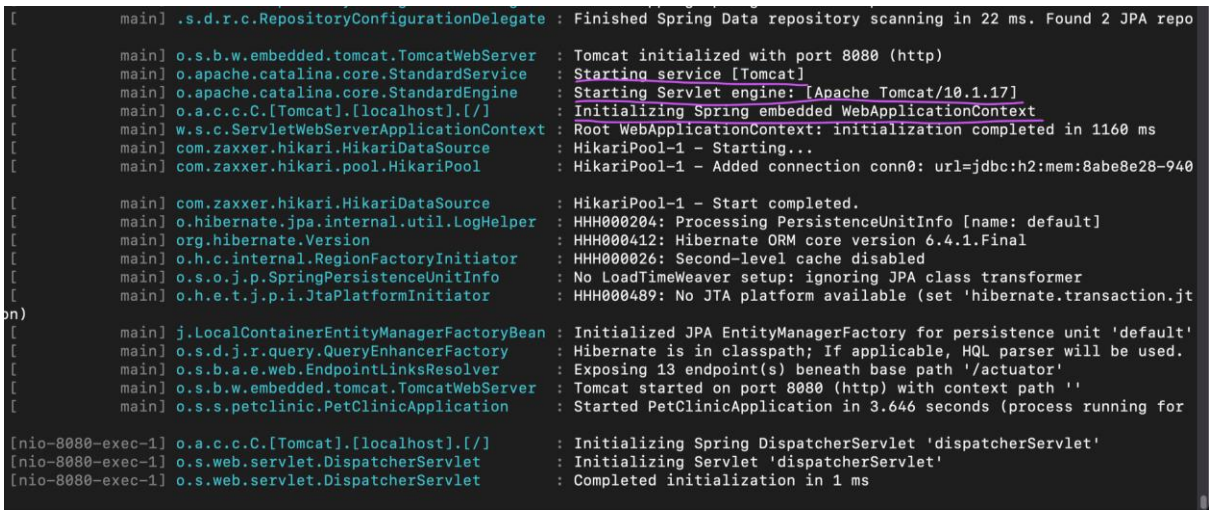
Όπως παρατηρούμε και στην παρακάτω εικόνα δε μπορέσαμε να βρούμε ευπάθειες για το Spring με τη χρήση του Nikto, επομένως η διείσδυση δεν ολοκληρώθηκε με επιτυχία. Τα αποτελέσματα σάρωσης δείχνουν επίσης ότι ο διακομιστής δεν χρησιμοποιεί τις ακόλουθες επικεφαλίδες ασφαλείας:

- X-Frame-Options: Αυτή η επικεφαλίδα βοηθά στην αποτροπή επιθέσεων clickjacking.
- Strict-Transport-Security: Αυτή η επικεφαλίδα βοηθά στην αποτροπή επιθέσεων man-in-the-middle.
- X-Content-Type-Options: Αυτή η επικεφαλίδα βοηθά στην αποτροπή του προγράμματος περιήγησης από την απόδοση του περιεχομένου του ιστότοπου με διαφορετικό τρόπο από τον τύπο MIME.

Τα αποτελέσματα σάρωσης δείχνουν επίσης ότι ο διακομιστής χρησιμοποιεί την εξής επικεφαλίδα:

- content-disposition: Αυτή η επικεφαλίδα χρησιμοποιείται για να καθορίσει τη διάθεση ενός αρχείου, όπως για παράδειγμα αν πρέπει να εμφανιστεί ενσωματωμένο ή να ληφθεί.

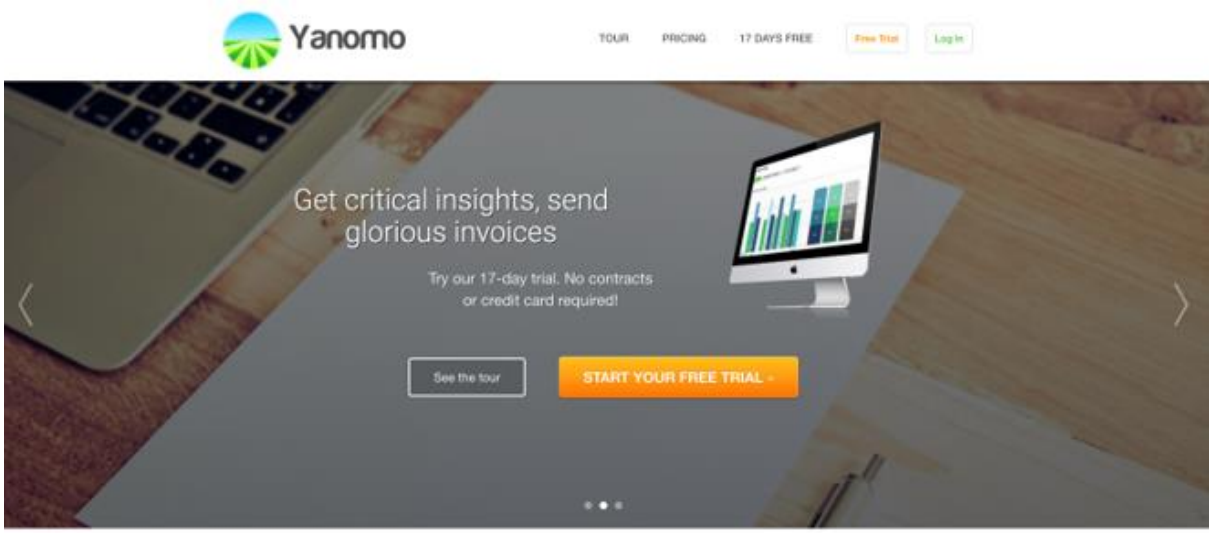
Συνολικά, τα αποτελέσματα σάρωσης του Nikto δείχνουν ότι ο web server, ο ενσωματωμενος Tomcat servlet container, έχει αρκετές πιθανές ευπάθειες ασφαλείας. Ο κάτοχος του κεντρικού υπολογιστή πρέπει να λάβει μέτρα για την αντιμετώπιση αυτών των ευπαθειών, όπως:



Εικόνα 72 Απόδειξη πρόκειται για τον Tomcat web server

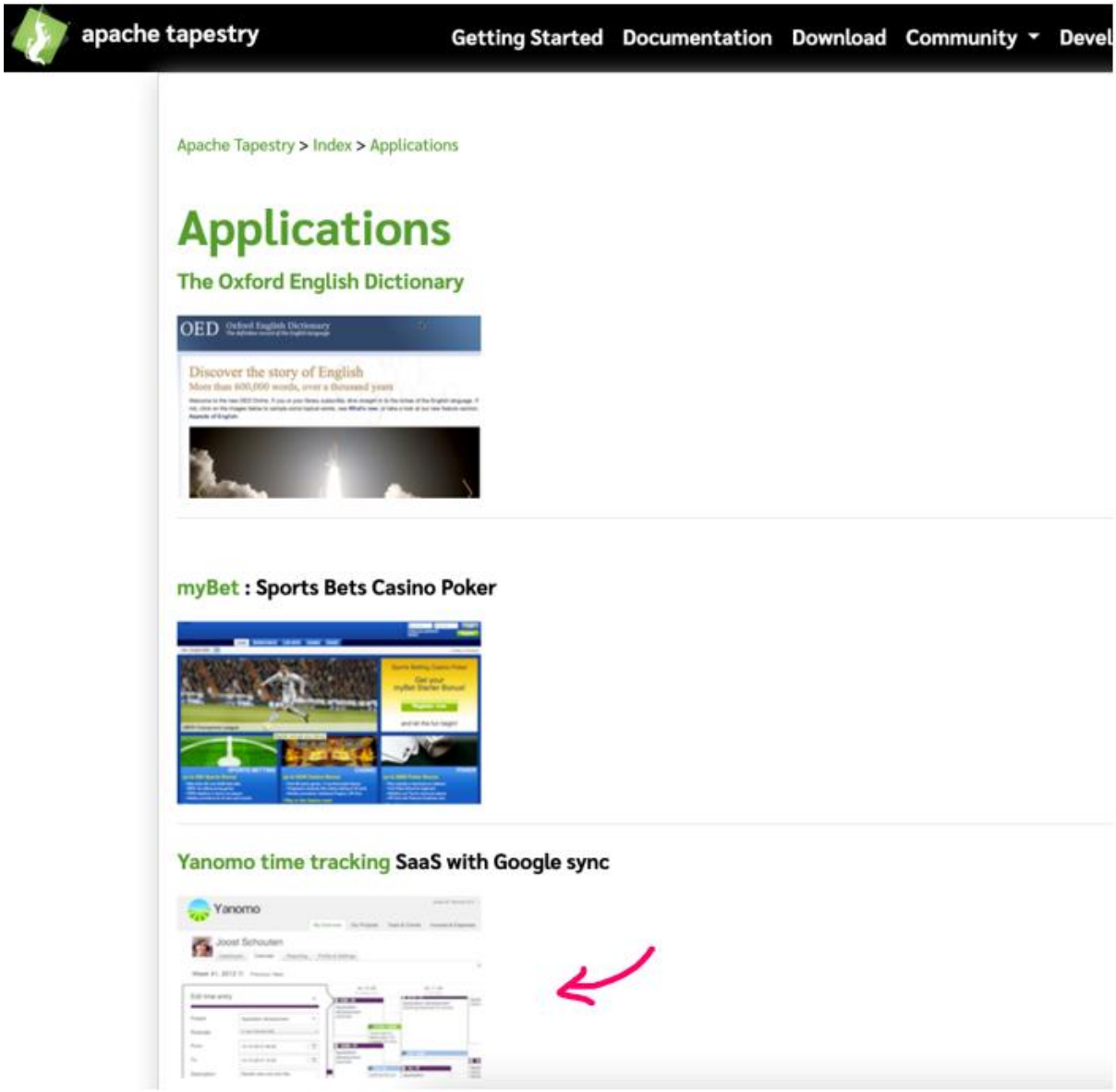
- Ενεργοποίηση των επικεφαλίδων X-Frame-Options, Strict-Transport-Security και X-Content-Type-Options.
- Αφαίρεση των ασυνήθιστων επικεφαλίδων.

5.4.3 Δοκιμή διείσδυσης στο Tapestry



Εικόνα 73 Ιστοσελίδα με χρήση Tapestry

Για το πλαίσιο Tapestry ελέγξαμε την ιστοσελίδα https://www.yanomo.com/, η οποία χρησιμοποιεί κυρίως το πλαίσιο αυτό για την ανάπτυξη των λειτουργιών της.



Εικόνα 74 Ιστοσελίδα με χρήση Tapestry

Επιχειρούμε διείσδυση αρχικά με χρήση του WMAP.

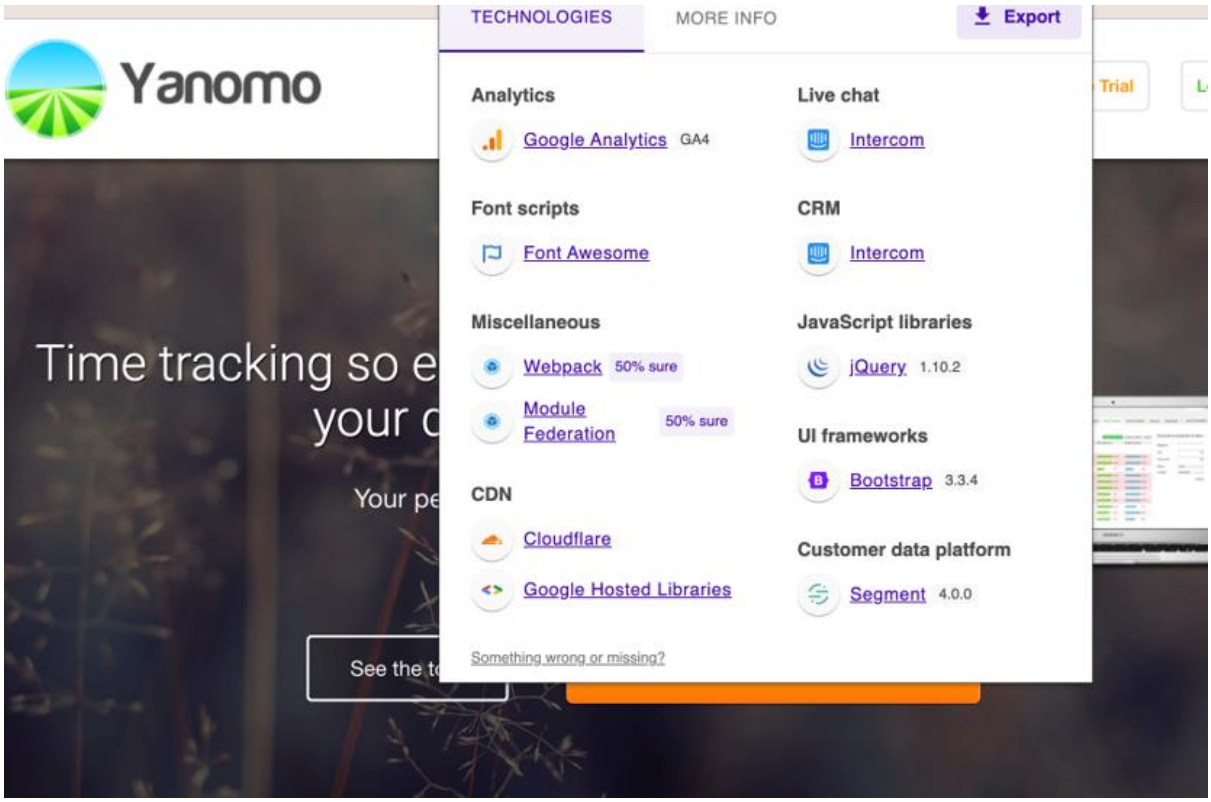
```
msf6 > wmap_run -e
[*] Using ALL wmap enabled modules.
[-] NO WMAP NODES DEFINED. Executing local modules
[*] Testing target:
[*]     Site: 172.17.0.1 (172.17.0.1)
[*]     Port: 8080 SSL: false
[*] Testing started. 2023-11-21 21:33:32 +0000
[*]
=[ SSL testing ]=
[*] Target is not SSL. SSL modules disabled.
[*]
=[ Web Server testing ]=
[*] Module auxiliary/scanner/http/http_version
[*] Module auxiliary/scanner/http/open_proxy
[*] Module auxiliary/admin/http/tomcat_administration
[*] Module auxiliary/admin/http/tomcat_utf8_traversal

[*] Attempting to connect to 172.17.0.1:8080
[-] No File(s) found
[*] Module auxiliary/scanner/http/drupal_views_user_enum
[-] 172.17.0.1 does not appear to be vulnerable, will not continue
[*] Module auxiliary/scanner/http/frontpage_login
[*] Module auxiliary/scanner/http/host_header_injection
[*] Module auxiliary/scanner/http/options
[*] Module auxiliary/scanner/http/robots_txt
[-] [172.17.0.1] /robots.txt - No response
[*] Module auxiliary/scanner/http/scraper
[-] [172.17.0.1] / - No response
[*] Module auxiliary/scanner/http/svn_scanner
[*] Module auxiliary/scanner/http/trace
[*] Module auxiliary/scanner/http/vhost_scanner
[*] [172.17.0.1] Sending request with random domain PzPlt.
[*] [172.17.0.1] Sending request with random domain yFnRT.
[*] [172.17.0.1] NO Response
[*] [172.17.0.1] NO Response
[*] [172.17.0.1] NO Response
[*] [172.17.0.1] NO Response
[*] [172.17.0.1] NO Response
[*] [172.17.0.1] NO Response
[*] [172.17.0.1] NO Response
[*] [172.17.0.1] NO Response
[*] [172.17.0.1] NO Response
[*] [172.17.0.1] NO Response
[*] [172.17.0.1] NO Response
[*] Module auxiliary/scanner/http/webdav_internal_ip
[*] Module auxiliary/scanner/http/webdav_scanner
[*] Module auxiliary/scanner/http/webdav_website_content
[*]
=[ File/Dir testing ]=
```

Εικόνα 75 Αποτελέσματα χρήσης WMAP

```
[+] Server 104.26.6.16:443 responded to SOAPAction: queryon with HTTP: 403 Forbidden.
[+] Server 104.26.6.16:443 responded to SOAPAction: queryoff with HTTP: 403 Forbidden.
[*] Module auxiliary/scanner/http/trace_axd
[*] Path: /
[*] Module auxiliary/scanner/http/verb_auth_bypass
[*]
=[ Unique Query testing ]=
[*] Module auxiliary/admin/vmware/vcenter_forge_saml_token
[*] Module auxiliary/scanner/http/blind_sql_query
[*] Module auxiliary/scanner/http/error_sql_injection
[*] Module auxiliary/scanner/http/http_traversal
[*] Module auxiliary/scanner/http/rails_mass_assignment
[*] Module exploit/multi/http/lcms_php_exec
[*]
=[ Query testing ]=
[*]
=[ General testing ]=
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Launch completed in 37022.4358997345 seconds.
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
[*] Done.
msf6 >
```

Εικόνα 76 Αποτελέσματα χρήσης WMAP

Τα αποτελέσματα του Wmap, είναι παρόμοια με τα προηγούμενα πλαίσια, όπου επιτρέπεται η σάρωση της ιστοσελίδας για πιθανούς καταλόγους, ενώ η επίθεση της σφυρηλάτησης(Forging) ενός SAML token αποτυγχάνει. Το τελευταίο module που δόθηκε πιθανότητα να λειτουργήσει είναι το lcms_php_exec, το οποίο εκμεταλλεύεται μια ευπάθεια που βρίσκεται στη συνάρτηση Router() του Lotus CMS 3.0. Δοκιμάσαμε να ελέγξουμε εάν η ιστοσελίδα χρησιμοποιεί το Lotus μέσω του εργαλείου Wappalyzer που εμφανίζει τις τεχνολογίες που χρησιμοποιούνται από κάποια ιστοσελίδα.
Όπως φαίνεται στην παρακάτω εικόνα, το Lotus δεν εμφανίζεται ωστόσο δοκιμάσαμε να εκτελέσουμε την επίθεση. Αυτό γίνεται με την ενσωμάτωση του κώδικα PHP στην παράμετρο 'page', η οποία θα μεταβιβαστεί σε μια κλήση eval, επιτρέποντας έτσι την απομακρυσμένη εκτέλεση κώδικα. Ωστόσο και σε αυτήν την περίπτωση η ενότητα επίθεσης/εκμετάλλευσης απέτυχε να δημιουργήσει κάποια συνεδρία.



Εικόνα 77 Στιγμιότυπο από Wappalyzer που εμφανίζει τις τεχνολογίες που χρησιμοποιούνται

Στη συνέχεια επιχειρούμε διείσδυση με Nikto

Εικόνα 78 Αποτελέσματα Nikto για Tapestry

Το σκανάρισμα του **Nikto** έδειξε και σε αυτήν την εφαρμογή ότι μπορεί να προκληθεί επίθεση BREACH. Η επίθεση BREACH, που ανακαλύφθηκε τον Σεπτέμβριο του 2012, είναι μια επίθεση που λαμβάνει πληροφορίες σχετικά με δεδομένα απλού κειμένου από τον κρυπτογραφικό αλγόριθμο που χρησιμοποιείται στη συμπίεση σε επίπεδο HTTP, γνωστή και ως επίθεση πλευρικού καναλιού συμπίεσης. Εν ολίγοις, αυτή η επίθεση εξετάζει τη συμπεριφορά του αλγόριθμου συμπίεσης HTTP για να αποκτήσει πληροφορίες σχετικά με τα δεδομένα που κρυπτογραφήθηκαν.

Οι επιθέσεις BREACH μπορεί να περιλαμβάνουν τις εξής δραστηριότητες:

- **Ανεπιθύμητη Πρόσβαση:** Προσπάθεια εισβολής σε συστήματα ή δίκτυα χωρίς άδεια ή εξουσιοδότηση.
- **Κλοπή Δεδομένων:** Απόκτηση και αντιγραφή ευαίσθητων δεδομένων, όπως προσωπικές πληροφορίες, πιστωτικές κάρτες, ή επαγγελματικά μυστικά.
- **Εξόρυξη Πληροφοριών**: Συλλογή πληροφοριών για τη δημιουργία προφίλ ή για την υλοποίηση επιθέσεων σε βάρος των θυμάτων.
- **Καταστροφή ΄Η Παραποίηση Δεδομένων**: Καταστροφή ή παραποίηση των δεδομένων για να προκληθεί ζημιά ή παραπλάνηση.
- **Καταστροφή ΄Η Παραβίαση Συστημάτων:** Επιθέσεις που στοχεύουν στην καταστροφή των συστημάτων ή στην παραβίαση των δικαιωμάτων πρόσβασης.

Όσον αφορά την ευπάθεια X-Frame-Options, όταν ένας ιστότοπος δεν διαθέτει τη κεφαλίδα X-FRAME-OPTIONS, τα προγράμματα περιήγησης επιτρέπουν από προεπιλογή σε οποιοδήποτε άλλο ιστότοπο να εμφανίσει το περιεχόμενό του μέσα σε πλαίσια (iframe). Αυτό δημιουργεί ένα τρωτό σημείο που μπορούν να εκμεταλλευτούν οι επιτιθέμενοι. Μπορούν να ενσωματώσουν το ευάλωτο περιεχόμενο σε ένα κακόβουλο πλαίσιο στον δικό τους ιστότοπο, σχεδιασμένο προσεκτικά για να εξαπατήσει τους χρήστες να κάνουν κλικ σε κάτι που είναι κρυμμένο από κάτω (hidden layer). Για παράδειγμα, ένα κουμπί σύνδεσης στον ιστότοπο του

επιτιθέμενου μπορεί να τοποθετηθεί ακριβώς πάνω από το νόμιμο κουμπί σύνδεσης που είναι ενσωματωμένο στο πλαίσιο, ξεγελώντας τους χρήστες να αποκαλύψουν τα διαπιστευτήριά τους στον επιτιθέμενο. Επομένως, η απουσία ενός header X-FRAME-OPTIONS αυξάνει σίγουρα τον κίνδυνο επιθέσεων clickjacking. Είναι κρίσιμο για τους ιδιοκτήτες ιστοτόπων να εφαρμόσουν αυτό το header με μια ασφαλή τιμή για να προστατεύσουν τους χρήστες και το περιεχόμενό τους.

Ο διακομιστής χρησιμοποιεί το Cloudflare. Αυτό σημαίνει ότι ορισμένες από τις ευπάθειες ασφαλείας που εντοπίστηκε το Nikto μπορεί να μετριαστούν από τα μέτρα ασφαλείας του Cloudflare.

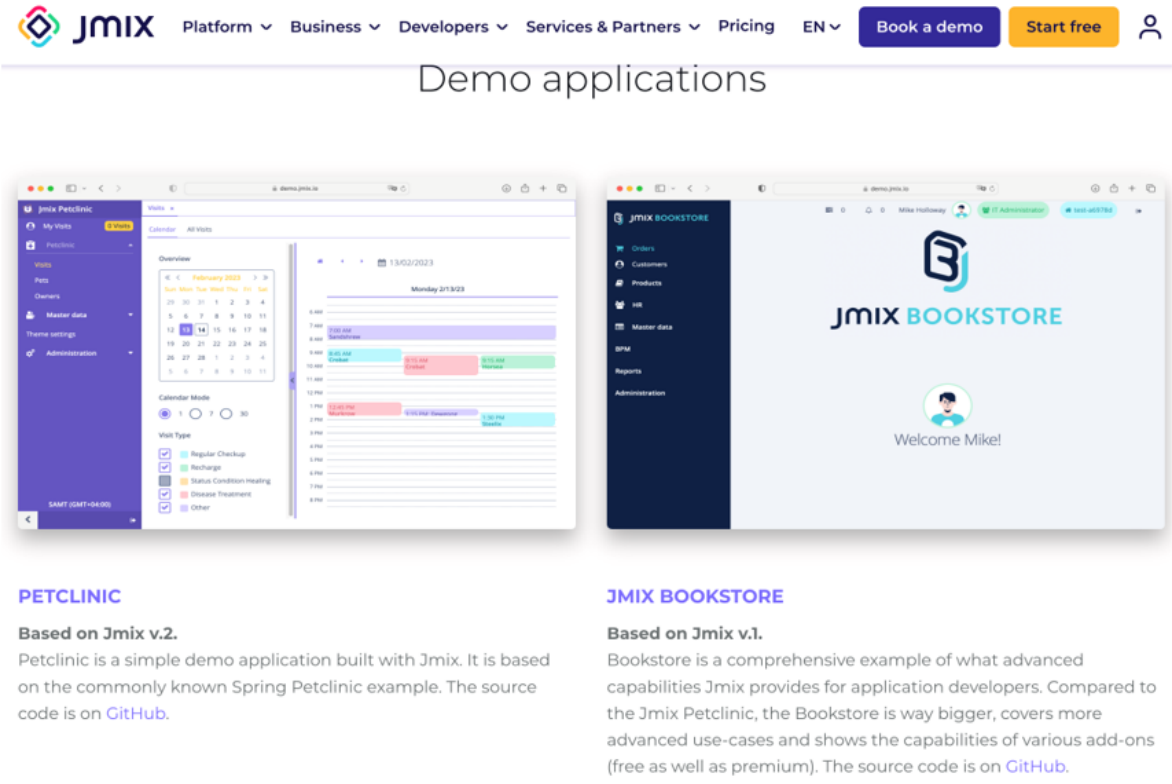
Ο διακομιστής δεν χρησιμοποιεί τον επικεφαλίδα HTTP Strict-Transport-Security (HSTS). Αυτό θα μπορούσε να επιτρέψει σε έναν εισβολέα να υποβαθμίσει τη σύνδεση ενός χρήστη σε μια μη ασφαλή σύνδεση HTTP.

Ο διακομιστής δεν χρησιμοποιεί τον επικεφαλίδα X-Content-Type-Options. Αυτό θα μπορούσε να επιτρέψει σε έναν εισβολέα να εισαγάγει κακόβουλο κώδικα στον ιστότοπο.

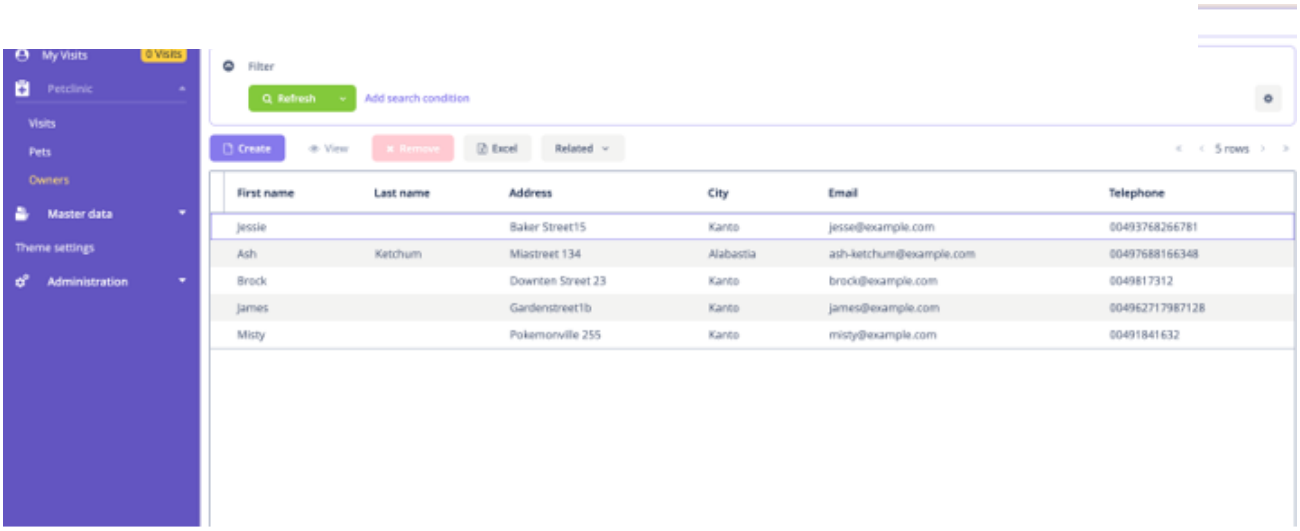
Τα αποτελέσματα της σάρωσης Nikto υποδεικνύουν ευπάθειες που προέρχονται από τη διαμόρφωση του διακομιστή, και όχι από τον ίδιο τον κώδικα της εκάστοτε εφαρμογής. Αυτά τα ευρήματα τονίζουν τη σημασία της σωστής διαμόρφωσης του διακομιστή για τη διασφάλιση ενός ασφαλούς περιβάλλοντος εκτέλεσης εφαρμογών. Ενώ τα πλαίσια μπορούν να προσφέρουν ορισμένα χαρακτηριστικά ασφαλείας, τελικά, η διαμόρφωση του διακομιστή διαδραματίζει κρίσιμο ρόλο στην προστασία της εφαρμογής και των χρηστών της. Επομένως, η αντιμετώπιση αυτών των ευπαθειών απαιτεί προσαρμογές στο διακομιστή, και όχι στον κώδικα της εφαρμογής. Ο διαχειριστής χρειάζεται να εφαρμόσει τις απαραίτητες επικεφαλίδες ασφαλείας και να διαμορφώσει σωστά το Content-Encoding για την προστασία του διακομιστή και των χρηστών του.

5.4.4 Δοκιμή διείσδυσης για το Jmix

Η εφαρμογή που ελέγχτηκε είναι το Petclinic[6] που είναι υλοποιημένο με το Jmix. Για τα πλαίσια της διπλωματικής, η εφαρμογή αυτή ελέγχτηκε online.



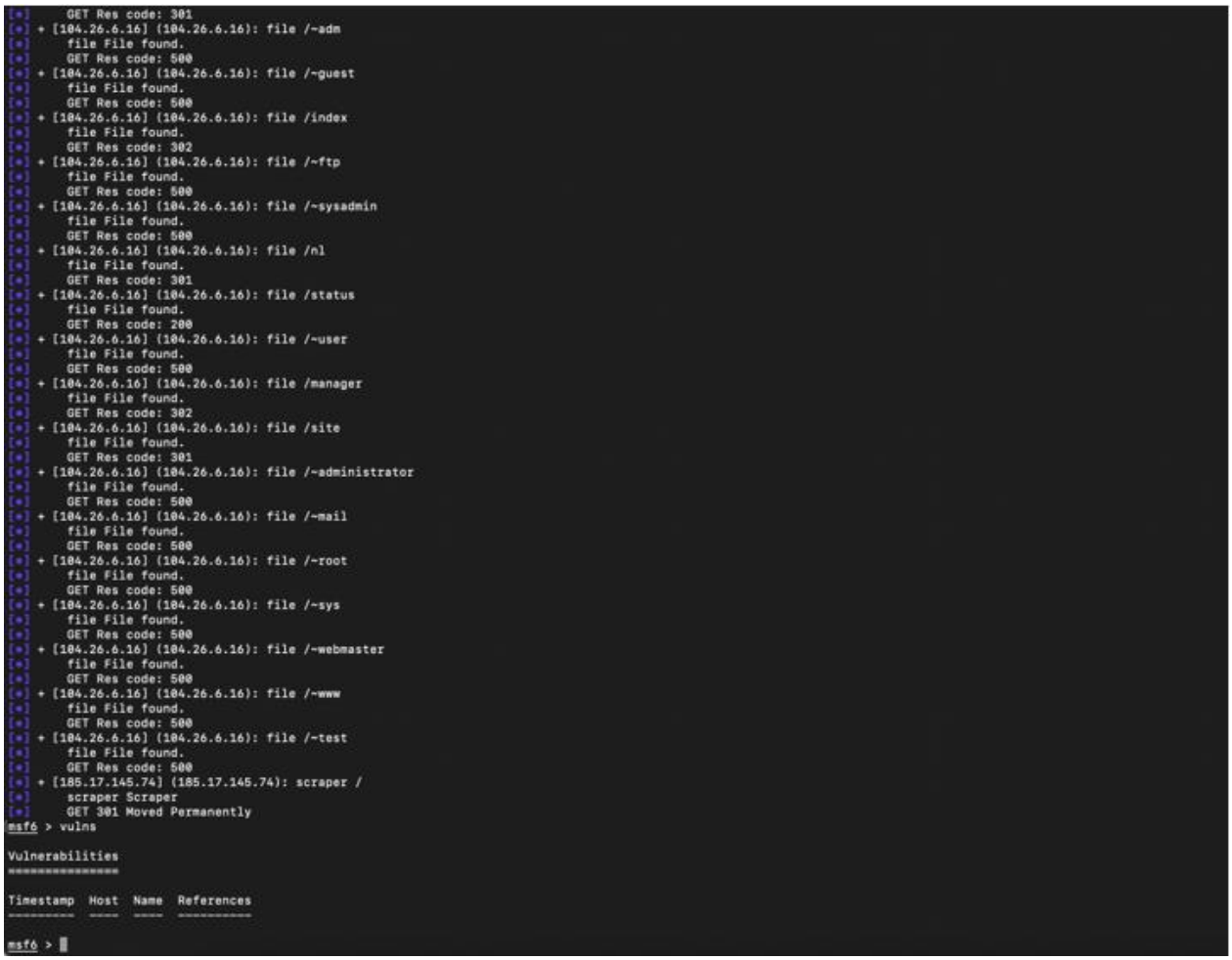
Εικόνα 79 Εφαρμογή με χρήση Jmix



Εικόνα 80 Περιεχόμενο που αφορά κλινική κατοικιδίων

[6] https://demo.jmix.io/petclinic/.

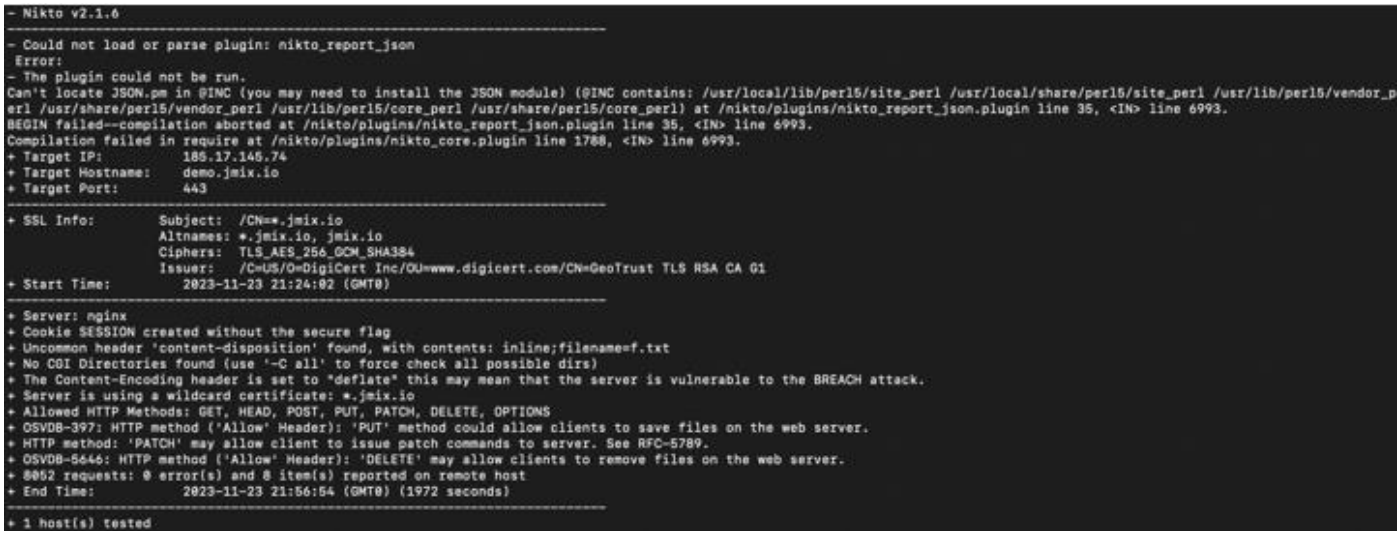
Σκανάρουμε την εφαρμογή αρχικά με τη βοήθεια του WMAP



Εικόνα 81 Αποτελέσματα WMAP

Όπως παρατηρούμε στην παραπάνω εικόνα, το WMAP δεν κατάφερε να εντοπίσει καμία ευπάθεια για το JMix.

Στη συνέχεια πραγματοποιούμε την ίδια διαδικασία με τη χρήση του Nikto.



Εικόνα 82 Αποτελέσματα του Nikto

Το Nikto εντόπισε τις HTTP μεθόδους που επιτρέπονται στην παραπάνω εφαρμογή. Τα αποτελέσματα σάρωσης στην εικόνα δείχνουν ότι ο στόχος -διακομιστής είναι ευάλωτος σε διάφορα πιθανά προβλήματα ασφαλείας.

- Βρέθηκε μη συνηθισμένο περιεχόμενο κεφαλίδας content-disposition, με περιεχόμενο: inline; filename=f.txt: Αυτό το εύρημα υποδηλώνει ότι ο διακομιστής ενδέχεται να είναι ευάλωτος σε επίθεση Cross-Site Scripting (XSS). Οι επιθέσεις XSS επιτρέπουν στους επιτιθέμενους να εισάγουν κακόβουλο κώδικα σε έναν ιστότοπο που στη συνέχεια μπορεί να εκτελεστεί από χρήστες που επισκέπτονται τον ιστότοπο.

1. Ψευδής Αίσθηση Ασφάλειας:

Η παρουσία της οδηγίας inline μπορεί να δημιουργήσει μια ψευδή αίσθηση ασφάλειας, οδηγώντας τους προγραμματιστές σε υποτίμηση της ανάγκης για σωστή επικύρωση δεδομένων εισόδου και κωδικοποίηση στην πλευρά του εξυπηρετητή. Οι επιτιθέμενοι μπορεί να εκμεταλλευτούν αυτό, εισάγοντας κακόβουλα scripts σε δεδομένα ελεγχόμενα από τον χρήστη, όπως ονόματα αρχείων, ακόμη και αν ο εξυπηρετητής ορίζει το περιεχόμενο ως inline. Εάν ο εξυπηρετητής δεν απολυμάνει σωστά τα δεδομένα πριν τα χρησιμοποιήσει στο header, το script θα μπορούσε να εκτελεστεί όταν εμφανιστεί στο πρόγραμμα περιήγησης του χρήστη.

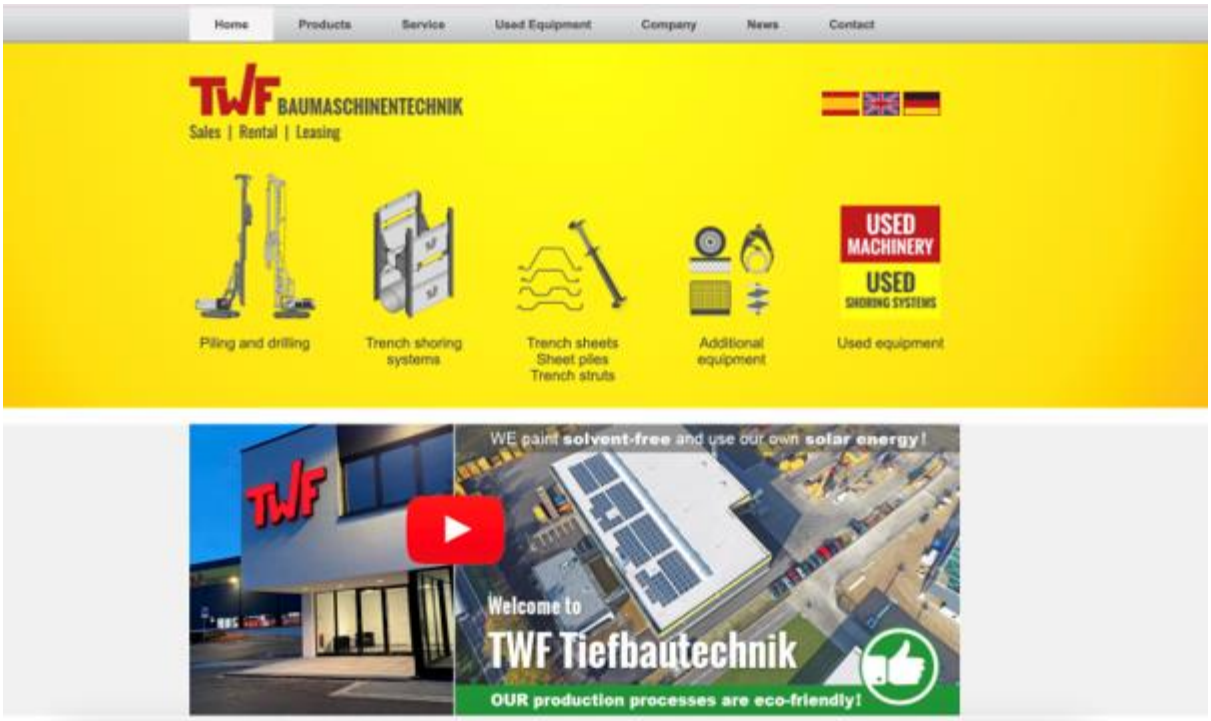
2. Δυνατότητα Παράκαμψης σε Παλαιότερα Προγράμματα Περιήγησης:

Ορισμένα προγράμματα περιήγησης μπορεί να έχουν αδυναμίες στον χειρισμό του header Content-Disposition. Αυτό θα μπορούσε να επιτρέπει στους επιτιθέμενους να παρακάμψουν την οδηγία inline και να εισάγουν scripts που εκτελούνται ακόμη και αν το περιεχόμενο προοριζόταν να εμφανιστεί inline.

- Η κεφαλίδα Content-Encoding έχει οριστεί σε "deflate", αυτό μπορεί να σημαίνει ότι ο διακομιστής είναι ευάλωτος στην επίθεση BREACH: Η επίθεση BREACH είναι ένας τύπος παραβίασης δεδομένων που μπορεί να χρησιμοποιηθεί για να κλέψει ευαίσθητες πληροφορίες από διακομιστές web.
- Ο διακομιστής χρησιμοποιεί πιστοποιητικό wildcard: *.jmix.io: Τα πιστοποιητικά wildcard μπορούν να είναι χρήσιμα για την εξασφάλιση πολλαπλών subdomains κάτω από ένα μόνο πιστοποιητικό, αλλά μπορούν επίσης να είναι πιο επικίνδυνα από τα παραδοσιακά πιστοποιητικά. Εάν το ιδιωτικό κλειδί για το πιστοποιητικό wildcard παραβιαστεί, οι επιτιθέμενοι θα μπορούσαν ενδεχομένως να το χρησιμοποιήσουν για να αποκρυπτογραφήσουν την κίνηση για οποιοδήποτε από τα subdomains που προστατεύονται από το πιστοποιητικό.
- Επιτρεπόμενες μέθοδοι HTTP: GET, HEAD, POST, PUT, PATCH, DELETE, OPTIONS: Ο διακομιστής επιτρέπει ένα ευρύ φάσμα μεθόδων HTTP, πράγμα που μπορεί να αυξήσει τον κίνδυνο επιθέσεων. Για παράδειγμα, η μέθοδος PUT θα μπορούσε να επιτρέψει σε επιτιθέμενους να ανεβάσουν κακόβουλα αρχεία στον διακομιστή, ενώ η μέθοδος DELETE θα μπορούσε να τους επιτρέψει να διαγράψουν αρχεία, εφόσον φυσικά ο κακόβουλος χρήστης αποκτούσε πρώτα κάποιου είδος εξουσιοδότηση για την εκτέλεση τέτοιων ερωτημάτων.
- OSVDB-397: Μέθοδος HTTP ('Allow Header): Η μέθοδος 'PUT' θα μπορούσε να επιτρέψει στους πελάτες να αποθηκεύουν αρχεία στο διακομιστή web.
- Μέθοδος HTTP: 'PATCH' μπορεί να επιτρέψει στον πελάτη να εκδώσει εντολές patch στον διακομιστή. Δείτε το RFC-5789.
- OSVDB-5646: Μέθοδος HTTP ('Allow' Header): Το 'DELETE' μπορεί να επιτρέψει στους πελάτες να διαγράψουν αρχεία στον διακομιστή web.

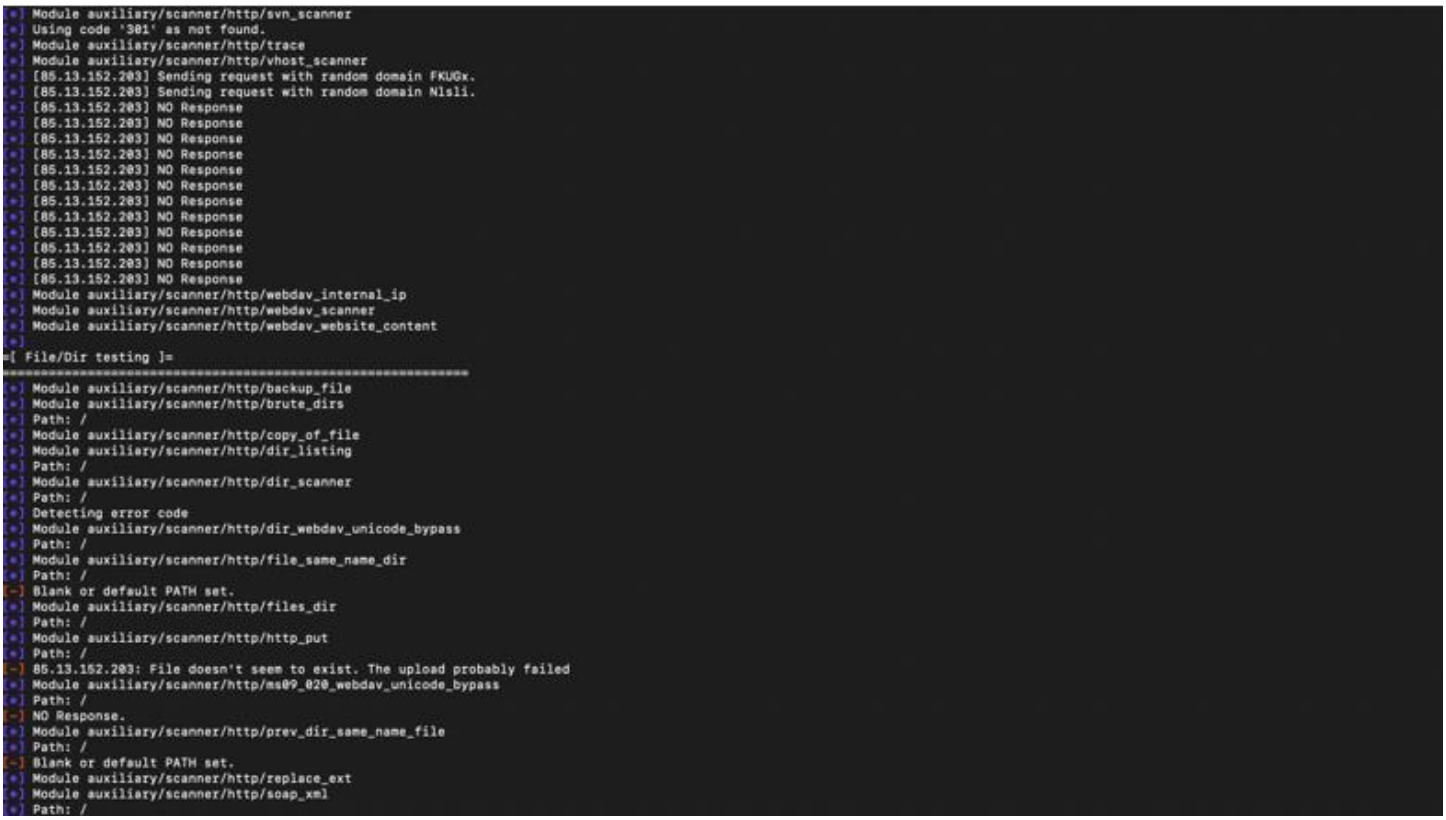
5.4.5 Δοκιμή διείσδυσης στο Struts

Για τη δοκιμή διείσδυσης στο Struts χρησιμοποιήθηκε ως στόχος δοκιμής μια απλή ιστοσελίδα ιστού με όνομα TWF (https://www.twf.at/en).

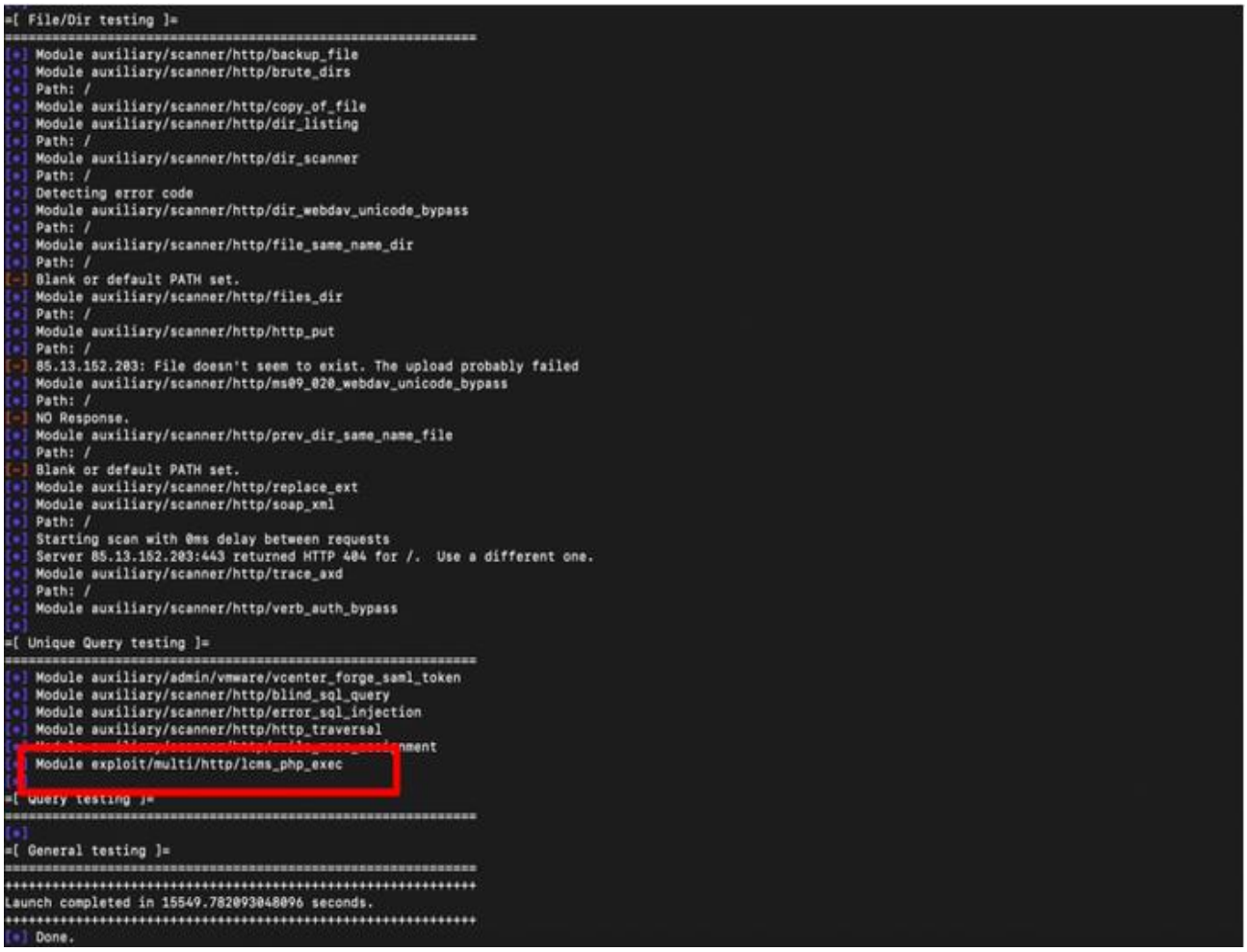


Εικόνα 83 Ιστοσελίδα TWF

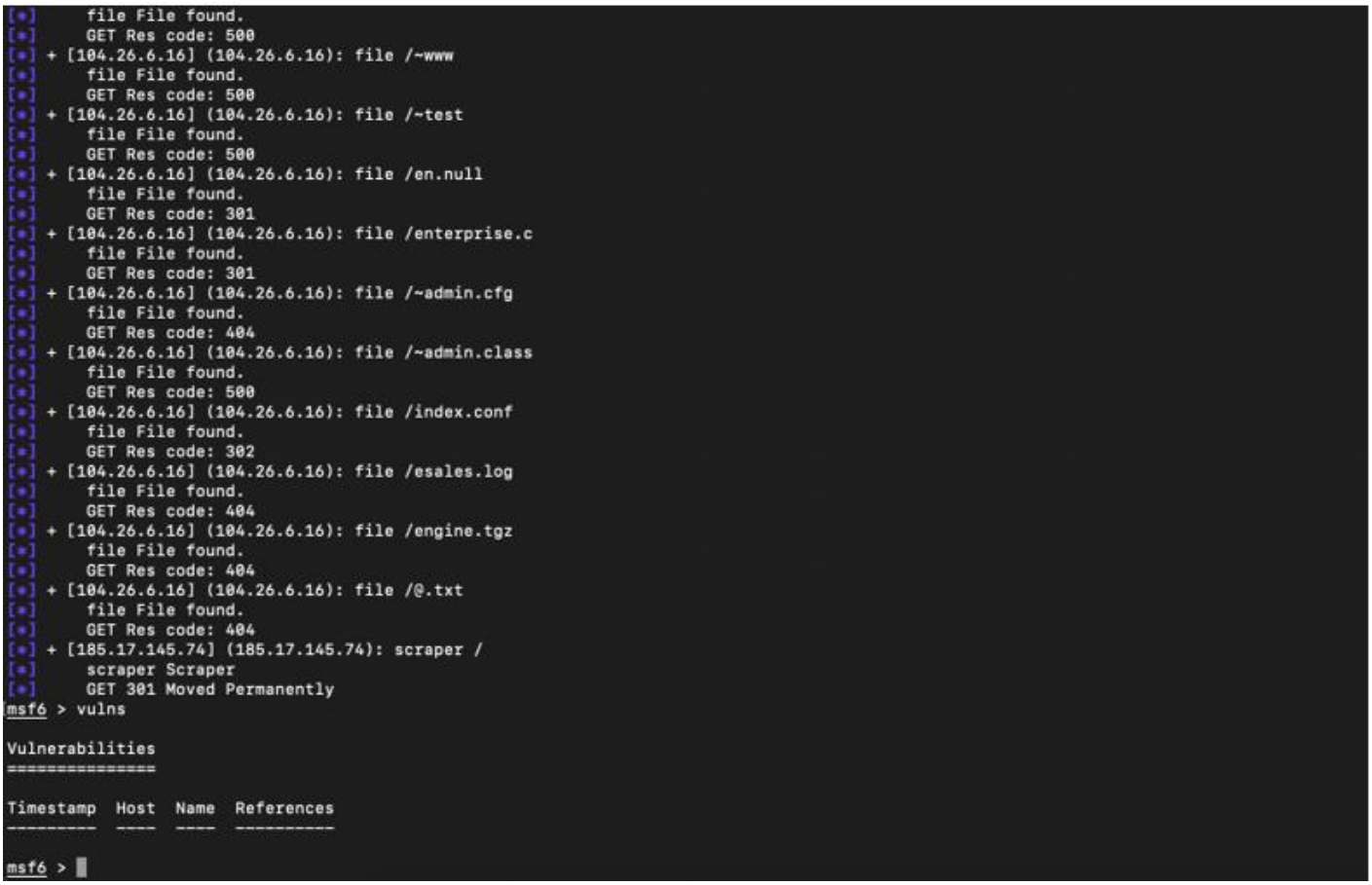
Χρησιμοποιήσαμε αρχικά το WMAP για να σκανάρουμε για ευπάθειες



Εικόνα 84 Αποτελέσματα WMAP



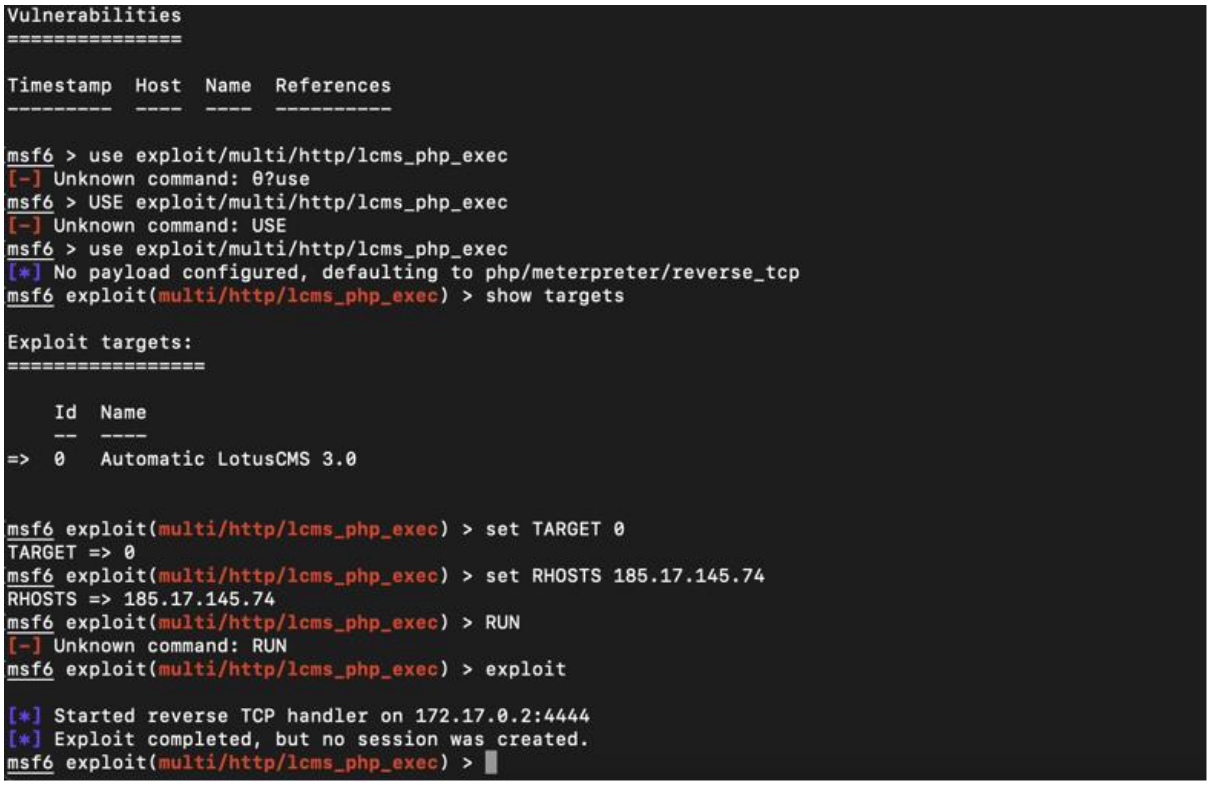
Εικόνα 85 Αποτελέσματα WMAP



Εικόνα 86 Αποτελέσματα WMAP

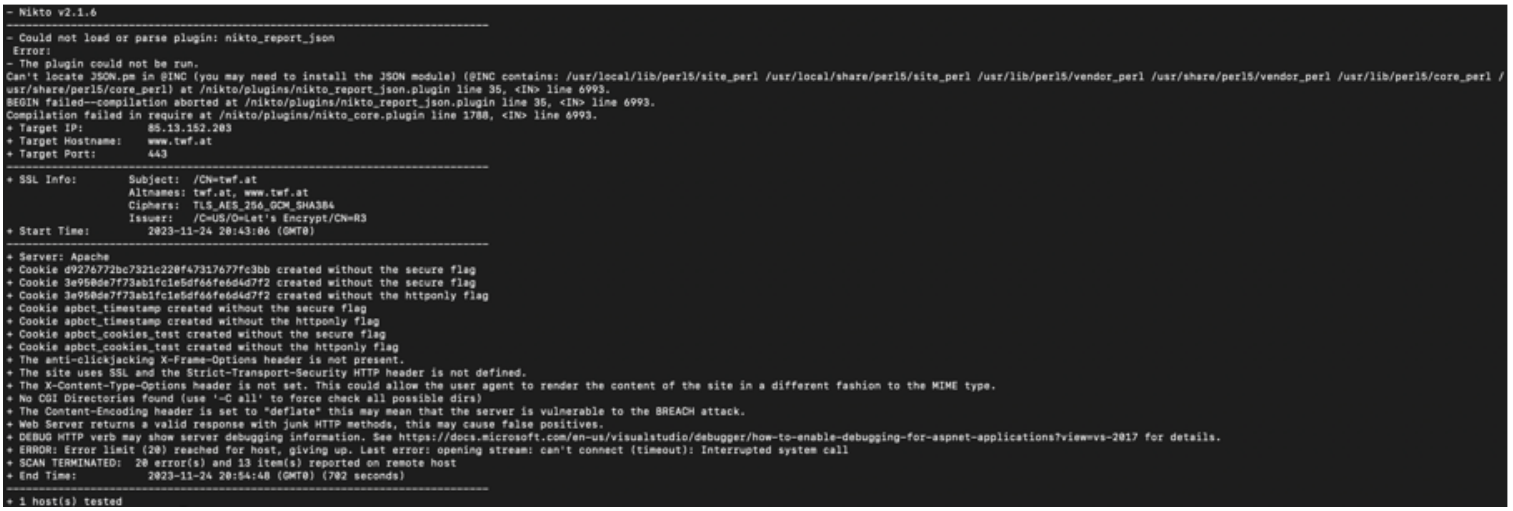
Όπως φαίνεται και στην παραπάνω εικόνα το WMAP δεν κατάφερε να βρει ευπάθειες για αυτή την ιστοσελίδα.

Χρησιμοποιήσαμε το exploit που φαίνεται στην εικόνα 84 και όπως φαίνεται και στην παρακάτω εικόνα το Metasploit μας ενημέρωσε πως η εκμετάλλευση ολοκληρώθηκε ωστόσο δεν ήταν εφικτή η δημιουργία απομακρυσμένης συνόδου με τον διακομιστή.



Εικόνα 87 Αποτελέσματα χρήσης exploits

Στη συνέχεια χρησιμοποιούμε το εργαλείο Nikto.



Εικόνα 88 Αποτελέσματα χρήσης Nikto

Τα αποτελέσματα σάρωσης Nikto υποδεικνύουν ότι ο διακομιστής προορισμού έχει αρκετές πιθανές ευπάθειες ασφαλείας. Ακολουθεί μια ανάλυση των βασικών ευρημάτων από τη σάρωση Nikto, όπως αυτά εμφανίζονται στην Εικόνα 87:

Πληροφορίες διακομιστή: Ο διακομιστής εκτελεί τον εξυπηρετητή Apache και βρίσκεται στη διεύθυνση IP 86.13.152.203. Δεν ήταν δυνατή η επίλυση του ονόματος κεντρικού υπολογιστή.

Πληροφορίες SSL: Ο διακομιστής χρησιμοποιεί SSL με αυτο-υπογεγραμμένο πιστοποιητικό. Το πιστοποιητικό υπόκειται σε ζητήματα επαλήθευσης ονόματος κεντρικού υπολογιστή και ενδέχεται να μην είναι αξιόπιστο από όλους τους πελάτες.

Cookies: Πολλά cookies ορίζονται χωρίς τη σημαία Secure, που σημαίνει ότι θα μπορούσαν να σταλούν μέσω μη κρυπτογραφημένης σύνδεσης και να υποκλαπούν από εισβολείς.

Κεφαλίδες ασφαλείας: Από τον διακομιστή λείπουν πολλές κεφαλίδες ασφαλείας, συμπεριλαμβανομένων των X-Frame-Options, X-Content-Type-Options και Strict-Transport-Security. Αυτές οι κεφαλίδες βοηθούν στον μετριασμό διαφόρων κινδύνων ασφαλείας.

Ευπάθειες: Η σάρωση εντόπισε πολλά πιθανά τρωτά σημεία, όπως:

- Η κεφαλίδα "Content-Encoding" έχει οριστεί σε "deflate", γεγονός που μπορεί να καταστήσει τον διακομιστή ευάλωτο στην επίθεση BREACH.
- Ο διακομιστής επιστρέφει μια έγκυρη απόκριση με ανεπιθύμητες μεθόδους HTTP, τις οποίες μπορεί κάποιος να εκτελέσει και να προκαλέσει προβλήματα στον ιστότοπο αποδίδοντας ψευδώς θετικά αποτελέσματα.
- Ο διακομιστής μπορεί να αποκαλύπτει πληροφορίες εντοπισμού σφαλμάτων.

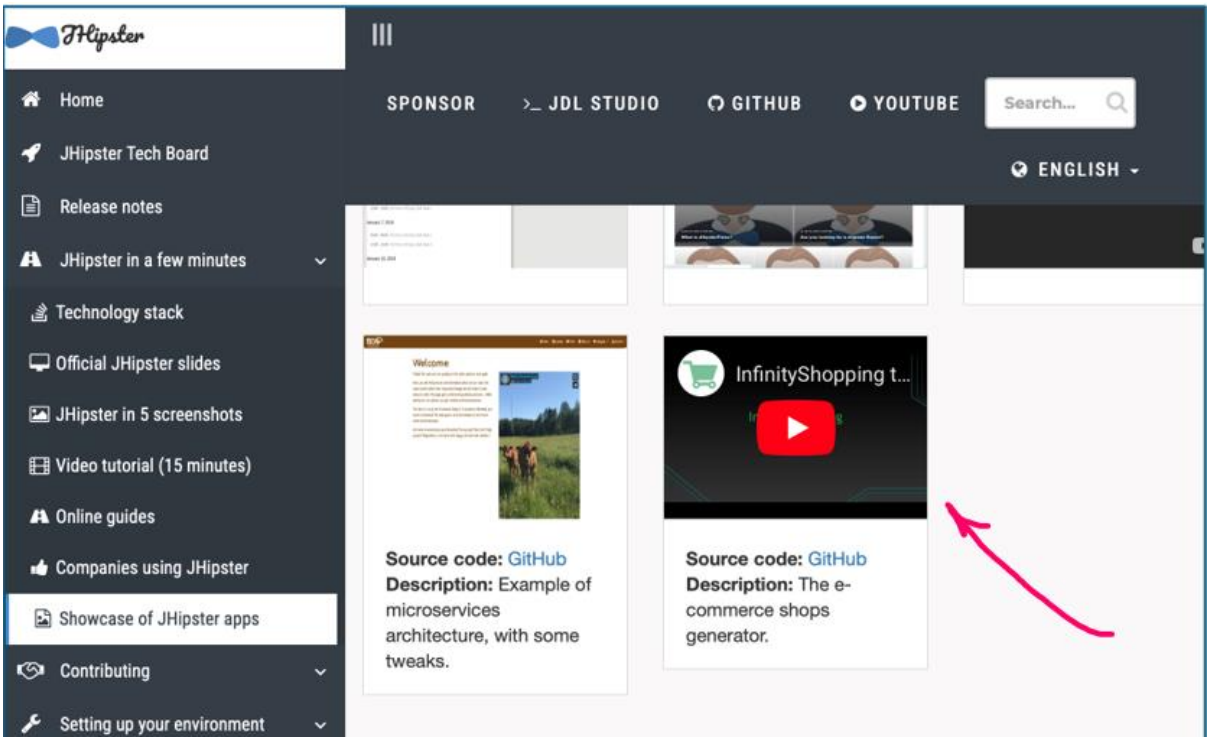
Οφείλουμε να παρατηρήσουμε ωστόσο πως αρκετά από τα αποτελέσματα του Nikto αφορούν εσφαλμένη παραμετροποίηση του Apache και όχι τόσο της ίδιας της εφαρμογής που εκτελείται σε αυτόν.

5.4.6 Δοκιμή διείσδυσης στο JHipster

Τέλος, πραγματοποιήσαμε δοκιμή διείσδυσης για το πλαίσιο JHipster. Επιλέξαμε την ιστοσελίδα Infinity Shopping[7], η οποία είναι κατασκευασμένη με μία από τις τελευταίες εκδόσεις του JHipster και πιο συγκεκριμένα την έκδοση 7.

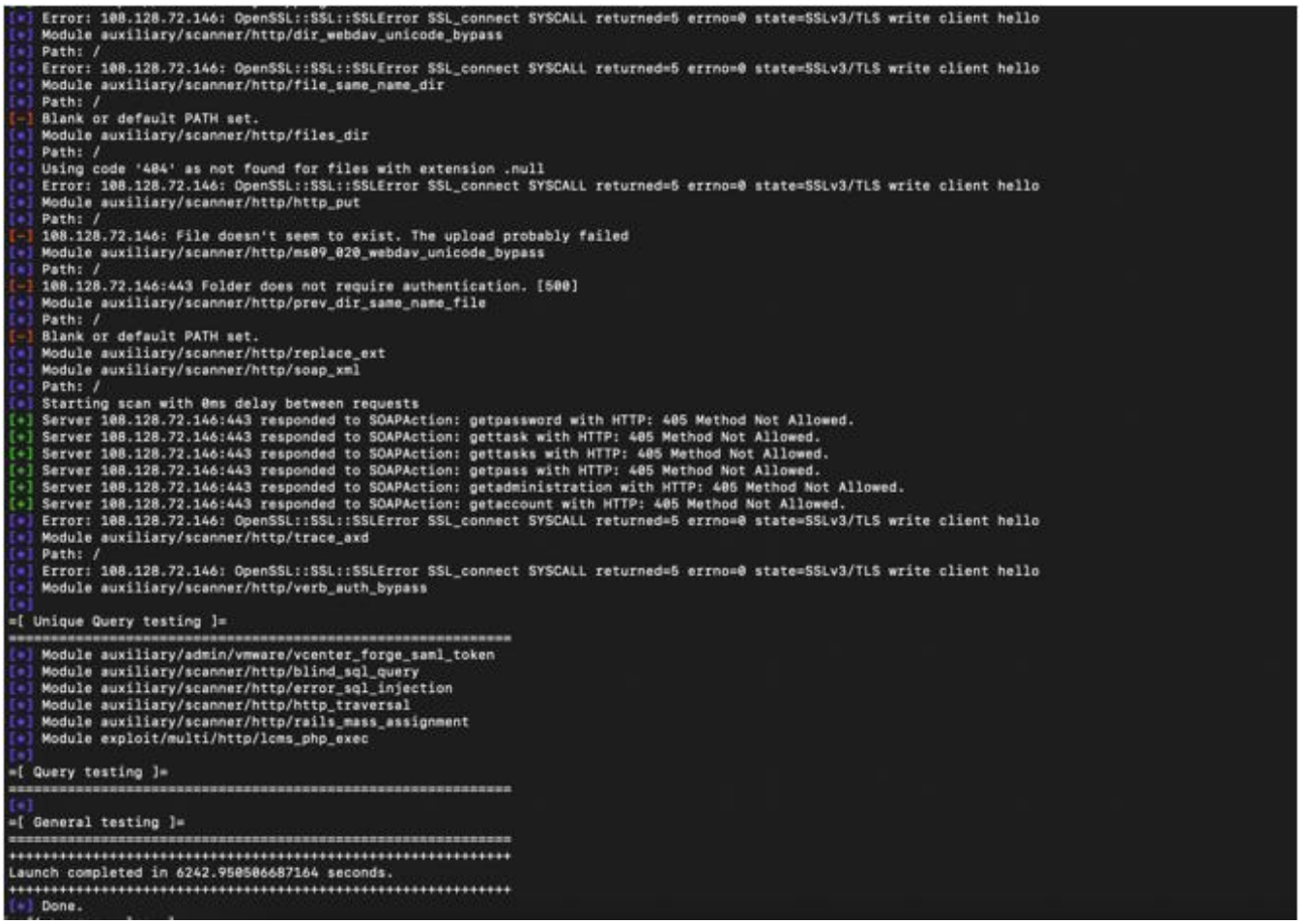
Εικόνα 89 Η Ιστοσελίδα υλοποιημένη με JHipster που ελέγχεται για ευπάθειες

[7] https://www.infinityshopping.online

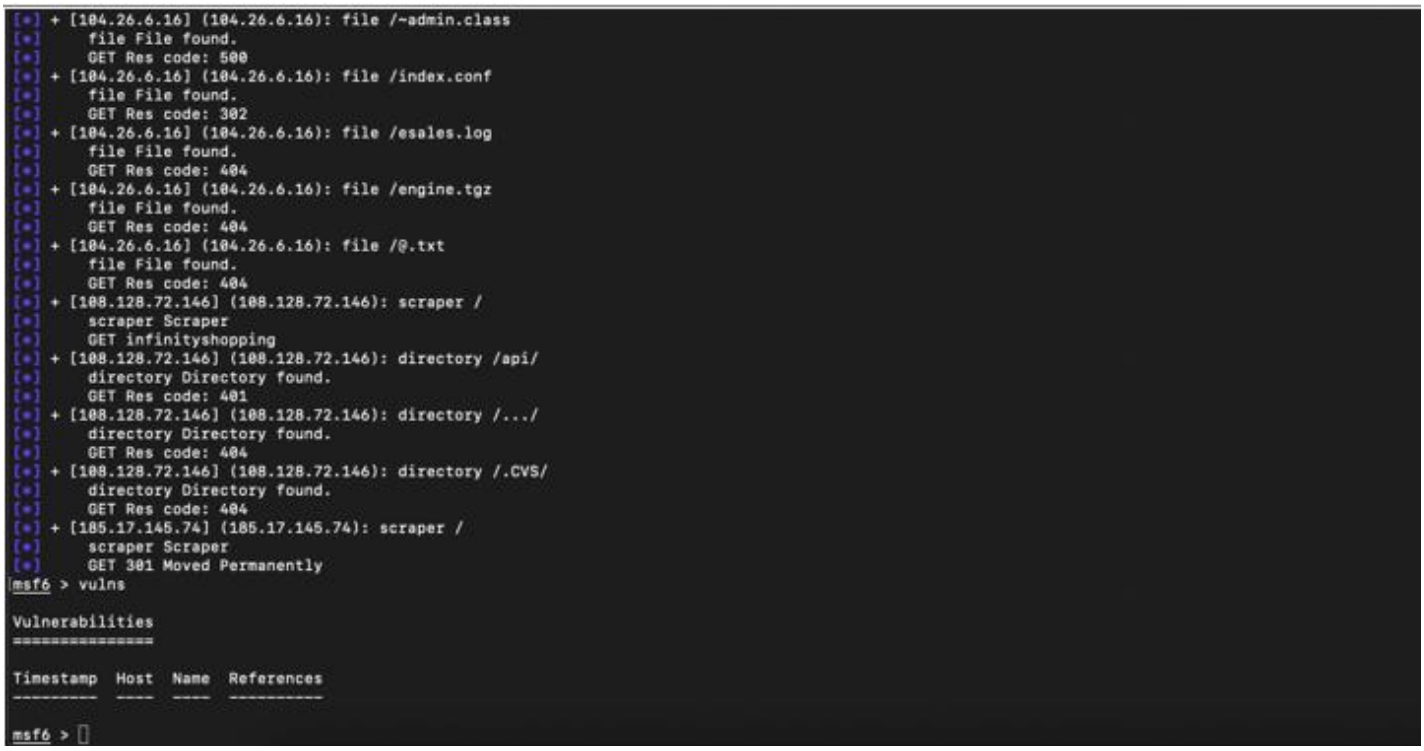


Εικόνα 90 Ιστοσελίδα υλοποιημένη με JHipster

Και σε αυτήν την περίπτωση το WMAP, δεν έδωσε αποτελέσματα για ευπάθειες.

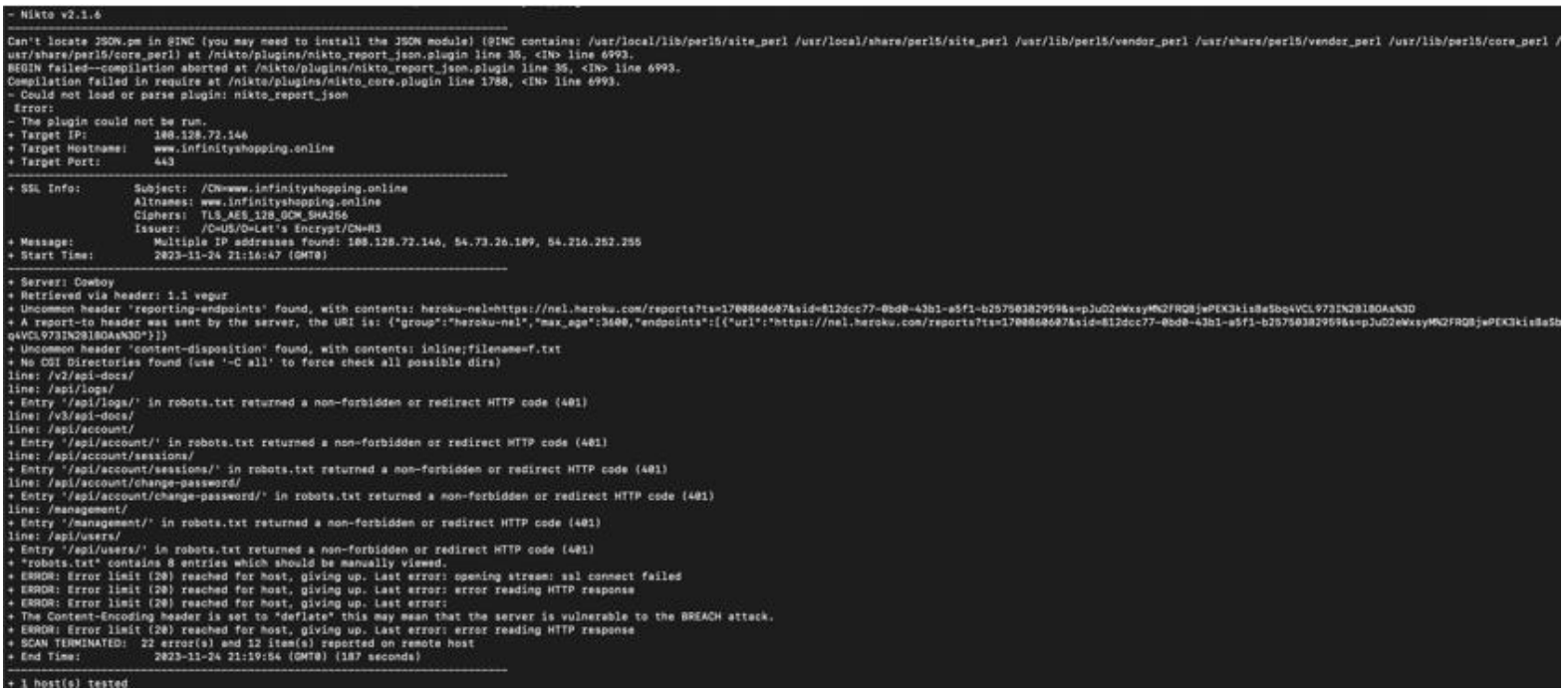


Εικόνα 91 Αποτελέσματα χρήσης WMAP



Εικόνα 92 Αποτελέσματα χρήσης WMAP

Στη συνέχεια χρησιμοποιήσαμε το Nikto



Εικόνα 93 Αποτελέσματα χρήσης Nikto

Και σε αυτήν την περίπτωση το Nikto δεν ανακάλυψε κάποια σημαντική ευπάθεια. Ωστόσο φαίνεται πως μερικά links που υπάρχουν στο αρχείο robots.txt, οδηγούν σε σφάλμα 401, το οποίο χρήζει αντιμετώπισης από τους διαχειριστές της σελίδας.

5.5 Συμπεράσματα Δοκιμών

Στον παρακάτω πίνακα έχει πραγματοποιηθεί μια ταξινόμηση των πλαισίων με βάση τις ευπάθειες που εμφάνισαν κατά την σάρωση τους με τα εργαλεία WMap και Nikto με σκοπό την εκμετάλλευσή αυτών για διείσδυση.

Πλαίσια	WMap	Nikto
JMix	-	7
Vaadin	1	1
Struts	-	3
Tapestry	1	2
JHipster	-	-
Spring-Boot	-	-

Πίνακας 19 Συμπεράσματα Δοκιμών Διείσδυσης

Στον πίνακα 20 συγκεντρώσαμε αθροιστικά τις ευπάθειες που ανιχνεύθηκαν και από τα 2 εργαλεία για να μπορέσουμε να καταλήξουμε σε περισσότερα καθολικά συμπεράσματα. Παρατηρούμε ότι το JMix είχε τις περισσότερες ευκαιρίες για διείσδυση καθιστώντας το “μη ασφαλές πλαίσιο” ενώ τα JHipster και Spring-Boot δεν εμφάνισαν καμία ευπάθεια μη επιτρέποντας τη διείσδυση, καθιστώντας τα “ασφαλή πλαίσια”.

Βέβαια, αυτή η ταξινόμηση υπόκειται στον περιορισμό πως οι ευπάθειες που εντοπίστηκαν αφορούν συγκεκριμένες εφαρμογές που υλοποιήθηκαν από τα πλαίσια αυτά. Επίσης, πολλές από τις ευπάθειες αυτές αφορούν τη διαμόρφωση του περιβάλλοντος φιλοξενίας και όχι τις ίδιες τις εφαρμογές. Από την άλλη μεριά, στο πλαίσιο μιας εφαρμογής ή ενός ιστοτόπου είναι σημαντικό να εφαρμόζονται κατάλληλα μέτρα ασφαλείας, όπως είναι η προσθήκη των απαραίτητων κεφαλίδων ασφάλειας με προκαθορισμένο τρόπο. Εδώ, λοιπόν, παρατηρούμε πως ενδεχομένως τα πλαίσια που καθορίστηκαν να έχουν τις ευπάθειες αυτές έχουν ένα μερίδιο ευθύνης για την εμφάνισή τους.

Πλαίσια	Βαθμολογία
JMix	**7**
Struts	**3**
Tapestry	**3**
Vaadin	**2**
JHipster	**-**
Spring - Boot	**-**

Πίνακας 20 Καθολικά Συμπεράσματα Δοκιμών Διείσδυσης

6. Συμπεράσματα και Μελλοντικά βήματα

6.1 Συμπεράσματα

Η παρούσα διπλωματική εργασία αντιπροσωπεύει μια εκτενή και συστηματική προσέγγιση στην αξιολόγηση και βελτίωση της ασφάλειας εφαρμογών ιστού που χρησιμοποιούν πλαίσια ανάπτυξης εφαρμογών ιστού ανοικτού κώδικα σε συνδυασμό με τη γλώσσα προγραμματισμού Java.

Μέσα από την επιλογή κατάλληλων πλαισίων, τη θεωρητική ανάλυση των χαρακτηριστικών ασφάλειας και την πρακτική αξιολόγηση μέσω ανιχνευτών τρωτοτήτων, καταλήξαμε σε σημαντικά ευρήματα. Η σύγκριση των πλαισίων από την άποψη των τρωτοτήτων και του επιπέδου ρίσκου παρείχε σημαντική εισήγηση για την επιλογή και βελτίωση της ασφάλειας των εφαρμογών.

Τα αποτελέσματα των ανιχνευτών τρωτοτήτων έκριναν ως πιο ασφαλές πλαίσιο το JHipster, καθώς ήταν το μοναδικό που δεν εμφάνισε καμία ευπάθεια κατά την αξιολόγηση και με τα 3 εργαλεία ανίχνευσης. Από την άλλη πλευρά το πλαίσιο που κρίθηκε ως το λιγότερο ασφαλές ήταν το Spring το οποίο εμφάνισε συνολικά 6 κρίσιμες τρωτότητες.

Η δοκιμή διείσδυσης προσφέρει τη δυνατότητα περαιτέρω ενίσχυσης της ασφάλειας των εφαρμογών. Ο συνδυασμός θεωρητικής έρευνας και πρακτικού πειραματισμού έχει ως στόχο τη δημιουργία ολοκληρωμένης και αποτελεσματικής στρατηγικής ασφάλειας.

Σε συνολικό επίπεδο, το πλαίσιο που είναι πιο ασφαλές είναι το JHipster, το οποίο κατάφερε να βγει “νικητής” και κατά την ανίχνευση τρωτοτήτων, εμφανίζοντας μηδενικές τρωτότητες, αλλά και κατά την δοκιμή διείσδυσης, όπου παρά τις απόπειρες δεν επιτεύχθηκε κάποια διείσδυση στη σχετική εφαρμογή που ελέγχθηκε και υλοποιήθηκε μέσω αυτού.

Από την άλλη μεριά, το Jmix ήταν αυτό που εμφανίστηκε να παρουσιάζει τις περισσότερες κρίσιμες τρωτότητες, στο σύνολο 7, καθορίζοντάς το, το λιγότερο ασφαλές πλαίσιο. Παρ’ όλα αυτά καμία από τις ευπάθειες αυτές δεν ήταν τόσο κρίσιμη ώστε να επιτευχθεί διείσδυση στην εφαρμογή του.

Παρά το γεγονός ότι η ανίχνευση ευπαθειών έκρινε ως ασφαλέστερο πλαίσιο το JHipster, οφείλουμε να παραδεχτούμε ότι όλα τα πλαίσια είναι πολύ ασφαλή και χρησιμοποιούνται από μεγάλους οργανισμούς. Το γεγονός ότι το JHipster κρίθηκε πιο ασφαλές σε σχέση με το Spring-Boot για παράδειγμα θα μπορούσαμε να πούμε ότι οφείλεται στο γεγονός ότι το spring-Boot αποτελείται από πολύ περισσότερες γραμμές κώδικα σε σχέση με το JHipster, επομένως είναι λογικό να εμφανίσει περισσότερες ευπάθειες. Αυτό όμως δεν το καθιστά λιγότερο ασφαλές καθώς οι ευπάθειες που βρήκαμε αφορούν κώδικα δοκιμής, επομένως δε μπορούν να εκμεταλλευθούν για να γίνει επίθεση. Συμπερασματικά, το Spring-Boot εφόσον προσφέρει πολύ περισσότερες λειτουργίες θεωρούμε ότι είναι το πιο ισχυρό πλαίσιο παρά τις ευπάθειες που εντοπίστηκαν.

Τα παραπάνω ευρήματα προσφέρουν σημαντική συνεισφορά στον τομέα της ασφάλειας εφαρμογών ιστού, παρέχοντας πολύτιμες κατευθυντήριες γραμμές για τους προγραμματιστές και τους ερευνητές που ενδιαφέρονται για τη βελτίωση της ασφάλειας στον ψηφιακό χώρο.

Με αυτές τις παρατηρήσεις και ευρήματα, ελπίζουμε πως η παρούσα διπλωματική εργασία θα συνεισφέρει στη συνεχή προοδευτική εξέλιξη του τομέα και θα ενισχύσει τον κοινό σκοπό για πιο ασφαλείς και αξιόπιστες εφαρμογές ιστού.

6.2 Μελλοντικά Βήματα

Για τα μελλοντικά βήματα της εργασίας, υπάρχουν πολλές ενδιαφέρουσες προοπτικές που μπορούν να ενισχύσουν τη σφαιρική κατανόηση και εφαρμογή των αποτελεσμάτων.

Καταρχάς, προτείνεται η επέκταση της ανάλυσης ασφάλειας με επιπλέον βάρος στην αποτελεσματικότητα των χαρακτηριστικών ασφαλείας κατά τη διάρκεια πραγματικής χρήσης. Αυτό θα επιτρέψει μια πιο ολοκληρωμένη κατανόηση του πώς αντιμετωπίζονται από τους χρήστες και ποια ζητήματα ασφαλείας είναι πιο κρίσιμα.

Επιπλέον, μια συγκριτική ανάλυση πλαισίων με περισσότερα κριτήρια, συμπεριλαμβανομένης της απόδοσης, της ευελιξίας και εν γένει της ποιότητάς τους, θα παρείχε πιο ολοκληρωμένη εικόνα της καταλληλότητάς τους.

Τέλος, η συνεχής παρακολούθηση και αναθεώρηση των πλαισίων, καθώς και η εφαρμογή των αποτελεσμάτων στην πραγματική ανάπτυξη εφαρμογών, θα εξασφαλίσουν την ενημέρωση και την προσαρμογή στην εξέλιξη των απαιτήσεων ασφαλείας.

Με αυτό τον τρόπο, μπορούμε να διασφαλίσουμε ότι η έρευνα στην ασφάλεια των εφαρμογών ιστού που βασίζονται σε Java προχωρά σε ενδιαφέρουσες και παραγωγικές κατευθύνσεις.

7. Βιβλιογραφία

[1] A. Aborujilah, J. Adamu, S. M. Shariff and Z. Awang Long, "Descriptive Analysis of Built-in Security Features in Web Development Frameworks," 2022 16th International Conference on Ubiquitous Information Management and Communication (IMCOM), Seoul, Korea, Republic of, 2022, pp. 1-8
[2] Finifter, Matthew. “Exploring the Relationship Between Web Application Development Tools and Security.” USENIX Conference on Web Application Development (2011)
[3] Bryan Sullivan and Vincent Liu. 2011. Web Application Security, A Beginner's Guide (1st. ed.). McGraw-Hill Education Group.
[4] David Kennedy, Jim O'Gorman, Devon Kearns, and Mati Aharoni. 2011. Metasploit: The Penetration Tester's Guide (1st. ed.). No Starch Press, USA.
[5] Y. Makino and V. Klyuev, "Evaluation of web vulnerability scanners," 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Warsaw, Poland, 2015, pp. 399-402
[6] https://www.cve.org/
[7] https://www.g2.com/categories/java-web-frameworks
[8] Beale, J. (2019). Penetration Testing: A Hands-On Introduction to Hacking. San Francisco, CA: No Starch Press