Πανεπιστήμιο Πειραιώς, Τμήμα Ψηφιακών Συστημάτων

Π.Μ.Σ. Ασφάλεια Ψηφιακών Συστημάτων

Ακαδημαϊκό έτος 2010-2011

Διπλωματική Εργασία

Δημιουργία Αυτοματοποιημένου Εργαλείου για Ελέγχους Ασφάλειας σε Διαδικτυακές Εφαρμογές

Επιβλέπων καθηγητής: Κωνσταντίνος Λαμπρινουδάκης
Σε συνεργασία με: Γενικό Επιτελείο Στρατού – ΚΕ.Π.Υ.Ε.Σ

Τριμελής επιτροπή: Κωνσταντίνος Λαμπρινουδάκης
Σωκράτης Κάτσικας
Χρήστος Ξενάκης

Σπουδαστής: Ζαχαριάδης Μιχάλης
Αριθμός μητρώου: MTE/1049
Email: zachmich@gmail.com
Ημερ. παράδοσης: 28-02-2012

ΠΡΟΛΟΓΟΣ

Η παρούσα διπλωματική εργασία ολοκληρώθηκε με τα επιθυμητά αποτελέσματα, σε ένα ενδιαφέρον γνωστικό αντικείμενο, όπως αυτό των ελέγχων ασφάλειας σε διαδικτυακές εφαρμογές. Υπήρξε ένα ευχάριστο κλίμα συνεργασίας με τον επιβλέποντα καθηγητή μου κύριο Κωνσταντίνο Λαμπρινουδάκη, τον οποίο θα ήθελα να ευχαριστήσω για την πολύτιμη βοήθεια που μου παρείχε καθ' όλη τη διάρκεια της διπλωματικής εργασίας.

Επιπλέον, η παρούσα διπλωματική εργασία έγινε σε συνεργασία με το ΚΕ.Π.Υ.Ε.Σ (Κέντρο Πληροφορικής Υποστήριξης Ελληνικού Στρατού). Θα ήθελα να ευχαριστήσω τους αρμόδιους του ΚΕ.Π.Υ.Ε.Σ που επέλεξαν να συνεργαστούν μαζί μου. Μέσω αυτής της συνεργασίας αποκόμισα ακόμα μεγαλύτερη εμπειρία στο συγκεκριμένο γνωστικό αντικείμενο και η βοήθεια τους ήταν πολύτιμη.

ΠΕΡΙΛΗΨΗ

Η διπλωματική εργασία με τίτλο «Δημιουργία Αυτοματοποιημένου Εργαλείου για Ελέγχους Ασφάλειας σε Διαδικτυακές Εφαρμογές» είναι μια πρακτική απεικόνιση για το πως μπορούν να αυτοματοποιηθούν χρονοβόρες διαδικασίες που μέχρι πρότινος πραγματοποιούσε ένας tester χειροκίνητα. Βασικός στόχος του εργαλείου αυτού, που θα το καθιστά διαφορετικό από τα υπόλοιπα εργαλεία αυτού του τύπου, είναι ο τρόπος με τον οποίο σχεδιάζεται ώστε να παρέχει ευελιξία, φορητότητα και ταχύτητα στους testers. Για το λόγο αυτό επιλέχθηκε μια δυναμική γλώσσα διαδικτυακού προγραμματισμού, η PHP. Ουσιαστικά, το εργαλείο αυτό θα μπορεί να ενεργοποιείται μέσω ενός web browser, ενώ το περιβάλλον εκτέλεσης του κώδικα της εφαρμογής θα είναι ένας web server ο οποίος θα αναλαμβάνει να εκτελεί τις διαδικασίες που θα ζητηθούν από μεριάς του client μέσω ενός εύχρηστου WUI (Web User Interface). Με αυτόν τον τρόπο, όλος ο φόρτος εργασίας των ελέγχων ασφάλειας μεταφέρεται στον web server και έτσι ακόμα και ένας υπολογιστής χαμηλών υπολογιστικών δυνατοτήτων θα μπορεί να εκτελεί τους ελέγχους με μεγάλη ευκολία.

Στόχος του εργαλείου αυτού είναι να συγκεντρώσει όσες περισσότερες επιθέσεις μπορούν να πραγματοποιηθούν σε web applications και όχι να επικεντρωθεί σε μια συγκεκριμένη επίθεση. Οι έλεγχοι ασφάλειας που καλύπτονται στην παρούσα εργασία είναι Information Disclosure, Cross Site Scripting και SQL Injection.

Επιστημονική περιοχή: Web Vulnerability Assessment, Penetration Testing

Λέξεις κλειδιά: Web Vulnerability Assessment, Cross Site Scripting, SQL Injection, Information Disclosure

Πίνακας Περιεχομένων

1. **Εισαγωγή 10**

2. **Τύποι Εργαλείων για Web Vulnerability Assessment 12**

2.1 Crawler based vulnerability scanners 12

2.2 Proxy based vulnerability scanners 18

2.3 Σύγκριση των δύο κατηγοριών 23

3. **Παρουσίαση του Εργαλείου 25**

4. **Οδηγός Εγκατάστασης Εργαλείου 39**

5. **Συμπεράσματα και Μελλοντική Εργασία 42**

6. **Διαδικτυακές Αναφορές 44**

Κατάλογος Σχημάτων

Εικόνα 2.1: WebSecurify crawler based vulnerability scanning tool 13

Εικόνα 2.2: WebSecurify vulnerability scanning process 14

Εικόνα 2.3: Εύρημα XSS στο WebSecurify 15

Εικόνα 2.4: Αρχικό interface του w3af 17

Εικόνα 2.5: w3af output 18

Εικόνα 2.6: Interface του OWASP ZAP 20

Εικόνα 2.7: Ορισμός proxy στην 8080 21

Εικόνα 2.8: Καταγραφή ιστοσελίδων και εύρεση ευπαθειών 22

Εικόνα 2.9: OWASP ZAP – Alerts, vulnerabilities 22

Εικόνα 2.10: Αποτελέσματα του ZAP μετά από crawling 23

Εικόνα 3.1: Διάγραμμα κλάσεων του VAbB tool 26

Εικόνα 3.2: Εμφάνιση του VAbB στον browser εκτελώντας την αρχική σελίδα . 27

Εικόνα 3.3: Επιλογές του VAbB πριν τη διαδικασία εκκίνησης του assessment . 28

Εικόνα 3.4: Output WUI κατά τη διαδικασία του crawling 30

Εικόνα 3.5: Output WUI κατά τη διαδικασία του vulnerability assessment 31

Εικόνα 3.6: Σύνοψη αποτελεσμάτων του VAbB tool 31

Εικόνα 3.7: Ευρήματα για banner disclosure 32

Εικόνα 3.8: Ευρήματα για reflected XSS 33

Εικόνα 3.9: Σύνδεση στο openeclass ως student 34

Εικόνα 3.10: Αντιγραφή Session Cookie του openclass 35

Εικόνα 3.11: Ορισμός authentication cookie και pattern για αποφυγή σελίδων ... 35

Εικόνα 3.12: Εκκίνηση του VAbB για έλεγχο του openeclass ως student user.... 36

Εικόνα 3.13: Τελικά αποτελέσματα του VAbB για τον έλεγχο στο openeclass ως student user 37

Εικόνα 3.14: SQL Injection error 38

Συντομογραφίες

WUI – Web User Interface

XSS – CSS – Cross Site Scripting

w3af – Web Application Attack and Audit Framework

OWASP – Open Web Application Security Project

ZAP – Zed Attack Proxy

Κεφάλαιο 1

Εισαγωγή

Το Web Hacking στις μέρες μας αποτελεί σύνηθες φαινόμενο. Καθημερινά γίνεται παραπάνω από μια αναφορά στα μέσα ενημέρωσης, κυρίως στα διαδικτυακά μέσα, για κάποιο περιστατικό που συνέβη σε κάποια επώνυμη ιστοσελίδα. Με τη χρήση των blogs και των κοινωνικών δικτύων εξαπλώνονται γρήγορα τρόποι και μεθοδολογίες για το πώς μπορεί κάποιος να ανακαλύψει κενά ασφάλειας σε μια διαδικτυακή εφαρμογή και πώς να τα εκμεταλλευτεί.

Έχουν δημιουργηθεί πολλά αυτοματοποιημένα εργαλεία τα οποία πραγματοποιούν επιθέσεις. Πλέον ακόμα και ένας χρήστης με ελάχιστες γνώσεις, μπορεί να τα χρησιμοποιεί με ευκολία. Τα περισσότερα από αυτά είναι ανοιχτού κώδικα και πλέον προσφέρονται και με γραφικό περιβάλλον κάνοντας τα ακόμα πιο εύχρηστα.

Έτσι από την πλευρά τους οι μηχανικοί ασφάλειας για να προστατεύσουν τα συστήματα που επιτηρούν, δημιούργησαν εργαλεία που προσφέρουν αποτίμηση της ασφάλειας σε ένα πληροφοριακό σύστημα. Τα εργαλεία αυτά είναι αντίστοιχης λογικής με τα εργαλεία που χρησιμοποιούνται για να βλάψουν ένα πληροφοριακό σύστημα. Δηλαδή εφαρμόζουν τεχνικές και μεθοδολογίες που εφαρμόζονται και στα κακόβουλα εργαλεία, αλλά φτιάχνονται με σκοπό να ενημερώνονται άμεσα οι μηχανικοί για τυχών ευπάθειες που παρουσιάζονται στο πληροφοριακό τους σύστημα.

Υπάρχουν εργαλεία όπως το Nessus το οποίο κάνει αποτίμηση της ασφάλειας στο δίκτυο της υποδομής και στο λειτουργικό σύστημα, το Metasploit κ.α. αλλά υπάρχουν και εργαλεία που επικεντρώνονται σε επιθέσεις που μπορούν να πραγματοποιηθούν σε διαδικτυακές εφαρμογές.

Στην παρούσα εργασία θα γίνει αναφορά των δύο κατηγοριών που απαρτίζουν τα Web vulnerability scanner tools και θα παρουσιαστούν τέσσερα από τα πιο γνωστά εργαλεία που πραγματοποιούν Web vulnerability assessment. Στη συνέχεια θα παρουσιαστεί το εργαλείο που υλοποιήθηκε για τις ανάγκες της διπλωματικής εργασίας καθώς και οδηγός εγκατάστασης του εργαλείου.

Κεφάλαιο 2

Τύποι Εργαλείων για Web Vulnerability Assessment

Τα Web vulnerability scanners βασίζονται σε τρεις βασικές λειτουργίες. Αυτές είναι ο **Crawler** ο οποίος αναλαμβάνει τη συλλογή των δεδομένων που χρειάζονται ώστε να γίνει το scanning, μια καλή τεχνική για γρήγορο **pattern matching** και τέλος, δυνατότητα επικοινωνίας με τον web server (**http requests**) της εφαρμογής που γίνεται το assessment. Υπάρχουν και άλλες λειτουργίες που χρειάζονται, αλλά αποτελούν υποκατηγορίες των παραπάνω ή δεν είναι τόσο αναγκαίες για τη δημιουργία ενός τέτοιου εργαλείου. Τέτοιες είναι ένας repeater για την επανεκπομπή http request που έχει ανακαλύψει ο scanner ως ευπαθή, fuzzer, decoder κλπ.

Μερικά εργαλεία αντί για crawler χρησιμοποιούν proxy για τη συλλογή των δεδομένων. Δηλαδή αντί να συλλέγουν δεδομένα μιας ιστοσελίδας αυτόματα, ορίζονται ως ενδιάμεσοι μεταξύ browser και διαδικτύου και καταγράφουν την κίνηση.

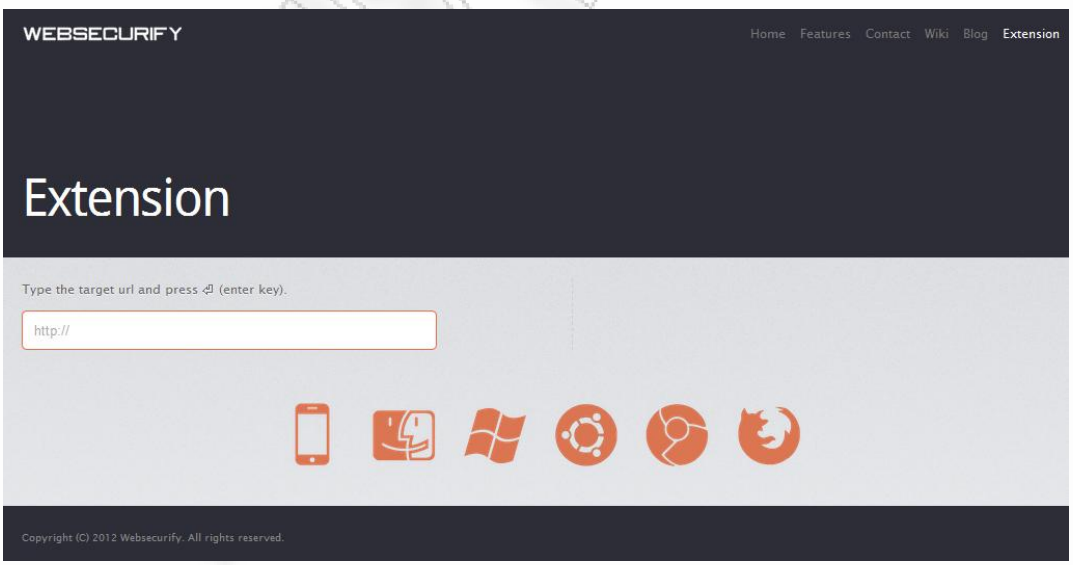
Σε αυτό το κεφαλαίο θα αναλυθούν οι δύο κατηγορίες crawler και proxy based vulnerability scanners, καθώς επίσης θα ακολουθήσει σύγκριση μεταξύ των δύο ώστε να γίνουν σαφή τα πλεονεκτήματα και τα μειονεκτήματα του καθενός.

2.1 Crawler based vulnerability scanners

Ένα crawler based vulnerability scanner, ζητώντας απλά και μόνο τη σελίδα που θέλει να ελέγξει για πιθανές ευπάθειες, συλλέγει για τον tester όλα τα links που αφορούν τη συγκεκριμένη σελίδα αλλά και HTML φόρμες που εμπεριέχονται

μέσα σε αυτές. Τα δεδομένα που συλλέγονται διαχωρίζονται σε POST και GET links μαζί με τα variables που τα απαρτίζουν και αφού ολοκληρωθεί η διαδικασία του crawling ή και κατά την διάρκεια αυτού, γίνεται το vulnerability scanning. Υπάρχουν πολλές επιλογές που προσφέρονται πριν την εκκίνηση του crawling. Μερικές είναι η δυνατότητα αλλαγής της πόρτας σύνδεσης με το server (προεπιλεγμένη είναι η 80), περιορισμός στον αριθμό των σελίδων που θα κάνει crawl, να κρατάει στην μνήμη σελίδες με συγκεκριμένα keywords ή να απορρίπτει σελίδες με συγκεκριμένα keywords και άλλες πολλές επιλογές. Οι σημαντικότερες από αυτές θα παρουσιαστούν στο κεφάλαιο 3 όπου εκεί θα αναλυθεί το εργαλείο που υλοποιήθηκε και είναι βασισμένο σε crawler.

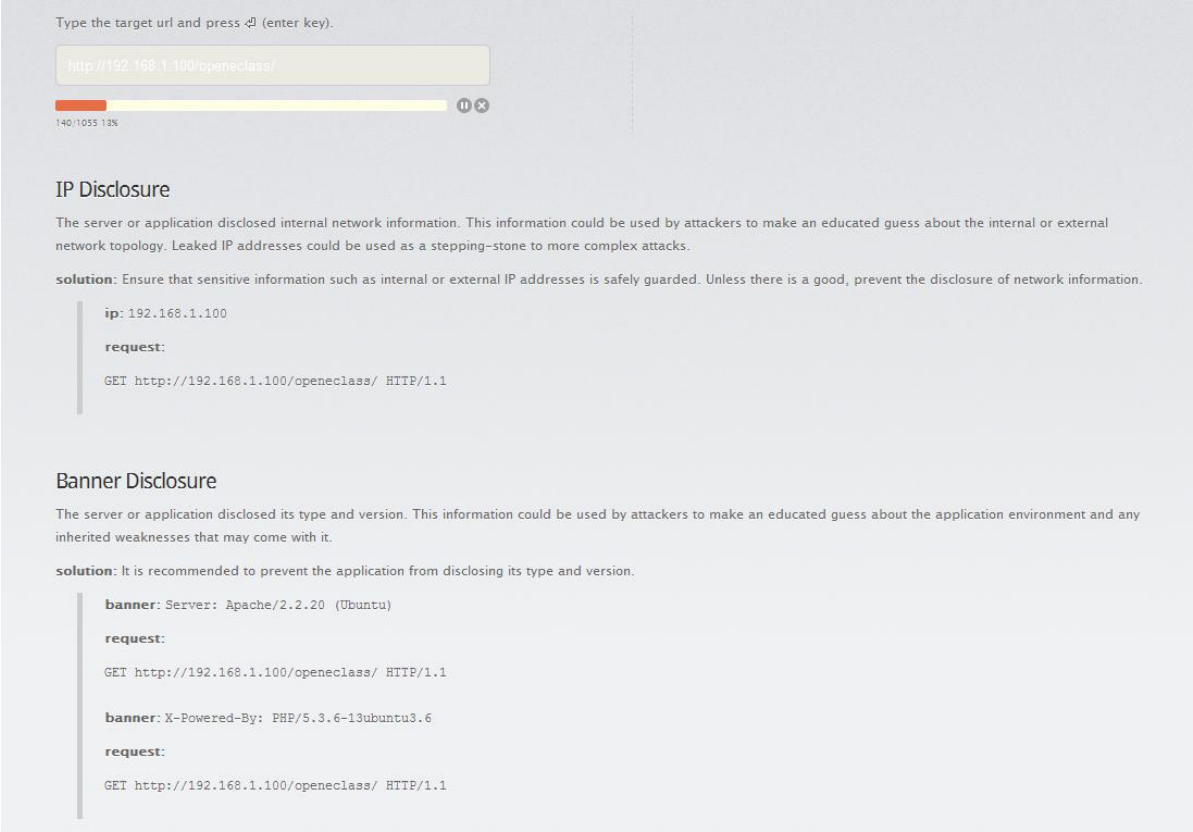
Δύο πολύ γνωστά crawler based εργαλεία για vulnerability scanning και είναι και ανοιχτού κώδικα είναι τα WebSecurify [11] και w3af [9]. Και τα δυο αυτά εργαλεία κατά τη διάρκεια του crawling κάνουν ταυτόχρονα και το vulnerability scanning. Στην εικόνα 2.1 παρατηρούμε το αρχικό interface του WebSecurify στο οποίο ζητείται από τον tester μόνο η σελίδα που πρόκειται να ελεγχθεί. Το WebSecurify όπως φαίνεται και στην εικόνα 2.1 είναι ένα portable, multi - platform εργαλείο. Μπορεί να εγκατασταθεί σε Windows, Linux, MAC



Εικόνα 2.1: WebSecurify crawler based vulnerability scanning tool

λειτουργικά συστήματα αλλά και σε έξυπνες συσκευές (smartphones), καθώς επίσης και σαν extension σε browsers όπως ο Google Chrome, Mozilla Firefox. Στη συγκεκριμένη παρουσίαση του εργαλείου χρησιμοποιήθηκε το extension για Google Chrome σε λειτουργικό σύστημα MS Windows 7.

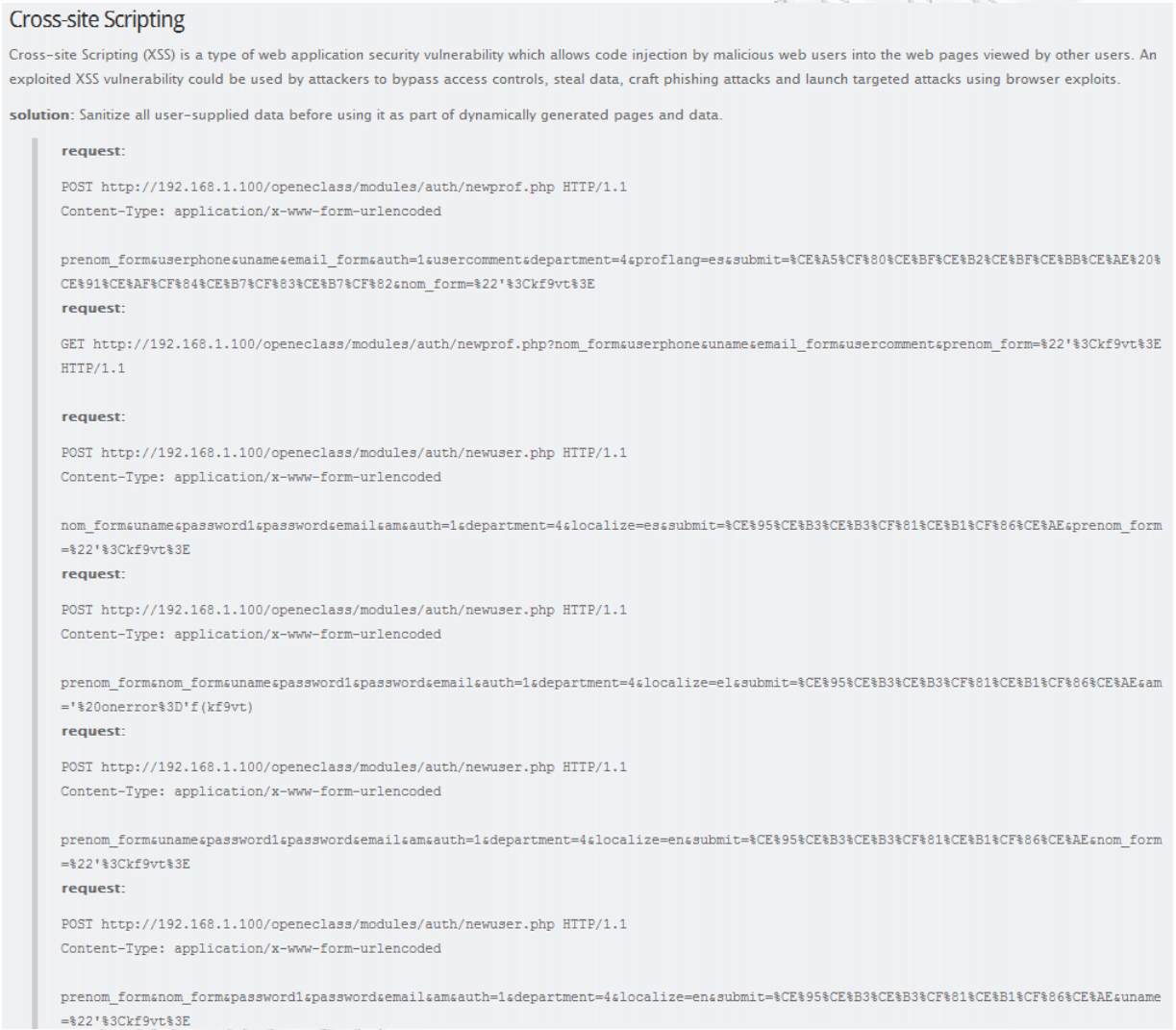
Δίνοντας μια σελίδα στο εργαλείο και πατώντας το enter, ξεκινάει η διαδικασία η οποία φαίνεται στην εικόνα 2.2. Για το παράδειγμα μας καθώς και για τα επόμενα χρησιμοποιήθηκε η εφαρμογή του openeclass [19], έκδοση 2.3.1.



Εικόνα 2.2: WebSecurify vulnerability scanning process

Παρατηρούμε ότι κατά τη διάρκεια του crawling εμφανίζονται λεπτομέρειες από το vulnerability scanning όπως το banner disclosure που έχει να κάνει με αποκάλυψη πληροφοριών του web server καθώς και το framework που χρησιμοποιείται για την πλατφόρμα του eclass. Καθώς το crawling process

συνεχίζεται εμφανίζονται όλο και περισσότερα ευρήματα όπως Cross Site Scripting που θεωρείται από τις πιο γνωστές επιθέσεις που εφαρμόζουν οι hackers. Έτσι στην εικόνα 2.3 βλέπουμε μερικά ευρήματα για XSS. Το output



Εικόνα 2.3: Εύρημα XSS στο WebSecurify

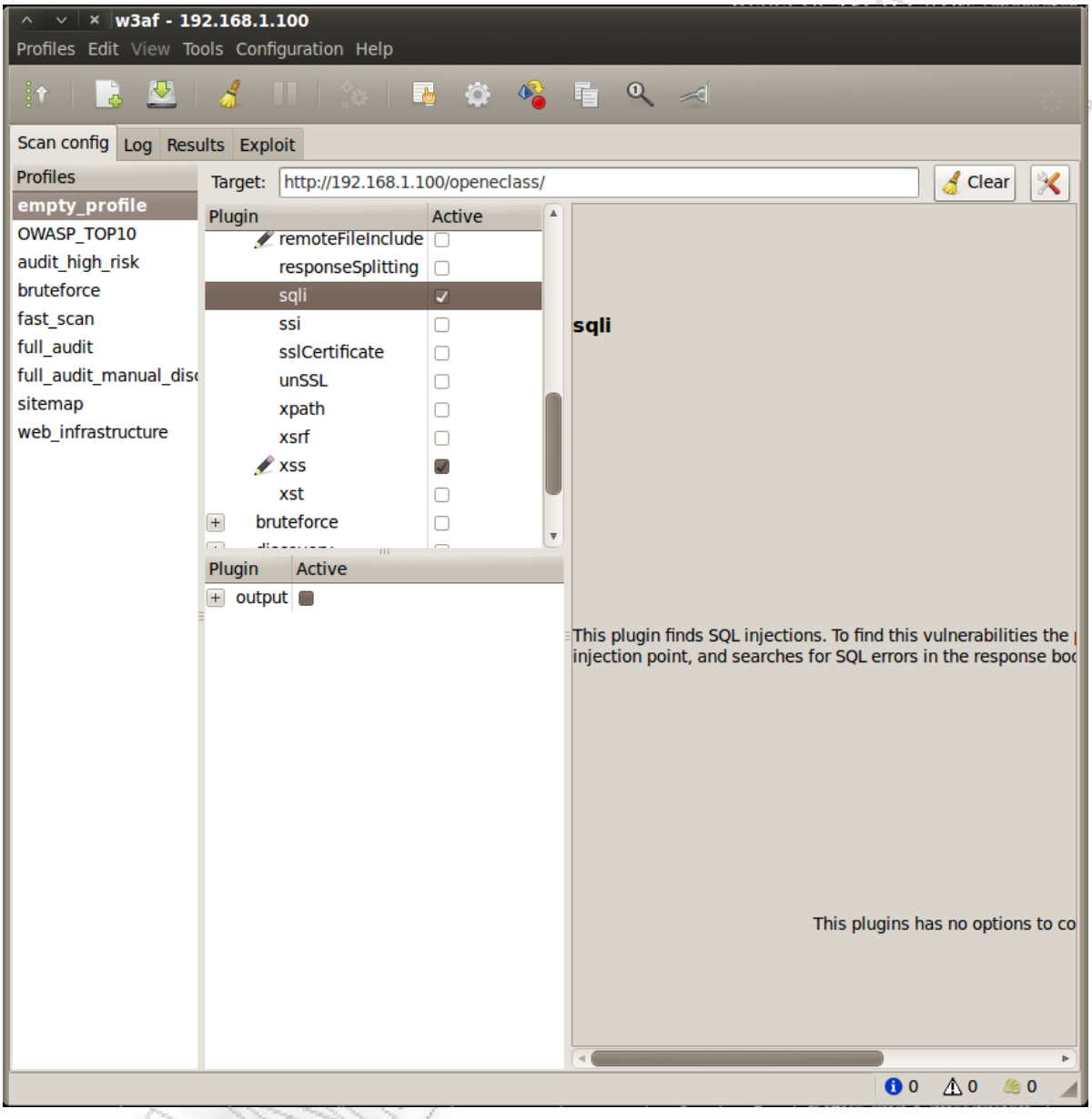
προς τον tester είναι το requested URL, αν είναι POST ή GET, η έκδοση του πρωτοκόλλου και τα variables μαζί με την ευπαθή GET ή POST μεταβλητή και το περιεχόμενο που δέχτηκε για να προκληθεί το XSS. Για παράδειγμα στο πρώτο

request η ευπαθή μεταβλητή του POST request είναι η **prenom_form** που δέχτηκε σαν τιμή **%22'%3Ckf9vt%3E** σε hex ή **"'<kf9vt>** σε ascii. Στην παρούσα εργασία δεν θα γίνει αναφορά για το τι είναι και το πώς πραγματοποιούνται XSS ή SQL Injection επιθέσεις. Θεωρείται ότι είναι γνωστές προς τον αναγνώστη.

Για το w3af παρέχεται έκδοση με γραφικό περιβάλλον αλλά και κονσόλα. Έχει σχεδιαστεί σε γλώσσα Java και είναι ένα πολλά υποσχόμενο εργαλείο. Περιέχει μια μεγάλη γκάμα από ελέγχους ασφάλειας οι οποίες μπορούν να ενεργοποιηθούν ή να απενεργοποιηθούν μέσα από profiles που μπορεί να φτιάξει ο tester ανάλογα με τις ανάγκες τις κάθε εφαρμογής. Στην εικόνα 2.4 παρατηρείται το αρχικό interface του w3af με τα έτοιμα profiles στο αριστερό μέρος, το πεδίο target όπου εκεί εισάγεται το URL του στόχου, τα plugins που προσφέρονται καθώς και τη μορφή του output που επιθυμεί ο tester να αποθηκεύσει στο τέλος της διαδικασίας.

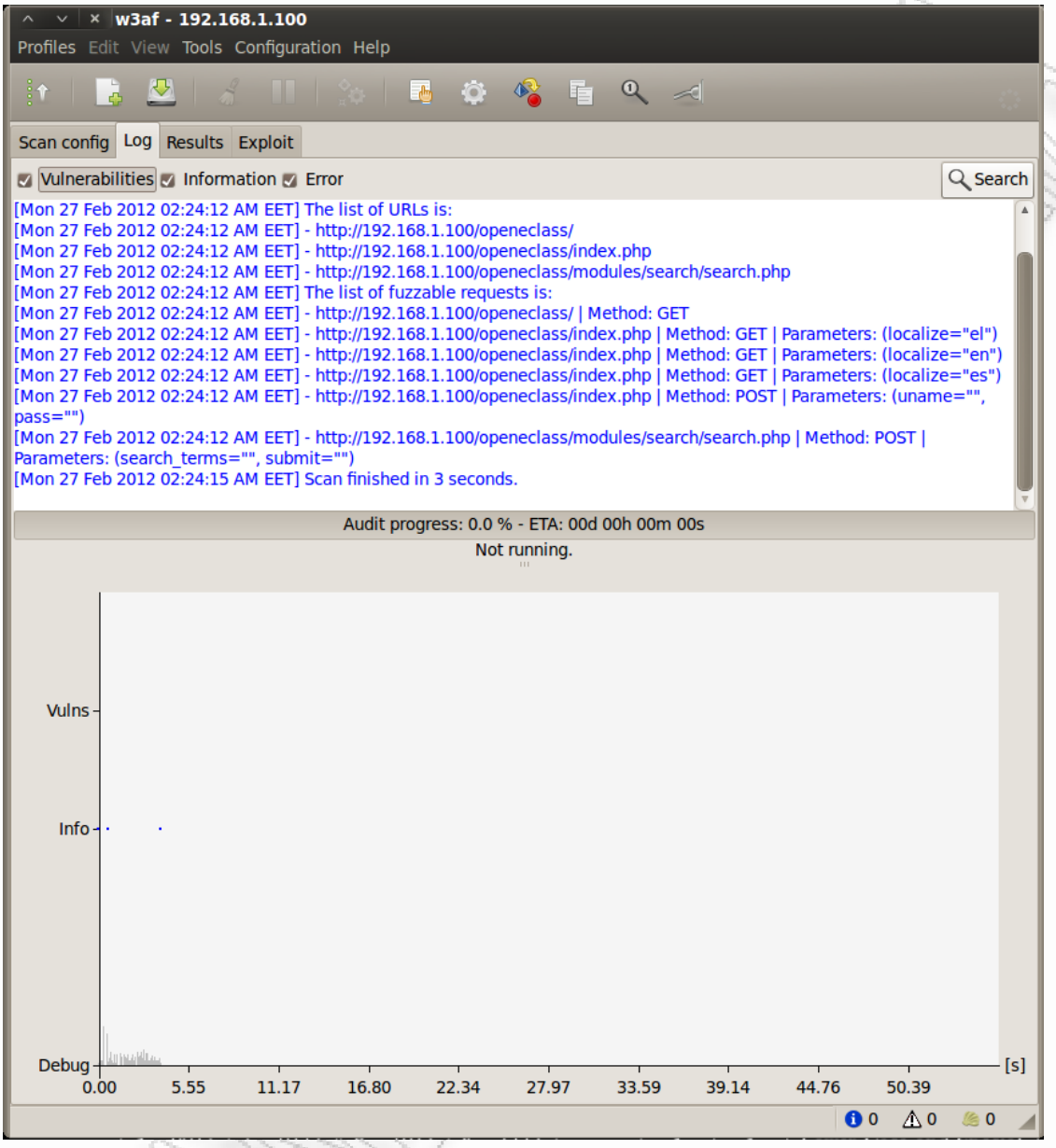
Το πιο πολυχρησιμοποιημένο profile είναι αυτό του OWASP_TOP10. Το OWASP Top 10 [20] περιέχει τις δέκα διασημότερες επιθέσεις που πραγματοποιούνται στο διαδίκτυο κάθε χρόνο. Η έρευνα πραγματοποιείται από την κοινότητα του OWASP. Ο tester έχει τη δυνατότητα να δημιουργεί δικά του profile ενεργοποιώντας τα plugin που επιθυμεί ανάλογα με τον έλεγχο που κάνει. Το profile μπορεί να αποθηκευτεί και να χρησιμοποιηθεί μελλοντικά σε κάποιο άλλο έλεγχο.

Το output του security audit φαίνεται στην εικόνα 2.5. Το output χωρίζεται σε δυο μέρη. Στο κάτω μέρος εμφανίζονται τα ευρήματα σε μορφή γραφήματος και στο πάνω μέρος που εμφανίζονται σε κονσόλα λεπτομέρειες για τα ευρήματα. Στον άξονα τον y έχει την περιοχή των vulnerabilities με όσο πιο ψηλά εμφανίζεται ένδειξη να είναι και πιο κρίσιμη η ευπάθεια και την περιοχή info που δεν αφορά vulnerabilities αλλά information disclosure τα οποία μπορεί να δώσουν σε έναν κακόβουλο χρήστη χρήσιμες πληροφορίες για να επιτεθεί μελλοντικά.



Εικόνα 2.4: Αρχικό interface του w3af

Στην εικόνα 2.5 δεν εμφανίστηκαν vulnerabilities παρόλο που, όπως φάνηκε και με το εργαλείο WebSecurify, υπάρχουν αρκετές. Αυτό ίσως οφείλεται σε επιπλέον ρυθμίσεις που δεν έγιναν στο εργαλείο.



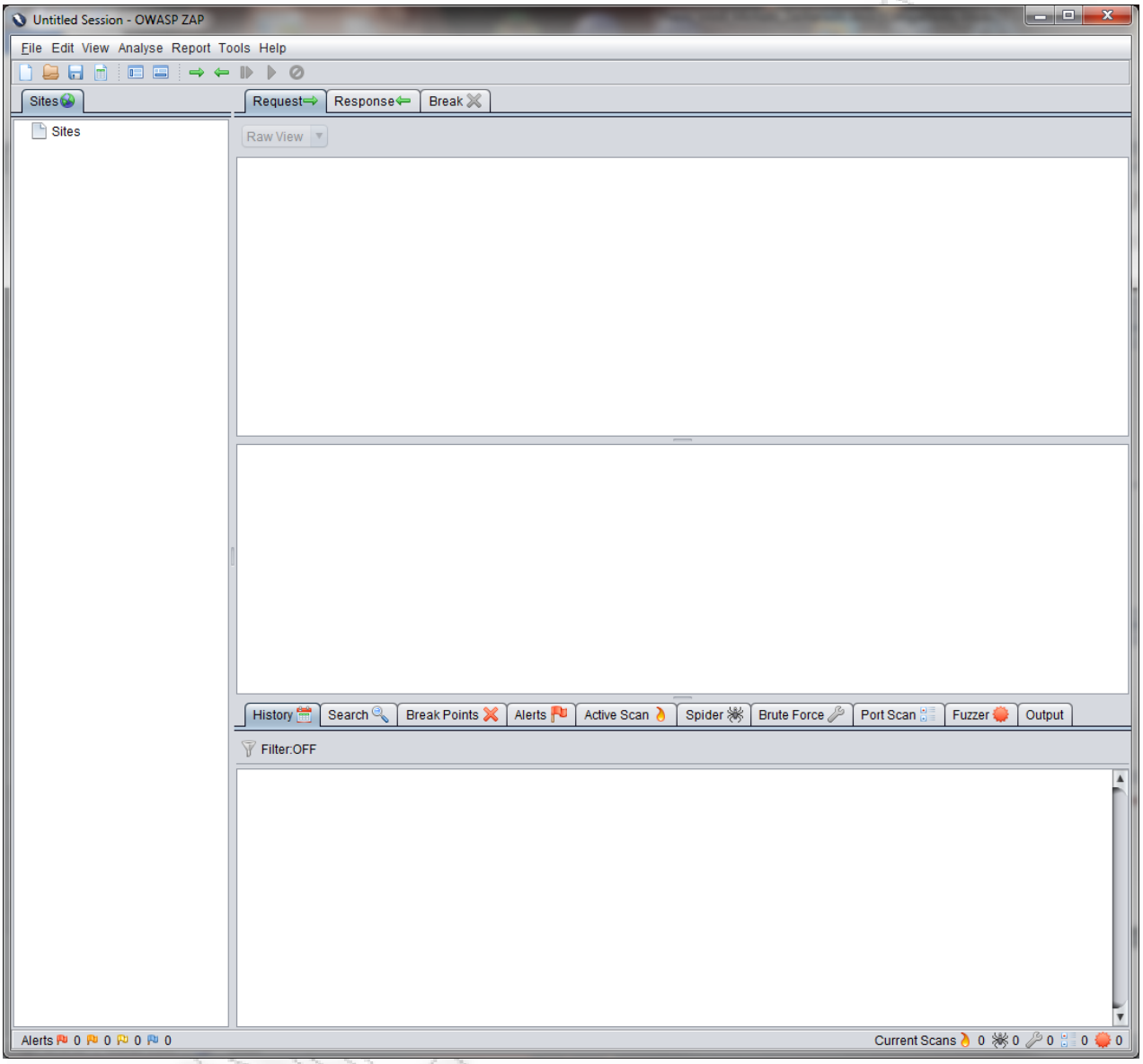
Εικόνα 2.5: w3af output

2.2 Proxy based vulnerability scanners

Τα vulnerability scanning tools τα οποία βασίζονται σε proxy, λειτουργούν ως ενδιάμεσοι μεταξύ browser και διαδικτύου και καταγράφουν την κίνηση. Η κίνηση ουσιαστικά είναι http request και responses. Τα πακέτα αποθηκεύονται και ταυτοχρόνως ελέγχονται για πιθανές ευπάθειες.

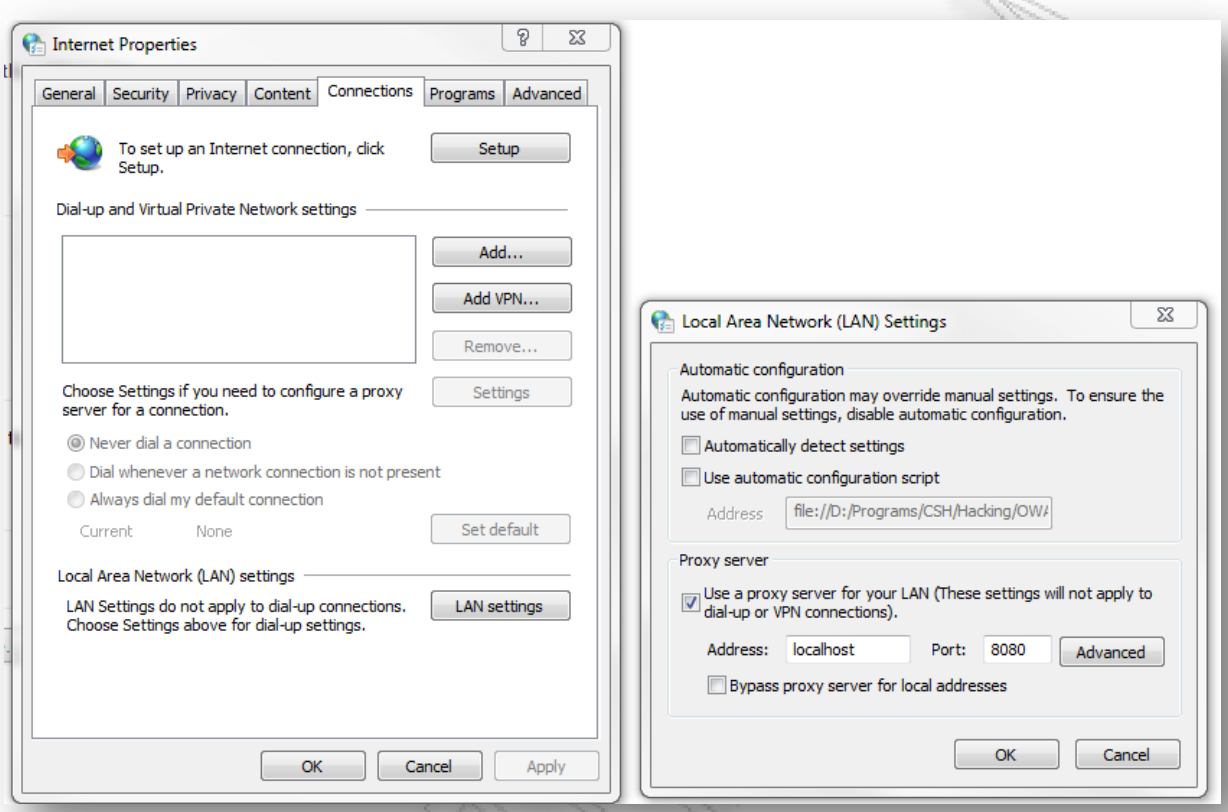
Ένα proxy based vulnerability scanning tool βασίζεται σε μεγάλο βαθμό στον tester. Δηλαδή από τη στιγμή που θα οριστεί σαν ενδιάμεσος καταγράφει την κίνηση που προκαλεί ο tester μέσα από το browser του. Αν ο tester ελέγξει χειροκίνητα για SQL Injection μια HTML φόρμα μιας ιστοσελίδας, τότε το εργαλείο αναλύει τα responses και ελέγχει για πιθανή ευπάθεια. Αν ο tester πετύχει το SQL Injection τότε το εργαλείο τον ειδοποιεί για το εύρημα. Όποτε ο tester επιθυμεί μπορεί να επιλέξει μέσα από το εργαλείο να εκτελέσει και αυτό με τη σειρά του ελέγχους πάνω στις φόρμες και τα URLs που ο tester προηγουμένως είχε επισκεφτεί.

Τα δύο πιο γνωστά εργαλεία τέτοιου τύπου είναι το OWASP-ZAP [10] και το Burp Suite [8]. Στην εικόνα 2.6 απεικονίζεται το interface του εργαλείου ZAP. Στο αριστερό μέρος (sites) εμφανίζονται οι ιστοσελίδες που επισκέπτεται ο tester στον browser. Για να ξεκινήσει το ZAP να συλλέγει δεδομένα θα πρέπει να ορίσουμε έναν browser να συνδέεται στο διαδίκτυο μέσω proxy. Η πόρτα που ακούει το ZAP είναι η 8080. Ανοίγουμε για παράδειγμα τον Google Chrome και πηγαίνουμε στις επιλογές για να ορίσουμε τον proxy (εικόνα 2.8). Επιλέγουμε localhost γιατί το ZAP βρίσκεται στο ίδιο host με τον browser. Όταν αρχίσουμε να ανοίγουμε links αυτομάτως στην αριστερή μπάρα (sites) του εργαλείου θα εμφανιστεί το link της σελίδας. Έτσι, στην εικόνα 2.8 παρατηρούμε την IP του eclass που χρησιμοποιήσαμε και σε προηγούμενα παραδείγματα. Τα κίτρινα σημαιάκια (flags) δίπλα από την σελίδα δηλώνουν ευρήματα από το αυτόματο vulnerability scanning που γίνεται. Για να δούμε λεπτομέρειες για τα ευρήματα επιλέγουμε το tab alerts. Στην εικόνα 2.9 εμφανίζονται τα alerts με λεπτομερή περιγραφή. Με το πρώτο request στην σελίδα εμφάνισε το ZAP εμφάνισε και



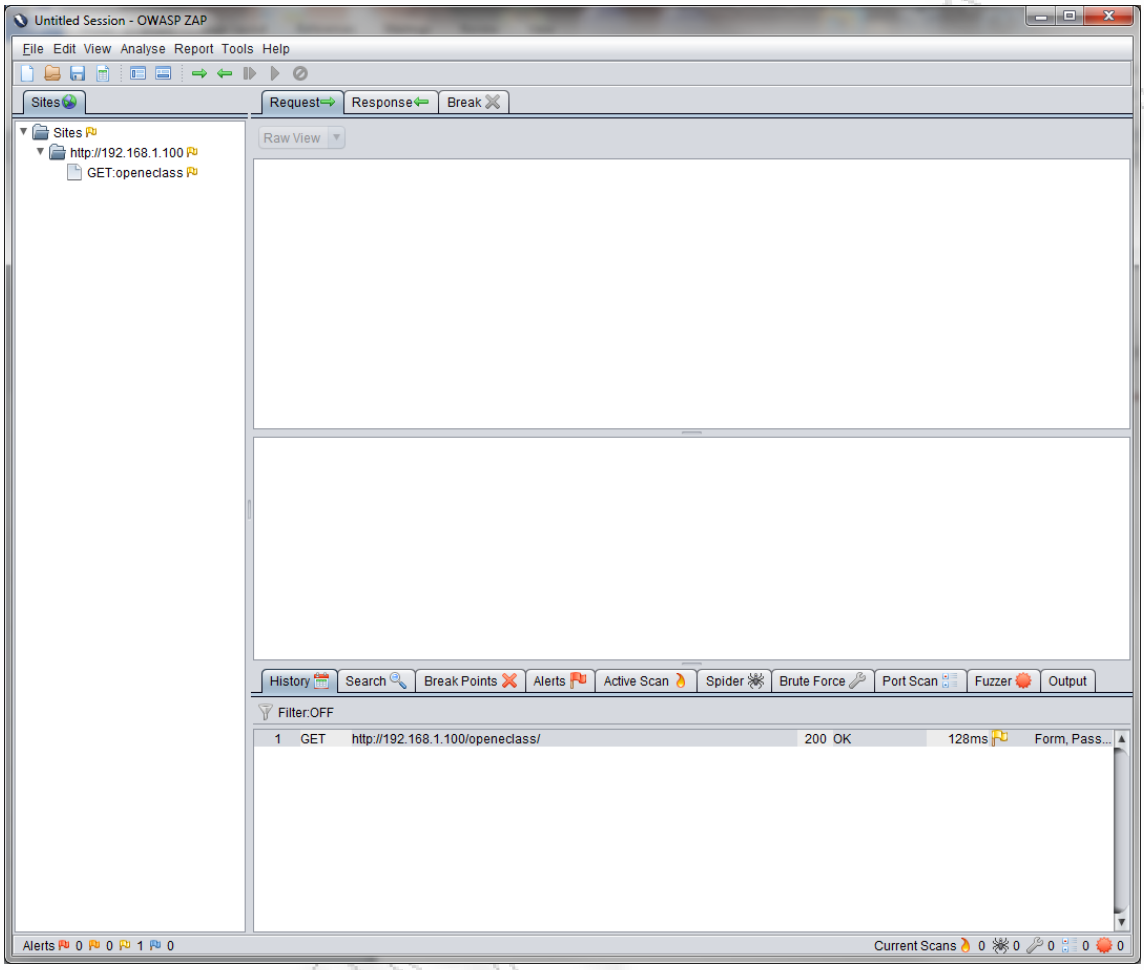
Εικόνα 2.6: Interface του OWASP ZAP

κάποιο alert. Εάν επιλέξουμε το tab Active Scan, το ZAP ελέγχει όλα τα ευρήματα από τα requests που έχει κάνει ο tester μέσα από τον browser. Σε αυτή τη φάση δεν θα βρει τίποτα.

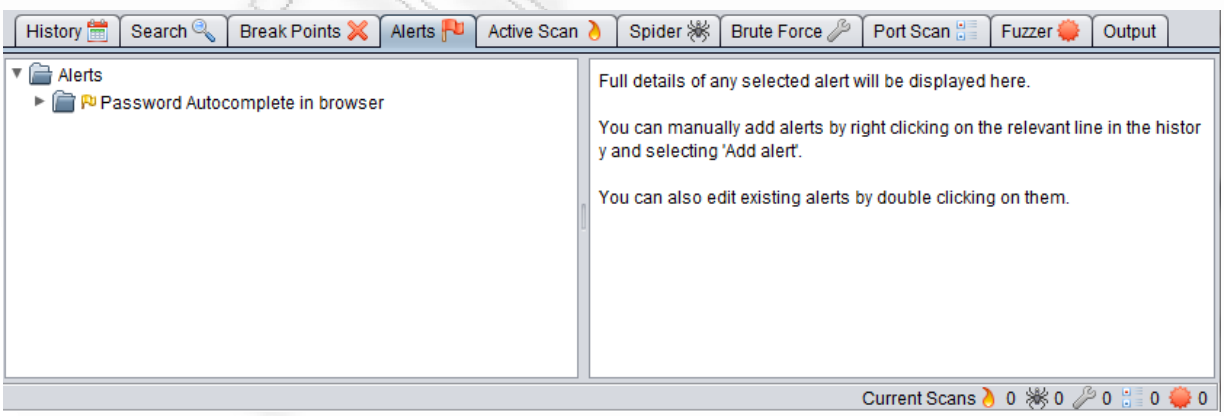


Εικόνα 2.7: Ορισμός proxy στην 8080

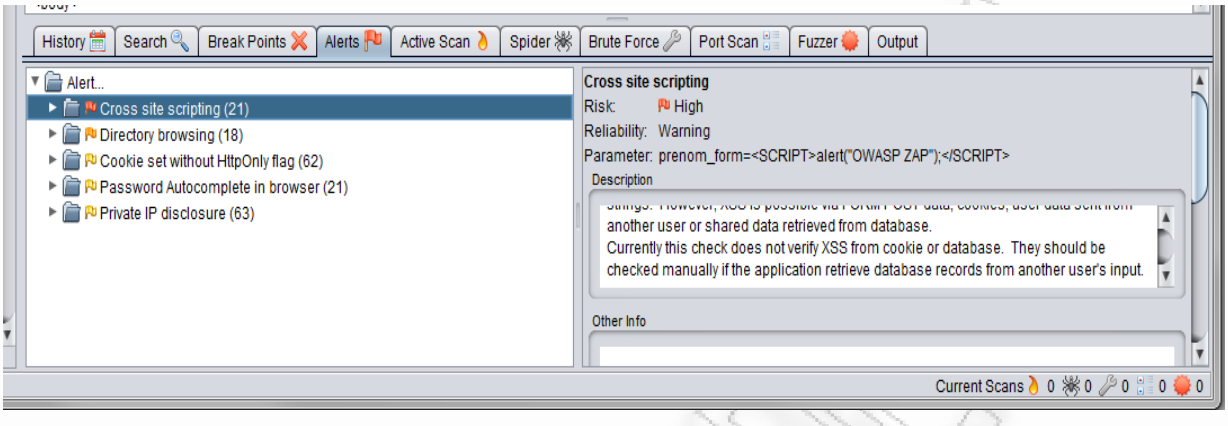
Το ZAP αν και proxy based εργαλείο, περιέχει και crawler. Ο crawler βρίσκεται στο tab spider. Εφόσον το spider ολοκληρώσει τη διαδικασία ξανά επιλέγουμε το Active Scan. Πλέον στα Alerts εμφανίζονται πολλά περισσότερα vulnerabilities. Στην εικόνα 2.10 με κόκκινο έντονο χρώμα στα flags, ξεχωρίζει το XSS.



Εικόνα 2.8: Καταγραφή ιστοσελίδων και εύρεση ευπαθειών



Εικόνα 2.9: OWASP ZAP – Alerts, vulnerabilities



Εικόνα 2.10: Αποτελέσματα του ZAP μετά από crawling

2.3 Σύγκριση των δύο κατηγοριών

Πλεονεκτήματα και μειονεκτήματα crawler και proxy based εργαλείων:

1) Σε ένα crawler based εργαλείο δεν χρειάζεται ο tester να επεξεργαστεί την προς εξέταση σελίδα – εφαρμογή. Ενώ σε ένα καθαρά proxy based εργαλείο ο χρήστης πρέπει να έρθει σε επαφή με τη σελίδα.
2) Σε περίπτωση που ο tester θελήσει να κάνει crawl σελίδες που απαιτούν σύνδεση χρήστη (login), τότε θα πρέπει ο crawler να αποκτήσει με κάποιο τρόπο το session cookie που προσφέρει πρόσβαση στις διαχειριστικές σελίδες. Στα περισσότερα crawler based εργαλεία ο tester θα πρέπει να δώσει το cookie χειροκίνητα κάνοντας το εργαλείο λίγο πιο δύσχρηστο, αντίθετα με κάποιο proxy based εργαλείο το οποίο καταγράφοντας την κίνηση στον browser (δηλαδή τα http requests), καταγράφει και τα session cookies. Εφόσον το login πραγματοποιείται μέσω του browser, αποτελεί σαφώς την πιο εύχρηστη λύση.
3) Όταν η προς έλεγχο εφαρμογή – ιστοσελίδα βασίζεται σε τεχνολογία flash, ή αντίστοιχη, τότε ένας crawler δεν μπορεί να ανακαλύψει τα links και τα POST δεδομένα που κρύβονται πίσω από την ιστοσελίδα. Σε αυτήν την περίπτωση ένα proxy based εργαλείο απαιτείται για τον έλεγχο της σελίδας.

4) Σε περίπτωση που οι html φόρμες έχουν σχεδιαστεί να εκτελούν το submit των δεδομένων με JavaScript τότε ο crawler πιθανών να μην μπορεί να βρει σε ποιο link θα γίνει το submission. Σε αυτές τις περιπτώσεις το attribute action είναι κενό και όλα γίνονται μέσω JavaScript.
5) Σε μερικές γλώσσες προγραμματισμού δεν προσφέρεται βιβλιοθήκη για εύκολη δημιουργία http requests. Έτσι, η δημιουργία ενός crawler based εργαλείου είναι δυσκολότερη σε σχέση με ένα proxy based γιατί θα πρέπει να προσομοιωθεί προγραμματιστικά η σύνδεση που επιτυγχάνει ένας web browser. Στα proxy based εργαλεία το connection γίνεται μέσω του ίδιου του browser ο οποίος είναι ήδη προγραμματισμένος και σχεδιασμένος για να επιτυγχάνει τη σύνδεση με τον εκάστοτε web server.

   Πολλά προβλήματα παρουσιάζονται όταν προκύπτουν πολλά redirections από ένα http request. Φυσικά οι περισσότερες γλώσσες προγραμματισμού προσφέρουν έτοιμες βιβλιοθήκες για http connections και έτσι τα προβλήματα λύνονται. Διαφορετικά η λύση είναι τα sockets για τα οποία θα πρέπει ο προγραμματιστής να δημιουργεί τα http πακέτα χειροκίνητα και να προβλέψει πολλές περιπτώσεις ανάλογα με το status code που δέχεται στα http responses ώστε να πράττει ανάλογα.

Φυσικά ο συνδυασμός και των δυο είναι ο καλύτερος. Το ZAP που περιληπτικά παρουσιάστηκε στο υποκεφάλαιο 2.2 είναι από τα καλύτερα εργαλεία που επιτυγχάνουν αυτό το συνδυασμό.

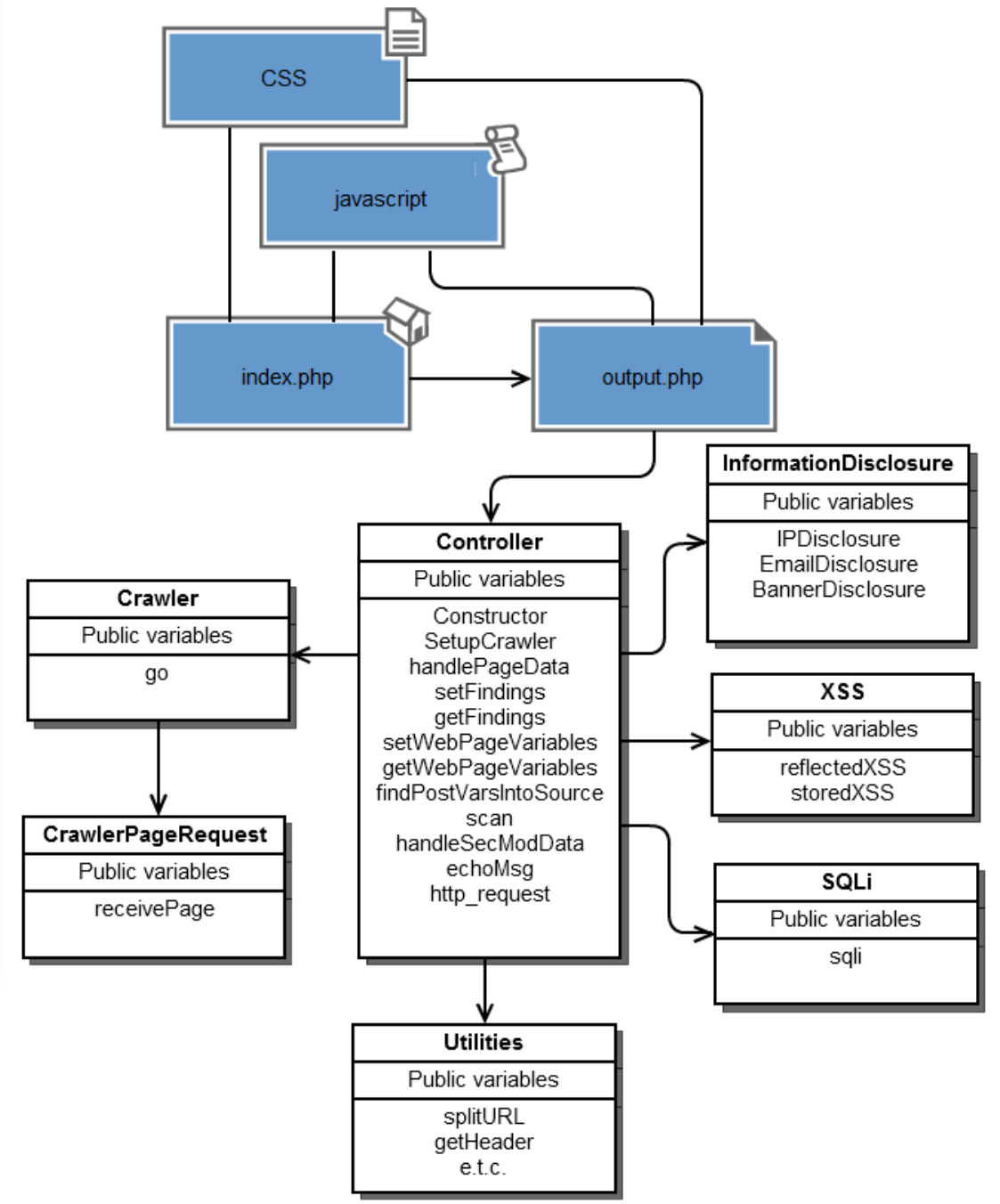
Κεφάλαιο 3

Παρουσίαση του Εργαλείου

Το εργαλείο που δημιουργήθηκε όπως προαναφέρθηκε σε προηγούμενο κεφάλαιο είναι βασισμένο σε crawler για την αυτόματη συλλογή των δεδομένων μιας ιστοσελίδας. Η ονομασία του είναι VAbB και αυτή θα χρησιμοποιείται από εδώ και πέρα όταν γίνεται αναφορά στο εργαλείο. Το VAbB βγαίνει από το αρχικά Vulnerability Assessment by Browsing. Το εργαλείο αυτό αποτελεί μια πρακτική απεικόνιση για το πως μπορούν να αυτοματοποιηθούν χρονοβόρες διαδικασίες που μέχρι πρότινος πραγματοποιούσε ένας tester χειροκίνητα. Βασική ιδέα του εργαλείου αυτού είναι ο τρόπος με τον οποίο σχεδιάζεται ώστε να παρέχει ευελιξία, φορητότητα και ταχύτητα στους testers. Για το λόγο αυτό επιλέχθηκε μια δυναμική γλώσσα διαδικτυακού προγραμματισμού, η PHP. Η PHP είναι μια server side γλώσσα προγραμματισμού, δηλαδή εκτελείται στην πλευρά του server. Ουσιαστικά, το εργαλείο αυτό θα μπορεί να ενεργοποιείται μέσω ενός web browser. Με αυτόν τον τρόπο, όλος ο φόρτος εργασίας των ελέγχων ασφαλείας μεταφέρεται στον web server και έτσι ακόμα και ένας υπολογιστής χαμηλών υπολογιστικών δυνατοτήτων θα μπορεί να εκτελεί τους ελέγχους με μεγάλη ευκολία και από οποιοδήποτε λειτουργικό σύστημα ή και ακόμα και κάποιο smartphone ή tablet.

Τέλος, η προοπτική του εργαλείου είναι να υποστηρίξει όσους περισσότερους ελέγχους ασφάλειας γίνεται και να δημιουργηθεί ένα πολυχρηστικό εργαλείο με δυνατότητα αποθήκευσης κίνησης (ιστορικό) και profile χρηστών.

Το VAbB έχει σχεδιαστεί με τέτοιο τρόπο ώστε να είναι εύκολα επεκτάσιμο μελλοντικά. Αξιοποιεί τη δυνατότητα που δίνει η PHP για χρήση αντικειμένων. Κάθε κλάση αποτελεί και μια βασική λειτουργία του εργαλείου. Στην εικόνα 3.1 φαίνεται το διάγραμμα κλάσεων του VAbB.



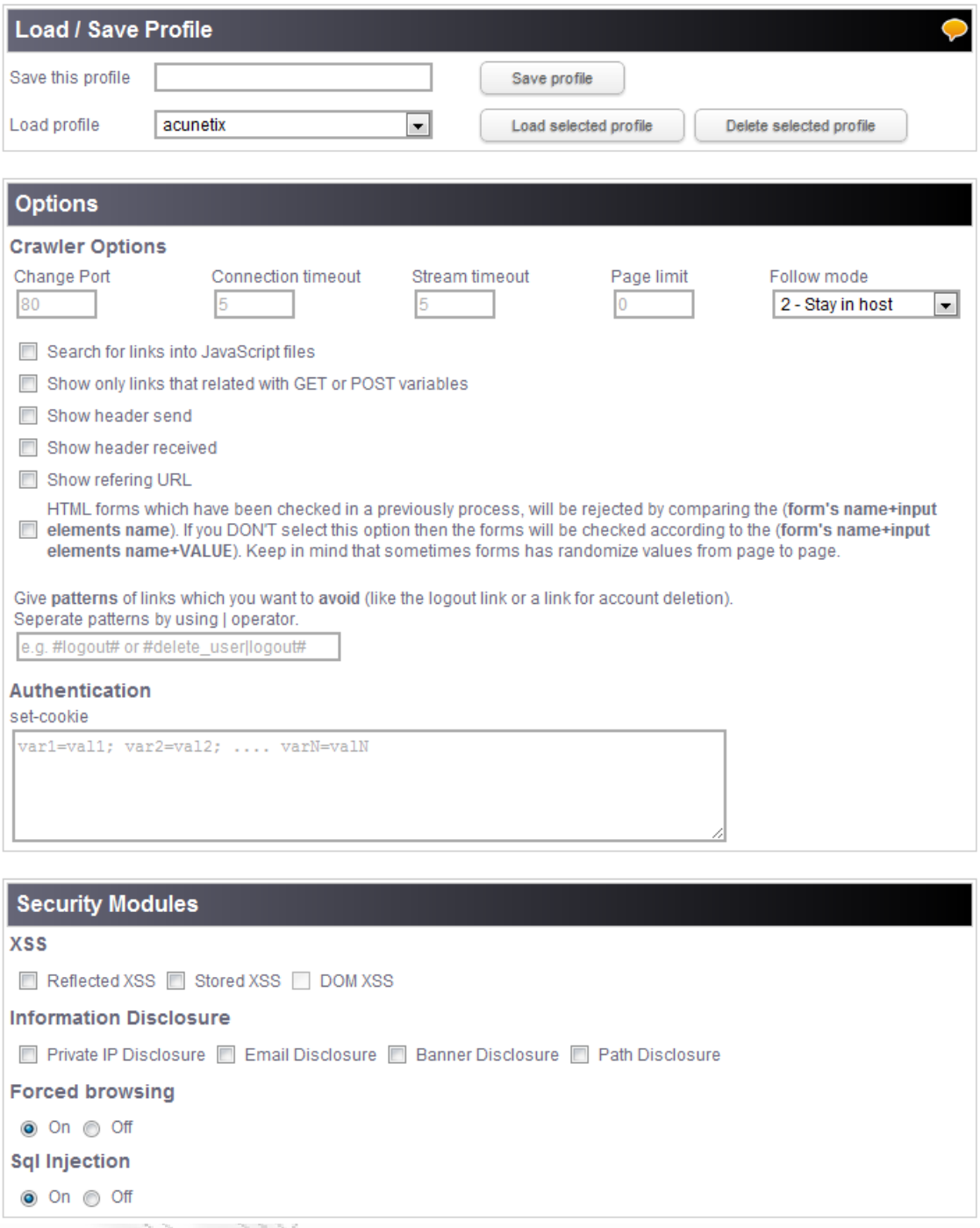
Εικόνα 3.1: Διάγραμμα κλάσεων του VAbB tool

Η κλάση Controller είναι η κεντρική και η πιο βασική κλάση. Μέσω αυτής ελέγχονται όλες οι υπόλοιπες κλάσεις. Το index.php και output.php δεν

αποτελούν κλάσεις. Περιέχουν html και php κώδικα ενσωματώνοντας CSS για τη μορφοποίηση και JavaScript για την εκτέλεση script κώδικα που χρειάζεται για τη σωστή λειτουργία του WUI. Όταν εκτελεστεί το index.php σε ένα browser εμφανίζεται η σελίδα που φαίνεται στην εικόνα 3.2.

Εικόνα 3.2: Εμφάνιση του VAbB στον browser εκτελώντας την αρχική σελίδα

Πατώντας το κουμπί **Options** εμφανίζονται επιλογές μέσω των οποίων ο χρήστης μπορεί να επιλέξει μερικές ρυθμίσεις για το crawling process, να ενσωματώσει ένα cookie ώστε το crawling αλλά και οι έλεγχοι να γίνουν σε σελίδες που απαιτούν authentication, να αποθηκεύσει το profile που επιθυμεί και να επιλέξει τα security modules που επιθυμεί να συμπεριληφθούν στον έλεγχο ασφάλειας. Πολύ σημαντική είναι η δυνατότητα που δίνεται στο χρήστη να προσθέσει κάποιο pattern για αποφυγή σελίδων. Για παράδειγμα, όταν το vulnerability scanning γίνεται σε σελίδες που απαιτούν authentication, τότε σελίδες που πραγματοποιούν αποσύνδεση ή διαγραφή χρήστη δεν θα πρέπει να εκτελεστούν από τον crawler. Οι επιλογές αυτές φαίνονται στην εικόνα 3.3.



Εικόνα 3.3: Επιλογές του VAbB πριν τη διαδικασία εκκίνησης του assessment

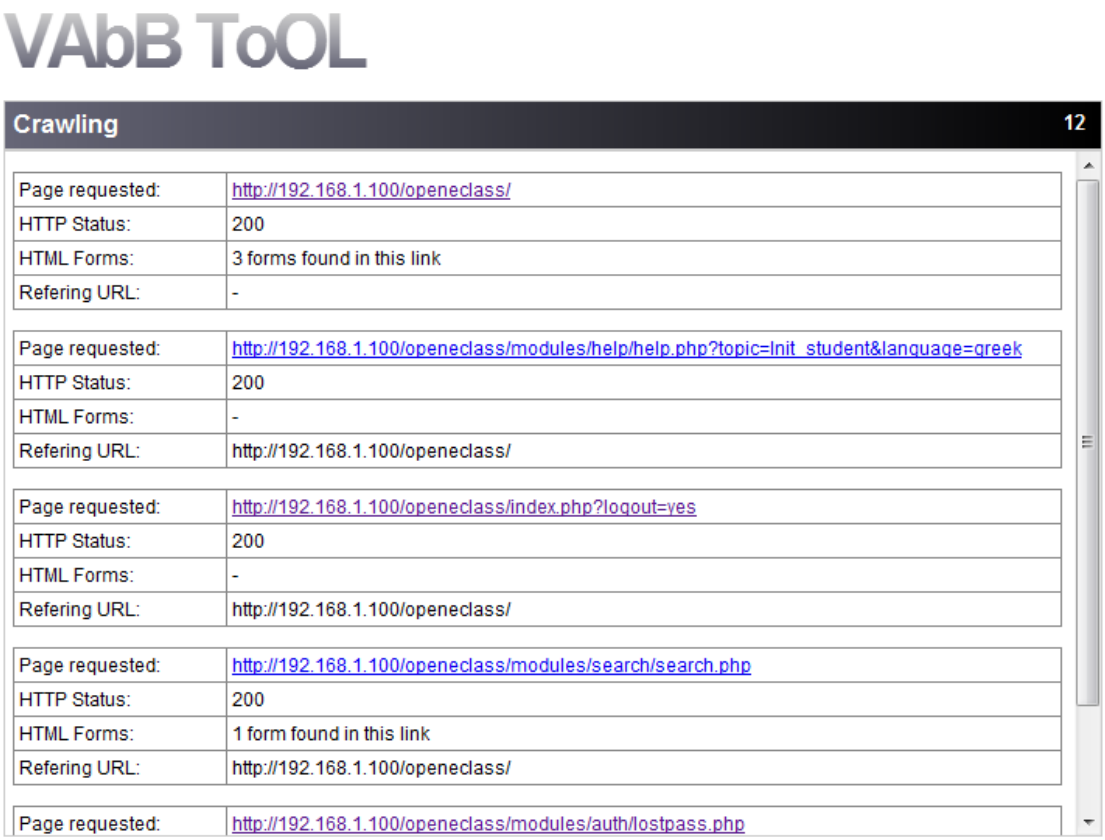
Έτσι μέσω αυτής της επιλογής ο tester μπορεί να προσθέσει κάποια pattern που πιθανών να συναντήσει μέσα στα links που θα συναντήσει ο crawler. Έτσι ένα

pattern της μορφής **#logout#i** θα αποφύγει το link http://www.example.com/user/logout.php ή ακόμα και αν το logout γράφεται με πεζά ή κεφαλαία γράμματα (αυτό το σκοπό έχει το **i** μέσα στο pattern). Σε περίπτωση που το αρχείο της ιστοσελίδας που πραγματοποιεί το logout είναι το log.php τότε το pattern δεν θα εντοπίσει το link. Γι' αυτό το λόγο θα πρέπει ο tester να είναι ενήμερος για σελίδες τέτοιου τύπου ώστε να φτιάξει τα σωστά patterns.

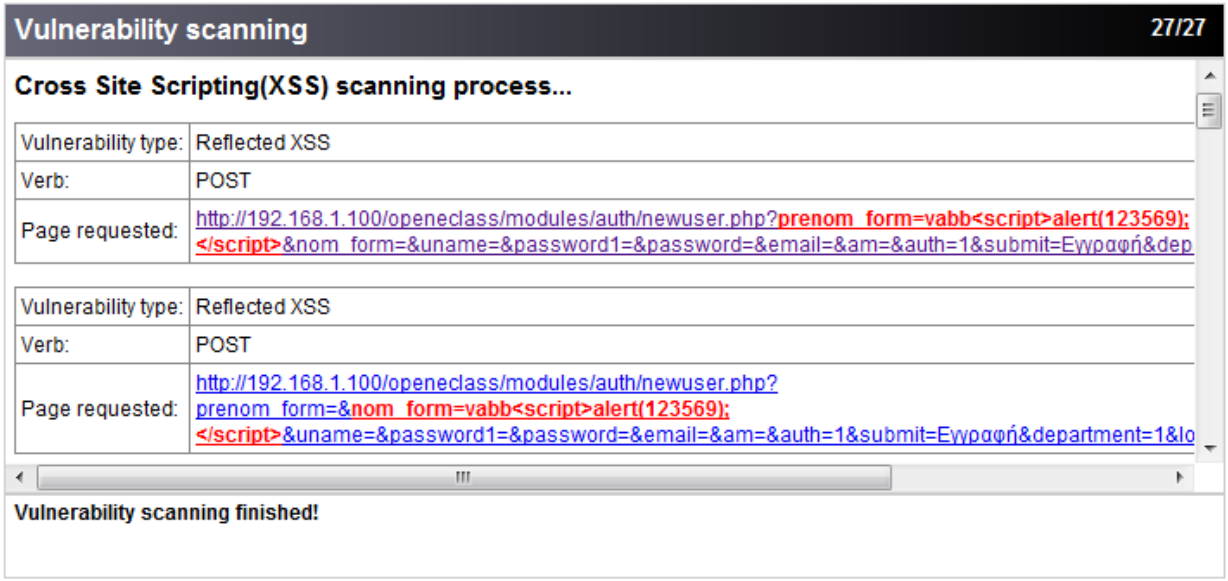
Έστω ότι στο URL τοποθετείται το link της εφαρμογής του openeclass. Από τη στιγμή που ο χρήστης επιλέξει τις επιλογές που θέλει και συμπληρώσει το URL στη μπάρα που φαίνεται στην εικόνα 3.2, επιλέγει το κουμπί Start Scan. Σε αυτό το σημείο ανοίγει το αρχείο output.php μέσω του οποίου καλείται ο Controller και ξεκινάει η διαδικασία του crawling. Όσο το crawling εκτελείται εμφανίζονται στην οθόνη links που έχουν γίνει crawl, όπως φαίνεται στην εικόνα 3.4. Πάνω δεξιά εμφανίζεται ο αριθμός των links που μέχρι στιγμής έχουν γίνει crawl. Μόλις ολοκληρωθεί η συλλογή των δεδομένων φορτώνεται ένας δεύτερος πίνακας ίδιας μορφής με αυτή του crawler, ο οποίος εμφανίζει ευρήματα, εάν υπάρχουν, κατά τη διάρκεια της διαδικασίας του vulnerability scanning. Στην εικόνα 3.5 φαίνεται αυτός ο πίνακας έχοντας κάποια ευρήματα από XSS links. Με κόκκινο χρώμα εμφανίζονται οι μεταβλητές GET ή POST με την τιμή που δόθηκε από το εργαλείο ώστε να προκληθεί το vulnerability. Για το openeclass βρέθηκαν συνολικά έντεκα ευπάθειες τύπου XSS ελέγχοντας τη σελίδα ως guest user.

Στο τέλος εμφανίζεται μια σύνοψη από πιθανά ευρήματα όπως απεικονίζεται στην εικόνα 3.6. Το Information Disclosure δεν αποτελεί κάποια ευπάθεια αλλά ενημερώνει τον tester ότι η σελίδα την οποία ελέγχει αποκαλύπτει κάποιες πληροφορίες δημόσια. Στην περίπτωση του openeclass που δοκιμάστηκε έχουμε Banner Disclosure και Private IP Disclosure. Εμφανίστηκαν ευρήματα για το private IP disclosure, επειδή το openeclass έχει εγκατασταθεί σε private δίκτυο και έτσι τα links του openclass είναι της μορφής http://192.168.1.100/openeclass/. Το banner disclosure έχει να κάνει με πληροφορίες

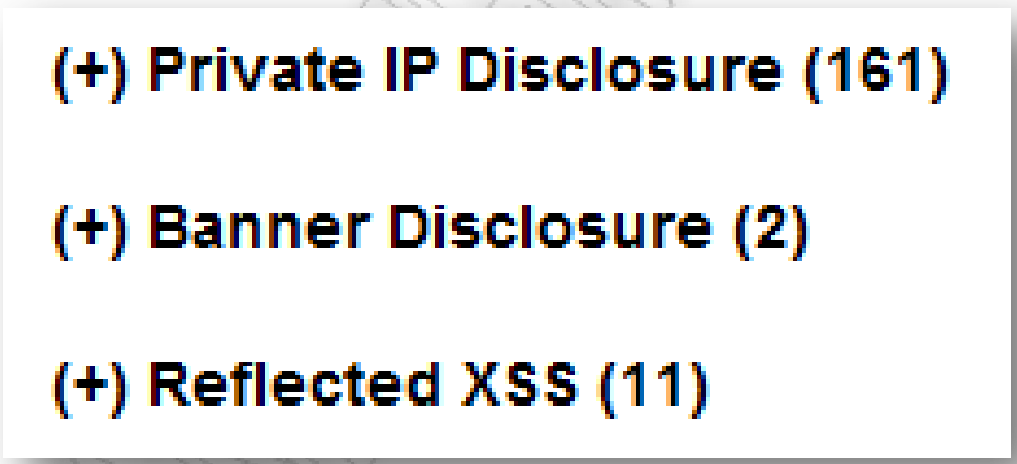
του server που είναι εγκατεστημένη η ιστοσελίδα που ελέγχουμε. Αν πατήσουμε το σύμβολο (+) που βλέπουμε στην εικόνα 3.6 για το banner disclosure θα εμφανιστούν τα ευρήματα. Έτσι όπως βλέπουμε στην εικόνα 3.7 έχουμε αποσπάσει τις πληροφορίες ότι ο web server είναι ο apache έκδοσης 2.2.20, το λειτουργικό σύστημα είναι Ubuntu και το framework php έκδοσης 5.3.6. Τέλος, για το reflected XSS τα ευρήματα παρουσιάζονται στην εικόνα 3.8.



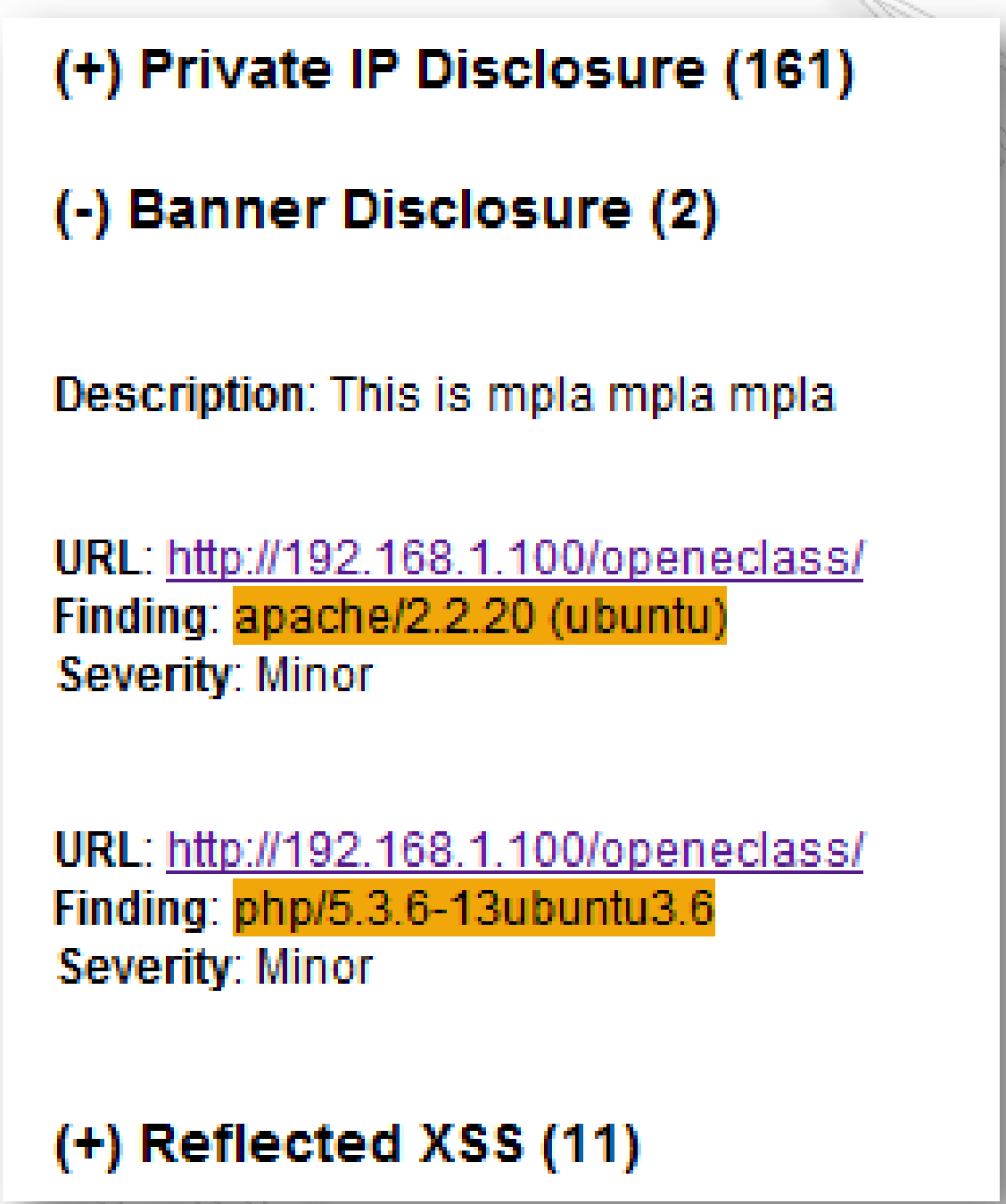
Εικόνα 3.4: Output WUI κατά τη διαδικασία του crawling



Εικόνα 3.5: Output WUI κατά τη διαδικασία του vulnerability assessment



Εικόνα 3.6: Σύνοψη αποτελεσμάτων του VAbB tool



Εικόνα 3.7: Ευρήματα για banner disclosure

(-) Reflected XSS (11)

Description: Reflected Cross-site Scripting (XSS) is another name for non-persistent XSS, where the attack doesn't load with the vulnerable web application but is originated by the victim loading the offending URI.

URL: http://192.168.1.100/openeclass/modules/auth/newuser.php?prenom_form=vabb<script>alert(123569);</script>&nom_form=&uname=&password1=&password=&email=&am=&auth=1&submit=Î�Î³Î¹Ï�Î±Î�Î®&department=1&localize=el&
Finding: prenom_form=vabb<script>alert(123569);</script>
Severity: High

URL: http://192.168.1.100/openeclass/modules/auth/newuser.php?prenom_form=&nom_form=vabb<script>alert(123569);</script>&uname=&password1=&password=&email=&am=&auth=1&submit=Î�Î³Î¹Ï�Î±Î�Î®&department=1&localize=el&
Finding: nom_form=vabb<script>alert(123569);</script>
Severity: High

URL: http://192.168.1.100/openeclass/modules/auth/newuser.php?prenom_form=&nom_form=&uname=vabb<script>alert(123569);</script>&password1=&password=&email=&am=&auth=1&submit=Î�Î³Î¹Ï�Î±Î�Î®&department=1&localize=el&
Finding: uname=vabb<script>alert(123569);</script>
Severity: High

URL: http://192.168.1.100/openeclass/modules/auth/newuser.php?prenom_form=&nom_form=&uname=&password1=&password=&email=vabb<script>alert(123569);</script>&am=&auth=1&submit=Î�Î³Î¹Ï�Î±Î�Î®&department=1&localize=el&
Finding: email=vabb<script>alert(123569);</script>
Severity: High

URL: http://192.168.1.100/openeclass/modules/auth/newuser.php?prenom_form=&nom_form=&uname=&password1=&password=&email=&am=vabb<script>alert(123569);</script>&auth=1&submit=Î�Î³Î¹Ï�Î±Î�Î®&department=1&localize=el&
Finding: am=vabb<script>alert(123569);</script>
Severity: High

URL: http://192.168.1.100/openeclass/modules/auth/newprof.php?nom_form=vabb<script>alert(123569);</script>&prenom_form=&userphone=&uname=&email_form=&submit=Î¥Î�Î¿Î³Î¿Î»Î®Î�Î¯Î·Î·Î·Î·Î·&auth=1&department=1&proflang=el&usercomment=&
Finding: nom_form=vabb<script>alert(123569);</script>
Severity: High

URL: http://192.168.1.100/openeclass/modules/auth/newprof.php?nom_form=&prenom_form=vabb<script>alert(123569);</script>&userphone=&uname=&email_form=&submit=Î¥Î�Î¿Î³Î¿Î»Î®Î�Î¯Î·Î·Î·Î·Î·&auth=1&department=1&proflang=el&usercomment=&
Finding: prenom_form=vabb<script>alert(123569);</script>
Severity: High

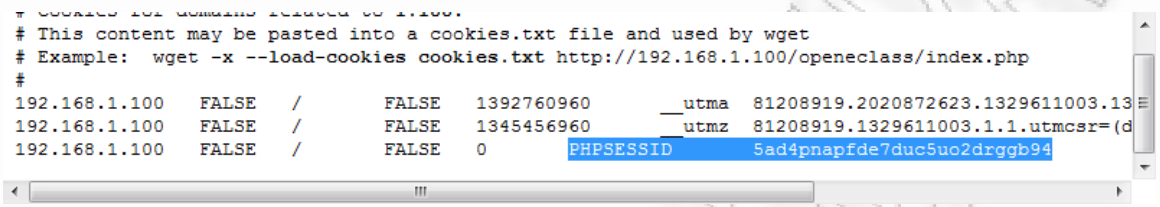
URL: http://192.168.1.100/openeclass/modules/auth/newprof.php?nom_form=&prenom_form=&userphone=vabb<script>alert(123569);</script>&uname=&email_form=&submit=Î¥Î�Î¿Î³Î¿Î»Î®Î�Î¯Î·Î·Î·Î·Î·&auth=1&department=1&proflang=el&usercomment=&
Finding: userphone=vabb<script>alert(123569);</script>

Εικόνα 3.8: Ευρήματα για reflected XSS

Για τα ίδια security modules θα επαναληφθεί ο έλεγχος αλλά αυτή τη φορά θα χρησιμοποιηθεί το cookie που δημιουργείται αφού γίνει login κάποιος χρήστης στο eclass. Τα groups που υποστηρίζει το openeclass είναι student, professor, administrator και guest. Σαν guest ο έλεγχος πραγματοποιήθηκε προηγουμένως. Τώρα θα γίνει έλεγχος ως student. Έτσι λοιπόν ανοίγουμε σε ένα browser την αρχική σελίδα του openeclass και βάζουμε τα credentials για ένα account που μας έχει δοθεί ώστε να γίνει το vulnerability assessment (εικόνα 3.9).

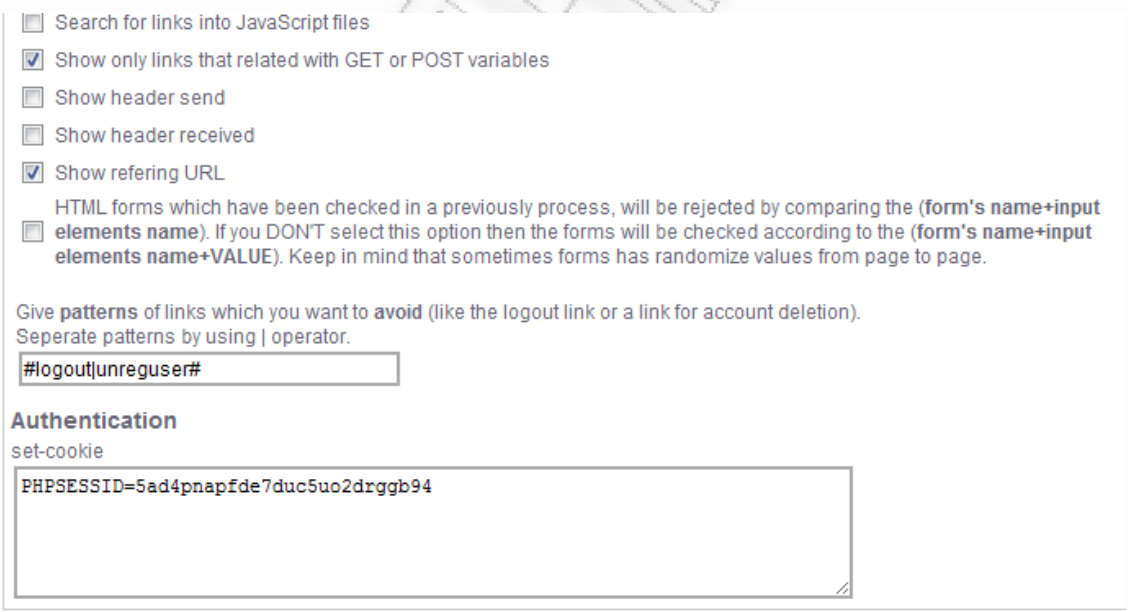
Εικόνα 3.9: Σύνδεση στο openeclass ως student

Αφού συνδεθούμε στο σύστημα πρέπει να βρούμε το session cookie και να το αντιγράψουμε στο εργαλείο. Έγινε χρήση ενός plugin του Google Chrome με το οποίο εύκολα μπορεί κάποιος να δει τα cookies μιας σελίδας. Στην εικόνα 3.10 φαίνεται επιλεγμένο το cookie που πρόκειται να αντιγραφεί στο VAbB.



Εικόνα 3.10: Αντιγραφή Session Cookie του openclass

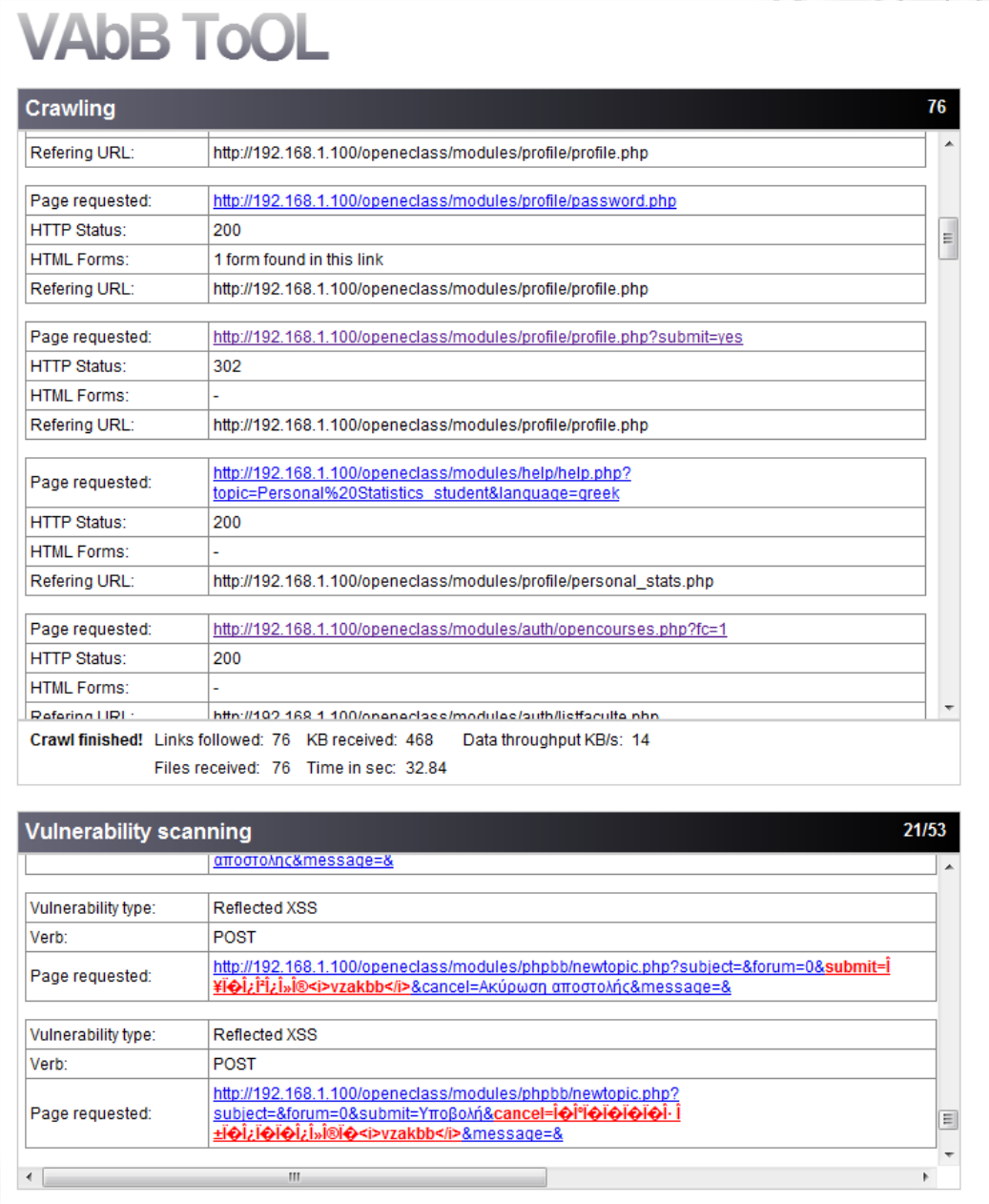
Στην εικόνα 3.11 φαίνεται το cookie αντιγραμμένο πλέον μέσα στο κατάλληλο πεδίο του VAbB. Επίσης δόθηκε και ένα pattern για να αποφύγει ο crawler το logout link αλλά και το link διαγραφής profile του χρήστη. Αν δεν δοθεί αυτό το pattern, είτε θα γίνει πρόωρη αποσύνδεση και έτσι ο crawler δεν θα συλλέξει όλα τα δεδομένα που θα έπρεπε, είτε θα διαγραφεί το profile του δοκιμαστικού χρήστη που δημιουργήθηκε για το σκοπό του assessment.



Εικόνα 3.11: Ορισμός authentication cookie και pattern για αποφυγή σελίδων

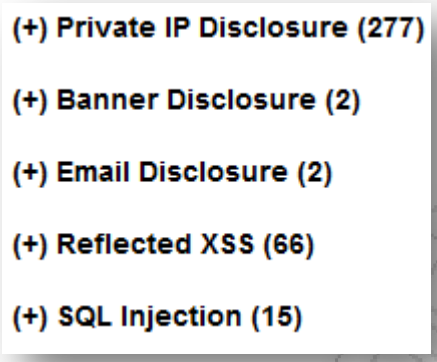
Στην εικόνα 3.12 βλέπουμε το εργαλείο να έχει ξεκινήσει την διαδικασία του assessment. Τα links που συλλέγονται πλέον είναι πολλά περισσότερα απ' ότι προηγουμένως που ο έλεγχος έγινε ως

guest user. Σαν guest ο crawler ακολούθησε 12 links ενώ σαν student user ακολούθησε 76 links. Το vulnerability scanning έγινε για 53 GET και POST links.



Εικόνα 3.12: Εκκίνηση του VAbB για έλεγχο του openeclass ως student user

Τα τελικά αποτελέσματα φαίνονται στην εικόνα 3.13.



Εικόνα 3.13: Τελικά αποτελέσματα του VAbB για τον έλεγχο στο openeclass ως student user

Τα vulnerabilities πλέον είναι πολλά περισσότερα σε σχέση με τον έλεγχο που έγινε ως guest user. Πλέον η σελίδα παρουσιάζει και SQL Injection vulnerability που είναι μια αρκετά κρίσιμη ευπάθεια. Αν ανοίξουμε στον browser το ευπαθές requested URL θα παρατηρήσουμε ένα error στην database, που παρουσιάστηκε λόγο κάποιου input που έδωσε το VAbB. Από τα 15 ευρήματα σε SQLi επιλέχθηκε ένα που έχει να κάνει με το forum που υποστηρίζει το openeclass. Στην εικόνα 3.14 φαίνεται το error που παρουσιάστηκε και εντόπισε το VAbB. Επίσης έχει επιλεχθεί και η GET μεταβλητή στην μπάρα του browser ώστε να φαίνεται το περιεχόμενο που δόθηκε και προκάλεσε το error.

Εικόνα 3.14: SQL Injection error

Κεφάλαιο 4

Οδηγός Εγκατάστασης Εργαλείου

Για την εγκατάσταση του παρόντος εργαλείου δεν έχει σημασία το λειτουργικό σύστημα του web server αρκεί να είναι εγκατεστημένος ο Apache και η PHP. Για χρήστες με desktop H/Y και λειτουργικό σύστημα windows προτείνεται το Xampp [17]. Η εγκατάσταση του Xampp περιέχει apache/php, mysql, filezilla για ftp server, Mercury για email server, phpMyAdmin για τη διαχείριση των βάσεων δεδομένων της MySql κ.α.

Καθ' όλη τη διάρκεια υλοποίησης του VAbB χρησιμοποιήθηκαν τα Ubuntu Server 10.04 [16] τα οποία εγκαταστάθηκαν σε Virtual Machine με χρήση του VirtualBox [15]. Επίσης χρησιμοποιήθηκε το extension PECL για την PHP το οποίο σε οποιαδήποτε Linux διανομή εγκαθίσταται πολύ εύκολα. Γι' αυτούς τους λόγους προτείνεται η εγκατάσταση του VAbB να γίνει σε κάποιο Linux Server μηχάνημα.

Μαζί με την εγκατάσταση των Ubuntu Server ήταν προεγκατεστημένο και το πακέτο LAMP το οποίο περιέχει ότι και το XAMPP σε Windows. Σε περίπτωση που χρησιμοποιηθεί διαφορετικό Linux based λειτουργικό σύστημα, μπορεί να αναζητηθεί στα repositories της εκάστοτε διανομής το LAMP και να εγκατασταθεί. Δεν χρειάζεται κάποια επιπλέον ρυθμίσει σε κάποιο configuration αρχείο όπως apache.conf ή http.conf, php.ini κλπ. Όλα λειτουργούν κανονικά απλά και μόνο με την εγκατάσταση.

Απαραίτητο extension για την PHP είναι το PECL-HTTP [6]. Μέσω αυτής της βιβλιοθήκης προσφέρεται δυνατότητα για http connections. Ουσιαστικά πρόκειται για έτοιμες κλάσεις που προσφέρουν αυτή τη δυνατότητα και είναι απαραίτητη για το εργαλείο VAbB. Το PECL **δεν** είναι προεγκατεστημένο στην βασική εγκατάσταση της PHP. Για την εγκατάσταση του PECL-HTTP extension αρκεί να

ακολουθηθούν τα παρακάτω βήματα για Ubuntu (τα ίδια πακέτα πρέπει να εγκατασταθούν και σε άλλες διανομές Linux, αλλά πιθανών να διαφέρουν οι εντολές):

1. Εγκαθιστούμε το πακέτο PEAR μέσω του οποίου εγκαθίστανται extensions της PECL:
   sudo apt-get install php-pear
2. Ύστερα χρειάζεται το πακέτο php5-dev για να εγκατασταθούν τα απαραίτητα php αρχεία που θα βοηθήσουν το compilation επεκτάσεων, όπως αυτή της PECL_HTTP:
   sudo apt-get install php5-dev
3. Εγκαθιστούμε το πακέτο libcurl3-openssl-dev:
   sudo apt-get install libcurl3-openssl-dev
4. Με την ακόλουθη εντολή εγκαθιστούμε το extension PECL_HTTP:
   sudo pecl install pecl_http
   Κατά τη διάρκεια της εγκατάστασης θα ζητηθούν από τον χρήστη ορισμένα inputs για τα οποία αφήνουμε τις default επιλογές πατώντας απλά το enter.
5. Αφού ολοκληρωθεί η εγκατάσταση με επιτυχία, ανοίγουμε το php.ini αρχείο και προσθέτουμε την ακόλουθη γραμμή κάτω από το section Dynamic Extensions:
   extension=http.so
   Το php.ini αρχείο συνήθως βρίσκεται στο ακόλουθω path:
   sudo nano /etc/php5/apache2/php.ini
6. Επανεκκινούμε τον apache για να εφαρμοστεί η νέα αλλαγή στο php.ini αρχείο:
   sudo /etc/init.d/apache2 restart ή sudo /etc/init.d/httpd restart

Πηγή των παραπάνω βημάτων εγκατάστασης του PECL_HTTP extension → http://www.mkfoster.com/2009/01/04/how-to-use-the-pecl-http-pecl_http-extension-to-make-http-requests-from-php/

Το IDE που χρησιμοποιήθηκε για τη συγγραφή του κώδικα της εφαρμογής είναι το eclipse Indigo IDE [18]. Επιπλέον εγκαταστάθηκε στο eclipse η επέκταση Aptana studio [19] που παρέχει δυνατότητα για συγγραφή PHP κώδικα. Το eclipse είναι γραμμένο σε Java και έτσι αποτελεί ένα multi – platform IDE. Έτσι αν κάποιος επιθυμεί να ελέγξει τον κώδικα της εφαρμογής η παραπάνω εγκατάσταση του eclipse και του plug-in προτείνεται, αλλά επειδή τα .php αρχεία είναι text-based, μπορεί κάποιος και με έναν απλό text editor να τα ανοίξει.

Συμπεράσματα και Μελλοντική Εργασία

Η παρούσα εργασία ολοκληρώθηκε με επιτυχία. Οι αρχικοί στόχοι ήταν να υλοποιηθεί ένα εργαλείο για vulnerability assessment το οποίο θα πραγματοποιεί ελέγχους για Information Disclosure, Reflected XSS, Stored XSS και SQL Injection. Τα αποτελέσματα είναι αρκετά θετικά και σε σύγκριση με άλλα δοκιμασμένα εργαλεία δεν απέχει πολύ με εξαίρεση τον έλεγχο για SQL Injection για τον οποίο έχουν προγραμματιστεί προσθήκες στην επόμενη version του VAbB.

Παρόλα αυτά υπάρχουν μερικές παρατηρήσεις από πλευράς υλοποίησης του εργαλείου. Η πρώτη και βασική παρατήρηση έχει να κάνει με τη ροή εκτέλεσης του εργαλείου. Τα δύο μέρη του εργαλείου crawling και vulnerability scanning process εκτελούντα χωριστά. Δηλαδή πρώτα εκτελείται το crawling και αφού συλλεχτούν τα δεδομένα που χρειάζονται, τότε συνεχίζει στο vulnerability scanning. Αυτό έχει ένα σοβαρό μειονέκτημα το οποίο είναι ότι το crawling για μεγάλες διαδικτυακές εφαρμογές μπορεί να μην ολοκληρωθεί σε εύλογο χρονικό διάστημα ή και ποτέ, επειδή μπορεί να μην ανταποκρίνεται ο browser μετά από κάποια ώρα εκτέλεσης του VAbB. Γι' αυτό το λόγω στην παρούσα φάση έχει δημιουργηθεί μια επιλογή "Page Limit" μέσω της οποίας μπορεί ο tester να επιλέξει πόσες σελίδες θέλει να κάνει crawling. Αυτό φυσικά δεν είναι και η καλύτερη δυνατή λύση γιατί μετά από κάποιο αριθμό σελίδων μπορεί να υπάρχει κάποια σοβαρή ευπάθεια την οποία να μην ανακαλύψουμε ποτέ.

Το σωστό λοιπόν θα ήταν κατά τη διάρκεια του crawling να πραγματοποιείται και έλεγχος για τις ευπάθειες XSS και SQLi. Έτσι θα μπορεί να διακοπεί ανά πάσα στιγμή το εργαλείο και θα είναι γνωστό ότι τα μέχρι εκείνη τη στιγμή, τα links που έγιναν crawl, ελέγχθηκαν και είτε υπήρχαν ευρήματα τα οποία και εμφανίστηκαν στην σελίδα output.php, είτε δεν υπήρχαν εφόσον δεν εμφανίστηκαν.

Η καλύτερη μέθοδος υλοποίησης ταυτόχρονου ελέγχου διαφορετικών διαδικασιών είναι με χρήση threads. Εδώ όμως έρχεται ένα δεύτερο πρόβλημα που έχει να κάνει με την γλώσσα που υλοποιήθηκε το VAbB. Η PHP δεν υποστηρίζει threads μέχρι και την τελευταία έκδοση που κυκλοφορεί. Υπάρχει ένα extension της PHP (πηγή) το οποίο όμως δεν είναι supported και είναι καθαρά πειραματικό. Έτσι λοιπόν στην προκειμένη φάση δεν μπορούμε να πούμε με σιγουριά ότι με το συγκεκριμένο extension δεν μπορεί να υλοποιηθεί αυτή η παράλληλη εκτέλεση crawler και scanner, αλλά σίγουρα όταν δεν υποστηρίζεται μια native βιβλιοθήκη αποτελεί από μόνο του ένα πρόβλημα.

Αυτό είναι το βασικό μειονέκτημα της υλοποίησης μέχρι στιγμής. Το πρόβλημα ελαχιστοποιείται με την επιλογή page limit αλλά και με την αποφυγή σελίδων μέσω του pattern που είδαμε σε προηγούμενα κεφάλαια. Όπως προαναφέρθηκε όμως, αυτές οι δυο επιλογές δεν λύνουν το πρόβλημα.

Για μελλοντική εργασία έχουν προγραμματιστεί τα εξής κατά προτεραιότητα εμφάνισης:

1. Να ελεγχθεί το extension της php που παρέχει δυνατότητα για δημιουργία threads ώστε να υλοποιηθεί με κάποιο τρόπο η παράλληλα εκτέλεση crawler και scanner.
2. Το κομμάτι κώδικα του crawler που πραγματοποιεί http connections έχει υλοποιηθεί με sockets. Αντικατάσταση αυτού του κώδικα με χρήση των κλάσεων PECL_HTTP.
3. Βελτίωση του γραφικού περιβάλλοντος στην index.php και output.php. Προσθήκη JQuery ίσως και Ajax όπου χρειάζεται.
4. Προσθήκες στον έλεγχο για SQL Injection ώστε να έχουμε λιγότερα false positives.
5. Προσθήκη περισσότερων ελέγχων ασφάλειας. Πρώτος στόχος είναι να υλοποιηθούν έλεγχοι για όλες τις επιθέσεις που αναφέρονται στο OWASP Top 10 [21] και ύστερα να συνεχιστούν οι προσθήκες.

Διαδικτυακές Αναφορές

1. PHP Language - *http://www.php.net/*
2. PHP Crawler - *http://phpcrawl.cuab.de/*
3. Regular Expressions - *http://weblogtoolscollection.com/regex/regex.php*
4. HTTP Protocol - *http://www.w3.org/Protocols/rfc2616/rfc2616.html*
5. Cookies - *http://en.wikipedia.org/wiki/HTTP_cookie*
6. PECL HTTP Extension - *http://pecl.php.net/package/pecl_http*
7. Web Vulnerability Scanners - *http://sectools.org/tag/web-scanners/*
8. Burp Suite - *http://portswigger.net/burp/*
9. w3af - *http://w3af.sourceforge.net/*
10. OWASP ZAP - *https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project*
11. WebSecurify - *http://www.websecurify.com/*
12. Information Gathering - *http://www.milescan.com/milescan-help/1.8.0/index.html?information_gathering.htm*
13. Cross Site Scripting - *https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)*
14. SQL Injection - *https://www.owasp.org/index.php/SQL_Injection*
15. VirtualBox - *https://www.virtualbox.org/*
16. Ubuntu Server - *http://www.ubuntu.com/business/server/overview*
17. XAMPP - http://www.apachefriends.org/en/index.html
18. Eclipse - *http://www.eclipse.org/*
19. Aptana Studio, Eclipse plugin - *http://aptana.com/*
20. Open eclass - *http://www.openeclass.org/*
21. OWASP Top 10 - *https://www.owasp.org/images/0/0f/OWASP_T10_-_2010_rc1.pdf*