Vulnerability Scanner

vulnerability scanner, εφαρμογή σε πειραματικο web server

Παπαδόπουλος Παντελεήμον Νεκτάριος it174891  
*Διεθνες Πανεπιστιμιο Ελλάδος*  
Θεσσαλονίκη Grecce  
 pantpap15@gmail.com

*Περίληψη* — Ουσιαστικά έτρεξα το owasp zap μια από της πιο γνωστές vulnerability scanner εφαρμογές, σε έναν πειραματικό web server που έστησα στο σύστημα μου , αναλύω πως έστησα όλο το περιβάλλον, τη βρήκα και με πιο τρόπο.

Keywords — Vulnerability scanner, owasp, zap, cyber security, server, web server, web application

# εισαγωγη

Η περιοχή της έρευνας είναι το testing των προσωπικών μας web εφαρμογών μέσο κάποιου γνωστού Vulnerability Scanner. Οι Vulnerability Scanners είναι αυτοματοποιημένα εργαλεία που σαρώνουν εφαρμογές ιστού, για να αναζητήσουν ευπάθειες ασφαλείας, όπως Cross-site scripting, SQL Injection, Command Injection, Path Traversal and insecure server configuration. Η σημασία αυτού του πειράματος είναι αρκετά μεγάλη καθώς μας παρουσιάζει τα προβλήματα που μπορεί να έχει μια web εφαρμογή που μπορεί να σχεδιάζουμε καιρώ και μας επισημάνει της σοβαρές ευπάθειες που μπορεί να έχουν, αλλά και υποδείξεις για το πως να της εξαλείψουμε. Παρακάτω αναλύω το πως έστησα των web server ,τι vulnerability scanner χρησιμοποίησα ,τι ευπάθειες βρήκα και πως της βρήκα.

# προσωπικες επιλογες

## Επιλογή Scanner (Πρόγγραμα)

Για την Scanner εφαρμογή επέλεξα το owasp zap καθώς μετά από έρευνα είδα ότι εκτός από open source πρόγραμμα που δέχεται συνεχώς αναβαθμίσεις είναι και αρκετά αποτελεσματικό για της περισσότερες περιπτώσεις και είναι αρκετά εύκολο στην χρήση.

## Επιλογή Web server (Πειραματικη εφαρμογη)

Για πειραματικό web server διάλεξα το owasp juice shop καθώς είναι αρκετό για το πείραμα που ήθελα να δείξω και επίτηδες έχει αρκετές ευπάθειες επίτηδες για εκπαιδευτικό σκοπό .

# Παρουσιαση περιβαλλοντος

Αρχικά το πειραματικό μου περιβάλλον είναι σε linux mint.Οποιοδήποτε λειτουργικό σύστημα linux θα μπορούσε να εγκαταστήσει το zap αλλά για λόγους προσωπικής προτίμησης διάλεξα τα mint. Η εγκατάσταση του zap ήταν εύκολη και γρήγορη καθώς έγινε με script εγκατάστασης έτοιμο από τον owasp.

Έπειτα για να μπορέσει να τρέξει το juice shop έπρεπε να εγκαταστήσω τον npm καθώς το juice shop είναι γραμμένο σε node.js. Δεν χρησιμοποίησα κάποιον άλλον manager καθώς ο npm είναι ο πιο διαδεδομένος και ο πιο λειτουργικός αυτόν τον καιρό

## Μετά την εγκατάσταση

Μετά την εγκατάσταση όλων των βασικών προγραμμάτων που χρειαζόμουνα ήρθε η στιγμή για το πείραμα το οποίο έγινε ως εξής, άνοιξα τον server (npm start) το οποίο ανοίγει έναν local server στο σύστημα και σελίδα το juice shop , και έπειτα ξεκίνησα το zap για το scann .

## Zap Prosedure(Automate Scann)

Αρχικά επιλέγουμε στόχο βάζοντας το url στο συγκεκριμένο box . Στην περίπτωση αυτή βάζουμε το http://localhost καθώς ο server είναι πειραματικός και τρέχει τοπικά στο σύστημα μας και έπειτα κάνουμε κλικ στο Attack.

Μετά την εκκίνηση της αυτοματοποιημένης διαδικασίας το zap ξεκινάει την διαδικασία να βρε ευπάθειες στην web εφαρμογή μας που στην περίπτωση αυτήν είναι το juice shop του owasp που το βάλαμε να τρέχει στο localhost . Εδώ πρέπει να αναφερθεί πως το αυτόματο scan είναι η πιο απλή και γρήγορη διαδικασία εύρεσης ευπαθειών στην εφαρμογή μας καθώς ουσιαστικά γίνετε μα ένα κλικ , αλλά δεν είναι τόσο αποτελεσματικό όσο η χειροκίνητη λειτουργία.

## Zap Prosedure(Manual Scann)

Το zap έχει και χειροκίνητη λειτουργία. Με τον ίδιο τρόπο θέτουμε στο box το επιθυμητό λινκ , στην δικιά μας περίπτωση πάλι βάζουμε το localhost . Με το που πατήσουμε attack ανοίγει ένα παράθυρο firefox το οποίο είναι υπό την κατοχή του zap , αυτό το καταφέρνει με selenium web driver, το οποίο είναι ένα framework για testing σε web εφαρμογές και για υλοποίηση διάφορων αυτοματοποιήσεων μέσω του browser.

Με το που άνοιξη το παράθυρο με τον browser μας παρουσιάζει ένα μενού με πολλές επιλογές και ενδείξεις (alerts), έπειτα στο παρασκήνιο βρίσκετε η web εφαρμογή που τρέχουμε το τεστ . Από εδώ και πέρα διαλέγουμε εμείς τι ακριβός δοκιμές θα κάνουμε στην εφαρμογή μας . Εδώ να αναφέρουμε ότι το manual scan έχει λειτουργίες που δεν υπάρχουν στο automate scan όπως : ajax spider , attack mode , κα . Τα οποία σαν σύνολο και σε συνδυασμό με το αυτόματο σκαν μπορούν να δώσουν περισσότερα αποτελέσματα και πιο σίγουρα.

Note : Εδώ να σημειώσουμε ότι οποιοδήποτε τεστ τρέχουμε γίνετε πρέπει να γίνετε σε δικό μας πειραματικό server η σε σύστημα στο οποίο έχουμε την ευθύνη η μας έχουνε δώσει την άδεια να το κάνουμε. Έπειτα τα αποτελέσματα χωρίζονται σε High Risk , τα οποία είναι τα πιο σημαντικά και έχουν άμεση προτεραιότητα, μετά έχουμε τα Medium Risk τα οποία είναι μεσαίας προτεραιότητας προβλήματα αλλά σε καμία περίπτωση απλά η ανώδυνα και τέλος έχουμε τα Informative – Low Risk που όπως λέει και η λέξη είναι πιο πολύ αναφορικά και όχι υψηλής προτεραιότητας .

## Αποτελεσματα Scan

* Cross-Domain Misconfiguration (High Risk). Αρχικά τι είναι CORS. CORS στημένη Cross-Origin Resource Sharing και αυτό μας προσφέρει την δυνατότητα να μπορούμε να εκθέσουμε πόρους σε κάποιο συγκεκριμένο Domain, και επίσης ένας client να μπορεί να υποβάλει αιτήματα ajax για κάποιον πόρο σε τρίτο domain και όχι στον κύριο. Όταν λοιπόν αυτό το σύστημα δεν λειτουργεί σωστά μπορεί ο server μας να τρέξει ένα request από κάποιο τρίτο κακόβουλο.
* Session ID in URL Rewrite (High Risk). Το URL rewrite χρησιμοποιείτε για να εντοπίζει του κάθε χρήστη το session id. Καταλαβαίνουμε ότι είναι κακό καθώς αυτό το URL μπορεί να αποθηκευτή και να ξαναχρησιμοποιηθεί από τον χρήστη η από άλλους χρήστες ενώ δεν πρέπει.
* Cross-Domain JavaScript Source File Inclusion (Medium Risk). Αυτό σημαίνει ότι η σελίδα η το web Application τρέχει - περιέχει scripts από τρίτο domain. Δεν έχουμε κανέναν έλεγχο επί του τι υπάρχει σε αυτό το κομμάτι κώδικα και δεν έχουμε κανένα έλεγχο της ασφάλειας του διακομιστή στον οποίο φιλοξενείται .
* Incomplete or No Cache-control and Pragma HTTP Header (Medium Risk). Η κεφαλίδα cache-control και pragma http δεν έχει ρυθμιστεί σωστά η λείπει , επιτρέποντας στο πρόγραμμα περιήγησης και τους διακομιστές μεσολάβησης να αποθηκεύουν προσωρινά το περιεχόμενο.
* Private IP Disclosure (Medium Risk). Αυτή είναι μια αρκετά σημαντική ευπάθεια καθώς μια ιδιωτική διεύθυνση IP (172.x.x.x ή 10.x.x.x) η ένα ιδιωτικό όνομα κεντρικού υπολογιστή έχει βρεθεί στο σώμα απόκρισης HTTP. Αυτές οι πληροφορίες καταλαβαίνουμε ότι αν πέσουν στα λάθος χέρια προφανώς και θα οδηγήσουν σε περαιτέρω επιθέσεις που στοχεύον εσωτερικά συστήματα.
* X-Content-type-options header missing (Medium Risk). Η κεφαλίδα Anti-MIME-Sniffing X-Content-type-options δεν έχει οριστεί σε <<nosniff>>. Αυτό επιτρέπει σε παλαιότερες εκδόσεις του chrome και του explorer να εκτελούν mime-sniffing στο σώμα απόκρισης, προκαλώντας δυνητικά το σώμα απόκρισης να ερμηνευτεί και να εμφανιστεί ως τύπος περιεχομένου διαφορετικός από τον δηλωμένο τύπο περιεχομένου. Αυτό δεν συμβαίνει από της εκδόσεις του 2014 και μετά στον Firefox
* Information Disclosure Suspicious Comments (Informative – Low Risk). Είναι comments που μπορεί κάποιοι χρήστες να προσπάθησαν να εξάγουν χρήσιμες πληροφορίες κυρίως με xml
* Loosely Scoped Cookie (Informative – Low Risk). Τα cookies μπορούν να εντοπιστούν ανά domain η διαδρομή. Αυτός ο έλεγχος αφορά μόνο το πεδίο εφαρμογής. Το πεδίο εφαρμογής του domain που εφαρμόζεται σε ένα cookie καθορίζει ποιοι domains μπορούν να έχουν πρόσβαση σε αυτό. Σε αυτήν την περίπτωση οποιασδήποτε subdomain μπορεί να έχει πρόσβαση στην εφαρμογή μας καθώς έχουμε loosely scoped cookies.Αυτή η περίπτωση είναι πολύ κοινή σε μεγάλα domains όπως google
* Timestamp Disclosure (Informative – Low Risk). Το timestamp χρησιμοποιείται για μια ποικιλία σκοπών κύριος συγχρονισμού όπως μιας σειράς ακολουθίας για μια συναλλαγή. Το συγκεκριμένο απλά μας ενημερώνει απλά καθώς η ύπαρξη χρονικής σήμανσης δεν αντιπροσωπεύει απαραίτητα κανένα σοβαρό κίνδυνο για την εφαρμογή.

## False Positives

Ένα false positive είναι σαν ένα ψευδές συναγερμό. ενεργοποιείται ο συναγερμός του σπιτιού σας και δεν υπάρχει διαρρήκτης. Στην ασφάλεια εφαρμογών ιστού, ένα false positive είναι όταν ένας vulnerability scanner υποδεικνύει ότι υπάρχει ευπάθεια στον ιστότοπό σας, όπως το SQL Injection, αλλά στην πραγματικότητα, δεν υπάρχει.

Οι vulnerability scanners είναι γνωστό ότι αναφέρουν false positive αποτελέσματα, επομένως ένα penetration testing εφαρμογών ιστού καταναλώνει σημαντικό χρονικό διάστημα επειδή οι penetration testers πρέπει να περάσουν από όλες τις αναφερόμενες ευπάθειες και να τις επαληθεύσουν χειροκίνητα. Λόγω αυτής της μακράς διαδικασίας, η ασφάλεια εφαρμογών ιστού δεν είναι προσιτή για πολλές επιχειρήσεις. Τέλος η πιο παραγωγική και οικονομικά αποδοτική λύση ασφάλειας εφαρμογών ιστού είναι ένα vulnerability scanner με τεχνολογία Proof-Based Scanning . Ο scanner μπορεί να επαληθεύσει αυτόματα τα ευρήματά του εκμεταλλευόμενος τις αναγνωρισμένες ευπάθειες και να παρουσιάσει στον χρήστη ένα proof of exploitation. Τα οφέλη από την ύπαρξη ενός τέτοιου σαρωτή είναι πολλαπλά. Οι δοκιμές ασφαλείας θα καταναλώσουν πολύ λιγότερο χρόνο και οι υπάλληλοί δεν χρειάζεται να έχουν χρόνια εμπειρίας στο hacking για να επαληθεύσουν τα αποτελέσματα. Το πιο πιθανό είναι ότι τα τελευταία τρία αποτελέσματα είναι false positive καθώς δεν χρήζουν άμεσης ανάγκης και είναι πολύ απλά να ξεπεραστούν.

# Εργαλεια και λειτουρια ZAP

Όλα τα παραπάνω όπως προ είπα είναι ευρήματα από μια μίξη automated scan και manual scan , παρακάτω θα αναλύσω λίγο παραπάνω τα εργαλεία τα οποία χρησιμοποιεί το zap για να βρει αυτές της ευπάθειες στην εφαρμογή μας . Τα παρακάτω είναι εργαλεία που ουσιαστικά χρησιμοποιεί το λεγόμενο active scan για να βρει όλες της αναφορές της ιστοσελίδας και να δημιουργήσει ένα χάρτη της web εφαρμογής. Επιπρόσθετος να πω ότι στο zap υπάρχουν τέσσερα modes (Safe – Δεν επιτρέπονται δυνητικά επικίνδυνες λειτουργίες , Protected – Μπορούμε να εκτελέσουμε μόνο επικίνδυνες ενέργειες σε URL στο πεδίο εφαρμογής ή Scope , Standard – Μπορούμε να κάνουμε το οτιδήποτε , ATTACK—Νέοι κομβόι που βρίσκονται στο πεδίο εφαρμογής σαρώνονται ενεργά μόλις ανακαλυφθούν ) . Εγώ έτρεξα το πείραμα μου χρησιμοποιώντας το Protected όπως και συνίσταται από τον owasp.

## Spider

#### Ένα crawler η αλλιως spider ειναι ένα εργαλείο ανήχνευσης που αυτοματα περνα μεσο ενός ιστοτοπου ακολουθώντας ολους τους συνδέσμους , και σε μερικες φορες γεμιζει και στελνει ολόκληρες φόρμες, αυτό μας επιρτέπει να πάρουμε έναν πλήρη χάρτη όλων των σελίδων που αναφέρονται στον ιστότοπο και να καταγράψουμε τα αιτήματα που υποβλήθηκαν για να λάβουμε αυτές και τεις απαντήσεις τους (responses). Αυτο στην περήπτωση του προγράμματος μας λειτουργεί παρόμεια και μας επιτρτεπει να παρουμε ένα πλήρη χάρτη όλων των σελιδών της εφαρμογής μας και των ευπαθειών που φαίνονται επιφανειακά.

## Ajax Spider και Active scan

Το πρόσθετο AJAX Spider ενσωματώνεται στο zap. Αυτός ο crawler εξειδικεύετε στο να βρίσκει ευπάθειες σε web εφαρμογές που χρησιμοποιούν AJAX. Εάν η εφαρμογή που τεστάρουμε χρησιμοποιεί Ajax και δεν χρησιμοποιήσουμε αυτό το πρόσθετο δεν θα βρούμε αρκετές σημαντικές ευπάθειες. Το AJAX spider είναι αρκετά πιο αργό από το κλασικό spider και επίσης χρειάζεται έναν εξωτερικό browser για να λειτουργήσει και να ψάξει τον ισότοπο, για αυτό τον λόγο και χρειαζόμαστε το selenium (όπως έχω προαναφερθεί) και έναν driver για όποιον browser επιλέξουμε εξαρχής ,καθώς θα τρέξει από εκεί μέσα. Το spider αυτό παράλληλα με το απλό spider μας επιτρέπει να βρούμε ευπάθειες σε πολύ μεγαλύτερο βάθος από το να χρησιμοποιούσαμε μονάχα το απλό.

Έπειτα μόλις ολοκληρωθεί η ανίχνευση του Ajax spider ή του απλού spider ξεκινάει η λεγομένη ενεργή σάρωση (active scan).To zap θα ξεκινήσει μια ποικιλία σεναρίων επίθεσης στις διευθύνσεις URL που αναφέρονται στην καρτέλα του spider.Η πρόοδος της επίθεσης θα εμφανιστεί στην καρτέλα active scan.Μόλις ολοκληρωθεί η ενεργή σάρωση τα αποτελέσματα θα εμφανιστούν στην καρτέλα Alerts.Αυτό θα περιέχει όλα τα ζητήματα ασφάλειας που θα έχει εντοπίσει το zap και στην συγκεκριμένη εφαρμογή τα ανέφερα πιο πάνω.

Τέλος όταν πάμε στην κάθε μια ευπάθεια ξεχωριστά στην καρτέλα alerts το πρόγραμμα μπορεί να μας δείξει ακριβός το σημείο στον κώδικα που είναι η ευπάθεια και να μας προτείνει κιόλας τρόπους αντιμετώπισης , οι οποίοι φυσικά δεν είναι συγκεκριμένοι αλλά γενικοί .

#### V. ΜΕΛΛΟΝΤΙΚΕΣ ΕΡΓΑΣΙΕΣ

#### Μια από τεις ιδεες μου για μελοντικες εργασίες είναι να φτιαξω κατι τελειως δικο μου και να το τεσταρω με το zap. Η ιδεα μου εινια να εινια μια ολοκλιρομενη εφαρμογη με βαση δεδομενων στην οποια βασικος μου στοχος εινια να την κανω οσο το δηνατον πιο σοστη σε θεμα ασφαλειας σε σχεση με αυτά που εμαθα από αυτην την εργασια

Ο στόχος μου είναι ένα παιχνίδι uno το οποίο για τής κάρτες να χρησιμοποιεί sql βάση δεδομένων και να έχει για κάθε ένα παίκτη login page. Θα πρέπει λοιπόν από πλευρά ασφάλειας να μην μπορεί να έχει κανένας πρόσβαση με οποιοδήποτε τρόπο στην βάση αυτήν και να μην μπορεί να παρακάμψει το login page υποκλέπτοντας κάποιον κωδικό με sniffing

##### Συμπερασματα

Από όλη αυτήν την εργασία καταλαβαίνουμε πόσο σημαντική στην δημιουργία μιας εφαρμογής είναι η εύρεση των ευπαθειών από ένα τέτοιο πρόγραμμα .Η έλλειψη γνώσης των ευπαθειών σε μια μεγάλη εφαρμογή όπως το juice shop μπορεί να είναι μοιραία και να χρειαστεί πολύ λίγο χρονικό διάστημα ώστε ένας κακόβουλος να επιτεθεί στην εφαρμογή μας. Προφανώς το juice shop τα έχει επίτηδες έτσι ώστε να μας δήξη πόσο σημαντικό είναι ένα τέτοιο software με έμμεσο τρόπο. Έπειτα θα πρέπει ,όταν κάνουμε μια τέτοια ανάλυση στην εφαρμογή μας να έχουμε υπόψιν μας την εμφάνιση false positive αποτελεσμάτων και να τα αντιμετωπίσουμε με τους προαναφερθέντες τρόπους .Τέλος η χρήση του zap συγκεκριμένα δεν είναι απαραίτητη υπάρχουν πολλά αλλά καλά λογισμικά για τέτοιου είδους διεργασίες που μπορούν να μας φανούν πολύ χρήσιμα έως απαραίτητα , αν θέλουμε να κάνουμε την εφαρμογή μας διαθέσιμη στο ευρύ κοινό.

##### Βιβλιογραφια-Αναφορες

Παρακάτω θα βρείτε αναφορές σε άρθρα κύριος του OWASP για όλα τα αποτελέσματα του scan του οποίου έκανα , από την επίσημη βιβλιογραφία του λογισμικού [1] έως [13]. Έπειτα θα βρείτε άρθρα που χρησιμοποίησα για να μάθω και να κατανοήσω της λειτουργίες του zap οπός το spider , ajax spider , active scan και γενικότερα την στρατηγική την οποία χρησιμοποιεί εκτελώντας τα spider και μετέπειτα την λειτουργία του [14] έως [18] και τέλος τα links για το περιβάλλον χρήση το οποίο είχα, όπως το juice shop το zap και τον npm τον manager των node modules [19] έως [21] και για τα false positives [22] έως [23].

1. https://owasp.org/www-community/attacks/CORS\_OriginHeaderScrutiny
2. https://www.zaproxy.org/docs/alerts/10098/
3. https://www.zaproxy.org/docs/alerts/3/
4. https://seclists.org/webappsec/2002/q4/111
5. https://owasp.org/www-community/attacks/Session\_fixation
6. https://www.zaproxy.org/docs/alerts/10017/
7. https://www.zaproxy.org/docs/alerts/10015/
8. https://cheatsheetseries.owasp.org/cheatsheets/Session\_Management\_Cheat\_Sheet.html#Web\_Content\_Caching
9. https://www.zaproxy.org/docs/alerts/2/
10. https://tools.ietf.org/html/rfc1918
11. https://www.zaproxy.org/docs/alerts/10021/
12. https://www.zaproxy.org/docs/desktop/addons/websockets/pscanrules/
13. https://www.zaproxy.org/docs/alerts/90033/
14. https://groups.google.com/forum/#!topic/zaproxy-develop/BAMfOXcu2hE
15. https://en.wikipedia.org/wiki/Spider
16. https://www.sarangjaiswal.com/ajax-spider-with-authentication-using-owasp-zap.html
17. https://www.zaproxy.org/docs/desktop/addons/ajax-spider/options/
18. https://www.zaproxy.org/docs/desktop/start/features/modes/
19. https://owasp.org/www-project-zap/
20. https://owasp.org/www-project-juice-shop/
21. https://docs.npmjs.com/
22. https://www.netsparker.com/blog/web-security/false-positives-web-application-security/
23. https://www.zaproxy.org/faq/how-do-i-handle-a-false-positive/