ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΙΓΑΙΟΥ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΑΚΩΝ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΑΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Υπηρεσία Ανίχνευσης Τρωτοτήτων Εφαρμογών Υπολογιστικού Νέφους

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Αραπαντζή Βασίλειου-Παναγιώτη

Επιβλέπων: Κρητικός Κυριάκος (Αν. Καθηγητής)

Μέλη εξεταστικής επιτροπής: Καρύδα Μαρία (Καθηγήτρια), Στεργιόπουλος Γεώργιος (Επ. Καθηγητής)

Σάμος, Δεκέμβριος 2023

ΠΡΟΛΟΓΟΣ ΚΑΙ ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα εκφράσω τις ευχαριστίες μου στον επιβλέποντα καθηγητή μου, κ. Κυριάκο Κρητικό, για τη βοήθεια και τις χρήσιμες ιδέες του, που συνέβαλαν στην βελτίωση της εργασίας, καθώς και για τις γνώσεις που μου προσέφερε κατά την διδασκαλία του μαθήματος Ασφάλειας Πληροφοριακών Συστημάτων και Δικτύων Υπολογιστών. Ευχαριστώ επίσης τον καθηγητή κ. Γεώργιο Καμπουράκη, χάριν στον οποίο είχα τις βάσεις για τη συγγραφή της εργασίας, έχοντας την ευκαιρία να παρακολουθήσω μαθήματα Ασφάλειας κύκλου τα προηγούμενα χρόνια. Ευχαριστώ τους καθηγητές της σχολής που συνέβαλαν στην απόκτηση των απαραίτητων γνώσεων για την επιτυχή φοίτησή μου και την εκπόνηση της πτυχιακής εργασίας, αλλά κυρίως που ενίσχυσαν την αγάπη μου για τον κλάδο της Πληροφορικής, που υπήρχε από τα παιδικά μου χρόνια. Περισσότερο από όλους, οφείλω να ευχαριστήσω την οικογένεια μου και τους φίλους μου που μου συμπαραστάθηκαν και με βοήθησαν σε αυτήν την ακαδημαϊκή μου πορεία. Τους ευχαριστώ που στάθηκαν δίπλα μου όλα αυτά τα χρόνια και για την υπομονή και εμπιστοσύνη που υπέδειξαν.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΛΙΣΤΑ ΣΧΗΜΑΤΩΝ

ΛΙΣΤΑ ΠΙΝΑΚΩΝ

ΣΥΝΟΨΗ

Βασικό αντικείμενο αυτής της εργασίας είναι η δημιουργία μιας RESTful υπηρεσίας ιστού (web service) για την σύνθεση εργαλείων ανίχνευσης τρωτοτήτων (vulnerability scanning tools) με σκοπό την μεγαλύτερη ακρίβεια στην ανίχνευση διαφορετικών ειδών τρωτοτήτων σε εφαρμογές ιστού (web applications). Η πρόκληση που υπήρξε αφορμή για την δημιουργία της συγκεκριμένης υπηρεσίας, ήταν το γεγονός ότι ελάχιστα εργαλεία από μόνα τους παρέχουν κάλυψη για όλες τις ευπάθειες που μπορεί να έχει κάποιο συστατικό μέρος μιας εφαρμογής ή ολόκληρη η εφαρμογή. Η σύνθεση, λοιπόν, πολλαπλών εργαλείων ανίχνευσης λύνει σε ένα σημαντικό ποσοστό το πρόβλημα αυτό. Η υπηρεσία που αναπτύχθηκε συνδυάζει διαφορετικά είδη εργαλείων ανίχνευσης τρωτοτήτων και έπειτα παράγει ένα τελικό αποτέλεσμα σε μορφή αναφοράς. Τα εργαλεία ανίχνευσης τρωτοτήτων που επιλέχθηκαν και χρησιμοποιούνται από την υπηρεσία είναι ανοικτού κώδικα και συνδυασμένα υποστηρίζουν τόσο στατική όσο και δυναμική ανάλυση/ανίχνευση τρωτοτήτων. Η αποτίμηση (evaluation) της υπηρεσίας που αναπτύχθηκε δείχνει ότι η σύνθεση πολλαπλών εργαλείων ανίχνευσης μπορεί πράγματι να οδηγήσει στην ανίχνευση τρωτοτήτων με μεγαλύτερη ακρίβεια. Τέλος, τονίζεται πως εκτός από την υπηρεσία την ίδια, αναπτύχθηκε μια ολοκληρωμένη εφαρμογή ιστού, η οποία επιτρέπει την απομακρυσμένη εκμετάλλευση της λειτουργικότητας της υπηρεσίας από χρήστες (και όχι μόνο από προγράμματα ή πράκτορες).

Λέξεις Κλειδιά: *υπηρεσία ιστού, REST, Java Spring, στατική ανάλυση, δυναμική ανάλυση, εργαλεία ανίχνευσης τρωτοτήτων, ακρίβεια ανίχνευσης τρωτοτήτων*

ABSTRACT

The main object of this work is the creation of a RESTful web service for the synthesis of vulnerability scanning tools in order to increase the accuracy in the detection of different types of vulnerabilities in web applications. The challenge that led to the creation of this service was the fact that the vulnerability scanning tools cannot individually provide complete coverage for all the vulnerabilities that any component of an application or the entire application may have. Therefore, the composition of multiple scanning tools solves this problem to a significant extent. The RESTful service developed combines different kinds of vulnerability detection tools while has the ability to produce a single, final report per scanning request. The vulnerability detection tools selected and used by the service are open source. In addition, they support in combination both static and dynamic vulnerability analysis/detection. The evaluation of the developed service shows that the synthesis of multiple detection tools can indeed lead to a more accurate vulnerability detection. Finally, it must be noted that the capabilities of this service have been wrapped into the form of a web-based application to ease its use also by normal users (and not just other programs or agents).

Key Words: *Web service, REST, Java Spring, static analysis, dynamic analysis, vulnerability scanning tools, vulnerability scanning accuracy, precision, recall*

ΚΕΦΑΛΑΙΟ 1 – ΕΙΣΑΓΩΓΗ

Όσο η τεχνολογία αναπτύσσεται και εξελίσσεται, εμφανίζονται όλο και περισσότεροι κίνδυνοι και αδυναμίες στα πληροφοριακά συστήματα και τις εφαρμογές που δημιουργούνται [12]. Επομένως, ο χώρος στον οποίο εστιάζει το θέμα της παρούσα εργασίας είναι η ασφάλεια πληροφοριακών συστημάτων, η οποία στηρίζεται σε τρεις βασικές αρχές:

- Ακεραιότητα

Η *ακεραιότητα* (integrity) αναφέρεται στη διατήρηση των δεδομένων, προγραμμάτων, και άλλων ειδών πόρων ενός πληροφοριακού συστήματος σε μια γνωστή, έγκυρη κατάσταση χωρίς ανεπιθύμητες τροποποιήσεις, αφαιρέσεις ή προσθήκες από μη εξουσιοδοτημένα άτομα [17].

- Διαθεσιμότητα

Η δ*ιαθεσιμότητα* (availability) δεδομένων και υπολογιστικών πόρων σημαίνει την εξασφάλιση ότι τα δεδομένα και οι πόροι αυτοί θα είναι στη διάθεση των χρηστών [17] όποτε απαιτείται η χρήση τους[1].

Μία τυπική απειλή που αντιμετωπίζουν τα σύγχρονα πληροφοριακά συστήματα είναι η επίθεση άρνησης υπηρεσίας (Denial of Service - DoS), η οποία έχει ως σκοπό να τεθούν εκτός λειτουργίας οι στοχευμένοι πόροι, είτε προσωρινά, είτε μόνιμα.

- Εμπιστευτικότητα

Η *εμπιστευτικότητα* (confidentiality) σημαίνει ότι ευαίσθητες πληροφορίες δεν θα πρέπει να αποκαλύπτονται σε μη εξουσιοδοτημένα άτομα[1].

Πλέον, οι επιτιθέμενοι διαθέτουν πολλούς τρόπους για να υποκλέψουν στοιχεία και δεδομένα από μια εφαρμογή, είτε αυτή βρίσκεται σε περιβάλλον παραγωγής, είτε στα πρώιμα στάδια υλοποίησής της. Βάσει έρευνας [6], διαπιστώθηκε ότι το 75% όλων των εφαρμογών έχουν τουλάχιστον ένα ελάττωμα ασφάλειας, με το 24% αυτών να έχουν τουλάχιστον ένα ελάττωμα υψηλής επικινδυνότητας.

[1] https://el.wikipedia.org/wiki/%CE%91%CF%83%CF%86%CE%AC%CE%BB%CE%B5%CE%B9%CE%B1_%CF%80%CE%BB%CE%B7%CF%81%CE%BF%CF%86%CE%BF%CF%81%CE%B9%CE%B1%CE%BA%CF%8E%CE%BD_%CF%83%CF%85%CF%83%CF%84%CE%B7%CE%BC%CE%AC%CF%84%CF%89%CE%BD

Η ασφάλεια ενός πληροφοριακού συστήματος ή μιας εφαρμογής εξαρτάται τελικά από το πόσο πιο γρήγορα και νωρίτερα μπορεί κάποιος να εντοπίσει και να διορθώσει ζητήματα ασφάλειας στη διαδικασία ανάπτυξης του αντίστοιχου λογισμικού. Ένα πολύ συνηθισμένο σφάλμα κώδικα (πχ. μη έλεγχος δεδομένων εισόδου) κατά την διαδικασία της συγγραφής κώδικα μπορεί να οδηγήσει σε επιθέσεις υπερχείλισης (overflow attacks) που μπορεί να επιτρέψουν σε επιτιθέμενους να εκχύσουν κακόβουλο λογισμικό στο περιβάλλον εκτέλεσης της εφαρμογής ιστού.

Επομένως, είναι επιτακτική η χρήση εργαλείων ασφαλείας εφαρμογών (π.χ. Arachni[2], Owasp Zap[3], Insider[4], NodeJsScan[5]) που να ενσωματώνονται στο περιβάλλον ανάπτυξης των εφαρμογών. Αυτά τα εργαλεία ασφαλείας μπορούν τελικά να εξοικονομήσουν χρόνο και έξοδα. Κι αυτό διότι η σάρωση ευπαθειών (όπως μπορεί να υποστηριχθεί από ένα εργαλείο ασφάλειας) μετριάζει τους κινδύνους παραβίασης δεδομένων, η οποία θα συνεπάγεται μια σειρά κοστών, συμπεριλαμβανομένης της αποκατάστασης και της απώλειας πελατών ως αποτέλεσμα της ζημιάς στη φήμη, εντοπίζοντας προβλήματα σε πολύ πρώιμο στάδιο, μάλιστα πριν ακόμη γίνει η διάθεση της αντίστοιχης εφαρμογής στην αγορά.

Ορισμένα από τα προβλήματα που μπορούν να δημιουργηθούν, σε περίπτωση εκμετάλλευσης τυχόν ευπαθειών από επιτιθέμενους, είναι τα εξής:

- Ύπαρξη κάποιου κακόβουλου λογισμικό που χρησιμοποιείται για να μολύνει ιστότοπους, να συλλέγει δεδομένα και να κλέβει πόρους υπολογιστών.

Ο αριθμός των ιστοτόπων που έχουν παραβιαστεί αυξάνεται ραγδαία [6]. Επίσης, το ίδιο ισχύει για τα δεδομένα που υποκλέπτονται. Η αποκάλυψη ευαίσθητων δεδομένων έχει επίσης το αρνητικό αντίκτυπο μη συμμόρφωση με νόμους και κανονισμούς όπως τον GDPR.

- Η παραβίαση της ασφάλειας οδηγεί σε απώλεια επιχειρηματικής φήμης, μείωση πελατών και πτώση των εσόδων

Όταν μια εφαρμογή ιστού παραβιάζεται, ένας πελάτης χάνει την εμπιστοσύνη του. Αυτό, ως αποτέλεσμα, μπορεί να οδηγήσει σε απώλεια φήμης στον πάροχο της εφαρμογής και μπορεί να

[2] https://github.com/Arachni/arachni
[3] https://www.zaproxy.org/
[4] https://github.com/insidersec/insider
[5] https://github.com/ajinabraham/nodejsscan

σημαίνει το τέλος της επιχείρησής του, εφόσον το πελατολόγιό του μειωθεί στο ελάχιστο. Η ζημιά αυτή μπορεί να απαιτήσει την κατανάλωση αρκετού χρόνου και χρήματος για να αποκατασταθεί/διορθωθεί.

- Ο καθαρισμός ενός ιστότοπου ή μιας εφαρμογής ιστού κοστίζει περισσότερο από την πρόληψη απειλών

Η αφαίρεση ενός κακόβουλου λογισμικού μπορεί να πάρει χρόνο καθώς και να κοστίσει πολύ. Αποτελεί μια δύσκολη διαδικασία και μπορεί να μην είναι πάντοτε εφικτή (πχ. σε επιθέσεις ramsonware αν δεν γίνονται backup δεδομένα & προγράμματα).

Ο ιστότοπος μπαίνει στη μαύρη λίστα λόγω επίθεσης

Η εισαγωγή ενός ιστότοπου σε μαύρη λίστα (αποκλεισμένων ή περιορισμένων ιστοτόπων) αποτελεί μια πρακτική για την αποτροπή περαιτέρω κακόβουλης δραστηριότητας ή για τον μετριασμό των επιπτώσεων μιας παραβίασης ασφάλειας. Αυτή η πρακτική χρησιμοποιείται από διαχειριστές δικτύου, ομάδες ασφαλείας και παρόχους φιλοξενίας Ιστού για την προστασία των συστημάτων και των χρηστών τους.

Όταν ένας ιστότοπος βρίσκεται στη μαύρη λίστα, η μηχανή αναζήτησης αποβάλλει τον ιστότοπο από τα αποτελέσματα αναζήτησης. Ένας ιστότοπος χάνει το 95% της οργανικής του επισκεψιμότητας όταν μπαίνει στη μαύρη λίστα [18], κάτι που θα επηρεάσει τελικά τα έσοδα του οργανισμού που τον προσφέρει.

Η παρούσα διπλωματική εστιάζει στον εντοπισμό των παραπάνω κινδύνων σε διάφορες φάσεις της διαδικασίας ανάπτυξης λογισμικού μέσω της κατασκευής μιας RESTful υπηρεσίας που είναι ικανή να συνδυάζει και να εφαρμόζει εργαλεία ανίχνευσης τρωτοτήτων πάνω στο λογισμικό. Κατά την υλοποίηση, εστιάζει στην χρήση στατικών εργαλείων ανίχνευσης πάνω στον κώδικα του λογισμικού. Κατά την δοκιμή ασφάλειας, εστιάζει στον έλεγχο του λογισμικού για ευπάθειες μέσω της χρήσης δυναμικών εργαλείων ανάλυσης. Η δυναμική ανάλυση μπορεί να χρησιμοποιηθεί και σε λογισμικό/εφαρμογές ιστού που είναι ήδη σε περιβάλλον παραγωγής, αν και το κόστος αντιμετώπισης των ανιχνευμένων ευπαθειών θα είναι μεγαλύτερο ενώ ενδέχεται να επηρεαστεί η απόδοση των εφαρμογών κατά την ανάλυση. Μετά την χρήση της υπηρεσίας, ο χρήστης μπορεί γρήγορα και εύκολα μέσα των αναφορών που παράχθηκαν από τον συνδυασμό πολλαπλών εργαλείων, οι οποίες συνδυάζονται σε μια ολική αναφορά, να δει αναλυτικά τις ευπάθειες που

υπάρχουν, τα σημεία όπου αυτές βρίσκονται στον κώδικα, και την αποτίμηση του βαθμού επικινδυνότητάς τους. Εκτός από την ίδια την υπηρεσία που μπορεί να ενοποιηθεί με άλλες εφαρμογές μέσω του API της καθώς και να εκτελεστεί από άλλα προγράμματα και πράκτορες, έχει αναπτυχθεί και μια εφαρμογή ιστού που την περιτυλίγει ώστε να παρέχεται κατάλληλη και εύχρηστη πρόσβαση και σε ανθρώπινους χρήστες.

Η χρήση πολλαπλών εργαλείων ανίχνευσης έγκειται στο γεγονός πως η συνδυασμένη εκτέλεσή τους οδηγεί σε μεγαλύτερη ακρίβεια ανίχνευσης σε σχέση με την μεμονωμένη εκμετάλλευση των εργαλείων αυτών. Ένα επιπλέον ενδιαφέρον χαρακτηριστικό της υλοποιημένης υπηρεσίας είναι πως λογαριάζει τις προτιμήσεις του χρήστη για την παραγωγή του τελικού συνδυασμού εργαλείων προς χρήση (πχ. ο χρήστης μπορεί να αιτηθεί μόνο στατική ανάλυση ή να ζητήσει την χρήση ενός συγκεκριμένου εργαλείου μόνο για κάθε είδος ανάλυσης). Τέλος, επιτρέπει στον χρήστη να αναζητήσει προηγούμενες αναλύσεις και να δει τις λεπτομέρειές τους. Αυτό μπορεί να είναι χρήσιμο για διάφορους λόγους. Για παράδειγμα, ο χρήστης μπορεί να παρακολουθεί την πορεία βελτίωσης της εφαρμογής του ως προς το επίπεδο της ασφάλειας που προσφέρει (πχ. από μια ανάλυση με πολλές τρωτότητες η εφαρμογή μεταβαίνει σε μια ανάλυση με πολύ μικρότερο αριθμό από τρωτότητες).

Ο χρήστης, έχοντας στην διάθεσή του μια λεπτομερή λίστα από τις ευπάθειες που βρέθηκαν από την προτεινόμενη RESTful υπηρεσία, μπορεί να προχωρήσει σε διορθώσεις στον κώδικα της εφαρμογής του καθώς και σε άλλες σχετικές ενέργειες [19]:

- ***Τεκμηρίωση και ιεράρχηση:*** Δημιουργία μιας ολοκληρωμένης λίστας με όλα τα τρωτά σημεία που ανακαλύφθηκαν κατά τη διαδικασία ανίχνευσης. Ταξινόμηση με βάση τη σοβαρότητα ή το επίπεδο κινδύνου ώστε να εστιαστούν πρώτα τα κρίσιμα τρωτά σημεία. Η RESTful υπηρεσία ανίχνευσης παρέχει προς αυτό το σκοπό πληροφορία επικινδυνότητας για την κάθε εντοπισμένη τρωτότητα.

- ***Κατανόηση των τρωτών σημείων:*** Ο χρήστης θα πρέπει να συγκεντρώσει λεπτομερείς πληροφορίες για κάθε ευπάθεια, συμπεριλαμβανομένων των επιπτώσεών της, των επηρεαζόμενων συστημάτων ή στοιχείων και των πιθανών εκμεταλλεύσεων. Τα εργαλεία που έχουν χρησιμοποιηθεί στην RESTful υπηρεσία ανίχνευσης δίνουν μια καλή περιγραφή των τρωτοτήτων που βρέθηκαν, καθώς και τρόπους αντιμετώπισης. Σε πολλές περιπτώσεις,

παρέχουν και συνδέσμους για ανακάλυψη περαιτέρω λεπτομερειών για τις τρωτότητες αυτές.

- ***Ανάπτυξη σχεδίου δράσης:*** Δημιουργία ενός σχεδίου για την αντιμετώπιση κάθε ευπάθειας. Αυτό το σχέδιο πρέπει να περιλαμβάνει συγκεκριμένα βήματα, υπεύθυνα μέρη και χρονοδιαγράμματα (π.χ. πως πρέπει να αντιμετωπιστούν οι κίνδυνοι, με ποια σειρά και ποιοι θα είναι υπεύθυνοι)

- ***Εφαρμογή αλλαγών διαμόρφωσης:*** Μερικές φορές, προκύπτουν ευπάθειες λόγω εσφαλμένων ρυθμίσεων συστημάτων ή εφαρμογών.

- ***Επιδιορθώσεις κώδικα:*** Εάν εντοπιστούν ευπάθειες σε λογισμικό ή εφαρμογές, ο χρήστης θα πρέπει να προχωρήσει στην επανασυγγραφή ή την αναδιαμόρφωση τμημάτων κώδικα για την αντιμετώπιση τρωτών σημείων.

- ***Εκτέλεση τακτικών σαρώσεων:*** Προτείνεται ο χρήστης να χρησιμοποιεί συχνά την RESTful υπηρεσία. Οι τακτικές σαρώσεις θα βοηθήσουν στον εντοπισμό νέων τρωτών σημείων και θα παρέχουν την ευκαιρία αυτές να αντιμετωπιστούν προληπτικά.

Η δομή του υπόλοιπου μέρους της παρούσας αναφοράς περιλαμβάνει τα εξής:

- Το υπόβαθρο, όπου παρουσιάζονται οι βασικές έννοιες που σχετίζονται με το αντικείμενο της τρέχουσας διπλωματικής εργασίας (Κεφάλαιο 2)
- Την ανάλυση παρόμοιων εργασιών πάνω σε αυτό το αντικείμενο (Κεφάλαιο 3)
- Την περιγραφή των τεχνολογιών που χρησιμοποιήθηκαν για την υλοποίηση της RESTful υπηρεσίας και της εφαρμογής ιστού που την περιτυλίγει (Κεφάλαιο 4)
- Οδηγίες εγκατάστασης και διαμόρφωσης της προτεινόμενης εφαρμογής ιστού (Κεφάλαιο 5)
- Σενάρια χρήσης της εφαρμογής, βήμα προς βήμα για κάθε στάδιο (Κεφάλαιο 6)
- Τα συμπεράσματα που αντλήθηκαν από την έρευνα και την χρήση της εφαρμογής καθώς και μελλοντικές κατευθύνσεις βελτίωσης και επέκτασής της (Κεφάλαιο 7)

ΚΕΦΑΛΑΙΟ 2 – ΥΠΟΒΑΘΡΟ

2.1 Ορισμός Ευπάθειας

Μια ευπάθεια, στο πλαίσιο της ασφάλειας στον κυβερνοχώρο και της ανάπτυξης λογισμικού, αναφέρεται σε μια αδυναμία ή ελάττωμα σε ένα σύστημα, εφαρμογή λογισμικού ή συσκευή υλικού που θα μπορούσε να εκμεταλλευτεί ένας εισβολέας για να θέσει σε κίνδυνο την εμπιστευτικότητα, την ακεραιότητα ή τη διαθεσιμότητα του συστήματος ή των δεδομένων του. Τα τρωτά σημεία μπορεί να υπάρχουν για διάφορους λόγους και μπορεί να κυμαίνονται από απλά προγραμματιστικά σφάλματα έως πιο σύνθετα ελαττώματα σχεδιασμού.

2.2 OWASP Top 10

Το OWASP Top 10[6] είναι ένα τυπικό έγγραφο ευαισθητοποίησης για προγραμματιστές όσον αφορά την ασφάλεια εφαρμογών ιστού. Αντιπροσωπεύει μια ευρεία συναίνεση σχετικά με τους πιο κρίσιμους κινδύνους ασφάλειας για τις εφαρμογές Ιστού. Ειδικότερα, περιλαμβάνει τις 10 πιο σημαντικές τρωτότητες ως προς την ασφάλεια των εφαρμογών ιστού, οι οποίες αναλύονται στην συνέχεια. Εν γένει, αναγνωρίζεται παγκοσμίως από τους προγραμματιστές ως το πρώτο βήμα προς την ασφαλέστερη κωδικοποίηση [1].

[6] https://owasp.org/www-project-top-ten/

API1:2023 - Broken Object Level Authorization (“Ραγισμένη” Εξουσιοδότηση σε Επίπεδο Αντικειμένων)	Τα API τείνουν να εκθέτουν τελικά σημεία που χειρίζονται αναγνωριστικά αντικειμένων, δημιουργώντας μια ευρεία επιφάνεια επίθεσης ζητημάτων ελέγχου πρόσβασης (π.χ αυθεντικοποίησης, εξουσιοδότησης) σε επίπεδο αντικειμένου. Οι έλεγχοι εξουσιοδότησης σε επίπεδο αντικειμένου θα πρέπει να λαμβάνονται υπόψη σε κάθε λειτουργία που έχει πρόσβαση σε μια πηγή δεδομένων την οποία αιτείται ο χρήστης.
API2:2023 - Broken Authentication (“Ραγισμένη” Ταυτοποίηση)	Οι μηχανισμοί ελέγχου ταυτότητας συχνά υλοποιούνται εσφαλμένα, επιτρέποντας στους εισβολείς να διακυβεύουν τα διακριτικά ελέγχου ταυτότητας ή να εκμεταλλεύονται ελαττώματα της εφαρμογής για να υποκλέψουν προσωρινά ή μόνιμα τις ταυτότητες άλλων χρηστών. Η πιθανή ανικανότητα ενός συστήματος να αναγνωρίζει τον πελάτη με πλήρη και αξιόπιστο τρόπο, θέτει σε κίνδυνο την ασφάλεια ενός API συνολικά.
API3:2023 - Broken Object Property Level Authorization (“Ραγισμένη” Εξουσιοδότηση στο Επίπεδο των Ιδιοτήτων Αντικειμένων)	Αυτή η κατηγορία συνδυάζει το API3:2019 Excessive Data Exposure (Υπερβολική Έκθεση Δεδομένων) και το API6:2019 - Mass Assignment (Μαζική Ανάθεση), εστιάζοντας στη βασική αιτία: την έλλειψη ή ακατάλληλη επικύρωση εξουσιοδότησης σε επίπεδο ιδιοκτησίας αντικειμένου. Η εκμετάλλευση της ευπάθειας API6:2019 απαιτεί συνήθως κατανόηση της επιχειρηματικής λογικής, των σχέσεων αντικειμένων και της δομής του API. Η εκμετάλλευση της μαζικής ανάθεσης είναι ευκολότερη στα API επειδή, από τη σχεδίασή τους, εκθέτουν την υλοποίηση της εφαρμογής καθώς και τα ονόματα των ιδιοτήτων (αντικειμένων). Τα σύγχρονα πλαίσια ενθαρρύνουν τους προγραμματιστές να χρησιμοποιούν συναρτήσεις που συνδέουν

αυτόματα την είσοδο από τον πελάτη σε μεταβλητές κώδικα και εσωτερικά αντικείμενα. Οι εισβολείς μπορούν να χρησιμοποιήσουν αυτήν τη μεθοδολογία για να ενημερώσουν ή να αντικαταστήσουν τις ιδιότητες ευαίσθητων αντικειμένων που οι προγραμματιστές δεν σκόπευαν ποτέ να εκθέσουν. Όσο για την ευπάθεια API3:2019, αυτή αφορά την εκμετάλλευση της υπερέκθεσης δεδομένων και συνήθως γίνεται με την παρακολούθηση και την ανάλυση της κίνησης των δεδομένων που επιστρέφονται από το API στον χρήστη. Η υπερβολική έκθεση δεδομένων προκύπτει όταν τα API αποκαλύπτουν περισσότερα πεδία, δεδομένα και πληροφορίες από ό,τι απαιτεί ο πελάτης μέσω της απόκρισης API. Τα ελαττώματα υπερβολικής έκθεσης δεδομένων εκθέτουν όλες τις ιδιότητες του αντικειμένου σε κλήσεις API και όχι αυτές όπου χρειάζεται να ενεργήσει ο χρήστης χωρίς να λαμβάνεται υπόψη το επίπεδο ευαισθησίας του αντικειμένου.

API4:2023 - Unrestricted Resource Consumption (Μη Περιορισμένη Κατανάλωση Πόρων)

Η ικανοποίηση αιτημάτων API απαιτεί πόρους, όπως εύρος ζώνης δικτύου, κύκλοι κεντρικής μονάδας επεξεργασίας (Central Processing Unit - CPU), μνήμη και αποθήκευση. Άλλοι πόροι, όπως μηνύματα ηλεκτρονικού ταχυδρομείου / SMS / τηλεφωνικές κλήσεις ή επικύρωση βιομετρικών στοιχείων διατίθενται από τους παρόχους υπηρεσιών μέσω ενσωματώσεων API και πληρώνονται ανά αίτημα. Οι επιτυχείς επιθέσεις (κατανάλωση υπερβολικών πόρων του συστήματος, όπως CPU, μνήμη, εύρος ζώνης δικτύου ή χώρο στο δίσκο) που στοχεύουν στο να υποβαθμίσουν ή να διαταράξουν την κανονική λειτουργία ενός συστήματος ή μιας υπηρεσίας μπορεί να οδηγήσουν σε άρνηση υπηρεσίας ή σε αύξηση του λειτουργικού κόστους.

API5:2023 - Broken Function Level Authorization (“Ραγισμένη” Εξουσιοδότηση στο Επίπεδο των Λειτουργιών)	Οι πολύπλοκες πολιτικές ελέγχου πρόσβασης με διαφορετικές ιεραρχίες, ομάδες και ρόλους, καθώς και με έναν ασαφή διαχωρισμό μεταξύ διοικητικών (π.χ. διαμόρφωση δικτύου) και τακτικών λειτουργιών (π.χ. εισαγωγή δεδομένων), τείνουν να οδηγούν σε ελαττώματα εξουσιοδότησης (π.χ. ανεπαρκής έλεγχος πρόσβασης βάσει ρόλου, κλιμάκωση προνομίων). Εκμεταλλευόμενοι αυτά τα ζητήματα, οι εισβολείς μπορούν να αποκτήσουν πρόσβαση σε πόρους ή/και διοικητικές λειτουργίες άλλων χρηστών.
API6:2023 - Unrestricted Access to Sensitive Business Flows (Μη Περιορισμένη Πρόσβαση σε Ευαίσθητες Επιχειρησιακές Ροές)	Τα API που είναι ευάλωτα (για λόγους όπως κακώς καθορισμένα στοιχεία ελέγχου πρόσβασης, εσφαλμένα δικαιώματα, ελλιπής ή ασυνεπής επικύρωση) σε αυτόν τον κίνδυνο εκθέτουν μια επιχειρηματική ροή (business flow) - όπως η αγορά ενός εισιτηρίου ή η δημοσίευση ενός σχολίου - χωρίς να αντισταθμίζουν τον τρόπο με τον οποίο η αντίστοιχη λειτουργικότητα θα μπορούσε να βλάψει την επιχείρηση εάν χρησιμοποιηθεί υπερβολικά / κακόβουλα και με αυτοματοποιημένο τρόπο. Αυτό δεν προέρχεται απαραίτητα από σφάλματα υλοποίησης.
API7:2023 - Server Side Request Forgery (SSRF) (Πλαστογραφία Αιτήσεων στην Πλευρά του Εξυπηρετητή)	Τα σφάλματα πλαστογράφησης αιτημάτων από την πλευρά του διακομιστή / εξυπηρετητή (SSRF) μπορούν να προκύψουν όταν ένα API ανακτά έναν απομακρυσμένο πόρο χωρίς να επικυρώσει το URI που παρέχεται από τον χρήστη. Αυτό δίνει τη δυνατότητα σε έναν εισβολέα να εξαναγκάσει την εφαρμογή να στείλει ένα δημιουργημένο αίτημα για λογαριασμό του σε άλλους πόρους ή υπηρεσίες Ιστού. Αυτά τα αιτήματα διαμορφώνονται συνήθως από την οπτική γωνία του στοχευμένου διακομιστή, ακόμη και όταν προστατεύεται από ένα τείχος προστασίας ή ένα VPN.

API8:2023 - Security Misconfiguration (Κακή Διαμόρφωση Ασφάλειας)	Τα API και τα συστήματα που τα υποστηρίζουν συνήθως περιέχουν πολύπλοκες διαμορφώσεις, με σκοπό να κάνουν τα API πιο προσαρμόσιμα. Οι μηχανικοί λογισμικού και οι DevOps μπορεί να χάσουν αυτές τις διαμορφώσεις ή να μην ακολουθήσουν τις βέλτιστες πρακτικές ασφαλείας όσον αφορά τη ρύθμισή τους, ανοίγοντας την πόρτα για διαφορετικούς τύπους επιθέσεων (π.χ. διέλευση καταλόγου, cross-site scripting, κ.λ.π.).
API9:2023 - Improper Inventory Management (Ακατάλληλη Διαχείριση Αποθεμάτων)	Τα API τείνουν να εκθέτουν περισσότερα τελικά σημεία από τις παραδοσιακές εφαρμογές Ιστού, καθιστώντας τη σωστή και ενημερωμένη τεκμηρίωση εξαιρετικά σημαντική. Ένα κατάλληλο απόθεμα των κεντρικών υπολογιστών και των εκδόσεων API που έχουν αναπτυχθεί σε αυτούς είναι επίσης σημαντικά για τον μετριασμό ζητημάτων, όπως τα εκτεθειμένα τελικά σημεία εντοπισμού σφαλμάτων.
API10:2023 - Unsafe Consumption of APIs (Ανασφαλής Κατανάλωση API)	Οι προγραμματιστές τείνουν να εμπιστεύονται τα δεδομένα που λαμβάνονται από APIs τρίτων μερών περισσότερο από τις εισαγωγές χρηστών, και έτσι τείνουν να υιοθετούν πιο αδύναμα πρότυπα ασφαλείας. Προκειμένου να θέσουν σε κίνδυνο τα APIs, οι εισβολείς αναζητούν ενσωματωμένες υπηρεσίες τρίτων αντί να προσπαθούν να παραβιάσουν το API-στόχο απευθείας. Η διαφορά με την SSRF που αναφέρθηκε προηγουμένως εστιάζει κυρίως στο πεδίο εφαρμογής, όπου περιλαμβάνει διάφορα ζητήματα ασφάλειας, όπως εσφαλμένη ρύθμιση παραμέτρων χρήσης API, ακατάλληλο έλεγχο ταυτότητας, σε αντίθεση με το SSRF όπου ταυτίζεται κυρίως με εφαρμογές ιστού και προκύπτει όταν οι εφαρμογές αυτές επιτρέπουν στους χρήστες να προσδιορίζουν διευθύνσεις URL κατά την υποβολή αιτημάτων HTTP.

Η παραπάνω λίστα δεν είναι στάσιμη αλλά δυναμική. Ειδικότερα, αλλάζει κάπως συχνά για να αντικατοπτρίσει τις τρέχουσες απειλές και ευπάθειες στις εφαρμογές ιστού και τους ιστοτόπους. Η λίστα OWASP Top 10 που παρέχεται σε αυτή την αναφορά είναι η πιο πρόσφατη (έκδοση 2023).

2.3 Σάρωση Ευπαθειών

Η σάρωση ευπάθειας είναι η διαδικασία εντοπισμού και αξιολόγησης τρωτών σημείων ή αδυναμιών σε συστήματα υπολογιστών, δίκτυα και εφαρμογές [10]. Ο στόχος της σάρωσης ευπάθειας είναι να ανιχνεύσει πιθανές αδυναμίες ασφάλειας πριν τις εκμεταλλευτούν οι εισβολείς.

Η σάρωση ευπάθειας είναι ένα σημαντικό στοιχείο ενός ολοκληρωμένου προγράμματος κυβερνοασφάλειας. Βοηθά τους οργανισμούς να εντοπίσουν και να αποκαταστήσουν τις ευπάθειες προτού τις εκμεταλλευτούν οι εισβολείς, μειώνοντας τον κίνδυνο παραβίασης δεδομένων, κλοπής ή άλλων περιστατικών ασφάλειας στον κυβερνοχώρο.

Εν γένει, μόλις εντοπιστούν τα τρωτά σημεία (από το αντίστοιχο εργαλείο που εφαρμόζει μια ή παραπάνω τεχνικές σάρωσης), δημιουργείται μια αναφορά με λεπτομερείς πληροφορίες σχετικά με τη φύση της ευπάθειας, τη σοβαρότητά της και συστάσεις για αποκατάσταση/αντιμετώπισή της.

Υπάρχουν διάφοροι τύποι τεχνικών σάρωσης ευπαθειών, οι οποίοι αναλύονται παρακάτω [11].

2.3.1 Σάρωση Θυρών / Port Scanning

Αυτή η τεχνική σαρώνει τις θύρες των υπολογιστών ενός δικτύου για να εντοπίσει ανοιχτές θύρες και υπηρεσίες στις οποίες είναι δυνατή η απομακρυσμένη πρόσβαση. Οι εισβολείς μπορούν να βρουν ποιες υπηρεσίες (όπως διακομιστές web, διακομιστές email, και άλλα) εκτελούνται σε ανοιχτές θύρες. Αυτές οι πληροφορίες βοηθούν στην κατανόηση της υποδομής του στόχου και των πιθανών τρόπων επίθεσης. Η γνώση που μπορεί να εξαχθεί σε αυτή την περίπτωση περιλαμβάνει τα εξής:

- Εκτίμηση ευπάθειας: Εντοπίζοντας ανοιχτές θύρες και σχετικές υπηρεσίες, οι εισβολείς μπορούν να αναζητήσουν γνωστά ευπαθή σημεία που σχετίζονται με αυτές τις υπηρεσίες και στη συνέχεια να αποκτήσουν πρόσβαση.
- Τείχος προστασίας: Η σάρωση θύρας μπορεί να αποκαλύψει πληροφορίες σχετικά με τους κανόνες του τείχους προστασίας. Οι επιτιθέμενοι μπορούν να χρησιμοποιήσουν αυτές τις πληροφορίες για να σχεδιάσουν στρατηγικές που να παρακάμπτουν τα μέτρα ασφαλείας, όπως είναι το τείχος προστασίας.
- Χαρτογράφηση δικτύου: Η σάρωση θυρών βοηθά τους εισβολείς να χαρτογραφήσουν την τοπολογία του δικτύου και να εντοπίσουν πιθανούς στόχους εντός του δικτύου.

Ο εισβολέας θα μπορούσε με τους εξής τρόπους να χρησιμοποιήσει τη παραπάνω γνώση:

- **Επιθέσεις ωμής βίας:** Εφόσον ένας επιτιθέμενος γνωρίζει ποιες υπηρεσίες εκτελούνται σε ανοιχτές θύρες, του επιτρέπεται να εστιάσει σε συγκεκριμένα τρωτά σημεία και να εξαπολύσει επιθέσεις ωμής βίας (brute force) για να αποκτήσει μη εξουσιοδοτημένη πρόσβαση.
- **Επίθεση λεξικού (Dictionary Attack):** Αντί να δοκιμάσει κάθε πιθανό συνδυασμό, ένας εισβολέας μπορεί να χρησιμοποιήσει μια λίστα με κωδικούς που χρησιμοποιούνται συνήθως ή συνδυασμούς που είναι εύκολο να μαντέψει κανείς με βάση πληροφορίες σχετικά με τον στόχο (πχ. ημερομηνία γενεθλίων κ.λ.π.).
- **Πίνακας "ουράνιου τόξου" (Rainbow Table):** Οι εισβολείς μπορούν να χρησιμοποιήσουν προυπολογισμένους πίνακες κωδικών και τους αντίστοιχους κατακερματισμούς τους για να προσδιορίσουν γρήγορα τον κωδικό ενός δεδομένου κατακερματισμού. Αυτό προϋποθέτει πως ο κατακερματισμένος κωδικός ενός χρήστη έχει κλαπεί.
- **Γέμισμα διαπιστευτηρίων (Credential Stuffing):** Εάν ένας εισβολέας έχει μια λίστα ονομάτων χρήστη και κωδικούς από άλλη παραβίαση, θα μπορούσε να επιχειρήσει να χρησιμοποιήσει αυτά τα διαπιστευτήρια στο σύστημα προορισμού με την ελπίδα ότι ορισμένοι χρήστες επαναχρησιμοποιούν τους κωδικούς τους και σε άλλες εφαρμογής ή υπηρεσίες.

Μερικά τρωτά σημεία είναι:

1. **Κλειδί κρυπτογράφησης:** Στην περίπτωση κρυπτογραφημένων δεδομένων, ένας εισβολέας μπορεί να επιχειρήσει να ασκήσει ωμή βία στο κλειδί κρυπτογράφησης (αυτό συνεπαγέται ότι ο εισβολέας μπορεί να είναι ένα είδος ενδιάμεσου). Αυτό περιλαμβάνει τη δοκιμή όλων των πιθανών κλειδιών μέχρι να βρεθεί το σωστό, επιτρέποντας στον εισβολέα να αποκρυπτογραφήσει τα δεδομένα. Αυτός ο τύπος επίθεσης μπορεί να είναι ιδιαίτερα χρονοβόρος και να απαιτεί μεγάλο αριθμό πόρων, ανάλογα με τον αλγόριθμο κρυπτογράφησης και το μήκος του κλειδιού.
2. **Token API:** Στο πλαίσιο μιας διεπαφής προγραμματισμού εφαρμογών (API), ένας εισβολέας μπορεί να επιχειρήσει να χρησιμοποιήσει διακριτικά (diacritical) API ή διακριτικά ελέγχου ταυτότητας για να αποκτήσει μη εξουσιοδοτημένη πρόσβαση σε μια εφαρμογή ή υπηρεσία (ιστού).
3. **Κλειδί ασύρματου δικτύου:** Οι εισβολείς μπορούν να χρησιμοποιήσουν τεχνικές ωμής βίας για να αποκρυπτογραφήσουν το κλειδί κρυπτογράφησης που χρησιμοποιείται από ένα δίκτυο Wi-Fi, επιτρέποντάς τους μη εξουσιοδοτημένη πρόσβαση στο δίκτυο. Εναλλακτικά, εφόσον ο δρομολογητής WiFi χρησιμοποιεί ένα προκαθορισμένο κλειδί, αυτό μπορεί να εντοπιστεί από λίστες προϊόντων του προμηθευτή που είναι διαθέσιμες στο Διαδίκτυο.
4. **Πορτοφόλι κρυπτονομισμάτων:** Οι επιθέσεις ωμής βίας μπορούν να χρησιμοποιηθούν για να μαντέψουν ιδιωτικά κλειδιά που σχετίζονται με πορτοφόλια κρυπτονομισμάτων. Εάν είναι επιτυχής, ο εισβολέας μπορεί να κλέψει κρυπτονομίσματα που είναι αποθηκευμένα στο πορτοφόλι.
5. **Κοινωνική Μηχανική**: Οι εισβολείς μπορούν να χρησιμοποιήσουν τις πληροφορίες που συλλέγονται για να δημιουργήσουν μηνύματα ηλεκτρονικού ψαρέματος (fishing) ή μηνύματα που εκμεταλλεύονται τις γνωστές υπηρεσίες του στόχου, καθιστώντας πιο πιθανό το θύμα να αλληλεπιδράσει με κακόβουλο περιεχόμενο. Εναλλακτικά, η κοινωνική μηχανική θα μπορούσε να χρησιμοποιηθεί για την εκμαίευση προσωπικής πληροφορίας του χρήστη που μπορεί να χρησιμοποιείται στους κωδικούς του. Οπότε, αυτό κάνει δυνατή τη χρήση πιο ενημερωμένων brute-force μεθόδων που εξετάζουν διάφορους συνδυασμούς προσωπικών πληροφοριών για να μαντέψουν τον κωδικό του χρήστη.

Μερικοί πιθανοί τρόποι αντιμετώπισης είναι:

- **Παρακολούθηση δικτύου**: Τακτική παρακολούθηση της κυκλοφορίας του δικτύου για ύποπτες ή μη εξουσιοδοτημένες δραστηριότητες σάρωσης θυρών. Τα συστήματα ανίχνευσης εισβολής (Intrusion Detection Systems - IDS) και τα συστήματα πρόληψης εισβολής (Intrusion Protection Systems - IPS) μπορούν να βοηθήσουν στον εντοπισμό και τον αποκλεισμό τέτοιων επιθέσεων.
- **Διαμόρφωση τείχους προστασίας**: Διαμόρφωση του τείχους προστασίας για τον περιορισμό της περιττής έκθεσης υπηρεσιών και κλείσιμο των περιττών υπηρεσιών μέσω σκλήρυνσης λειτουργικού συστήματος και εφαρμογών.
- **Διαχείριση ενημερώσεων κώδικα:** Διατήρηση όλων των συστημάτων και του λογισμικού ενημερωμένα με τις πιο πρόσφατες ενημερώσεις κώδικα ασφαλείας για την ελαχιστοποίηση του κινδύνου εκμετάλλευσης γνωστών τρωτών σημείων από επιτιθέμενους.

2.3.2 Σάρωση Ευπάθειας Δικτύου / Network Scanning

Αυτή η τεχνική εξετάζει την υποδομή δικτύου και εντοπίζει πιθανές ευπάθειες σε δρομολογητές, μεταγωγείς και άλλες συσκευές δικτύου.

Μερικοί πιθανοί τρόποι επίθεσης είναι:

- **Κατανάλωση εύρους ζώνης (bandwidth consumption)**: Οι εισβολείς μπορεί να δημιουργήσουν υπερφόρτωση της κυκλοφορίας, που θα καταστήσει το δίκτυο υποτονικό ή να μην ανταποκρίνεται.
- **Εξάντληση Πόρων (resource exhaustion)**: Οι εισβολείς ενδέχεται να πραγματοποιήσουν πολλαπλές αιτήσεις που παράγουν μεγάλες απαντήσεις ώστε να οδηγήσουν στην κατανάλωση όλης της CPU, μνήμης ή τους πόρους αποθήκευσης του διακομιστή ή των συστημάτων προορισμού. Αν ο διακομιστής παρέχει υπηρεσίες σάρωσης, αυτό σημαίνει πως το σύστημα δεν θα μπορεί πλέον να πραγματοποιεί σαρώσεις για τρωτότητες που έπειτα θα μπορούν να αντιμετωπιστούν.
- **Πλημμύρα πακέτων (packet flooding)**: Οι εισβολείς αποστέλλουν μεγάλο όγκο πακέτων στον στόχο, προκαλώντας συμφόρηση δικτύου ή υπερφόρτωση διακομιστή.
- **Εξάντληση σύνδεσης (connection exhaustion)**: Οι εισβολείς ενδέχεται να επιχειρήσουν να εξαντλήσουν τον μέγιστο αριθμό παράλληλων συνδέσεων που επιτρέπεται από τα συστήματα στόχου για να τα καταστήσουν μη διαθέσιμα.

- **Επιθέσεις σε υποδομή σάρωσης (infrastructure attacks)**: Οι εισβολείς μπορεί να στοχεύσουν απευθείας την υποδομή σάρωσης ευπάθειας, επιχειρώντας να διακόψουν ή να υπονομεύσουν τους διακομιστές ή τα εργαλεία σάρωσης.

Μερικοί πιθανοί τρόποι αντιμετώπισης είναι οι εξής:

- **Εφαρμογή πολιτικών φιλτραρίσματος κυκλοφορίας δικτύου**, μέσω της διαμόρφωσης των απαραίτητων παραμέτρων.
- **Εφαρμογή κανόνων τείχους προστασίας (firewall)** και συστημάτων **ανίχνευσης/αποτροπής εισβολής (IDS/IPS)** για τον εντοπισμό και τον αποκλεισμό της κυκλοφορίας που σχετίζεται με επιθέσεις πλημμύρας πακέτων.
- **Σωστή διαμόρφωση εργαλείων σάρωσης** ώστε να χρησιμοποιούν τη **συγκέντρωση συνδέσεων** και να εφαρμόζουν περιορισμό ρυθμού (συνδέσεων) για τον έλεγχο του αριθμού των ταυτόχρονων συνδέσεων.
- **Απομόνωση** της υποδομή σάρωσης από το δίκτυο παραγωγής, με την χρήση ισχυρών ελέγχων πρόσβασης και μέτρων ασφαλείας.

2.3.3 Σάρωση Εφαρμογών / Application Scanning

Αυτή η τεχνική σαρώνει εφαρμογές για ευπάθειες που μπορούν να γίνουν εκμεταλλεύσιμες από επιθέσεις, όπως η έγχυση SQL, η δέσμη ενεργειών μεταξύ τοποθεσιών (XSS), η έγχυση εντολών (command injection), και η διάβαση μονοπατιού (path crossing).

Παρακάτω θα αναλύσουμε τις προαναφερόμενες επιθέσεις:

- **Έκχυση SQL (SQL Injection):** Ένας εισβολέας υποβάλλει κακόβουλο αίτημα (input request) που περιλαμβάνει κώδικα SQL ως μέρος των δεδομένων εισόδου. Αυτός ο κώδικας μπορεί να εισαχθεί σε πεδία εισαγωγής, παραμέτρους URL ή άλλα δεδομένα. Η είσοδος του εισβολέα περιλαμβάνεται από την εφαρμογή ιστού σε ένα ερώτημα SQL (SQL query) χωρίς κατάλληλη επικύρωση. Ως αποτέλεσμα, το ερώτημα SQL γίνεται ένας συνδυασμός του νόμιμου ερωτήματος και της κακόβουλης εισόδου του εισβολέα. Το επεξεργασμένο ερώτημα SQL στη συνέχεια εκτελείται από την εφαρμογή έναντι της βάσης δεδομένων. Εάν είναι επιτυχής, ο εισβολέας μπορεί να συλλέξει, να τροποποιήσει ή να διαγράψει δεδομένα από τη βάση δεδομένων, ανάλογα με το είδος του ερωτήματος και του κώδικα που έχει εκχύσει.

- **Έγχυση εντολών (command injection):** Σε μια ευάλωτη εφαρμογή, οι είσοδοι του χρήστη (π.χ. δεδομένα που εισάγονται σε φόρμες ιστού) δεν επικυρώνονται σωστά. Έτσι, ένας εισβολέας μπορεί να υποβάλλει κακόβουλη είσοδο που περιλαμβάνει ειδικούς χαρακτήρες ή εντολές ως μέρος των δεδομένων εισόδου. Αυτή η είσοδος μπορεί να εισαχθεί σε πεδία εισαγωγής και παραμέτρους URL. Ως αποτέλεσμα, ο εισβολέας μπορεί να εισάγει τις δικές του εντολές στην αρχική εντολή συστήματος που εκτελείται από την εφαρμογή. Ας πάρουμε για παράδειγμα την εξής εντολή:

  ```
  system("cd /var/yp && make &> /dev/null");
  ```

  Η εντολή σε αυτό το παράδειγμα είναι κωδικοποιημένη, επομένως ένας εισβολέας δεν μπορεί να ελέγξει το όρισμα που μεταβιβάζεται στην κλήση συστήματος. Ωστόσο, δεδομένου ότι το πρόγραμμα δεν καθορίζει μια απόλυτη διαδρομή για το make και δεν καθαρίζει καμία μεταβλητή περιβάλλοντος πριν από την κλήση της εντολής, ο εισβολέας μπορεί να τροποποιήσει τη μεταβλητή $PATH για να υποδείξει ένα κακόβουλο δυαδικό όνομα make προς εκτέλεση στο κέλυφος (του λειτουργικού συστήματος φιλοξενίας). Δεδομένου ότι το πρόγραμμα εκτελείται με δικαιώματα ριζικού χρήστη, το ίδιο θα ισχύει για την έκδοση make του εισβολέα. Προφανώς, βασική προϋπόθεση είναι ο εισβολέας να έχει κατορθώσει να εκχύσει το εναλλακτικό πρόγραμμα make με κάποια άλλη επίθεση.
  (δείτε το 5ο παράδειγμα στην ιστοσελίδα του OWASP για την έκχυση εντολών[7]).
  Η κακόβουλη αυτή εντολή εκτελείται από την εφαρμογή ή το σύστημα με τα δικαιώματα της εφαρμογής ή του χρήστη που την εκτελεί. Αυτό μπορεί να οδηγήσει σε σημαντικές συνέπειες, όπως η εκτέλεση αυθαίρετων εντολών στο σύστημα φιλοξενίας της εφαρμογής.
- **Διάβαση μονοπατιού (path crossing):** Σε αυτόν τον τύπο επίθεσης, ένας εισβολέας επιχειρεί να αποκτήσει μη εξουσιοδοτημένη πρόσβαση σε αρχεία ή καταλόγους σε έναν διακομιστή ιστού. Η επίθεση βασίζεται στον χειρισμό εισόδου που περιλαμβάνει διαδρομές αρχείων για πλοήγηση σε καταλόγους εκτός του προβλεπόμενου πεδίου. Εάν είναι επιτυχής, ο εισβολέας ενδέχεται να μπορεί να προβάλει, να τροποποιήσει ή να διαγράψει αρχεία που θα μπορούσαν να είναι και ευαίσθητα ή να επηρεάσουν την εκτέλεση της εφαρμογής.

[7] https://owasp.org/www-community/attacks/Command_Injection

Μερικοί τρόποι αντιμετώπισης των επιθέσεων που αφορούν την ασφάλεια των εφαρμογών ιστού είναι οι εξής:

- **Χρήση έτοιμων, παραμετροποιημένων δηλώσεων** (prepared statements) που παρέχονται από τη γλώσσα προγραμματισμού ή το πλαίσιο (ανάπτυξης εφαρμογών) ιστού. Αυτοί οι μηχανισμοί χειρίζονται αυτόματα την επικύρωση εισόδου, αποτρέποντας επιθέσεις όπως είναι η έγχυση SQL.
- **Εφαρμογή ισχυρής επικύρωσης εισόδου** για να διασφαλιστεί ότι τα πεδία εισόδου των χρηστών πληρούν τα αναμενόμενα μοτίβα. Απόρριψη οποιασδήποτε εισόδου περιέχει μη αναμενόμενους χαρακτήρες, σύμβολα ή λέξεις-κλειδιά (για την SQL ή για τη γραμμή εντολών) ή εναλλακτικά αφαίρεση όλων των μη αναμενόμενων χαρακτήρων, συμβόλων ή λέξεων.
- **Περιορισμός των δικαιωμάτων** του χρήστη της βάσης δεδομένων στο ελάχιστα απαραίτητο για την εφαρμογή. Αποφυγή της χρήση λογαριασμών υπερχρηστών (superusers) για συνδέσεις βάσης δεδομένων, καθώς αυτό μπορεί να αντιμετωπίσει πιθανές επιθέσεις (ουσιαστικά ακόμη και αν ο επιτιθέμενος υποκλέψει τα στοιχεία του κανονικού χρήστη, δεν θα μπορεί να προχωρήσει σε κακόβουλες ενέργειες σε άλλες βάσεις δεδομένων παρά μόνο σε ελάχιστες ενέργειες στην τρέχουσα βάση).
- **Τείχος προστασίας εφαρμογών Ιστού (WAF)**: Εφαρμογή τείχους προστασίας εφαρμογών ιστού, το οποίο ανιχνεύει και μπλοκάρει κακόβουλα αιτήματα HTTP προς τις εφαρμογές ιστού. Τα WAF μπορούν να αναγνωρίσουν και να φιλτράρουν διάφορα είδη επιθέσεων έκχυσης (πχ. SQL έκχυση) καθώς και άρνησης υπηρεσίας.
- **Κεφαλίδες ασφαλείας (security headers)**: Εφαρμογή κεφαλίδων ασφαλείας, όπως Πολιτική Ασφάλειας Περιεχομένου (Content Security Policy – CSP) (ένα πρόσθετο επίπεδο ασφάλειας που βοηθά στον εντοπισμό και τον μετριασμό ορισμένων τύπων επιθέσεων, συμπεριλαμβανομένου του Cross-Site Scripting (XSS) και των επιθέσεων εισαγωγής δεδομένων) και X-Content-Type-Options (ένας δείκτης που χρησιμοποιείται από τον διακομιστή για να υποδείξει ότι οι τύποι MIME που διαφημίζονται στις κεφαλίδες τύπου περιεχομένου πρέπει να ακολουθούνται και να μην αλλάζουν).
- **Ασφαλής χειρισμός αρχείων**: Εάν η εφαρμογή επιτρέπει μεταφορτώσεις αρχείων, θα πρέπει να επικυρωθεί ότι τα μεταφορτωμένα αρχεία αποθηκεύονται σε τοποθεσία που δεν είναι προσβάσιμη από τους χρήστες του Ιστού καθώς και δεν είναι άμεσα εκτελέσιμα.

- **Εφαρμογή καταγραφής πρόσβασης σε ευαίσθητα αρχεία και καταλόγους**: Τακτικός έλεγχος των αρχείων καταγραφής για να εντοπιστούν τυχόν προσπάθειες μη εξουσιοδοτημένης πρόσβασης.
- **Χρήση απόλυτων διαδρομών (absolute paths)**: Η προτίμηση των απόλυτων διαδρομών αρχείων αντί των σχετικών περιορίζει τη δυνατότητα για επιθέσεις διέλευσης μονοπατιών, καθώς ο εισβολέας θα πρέπει να διαφύγει από τον ριζικό κατάλογο, κάτι που δεν θα πρέπει να επιτρέπεται.
- **Πραγματοποίηση σάρωσης διαπιστευτηρίων**: Εντοπισμός και αποκατάσταση των κινδύνων ασφαλείας που σχετίζονται με τη διαχείριση κωδικών και τον έλεγχο ταυτότητας. Τα διαπιστευτήρια των χρηστών γίνονται αποδεκτά, εφόσον πληρούν κατάλληλα μοτίβα, κανόνες και πολιτικές ασφάλειας. Αυτό πραγματοποιείται τόσο κατά την εγγραφή των χρηστών όσο και κατά την ανανέωση των διαπιστευτηρίων τους.

2.3.4 Σάρωση Κακόβουλου Λογισμικού / Malware Scanning

Αυτή η τεχνική σαρώνει για κακόβουλο λογισμικό που θα μπορούσε να θέσει σε κίνδυνο την ασφάλεια του συστήματος ή του δικτύου. Ο επιτιθέμενος θα πρέπει πρώτα να μπορέσει να εγκαταστήσει το κακόβουλο λογισμικό στον υπολογιστή του θύματος. Αυτό μπορεί να γίνει με πολλούς τρόπους, όπως λήψη μολυσμένων αρχείων από την πλευρά του θύματος, λήψη και εκτέλεση συνημμένων αρχείων σε email, ευπάθειες λογισμικού, κ.α.

Ο επιτιθέμενος θα μπορούσε να δυσχαιράνει τον εντοπισμό του κακόβουλου λογισμικού που έχει εκχυθεί με τους εξής τρόπους:

- **Πολυμορφικό κακόβουλο λογισμικό**: Οι εισβολείς χρησιμοποιούν πολυμορφικό κακόβουλο λογισμικό, το οποίο αλλάζει τον κώδικα ή την εμφάνισή του κάθε φορά που μολύνει έναν νέο κεντρικό υπολογιστή.
- **Κρυπτογράφηση**: Οι εισβολείς μπορούν να κρυπτογραφήσουν το κακόβουλο λογισμικό τους, χρησιμοποιώντας προσαρμοσμένους αλγόριθμους κρυπτογράφησης, καθιστώντας δύσκολο για τα εργαλεία (ανίχνευσης, αντιβιοτικά, κ.α.) που βασίζονται σε υπογραφές να αναγνωρίσουν το κρυπτογραφημένο κακόβουλο λογισμικό.
- **Συσκότιση κώδικα (code obfuscation)**: Οι εισβολείς ενδέχεται να επεξεργαστούν τον κώδικα κακόβουλου λογισμικού τους με χρήσεις “καμουφλάζ”, καθιστώντας δύσκολο για τα εργαλεία στατικής ανάλυσης να το εντοπίσουν.

- **Zero-Day Exploits:** Οι εισβολείς χρησιμοποιούν προηγουμένως άγνωστες ευπάθειες (zero-days) για να παραδώσουν κακόβουλο λογισμικό. Εφόσον οι υπογραφές του αντίστοιχου κακόβουλου λογισμικού είναι ακόμα άγνωστες, ο επιτιθέμενος έχει πλεονέκτημα έναντι πιθανών εργαλείων αντιμετώπισης.

Μερικοί τρόποι αντιμετώπισης των παραπάνω είναι οι εξής:

- **Χρήση προηγμένων εργαλειών προστασίας από ιούς και κακόβουλα προγράμματα**: Οι σύγχρονες λύσεις προστασίας από ιούς και κακόβουλο λογισμικό μπορούν να εντοπίσουν όχι μόνο γνωστές υπογραφές (στον κώδικα του ιού / λογισμικού), αλλά γενική κακόβουλη συμπεριφορά (ιδιαίτερα χρήσιμο στην περίπτωση νέων ιών και κακόβουλου λογισμικού).
- **Εφαρμογή Ασφάλειας Δικτύου**: Χρήση συστημάτων ανίχνευσης και πρόληψης εισβολών δικτύου (Network-based IDS - NIDS / Network-based IPS - NIPS) για να εντοπιστούν ασυνήθιστες συμπεριφορές δικτύου που μπορεί να υποδηλώνουν δραστηριότητα κακόβουλου λογισμικού στις δικτυωμένες συσκευές / μηχανήματα.
- **Χρήση ανίχνευσης εισβολής βάσει κεντρικού υπολογιστή**: Ανάπτυξη συστημάτων ανίχνευσης εισβολής που βασίζονται σε υπολογιστή (Host-based IDS - HIDS) για την παρακολούθηση των δραστηριοτήτων του συστήματος στον υπολογιστή αυτόν και τον εντοπισμό πιθανών μολύνσεων από κακόβουλο λογισμικό.
- **Εφαρμογή λευκής λίστας εφαρμογών (White listing)**: Να επιτρέπεται μόνο σε εξουσιοδοτημένες εφαρμογές να εκτελούνται σε συστήματα/υπολογιστές, μειώνοντας την πιθανότητα επίθεσης.

2.4 Μεθοδολογίες Σάρωσης Ευπαθειών

Οι μεθοδολογίες σάρωσης ευπαθειών, οι οποίες χρησιμοποιούνται γενικότερα για την ανίχνευση τρωτοτήτων που μπορούν να επηρεάσουν τυχόν εφαρμογές, είναι δυο: SAST (Static Application Security Testing) (Στατική Δομική Ασφάλειας Εφαρμογής) και DAST (Dynamic Application Security Testing) (Δυναμική Δοκιμή Ασφάλειας Εφαρμογής).

Με την χρήση της μεθοδολογίας SAST, σκανάρεται ο πηγαίος κώδικας μια εφαρμογής με σκοπό την ανίχνευση τρωτοτήτων και αδυναμιών μέσα σε αυτόν, όπως για παράδειγμα μιας ευπάθειας έκχυσης SQL λόγω μη χρήσης έτοιμων δηλώσεων SQL (prepared statements), καθώς και άλλες

πολύ σημαντικές ευπάθειες που μπορούν να εντοπισθούν και στον προαναφερόμενο κατάλογο OWASP Top 10 [4],[11].

Αντιθέτως, η μεθοδολογία DAST, ελέγχει μια εφαρμογή σε περιβάλλον εκτέλεσης, με σκοπό την ανίχνευση τρωτοτήτων που θα μπορούσε κάποιος επιτιθέμενος να εντοπίσει (συνήθως με απομακρυσμένο τρόπο) καθώς η εφαρμογή εκτελείται.

Οι σαρωτές (DAST) μπορούν να εκτελούν τόσο απομακρυσμένη όσο και τοπική σάρωση, ανάλογα με τον σαρωτή και τη διαμόρφωσή του. Ο πρωταρχικός σκοπός του DAST είναι να εντοπίζει με δυναμικό τρόπο τρωτά σημεία ασφαλείας σε εφαρμογές Ιστού προσομοιώνοντας επιθέσεις πραγματικού κόσμου. Παρακάτω παραθέτουμε ορισμένες επιπλέον λεπτομέρειες για τα είδη σάρωσης DAST [5].

- **Απομακρυσμένη σάρωση DAST**: Η απομακρυσμένη σάρωση είναι η πιο κοινή προσέγγιση για τα εργαλεία DAST. Περιλαμβάνει τη σάρωση της εφαρμογής-στόχου από μια εξωτερική προοπτική, χωρίς άμεση πρόσβαση στην υποκείμενη υποδομή. Ο σαρωτής αλληλεπιδρά με την εφαρμογή Ιστού μέσω του (δια)δικτύου, στέλνοντας διάφορα αιτήματα και αναλύοντας τις απαντήσεις για τον εντοπισμό τρωτών σημείων. Η απομακρυσμένη σάρωση είναι χρήσιμη για τον εντοπισμό τρωτών σημείων που μπορούν να αξιοποιηθούν από την οπτική γωνία ενός εξωτερικού εισβολέα.
- **Τοπική σάρωση DAST**: Η τοπική σάρωση περιλαμβάνει την εκτέλεση του εργαλείου DAST στο ίδιο δίκτυο ή περιβάλλον με την εφαρμογή προορισμού. Αυτό επιτρέπει στον σαρωτή να έχει βαθύτερη ορατότητα στην εφαρμογή και να εντοπίζει πιθανά τρωτά σημεία που ενδέχεται να μην είναι προσβάσιμα από απομακρυσμένη προοπτική. Η τοπική σάρωση μπορεί να βοηθήσει στην αποκάλυψη ζητημάτων που σχετίζονται με εσωτερικές διαμορφώσεις δικτύου, στοιχεία ελέγχου πρόσβασης ή ευπάθειες που μπορούν να αξιοποιηθούν μόνο μέσα από το (εσωτερικό) δίκτυο.

Σε αυτό το σημείο, είναι σημαντικό να τονιστεί ότι η κάθε μεθοδολογία έχει τα θετικά και τα αρνητικά της. Επομένως, η επιλογή τους θα πρέπει να εξαρτάται από τις ανάγκες του χρήστη ή της επιχείρησης. Αυτό συμβαίνει γιατί μιλάμε για δύο εντελώς διαφορετικές προσεγγίσεις στην ανίχνευση τρωτοτήτων. Παρακάτω συγκρίνουμε τις δύο αυτές μεθοδολογίες με βάση συγκεκριμένα κριτήρια αφού πρώτα εξηγήσουμε τις δύο βασικές και πολύ γνωστές μετρικές της ακρίβειας (που αποδίδεται με τον αγγλικό όρο accuracy).

Η μετρική [3] της ακρίβειας (precision) αφορά το πόσο καλό είναι ένα εργαλείο στην εύρεση πραγματικών προβλημάτων ασφάλειας και όχι πλασματικών. Όσο μεγαλύτερη είναι η ακρίβεια, τόσο λιγότερο πιθανό είναι να δοθεί λάθος απάντηση. Η ακρίβεια ορίζεται ως το ποσοστό των σωστών στοιχείων (πχ. τρωτοτήτων) που εντοπίστηκαν σε σχέση με το συνολικό πλήθος των εντοπισμένων στοιχείων (δηλ. σε σχέση με τα στοιχεία ενός σύνολου δεδομένων). Υπολογίζεται ως εξής:

- Precision = TP / (TP + FP) όπου TP (true positives) είναι ο αριθμός των σωστών στοιχείων που εντοπίστηκαν και FP (false positives) είναι ο αριθμός των λανθασμένων στοιχείων που εντοπίστηκαν (π.χ. από ένα εργαλείο σάρωσης).

Εφόσον η ακρίβεια (precision) είναι υψηλή υπάρχει υψηλή εμπιστοσύνη στην καταλληλότητα των αποτελεσμάτων που παράγονται από ένα εργαλείο διότι υπάρχει μεγάλη πιθανότητα να είναι σωστά. Όμως, δεν είναι σίγουρο πως το μοντέλο / εργαλείο είναι σε θέση να εντοπίσει όλα τα δυνατά σχετικά στοιχεία.

Από την άλλη μεριά, η ανάκληση (recall) αφορά την ικανότητα ανίχνευσης όσο γίνεται περισσότερων τρωτοτήτων / στοιχείων, ιδανικά όλων των δυνατών, σε ένα σύνολο δεδομένων. Ορίζεται ως ο λόγος του αριθμού των σχετικών στοιχείων (τρωτοτήτων) που ανακτήθηκαν από το μοντέλο / εργαλείο προς τον συνολικό αριθμό των σχετικών στοιχείων (τρωτοτήτων) στο σύνολο δεδομένων. Υπολογίζεται χρησιμοποιώντας τον ακόλουθο τύπο:

- Recall= TP / (FN + TP) όπου FN (false negatives) είναι οι περιπτώσεις όπου το μοντέλο / εργαλείο ταξινόμησε εσφαλμένα τα σχετικά στοιχεία ως μη σχετικά ή δεν τα εντόπισε καθόλου.

Μια υψηλή τιμή ανάκλησης υποδηλώνει ότι το μοντέλο / εργαλείο μπορεί να εντοπίσει ένα μεγάλο ποσοστό από σωστά σχετικά στοιχεία (όπως ευπάθειες). Αυτό είναι σημαντικό ειδικά όταν θα πρέπει για ένα σύστημα να εντοπίσουμε όσο γίνεται περισσότερες ευπάθειες. Από την άλλη μεριά, δεν είναι σίγουρο πως όλα τα εντοπισμένα στοιχεία είναι σίγουρα σωστά. Επομένως, οι μετρικές της ακρίβειας και ανάκλησης είναι συμπληρωματικές και ένα εργαλείο είναι κατάλληλο προς επιλογή εφόσον έχει υψηλές ή αποδεκτές τιμές και στις δύο αυτές μετρικές.

Παραθέτουμε τώρα την σύγκριση μεταξύ των δύο μεθοδολογιών σάρωσης τρωτοτήτων.

Ακρίβεια αποτελεσμάτων.

Τα εργαλεία SAST δίνουν περισσότερη έμφαση στην ακρίβεια (precision) παρά στην ανάκληση (recall). Ειδικότερα, βασίζονται στην κρισιμότητα του να δώσουν ακριβή αποτελέσματα-ευρήματα στον προγραμματιστή, ελαχιστοποιώντας τα ψευδώς θετικά ευρήματα (false positives). Αυτό είναι επίσης δυνατόν λόγω της φύσης των τεχνικών που χρησιμοποιούνται σε αυτή τη περίπτωση. Παρόλα αυτά, αυτή η έμφαση μπορεί να λειτουργήσει αρνητικά για τον βαθμό ανάκλησης. Επιπλέον, τα SAST εργαλεία δεν μπορούν να καλύψουν την δυναμική συμπεριφορά της εφαρμογής. Επομένως, ιδιαίτερα αν ο κώδικας που σαρώνεται είναι πολύ εκτενής και περίπλοκος, τα εργαλεία SAST μπορούν να δώσουν αρκετά ψευδή αρνητικά (false negatives).

Απο την άλλη μεριά, επιλέγοντας την δυναμική ανάλυση κώδικα, η ανίχνευση θα τρέξει στο περιβάλλον εκτέλεσης της εφαρμογής χωρίς καν να χρειαστεί να ελεγχθεί ο κώδικας.

Επομένως, η στατική ανάλυση μας δίνει εγγενώς λιγότερα ψευδώς θετικά (άρα η ακρίβεια είναι καλή) αλλά και ψευδώς αρνητικά επειδή δεν μπορεί να ελέγξει τη δυναμική συμπεριφορά. Από την άλλη πλευρά, η δυναμική ανάλυση μας δίνει ψευδώς αρνητικά, επειδή δεν μπορεί να εντοπίσει ορισμένα τρωτά σημεία χωρίς να εξετάσει τον πηγαίο κώδικα. Αλλά αυτό που εντόπισε ήταν μια πραγματική ευπάθεια επειδή πραγματοποίησε μια πραγματική επίθεση. Επομένως, δεν έχουμε ψευδώς θετικά και άρα προκύπτει πολύ υψηλή ακρίβεια (high precision).

Παρατηρούμε πως η ακρίβεια παραμένει υψηλή και για τους δύο τύπους ανάλυσης. Επίσης, τα τρωτά σημεία που εντοπίζονται από τους δύο αυτούς τύπους ανάλυσης μπορούν να αλληλοσυμπληρώνονται (μερικά εντοπίζονται μόνο με δυναμική ανάλυση και άλλα μόνο με στατική ανάλυση). Συνεπώς, είναι καλύτερο να πραγματοποιούνται και οι δύο τύποι ανάλυσης όσο το δυνατόν περισσότερο για να έχουμε περισσότερη ανάκληση και σχεδόν εξίσου υψηλή ακρίβεια.

Σημεία εντοπισμού.

Η στατική ανάλυση, λόγω της φύσης της, προσδιορίζει επακριβώς τα σημεία στον πηγαίο κώδικα όπου εντοπίζονται τρωτότητες. Αντιθέτως, στην δυναμική ανάλυση, δεν προσδιορίζονται επακριβώς τα σημεία / μέρη της εφαρμογής όπου εντοπίζονται οι τρωτότητες. Παραδείγματος χάριν, μια τρωτότητα θα εντοπιστεί σε κάποια ιστοσελίδα, οπότε δεν είναι δυνατόν να προσδιοριστεί άμεσα ο ακριβής κώδικας της εφαρμογής που οδηγεί σε αυτή την τρωτότητα.

Ταχύτητα εκτέλεσης.

Η στατική ανάλυση κώδικα εκτελείται πιο γρήγορα, καθώς σαρώνει και αναλύει τον κώδικα της εφαρμογής χρησιμοποιώντας γρήγορες τεχνικές (ανάλυσης). Αντιθέτως, η απόδοση σάρωσης είναι κακή στην δυναμική ανάλυση, διότι η διαδικασία δυναμικής σάρωσης είναι χρονοβόρα.

Δυνατότητα εκτέλεσης.

Εφαρμογές που χρησιμοποιούν κάποιον μηχανισμό ταυτοποίησης (authentication), όπως για παράδειγμα κάποιο IDS (Intrusion Detection System) (Σύστημα Ανίχνευσης Εισβολών), για έλεγχο πρόσβασης του χρήστη μπορούν να αποκλείσουν την σάρωση στην περίπτωση της δυναμικής ανάλυσης. Αντιθέτως, αυτό δεν μπορεί να συμβεί στην στατική ανάλυση.

Χρήση πόρων.

Η στατική ανάλυση δεν απαιτεί κατά την εκτέλεσή της αρκετούς πόρους από το σύστημα φιλοξενίας ενώ όπως αναφέραμε εκτελείται πολύ γρήγορα. Αντιθέτως, η δυναμική ανάλυση είναι χρονοβόρα και απαιτεί αρκετούς πόρους από το σύστημα στο οποίο εκτελείται. Μάλιστα, πολλές φορές η τοπική σάρωση DAST δεν είναι δυνατή λόγω ανεπάρκειας πόρων στο σύστημα φιλοξενίας της εφαρμογής. Αλλά ακόμα και αν οι πόροι στο σύστημα είναι επαρκής, η επίπονη δέσμευση των πόρων του συστήματος θα οδηγεί σίγουρα σε κακή απόδοση της εφαρμογής ιστού που δοκιμάζεται. Μάλιστα, η σάρωση DAST μπορεί να θεωρηθεί και επικίνδυνη διότι μπορεί να οδηγήσει στην παύση εκτέλεσης της εφαρμογής. Κάτι τέτοιο προφανώς δεν μπορεί να συμβεί στην περίπτωση της στατικής ανάλυσης / σάρωσης.

Λαμβάνοντας υπόψην τα παραπάνω, καταλήγουμε στο συμπέρασμα ότι ο χρήστης θα πρέπει να επιλέξει μια από τις δύο τεχνικές ανίχνευσης, συγκρίνοντας τα πλεονεκτήματα και μειονεκτήματα της κάθε μιας, ανάλογα με τις ανάγκες και προτιμήσεις του, την τρέχουσα φάση ανάπτυξης της εφαρμογής καθώς και την τρέχουσα κατάσταση του περιβάλλοντος φιλοξενίας της εφαρμογής. Αν έχουμε μη πλήρη κώδικα όπου λείπει η γραφική διεπαφή ή η REST διεπαφή, τότε εφαρμόζουμε μόνο SAST. Από την άλλη, αν έχουμε πλήρη ή μη πλήρη κώδικα με γραφική / REST διεπαφή, μπορούμε να έχουμε SAST & DAST, ανεξάρτητα αν βρισκόμαστε σε περιβάλλον ανάπτυξης ή δοκιμής. Αλλά η εκτέλεση DAST θα είναι δυνατή εφόσον μπορεί να την “αντέξει” το περιβάλλον φιλοξενίας της εφαρμογής.

Δεδομένου ότι η επιλογή εξαρτάται από τις ανάγκες του χρήστη, είναι σωστό να επιτρέπουμε στον χρήστη να επιλέγει την τεχνική σάρωσης που ταιριάζει καλύτερα στις απαιτήσεις και προτιμήσεις του. Εφόσον ο χρήστης χρειάζεται την υψηλότερη δυνατή ανάκληση ανίχνευσης, τότε, δεδομένου ότι κάθε τεχνική μπορεί να ανιχνεύσει διαφορετικούς τύπους τρωτών σημείων, μπορούμε να συνδυάσουμε τις τεχνικές αυτές για να επιτύχουμε υψηλότερα ποσοστά ανάκτησης / ανάκλησης. Μπορούμε, επίσης, να συνδυάσουμε εργαλεία από την ίδια τεχνική για την καλύτερη δυνατή ανάκτηση με την λογική πως κάθε εργαλείο (ίδιου τύπου τεχνικής) δεν είναι πάντοτε τέλειο και μπορεί να έχει διαφορετικά δυνατά και αδύνατα σημεία σε σχέση με ένα άλλο. Ως αποτέλεσμα, αναμένουμε να επιτύχουμε την υψηλότερη δυνατή απόδοση ανάκτησης με σταθερά υψηλή ακρίβεια

(precision), επιτρέποντας στους χρήστες να αντιμετωπίζουν όλες τις πιθανές ευπάθειες στις εφαρμογές τους.

Συμπερασματικά, για να καλύψουμε τις διαφορετικές ανάγκες χρηστών ως προς το είδος ανάλυσης, στα πλαίσια της τρέχουσας διπλωματικής παρέχουμε ένα ευέλικτο εργαλείο που μπορεί είτε να εκτελεί μεμονωμένα εργαλεία από ένα μόνο είδος σάρωσης είτε πολλαπλά εργαλεία και από τα δύο είδη σάρωσης. Στην πρώτη περίπτωση, καλύπτουμε τις συγκεκριμένες ανάγκες ενός χρήστη αλλά και την τρέχουσα κατάσταση ανάπτυξης του λογισμικού του. Στη δεύτερη περίπτωση, καλύπτουμε και την ανάγκη που αφορά την επίτευξη της υψηλότερης δυνατής ακρίβεια.

Παρακάτω παρατίθενται σε μορφή πίνακα οι διαφορές μεταξύ SAST και DAST εργαλείων.

Πίνακας 1. Διαφορές μεταξύ σάρωσης DAST και SAST

SAST	**DAST**
White Box Security Testing: Ο δοκιμαστής πρέπει να έχει πρόσβαση στο πλαίσιο (framework) ανάπτυξης και την υλοποίηση της εφαρμογής. Η εφαρμογή δοκιμάζεται στο εσωτερικό της. Αυτός ο τύπος δοκιμής εκπροσωπεί την προσέγγιση από μεριάς προγραμματιστή.	**Black Box Security Testing**: Ο δοκιμαστής δεν έχει γνώση των τεχνολογιών ή των πλαισίων που έχουν χρησιμοποιηθεί για την υλοποίηση. Η εφαρμογή δοκιμάζεται εξωτερικά. Αυτός ο τύπος προσέγγισης εκπροσωπεί τον επιτιθέμενο.
Απαιτείται πηγαίος κώδικας: Η στατική ανάλυση δεν χρειάζεται εφαρμογή που έχει αναπτυχθεί πλήρως. Αναλύει τον πηγαίο κώδικα χωρίς να εκτελεστεί η εφαρμογή.	**Απαιτείται η εφαρμογή να είναι διαθέσιμη**: Η δυναμική ανάλυση δεν χρειάζεται πηγαίο κώδικα. Αναλύει την εφαρμογή την στιγμή που αυτή εκτελείται. Η εφαρμογή δεν χρειάζεται να είναι ολοκληρωμένη αλλά θα πρέπει να έχει κάποια διεπαφή χρήσης (π.χ. γραφική ή REST).
Φθηνότερη λύση για την διόρθωση τρωτοτήτων: Αυτό συμβαίνει επειδή οι τρωτότητες μπορούν να ανιχνευθούν νωρίτερα στον κύκλο ανάπτυξης λογισμικού, και είναι πιο	**Ακριβότερη λύση για την διόρθωση τρωτοτήτων**: Αυτό συμβαίνει επειδή υπάρχει πιθανότητα (ιδιαίτερα όταν η εφαρμογή είναι σχεδόν ολοκληρωμένη) οι τρωτότητες να ανιχνεύονται

εύκολο και γρήγορο να αντιμετωπιστούν. Συνεπώς τα ευρήματα μπορούν να διορθωθούν πριν η εφαρμογή μπει στον κύκλο της διασφάλισης ποιότητας.	προς το τέλος του κύκλου ανάπτυξης λογισμικού, οπότε η αντιμετώπιση τυχόν ευρημάτων θα πρέπει να γίνει σε μετέπειτα στάδιο. Συνεπώς, τα κρίσιμα ευρήματα μπορούν να διορθωθούν σε κάποια επείγουσα έκδοση.
Δεν μπορούν να εντοπιστούν τρωτότητες σε πραγματικό χρόνο εκτέλεσης και που αφορούν θέματα με το περιβάλλον: Αυτό συμβαίνει επειδή τα εργαλεία στατικής ανίχνευσης επικεντρώνονται στον πηγαίο κώδικα και όχι στην δυναμική συμπεριφορά της εφαρμογής.	**Μπορούν να εντοπιστούν τρωτότητες σε πραγματικό χρόνο εκτέλεσης και που αφορούν θέματα με το περιβάλλον**: Αυτό συμβαίνει επειδή τα εργαλεία δυναμικής ανίχνευσης εφαρμόζονται στον πραγματικό χρόνο εκτέλεσης μια εφαρμογής.
Τυπικά, υποστηρίζονται όλα τα είδη λογισμικού: Εφαρμογές ιστού, υπηρεσίες ιστού, πυκνοί πελάτες (thick clients) κ.α.	**Τυπικά, υποστηρίζονται μόνο εφαρμογές ιστού και υπηρεσίες ιστού**: Η δυναμική ανάλυση δεν είναι χρήσιμη για άλλα είδη λογισμικού.
Χρόνος εκτέλεσης: Η στατική ανάλυση είναι αρκετά γρήγορη όσον αφορά τη σάρωση και την ανάλυση του κώδικα.	**Χρόνος εκτέλεσης**: Περιλαμβάνει την αλληλεπίδραση με την εκτελούμενη εφαρμογή με βάση διάφορα σενάρια διείσδυσης, η οποία μπορεί να διαρκέσει περισσότερο, ειδικά για μεγαλύτερες και πιο σύνθετες εφαρμογές.
Είδη τρωτοτήτων στατικής ανάλυσης: Επιθέσεις έκχυσης, μη ασφαλής κρυπτογράφηση, μη ασφαλής αποσειριοποίηση (unsafe deserialization) κ.ά.	**Είδη τρωτοτήτων δυναμικής ανάλυσης**: Εσφαλμένη διαμόρφωση, έκθεση ευαίσθητων δεδομένων, λανθασμένος έλεγχος ταυτότητας κ.α.

2.5 Restful API / Service

Ένα Restful API / Service είναι μια υπηρεσία ιστού (web service) που χαρακτηρίζεται από τα εξής στοιχεία:

- Μια αρχιτεκτονική πελάτη-διακομιστή που αποτελείται από πελάτες, διακομιστές και πόρους, με αιτήματα που διαχειρίζονται μέσω HTTP.
- Stateless επικοινωνία, που σημαίνει ότι δεν αποθηκεύονται πληροφορίες πελάτη μεταξύ των αιτημάτων και κάθε αίτημα είναι ξεχωριστό και μη συνδεδεμένο.
- Μια ομοιόμορφη διεπαφή μεταξύ των στοιχείων, έτσι ώστε οι πληροφορίες να μεταφέρονται σε τυπική μορφή. Αυτό απαιτεί ότι:
    1. Οι πόροι που ζητούνται είναι αναγνωρίσιμοι και ξεχωριστοί από τις αναπαραστάσεις που αποστέλλονται στον πελάτη.
    2. Οι πόροι μπορούν να χειριστούν από τον πελάτη.
    3. Οι απαντήσεις που επιστρέφονται στον πελάτη έχουν αρκετές πληροφορίες για κάθε πόρο.
    4. Το υπερκείμενο είναι διαθέσιμο, πράγμα που σημαίνει ότι μετά την πρόσβαση σε έναν πόρο, ο πελάτης θα πρέπει να μπορεί να χρησιμοποιεί υπερσυνδέσμους για να βρει άλλους σχετικούς πόρους.
    5. Ένα RESTful API μπορεί να σχεδιαστεί ως σύστημα πολλαπλών επιπέδων για την επίτευξη καλύτερης οργάνωσης, επεκτασιμότητας και συντήρησης. Στο πλαίσιο των RESTful APIs, ένα σύστημα πολλαπλών επιπέδων αναφέρεται συνήθως στην ανάλυση του API σε πολλαπλά επίπεδα, με το καθένα να εξυπηρετεί διαφορετικές ανάγκες. Αυτή η προσέγγιση ακολουθεί τις αρχές του ‘Διαχωρισμού των Ανησυχιών και της Ενιαίας Ευθύνης’ (separation of concerns & single responsibility), καθιστώντας το API ευκολότερο στην κατανόηση, την ανάπτυξη και τη διατήρηση. Αλλά μπορεί να περιλαμβάνει και τη χρήση ενδιάμεσων συστατικών μεταξύ του πελάτη και του εξυπηρετητή, όπως είναι τα πληρεξούσια (proxies).

Αν και το REST API έχει αυτά τα κριτήρια για να συμμορφώνεται, όχι μόνο η συμμόρφωση δεν είναι δύσκολη αλλά εξακολουθεί να θεωρείται πιο εύκολο στη χρήση σε σχέση με SOAP υπηρεσίες

ιστού διότι οι τελευταίες χρησιμοποιούν πρότυπα πρωτόκολλα όπως το SOAP (Simple Object Access Protocol) (πρωτόκολλο πρόσβασης απλού αντικειμένου), το οποίο έχει συγκεκριμένες απαιτήσεις, όπως η ανταλλαγή μηνυμάτων XML, και είναι "βαρύ".

ΚΕΦΑΛΑΙΟ 3 – ΠΑΡΟΜΟΙΕΣ ΕΡΓΑΣΙΕΣ

3.1 IBM Security AppScan

Περιγραφή: Το IBM Security AppScan[8] είναι μια ολοκληρωμένη λύση δοκιμών ασφαλείας εφαρμογών που συνδυάζει τεχνικές SAST και DAST. Η μηχανή στατικής ανάλυσης σαρώνει τον πηγαίο κώδικα εφαρμογής για πιθανά ζητήματα ασφάλειας, ενώ η μηχανή δυναμικής ανάλυσης δοκιμάζει εφαρμογές σε περιβάλλοντα χρόνου εκτέλεσης για ευπάθειες. Το AppScan περιλαμβάνει, επίσης, υποστήριξη για δοκιμές ασφαλείας εφαρμογών για κινητά, καθώς και ενσωματώσεις με άλλα εργαλεία της IBM και συστήματα ασφαλείας.

Βιογραφία: Η IBM Security είναι ένα τμήμα της IBM που παρέχει λύσεις και υπηρεσίες ασφάλειας σε επίπεδο επιχείρησης.

Σύγκριση: Συγκριτικά με την δική μας προσέγγιση, και αυτή η εργασία συνδυάζει DAST και SAST τεχνικές, όμως χρησιμοποιεί προκαθορισμένα εργαλεία, ενώ η λειτουργικότητά της εξαρτάται αποκλειστικά από την αγορά και εγκατάσταση της εφαρμογής. Ο χρήστης χρειάζεται χρόνο και εκπαίδευση για να χρησιμοποιήσει την εφαρμογή σωστά μέσω του GUI που προσφέρει η εφαρμογή. Η εφαρμογή της IBM εγκαθίσταται σε έναν υπολογιστή και είναι διαθέσιμη αποκλειστικά στο εσωτερικό του. Στην δική μας περίπτωση, συνδυάζουμε πολλαπλά εργαλεία ανίχνευσης ανοικτού κώδικα, η εφαρμογή μας προσφέρεται δωρεάν, μπορεί να ενοποιηθεί εν γένει με άλλες εφαρμογές/υπηρεσίες (π.χ. ασφάλειας) μέσω του RESTful API που προσφέρει ενώ μπορεί να παρέχεται ως υπηρεσία σε πολλούς χρήστες ενός ή παραπάνω συστημάτων. Επίσης, η εφαρμογή είναι αρκετά εύχρηστη και κατανοητή και μπορεί να χρησιμοποιηθεί εύκολα από έναν χρήστη χωρίς κάποια ιδιαίτερη εκπαίδευση.

3.2 Συνδυασμός Διαδραστικών, Στατικών & Δυναμικών Σαρωτών

Στο άρθρο [13] η εργασία επικεντρώνεται στον συνδυασμό SAST, DAST αλλά και IAST εργαλείων για την εύρεση των OWASP Top-10 ευπαθειών με μεγαλύτερη ακρίβεια.

Η μεθοδολογία για των συνδυασμό των εργαλείων επικεντρώνεται στις εξής τεχνικές:

- Επιλογή έργου συγκριτικής αξιολόγησης OWASP
- Επιλογή εργαλείων SAST, DAST και IAST

[8] https://www.hcltechsw.com/appscan

- Σε αυτήν τη μεθοδολογική προδιαγραφή, οι συγγραφείς έχουν επιλέξει έξι εμπορικά εργαλεία και εργαλεία ανοιχτού κώδικα.
- Εκτέλεση των επιλεγμένων εργαλείων σε ένα έργο συγκριτικής αξιολόγησης OWASP με προεπιλεγμένες ρυθμίσεις παραμέτρων για κάθε εργαλείο.
- Επιλογή των κατάλληλων μετρήσεων για να αναλυθούν τα αποτελέσματα σύμφωνα με τρία διαφορετικά επίπεδα σημασίας εφαρμογών ιστού.
- Υπολογισμός των κατάλληλων μετρικών.
- Ανάλυση και ταξινόμηση των αποτελεσμάτων.

Ως συμπέρασμα της εργασίας αυτής, οι συγγραφείς καταλήγουν στο συμπέρασμα ότι ο συνδυασμός των IAST, SAST και των DAST εργαλείων δίνει τα καλύτερα αποτελέσματα, καθώς παράγονται πολλά True Positives (TP) [13]. Όμως, οι συγγραφείς δεν παρέχουν μια εφαρμογή ή μια υπηρεσία ιστού που να συνδυάζει τα εργαλεία αυτά.

3.3 Μια Υπηρεσία Ανίχνευσης Τρωτοτήτων

Το άρθρο [14] επικεντρώνεται σε ένα ανοικτού κώδικα project, το W3AF, για το οποίο μπορείτε να βρείτε περισσότερες πληροφορίες εδώ[9]. Αποτελεί ουσιαστικά ένα πλαίσιο (framework) που προσπαθεί να εντοπίσει όλες τις ευπάθειες σε μια εφαρμογή. Παρ'όλα αυτά, απαιτεί καλή γνώση των όρων της Ασφάλειας Πληροφοριακών Συστημάτων (ΠΣ) από τον χρήστη, και είναι αρκετά περίπλοκο και χρονοβόρο.

Με βάση τη μελέτη των υφιστάμενων τεχνολογιών ασφάλειας εφαρμογών Ιστού, το άρθρο αρχικά εξετάζει τους σαρωτές ευπάθειας εφαρμογών ιστού ανοιχτού κώδικα. Στην συνέχεια, οι συγγραφείς επέλεξαν το W3af ως πακέτο διεπαφής ιστού και σχεδιάζοντας τη δοκιμαστική μονάδα, βελτίωσαν τη δυνατότητα σάρωσης για ευπάθειες Clickjacking, ειδικά σε HTML5. Τέλος, δημιουργήθηκε ένα προσαρμοσμένο σενάριο και χρησιμοποιήθηκε σε ένα πραγματικό τεστ ανάλυσης, από το οποίο προκύπτει ότι μπορούν να εντοπιστούν σχετικές ευπάθειες και η αποτελεσματικότητα σάρωσης μπορεί να βελτιωθεί σημαντικά χωρίς να επηρεαστούν τα αποτελέσματα σάρωσης.

Σε σύγκριση με τη δική μας εργασία, το άρθρο απλώς χρησιμοποιεί ένα σαρωτή συγκεκριμένου είδους και όχι πολλαπλούς του ίδιου και διαφορετικών ειδών. Επομένως, θα έχει μικρότερη σχετική ακρίβεια σάρωσης σε σχέση με το δικό μας σύνθετο εργαλείο.

[9] http://w3af.org/

3.4 Αυτόματος Σαρωτής Τρωτοτήτων για Εφαρμογές Ιστού

Το άρθρο [16] επικεντρώνεται στην υλοποίηση ενός αυτοματοποιημένου σαρωτή για εφαρμογές ιστού. Οι συγγραφείς, βασιζόμενοι στο γεγονός ότι οι περισσότεροι σαρωτές εστιάζουν μόνο σε μοναδικούς στόχους χωρίς να χρησιμοποιούν άλλες χρήσιμες πληροφορίες, προτείνουν έναν αυτόματο σαρωτή ευπαθειών ιστού που ενσωματώνει τη συλλογή πληροφοριών με τον εντοπισμό ευπάθειας, ο οποίος καθοδηγείται από χρήσιμες πληροφορίες που συλλέγονται. Επομένως, όταν ο προτεινόμενος σαρωτής πετύχει έναν συγκεκριμένο στόχο, θα δημιουργηθεί μια βαθύτερη και πιο ολοκληρωμένη ικανότητα ανίχνευσης, η οποία μπορεί να βοηθήσει στην επίτευξη ιδανικών επιδόσεων. Οι συγγραφείς συγκρίνουν μάλιστα το εργαλείο που δημιούργησαν με το OWASP Zap[10], ώστε να δείξουν ότι τα αποτελέσματα με την χρήση του σαρωτή που δημιούργησαν με βάση τις παραπάνω προσεγγίσεις, είναι πιο αποτελεσματικός και ακριβής συγκριτικά με άλλα εργαλεία [16].

Το εν λόγω, όμως, εργαλείο δεν συνδυάζει σαρωτές διαφορετικών ειδών (στατικής και δυναμικής ανάλυσης). Συνεπώς το δικό μας εργαλείο, όπως θα δούμε και μετέπειτα, προσφέρει μεγαλύτερη ακρίβεια.

[10] https://www.zaproxy.org

ΚΕΦΑΛΑΙΟ 4 – ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ

Στο παρόν κεφάλαιο θα αναφερθούμε στα εξής:

1. Στις γλώσσες προγραμματισμού που χρησιμοποιήθηκαν για την υλοποίηση της εφαρμογής μας.
2. Στο πλαίσιο ανάπτυξης (Development Framework) που επιλέχθηκε με βάση την ικανοποίηση των απαιτήσεων και την αντιμετώπιση των προκλήσεων του έργου.
3. Στην ανάλυση της RESTful Web Υπηρεσίας που υλοποιήθηκε, συμπεριλαμβανομένων των APIs και των μεθόδων της.
4. Στη βάση δεδομένων που χρησιμοποιήθηκε μαζί με το αντίστοιχο εργαλείο αντικειμενο-σχεσιακής αντιστοίχισης (Object-Relational Mapping (ORM) tool – Hibernate).
5. Στην ανάλυση των ανοικτού κώδικα εργαλείων που χρησιμοποιήθηκαν για τον σκοπό της εργασίας.
6. Στην αρχιτεκτονική του συστήματος.

4.1 Γλώσσες Προγραμματισμού

Για την υλοποίηση της εφαρμογής, επιλέχθηκαν και χρησιμοποιήθηκαν οι παρακάτω **<u>γλώσσες προγραμματισμού:</u>**

- ***<u>Java</u>***: Μια εφαρμογή ιστού επιτρέπει στους χρήστες απομακρυσμένα να έχουν πρόσβαση σε δεδομένα, πληροφορίες, γνώση και λειτουργικότητα. Η Java είναι ιδανική για την ανάπτυξη μεγάλων εφαρμογών ιστού λόγω της ικανότητάς της να αλληλεπιδρά με μεγάλο αριθμό συστημάτων, τα οποία μπορούν να καλύπτουν μέρος της λειτουργικότητας των εφαρμογών αυτών. Υπηρεσίες, όπως ομότιμου ιστού ή δικτύου (peer network) και συνδεσιμότητας βάσεων δεδομένων (database connectivity), μπορούν επίσης να προσπελαστούν από εφαρμογές ιστού Java. Επομένως, στη δική μας περίπτωση, για την υλοποίηση της εφαρμογής σάρωσης τρωτοτήτων, η Java θεωρήθηκε η καλύτερη επιλογή.
- ***<u>Javascript</u>***: Αφορά περισσότερο την ανάπτυξης της διεπαφής χρήσης της εφαρμογής και την εμφάνιση των δεδομένων / πληροφοριών σε αυτήν, καθώς και την ενσωμάτωση σε αυτήν εύχρηστων στοιχείων διεπαφής χρήσης (UI Elements) (πχ. κουμπιών, inputs, κα.) που βοηθούν την διαδραστικότητα (είτε τοπική είτε με τον εξυπηρετητή της εφαρμογής). Έχει

χρησιμοποιηθεί templating για την διασύνδεση του νωτιαίου μέρους της εφαρμογής (back-end) (Java) με το μετωπικό (front-end) (HTML/CSS/JS).

4.2 Πλαίσιο Ανάπτυξης (Development Framework)

Το *πλαίσιο* ανάπτυξης που χρησιμοποιήθηκε για να εξυπηρετήσει τον συνδυασμό όλων των παραπάνω, είναι το ***Java Spring Boot Framework:***

Το πλαίσιο (framework) Spring προσφέρει την δυνατότητα του λεγόμενου “dependency injection” (έκχυση εξαρτήσεων) που επιτρέπει στα αντικείμενα να ορίζουν τις δικές τους εξαρτήσεις, οι οποίες έπειτα εισάγονται σε αυτά κατά τον χρόνο εκτέλεσης. Αυτό είναι δυνατό μέσω του Spring Container, ο οποίος βρίσκεται στον πυρήνα του Spring. Η δουλειά του Spring Container είναι να δημιουργεί αντικείμενα, να τα δένει μεταξύ τους, να τα παραμετροποιεί, καθώς και επίσης να διαχειρίζεται τον κύκλο ζωής τους. Αυτό επιτρέπει στους προγραμματιστές να δημιουργούν αρθρωτές εφαρμογές που αποτελούνται από χαλαρά συζευγμένα στοιχεία, τα οποία είναι ιδανικά για την ανάπτυξη μικροϋπηρεσιών (microservices) και κατανεμημένων εφαρμογών.

Το Spring Framework προσφέρει, επίσης, ενσωματωμένη υποστήριξη για τυπικές εργασίες που χρειάζεται να εκτελέσει μια εφαρμογή, όπως δέσμευση δεδομένων (data binding), μετατροπή τύπου (type casting), επικύρωση (validation), χειρισμός εξαιρέσεων (exception handling), καθώς διαχείριση πόρων και συμβάντων (resource & event management). Επιπλέον, απλοποιεί και πιο περίπλοκες διεργασίες, όπως η διαχείριση και επεξεργασία δεδομένων στην βάση (π.χ. χρήση JPA Repository για αντικειμενο-σχεσιακή αντιστοίχιση). Επιπλέον, ενσωματώνεται με διάφορες τεχνολογίες Java EE, όπως JAX-RS (για RESTful υπηρεσίες) και JSONB (API για JSON Binding). Εν ολίγοις, το πλαίσιο Spring παρέχει στους προγραμματιστές όλα τα εργαλεία και τις δυνατότητες που χρειάζονται για τη δημιουργία χαλαρά συζευγμένων, ανεξάρτητων πλατφόρμας εφαρμογών ιστού Java EE που εκτελούνται σε οποιοδήποτε περιβάλλον.

Όσο ικανό και περιεκτικό και αν είναι το πλαίσιο Spring, εξακολουθεί να απαιτεί σημαντικό χρόνο και γνώσεις για τη διαμόρφωση, τη ρύθμιση και την ανάπτυξη εφαρμογών Spring. Το Spring Boot μετριάζει αυτή την προσπάθεια με τρεις σημαντικές δυνατότητες: αυτόματη διαμόρφωση, κατάλληλη διαμόρφωση εξαρτήσεων εκκίνησης και αυτονομία εκτέλεσης εφαρμογών.

Η αυτόματη διαμόρφωση σημαίνει ότι οι εφαρμογές αρχικοποιούνται με προκαθορισμένες εξαρτήσεις που δεν χρειάζεται να διαμορφωθούν με μη αυτόματο τρόπο. Καθώς το Java Spring Boot διαθέτει ενσωματωμένες δυνατότητες αυτόματης διαμόρφωσης, διαμορφώνει αυτόματα τόσο το υποκείμενο πλαίσιο Spring όσο και τα πακέτα τρίτων (μερών) με βάση τις ρυθμίσεις του χρήστη (και με βάση τις βέλτιστες πρακτικές, γεγονός που βοηθά στην αποφυγή σφαλμάτων).

Το Spring Boot χρησιμοποιεί μια προσεγμένη προσέγγιση για την προσθήκη και τη διαμόρφωση εξαρτήσεων εκκίνησης, με βάση τις ανάγκες του έργου. Το Spring Boot επιλέγει ποια πακέτα θα εγκαταστήσει και ποιες προεπιλεγμένες τιμές θα χρησιμοποιήσει, αντί να απαιτεί από τον χρήστη να λαμβάνει όλες αυτές τις αποφάσεις μόνος του και να ρυθμίζει τα πάντα με μη αυτόματο τρόπο.

Το Spring Boot επιτρέπει στον χρήστη να δημιουργεί αυτόνομες εφαρμογές που εκτελούνται μόνες τους, χωρίς να βασίζονται σε εξωτερικό διακομιστή ιστού ή εφαρμογών (web or application server), ενσωματώνοντας έναν web / servlet container, όπως ο Tomcat ή ο Netty, σε αυτές κατά τη διαδικασία προετοιμασίας. Ως αποτέλεσμα, μια εφαρμογή μπορεί να εκκινηθεί σε οποιαδήποτε πλατφόρμα, π.χ. πατώντας απλά την εντολή "Run" σε κάποιο περιβάλλον ενοποιημένης ανάπτυξης (Integrated Development Environment – IDE) ή εκτελώντας κατάλληλη εντολή στη γραμμή εντολών ενός τερματικού.

4.3 RESTful API/Υπηρεσία Τεχνολογία Restful Υπηρεσίας / Σχεδίαση Μεθόδων

Για την υλοποίηση της εφαρμογής, χρησιμοποιήθηκε η τεχνολογία Java Spring Boot, και συγκεκριμένα εκμεταλλευτήκαμε τις παρακάτω δυνατότητές της.

4.3.1 Κατηγορίες Ελεγκτών

Οι ελεγκτές (στην δική μας περίπτωση MainController, ScanController) χειρίζονται αιτήματα HTTP και ορίζουν τελικά σημεία (endpoints) για την RESTful υπηρεσία μας. Στην περίπτωσή μας, αυτοί οι ελεγκτές περιλαμβάνουν μεθόδους που αντιστοιχούν σε διαφορετικά ρήματα HTTP (ειδικότερα GET και POST στην δική μας περίπτωση) και διαδρομές (URL), οι οποίες αντιπροσωπεύουν διαφορετικούς πόρους API. Ο MainController συγκεκριμένα χειρίζεται τα μονοπάτια (paths) (π.χ, index) της εφαρμογής, και το χειρισμό λαθών (error handling) (/error).

Παρακάτω παρατίθενται τα τελικά σημεία που προσφέρονται από την RESTful υπηρεσία μας προς κλήση:

- /dast με επιτρεπτές μεθόδους:
  - GET, όπου επιστρέφεται στον χρήστη μια Όψη (View) για δυναμική ανάλυση, η οποία θα εμφανίσει στον χρήστη μια φόρμα για την εισαγωγή του URL της εφαρμογής που θα σαρωθεί.
  - POST, όπου ο χρήστης στέλνει ως όρισμα στο Payload ένα URL (String), δηλ. Το τελικό σημείο της εφαρμογής που δοκιμάζεται. Αφού εκτελεστεί η μέθοδος, επιστρέφει στον χρήστη ως απάντηση ένα .zip αρχείο που περιέχει τις αναφορές όλων των εργαλείων δυναμικής ανάλυσης που χρησιμοποιήθηκαν. Η μέθοδος ενεργοποιείται με το πάτημα του κουμπιού "Submit" από τον χρήστη (trigger event) στην περίπτωση της εφαρμογής ιστού που περιτυλίγει την RESTful υπηρεσία.

- /sast με επιτρεπτές μεθόδους:
  - GET, όπου επιστρέφεται στον χρήστη μια Όψη για στατική ανάλυση, η οποία θα εμφανίσει στον χρήστη την φόρμα για εισαγωγή ZIP αρχείου του πηγαίου κώδικα της εφαρμογής που θα σαρωθεί.
  - POST, όπου ο χρήστης στέλνει ως όρισμα στο Payload ένα αρχείο (binary file) που περιλαμβάνει ένα συμπιεσμένο αρχείο με όλη την δομή του πηγαίου κώδικα. Εφόσον εκτελεστεί η μέθοδος, επιστρέφει στον χρήστη ως απάντηση ένα .zip αρχείο που περιέχει τις αναφορές όλων των στατικών εργαλείων που χρησιμοποιήθηκαν. Προφανώς, ο χρήστης μπορεί να παραθέσει και ZIP αρχείο που να περιλαμβάνει μόνο αρχεία ενός συστατικού μέρους της εφαρμογής του. Η μέθοδος ενεργοποιείται με το πάτημα του κουμπιού "Submit" από τον χρήστη (trigger event) στην περίπτωση της εφαρμογής ιστού που περιτυλίγει την RESTful υπηρεσία.

- /combined
  - GET, όπου επιστρέφεται στον χρήστη μια Όψη για συνδυασμένη ανάλυση, η οποία θα εμφανίσει στον χρήστη την φόρμα για εισαγωγή του URL της εφαρμογής προς δυναμική σάρωση καθώς και ένα πεδίο για την εισαγωγή του ZIP αρχείου του πηγαίου κώδικα της εφαρμογής αυτής προς στατική σάρωση.

- POST, όπου ο χρήστης στέλνει ως όρισμα στο Payload ένα URL (String) (τελικό σημείο δοκιμαζόμενης εφαρμογής) και επίσης ένα αρχείο (binary file) (κώδικας εφαρμογής). Η μέθοδος επιστρέφει στον χρήστη ως απάντηση ένα .zip αρχείο που περιέχει τις αναφορές όλων των ειδών εργαλείων, στατικών και δυναμικών, για εκφόρτωση. Η μέθοδος ενεργοποιείται με το πάτημα του κουμπιού "Submit" από τον χρήστη (trigger event) στην περίπτωση της εφαρμογής ιστού που περιτυλίγει την RESTful υπηρεσία.

- /totalScans
  - GET, όπου επιστρέφεται στον χρήστη Όψη που εμφανίζει την φόρμα αναζήτησης αναφορών με βάση κωδικό και εύρος ημερομηνιών της αναφοράς, καθώς και κάτω από αυτήν έναν πίνακα με τα αποτελέσματα ενός συγκεκριμένου αριθμού αναφορών (reports) σάρωσης από την βάση.

- /files/filename:id.zip με επιτρεπτές μεθόδους:
  - GET, όπου επιστρέφεται στον χρήστη το αρχείο αναφοράς σάρωσης προς εκφόρτωση με την μορφή Link (συνδέσμου). Η δυνατότητα εκφόρτωσης ενεργοποιείται με το πάτημα του κουμπιού Link από τον χρήστη (trigger event).

4.3.2 Αντιστοίχιση σε Μεθόδους HTTP

Ο ελεγκτής MainController διαχειρίζεται 2 HTTP GET μεθόδους, τις @GetMapping("/index") και @GetMapping("/error"), όπου επιστρέφονται στον χρήστη οι σελίδες "index", δηλαδή η αρχική σελίδα της εφαρμογής, και η σελίδα "error", δηλαδή η σελίδα σφάλματος όταν κάτι δεν πάει καλά.

Ο ελεγκτής ScanController, ο οποίος αποτελεί το κύριο μέρος της RESTful υπηρεσίας μας που προσφέρει τις βασικές λειτουργικότητες της εφαρμογής μας, αντιστοιχίζει μεθόδους HTTP σε συγκεκριμένες λειτουργίες πάνω σε πόρους που αφορούν συγκεκριμένο μονοπάτι/διαδρομή της υπηρεσίας, όπως POST για αποστολή αρχείων / δεδομένων ή GET για ανάκτηση αποτελεσμάτων σάρωσης. Αναλυτικά έχουμε:

- Οι αντιστοιχίσεις @PostMapping("/sast") και @GetMapping("/sast") ορίζουν τις HTTP μεθόδους και τις διαδρομές πόρων για τη σάρωση SAST, ενώ αντίστοιχα οι αντιστοιχίσεις @PostMapping("/dast") και @GetMapping("/dast") ορίζουν τις HTTP μεθόδους και τις

διαδρομές πόρων για τη σάρωση DAST. Τέλος, οι αντιστοιχίσεις @PostMapping("/combined") και @GetMapping("/combined") ορίζουν τις HTTP μεθόδους και τις διαδρομές πόρων για τη συνδυασμένη σάρωση και των 2 ειδών εργαλείων.

- Επιπλέον, έχουμε τον στολισμό @GetMapping("/totalScans") όπου ορίζεται η HTTP μέθοδος και η διαδρομή πόρων για επιστροφή ενός πίνακα με έναν συγκεκριμένο αριθμό από τα τελευταία αποτελεσμάτα σάρωσης, και τον @PostMapping("/scans") όπου ορίζεται η HTTP μέθοδος και η διαδρομή πόρων για την αναζήτηση ενός συγκεκριμένου αποτελέσματος σάρωσης από τον χρήστη.

4.3.3 Χειρισμός Αιτημάτων και Απαντήσεων

Οι μέθοδοι ελεγκτή μας (δηλ. μέθοδοι που αφορούν τις λειτουργίες που έχουν αντιστοιχιστεί σε ζεύγη HTTP ρήματος (verb) & διαδρομής πόρου) δέχονται αιτήματα και παρέχουν απαντήσεις με βάση το πρωτόκολλο HTTP. Χρησιμοποιούμε στολισμούς (annotations) Spring όπως **@RequestParam** , **@ModelAttribute** και **@PathVariable** για εξαγωγή δεδομένων από αιτήματα και επιστροφή δεδομένων στην απάντηση. Η χρήση του **ModelAndView** και στολισμών όπως **@ResponseBody** και **ResponseEntity** είναι επίσης ενδεικτικές του σχεδιασμού της υπηρεσίας ιστού RESTful. Ειδικότερα:

- **@RequestParam:** Χρησιμοποιείται για την εξαγωγή δεδομένων παραμέτρων επερώτησης (request parameters) σε διευθύνσεις URL ή δεδομένων φόρμας που αποστέλλονται σε αιτήματα HTTP.
- **@ModelAttribute:** Χρησιμοποιείται για τη σύνδεση δεδομένων φόρμας ή ιδιοτήτων μοντέλου σε ένα αντικείμενο-μοντέλο που μπορεί να χρησιμοποιηθεί σε μια μέθοδο.
- **@PathVariable:** Χρησιμοποιείται για την εξαγωγή τιμών / δεδομένων από τη διαδρομή URL, ειδικότερα για τη λήψη παραμέτρων διαδρομής (path parameters) από τη διαδρομή αυτή. Επιστρέφει την τιμή συγκεκριμένης παραμέτρου διαδρομής.
- **ModelAndView:** Αποτελεί μια κλάση στο Java Spring Boot, η οποία χρησιμοποιείται για την ενθυλάκωση τόσο του ονόματος προβολής όσο και των δεδομένων μοντέλου που θα χρησιμοποιηθούν για την απόδοση αυτής της προβολής (view). Αυτή η κλάση περιέχει απλώς και τα δύο για να καταστήσει δυνατό για έναν ελεγκτή να επιστρέψει τόσο το μοντέλο όσο και την προβολή σε μια ενιαία τιμή επιστροφής.

- **@ResponseBody:** Επεξεργάζεται και μετατρέπει την τιμή επιστροφής της στολισμένης μεθόδου στο σώμα απόκρισης, συνήθως σε JSON ή άλλες μορφές, και την επιστρέφει στον πελάτη.
- **@ResponseEntity:** Το ResponseEntity αποτελεί μια κλάση στο Java Spring Boot που αντιπροσωπεύει ολόκληρη την απόκριση HTTP και τα συστατικά της μέρη, όπως κωδικός κατάστασης, κεφαλίδες και σώμα. Επομένως, μπορούμε να το χρησιμοποιήσουμε για να διαμορφώσουμε πλήρως την απόκριση HTTP. Εάν θέλουμε να το χρησιμοποιήσουμε, πρέπει να το επιστρέψουμε από το τελικό σημείο.

4.3.4 Αναγνώριση Πόρων

Το έργο μας ακολουθεί τις αρχές REST χρησιμοποιώντας διευθύνσεις URL για τον εντοπισμό πόρων. Για παράδειγμα, τα /sast, /dast και /combined αντιπροσωπεύουν διαφορετικούς τύπους σαρώσεων (στατική, δυναμική και συνδυασμένη ανάλυση / ανίχνευση τρωτοτήτων, αντίστοιχα) και το /totalScans παρέχει έναν πόρο για την ανάκτηση λίστας όλων των σαρώσεων.

4.3.5 Κωδικός Κατάστασης HTTP

Ο κώδικάς μας επιστρέφει τον κατάλληλο κωδικό κατάστασης HTTP στην απάντηση. Για παράδειγμα, το ResponseEntity.ok() ή το ResponseEntity.notFound() χρησιμοποιούνται για να υποδείξουν την επιτυχία ή την αποτυχία των αιτημάτων API (ειδικότερα την αποτυχία εντοπισμού πόρου στην δεύτερη περίπτωση).

4.3.6 Διαχείριση Σφαλμάτων

Έχουμε εφαρμόσει τη διαχείριση σφαλμάτων στον ελεγκτή μας, όπως το χειρισμό του StorageFileNotFoundException και άλλων εξαιρέσεων. Η διαχείριση αυτή παράγει συγκεκριμένες απαντήσεις για την παροχή ουσιαστικών σχολίων / μηνυμάτων λάθους στους χρήστες του RESTful API μας (χωρίς όμως την συμπερίληψη του stack trace για λόγους ασφάλειας).

4.3.7 Κατηγορία Εξυπηρέτησης

Έχουμε ένα επίπεδο υπηρεσιών με το StorageService και άλλα επίπεδα που ενσωματώνουν την επιχειρηματική λογική και την αλληλεπίδραση με δεδομένα. Αυτός ο διαχωρισμός ακολουθεί τις βέλτιστες πρακτικές για τη δημιουργία RESTful API.

4.3.8 Κατηγορία Αποθήκευσης

Χρησιμοποιούμε το CodeScanRepository για να αλληλεπιδράσουμε με τη βάση δεδομένων και να εκτελούμε λειτουργίες CRUD στα δεδομένα μας (τα οποία σε αυτήν την περίπτωση είναι οι σαρώσεις που έγιναν με τις αναφορές τους).

4.3.9 Σχεδιασμός URI

Έχουμε σχεδιάσει τα URI ώστε να είναι προσανατολισμένα στους πόρους και καλά δομημένα.

4.4 Βάση Δεδομένων

Για την αποθήκευση πληροφοριών και επίσης την ανάκτησή τους χρησιμοποιήθηκε σχεσιακή βάση δεδομένων (σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων - ΣΔΔΒΔ) και επομένως η γλώσσα **SQL.** Η SQL βοηθά μέσω της χρήσης δηλώσεων διαχείρισης δεδομένων στην δημιουργία μιας βάσης σε οποιοδήποτε περιβάλλον στο οποίο έχει εγκατασταθεί το ΣΣΔΒΔ. Έπειτα, η βάση διασυνδέεται με την εφαρμογή μέσω του Spring Boot πλαισίου. Ο λόγος που χρησιμοποιήθηκε σχεσιακή βάση δεδομένων είναι επειδή χρησιμοποιήθηκε το πλαίσιο Hibernate (για την υλοποίηση ORM) προς ικανοποίηση των απαιτήσεων της εφαρμογής μας. Έτσι, μπορούμε να έχουμε δομημένη αποθήκευση πληροφορίας, ακεραιότητα δεδομένων και επεκτασιμότητα.

Η χρήση μιας σχεσιακής βάσης δεδομένων, ιδιαίτερα όταν χρησιμοποιείται Hibernate (ORM) σε μια εφαρμογή Java Spring, όπως η MySQL, είναι μια κοινή και ευρέως διαδεδομένη πρακτική για διάφορους λόγους:

- **Ακεραιότητα δεδομένων:** Οι σχεσιακές βάσεις δεδομένων παρέχουν έναν δομημένο τρόπο αποθήκευσης δεδομένων, διασφαλίζοντας την ακεραιότητα των δεδομένων μέσω της χρήσης περιορισμών, σχέσεων και ιδιοτήτων ACID (Atomicity, Consistency, Isolation,

Durability). Αυτό σημαίνει ότι τα δεδομένα αποθηκεύονται αξιόπιστα και με συνέπεια, ακόμη και με την παρουσία ταυτόχρονων λειτουργιών ή αστοχιών του συστήματος.

- **Υποστήριξη SQL:** Οι σχεσιακές βάσεις δεδομένων χρησιμοποιούν SQL (Structured Query Language) ως τη τυπική γλώσσα ερωτημάτων, διευκολύνοντας την αλληλεπίδραση και τον χειρισμό δεδομένων. Το Hibernate, όταν χρησιμοποιείται σε συνδυασμό με το Spring, παρέχει ένα πλαίσιο αντικειμενο-σχεσιακής αντιστοίχισης (ORM) που αφαιρεί μεγάλο μέρος της εργασίας SQL, επιτρέποντας στους προγραμματιστές να εργάζονται με αντικείμενα Java (όπως έχουν δηλαδή συνηθήσει) αντί να γράφουν σύνθετα ερωτήματα SQL.
- **Επεκτασιμότητα:** Οι σχεσιακές βάσεις δεδομένων είναι γνωστές για την επεκτασιμότητα και την ικανότητά τους να χειρίζονται μεγάλα σύνολα δεδομένων.
- **Συνέπεια δεδομένων:** Οι σχεσιακές βάσεις δεδομένων προσφέρουν υποστήριξη συναλλαγών, η οποία διασφαλίζει ότι οι λειτουργίες είτε πετυχαίνουν πλήρως είτε όχι διατηρώντας πάντοτε τη βάση δεδομένων σε συνεπή κατάσταση. Αυτό είναι ζωτικής σημασίας για τη διατήρηση της ακεραιότητας των δεδομένων. Επιπλέον, με την ύπαρξη ιδιοτήτων ACID (Atomicity, Consistency, Isolation, Durability) διασφαλίζεται ότι οι συναλλαγές της βάσης δεδομένων είναι αξιόπιστες και ότι τα δεδομένα παραμένουν συνεπή ακόμη και σε περίπτωση αστοχιών.
- **Δομημένα δεδομένα:** Οι σχεσιακές βάσεις δεδομένων είναι ιδανικές για δομημένα δεδομένα, όπου τα δεδομένα μπορούν να οργανωθούν σε πίνακες με καλά καθορισμένα σχήματα. Αυτή η δομή απλοποιεί τη διαχείριση δεδομένων και την υποβολή ερωτημάτων.
- **Ασφάλεια δεδομένων:** Οι σχεσιακές βάσεις δεδομένων παρέχουν συχνά ισχυρά χαρακτηριστικά ασφαλείας, όπως έλεγχο ταυτότητας χρήστη, εξουσιοδότηση και κρυπτογράφηση, για την προστασία ευαίσθητων δεδομένων.
- **Σχέσεις δεδομένων:** Οι σχεσιακές βάσεις δεδομένων υπερέχουν στην αναπαράσταση και τη διατήρηση σχέσεων μεταξύ οντοτήτων δεδομένων. Για παράδειγμα, μπορούν να χειριστούν εύκολα σχέσεις ένα προς ένα, ένας προς πολλά και πολλά προς πολλά.
- **Ώριμο Οικοσύστημα:** Οι σχεσιακές βάσεις δεδομένων, όπως η MySQL, έχουν ένα ώριμο οικοσύστημα με εκτενή τεκμηρίωση, υποστήριξη κοινότητας και ένα ευρύ φάσμα εργαλείων και βιβλιοθηκών που διατίθενται για προγραμματιστές.

4.5 Εργαλεία Σάρωσης προς Ενσωμάτωση

Σε αυτήν την ενότητα θα αναλύσουμε τα εργαλεία που επιλέχθηκαν για τους δύο τύπους σάρωσης:

1. DAST: Arachni & Nikto
2. SAST: Insider & NodeJsScan

4.5.1 DAST

4.5.1.1 Arachni

Πώς λειτουργεί το Arachni:

Το Arachni[11] χρησιμοποιεί έναν συνδυασμό αυτοματοποιημένων και μη αυτόματων τεχνικών δοκιμών για τον εντοπισμό ευπαθειών σε εφαρμογές ιστού. Έχει σχεδιαστεί για να είναι εύκολο στη χρήση και μπορεί να ρυθμιστεί ώστε να σαρώνει συγκεκριμένες διευθύνσεις URL, τομείς ή ακόμα και ολόκληρους ιστότοπους.

Το Arachni χρησιμοποιεί μια ποικιλία τεχνικών δοκιμών για τον εντοπισμό τρωτών σημείων, όπως:

- **Έλεγχος επικύρωσης εισόδου**: Το Arachni ελέγχει τα πεδία εισόδου για κοινές επιθέσεις έγχυσης, όπως η έκχυση SQL και η δέσμη ενεργειών μεταξύ τοποθεσιών (XSS).
- **Δοκιμή διαχείρισης συνόδου**: Το Arachni δοκιμάζει τη λειτουργικότητα διαχείρισης συνόδου για να διασφαλίσει ότι τα αναγνωριστικά συνόδων διαχειρίζονται και υλοποιούνται σωστά καθώς και δεν μπορούν εύκολα να μαντευθούν ή να παραβιαστούν.
- **Δοκιμή ελέγχου ταυτότητας**: Το Arachni δοκιμάζει τη λειτουργία ελέγχου ταυτότητας για να διασφαλίσει ότι οι κωδικοί πρόσβασης προστατεύονται σωστά και ότι οι χρήστες δεν μπορούν εύκολα να παρακάμψουν τα στοιχεία ελέγχου ταυτότητας.
- **Έλεγχος εξουσιοδότησης**: Το Arachni δοκιμάζει τα στοιχεία ελέγχου εξουσιοδότησης για να διασφαλίσει ότι οι χρήστες έχουν πρόσβαση μόνο σε πόρους στους οποίους έχουν εξουσιοδότηση πρόσβασης.

Το Arachni δημιουργεί μια λεπτομερή αναφορά των τρωτών σημείων που εντοπίστηκαν, μαζί με συστάσεις για τον τρόπο επίλυσής τους. Η αναφορά περιλαμβάνει πληροφορίες, όπως η

[11] https://github.com/Arachni/arachni

σοβαρότητα της ευπάθειας, οι επηρεαζόμενες διευθύνσεις URL και βήματα για την αναπαραγωγή της ευπάθειας.

Γιατί το Arachni είναι καλό:

- **Ολοκληρωμένη σάρωση ευπάθειας**: Το Arachni είναι σε θέση να σαρώσει ένα ευρύ φάσμα εφαρμογών ιστού, συμπεριλαμβανομένων εκείνων που έχουν κατασκευαστεί με διαφορετικές γλώσσες προγραμματισμού και πλαίσια.
- **Αυτοματοποιημένη δοκιμή**: Το Arachni αυτοματοποιεί τη διαδικασία σάρωσης ευπάθειας, η οποία μπορεί να εξοικονομήσει χρόνο σε σχέση με μια χειρωνακτική επιθεώρηση/δοκιμή και προσπάθεια για προγραμματιστές και επαγγελματίες ασφαλείας.
- **Εύκολο στη χρήση**: Το Arachni έχει σχεδιαστεί για να είναι εύκολο στη χρήση, με απλό περιβάλλον χρήστη και σαφή αναφορά τρωτών σημείων.
- **Ανοιχτός κώδικας**: Το Arachni είναι ένα εργαλείο ανοιχτού κώδικα, που σημαίνει ότι είναι δωρεάν διαθέσιμο στους προγραμματιστές για χρήση και συνεισφορά.
- **Επεκτάσιμο**: Το Arachni είναι επεκτάσιμο, πράγμα που σημαίνει ότι οι προγραμματιστές μπορούν να δημιουργήσουν προσαρμοσμένα πρόσθετα και ενότητες για να επεκτείνουν τη λειτουργικότητά του και να το προσαρμόσουν στις συγκεκριμένες ανάγκες τους.

Συνολικά, το Arachni είναι ένα ισχυρό εργαλείο για τον εντοπισμό ευπαθειών εφαρμογών ιστού. Η ολοκληρωμένη σάρωση ευπάθειας, ο αυτοματισμός και η ευκολία χρήσης το καθιστούν πολύτιμο εργαλείο για προγραμματιστές και επαγγελματίες ασφαλείας που ενδιαφέρονται για την ασφάλεια των εφαρμογών Ιστού τους.

4.5.1.2 Nikto

Πώς λειτουργεί το Nikto:

Το Nikto[12] λειτουργεί στέλνοντας αιτήματα HTTP σε έναν διακομιστή ιστού και αναλύοντας τις απαντήσεις για τον εντοπισμό πιθανών τρωτών σημείων. Χρησιμοποιεί μια ποικιλία τεχνικών για τον εντοπισμό τρωτών σημείων, όπως:

- Έλεγχος για προεπιλεγμένα αρχεία και καταλόγους που χρησιμοποιούνται συνήθως από διακομιστές Ιστού. (OWASP 2023 API:8 Security Misconfiguration)
- Δοκιμή για την παρουσία γνωστών ευάλωτων εκδόσεων λογισμικού.

[12] https://github.com/sullo/nikto

- Σάρωση για ξεπερασμένα ή εσφαλμένα διαμορφωμένα στοιχεία διακομιστή web.

Έλεγχος για κοινά τρωτά σημεία έκχυσης, όπως η έκχυση SQL και το XSS.

Το Nikto δημιουργεί μια λεπτομερή αναφορά όλων των ευπαθειών που εντοπίστηκαν, συμπεριλαμβανομένων των αξιολογήσεων σοβαρότητας και των προτεινόμενων βημάτων αποκατάστασης.

Γιατί το Nikto είναι καλό:

- **Εύκολο στη χρήση**: Το Nikto διαθέτει μια φιλική προς το χρήστη διεπαφή γραμμής εντολών που το καθιστά εύκολο στη χρήση ακόμη και για μη τεχνικούς χρήστες.
- **Δωρεάν και ανοιχτού κώδικα**: Το Nikto είναι ένα δωρεάν εργαλείο ανοιχτού κώδικα, το οποίο το καθιστά προσβάσιμο σε ένα ευρύ φάσμα χρηστών και επιτρέπει τις συνεισφορές της κοινότητας.
- **Ολοκληρωμένη σάρωση**: Το Nikto χρησιμοποιεί μια ποικιλία τεχνικών σάρωσης για τον εντοπισμό πιθανών τρωτών σημείων σε διακομιστές Ιστού και εφαρμογές Ιστού.
- **Λεπτομερείς αναφορές**: Το Nikto δημιουργεί λεπτομερείς αναφορές που περιλαμβάνουν αξιολογήσεις επικινδυνότητας και προτεινόμενα βήματα αποκατάστασης για κάθε ευπάθεια που εντοπίστηκε.
- **Προσαρμόσιμο**: Το Nikto μπορεί να προσαρμοστεί για να ταιριάζει σε συγκεκριμένες ανάγκες σάρωσης, όπως εξαίρεση ορισμένων καταλόγων ή χρήση προσαρμοσμένων προσθηκών.

Συνολικά, το Nikto είναι ένα εργαλείο για τον εντοπισμό πιθανών τρωτών σημείων σε διακομιστές ιστού και εφαρμογές Ιστού. Η ευκολία χρήσης, οι ολοκληρωμένες δυνατότητες σάρωσης και οι λεπτομερείς αναφορές το καθιστούν πολύτιμη προσθήκη σε κάθε εργαλειοθήκη ασφαλείας.

4.5.2 SAST

4.5.2.1 Insider

Πώς λειτουργεί το Insider:

Πολλά έργα ανοιχτού κώδικα, όπως το Insider[13], παρέχουν εργαλεία Command Line Interface (CLI) που επιτρέπουν στους προγραμματιστές και τους χρήστες να αλληλεπιδρούν με το λογισμικό, να διαχειρίζονται διαμορφώσεις, να εκτελούν εργασίες και να έχουν πρόσβαση σε διάφορες

[13] https://github.com/insidersec/insider

λειτουργίες από τη γραμμή εντολών. Το Insider επικεντρώνεται στην κάλυψη του OWASP Top 10, ως προς την ανάλυση του πηγαίου κώδικα μιας εφαρμογής, για να εντοπίσει τρωτά σημεία. Το Insider είναι εστιασμένο σε ένα ευέλικτο και εύκολο στην εφαρμογή λογισμικό εντός του αγωγού DevOps. Ο αγωγός DevOps ή αλλιώς DevOps pipeline, αποτελείται από ένα σύνολο αυτοματοποιημένων διαδικασιών και εργαλείων που επιτρέπουν στις ομάδες ανάπτυξης λογισμικού να δημιουργούν, να δοκιμάζουν και να αναπτύσσουν εφαρμογές γρήγορα και αποτελεσματικά. Ο αγωγός αυτοματοποιεί διάφορα στάδια του κύκλου ζωής παράδοσης λογισμικού, από την ενοποίηση και τη δοκιμή κώδικα έως την ανάπτυξη και την παρακολούθηση. Ο στόχος ενός αγωγού DevOps είναι να εξορθολογίσει τη διαδικασία ανάπτυξης και ανάπτυξης, να μειώσει τις μη αυτόματες παρεμβάσεις και να βελτιώσει τη συνεργασία μεταξύ των ομάδων ανάπτυξης και λειτουργίας.

Προς το παρόν το Insider υποστηρίζει τις ακόλουθες τεχνολογίες/γλώσσες προγραμματισμού: Java (Maven και Android), Kotlin (Android), Swift (iOS), .NET Full Framework, C# και Javascript (Node.js).

Γιατί το Insider είναι καλό:

- **Είναι δωρεάν και ανοικτού κώδικα**: Το Insider είναι ένα δωρεάν εργαλείο ανοιχτού κώδικα, το οποίο το καθιστά προσβάσιμο σε ένα ευρύ φάσμα χρηστών ενώ επιτρέπει και τις συνεισφορές της κοινότητας.
- **Εύχρηστη διεπαφή γραμμής εντολών**
- **Αναλυτικές αναφορές**: Οι αναφορές του Insider περιέχουν αναλυτικά τις ευπάθειες που βρέθηκαν, το CVSS[14] σκορ της κάθε μίας καθώς και το συνολικό βαθμό ασφάλειας (security score) της εφαρμογής μετά την σάρωση.

4.5.2.2 NodeJsScan

Πώς λειτουργεί το NodeJsScan:

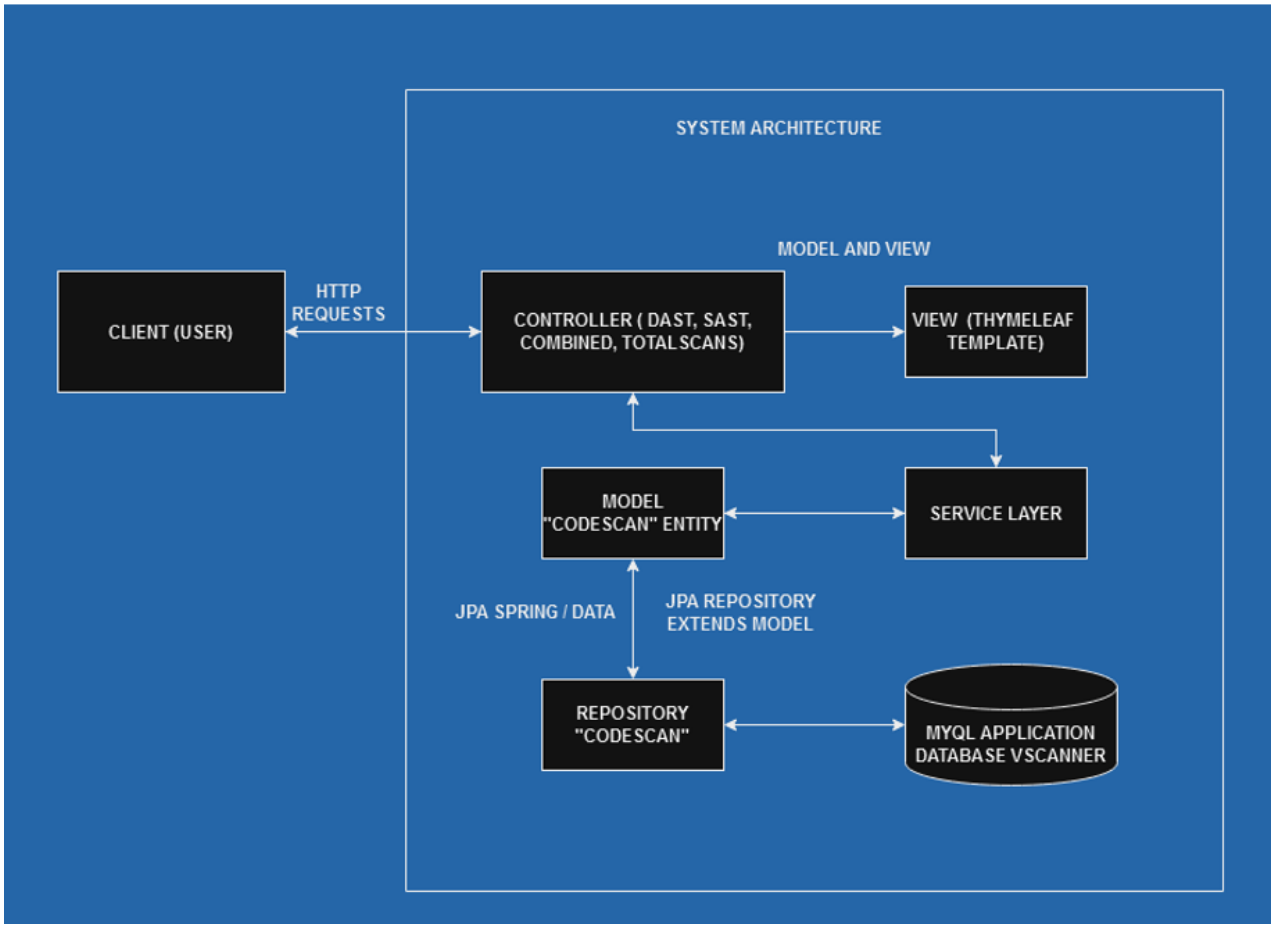
Το NodeJsScan[15] σαρώνει τον πηγαίο κώδικα μιας εφαρμογής, και επικεντρώνεται κυρίως σε εφαρμογές Node.js. Το εργαλείο αναλύει τον πηγαίο κώδικα για να κατανοήσει τη δομή του και στη συνέχεια τον σαρώνει για γνωστά μοτίβα ασφαλείας και τρωτά σημεία.

[14] https://www.first.org/cvss/
[15] https://github.com/ajinabraham/nodejsscan

Γιατί το NodeJsScan είναι καλό:

- **Σύνολο κανόνων**: Το NodeJsScan χρησιμοποιεί ένα σύνολο προκαθορισμένων κανόνων, υπογραφών και μοτίβων για τον εντοπισμό κοινών ζητημάτων ασφαλείας. Αυτοί οι κανόνες καλύπτουν πολλά ζητήματα ασφάλειας, όπως έκχυση κώδικα, XSS, και πολλά άλλα.
- **Αναλυτικές αναφορές σε πολλές μορφές**: Το NodeJsScan παρέχει αναλυτική περιγραφή των ευπαθειών που βρέθηκαν, τρόπους αντιμετώπισης, καθώς και διάφορες επιλογές στον χρήστη για την μορφοποίηση της αναφοράς (πχ. HTML, JSON).
- **Συνεχής ανάλυση**: Οι προγραμματιστές μπορούν να ενσωματώσουν το NodeJsScan στη διοχέτευση CI/CD για να εκτελέσουν αυτοματοποιημένη σάρωση ασφαλείας κατά τη διαδικασία κατασκευής, επιτρέποντας τον έγκαιρο εντοπισμό ζητημάτων ασφαλείας.

4.6 Αρχιτεκτονική Συστήματος



Εικόνα 1. Αρχιτεκτονική Συστήματος

Η αρχιτεκτονική του συστήματος φαίνεται στην παραπάνω εικόνα. Αποτελείται από τα εξής συστατικά:

(1) Ο Client (user) ουσιαστικά δεν είναι συστατικό της εφαρμογής (απλώς εμφανίζεται για να επιδείξει μια εξωτερική οντότητα του συστήματος και πως ακολουθείται μια αρχιτεκτονική πελάτη-εξυπηρετητή). Αναπαριστά τον πελάτη της εφαρμογής. Αν ο πελάτης είναι ένας χρήστης/άτομο, αυτός μπορεί να την χρησιμοποιεί μέσω ενός φυλλομετρητή (Web Browser). Αν είναι ένα άλλο πρόγραμμα ή πράκτορας, τότε θα χρησιμοποιεί το RESTful API της εφαρμογής.

(2) Ο Controller (ελεγκτής) λαμβάνει το αίτημα και το αντιστοιχίζει σε μια συγκεκριμένη μέθοδο της διεπαφής REST που εκτίθεται προς το UI (ή σε εξωτερικά προγράμματα πελάτη). Στη συνέχεια, αυτή η μέθοδος καλεί την κατάλληλη μέθοδο στο επίπεδο Service (Υπηρεσίας), περνώντας κατά μήκος τις απαραίτητες παραμέτρους. Τα mappings για τα αιτήματα του χρήστη ορίζονται στο επίπεδο του Ελεγκτή. Το επίπεδο αυτό είναι υπεύθυνο για τη λήψη εισερχόμενων αιτημάτων από το πρόγραμμα περιήγησης ιστού του χρήστη, την αντιστοίχιση τους σε συγκεκριμένα τελικά σημεία και την κλήση της κατάλληλης μεθόδου επιπέδου υπηρεσίας για τη διαχείριση του αιτήματος.

(3) Η Όψη (View) αποτελεί το UI (διεπαφή χρήσης) της εφαρμογής. Έχει υλοποιηθεί μέσω της χρήσης της βιβλιοθήκης Thymeleaf[16].

(4) Το επίπεδο Service είναι υπεύθυνο για την εφαρμογή της επιχειρηματικής λογικής της εφαρμογής.

(5) Το επίπεδο Database είναι υπεύθυνο για την πρόσβαση στη βάση δεδομένων MySQL και την εκτέλεση λειτουργιών CRUD (Δημιουργία, Ανάγνωση, Ενημέρωση, Διαγραφή) σε δεδομένα. Χρησιμοποιείται ORM οπότε δεν απαιτείται κάποιο είδος άμεσης πρόσβασης στη βάση δεδομένων.

(6) Το επίπεδο Repository ουσιαστικά επεκτείνει την διεπαφή JPA Repository. Χρησιμεύει για να πραγματοποιήσουμε τις απαιτούμενες λειτουργικότητες στην βάση μας (π.χ. δημιουργία ενός νέου αντικειμένου και αποθήκευση στην βάση). Χρησιμοποιεί δηλαδή το μοντέλο για να συνδέσει (αντιστοιχίσει) τα αντικείμενα στους πίνακες της βάσης και να εκτελέσει λειτουργίες CRUD (Create, Read, Update, Delete).

(7) Το επίπεδο Model είναι υπεύθυνο για να εκπροσωπεί τις κύριες οντότητες του συστήματος, οι οποίες λογικά λόγω της χρήσης της ORM θα συνδέονται με συγκεκριμένους πίνακες στην βάση

[16] https://www.thymeleaf.org/

δεδομένων. Το μοντέλο μας χρησιμοποιείται από το Repository για να ορίσει την δομή των δεδομένων που θέλουμε να αποθηκεύσουμε στην βάση και τον τρόπο με τον οποίο θα αντιστοιχιστεί από και προς αυτήν.

Κάθε μία από αυτές τις αντιστοιχίσεις σχετίζεται με μια συγκεκριμένη μέθοδο αιτήματος HTTP (συγκεκριμένα μεθόδους POST για τα /sast και /dast και GET μεθόδους για να λαμβάνουμε τις τελικές αναφορές) και μπορεί επίσης να περιλαμβάνει παραμέτρους εισόδου (διαφόρων μορφών – π.χ. επερώτησης) που διαβιβάζονται στο αίτημα (όπως η διεύθυνση URL για μια σάρωση DAST ή το όνομα ενός αρχείου για αναζήτηση στο /totalScans τελικό σημείο). Το πλαίσιο Spring παρέχει εργαλεία για τον καθορισμό και τη διαχείριση αυτών των αντιστοιχίσεων, διευκολύνοντας τον χειρισμό των εισερχόμενων αιτημάτων και τη δρομολόγησή τους στο κατάλληλο μέρος της εφαρμογής μας.

ΚΕΦΑΛΑΙΟ 5 – ΕΓΚΑΤΑΣΤΑΣΗ ΚΑΙ ΔΙΑΜΟΡΦΩΣΗ ΣΥΣΤΗΜΑΤΟΣ

Η εφαρμογή τρέχει σε περιβάλλον λειτουργικού συστήματος (ΛΣ) Linux και κατά προτίμηση Ubuntu. Εφόσον ο χρήστης δεν διαθέτει το εν λόγω ΛΣ, μπορεί να χρησιμοποιήσει κάποιο λογισμικό δημιουργίας και επεξεργασίας Εικονικών Μηχανημάτων (Virtual Machines). Προτείνεται το VMware Workstation 14[17]. Για να μπορέσει η εφαρμογή να εκτελεστεί επιτυχώς θα πρέπει να εγκατασταθούν τα απαραίτητα πακέτα (Packages), εξαρτήσεις (dependencies) καθώς και να γίνουν οι κατάλληλες παραμετροποιήσεις. Παρακάτω παρατίθενται αναλυτικά τα βήματα αυτά.

5.1 Βασική Διαδικασία Εγκατάστασης & Εκτέλεσης του Συστήματος

Τα βήματα παρακάτω αφορούν το λειτουργικό σύστημα Ubuntu:

Βήμα 1: Απαιτείται η εγκατάσταση Java 17 ή μεγαλύτερης έκδοσης αυστηρά για να τρέξει η εφαρμογή. Για να γίνει αυτό, ο χρήστης ανοίγει ένα τερματικό και πληκτρολογεί την εντολή:

sudo yum install java-17-openjdk (Ubuntu)

Βήμα 2: Απαιτείται η εγκατάσταση Maven για Java 17 ή μεγαλύτερη έκδοση αυστηρά για να τρέξει η εφαρμογή. Για να γίνει αυτό, ο χρήστης ανοίγει ένα τερματικό και πληκτρολογεί την εντολή:

sudo apt install maven

Για να βεβαιωθούμε ότι το Maven εκτελείται κανονικά, πληκτρολογούμε:

mvn -version

Βήμα 3: Εγκατάσταση του ΣΣΔΒ MySql. Εδώ απαιτείται μια σειρά από πολλαπλά βήματα. Στην περίπτωση του συστήματός μας, θα πρέπει να δημιουργηθεί μια βάση με όνομα “vscanner”:

sudo apt update

sudo apt install mysql-server

17 https://customerconnect.vmware.com/en/downloads/info/slug/desktop_end_user_computing/vmware_workstation_player/14_0

Κατά την εγκατάσταση, θα σας ζητηθεί να ορίσετε έναν κωδικό πρόσβασης για τον ριζικό χρήστη του MySQL ΣΣΔΒΔ. Εισαγάγετε έναν ισχυρό κωδικό πρόσβασης και θυμηθείτε τον. Αφού ολοκληρωθεί η εγκατάσταση, ξεκινάει η υπηρεσία MySQL.

Για να ελέγξετε εάν εκτελείται το ΣΣΔΒΔ MySQL, πληκτρολογήστε την ακόλουθη εντολή:

sudo systemctl status mysql

Τέλος, για να δημιουργηθεί η βάση "vscanner", ακολουθήστε τα εξής υποβήματα:

Α. Συνδεθείτε στη MySQL πληκτρολογώντας την ακόλουθη εντολή:

sudo mysql -u root -p (Δείτε Εικόνα 1).

Εισάγετε τον κωδικό πρόσβασης που ορίσατε κατά την εγκατάσταση όταν σας ζητηθεί.

Β. Δημιουργήστε τη βάση δεδομένων πληκτρολογώντας την ακόλουθη εντολή:

CREATE DATABASE vscanner;

Γ. Βεβαιωνόμαστε ότι η βάση δεδομένων έχει παραχθεί επιτυχώς με την εξής εντολή

SHOW DATABASES;

Δ. Για να βγει ο χρήστης από το τερματικό της sql εκτελεί την εντολή:

exit

```
vasilis@vasilis-virtual-machine:~$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 12
Server version: 8.0.31-0ubuntu0.22.04.1 (Ubuntu)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

Εικόνα 2. Εκκίνηση της MySQL από το Linux τερματικό

Εγκαταστήθηκε σε αυτό το σημείο με επιτυχία το ΣΣΔΒΔ της MySQL στο Ubuntu και δημιουργήσαμε μια βάση δεδομένων με το όνομα "vscanner".

Σημείωση: Συνιστάται να ασφαλίσετε την εγκατάσταση MySQL ακολουθώντας τις βέλτιστες πρακτικές για την ασφάλεια MySQL, όπως η απενεργοποίηση της απομακρυσμένης πρόσβασης στον διακομιστή MySQL και η δημιουργία ξεχωριστού λογαριασμού χρήστη για κάθε βάση δεδομένων.

Βήμα 4: Ο χρήστης κατεβάζει την εφαρμογή (πηγαίο κώδικα). https://www.dropbox.com/scl/fi/pnd8v3twbhxglqexdkw6p/VulnerabilityScannerFInalVersion.zip?rlkey=4yyoe6gxjpzeuj36zvrbjcywc&dl=0

Βήμα 5: Ως επόμενο βήμα, θα χρειαστεί να διασυνδέσουμε την εφαρμογή με την βάση που μόλις δημιουργήθηκε. Για να γίνει αυτό θα πρέπει να πάμε στο path του project: src/main/resources/application.properties

και να αλλάξουν τα εξής:

spring.datasource.username='type your username here'

spring.datasource.password='type your secret password here'

Όπου username το όνομα του χρήστη στο linux που κατέχει τα διαπιστευτήρια της βάσης, και password ο κωδικός του. Το βήμα αυτό αποτελεί ορόσημο για την λειτουργικότητα της εφαρμογής. Στην συγκεκριμένη περίπτωση, μιλάμε για βάση που δημιουργήθηκε από ριζικό (root) χρήστη. Οπότε, το username θα είναι root. Παρ'όλα αυτά, προτείνεται η δημιουργία νέου χρήστη του συστήματος βάσεων δεδομένων που θα έχει συγκεκριμένα δικαιώματα μόνο στην προκείμενη βάση της εφαρμογής (βλ. Βιβλιογραφία [7]).

Για να δημιουργήσετε έναν νέο χρήστη του ΣΔΒΔ που θα έχει δικαιώματα μόνο στην προκείμενη βάση, δείτε εδώ[18].

Βήμα 6: Εγκατάσταση Perl:

sudo apt install perl

Βήμα 7: Εγκατάσταση Python3 και NodeJsScan tool:

[18] https://www.digitalocean.com/community/tutorials/how-to-create-a-new-user-and-grant-permissions-in-mysql

Ο κώδικας εμπεριέχει προ-εγκατεστημένα τα εργαλεία και μόνο ένα από αυτά εγκαθίσταται κατά την διάρκεια της εγκατάστασης της εφαρμογής (NodeJsScan).

virtual env venv -p python3

source venv/bin/activate

pip install nodejsscan

Βήμα 8: Μετάβαση στον φάκελο της εφαρμογής (δείτε Εικόνα 3):

cd VulnerabilityScannertest

```
vasilis@vasilis-virtual-machine:~/Desktop$ chmod -R +x VulnerabilityScannertest
vasilis@vasilis-virtual-machine:~/Desktop$ cd VUlnerabilityScannertest
bash: cd: VUlnerabilityScannertest: No such file or directory
vasilis@vasilis-virtual-machine:~/Desktop$ cd VulnerabilityScannertest
vasilis@vasilis-virtual-machine:~/Desktop/VulnerabilityScannertest$ mvn spring-
boot:run
```

Εικόνα 3. Μετάβαση στην τοποθεσία του project και εκκίνηση της web εφαρμογής με Maven.

Βήμα 9: Εντολή για εκκίνηση της εφαρμογής:

mvn spring-boot:run

```
EntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence
 unit 'default'
2023-03-07T14:10:18.100+02:00  WARN 2868 --- [  restartedMain] JpaBaseConfigura
tion$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. There
fore, database queries may be performed during view rendering. Explicitly confi
gure spring.jpa.open-in-view to disable this warning
2023-03-07T14:10:18.559+02:00  INFO 2868 --- [  restartedMain] o.s.b.a.w.s.Welc
omePageHandlerMapping     : Adding welcome page template: index
2023-03-07T14:10:19.103+02:00  INFO 2868 --- [  restartedMain] o.s.b.d.a.Option
alLiveReloadServer        : LiveReload server is running on port 35729
2023-03-07T14:10:19.154+02:00  INFO 2868 --- [  restartedMain] o.s.b.w.embedded
.tomcat.TomcatWebServer   : Tomcat started on port(s): 8080 (http) with context
path ''
2023-03-07T14:10:19.174+02:00  INFO 2868 --- [  restartedMain] c.e.v.Vulnerabil
ityScannerApplication     : Started VulnerabilityScannerApplication in 7.885 sec
onds (process running for 8.663)
```

Εικόνα 4. Έξοδος από την εκτέλεση της εντολής mvn spring-boot:run

Μετά την τελευταία εντολή, λογικά η εφαρμογή θα έχει πια εκκινήσει επιτυχώς (δείτε Εικόνα 4) και μπορούμε να την ανοίξουμε σε ένα φυλλομετρητή στο URL: http://localhost:8080.

Η εντολή "mvn spring-boot:run" χρησιμοποιείται για την εκτέλεση μιας εφαρμογής Spring Boot από τη γραμμή εντολών χρησιμοποιώντας το Maven.

Το Maven είναι ένα εργαλείο κατασκευής που χρησιμοποιείται για τη διαχείριση έργων Java και εξαρτήσεων. Όταν εκτελείτε την εντολή "mvn spring-boot:run", το Maven μεταγλωττίζει την εφαρμογή και ξεκινά έναν web servlet/container (προεπιλεγμένα τον Tomcat) που εκτελεί την RESTful εφαρμογή τύπου Spring Boot. Η εφαρμογή Spring Boot ξεκινά με προεπιλεγμένες διαμορφώσεις (όπως την χρήση του Tomcat). Η εντολή "spring-boot:run" είναι μια προσθήκη Maven που παρέχεται από το πλαίσιο Spring Boot. Αυτή η εντολή μπορεί να χρησιμοποιηθεί για την εκτέλεση της εφαρμογής κατά την ανάπτυξη.

Από προεπιλογή, η εντολή "mvn spring-boot:run" αναζητά την κύρια κλάση της εφαρμογής η οποία εντοπίζεται από το `@SpringBootApplication`, η οποία είναι η `VulnerabilityScannerApplication` στον πηγαίο κώδικα του έργου. Συγκεκριμένα, καλείται η μέθοδος SpringApplication.run από την κύρια κλάση που προαναφέρθηκε παραπάνω με παράμετρο το όνομα της κύριας αυτής κλάσης (της εφαρμογής):

```
SpringApplication.run(VulnerabilityScannerApplication.class, args);
```

Η SpringApplication.run() είναι μια θεμελιώδης μέθοδος που χρησιμοποιείται για την εκκίνηση μιας εφαρμογής Spring Boot. Ρυθμίζει το περιβάλλον της εφαρμογής, αρχικοποιεί στοιχεία, χειρίζεται διαμορφώσεις και ξεκινά τον κύκλο ζωής της εφαρμογής, συμπεριλαμβανομένου οποιουδήποτε ενσωματωμένου διακομιστή ιστού (web server), εάν υπάρχει. Η συγκεκριμένη κλάση (VulnerabilityScannerApplication.class) που παρέχεται ως όρισμα είναι το κύριο σημείο εισόδου της εφαρμογής Spring Boot, η οποία περιέχει την κύρια μέθοδο.

Εάν βρεθεί η κύρια κλάση με τον στολισμό (annotation) `@SpringBootApplication`, ή εναλλακτικά αν έχει δηλωθεί στο pom.xml, για παράδειγμα:

```
<configuration>
    <mainClass>com.example.yourMainClassHere</mainClass>
</configuration>
```

το Maven μεταγλωττίζει την εφαρμογή και την εκκινεί. Εάν η κύρια κλάση δεν βρεθεί, το Maven θα αναφέρει ένα σφάλμα.

Συνολικά, το "mvn spring-boot:run" είναι ένας βολικός τρόπος για γρήγορη εκτέλεση μιας εφαρμογής Spring Boot από τη γραμμή εντολών χρησιμοποιώντας το Maven. Αυτή η εντολή είναι χρήσιμη κατά την ανάπτυξη και τη δοκιμή, επιτρέποντας στους προγραμματιστές να δοκιμάσουν

γρήγορα τις αλλαγές του κώδικα χωρίς να χρειάζεται να δημιουργήσουν και να αναπτύξουν την εφαρμογή ξεχωριστά, εφόσον η εφαρμογή μπορεί να εκκινηθεί πολύ γρήγορα με μια εντολή.

5.2 Εναλλακτικές Διαδικασίες Εγκατάστασης

Πριν περάσουμε στις εναλλακτικές διαδικασίες εγκατάστασης, θα πρέπει να αναλυθούν μερικές βασικές έννοιες.

Δοχεία (Containers): Είναι ελαφριά πακέτα του κώδικα μιας εφαρμογής μαζί με εξαρτήσεις (dependencies), όπως συγκεκριμένες εκδόσεις χρόνου εκτέλεσης της γλώσσας προγραμματισμού και βιβλιοθήκες που απαιτούνται για την εκτέλεση των υπηρεσιών λογισμικού. Τα πακέτα αυτά μπορούν να εγκατασταθούν σε οποιοδήποτε περιβάλλον, αρκεί να εκτελείται σε αυτό μια μηχανή δοχείων, όπως είναι το Docker[19].

Εικονική Μηχανή (VM): Είναι ένας υπολογιστικός πόρος βασισμένος σε λογισμικό που προσομοιώνει ένα φυσικό υπολογιστή για την εκτέλεση προγραμμάτων και την ανάπτυξη εφαρμογών. Μία ή περισσότερες εικονικές μηχανές "επισκέπτης" λειτουργούν σε μια φυσική μηχανή "κεντρικού υπολογιστή". Κάθε εικονική μηχανή εκτελεί το δικό της λειτουργικό σύστημα και λειτουργεί ξεχωριστά από τις άλλες, ακόμα και όταν όλες εκτελούνται στον ίδιο κεντρικό υπολογιστή. Αυτό σημαίνει ότι, για παράδειγμα, μια εικονική εικονική μηχανή MacOS και μια άλλη Windows μπορεί να εκτελεστεί σε φυσικό υπολογιστή (που πχ. εκτελεί Ubuntu).

Εν γένει, συνιστάται η χρήση των **Δοχείων** έναντι των **Εικονικών Μηχανών** για τους εξής λόγους:

1. Ένα δοχείο μπορεί να έχει μέγεθος μόνο **μερικές δεκάδες megabyte**, ενώ μια εικονική μηχανή με το δικό της πλήρες λειτουργικό σύστημα μπορεί να έχει μέγεθος **αρκετά gigabyte**. Για το λόγο αυτό, ένας διακομιστής μπορεί να φιλοξενήσει περισσότερα δοχεία από εικονικές μηχανές.
2. Ένα άλλο μεγάλο πλεονέκτημα είναι ότι οι εικονικές μηχανές μπορεί να χρειαστούν **αρκετά λεπτά** για να εκκινήσουν και να αρχίσουν να εκτελούν τις εφαρμογές που φιλοξενούν, ενώ οι εφαρμογές με δοχεία μπορούν να ξεκινήσουν **σχεδόν αμέσως**. Αυτό σημαίνει ότι τα δοχεία μπορούν να δημιουργηθούν "ακριβώς έγκαιρα" όταν χρειάζεται και μπορούν να

[19] https://www.docker.com

εξαφανιστούν όταν δεν χρειάζονται πλέον, ελευθερώνοντας πόρους στους κεντρικούς υπολογιστές τους.

3. Το τρίτο πλεονέκτημα είναι ότι η δοχειοποίηση μιας εφαρμογής επιτρέπει καλύτερο διαχειριστικό έλεγχο καθώς και μια πιο εστιασμένη και αποδοτική κλιμάκωση μιας εφαρμογής. Αντί να εκτελείται μια ολόκληρη σύνθετη εφαρμογή σε ένα μόνο δοχείο, η εφαρμογή μπορεί να **χωριστεί σε ενότητες** (όπως βάση δεδομένων, διεπαφή χρήστη εφαρμογής κ.λπ.). Αυτό ονομάζεται προσέγγιση μικροϋπηρεσιών (micro-services). Οι εφαρμογές που δημιουργούνται με αυτόν τον τρόπο είναι πιο εύκολο να διαχειριστούν, επειδή κάθε λειτουργική μονάδα είναι σχετικά απλή ενώ μπορούν να γίνουν αλλαγές σε λειτουργικές μονάδες χωρίς να χρειάζεται να ξαναδημιουργηθεί ολόκληρη η εφαρμογή. Επειδή τα δοχεία είναι τόσο ελαφριά, μεμονωμένες μονάδες (ή μικροϋπηρεσίες) μπορούν να δημιουργηθούν και να κλιμακωθούν μόνο όταν χρειάζεται και είναι διαθέσιμες σχεδόν αμέσως.

5.2.1 Εναλλακτική Διαδικασία Εγκατάστασης με Εγχώρια Υποστήριξη GraalVM

Αυτό το έργο έχει ρυθμιστεί ώστε να σας επιτρέπει να δημιουργείτε είτε ένα Lightweight δοχείο είτε ένα native εκτελέσιμο αρχείο. Είναι, επίσης, δυνατό να εκτελέσουμε τις δοκιμές μας σε ένα native-image (εγχώρια εικόνα).

Εάν είστε ήδη εξοικειωμένοι με την υποστήριξη εικόνων δοχείων Spring Boot, αυτός είναι ο ευκολότερος τρόπος για να ξεκινήσετε.

Το Docker θα πρέπει να εγκατασταθεί και να ρυθμιστεί στον υπολογιστή σας πριν από τη δημιουργία της εικόνας. Για την λίστα των εντολών που απαιτούνται σε περιβάλλον Ubuntu, δείτε εδώ[20] και για Windows εδώ[21].

Για να δημιουργήσετε την εικόνα, εκτελέστε το εξής βήμα:

```
$ ./mvnw spring-boot:build-image -Pnative
```

Στη συνέχεια, μπορείτε να εκτελέσετε την εφαρμογή όπως οποιοδήποτε άλλο δοχείο:

```
$ docker run --rm -p 8080:8080 VulnerabilityScannertest:0.0.1-SNAPSHOT
```

[20] https://docs.docker.com/engine/install/ubuntu/
[21] https://docs.docker.com/desktop/install/windows-install/

5.2.2 Εναλλακτική Διαδικασία Εγκατάστασης με Εκτελέσιμο Κατασκευασμένο από Εγχώρια Εργαλεία Κατασκευής

Χρησιμοποιήστε αυτήν την επιλογή εάν θέλετε να εξερευνήσετε περισσότερες επιλογές, όπως η εκτέλεση των δοκιμών σας σε μια εγχώρια (native) εικόνα.

Ο μεταγλωττιστής «native-image» GraalVM θα πρέπει να εγκατασταθεί και να ρυθμιστεί στον υπολογιστή σας. https://www.graalvm.org/22.0/reference-manual/native-image/

Σημείωση: Απαιτείται GraalVM 22.3+.

Για να τρέξει το executable:

$ ./mvnw native:compile -Pnative

Ύστερα η εφαρμογή μπορεί να τρέξει ως εξής:

$ target/VulnerabilityScannertest

Μπορείτε, επίσης, να εκτελέσετε την υπάρχουσα συλλογή δοκιμών σε μια εγγενή εικόνα (native image - εκτελέσιμο αρχείο που είναι προσαρμοσμένο για μια συγκεκριμένη πλατφόρμα υλικού και λειτουργικό σύστημα).

Αυτός είναι ένας αποτελεσματικός τρόπος για να επικυρώσουμε την λειτουργικότητα και απόδοση της εφαρμογής.

Για να εκτελέσουμε τις υπάρχουσες δοκιμές μας σε μια native εικόνα, εκτελούμε:

$ ./mvnw test -PnativeTest

ΚΕΦΑΛΑΙΟ 6 – ΕΠΙΔΕΙΞΗ ΚΑΙ ΑΠΟΤΙΜΗΣΗ ΕΦΑΡΜΟΓΗΣ

Στόχος αυτού του κεφαλαίου είναι η επίδειξη της λειτουργικότητας της προτεινόμενης εφαρμογής, και η αποτίμησή της με βάση μια εσκεμμένα τρωτή εφαρμογή ιστού[22], η οποία περιέχει τις ακόλουθες 12 ευπάθειες:

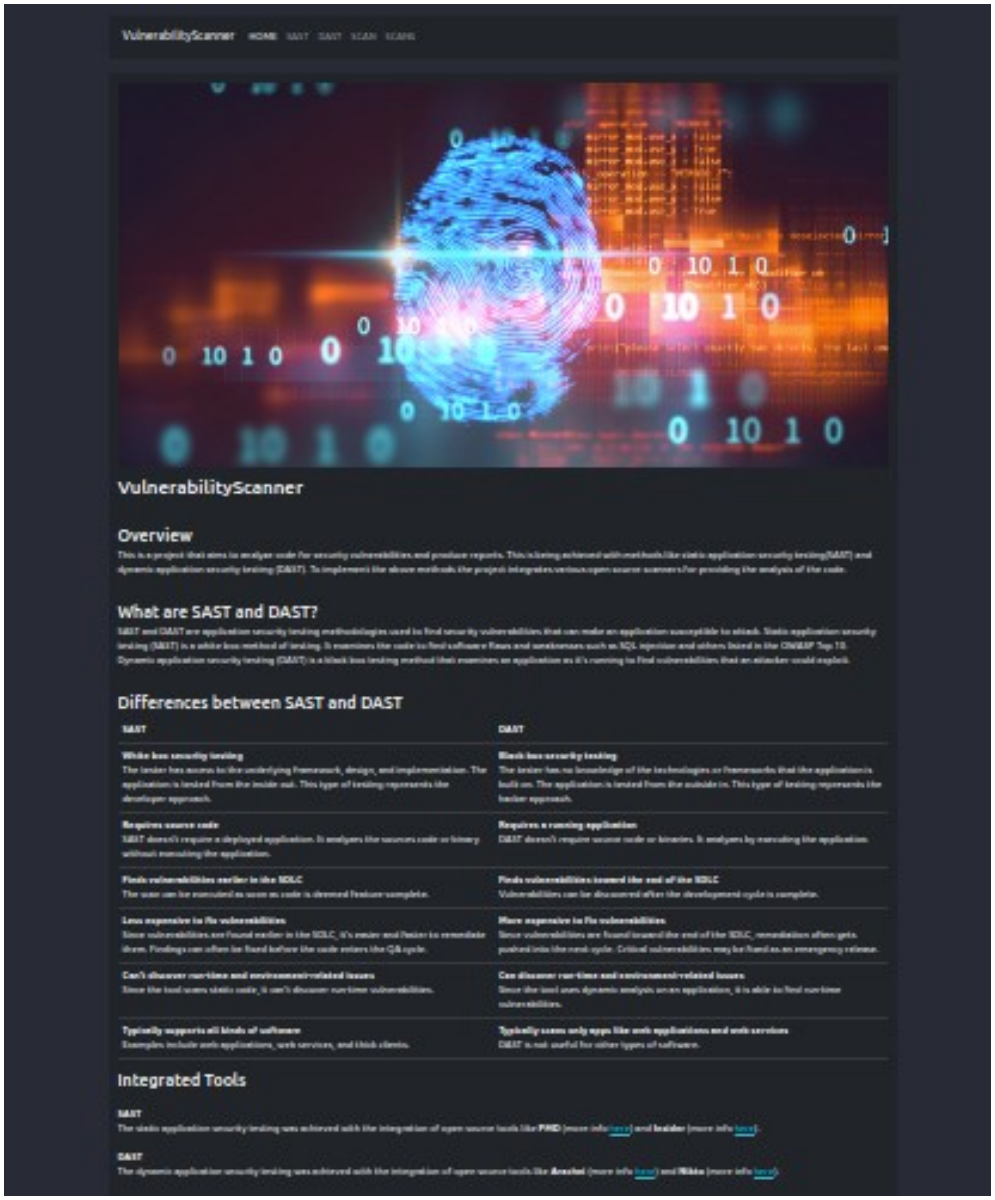
- Command Injection
- SQL Injection
- Brute Force
- Insecure Captcha
- File Inclusion
- CRF
- File Upload
- Weak Session IDs
- XSS (DOM)
- XSS (Reflected)
- XSS (Stored)
- CSP Bypadd

Συγκεκριμένα, θα αναλυθούν:

- Η συνδυασμένη εκτέλεση των δυναμικών σαρωτών.
- Η συνδυασμένη εκτέλεση των στατικών σαρωτών.
- Η συνδυασμένη εκτέλεση και των 2 τεχνικών σάρωσης.
- Η αποτίμηση της εφαρμογής επί της τρωτής εφαρμογής.

Στην επόμενη εικόνα, παρέχουμε την βασική ιστοσελίδα της εφαρμογής μας, η οποία αποτελεί τη βάση για όλα τα σενάρια χρήσης που θα επιδειχθούν. Εμφανίζεται στον χρήστη όταν τρέξει η εφαρμογή. Παρέχει μερικές πληροφορίες για την δυναμική και στατική ανάλυση που υποστηρίζει καθώς και ένα μενού.

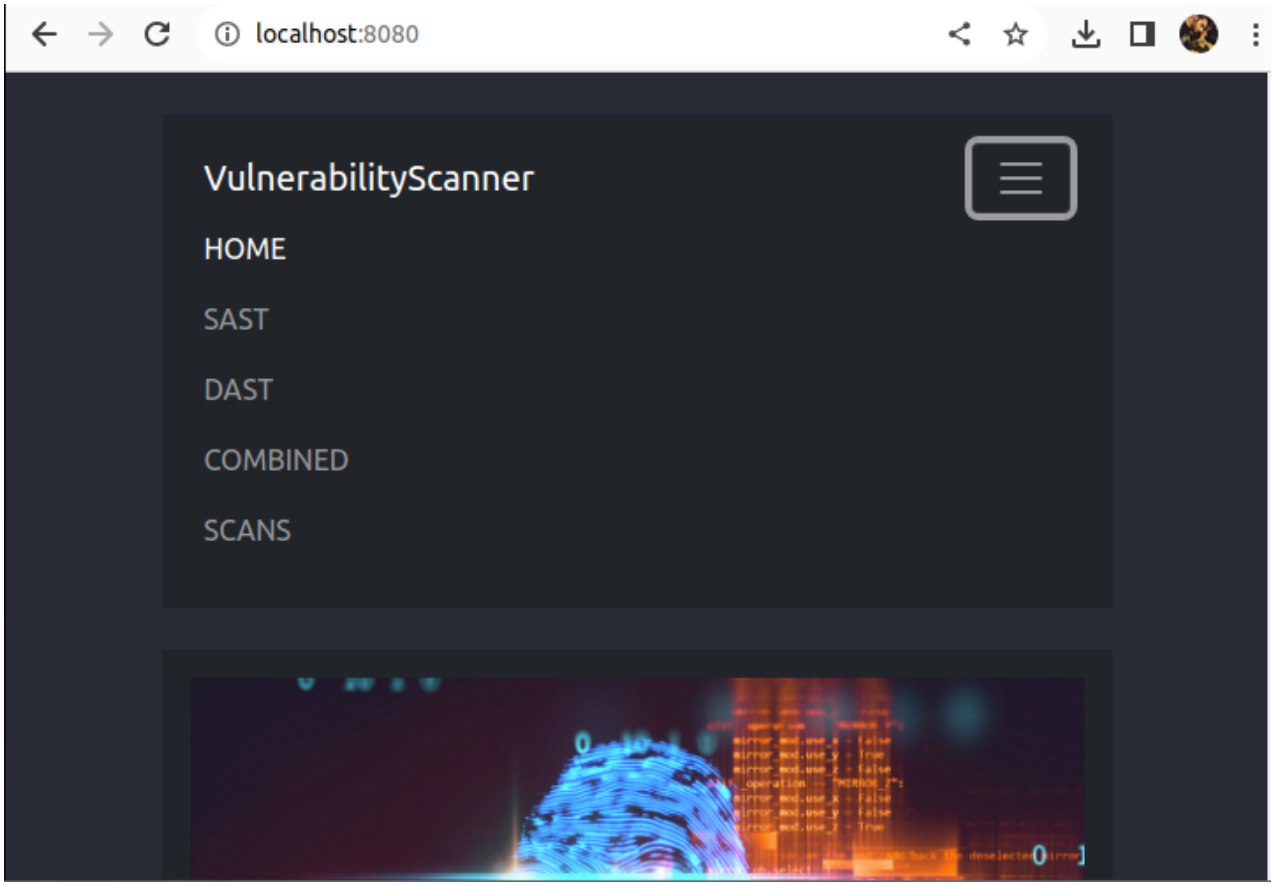
[22] https://github.com/digininja/DVWA



Εικόνα 5. Η κύρια ιστοσελίδα (homepage) της εφαρμογής.

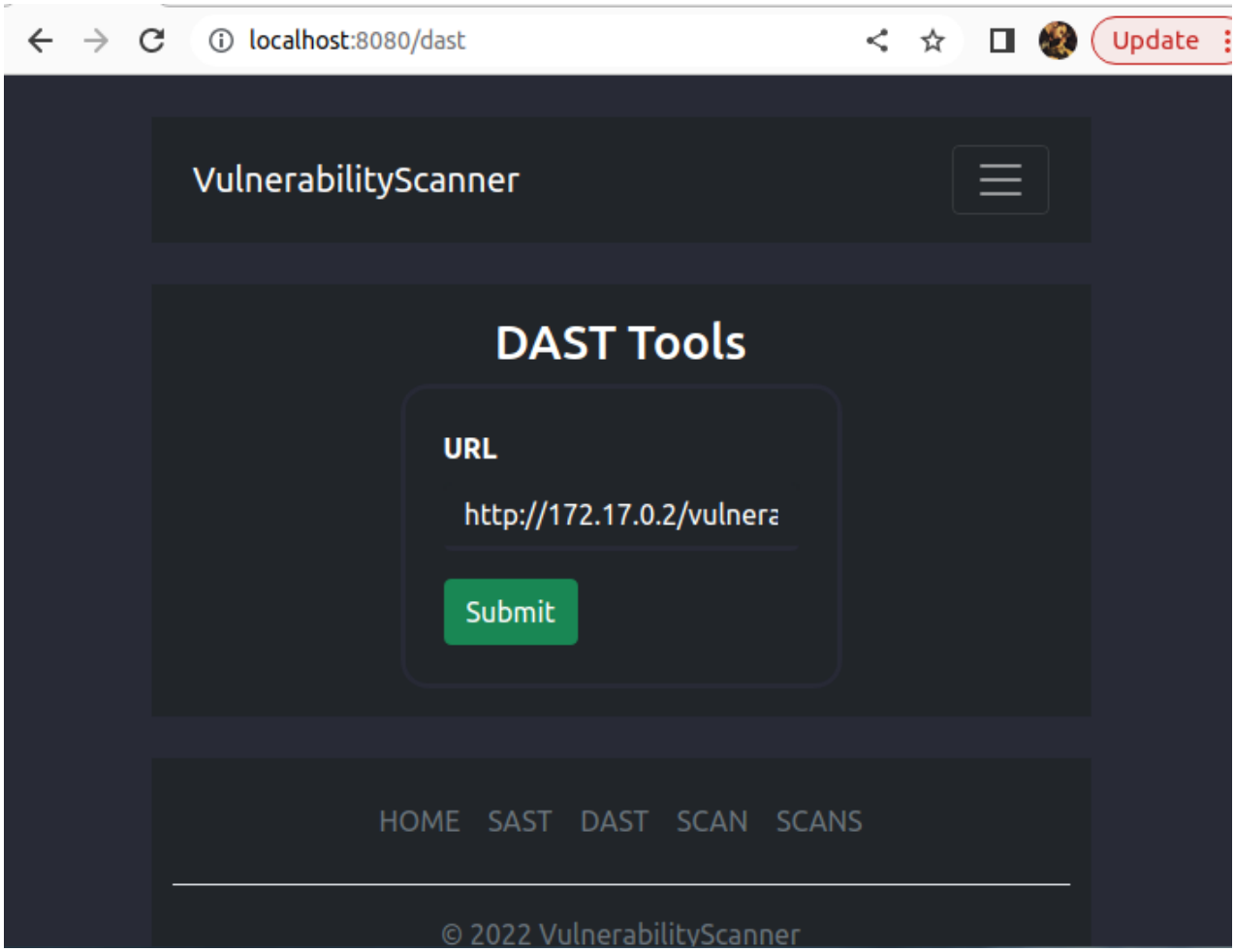
6.1 Χρήση Λειτουργίας Δυναμικής Σάρωσης (DAST)

Σε αυτήν την ενότητα, επιδεικνύουμε τη συνδυασμένη εκτέλεση δυναμικής σάρωσης (που υλοποιείται με βάση τα δύο εργαλεία δυναμικής σάρωσης που έχουμε ενσωματώσει). Ο χρήστης αρχικά ανοίγει το burger μενού της εφαρμογής πάνω δεξιά και πατάει την επιλογή ‘DAST’.



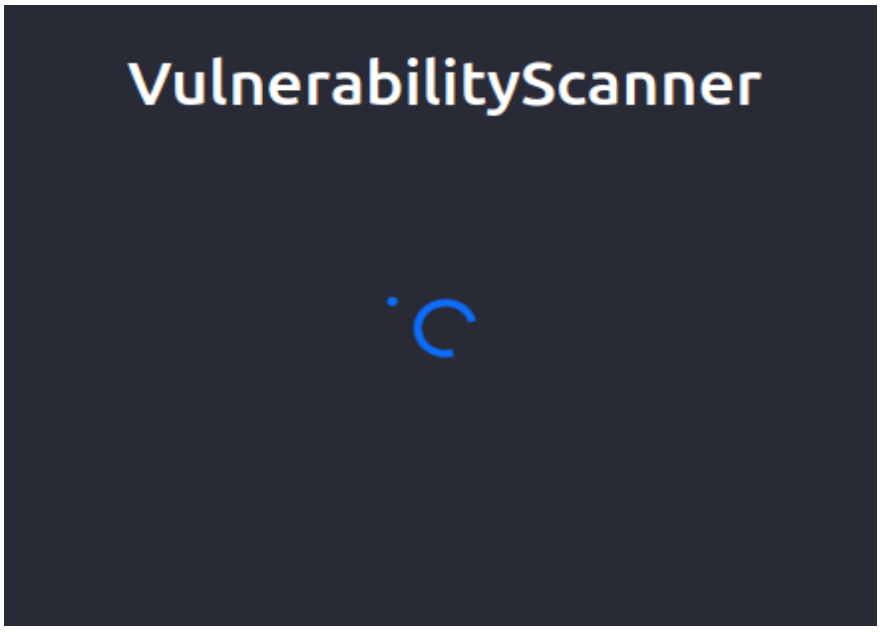
Εικόνα 6. Το burger μενού της εφαρμογής

Έπειτα, του εμφανίζεται η ιστοσελίδα ρύθμισης της σάρωσης (δείτε Εικόνα 7) όπου ο χρήστης θα πρέπει πρώτα να δώσει ως είσοδο το URL της εφαρμογής ιστού ή ιστοτόπου για την δυναμική σάρωση, χρησιμοποιώντας τον συνδυασμό των εργαλείων σάρωσης Arachni και Nikto, και έπειτα να πατήσει το κουμπί “Submit” ώστε να ξεκινήσει η συνδυασμένη εκτέλεση των δύο αυτών εργαλείων δυναμικής ανάλυσης.



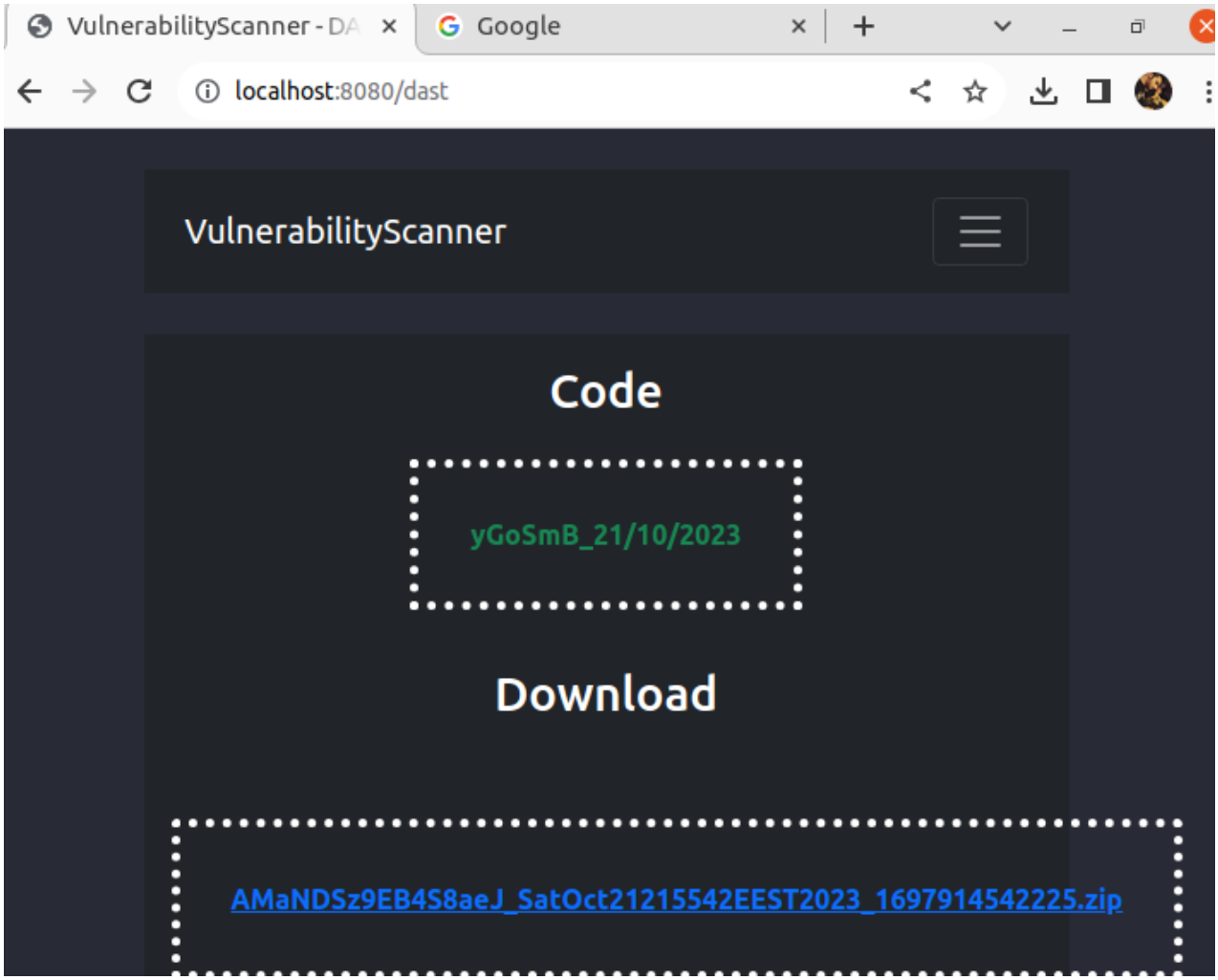
Εικόνα 7. Εισαγωγή του URL της εφαρμογής ιστού / ιστοτόπου από τον χρήστη.

Μόλις ολοκληρωθούν οι ενέργειες του χρήστη, η δυναμική σάρωση θα εκκινήσει (δείτε Εικόνα 8).

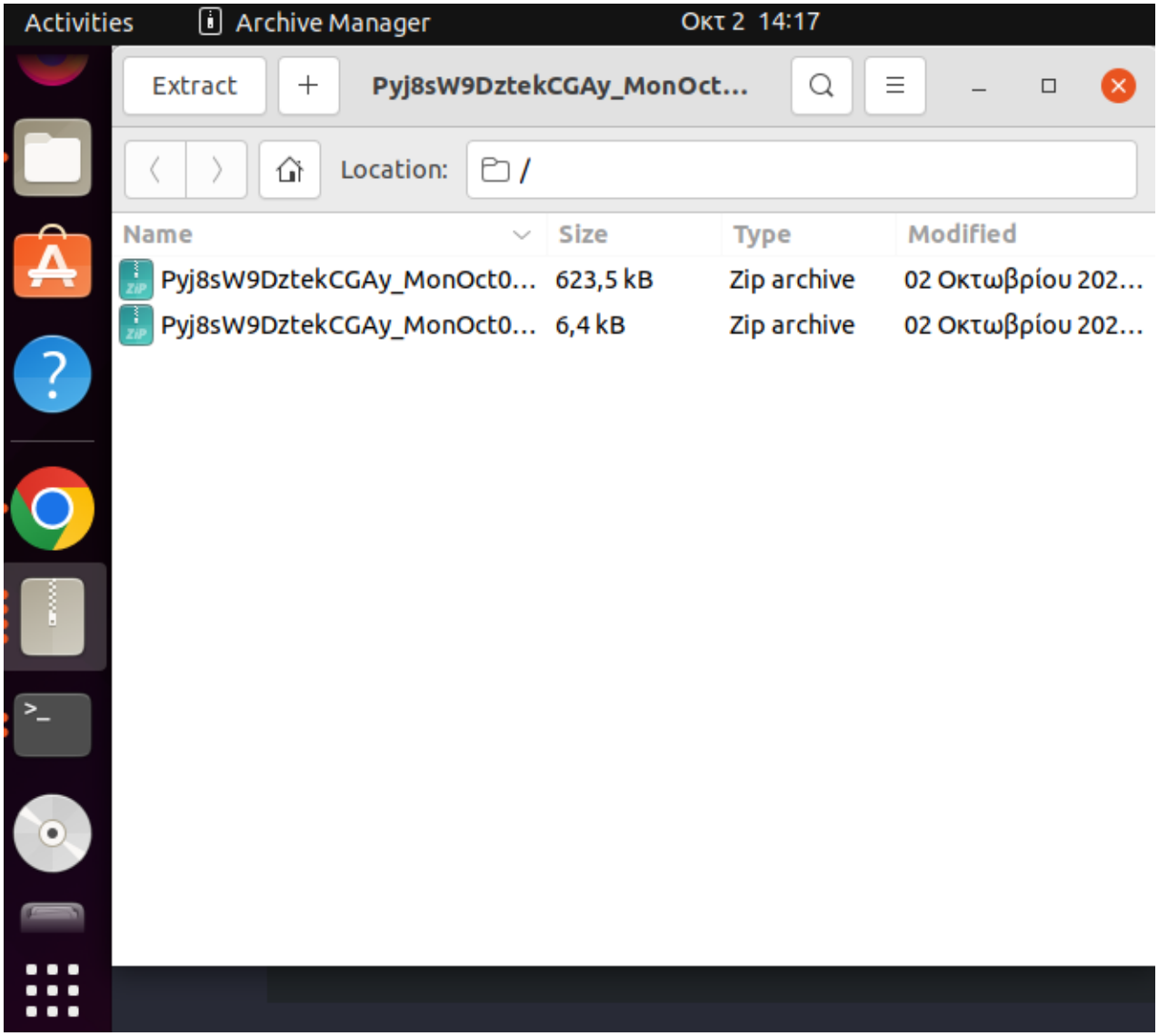


Εικόνα 8. Παράλληλη εκκίνηση των 2 DAST σαρωτών

Σε αυτό το σημείο πραγματοποιείται η εκτέλεση των 2 εργαλείων δυναμικής ανάλυσης και ο χρήστης περιμένει να τελειώσει η σάρωση. Όταν αυτή ολοκληρωθεί, τότε η ίδια/τρέχουσα σελίδα θα παρέχει στον χρήστη το παραγόμενο αρχείο που περιέχει τα αποτελέσματα της σάρωσης προς εκφόρτωση, με την μορφή ενός συνδέσμου (link) (δείτε Εικόνα 9).

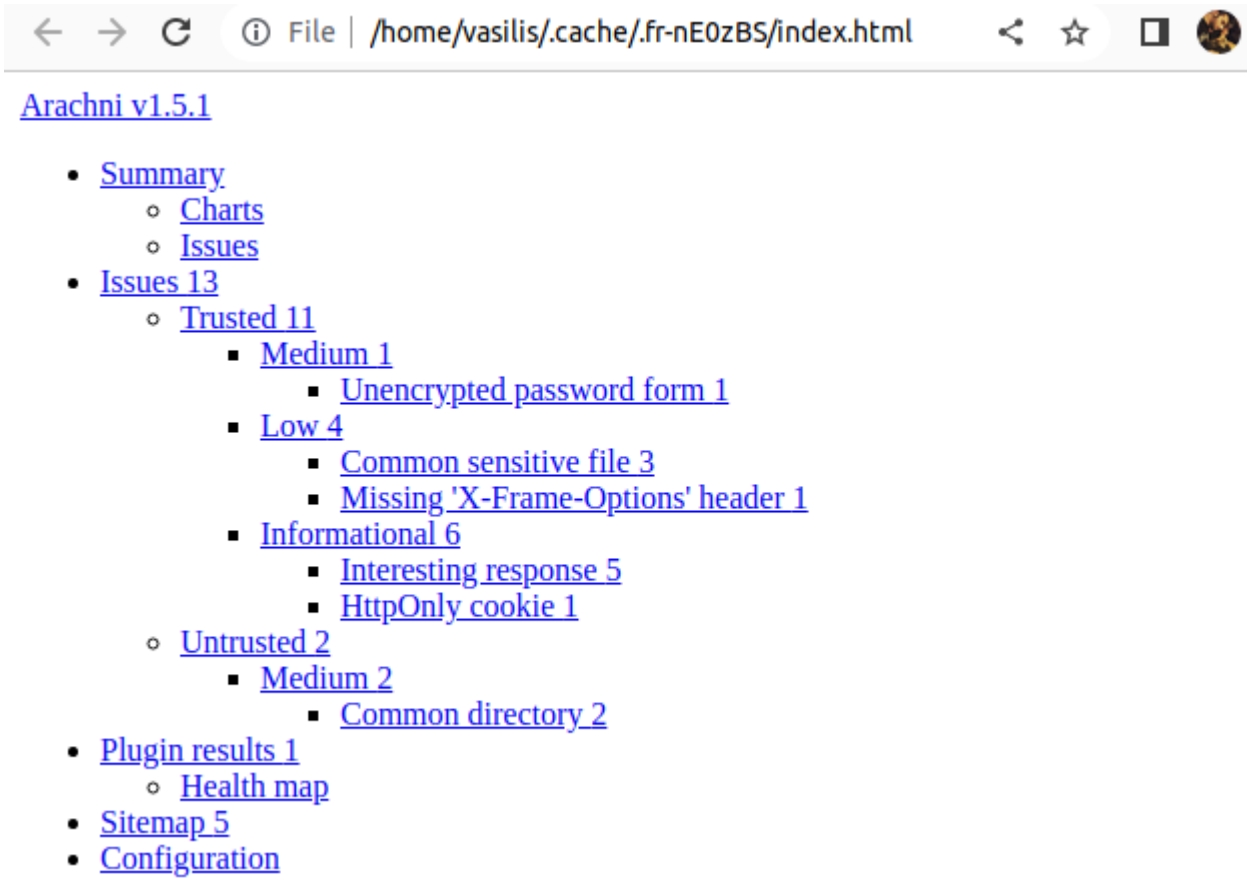


Εικόνα 9. Παροχή συνδέσμου για την εκφόρτωση των αποτελεσμάτων της σάρωσης



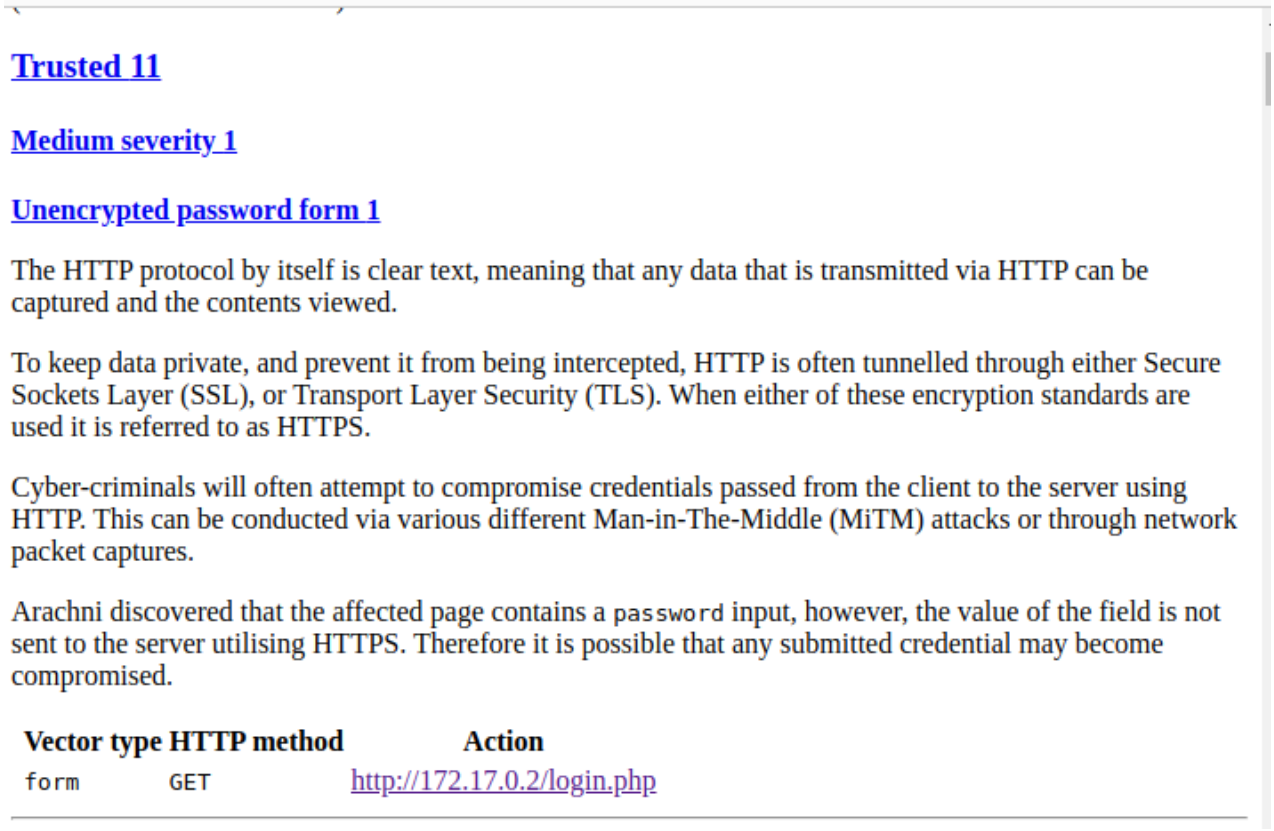
Εικόνα 10. Οι αναφορές των εργαλείων Arachni και Nikto

Ο χρήστης εφόσον πατήσει το link της αναφοράς σάρωσης, θα εκφορτώσει ένα zip (δηλ. συμπιεσμένο) αρχείο που περιέχει τις αναφορές των εργαλείων (δείτε το περιεχόμενο του αρχείου στην Εικόνα 10).



Εικόνα 11. Έξοδος του αποτελέσματος από το εργαλείο Arachni (δείγμα)

Η μορφή του αρχείου αναφοράς του Arachni είναι σε μορφή .html (δείτε Εικόνα 11). Κάτω από τα issues (ζητήματα) (και από δίπλα ο αριθμός τους συνολικά, εν προκειμένω 13), βλέπουμε το σύνολο των ευπαθειών που βρέθηκαν από το εργαλείο χωρισμένες σε κατηγορίες "Trusted" (Εμπιστευτικές), και "Untrusted" (Μη Εμπιστευτικές). Στην κατηγορία "Trusted" περιλαμβάνονται οι ευπάθειες που θεωρούνται αληθώς θετικές (true positive), ενώ στην κατηγορία των "Untrusted" περιλαμβάνονται πιθανές ψευδώς θετικές (false positive). Κάτω από αυτές της κατηγορίες παρουσιάζονται οι υποκατηγορίες τους με βάση τον βαθμό επικινδυνότητας (Medium, Low, Informational) (Μέσος, Χαμηλός, Πληροφοριακός) και από δίπλα ο αριθμός των τρωτοτήτων που εντάσσονται σε αυτές. Έπειτα, παρέχεται η περιγραφή των αντίστοιχων τρωτοτήτων (δείτε Εικόνες 12-14).



Trusted 11

Medium severity 1

Unencrypted password form 1

The HTTP protocol by itself is clear text, meaning that any data that is transmitted via HTTP can be captured and the contents viewed.

To keep data private, and prevent it from being intercepted, HTTP is often tunnelled through either Secure Sockets Layer (SSL), or Transport Layer Security (TLS). When either of these encryption standards are used it is referred to as HTTPS.

Cyber-criminals will often attempt to compromise credentials passed from the client to the server using HTTP. This can be conducted via various different Man-in-The-Middle (MiTM) attacks or through network packet captures.

Arachni discovered that the affected page contains a `password` input, however, the value of the field is not sent to the server utilising HTTPS. Therefore it is possible that any submitted credential may become compromised.

Vector type	HTTP method	Action
`form`	`GET`	http://172.17.0.2/login.php

Εικόνα 12. Ευπάθεια που βρέθηκε από το Arachni

Low severity 4

Common sensitive file 3

Web applications are often made up of multiple files and directories.

It is possible that over time some files may become unreferenced (unused) by the web application and forgotten about by the administrator/developer. Because web applications are built using common frameworks, they contain common files that can be discovered (independent of server).

During the initial recon stages of an attack, cyber-criminals will attempt to locate unreferenced files in the hope that the file will assist in further compromise of the web application. To achieve this they will make thousands of requests using word lists containing common filenames. The response headers from the server will then indicate if the file exists.

Arachni also contains a list of common file names which it will attempt to access.

Vector type	HTTP method	Action
server	GET	http://172.17.0.2/php.ini
server	GET	http://172.17.0.2/login.php
server	GET	http://172.17.0.2/robots.txt

Missing 'X-Frame-Options' header 1

Clickjacking (User Interface redress attack, UI redress attack, UI redressing) is a malicious technique of tricking a Web user into clicking on something different from what the user perceives they are clicking on, thus potentially revealing confidential information or taking control of their computer while clicking on seemingly innocuous web pages.

Εικόνα 13. Ευπάθειες που βρέθηκαν από το Arachni

HttpOnly cookie 1

HTTP by itself is a stateless protocol. Therefore the server is unable to determine which requests are performed by which client, and which clients are authenticated or unauthenticated.

The use of HTTP cookies within the headers, allows a web server to identify each individual client and can therefore determine which clients hold valid authentication, from those that do not. These are known as session cookies.

When a cookie is set by the server (sent the header of an HTTP response) there are several flags that can be set to configure the properties of the cookie and how it is to be handled by the browser.

The `HttpOnly` flag assists in the prevention of client side-scripts (such as JavaScript) accessing and using the cookie.

This can help prevent XSS attacks targeting the cookies holding the client's session token (setting the `HttpOnly` flag does not prevent, nor safeguard against XSS vulnerabilities themselves).

Vector type	HTTP method	Action
cookie	GET	http://172.17.0.2/vulnerabilities/fi/?page=file1.php

Εικόνα 14. Ευπάθεια που βρέθηκε από το Arachni

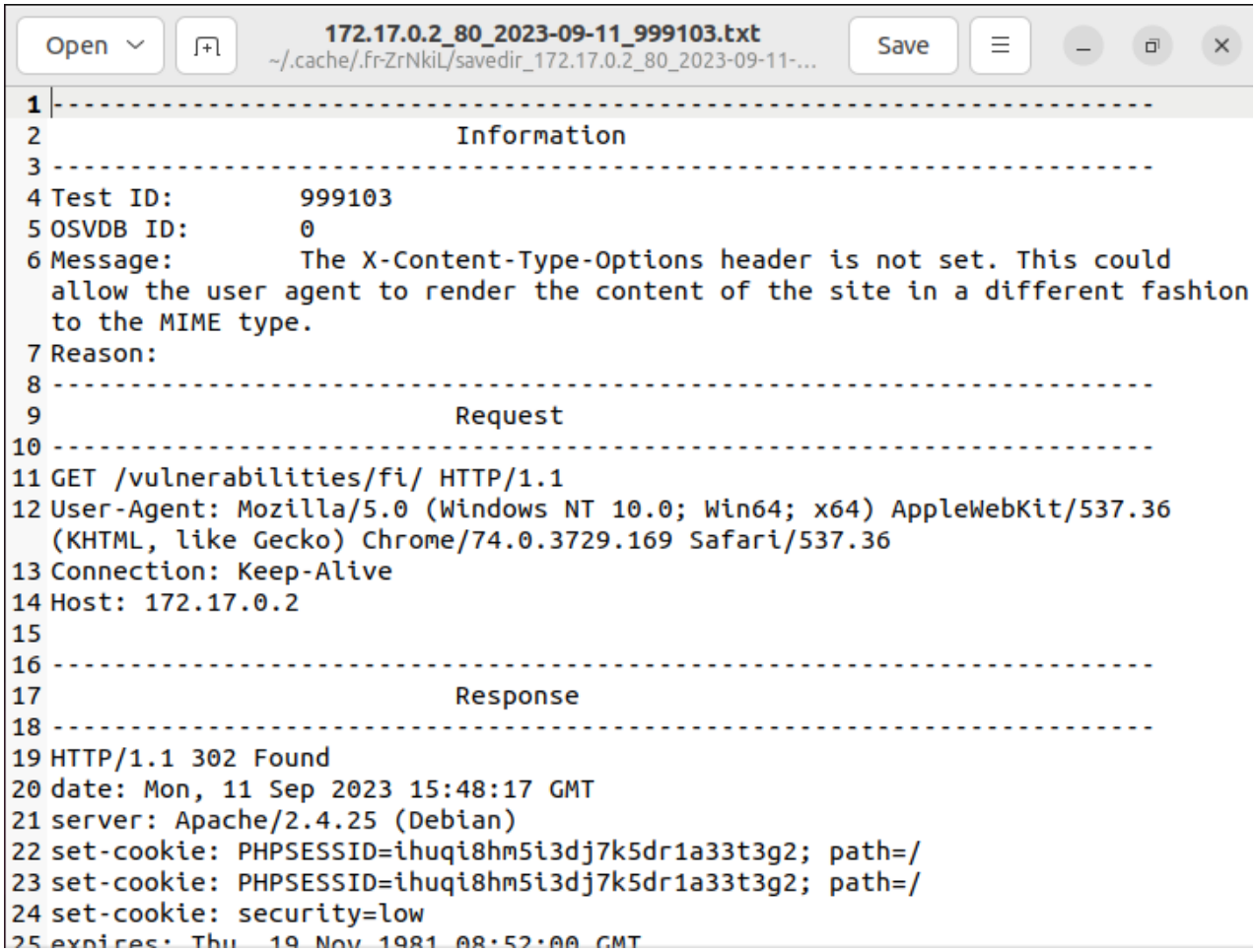
Το περίγραμμα της αναφοράς του Nikto περιλαμβάνει πολλαπλά αρχεία .html, κάθε ένα από το οποία περιλαμβάνει την ευπάθεια που βρέθηκε (δείτε Εικόνα 15).

```
1 ---------------------------------------------------------------------------
2                            Information
3 ---------------------------------------------------------------------------
4 Test ID:         999957
5 OSVDB ID:        0
6 Message:         The anti-clickjacking X-Frame-Options header is not present.
7 Reason:
8 ---------------------------------------------------------------------------
9                            Request
10 ---------------------------------------------------------------------------
11 GET /vulnerabilities/xss_s/ HTTP/1.1
12 Host: 172.17.0.3
13 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
   (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36
14 Connection: Keep-Alive
15
16 ---------------------------------------------------------------------------
17                           Response
18 ---------------------------------------------------------------------------
19 HTTP/1.1 302 Found
20 date: Mon, 25 Sep 2023 13:55:50 GMT
21 server: Apache/2.4.25 (Debian)
22 set-cookie: PHPSESSID=khpe70nfpmk26dmrcf37428r21; path=/
23 set-cookie: PHPSESSID=khpe70nfpmk26dmrcf37428r21; path=/
24 set-cookie: security=low
```

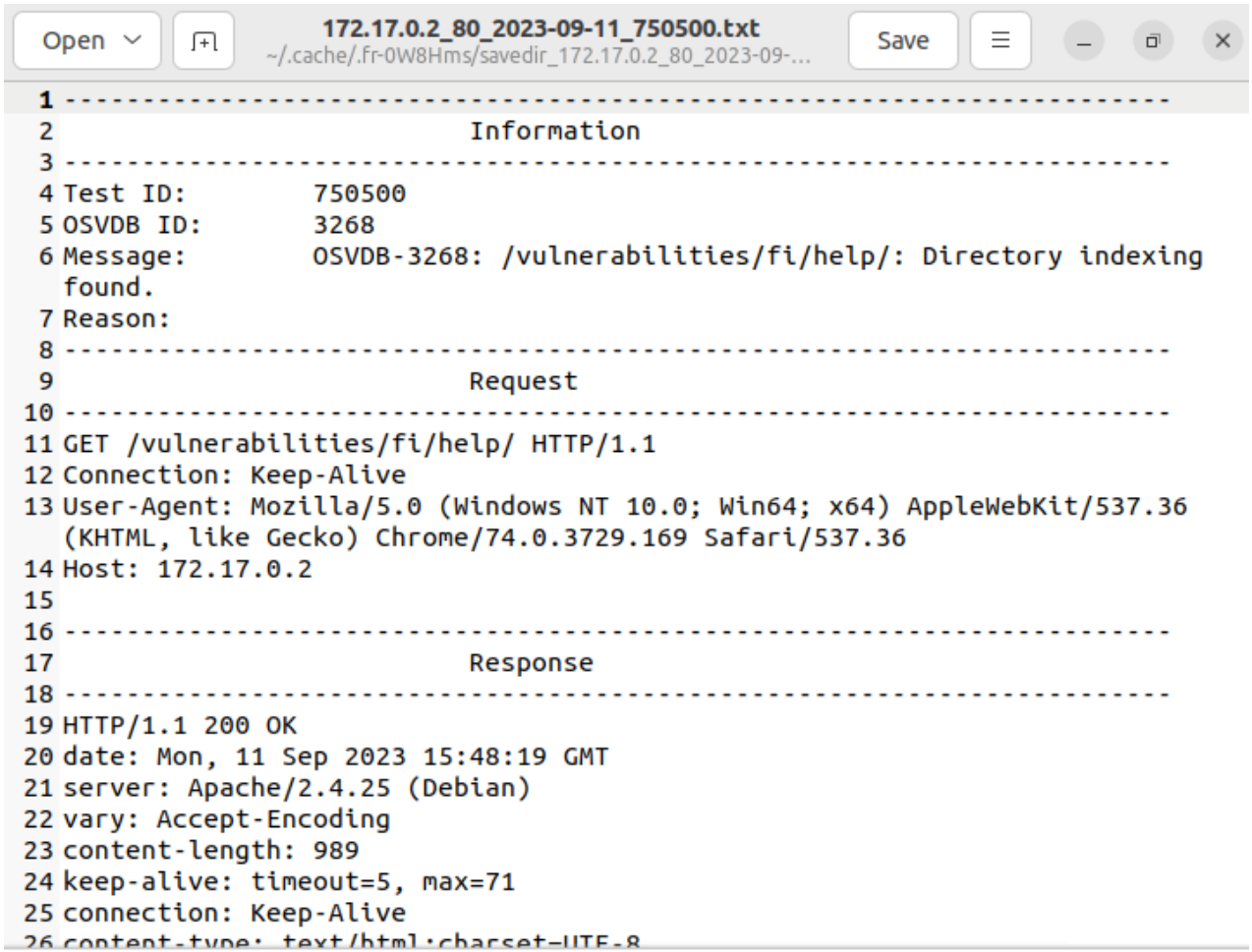
Εικόνα 15. Ευπάθεια που βρέθηκε από το Nikto (X-Frame-Options header missing)

Ο πρώτος τομέας της αναφοράς αντιστοιχεί στις πληροφορίες της ευπάθειας, όπου περιλαμβάνονται: (α) το OSVDB ID που είναι το μοναδικό αναγνωριστικό αναφοράς κάθε ευπάθειας, (β) το μήνυμα, όπου φαίνεται ο τίτλος της ευπάθειας, και (γ) η αιτιολόγηση, η οποία χωρίζεται στην αίτηση (request) και απάντηση (response) της κλήσης, όπου και βρέθηκε η ευπάθεια.

Εικόνα 16. Ευπάθεια που βρέθηκε από το Nikto (PHPSESSID without httponly flag)



Εικόνα 17. Ευπάθεια που βρέθηκε από το Nikto (X-Content-Type-Options header not set)



Εικόνα 18. Ευπάθεια που βρέθηκε από το Nikto (Directory Indexing)

```
1 ---------------------------------------------------------------------------
2                            Information
3 ---------------------------------------------------------------------------
4 Test ID:         000045
5 OSVDB ID:        0
6 Message:         /vulnerabilities/fi/help/: Help directory should not be
  accessible
7 Reason:          Response Code Match
8 ---------------------------------------------------------------------------
9                            Request
10 ---------------------------------------------------------------------------
11 GET /vulnerabilities/fi/help/ HTTP/1.1
12 Connection: Keep-Alive
13 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
   (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36
14 Host: 172.17.0.2
15
16 ---------------------------------------------------------------------------
17                           Response
18 ---------------------------------------------------------------------------
19 HTTP/1.1 200 OK
20 date: Mon, 11 Sep 2023 15:48:19 GMT
21 server: Apache/2.4.25 (Debian)
22 vary: Accept-Encoding
23 content-length: 989
24 keep-alive: timeout=5, max=71
```

Εικόνα 19. Ευπάθεια που βρέθηκε από το Nikto (Accessible Directory)

6.1.1 Αποτίμηση της Χρήσης της Εφαρμογής μας για το Παραπάνω Σενάριο και Συμπεράσματα

Μετά την συνδυασμένη εκτέλεση των 2 εργαλείων δυναμικής ανάλυσης, Nikto και Arachni, θα παραθέσουμε μία ανάλυση των τρωτοτήτων που βρέθηκαν. Συνδυαστικά έχουμε τις εξής τρωτότητες.

Σημαία HttpOnly (Εικόνες 14, 16):

Η HttpOnly είναι μια πρόσθετη σημαία που περιλαμβάνεται σε μια κεφαλίδα απόκρισης HTTP Set-Cookie. Η χρήση της σημαίας HttpOnly κατά τη δημιουργία ενός cookie συμβάλλει στον μετριασμό του κινδύνου πρόσβασης εκτέλεσης σεναρίων από την πλευρά του πελάτη στο προστατευμένο cookie (αν το υποστηρίζει το πρόγραμμα περιήγησης).

Εάν το πρόγραμμα περιήγησης υποστηρίζει αυτή τη σημαία, την HttpOnly, στην κεφαλίδα απόκρισης HTTP, τότε εμποδίζει τη δέσμη ενεργειών από την πλευρά του πελάτη όσον αφορά την πρόσβαση στο cookie. Ως αποτέλεσμα, το πρόγραμμα περιήγησης (κυρίως ο Internet Explorer) δεν θα αποκαλύψει το cookie σε τρίτο μέρος, ακόμη και αν υπάρχει αδυναμία δέσμης ενεργειών μεταξύ τοποθεσιών (XSS) και ένας χρήστης κάνει ακούσια κλικ σε έναν σύνδεσμο που το εκμεταλλεύεται.

Η σημαία HttpOnly θα αγνοηθεί από ένα πρόγραμμα περιήγησης εφόσον δεν την υποστηρίζει. Σε αυτή την περίπτωση, αν ένας ιστότοπος επιχειρήσει να δημιουργήσει ένα cookie HttpOnly, αυτό θα έχει ως αποτέλεσμα τη δημιουργία ενός συμβατικού, προσβάσιμου σε σενάριο cookie. Εξαιτίας αυτού, το cookie (συνήθως το cookie περιόδου λειτουργίας) γίνεται ανοιχτό σε κλοπή ή αλλοίωση από τον επιτιθέμενο. [https://owasp.org/www-community/HttpOnly]

Unencrypted Password Form (Εικόνα 12):

Η κεφαλίδα HTTP Strict Transport Security (STS) είναι μια δυνατότητα ασφαλείας που δίνει οδηγίες στα προγράμματα περιήγησης ιστού να χρησιμοποιούν αποκλειστικά το HTTPS όταν επικοινωνούν με έναν ιστότοπο. Αυτό βοηθά στην αποτροπή επιθέσεων μεσάζοντα/ενδιάμεσου (Man-in-the-Middle – MITM) και άλλων απειλών ασφαλείας, διασφαλίζοντας ότι όλη η κίνηση μεταξύ του πελάτη και του διακομιστή είναι κρυπτογραφημένη.

Εάν ένας ιστότοπος χρησιμοποιεί SSL/TLS για την κρυπτογράφηση της κυκλοφορίας, αλλά δεν ορίζει την κεφαλίδα STS, μπορεί να είναι ευάλωτος σε επιθέσεις υποβάθμισης. Μια επίθεση υποβάθμισης είναι ένας τύπος επίθεσης MITM στην οποία ένας εισβολέας παρεμποδίζει το αρχικό αίτημα HTTPS από τον πελάτη και υποβαθμίζει τη σύνδεση σε HTTP. Αυτό επιτρέπει στον εισβολέα να υποκλέψει και να τροποποιήσει την κίνηση μεταξύ του πελάτη και του διακομιστή, δυνητικά διακυβεύοντας ευαίσθητα δεδομένα, όπως διαπιστευτήρια σύνδεσης ή πληροφορίες πιστωτικής κάρτας. Αυτό μπορεί να πραγματοποιηθεί μέσω διαφόρων επιθέσεων MITM ή μέσω συλλήψεων πακέτων δικτύου

Χωρίς την κεφαλίδα STS, ένα πρόγραμμα περιήγησης μπορεί να είναι ευάλωτο σε αυτόν τον τύπο επίθεσης, καθώς ενδέχεται να μην επιστρέψει αυτόματα σε HTTPS μετά από μια επίθεση υποβάθμισης. Αυτό σημαίνει ότι ο χρήστης μπορεί να συνεχίσει να στέλνει ευαίσθητα δεδομένα μέσω μιας μη κρυπτογραφημένης σύνδεσης HTTP χωρίς να συνειδητοποιεί ότι η σύνδεση έχει υποβαθμιστεί.

Για να μετριαστεί αυτή η ευπάθεια, συνιστάται να οριστεί η κεφαλίδα Strict Transport Security (STS) στον διακομιστή ιστού. Αυτή η κεφαλίδα καθοδηγεί το πρόγραμμα περιήγησης να

χρησιμοποιεί μόνο το HTTPS όταν επικοινωνεί με τον ιστότοπο για μια καθορισμένη χρονική περίοδο, ακόμα κι αν ο χρήστης πληκτρολογήσει "http://" αντί για "https://" στη διεύθυνση URL. Αυτό βοηθά στην αποτροπή επιθέσεων υποβάθμισης και διασφαλίζει ότι όλη η κίνηση μεταξύ του πελάτη και του διακομιστή είναι κρυπτογραφημένη.

X-Content-Type-Options header not set (Εικόνα 17):

Η κεφαλίδα 'Content-Type' χρησιμοποιείται για τον καθορισμό του τύπου MIME του περιεχομένου που αποστέλλεται σε μια απόκριση HTTP. Ο τύπος MIME υποδεικνύει τη φύση και τη μορφή του περιεχομένου, όπως κείμενο, HTML, εικόνα ή βίντεο. Η κεφαλίδα "Επιλογές τύπου X-Content" (X-Content-Type-Options) είναι μια δυνατότητα ασφαλείας που καθοδηγεί το πρόγραμμα περιήγησης να μην παρακάμψει τον τύπο MIME που καθορίζεται στην κεφαλίδα "Τύπος περιεχομένου" και να αποτρέψει το MIME sniffing.

Το MIME sniffing είναι μια διαδικασία όπου το πρόγραμμα περιήγησης προσπαθεί να μαντέψει τον τύπο του περιεχομένου με βάση το περιεχόμενό του και όχι την κεφαλίδα «Τύπος περιεχομένου». Αυτό μπορεί να οδηγήσει σε ευπάθειες ασφαλείας όταν το περιεχόμενο που “οσφραίνεται” είναι κακόβουλο ή έχει δημιουργηθεί για να εκμεταλλευτεί αυτή τη δυνατότητα.

Εάν η κεφαλίδα "X-Content-Type-Options" δεν έχει οριστεί, θα μπορούσε να επιτρέψει στον φυλλομετρητή να αποδώσει το περιεχόμενο του ιστότοπου με διαφορετικό τρόπο από τον τύπο MIME. Αυτό μπορεί να οδηγήσει σε διάφορους κινδύνους ασφαλείας, όπως επιθέσεις μεταξύ δέσμης ενεργειών (XSS), όπου ένας εισβολέας μπορεί να εισάγει κακόβουλο κώδικα στη σελίδα και να ξεγελάσει το πρόγραμμα περιήγησης ώστε να τον αποδώσει ως διαφορετικό τύπο MIME, με αποτέλεσμα ο κακόβουλος κώδικας να εκτελείται κανονικά.

Για να μετριαστεί αυτή η ευπάθεια, συνιστάται να ορίσετε την κεφαλίδα 'X-Content-Type-Options' σε "nosniff". Αυτό θα δώσει οδηγίες στο πρόγραμμα περιήγησης να μην παρακάμψει τον τύπο MIME που καθορίζεται στην κεφαλίδα "Τύπος περιεχομένου" και να αποτρέψει το MIME sniffing. Αυτό βοηθά στην αποτροπή επιθέσεων XSS και άλλων απειλών ασφαλείας που μπορεί να προκύψουν από την απόδοση περιεχομένου με απροσδόκητους τρόπους. Επιπλέον, είναι σημαντικό να διασφαλιστεί ότι όλο το περιεχόμενο αποστέλλεται με την κατάλληλη κεφαλίδα «Τύπος περιεχομένου» για να αποφευχθεί οποιαδήποτε σύγχυση ή ασάφεια σχετικά με τον τύπο του περιεχομένου που αποστέλλεται.

Directory Indexing (Εικόνα 18):

Αυτή η ευπάθεια ανήκει στην κατηγορία της Εσφαλμένης Διαμόρφωσης Ασφάλειας (Security Misconfiguration) και συμπεριλαμβάνεται στο OWASP Top-10 2023 [https://owasp.org/API-Security/editions/2023/en/0xa8-security-misconfiguration/].

Η εσφαλμένη διαμόρφωση ενός συστήματος, εφαρμογής ή διακομιστή που υπονομεύει την ασφάλεια μπορεί να οδηγήσει σε ευπάθειες. Στο πλαίσιο των διακομιστών ιστού και των εφαρμογών, η μη τήρηση των βέλτιστων πρακτικών ή η μη ενημέρωση του λογισμικού μπορεί να οδηγήσει σε εσφαλμένες ρυθμίσεις παραμέτρων ασφαλείας.

Στην συγκεκριμένη περίπτωση της καταχώρισης καταλόγου (directory indexing), η ευρετηρίαση καταλόγου επιτρέπει στους χρήστες να προβάλλουν μια λίστα αρχείων και καταλόγων μέσα σε έναν δεδομένο κατάλογο, όταν δεν υπάρχει διαθέσιμη προεπιλεγμένη ιστοσελίδα (όπως index.html). Η δημιουργία ευρετηρίου καταλόγου μπορεί να εκθέσει ιδιωτικά αρχεία και φακέλους σε οποιονδήποτε έχει πρόσβαση σε μια διεύθυνση URL, εάν είναι ενεργοποιημένη σε έναν κατάλογο όπου δεν θα έπρεπε.

Ένα σενάριο επίθεσης αποτελεί το εξής:

Ο διακομιστής δεν απενεργοποιεί τη λίστα καταλόγου. Ένας εισβολέας μαθαίνει ότι η καταχώριση καταλόγων επιτρέπεται. Ο εισβολέας εντοπίζει και κατεβάζει τις κλάσεις Java, οι οποίες στη συνέχεια τροποποιούνται και απομεταγλωττίζονται για να αποκαλυφθεί ο πηγαίος κώδικας. Στη συνέχεια, ο εισβολέας ανακαλύπτει ένα σοβαρό ζήτημα ελέγχου πρόσβασης εφαρμογής.

Για να μετριαστεί αυτή η ευπάθεια, ο κύκλος ζωής του API πρέπει να περιλαμβάνει:

- Μια επαναλαμβανόμενη διαδικασία σκλήρυνσης που οδηγεί σε γρήγορη και εύκολη ανάπτυξη ενός σωστά κλειδωμένου περιβάλλοντος.
- Εργασίες για έλεγχο και ενημέρωση της διαμόρφωσης σε ολόκληρη τη στοίβα API. Η αξιολόγηση πρέπει να περιλαμβάνει: αρχεία ενορχήστρωσης, στοιχεία API και υπηρεσίες νέφους (cloud).
- Μια αυτοματοποιημένη διαδικασία για τη συνεχή αξιολόγηση της αποτελεσματικότητας της διαμόρφωσης και των ρυθμίσεων σε όλα τα περιβάλλοντα.

Accessible Directory (Εικόνα 19):

Αυτή η ευπάθεια ανήκει στην κατηγορία του Ελέγχου Κατεστραμμένης Πρόσβασης (Broken Access Control) [9].

Συνήθως, η ευπάθεια "ο κατάλογος δεν θα πρέπει να είναι προσβάσιμος" ταξινομείται ως "Ελεγχος κατεστραμμένης πρόσβασης". Όταν μια εφαρμογή ή ένα σύστημα επιβάλλει ακατάλληλα περιορισμούς σχετικά με το σε τι μπορούν να έχουν πρόσβαση ή να χειριστούν οι χρήστες, εμφανίζονται “σπασμένα”, τρωτά σημεία ελέγχου πρόσβασης. Στην περίπτωση προσβάσιμου καταλόγου, περιλαμβάνεται παράνομη πρόσβαση σε καταλόγους και αρχεία που πρέπει να προστατεύονται.

Είναι σημάδι ελαττωματικού ελέγχου πρόσβασης είναι όταν οι χρήστες έχουν μη εξουσιοδοτημένη πρόσβαση σε έναν κατάλογο, όπως αυτός που περιέχει ευαίσθητα δεδομένα ή αρχεία διαμόρφωσης. Για να μην μπορούν οι μη εξουσιοδοτημένοι χρήστες να βλέπουν ή να τροποποιούν αυτούς τους φακέλους, θα πρέπει να υπάρχουν κατάλληλες μέθοδοι ελέγχου πρόσβασης, όπως έλεγχος ταυτότητας, εξουσιοδότηση και δικαιώματα αρχείου/καταλόγου.

X-Frame-Options-Header (Εικόνες 13, 15):

Το πεδίο κεφαλίδας HTTP Επιλογές X-Frame υποδεικνύει μια πολιτική που καθορίζει εάν το πρόγραμμα περιήγησης πρέπει να αποδίδει τον μεταδιδόμενο πόρο μέσα σε ένα πλαίσιο ή ένα iframe (inframe html element). Σε περίπτωση που απουσιάζει αυτή η πολιτική, τότε υπάρχει η δυνατότητα επίθεσης clickjacking από τον επιτιθέμενο [https://owasp.org/www-community/attacks/Clickjacking].

Η επίθεση clickjacking είναι όταν ένας εισβολέας χρησιμοποιεί πολλαπλά διαφανή ή αδιαφανή επίπεδα για να ξεγελάσει έναν χρήστη ώστε να κάνει κλικ σε ένα κουμπί ή έναν σύνδεσμο σε μια πλαισιωμένη σελίδα ενώ ο χρήστης σκόπευε να κάνει κλικ στη σελίδα ανώτατου επιπέδου. Έτσι, ο εισβολέας "πειρατεύει" τα κλικ που προορίζονται για μια νόμιμη σελίδα και τα δρομολογεί σε άλλη, κακόβουλη, που πιθανότατα ανήκει σε άλλη εφαρμογή, τομέα ή και τα δύο.

Χρησιμοποιώντας μια παρόμοια τεχνική, τα πλήκτρα μπορούν επίσης να παραβιαστούν. Με έναν προσεκτικά σχεδιασμένο συνδυασμό φύλλων στυλ, iframes και πλαισίων κειμένου, ο χρήστης μπορεί να πιστέψει ότι πληκτρολογεί έναν κωδικό πρόσβασης σε ένα πεδίο εισόδου, π.χ. κωδικό email, αλλά αντ' αυτού πληκτρολογεί σε ένα αόρατο πλαίσιο που ελέγχεται από τον εισβολέα.

Οι διακομιστές μπορούν να δηλώσουν αυτήν την πολιτική στην κεφαλίδα των απαντήσεών τους HTTP για να αποτρέψουν επιθέσεις clickjacking, κάτι που διασφαλίζει ότι το περιεχόμενό τους δεν ενσωματώνεται σε άλλες σελίδες ή πλαίσια.

Common Sensitive File (Εικόνα 13):

Παρόλο που η διαχείριση των περισσότερων αρχείων σε έναν διακομιστή ιστού γίνεται απευθείας από τον ίδιο τον διακομιστή, δεν είναι ασυνήθιστο να βρίσκουμε αρχεία χωρίς αναφορά ή ξεχασμένα που μπορούν να χρησιμοποιηθούν για τη λήψη σημαντικών πληροφοριών σχετικά με την υποδομή ή την αναγνώριση πληροφοριών.

Οι πιο συνηθισμένες περιπτώσεις περιλαμβάνουν την παρουσία παλιών μετονομασμένων εκδόσεων τροποποιημένων αρχείων, συμπεριλαμβανομένων αρχείων που έχουν φορτωθεί και έχουν ληφθεί ως πηγή, ακόμη και διπλότυπα. Τα αρχεία αντιγράφων ασφαλείας μπορούν επίσης να δημιουργηθούν αυτόματα από το υποκείμενο σύστημα αρχείων στο οποίο είναι αποθηκευμένη η εφαρμογή, μια δυνατότητα που συνήθως αναφέρεται ως "στιγμιότυπο".

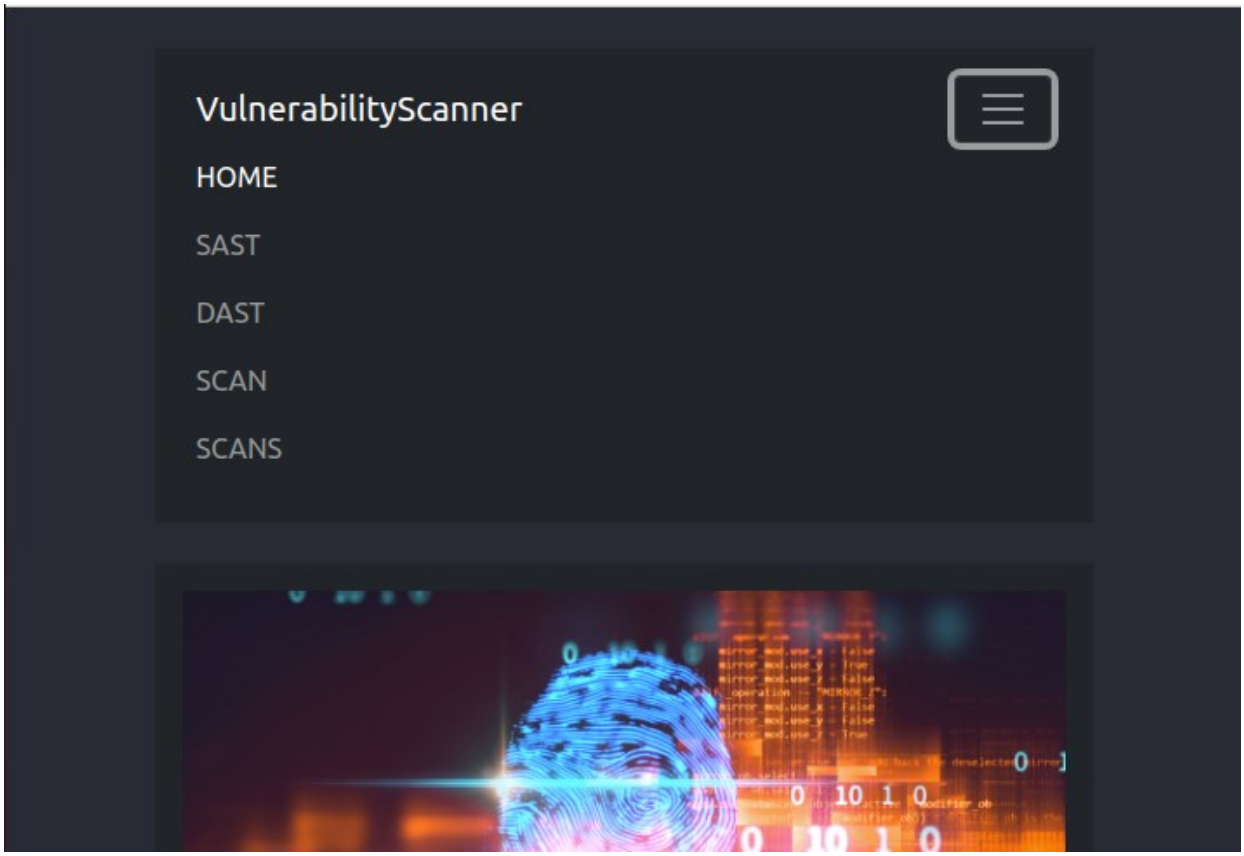
Όλα αυτά τα αρχεία μπορούν να δώσουν στον επιτιθέμενο πρόσβαση σε εσωτερικές λειτουργίες, κερκόπορτες, διεπαφές διαχειριστή ή ακόμα και διαπιστευτήρια για σύνδεση με τη διεπαφή διαχειριστή ή τον διακομιστή βάσης δεδομένων.

Τα αρχεία χωρίς αναφορά μπορούν να αποκαλύψουν ευαίσθητες πληροφορίες που θα μπορούσαν να διευκολύνουν μια στοχευμένη επίθεση κατά της εφαρμογής. Για παράδειγμα, η ύπαρξη αρχείων που περιέχουν διαπιστευτήρια βάσης δεδομένων, αρχεία διαμόρφωσης που περιέχουν αναφορές σε άλλο κρυφό περιεχόμενο, απόλυτες διαδρομές αρχείων κ.λ.π [20].

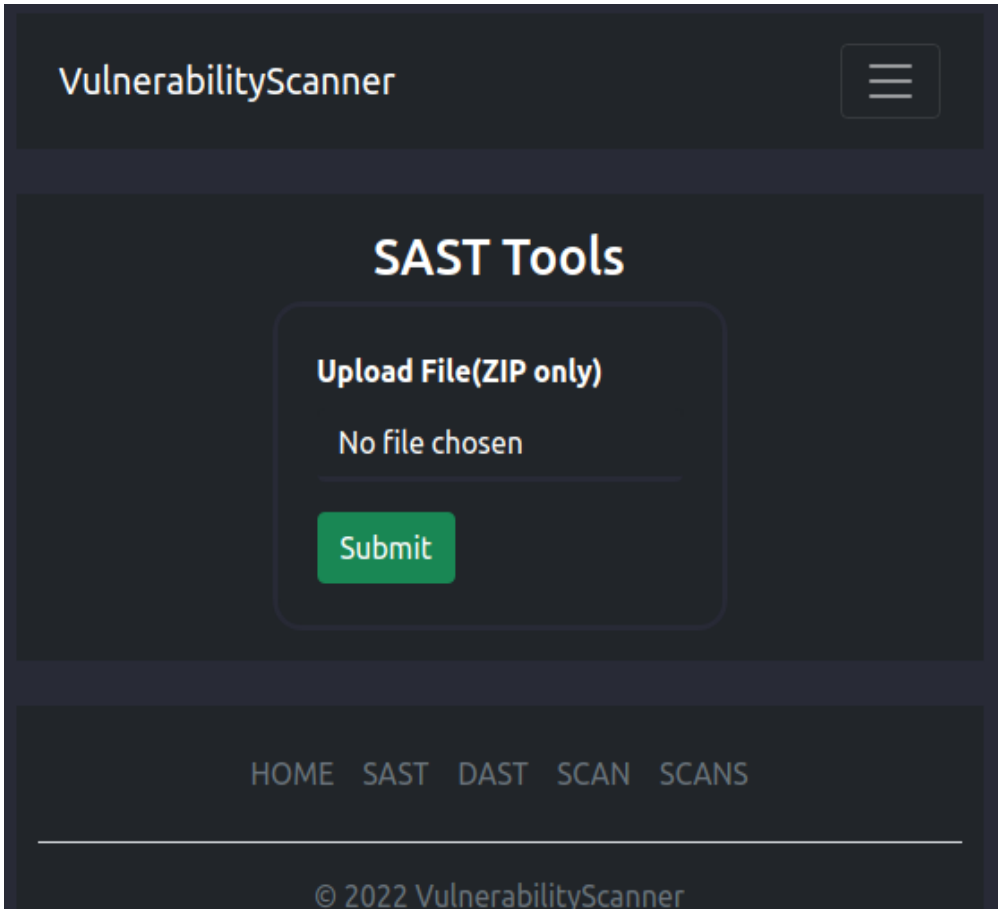
6.2 Χρήση Λειτουργίας Στατικής Σάρωσης (SAST)

Σε αυτήν την ενότητα, επιδεικνύουμε τη συνδυασμένη εκτέλεση στατικής σάρωσης.

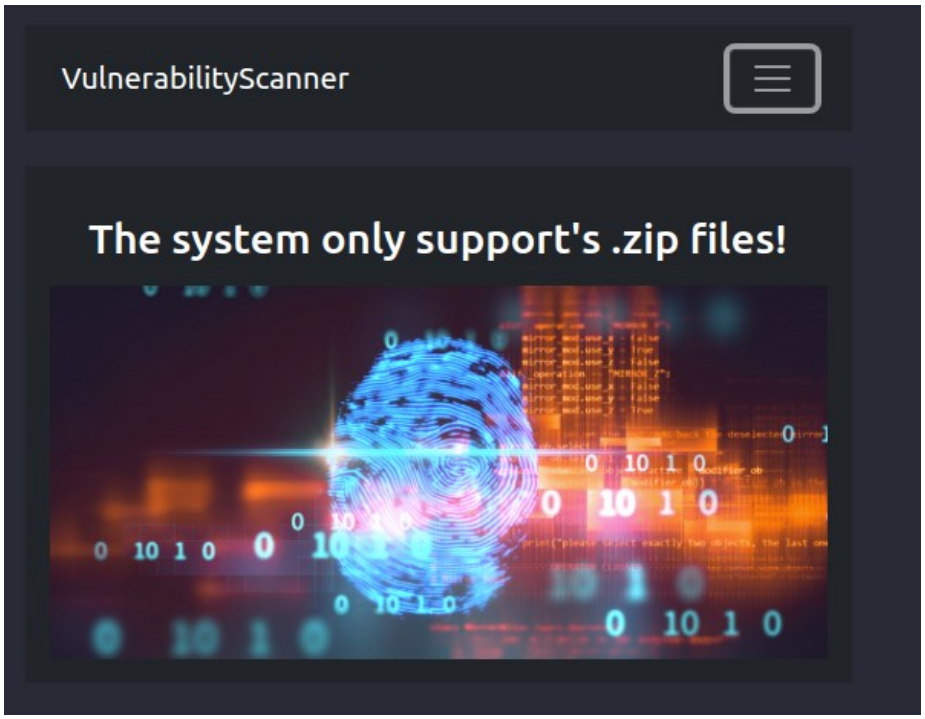
Ο χρήστης αρχικά ανοίγει το burger μενού της εφαρμογής πάνω δεξιά και πατάει την επιλογή ‘SAST’ (δείτε Εικόνα 20). Έπειτα, θα πρέπει να παρέχει ένα συμπιεσμένο αρχείο (δείτε Εικόνα 21) που θα περιλαμβάνει τον πηγαίο κώδικα της εφαρμογής. Σε περίπτωση παροχής άλλου είδους αρχείου, παρουσιάζεται μήνυμα λάθους στον χρήστη (δείτε Εικόνα 22).



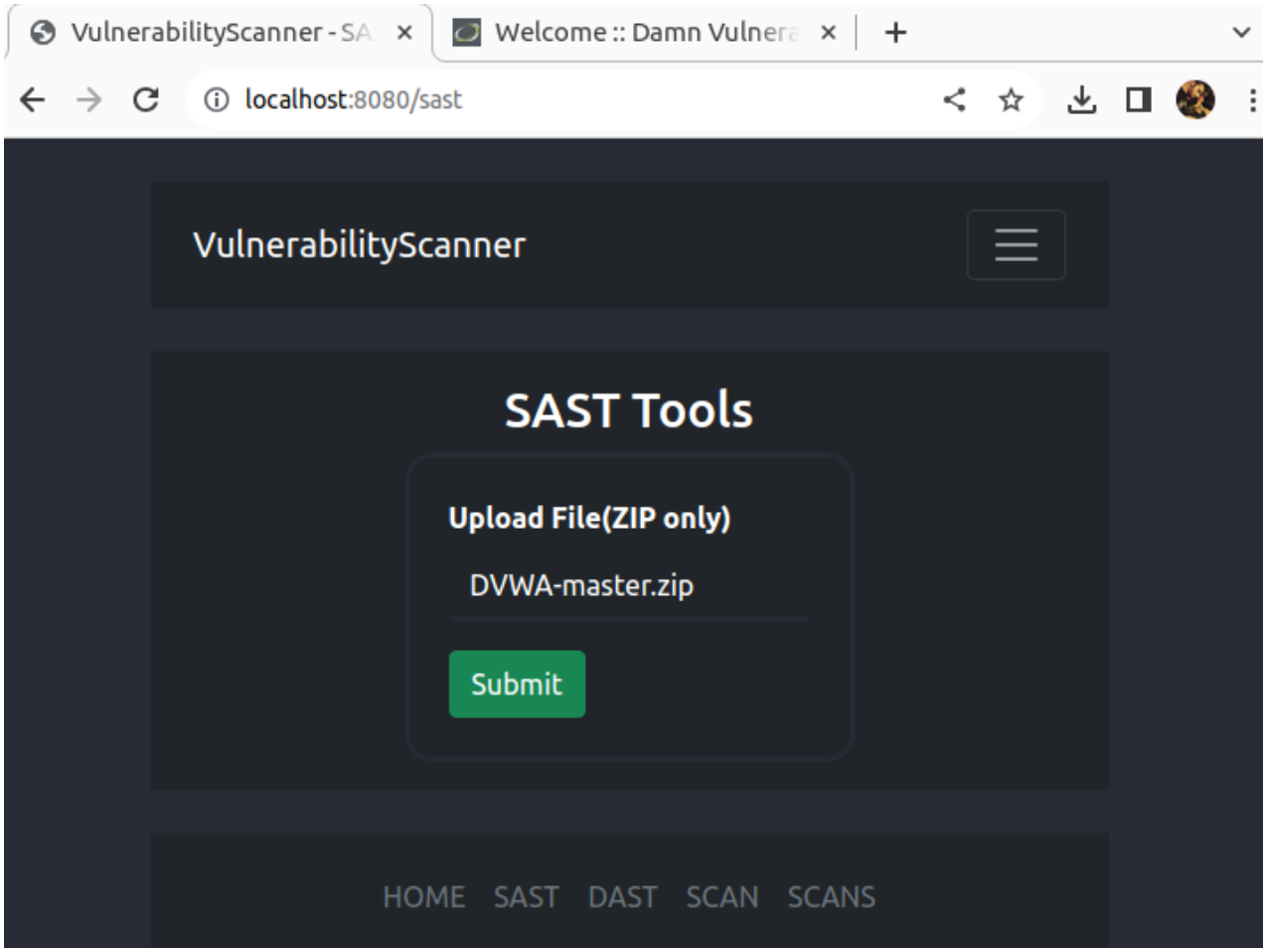
Εικόνα 20. Ο χρήστης πατάει το κουμπί SAST



Εικόνα 21. Φόρμα μεταφόρτωσης .zip αρχείου που περιέχει τον πηγαίο κώδικα προς στατική ανάλυση



Εικόνα 22. Εμφάνιση μηνύματος λάθους όταν ο χρήστης δεν εισάγει .zip αρχείο

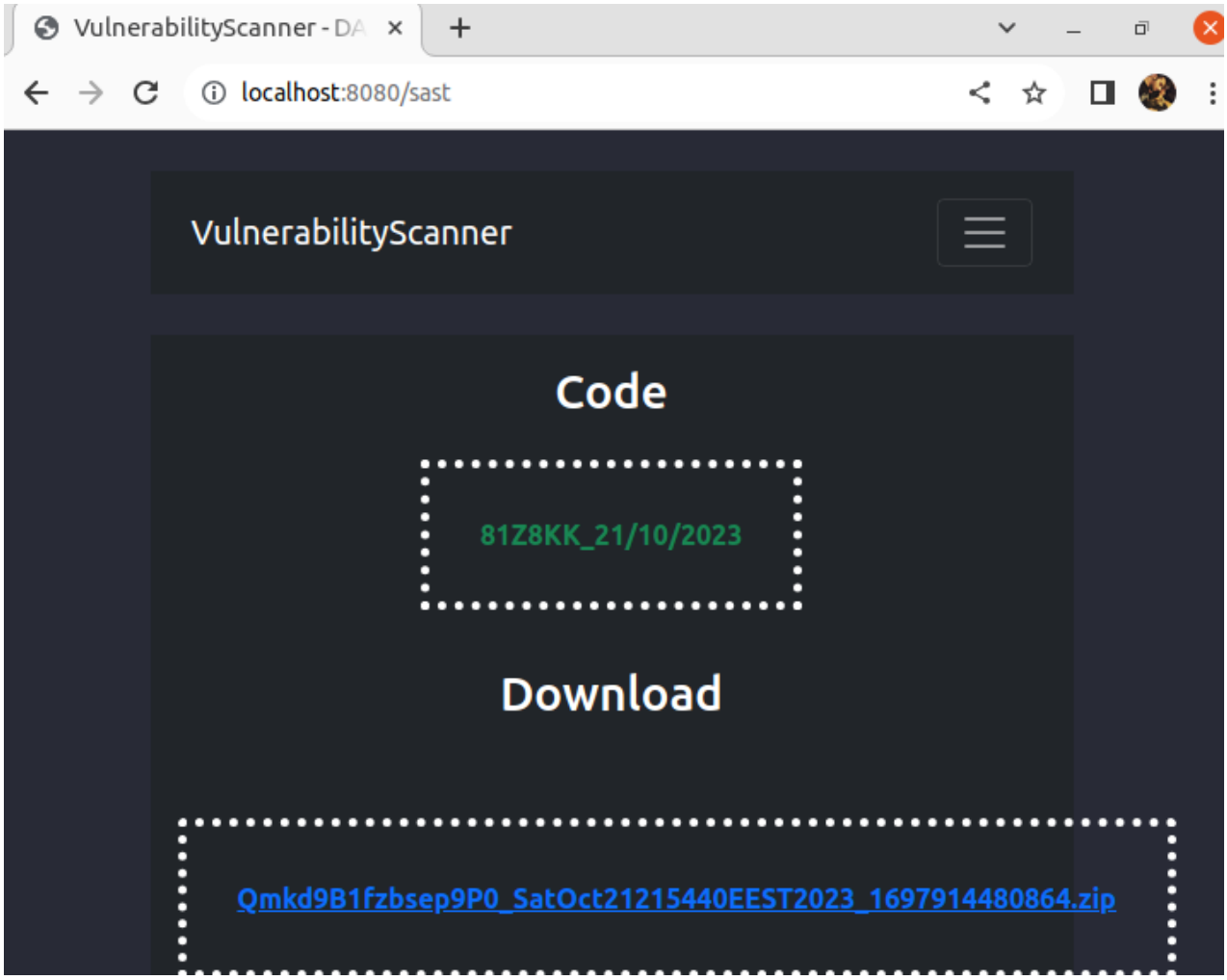


Εικόνα 23. Εισαγωγή (μεταφόρτωση) αρχείου .zip που περιέχει τον πηγαίο κώδικα της εφαρμογής DVWA

Στην Εικόνα 23 παρατηρούμε πως ο χρήστης παρέχει το αρχείο zip της εσκεμμένα τρωτής εφαρμογής DVWA, η οποία ελέγχθηκε και κατά την επίδειξη της δυνατότητας της δυναμικής ανάλυσης της εφαρμογής μας. Το αρχείο αυτό μπορεί να εκφορτωθεί από το αποθετήριο της εφαρμογής DVWA στο Github[23].

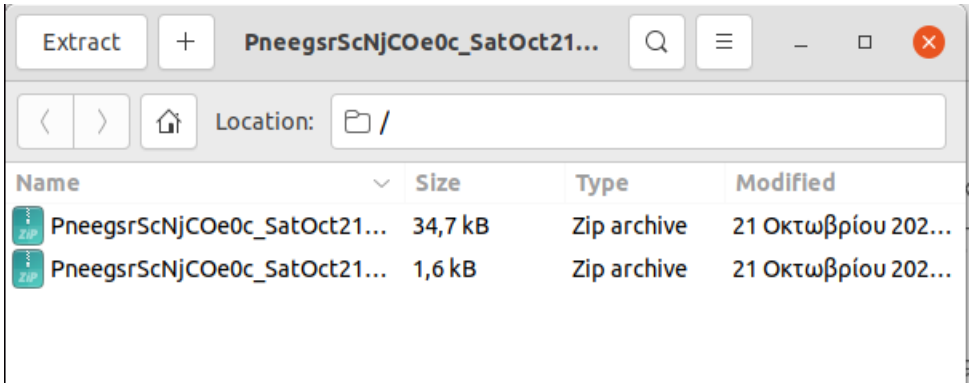
Μόλις ο χρήστης πατήσει το κουμπί Submit, θα αρχίσει η εκτέλεση των 2 εργαλείων στατικής σάρωσης: NodeJsScan, Insider.

[23] https://github.com/digininja/DVWA



Εικόνα 24. Ολοκλήρωση σάρωσης των εργαλείων Insider & NodeJsScan

Μόλις ολοκληρωθεί η συνδυαστική εκτέλεση των 2 εργαλείων (δείτε Εικόνα 24), επιστρέφεται στην ίδια σελίδα στον χρήστη ένα αρχείο με την μορφή συνδέσμου προς εκφόρτωση που περιέχει τις αναφορές και των 2 εργαλείων. Το περιεχόμενο του συμπιεσμένου αυτού αρχείου φαίνεται στην Εικόνα 25.



Εικόνα 25. Αναφορές των εργαλείων Insider & NodeJsScan

Score Security 26/100

Resume of Vulnerabilities

None	0
Low	0
Medium	0
High	13
Critical	0
Total	13

You are using the Insider open source version. If you like the product and want more features, visit http://insidersec.io and get to know our enterprise version.

If you are a developer, then you can contribute to the improvement of the software while using an ope source version

DRA - Data Risk Analytics

File: DVWA-master/CHANGELOG.md
Dra: http://hiderefer.com
Type: url
File: DVWA-master/CHANGELOG.md
Dra: http://www.dvwa.co.uk
Type: url
File: DVWA-master/COPYING.txt
Dra: http://fsf.org/
Type: url
File: DVWA-master/Dockerfile

CVSS: 7.4
Severity:
Class: login.php (39:58)
VulnerabilityID :1ca194ab8ebbbb60d691e384725a13b5
Method: $query = "SELECT * FROM `users` WHERE user='$user' AND password='$pass';";
Description: File contains sensitive information written directly, such as usernames, passwords, keys, etc.
ClassMessage: DVWA-master/login.php (39:58)
Recomendation: Credentials must not be stored in the Git code or repository. There are 'Secrets Management' solutions that can be used to store secrets or use Pipeline resources.

CVSS: 7.4
Severity:
Class: high.php (19:60)
VulnerabilityID :a0626aab7127d677db570c2c9ee2343f
Method: $query = "SELECT * FROM `users` WHERE user = '$user' AND password = '$pass';";
Description: File contains sensitive information written directly, such as usernames, passwords, keys, etc.
ClassMessage: DVWA-master/vulnerabilities/brute/source/high.php (19:60)
Recomendation: Credentials must not be stored in the Git code or repository. There are 'Secrets Management' solutions that can be used to store secrets or use Pipeline resources.

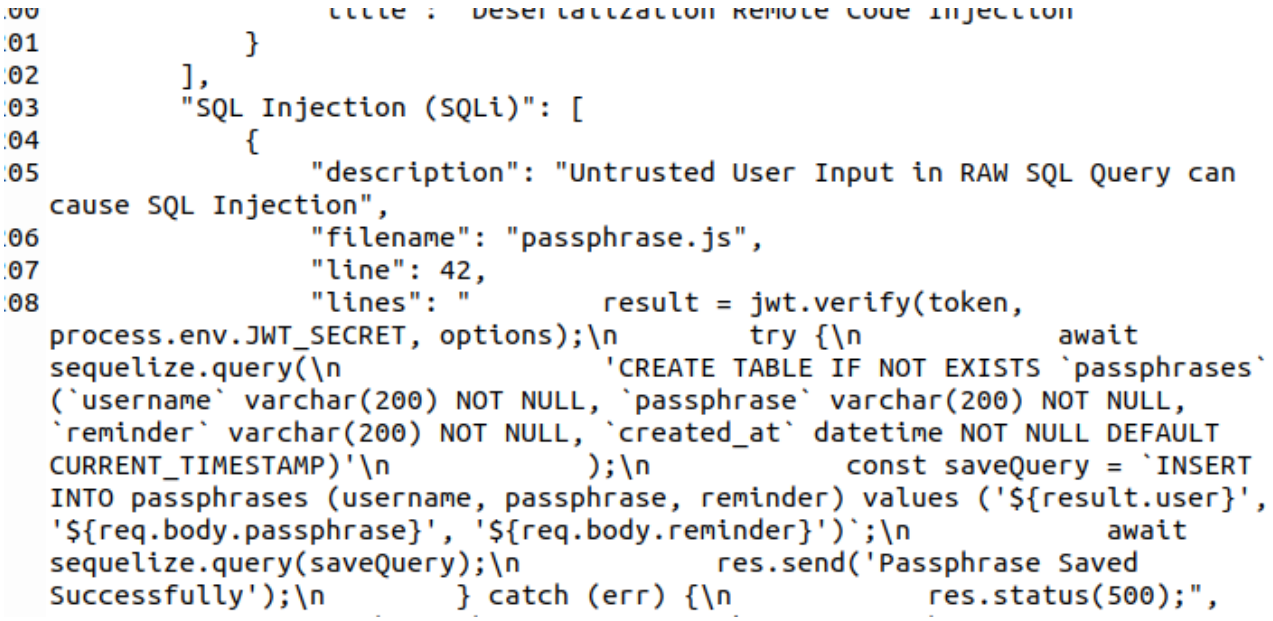
Εικόνα 26. Ευπάθειες που βρέθηκαν από το Insider (Sensitive Information Leak)

Η αναφορά του Insider (δείτε Εικόνα 26) περιέχει των αριθμό των ευπαθειών που βρέθηκαν, χωρισμένες σε χαμηλής, μεσαίας και υψηλής κρισιμότητας (low, medium & high severity). Για

κάθε σημείο στον κώδικα που βρέθηκε μια ευπάθεια, βλέπουμε το CVSS σκορ της [2] (δηλαδή τον δείκτη σοβαρότητας της ευπάθειας), σε ποια κλάση βρέθηκε, το ID (αναγνωριστικό) της ευπάθειας, την μέθοδο όπου βρέθηκε ο ευπαθής κώδικας, μια σύντομη περιγραφή, και μια πρόταση για αντιμετώπιση του κινδύνου.

```
141     "missing_sec_header": {
142         "Web Security": [
143             {
144                 "description": "Content Security Policy (CSP), a mechanism
    web applications can use to mitigate a broad class of content injection
    vulnerabilities, such as cross-site scripting (XSS). CSP Header was not
    found.",
145                 "tag": "web",
146                 "title": "Missing Security Header - Content-Security-Policy
    (CSP)"
147             },
148             {
149                 "description": "X-Frame-Options (XFO) header provides
    protection against Clickjacking attacks.",
150                 "tag": "web",
151                 "title": "Missing Security Header - X-Frame-Options (XFO)"
152             },
153             {
154                 "description": "Strict-Transport-Security (HSTS) header
    enforces secure (HTTP over SSL/TLS) connections to the server.",
155                 "tag": "web",
156                 "title": "Missing Security Header - Strict-Transport-
    Security (HSTS)"
157             },
158             {
159                 "description": "Public Key Pins (HPKP) ensures that
```

Εικόνα 27. Ευπάθειες που βρέθηκαν από το NodeJsScan (Missing Sec Header)



Εικόνα 28. Ευπάθεια που βρέθηκε από το NodeJsScan (SQL Injection)

Η αναφορά του NodeJsScan (δείτε Εικόνες 27-28) περιλαμβάνει σε μορφή JSON την κάθε ευπάθεια που βρέθηκε, την κατηγορία στην οποία ανήκει, μαζί με μια σύντομη περιγραφή της.

6.2.1 Αποτίμηση της Χρήσης της Εφαρμογής μας για το Παραπάνω Σενάριο και Συμπεράσματα

Μετά την συνδυασμένη εκτέλεση των 2 εργαλείων στατικής ανάλυσης, Insider και NodeJsScan, θα παραθέσουμε παρακάτω μία ανάλυση των τρωτοτήτων που βρέθηκαν.

“Data Exposure (Data Leakage)” (Εικόνα 26):

Όταν ευαίσθητες πληροφορίες, όπως ονόματα χρήστη, κωδικοί πρόσβασης και κλειδιά, αποθηκεύονται απευθείας σε ένα πίνακα και προσπελάζονται χρησιμοποιώντας εντολές SQL, όπως "SELECT * FROM file", αυτό αντιπροσωπεύει μια κρίσιμη ευπάθεια ασφαλείας. Αυτό αναφέρεται συχνά ως ευπάθεια "έκθεσης δεδομένων" (data exposure) ή "διαρροής δεδομένων" (data leakage). Ακολουθούν οι πιθανοί κίνδυνοι και οι συνέπειες που ενδέχεται να συντελεστούν εφόσον ένας εισβολέας εκμεταλλευτεί αυτήν την ευπάθεια:

Μη εξουσιοδοτημένη πρόσβαση: Ένας εισβολέας μπορεί να αποκτήσει μη εξουσιοδοτημένη πρόσβαση σε ευαίσθητα δεδομένα που είναι αποθηκευμένα σε πίνακες. Για παράδειγμα, χρήση ευαίσθητων δεδομένων για το μάντευμα/εικασία κωδικών ή την κλοπή εταιρικών μυστικών.

Κλοπή πληροφοριών ελέγχου ταυτότητας: Εάν τα ονόματα χρήστη και οι κωδικοί πρόσβασης αποθηκεύονται σε απλό κείμενο ή σε μορφή εύκολα ανακτήσιμη, ένας εισβολέας μπορεί να κλέψει αυτά τα διαπιστευτήρια και να τα χρησιμοποιήσει για διάφορους κακόβουλους σκοπούς, όπως κλοπή ταυτότητας, ανάληψη λογαριασμού ή κλιμάκωση προνομίων.

Βασικές ανταλλαγές: Εάν τα κλειδιά κρυπτογράφησης ή τα κλειδιά API εκτίθενται σε αρχεία, οι εισβολείς θα μπορούσαν να τα χρησιμοποιήσουν για να αποκρυπτογραφήσουν ευαίσθητα δεδομένα ή να εκτελέσουν ενέργειες για λογαριασμό του παραβιασμένου λογαριασμού ή συστήματος.

Χειραγώγηση Δεδομένων: Ανάλογα με το περιεχόμενο της εκτεινόμενης SQL επερώτησης ή εντολής συστήματος, ένας εισβολέας θα μπορούσε να τροποποιήσει, να διαγράψει ή να εισαγάγει δεδομένα στη βάση δεδομένων ή το σύστημά σας, οδηγώντας δυνητικά σε απώλεια δεδομένων ή χειραγώγηση.

“Missing Security Header” (Εικόνα 27):

Missing CSP Header: Η Πολιτική Ασφάλειας Περιεχομένου (CSP) είναι μια κεφαλίδα HTTP μέσω της οποίας οι κάτοχοι ιστότοπου ορίζουν ένα σύνολο κανόνων ασφαλείας που πρέπει να ακολουθεί το πρόγραμμα περιήγησης κατά την απόδοση του ιστοτόπου τους. Η πιο κοινή χρήση είναι ο καθορισμός μιας λίστας εγκεκριμένων πηγών περιεχομένου που μπορεί να φορτώσει το πρόγραμμα περιήγησης. Αυτό μπορεί να χρησιμοποιηθεί για τον αποτελεσματικό μετριασμό των επιθέσεων Cross-Site Scripting (XSS) και Clickjacking.

Missing HSTS Header: Ο απομακρυσμένος διακομιστής HTTPS δεν επιβάλλει HTTP Strict Transport Security (HSTS). Η έλλειψη HSTS επιτρέπει επιθέσεις υποβάθμισης, επιθέσεις MITM με απογύμνωση SSL καθώς και αποδυναμώνει τις προστασίες κατά της πειρατείας cookie.

Missing X-Frame-Options Header: δείτε Ενότητα 6.1.1 **Αποτίμηση της** Χρήσης **της** Εφαρμογής **μας για το** Παραπάνω Σενάριο **και** Συμπεράσματα.

“Untrusted User Input” (Εικόνα 28):

Η μη αξιόπιστη είσοδος χρήστη σε ανεπεξέργαστα ερωτήματα SQL είναι μια σοβαρή ευπάθεια ασφαλείας που μπορεί να οδηγήσει σε επιθέσεις έκχυσης SQL (SQL Injection). Η έγχυση SQL πραγματοποιείται όταν ένας εισβολέας είναι σε θέση να χειριστεί ή να εισαγάγει κακόβουλο κώδικα SQL στα ερωτήματα SQL μιας εφαρμογής, εκμεταλλευόμενος ευπάθειες για να εκτελέσει μη εξουσιοδοτημένες ενέργειες στη βάση δεδομένων.

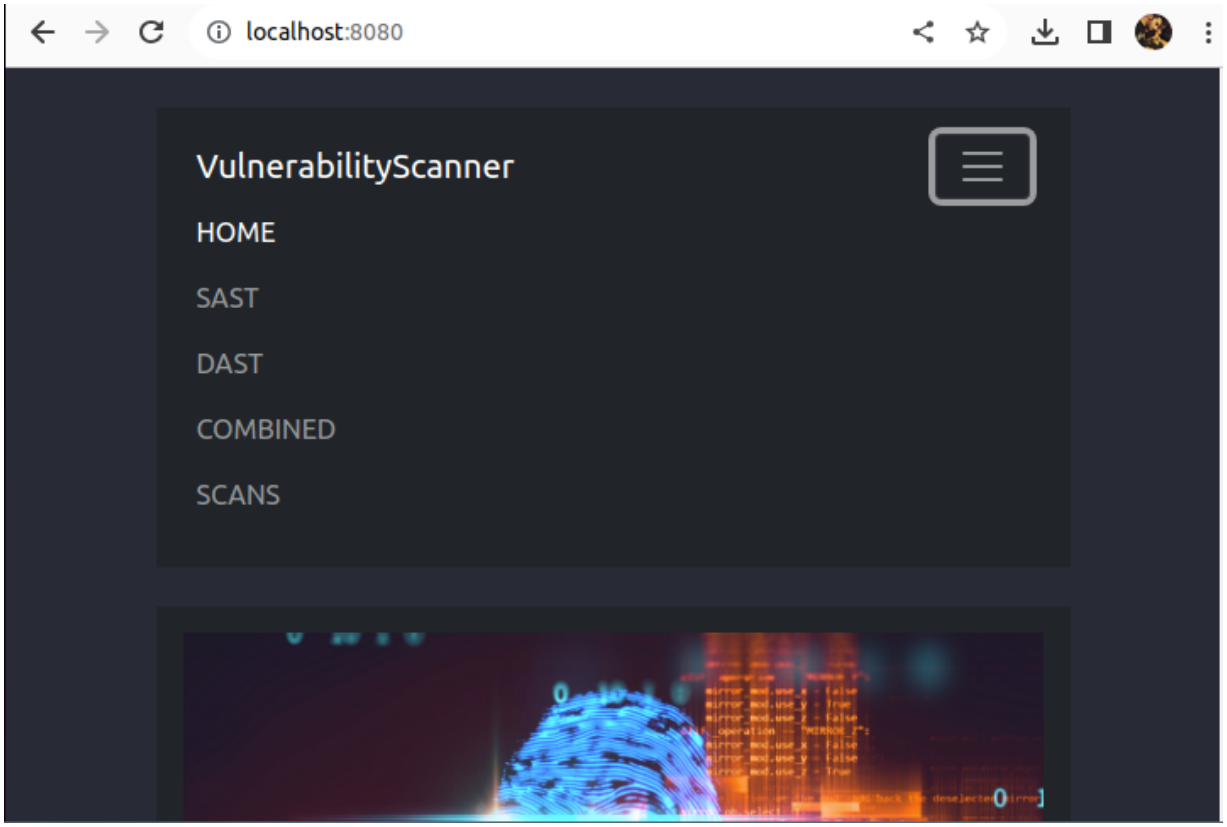
Συγκεκριμένα, ένας εισβολέας μπορεί να εξαγάγει ευαίσθητα δεδομένα από τη βάση δεδομένων. Για παράδειγμα, μπορεί να ανακτήσει ονόματα χρήστη, κωδικούς πρόσβασης, αριθμούς πιστωτικών καρτών ή άλλες εμπιστευτικές πληροφορίες που είναι αποθηκευμένες σε μια βάση δεδομένων. Επίσης, μπορεί να τροποποιήσει ή να χειραγωγήσει δεδομένα που είναι αποθηκευμένα στη βάση δεδομένων. Ειδικότερα, μπορεί να εισαγάγει, να ενημερώσει ή να διαγράψει εγγραφές, οδηγώντας ενδεχομένως σε καταστροφή δεδομένων, απώλεια ή μη εξουσιοδοτημένες αλλαγές.

Τέλος, θα μπορούσε να οδηγήσει ακόμα και σε επίθεση Άρνησης Υπηρεσίας (DoS).Οι εισβολείς μπορούν να δημιουργήσουν κακόβουλα ερωτήματα SQL που καταναλώνουν πάρα πολλούς πόρους συστήματος, με αποτέλεσμα ο διακομιστής της βάσης δεδομένων να γίνεται αργός ή να μην αποκρίνεται, οδηγώντας σε άρνηση υπηρεσίας.

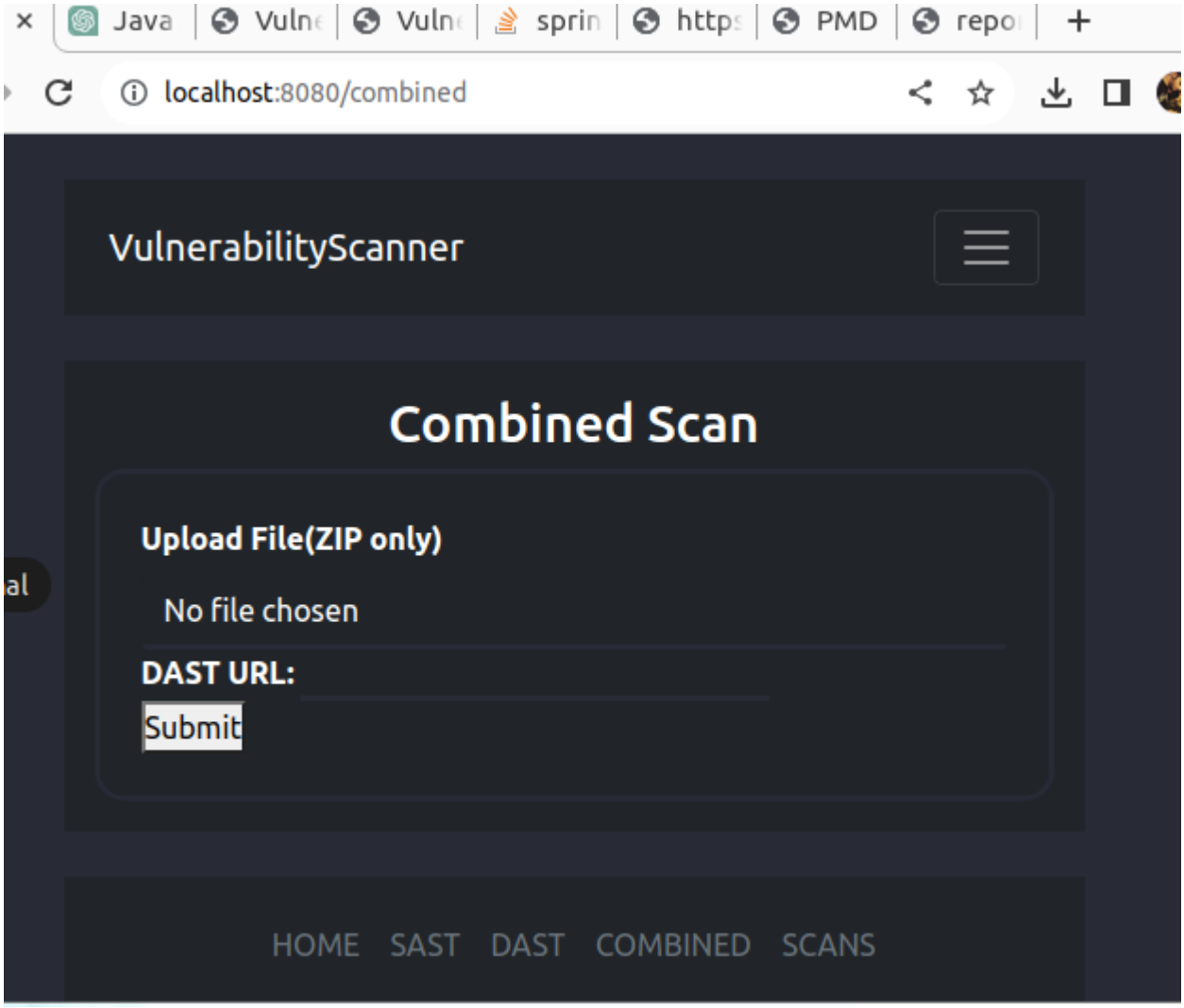
6.3 Συνδυασμένη ανάλυση

Σε αυτήν την ενότητα, θα «τρέξουμε» το σενάριο συνδυαστικής εκτέλεσης όλων των εργαλείων, δηλαδή τόσο δυναμικής όσο και στατικής σάρωσης.

Ο χρήστης αρχικά ανοίγει το burger menu της εφαρμογής πάνω δεξιά και πατάει την επιλογή ‘COMBINED’ (δείτε Εικόνα 29).

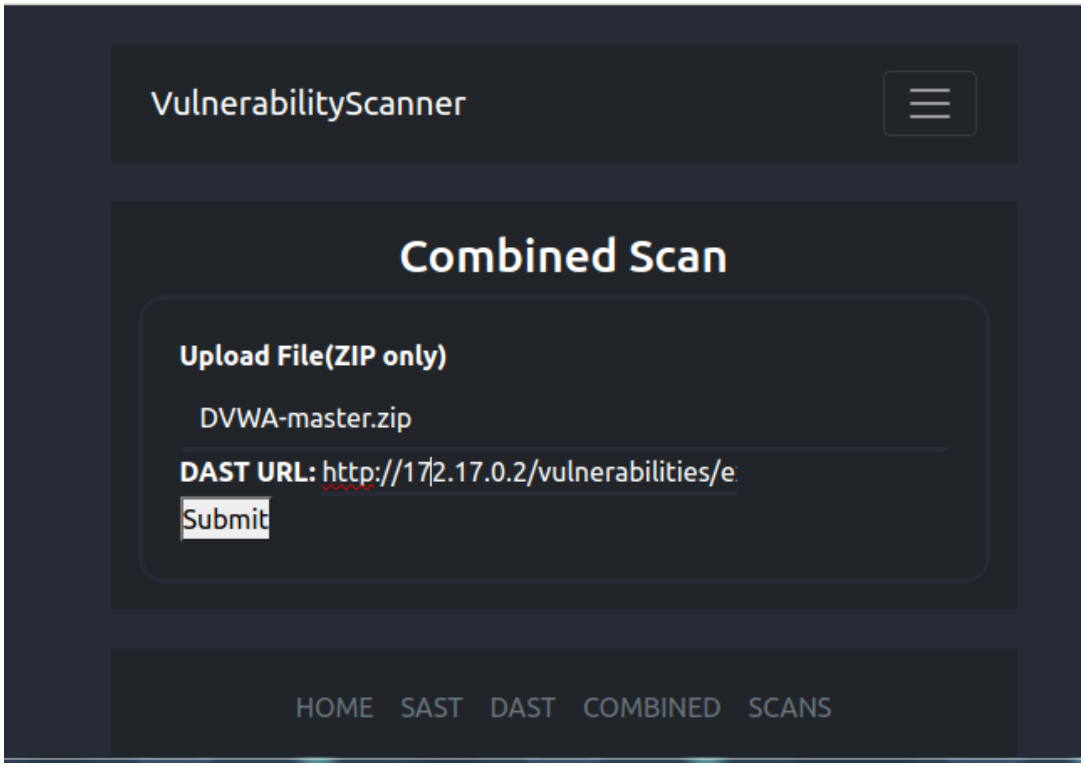


Εικόνα 29. Ο χρήστης επιλέγει από το burger menu την επιλογή "Combined".

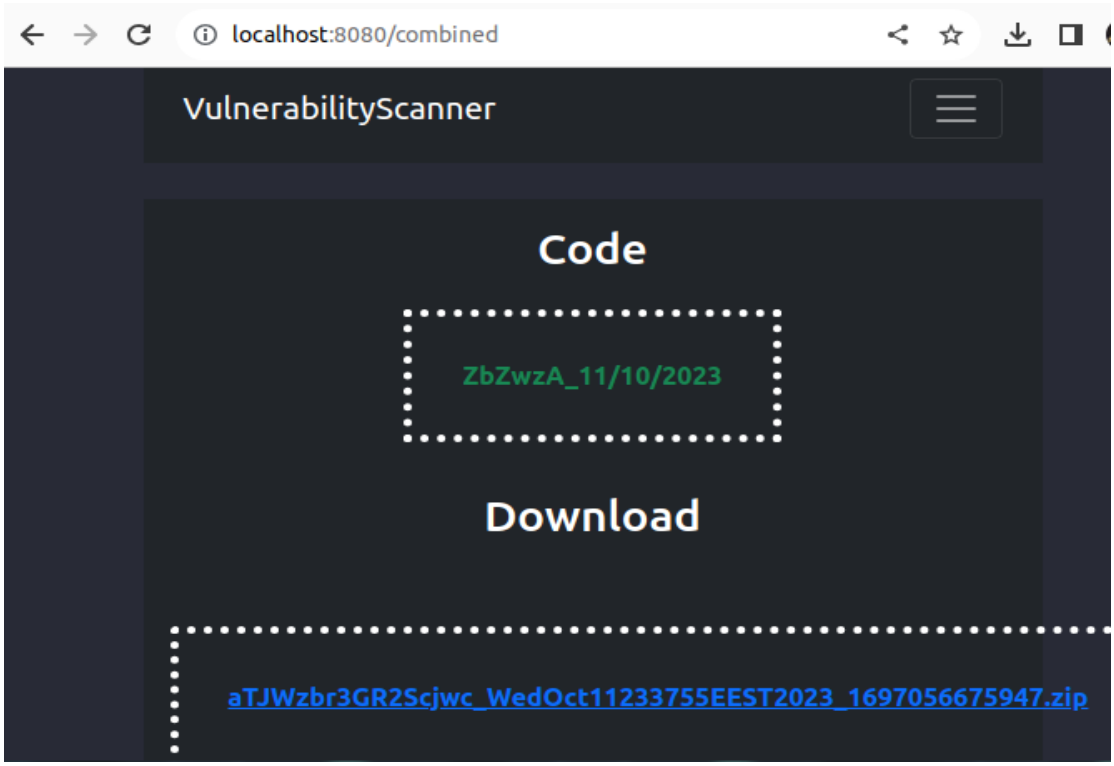


Εικόνα 30. Φόρμα εισαγωγής URL κύριας ιστοσελίδας και συμπιεσμένου αρχείου του πηγαίου κώδικα της εφαρμογής

Έπειτα, ζητείται από τον χρήστη η εισαγωγή του URL της εφαρμογής καθώς και του συμπιεσμένου αρχείου του πηγαίου κώδικα της εφαρμογής (δείτε Εικόνα 30). Στην συγκεκριμένη περίπτωση, δεν μπορεί να πραγματοποιηθεί κάποιος έλεγχος για να επιβεβαιώσει η εφαρμογή μας αυτόματα ότι ο κώδικας που εισάγει ο χρήστης, αντιστοιχεί στον κώδικα που περιλαμβάνεται στο URL της εφαρμογής, συνεπώς ο χρήστης θα πρέπει να είναι προσεκτικός στην συμπλήρωση των 2 πεδίων.



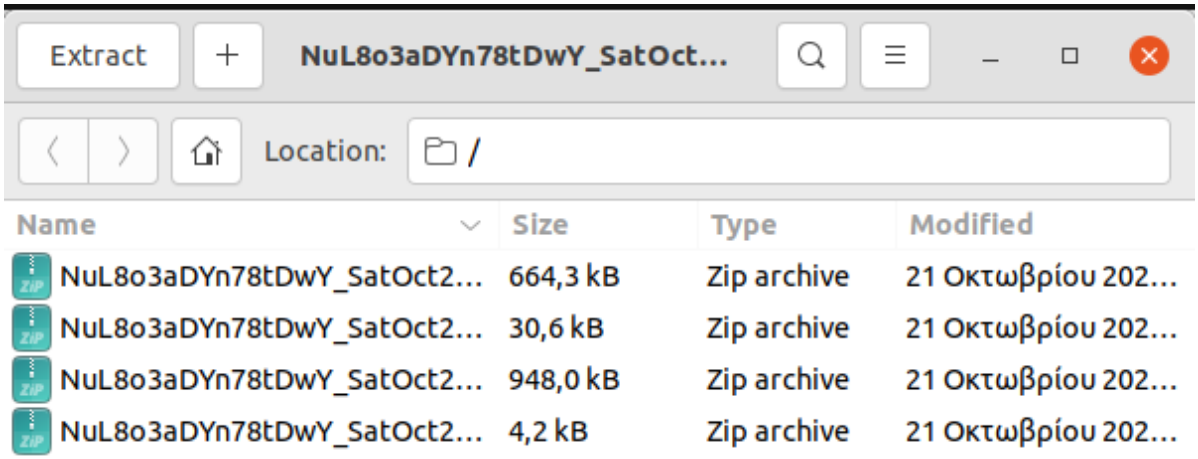
Εικόνα 31. Εισαγωγή συμπιεσμένου αρχείου πηγαίου κώδικα και URL για την εφαρμογή DVWA



Εικόνα 32. Ολοκλήρωση σάρωσης και σύνδεσμος προς το συμπιεσμένο αρχείο των αναφορών των εργαλείων σάρωσης

Εφόσον ο χρήστης συμπληρώσει τα 2 αυτά πεδία και πατήσει το κουμπί Submit (δείτε Εικόνα 31), τότε θα ξεκινήσει η σάρωση της εφαρμογής με στατική και δυναμική ανάλυση. Μόλις, αυτή ολοκληρωθεί, θα παραχθεί ένα συνολικό αρχείο αναφοράς, το οποίο θα είναι διαθέσιμο στον χρήση

προς εκφόρτωση (δείτε **Error! Reference source not found.**). Το περιεχόμενο του αρχείου αυτού φαίνεται στην Εικόνα 33. Δεν θα προχωρήσουμε στην ανάλυση των τρωτοτήτων που εντοπίσθηκαν, διότι αυτή πραγματοποιήθηκε στα 2 προηγούμενα σενάρια χρήσης (δυναμική ανάλυση με πολλαπλά εργαλεία σάρωσης & στατική ανάλυση με πολλαπλά εργαλεία σάρωσης – Ενότητες 6.1 & 6.2).

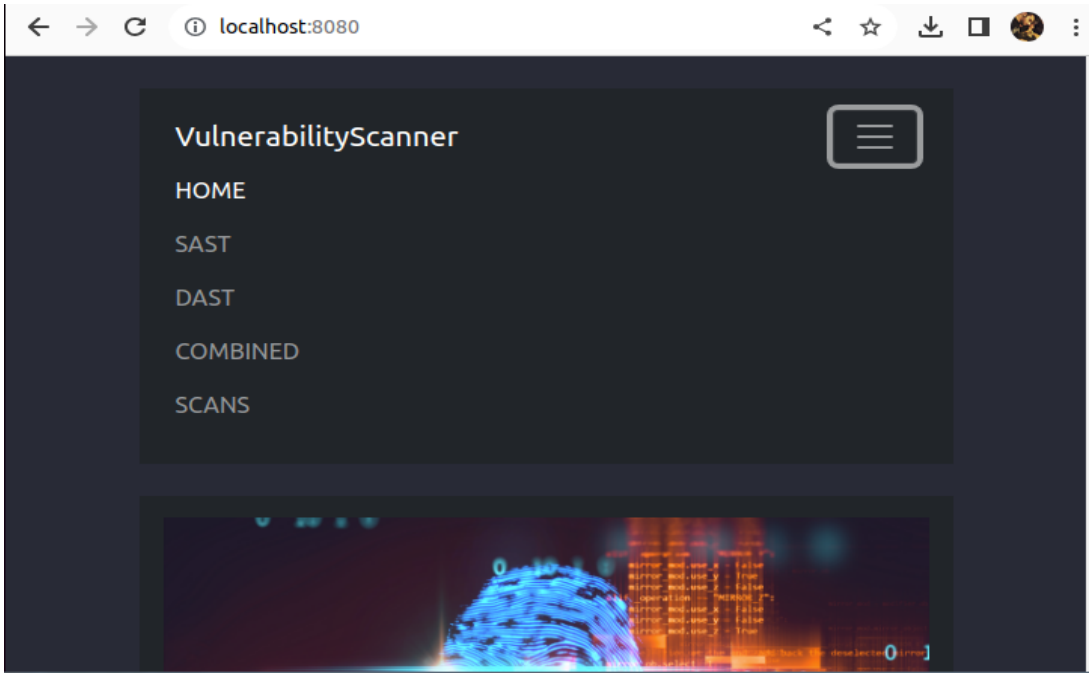


Εικόνα 33. Οι αναφορές όλων των εργαλείων σάρωσης

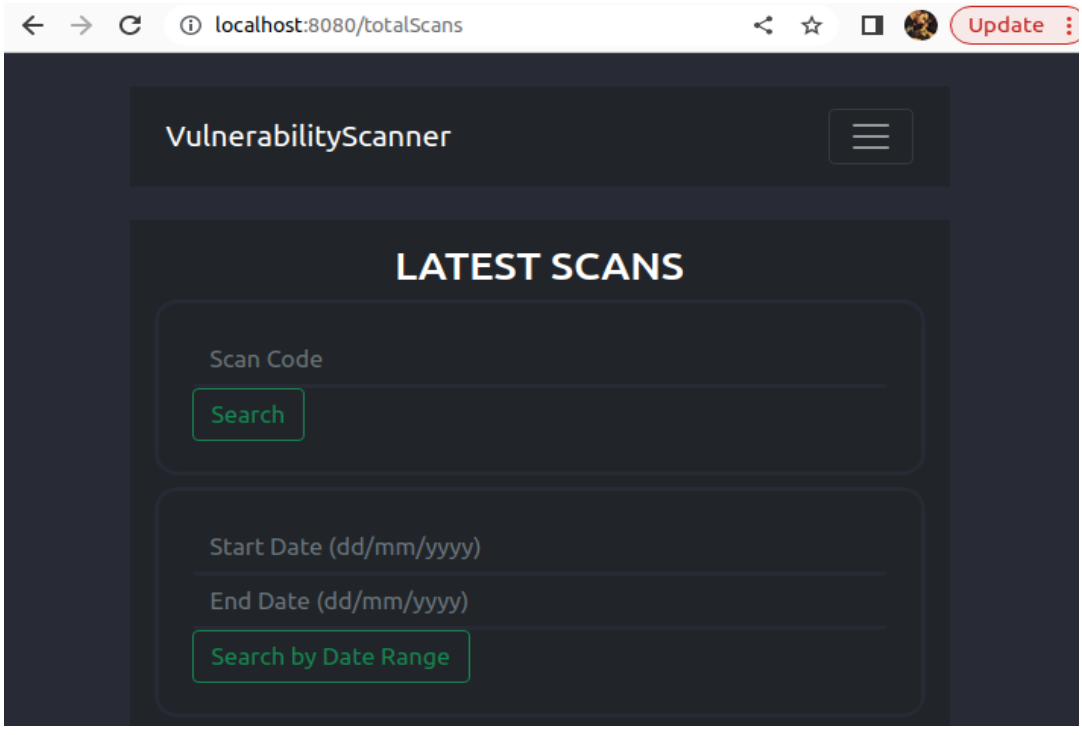
6.4 Αναζήτηση Σαρώσεων

Αυτή αποτελεί και την τελευταία ενότητα του σεναρίου χρήσης της εφαρμογής μας, όπου ο χρήστης μπορεί να δει έναν πίνακα με τις τελευταίες σαρώσεις, και να αναζητήσει κάποια με βάση αναγνωριστικό, ή εύρος ημερομηνιών.

Ο χρήστης αρχικά ανοίγει το burger menu της εφαρμογής πάνω δεξιά και πατάει την επιλογή ‘SCANS’ (δείτε Εικόνα 34). Θα του εμφανιστεί σελίδα που οπτικοποιεί μια φόρμα αναζήτησης (δείτε Εικόνα 35) καθώς και ένα πίνακα με τα αποτελέσματα των πιο πρόσφατων σαρώσεων (δείτε Εικόνα 36).



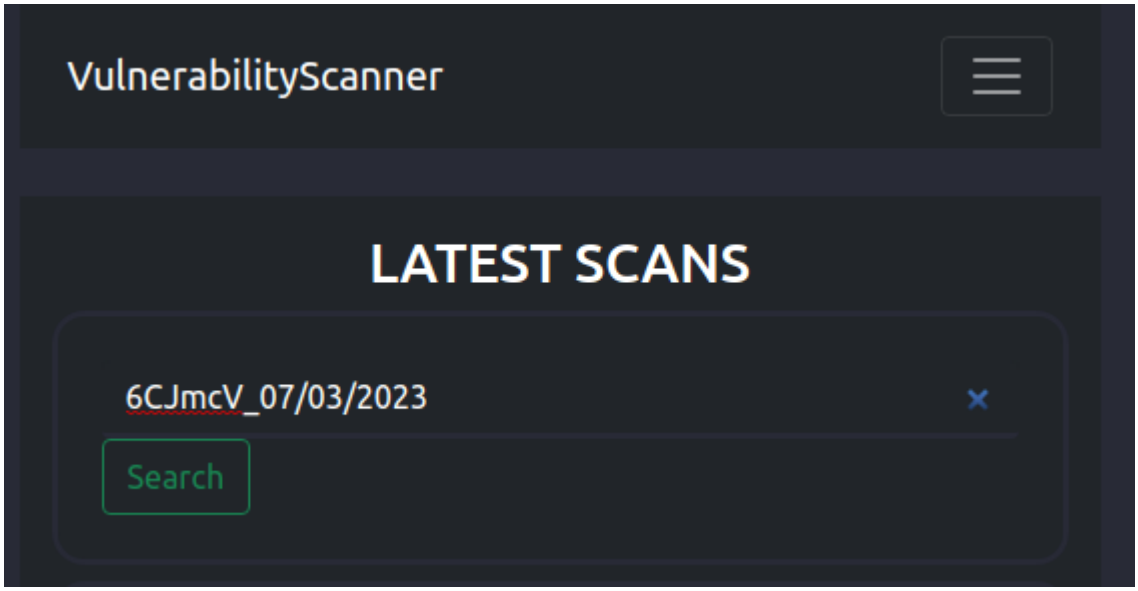
Εικόνα 34. Ο χρήστης πατάει το στοιχείο μενού SCANS



Εικόνα 35. Φόρμα αναζήτησης αναλύσεων / σαρώσεων

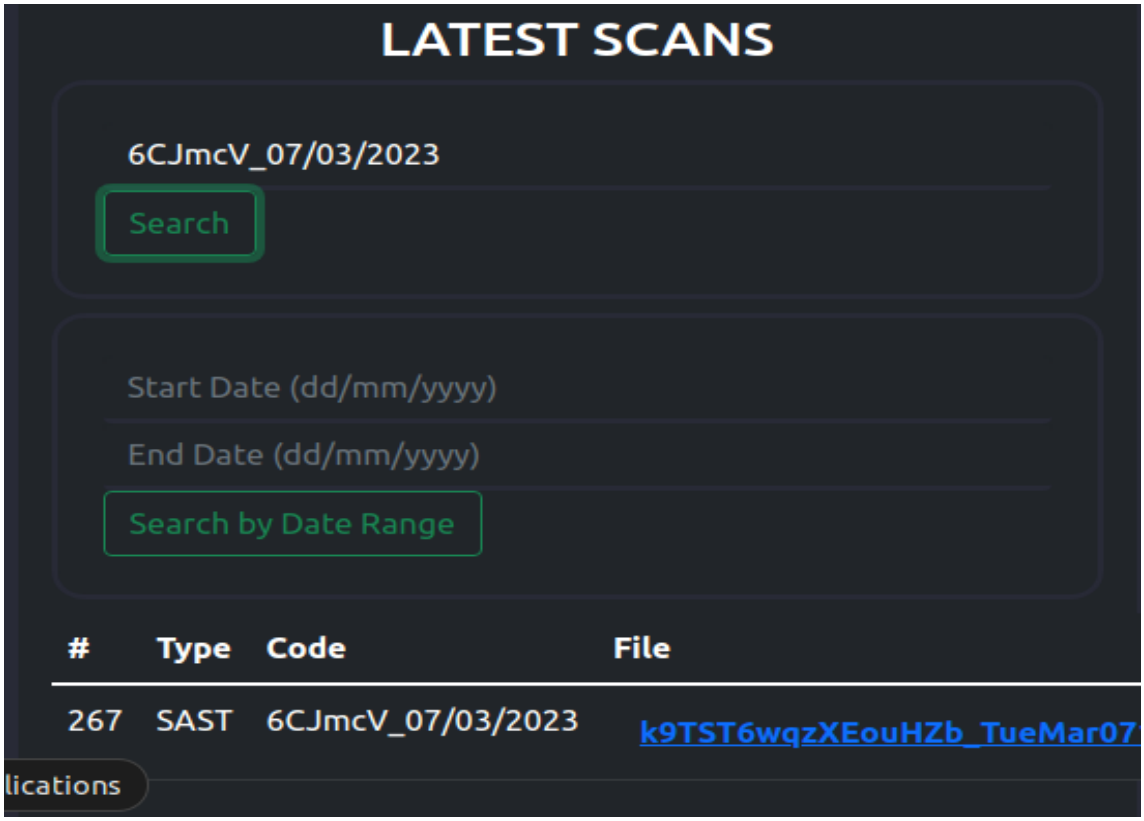
#	Type	Code	File
4002	COMBINED	ZbZwzA_11/10/2023	aTJWzbr3GR2Scjwc_WedOct11233
3952	COMBINED	sZIyXc_11/10/2023	MNNEqrb9x43kQOyn_WedOct1123
3902	COMBINED	KgyJBp_11/10/2023	VqZsM9ylE8V4JUxq_WedOct11231
3852	SAST	55vijc_11/10/2023	2gxlypSOpNWcqxRH_WedOct1122
3802	DAST	SrAyqb_11/10/2023	7OAQvXJZ8hvF9xta_WedOct11225
3752	DAST	WkR70P_11/10/2023	uYtmCJ21gWPHYBKF_WedOct112
3702	DAST	3WWuXF_11/10/2023	T1xcsJj8Wf1v9AsB_WedOct112229

Εικόνα 36. Πίνακας με τις τελευταίες σαρώσεις

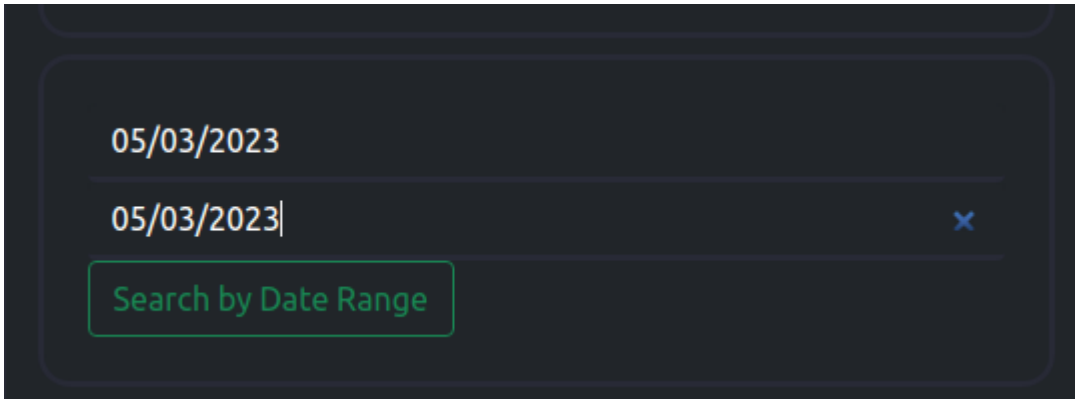


Εικόνα 37. Σενάριο χρήσης αναζήτησης με βάση τον κωδικό αναγνωριστικού σάρωση

Εφόσον ο χρήστης αναζητήσει μια σάρωση μέσω της χρήσης του κωδικού της (δείτε Εικόνα 37), τότε θα του εμφανιστεί το αντίστοιχο αποτέλεσμα προς εκφόρτωση στο κάτω μέρος της σελίδας (δείτε Εικόνα 38).

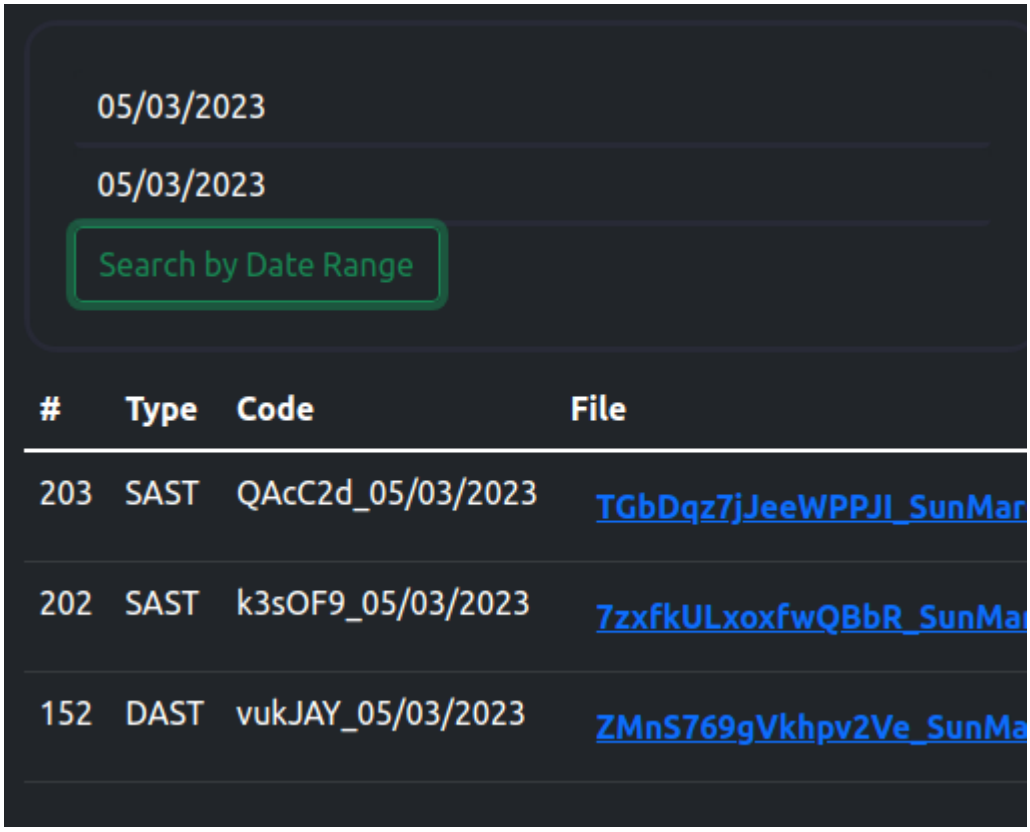


Εικόνα 38. Η αναφορά της σάρωσης που αναζητήθηκε



Εικόνα 39. Σενάριο χρήσης της αναζήτησης με εύρος ημερομηνιών (σε μορφή dd/MM/yyyy)

Εφόσον ο χρήστης ψάξει με ένα εύρος ημερομηνιών (δείτε Εικόνα 39), θα του εμφανιστεί από κάτω πίνακας με τις σαρώσεις που ταιριάζουν σε αυτό το εύρος (ως προς την ημερομηνία διεξαγωγής τους) (δείτε Εικόνα 40).



Εικόνα 40. Οι αναφορές σάρωσης για το συγκεκριμένο εύρος ημερομηνιών

6.5 Αποτίμηση της Εφαρμογής

Με βάση τα ευρήματά μας ανά σενάριο χρήσης (ειδικότερα στα σενάρια χρήσης μόνο δυναμικών εργαλείων ανάλυσης και μόνο στατικών εργαλείων ανάλυσης) που αφορούσε την ανάλυση της εσκεμμένα τρωτής εφαρμογής DVWA, τις τρωτότητες τις οποίες γνωρίζουμε εκ των προτέρων, καταλήγουμε στα εξής αποτελέσματα ως προς την ακρίβεια των εργαλείων (είτε μεμονωμένα είτε συνδυαστικά μέσω της χρήσης της εφαρμογής μας):

Για το Arachni, έχουμε 4 TP (True Positives), 1 FP (False Positive), 6 FN (False Negatives). Συνεπώς, έχουμε Recall: 4/(4+6) = 0.4 και Precision: 4/(4+1) = 0.8.

Για το Nikto, έχουμε 4 TP (True Positives), 0 FP (False Positives) , 6 FN (False Negatives). Συνεπώς, έχουμε Recall: 4/(4+6) = 0.4 και Precision: 4/(4+0) = 1.

Για το Insider, έχουμε 1 TP (True Positive) , 0 FP (False Positives) , 9 FN (False Negatives). Συνεπώς, έχουμε Recall: 1/(1+9) = 0.1 και Precision: 1/(1+0) = 1.

Για το NodeJsScan, έχουμε 3 TP (True Positives) , 1 FP (False Positive) , 7 FN (False Negatives). Συνεπώς, έχουμε: Recall 3/(3+7) = 0.3 και Precision: 3/(3+1) = 0.75.

Για τον συνδυασμό δυναμικών εργαλείων έχουμε Recall: 0.6 και Precision: 0.85.

Για τον συνδυασμό στατικών εργαλείων έχουμε Recall: 0.4 και Precision: 0.8.

Για τον συνδυασμό όλων των εργαλείων έχουμε Recall: 0.8 και Precision: 0.8.

Συνεπώς, καταλήγουμε στο συμπέρασμα, με βάση και τα παραπάνω αποτελέσματα, ότι ο συνδυασμός των δοκιμών DAST και SAST παρέχει καλύτερη και πιο ολοκληρωμένη κάλυψη για τον εντοπισμό τρωτών σημείων σε εφαρμογές Ιστού.

Επίσης, μπορούμε να πούμε με σιγουριά (με βάση τα αποτελέσματα αποτίμησης) πως έχουμε καλύτερο βαθμό ανάκτησης (recall) όταν έχουμε την χρήση πολλών εργαλείων του ίδιου είδους ανίχνευσης, και ακόμα καλύτερο εφόσον χρησιμοποιήσουμε και τα 2 είδη τεχνικών ανίχνευσης.

Επομένως, τα αποτελέσματα της αποτίμησης δικαιολογούν πλήρως την ανάγκη ύπαρξης του εργαλείου μας και ειδικότερα τη ικανότητα του για συνδυασμένη ανάπτυξη εργαλείων σάρωσης, τόσο πολλαπλών για το ίδιο είδος ανάλυσης, όσο και πολλαπλών για διαφορετικά είδη ανάλυσης.

ΚΕΦΑΛΑΙΟ 7 – ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΗ ΕΡΓΑΣΙΑ

7.1 Συμπεράσματα

Πρώτον, ο συνδυασμός των δοκιμών DAST και SAST (όπως είδαμε στο προηγούμενο κεφάλαιο) παρέχει καλύτερη και πιο ολοκληρωμένη κάλυψη για τον εντοπισμό τρωτών σημείων σε εφαρμογές Ιστού. Η δοκιμή DAST μπορεί να βοηθήσει στον εντοπισμό τρωτών σημείων που είναι ανιχνεύσιμες μόνο όταν εκτελείται η εφαρμογή, ενώ η δοκιμή SAST μπορεί να εντοπίσει ευπάθειες στον πηγαίο κώδικα πριν από την εκτέλεση της εφαρμογής.

Μπορούμε, επίσης, να πούμε με σιγουριά πως έχουμε καλύτερη ανάκτηση (recall) όταν έχουμε την χρήση πολλών εργαλείων του ίδιου είδους ανίχνευσης, και ακόμη καλύτερο αν χρησιμοποιήσουμε και τα δύο είδη τεχνικών ανίχνευσης (combined). Αυτό αποδεικνύεται από την αποτίμηση της εφαρμογής μας με μια εσκεμμένα τρωτή εφαρμογή, της οποίας οι τρωτότητες ήταν εκ τω προτέρων γνωστές.

Δεύτερον, με τη χρήση εργαλείων ανοιχτού κώδικα, οι επιχειρήσεις μπορούν να εξοικονομήσουν κόστος σε σύγκριση με τη χρήση εμπορικών εργαλείων για δοκιμές ασφαλείας. Τα εργαλεία ανοιχτού κώδικα είναι συνήθως ελεύθερα διαθέσιμα και συχνά έχουν ενεργές κοινότητες που συμβάλλουν στην ανάπτυξη τους, παρέχοντας ένα ευρύ φάσμα δυνατοτήτων και λειτουργιών αλλά παρέχουν και υποστήριξη στην χρήση και αποσφαλμάτωσή τους. Ευελπιστούμε πως το ίδιο θα πραγματοποιηθεί και στην δική μας περίπτωση διότι και το δικό μας σύνθετο εργαλείο σάρωσης είναι ανοικτού κώδικα.

Τρίτον, η χρήση μιας διαδικτυακής εφαρμογής Java Spring Boot για δοκιμές ευπάθειας μπορεί να βοηθήσει τις επιχειρήσεις να αξιοποιήσουν τα οφέλη του Spring Boot, όπως η ευκολία χρήσης, η ευελιξία και η συμβατότητά του με ένα ευρύ φάσμα τεχνολογιών.

Συνολικά, ο συνδυασμός εργαλείων ανοιχτού κώδικα DAST και SAST σε μια εφαρμογή web Java Spring Boot μπορεί να προσφέρει στις επιχειρήσεις μια ολοκληρωμένη και οικονομικά αποδοτική λύση για τον εντοπισμό τρωτών σημείων στις διαδικτυακές εφαρμογές τους. Με τον προληπτικό εντοπισμό και την αντιμετώπιση αυτών των τρωτών σημείων, οι επιχειρήσεις μπορούν να βελτιώσουν την ασφάλεια των εφαρμογών τους και να μειώσουν τον κίνδυνο παραβίασης δεδομένων, κάτι που μπορεί τελικά να συμβάλει στην προστασία της φήμης και της τελικής τους αξίας.

7.2 Μελλοντική Εργασία

Ως μελλοντική εργασία, θα μπορούσαμε να εμπλουτίσουμε την ήδη υπάρχουσα εφαρμογή και να την επεκτείνουμε, με τα εξής σημεία.

- Ενσωμάτωση περισσότερων εργαλείων σάρωσης τρωτοτήτων ανοικτού κώδικα, SAST και DAST, για ακόμη μεγαλύτερη ακρίβεια. Η εφαρμογή είναι δομημένη με τρόπο όπου η εισαγωγή επιπλέον εργαλείων σάρωσης στην ήδη υπάρχουσα υλοποίηση είναι αρκετά εύκολη. Παρ'όλα αυτά, θα πρέπει να εντοπιστούν και να επιλεγούν τα κατάλληλα εργαλεία σάρωσης προς ενσωμάτωση (π.χ. OWASP Zap[24]).
- Για την στατική σάρωση, μπορούμε, εκτός από μια μεταφόρτωση .zip αρχείου που περιλαμβάνει τον πηγαίο κώδικα προς σάρωση, να προσθέσουμε και μια επιπλέον εναλλακτική εισόδου όπου ο χρήστης θα μπορεί να εισάγει κάποιο Github URL ενός αποθετηρίου όπου εκεί βρίσκεται ο πηγαίος κώδικας προς σάρωση.
- Και για τα δύο είδη σάρωσης, θα μπορεί να πραγματοποιείται ενοποίηση σε ένα κοινό πληροφοριακό μοντέλο, το οποίο θα καλύπτει τα διάφορα (κοινά ή μη ως προς την σημασιολογία) μέρη των αναφορών. Το μοντέλο αυτό μπορεί να βασίζεται και σε μια κοινή βάση αναφοράς τρωτοτήτων[25], η οποία μπορεί να περιέχει επιπρόσθετη πληροφορία σε σχέση με αυτή που παράγεται από το κάθε εργαλείο για την κάθε ευπάθεια. Με αυτό τον τρόπο, είναι δυνατή η παραγωγή μιας μοναδικής αναφοράς ανά αίτηση σάρωσης που να προσδιορίζει με πιο πλήρη τρόπο τις ευπάθειες που έχουν εντοπιστεί συνολικά από όλα τα χρησιμοποιούμενα εργαλεία σάρωσης.
- Η αναφορά θα μπορούσε να εμπλουτιστεί με την παροχή του συνολικού ρίσκου της εφαρμογής με βάση τις ευπάθειες που έχουν εντοπιστεί για αυτήν και άρα την κρισιμότητα και πιθανότητα εκμετάλλευσής τους. Σε αυτή την περίπτωση, θα πρέπει να αναπτυχθεί ένας κατάλληλος αλγόριθμος υπολογισμού ρίσκου με βάση την διαθέσιμη πληροφορία για την κάθε εντοπισμένη ευπάθεια.
- Μεταφόρτωση του κώδικα της εφαρμογής μας στο GitHub προς χρήση και εκμετάλλευση από την ευρεία κοινότητα των προγραμματιστών.

[24] https://www.zaproxy.org/
[25] https://security.snyk.io/, https://vuldb.com/?kb.api, https://www.whitesourcesoftware.com/vulnerability-database

ΚΕΦΑΛΑΙΟ 8 – ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] OWASP Foundation (n.d.). *OWASP API Security Top Ten – 2023*. https://owasp.org/API-Security/editions/2023/en/0x00-header/.

[2] Risto, J. (2023, May 22). What is Common Vulnerability Scoring System (CVSS). *SANS Blog*. https://www.sans.org/blog/what-is-cvss/.

[3] Evidently AI (n.d.). *Accuracy, precision, and recall in multi-class classification*. https://www.evidentlyai.com/classification-metrics/multi-class-metrics.

[4] Synopsys. *What is SAST*. https://www.synopsys.com/glossary/what-is-sast.html.

[5] Snyk (n.d.). *Dynamic Application Security Testing (DAST)*. https://snyk.io/learn/application-security/dast-dynamic-application-security-testing/.

[6] O'Driscoll, A. (2023, June 14). Cybersecurity vulnerability statistics and facts of 2023. *Comparitech Blog: Information Security*. https://www.comparitech.com/blog/information-security/cybersecurity-vulnerability-statistics/.

[7] Sverdlov, E. (2012, June 12). *How To Create a New User and Grant Permissions in MySQL*. Digital Ocean. https://www.digitalocean.com/community/tutorials/how-to-create-a-new-user-and-grant-permissions-in-mysql.

[8] Winterfeld, S., Akamai Security Research (2023, April 18). Slipping Through the Security Gaps: The Rise of Application and API Attacks. Akamai Blog: Security. https://www.akamai.com/blog/security/the-rise-of-application-and-api-attacks.

[9] Balbix (n.d.). *What is Vulberability Scanning*. https://www.balbix.com/insights/what-is-vulnerability-scanning/.

[10] Scarfone, K., Souppaya, M., Cody, A., Orebaugh, A. (2008). *Technical Guide to Information Security Testing and Assessment* (NIST Special Publication 800-115). NIST. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf.

[11] Kuszczyński K, Walkowski M. Comparative Analysis of Open-Source Tools for Conducting Static Code Analysis. *Sensors*. 2023; 23(18):7978. https://doi.org/10.3390/s23187978

[12] Baek, Y. M. (2014), Solving the privacy paradox: A counter-argument experimental approach. *Computers in Human Behavior*, **38**, 33-42.

[13] Mateo Tudela F, Bermejo Higuera J-R, Bermejo Higuera J, Sicilia Montalvo J-A, Argyros MI. On Combining Static, Dynamic and Interactive Analysis Security Testing Tools to Improve OWASP Top Ten Security Vulnerability Detection in Web Applications. *Applied Sciences*. 2020; 10(24):9119. https://doi.org/10.3390/app10249119.

[14] W. Qianqian and L. Xiangjun, "Research and design on Web application vulnerability scanning service," *2014 IEEE 5th International Conference on Software Engineering and Service Science*, Beijing, China, 2014, pp. 671-674, doi: 10.1109/ICSESS.2014.6933657.

[15] Bairwa, Sheetal & Mewara, Bhawna & Gajrani, Jyoti. (2014). Vulnerability Scanners-A Proactive Approach to Assess Web Application Security. *International Journal on Computational Science & Applications*. 4. 10.5121/ijcsa.2014.4111.

[16] H. Chen, J. Chen, J. Chen, S. Yin, Y. Wu and J. Xu, "An Automatic Vulnerability Scanner for Web Applications," *2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, Guangzhou, China, 2020, pp. 1519-1524, doi: 10.1109/TrustCom50675.2020.00207.

[17] Kidd, C. (2023, January 11). Is The CIA Triad Relevant? Confidentiality, Integrity & Availability Today. *Splunk Blog*. https://www.splunk.com/en_us/blog/learn/cia-triad-confidentiality-integrity-availability.html.

[18] Sucuri (n.d.). *How To Remove Google Blocklist Warning*. https://sucuri.net/guides/how-to-remove-google-blocklist-warning/.

[19] CrowdStrike (2022, October 14). *Vulnerability Management Lifecycle*. https://www.crowdstrike.com/cybersecurity-101/vulnerability-management/vulnerability-management-lifecycle/.

[20] OWASP (n.d.). *Review Old Backup and Unreferenced Files for Sensitive Information*. https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/04-Review_Old_Backup_and_Unreferenced_Files_for_Sensitive_Information.

ΠΑΡΑΡΤΗΜΑ Α – Χρήσιμοι Σύνδεσμοι

- Apache (2023, December 8). *Official Apache Maven Documentation*. https://maven.apache.org/guides/index.html.
- Nicoll, S., Wilkinson, A., Frederick, S. (n.d.). *Spring Boot Maven Plugin Documentation*. Spring. https://docs.spring.io/spring-boot/docs/3.0.1/maven-plugin/reference/html/.
- Spring (n.d.). *Producing a SOAP web service*. https://spring.io/guides/gs/producing-web-service/.
- Spring (n.d.). *Handling Form Submission*. https://spring.io/guides/gs/handling-form-submission/.
- Spring (n.d.). *Building a RESTful Web Service*. https://spring.io/guides/gs/rest-service/.
- Spring (n.d.). *Serving Web Content with Spring MVC*. https://spring.io/guides/gs/serving-web-content/.
- Spring (n.d.). *Building REST services with Spring*. https://spring.io/guides/tutorials/rest/.
- Spring (n.d.). *Accessing JPA Data with REST*. https://spring.io/guides/gs/accessing-data-rest/.
- Spring (n.d.). *Accessing Neo4j Data with REST*. https://spring.io/guides/gs/accessing-neo4j-data-rest/.
- Spring (n.d.). *Accessing MongoDB Data with REST*. https://spring.io/guides/gs/accessing-mongodb-data-rest/.
- Spring (n.d.). *Accessing Relational Data using JDBC with Spring*. https://spring.io/guides/gs/relational-data-access/.
- Spring (n.d.). *Managing Transactions*. https://spring.io/guides/gs/managing-transactions/.
- Arachni (n.d.). *Arachni Documentation*. https://github.com/Arachni/arachni/wiki.
- Arachni (n.d.). *Arachni Code Documentation*. https://rubydoc.info/github/Arachni/arachni.
- Laskos, T. (2011, May 26). *Web application testing with Arachni*. INFOSEC. https://resources.infosecinstitute.com/topics/application-security/web-application-testing-with-arachni/.