Π Α Ν Ε Π Ι Σ Τ Η Μ Ι Ο Π Ε Ι Ρ Α Ι Ω Σ

Τμήμα Ψηφιακών Συστημάτων

Μεταπτυχιακή Διπλωματική Εργασία

[Αξιολόγηση ευπαθειών και τρόποι αντιμετώπισης τους σε διαδικτυακές εφαρμογές και εξυπηρετητές]

[Μακρής Χαράλαμπος]

Α.Μ. [ΜΕ09060]

Επιβλέπων: [Λαμπρινουδάκης Κωνσταντίνος, Επίκουρος Καθηγητής]

Περιεχόμενα

Ευρετήριο Εικόνων

Ευρετήριο Πινάκων

Πρόλογος

Η παρούσα μεταπτυχιακή διπλωματική εργασία πραγματοποιήθηκε στα πλαίσια του Προγράμματος Μεταπτυχιακών Σπουδών «Διδακτικής της Τεχνολογίας και Ψηφιακά Συστήματα» του τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς. Η υλοποίησή της ξεκίνησε τον Μάιο του 2011 και ολοκληρώθηκε τον Νοέμβριο του ίδιου έτους. Στην εκπόνηση της εργασίας έπαιξε σημαντικό ρόλο το περιβάλλον στο οποίο εργάζομαι, καθώς είμαι μέλος της ομάδας της Τεχνικής Υποστήριξης του τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς, και έτσι είχα την ευκαιρία να μελετήσω ένα θέμα το οποίο έχει απασχολήσει την ομάδα εργασίας μου καθώς και τον υπεύθυνο καθηγητή μου. Μου δόθηκε η δυνατότητα λόγω της πρότερης γνώσης στο ζήτημα το οποίο μελέτησα να εκμεταλλευτώ την εσωτερική πληροφόρηση από την ομάδα εργασίας μου προς όφελος της διπλωματικής μου εργασίας και προς αποφυγή της πρόκλησης κάποιου προβλήματος στην ασφάλεια του ιστότοπου του τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς που αποτελεί μια κρίσιμη εφαρμογή. Θα ήθελα να ευχαριστήσω προσωπικά τον υπεύθυνο καθηγητή μου κ. Λαμπρινουδάκη, που τυγχάνει να είναι και υπεύθυνος της ομάδας εργασίας μου, ο οποίος έπαιξε πολύ σημαντικό ρόλο σε σχέση τόσο με την επιλογή ενός τόσο κρίσιμου θέματος προς μελέτη όσο και για την πολύτιμη καθοδήγησή του κατά τις διάφορες φάσεις ανάπτυξης και υλοποίησης της μεταπτυχιακής διπλωματικής εργασίας αυτής. Επίσης ιδιαιτέρως ξεχωριστό ρόλο έπαιξε η βοήθεια και η άψογη συνεργασία που είχα με τον κ. Γιώργο Πάφιο, μέλος της ομάδας της Τεχνικής Υποστήριξης του τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς, σε θέματα διάθεσης, συλλογής και επεξεργασίας των στοιχείων της εργασίας. Στα κριτήρια επιλογής του συγκεκριμένου θέματος συμπεριλαμβάνεται και το να αποτελέσει αυτή η διπλωματική εργασία ένα χρήσιμο εργαλείο μελέτης και βοήθειας για το τμήμα Ψηφιακών Συστημάτων και τους εργαζόμενους στον IT τομέα του (developers, τεχνική υποστήριξη).

Περίληψη

Όπως έχει αναγνωριστεί από τη διεθνή βιβλιογραφία όσο εξελίσσονται οι εφαρμογές ιστού (Web applications), τόσο το πρόβλημα της ασφάλειας γίνεται πολυσύνθετο. Οι πιθανές αδυναμίες, που υπάρχουν σε αυτές μπορεί να επιτρέψουν σε επίδοξους εισβολείς να παραβιάσουν την ασφάλεια του Πληροφοριακού Συστήματος ενός οργανισμού. Είναι ελάχιστες οι εταιρείες, οι οποίες στην προσπάθεια τους να διαθέτουν συνεχώς νέο λογισμικό εφαρμογών στην αγορά, κάνουν επισταμένους ελέγχους ασφαλείας στα προϊόντα τους. Έτσι, ακόμη και αν υπάρχει προστασία με τη χρήση των καλύτερων συμβατικών συστημάτων ασφάλειας, ο κάθε οργανισμός είναι ευάλωτος σε επιθέσεις μέσω των εφαρμογών ιστού. Οι προγραμματιστές και οι υπεύθυνοι ασφαλείας οφείλουν να είναι ικανοί να ανιχνεύουν την ύπαρξη και τη σοβαρότητα των αδυναμιών που υπάρχουν στις εφαρμογές ιστού και να προτείνουν τα κατάλληλα μέτρα, που θα παρέχουν προστασία από πιθανές παραβιάσεις ασφάλειας. Μια τυπική διαδικασία περιλαμβάνει τον έλεγχο ασφάλειας όλων των εφαρμογών, που τρέχουν σε όλα τα συνδεδεμένα με τα Web συστήματα και την εξέταση κάθε γραμμής κώδικα των εφαρμογών για πιθανές αδυναμίες ασφαλείας.

Αντικείμενο της παρούσας μεταπτυχιακής διπλωματικής εργασίας αποτελεί η μελέτη και η διερεύνηση ευπαθειών και αδυναμιών εφαρμογών παγκόσμιου ιστού και η πειραματική εφαρμογή εναλλακτικών εργαλείων και μεθόδων ελέγχου ασφάλειας στα πλαίσια μιας μεθοδολογικής προσέγγισης για την αυτόματη ή μη αξιολόγηση της ασφάλειας των εφαρμογών ιστού. Σε αυτή την εργασία εξετάζονται και οι έλεγχοι οι οποίοι επιβάλλεται να γίνονται από τους υπεύθυνους ανάπτυξης των εφαρμογών και τους υπεύθυνους ασφαλείας των οργανισμών- επιχειρήσεων, που χρησιμοποιούν τέτοιου είδους εφαρμογές. Η μεθοδολογία της έρευνας περιλαμβάνει μεταξύ άλλων την εξοικείωση με τις σχετικές έννοιες και το κανονιστικό πλαίσιο, που έχει ήδη ορισθεί από αναγνωρισμένους οργανισμούς, και τη μελέτη των οδηγιών και προτύπων ασφάλειας εφαρμογών στο Web.

Συγκεκριμένα μελετήθηκαν τα προβλήματα και οι απειλές ασφάλειας στο Web και έγινε προσπάθεια να αναδειχθούν οι αδυναμίες, καθώς και μια σειρά από μέτρα προφύλαξης. Παράλληλα διερευνήθηκαν και ταξινομήθηκαν σχετικές μέθοδοι, κριτήρια, εργαλεία αξιολόγησης και ελέγχου, που υπάρχουν και είναι είτε διαθέσιμα για ελεύθερη χρήση, είτε εμπορικά. Στα πλαίσια ενός σχεδίου μέτρησης πραγματοποιήθηκαν μετρήσεις ασφάλειας εφαρμογών ιστού με πειραματική εφαρμογή των εργαλείων και των μεθόδων αξιολόγησης. Συγκεκριμένα έγινε χρήση του εργαλείου ανοιχτού λογισμικού WebScarab, με το οποίο αυτοματοποιούνται τα στάδια ελέγχου ασφάλειας των εφαρμογών ιστού με τη μέθοδο της δοκιμής διείσδυσης.

Σκοπός της εργασίας αυτής είναι η εξοικείωση με τις σχετικές έννοιες και το κανονιστικό πλαίσιο σε θέματα ασφάλειας, που έχει ήδη ορισθεί από αναγνωρισμένους οργανισμούς, όπως είναι ο CERT και ο OWASP (Open Web Application Security Project – ένας παγκόσμιος οργανισμός που έχει ως σκοπό την βελτίωση της ασφάλειας του λογισμικού εφαρμογών) κ.α., καθώς και η μελέτη των οδηγιών και προτύπων ασφάλειας για την ανάπτυξη και έλεγχο των εφαρμογών ιστού. Επίσης, είναι η διερεύνηση και χρησιμοποίηση διαφόρων εργαλείων, είτε είναι ελεύθερης χρήσης είτε εμπορικά, για τον έλεγχο γνωστών

ευπαθειών των εφαρμογών ιστού και η αναγνώριση της χρησιμότητας τους από τους υπεύθυνους ασφάλειας, για το συνεχή έλεγχο των εφαρμογών. Με διάφορα τέτοια εργαλεία και κυρίως με τη χρήση του WebScarab το οποίο αποτελεί ένα εργαλείο ανοιχτού λογισμικού του OWASP, έγινε μια προσπάθεια συνολικής μέτρησης της ασφάλειας του ιστότοπο(website) του Τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς (www.ds.unipi.gr) στα πλαίσια ενός οδηγού ελέγχου ασφάλειας εφαρμογών ιστού του OWASP (OWASP Testing Guide 2008 V3.0). Επιπλέον χρησιμοποιήθηκαν τα εργαλεία Netcraft και Zenmap (Nmap) για τον εντοπισμό, συλλογή και ανάλυση πληροφοριών σχετικά με διάφορες εφαρμογές ιστού, που υπάρχουν σε κάποιο συγκεκριμένο web server-στόχο, καθώς και τα εργαλεία Integrigy, Brutus και Netcat τα οποία αυτοματοποιούν διάφορους ελέγχους ασφάλειας εφαρμογών ιστού.

Από την εργασία αυτή εξάγονται τα ακόλουθα βασικά συμπεράσματα:

- Η μέθοδος της δοκιμής διείσδυσης με την τεχνική της Black Box προσέγγισης αποτελεί την αποτελεσματικότερη μέθοδο για την μέτρηση της ασφάλειας των εφαρμογών ιστού, αφού αξιολογείται η ασφάλεια ακολουθώντας τεχνικές και βήματα, που χρησιμοποιούν οι εισβολείς για την πραγματοποίηση των επιθέσεών τους.
- Είναι απαραίτητη η χρήση εργαλείων για την πραγματοποίηση των ελέγχων, είτε πρόκειται για αυτοματοποιημένους, είτε όχι, λόγω της απαίτησης για συνεχείς και επαναλαμβανόμενους ελέγχους.
- Η επιλογή των εργαλείων, που θα χρησιμοποιηθούν κατά τους ελέγχους αποτελεί καθοριστικό παράγοντα για την όλη διαδικασία μέτρησης της ασφάλειας, αλλά και για την αντικειμενικότητα των αποτελεσμάτων.

Στα επιτεύγματα της παρούσας εργασίας συμπεριλαμβάνονται πέρα από τη μέτρηση της ασφάλειας του συγκεκριμένου ιστότοπου, η διερεύνηση των δυνατοτήτων διαφόρων εργαλείων για τον έλεγχο συγκεκριμένων ευπαθειών, καθώς και η παράθεση διαφόρων σημαντικών λειτουργιών τους. Από την όλη προσπάθεια συμπεραίνουμε ότι τα εργαλεία αυτά μπορεί να κάνουν πολύ πιο εύκολη τη λειτουργία του ελέγχου των εφαρμογών ιστού, αλλά δεν θα πρέπει να θεωρούνται και πανάκεια για την επίλυση του προβλήματος του ελέγχου ασφάλειας των εφαρμογών αυτών. Το τελευταίο είναι ίσως και το πιο σημαντικό στοιχείο, που πρέπει να έχει κατά νου κάποιος, που θα ασχοληθεί με τον έλεγχο ασφάλειας των εφαρμογών ιστού.

Τα εργαλεία, που χρησιμοποιήθηκαν στην μέτρηση ασφάλειας του συγκεκριμένου ιστότοπου, είναι κυρίως εργαλεία ελεύθερης χρήσης και ανοικτού λογισμικού. Ένας περιορισμός της παρούσας εργασίας είναι το γεγονός ότι αρκετά εργαλεία απαιτούν προχωρημένες γνώσεις προγραμματισμού και μεγάλη εμπειρία στο αντικείμενο της ασφάλειας εφαρμογών, το οποίο δυσκόλεψε την έρευνα και περιόρισε σε κάποιο βαθμό όχι μόνο το φάσμα των ελέγχων αλλά και τη χρήση των εργαλείων.

Η δομή της εργασίας έχει ως εξής: Στο κεφάλαιο 1 (Θεωρητικό Υπόβαθρο) γίνεται μια αναφορά σε βασικούς ορισμούς ασφάλειας, στις σημαντικότερες ευπάθειες των εφαρμογών ιστού και παρουσιάζεται η μεθοδολογία, που χρησιμοποιήθηκε για τους ελέγχους, που πραγματοποιήθηκαν. Στο

κεφάλαιο 2 (Παρουσίαση του εργαλείου WebScarab), γίνεται μια εκτενέστερη παρουσίαση των λειτουργιών και δυνατοτήτων του κύριου εργαλείου που χρησιμοποιείται για τους ελέγχους. Στο κεφάλαιο 3 (Μετρήσεις και Ανάλυση Στοιχείων) παρουσιάζεται η χρήση των εργαλείων που επιλέξαμε για τη μέτρηση ασφάλειας του ιστότοπου παραθέτοντας ελέγχους και αποτελέσματα. Η λογική που ακολουθείται εδώ είναι να καταγράψουμε τους τρόπους που προτείνει ο οργανισμός OWASP στο Testing Guide και αναλόγως να επιλέξουμε στην δική μας περίπτωση να εφαρμόσουμε ότι ταιριάζει στην εφαρμογή μας ή ότι μπορούμε βάση γνώσεων να εφαρμόσουμε καλύτερα ως μέθοδο ελέγχου. Στο κεφάλαιο 4 (Αποτελέσματα και Συμπεράσματα από τους ελέγχους) παραθέτονται συγκεντρωτικά τα όλα τα στοιχεία που προέκυψαν από τη διαδικασία των ελέγχων.

Κεφάλαιο 1ο : [Θεωρητικό Υπόβαθρο]

1.1 Εισαγωγή

Καθώς το Internet αναπτύσσεται και εξελίσσεται συνεχώς, όλο και περισσότεροι οργανισμοί και επιχειρήσεις αντικαθιστούν τους απλούς ιστότοπους με κρίσιμες εφαρμογές σχεδιασμένες κατά τέτοιο τρόπο, που να επιτυγχάνουν την ενοποίηση/ενσωμάτωση των υπαρχόντων συστημάτων τους και την παροχή καλύτερων υπηρεσιών προς τους πελάτες τους. Έτσι λοιπόν σήμερα υπάρχει πληθώρα εμπορικών ιστοσελίδων για αγορά προϊόντων και παροχή κάθε είδους υπηρεσιών, στις οποίες λειτουργούν εφαρμογές ιστού και υπηρεσίες, οι οποίες είναι οι βασικοί συντελεστές του Internet επόμενης γενιάς ή αλλιώς Internet2. Το Internet2 είναι μια νέα σειρά τεχνολογιών υποδομής και εφαρμογών, που έχουν ως σκοπό την ανάπτυξη εξελιγμένων εφαρμογών, που θα καθιστούν ικανή τη συνεργασία μεταξύ ανθρώπων και θα παρέχουν αλληλεπιδραστική πρόσβαση σε πληροφορίες και πηγές με τέτοιο τρόπο, ο οποίος δεν είναι εφικτός με την τωρινή μορφή του Internet.

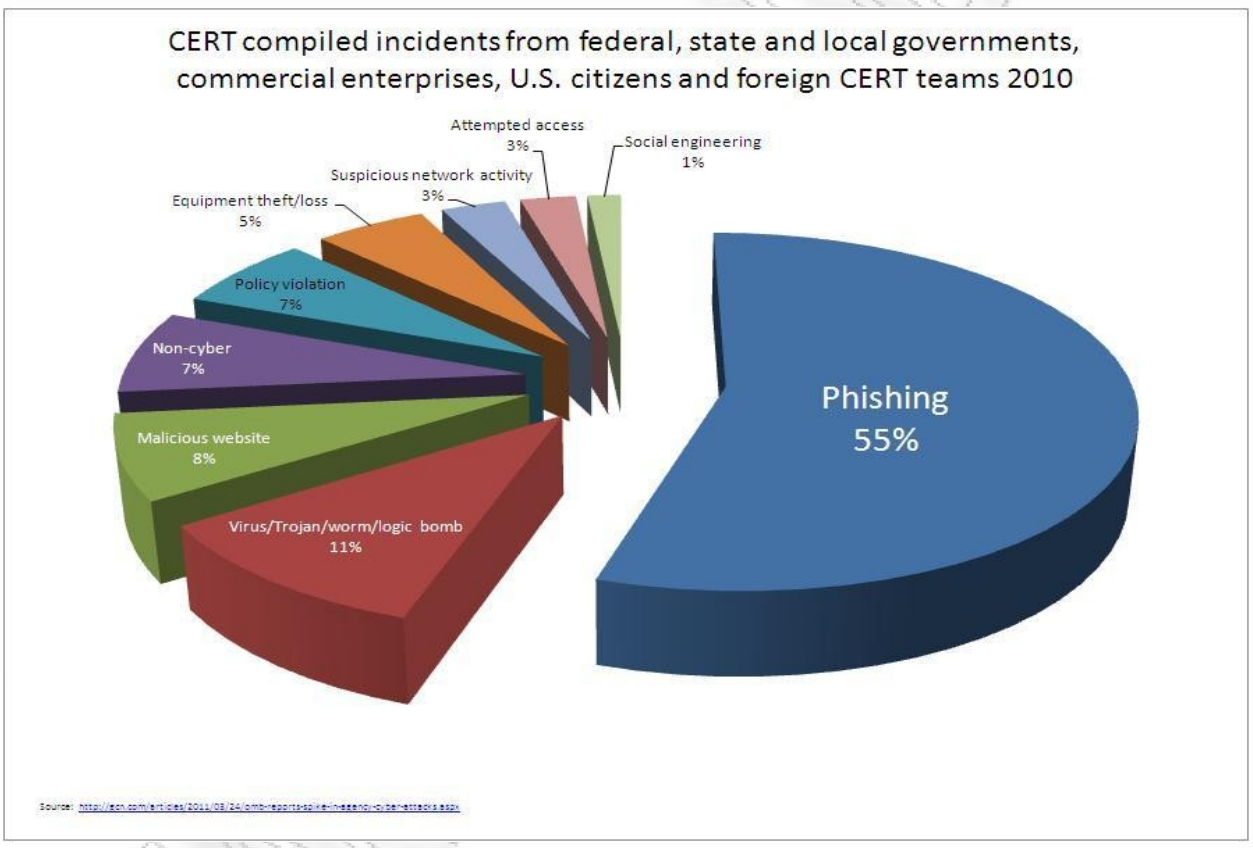
Το Internet αρχικά σχεδιάστηκε με κύριο στόχο να είναι «ανοικτό». Αυτό σήμαινε ότι ο καθένας θα μπορούσε να έχει δυνατότητα πρόσβασης σε πληροφορίες, που δημοσιεύονται στο internet και για αυτό το λόγο δεν θα υπήρχε η ανάγκη προστασίας τους. Η φύση όμως των δεδομένων και καθώς και το περιεχόμενό τους πολλές φορές καθιστούν απαραίτητη την προστασία τους όταν αυτά μπορούν να γίνουν εύκολα προσπελάσιμα μέσω του παγκόσμιου ιστού. Γι' αυτό το λόγο οι περισσότεροι οργανισμοί και επιχειρήσεις, που έχουν εγκαταστήσει και λειτουργούν εφαρμογές ιστού, ανακαλύπτουν συνεχώς ότι είναι εκτεθειμένοι σε νέους κινδύνους ως προς τη σωστή λειτουργία των συστημάτων τους και κυρίως ως προς την ασφάλεια αυτών με συνέπεια τις επιπτώσεις στην ιδιωτικότητα των χρηστών και στην προστασία των αγαθών.

Αυτός ο φόβος είναι δικαιολογημένος, αν εξετάσει κάποιος τα στατιστικά στοιχεία του Συντονιστικού Κέντρου CERT - Computer Emergency Response Team Coordination Center, ενός οργανισμού που χρηματοδοτείται από την κυβέρνηση των Ηνωμένων Πολιτειών, και ο οποίος έχει ως σκοπό την πληροφόρηση επί θεμάτων ασφάλειας στην κοινωνία του διαδικτύου (internet) και την ενίσχυση της άμεσης και αποτελεσματικής συνεργασίας μεταξύ ειδικών σε καταστάσεις επείγουσας ανάγκης - σύμφωνα με τα οποία οι κυβερνοεπιθέσεις στις υπηρεσίες και τους οργανισμούς αυξάνονται όσο η ετοιμότητα τους σε θέματα ασφάλειας υστερεί.

Η έκθεση του Γραφείου Διαχείρισης και Προϋπολογισμού αναφέρει ότι τα περιστατικά στον κυβερνοχώρο αυξήθηκαν 39 τοις εκατό κατά το τελευταίο οικονομικό έτος χρήσης. Οι κρατικές υπηρεσίες παρατήρησαν μια απότομη αύξηση κατά το τελευταίο οικονομικό έτος χρήσης στα περιστατικά ασφάλειας που συμβαίνουν στον κυβερνοχώρο , τα οποία αυξήθηκαν κατά 39 τοις εκατό το 2009, σύμφωνα με την ετήσια έκθεση από το Γραφείο Διαχείρισης και Προϋπολογισμού. Τριάντα τοις εκατό αυτών των περιστατικών ήταν κακόβουλες επιθέσεις κώδικα. Στην ετήσια έκθεση του Γραφείου Διαχείρισης και Προϋπολογισμού σχετικά με την εφαρμογή του ομοσπονδιακού Νόμου του 2002 περί της Διαχείρισης Ασφάλειας Πληροφοριών αναφέρθηκαν 41.776 ομοσπονδιακά περιστατικά σε 24

γραφεία το 2010, σε σύγκριση με 30.000 περιστατικά το 2009. «Κακόβουλος κώδικας μέσω πολλαπλών μέσων (π.χ. phishing, ιοί, λογική βόμβα) εξακολουθεί να αποτελεί την πιο ευρέως χρησιμοποιούμενη προσέγγιση επίθεσης» όπως αναφέρει η έκθεση.

Εκτός από κακόβουλο κώδικα, σχεδόν το 14 τοις εκατό που εμπλέκεται με μη εξουσιοδοτημένη πρόσβαση, το 18 τοις εκατό με ανάρμοστη χρήση και το 27 τοις εκατό παρατίθεται ως υπό έρευνα ή κάτι άλλο. Έντεκα τοις εκατό που περιλαμβάνει σαρώσεις, ανιχνεύσεις και προσπάθεια πρόσβασης και 0,1 τοις εκατό ήταν denial of service επιθέσεις. Η Ομάδα Πληροφορικής Επείγουσας Ετοιμότητας των ΗΠΑ(US-CERT) κατάρτισε περιστατικά από ομοσπονδιακές, πολιτειακές και τοπικές κυβερνήσεις, εμπορικές επιχειρήσεις, πολίτες των ΗΠΑ και ξένες ομάδες CERT. Το 2010, ο οργανισμός έλαβε συνολικά 107.439 αναφορές και 108.710 το 2009. Περίπου το 53 τοις εκατό των αναφερθέντων περιστατικών το 2010 ήταν phishing επιθέσεις. Η κατανομή των εκθέσεων του US-CERT φαίνεται στην εικόνα παρακάτω:



Εικόνα : Η κατανομή των εκθέσεων με βάση τα αποτελέσματα από το US-CERT.

- Phishing: 56,579 περιστατικά (52.7 %)
- Ιοί/Δούρειος Ίππος(Trojan)/Σκουλήκια(warms)/λογικές βόμβες(logic bombs): 11,001 περιστατικά (10.2%).
- Κακόβουλη Ιστοσελίδα: 7,971 περιστατικά (7.4%)
- Μη-κυβερνοχώρου(Non-cyber): 7,741 περιστατικά (7.2%)
- Παραβίαση Πολιτικής Ασφάλειας: 6,888 περιστατικά (6.4%)

- Κλοπή Εξοπλισμού/Απώλεια: 5,395 περιστατικά (5%)
- Ύποπτη δικτυακή δραστηριότητα: 3,121 περιστατικά (2.9%)
- Προσπάθειες ανάκτησης πρόσβασης: 2,712 περιστατικά (2.5%)
- Social engineering: 1,571 περιστατικά (1.5 %)

Παρά το γεγονός ότι μόνο το 1,5 τοις εκατό των συνολικών αναφερθεισών επιθέσεων περιελάμβαναν social engineering, η έκθεση του Γραφείου Διαχείρισης και Προϋπολογισμού σημειώνει ότι οι επιθέσεις αυτές είναι που προκαλούν ιδιαίτερη ανησυχία, καθώς εκμεταλλεύονται κώδικες που συχνά γίνονται διαθέσιμες στο κοινό. "Υπήρξαν επανειλημμένες επιθέσεις σε zero-day ευπάθειες(άγνωστες ευπάθειες) μέσω του social engineering. Αυτές οι επιθέσεις συνήθως απαιτούν social engineering για να ξεγελάσουν τους χρήστες να επισκεφτούν επικίνδυνες ιστοσελίδες που φιλοξενούν κακόβουλα προγράμματα ή να ανοίξουν ένα κακόβουλο επισυναπτόμενο για την εκτέλεση του κακόβουλου λογισμικού στο δικό τους περιβάλλον", αναφέρει η έκθεση . Επιπρόσθετα, η έκθεση διαπίστωσε «φτωχή» ομοσπονδιακή συμμόρφωση με τις κατευθυντήριες γραμμές της ασφάλειας πληροφοριών. "Μόνο μία υπηρεσία έλαβε βαθμολογία συμμόρφωσης 100 τοις εκατό για το πρόγραμμα ασφάλειας των πληροφοριών της, η οποία, με βάση την αναθεώρηση του IG της, συνάντησε όλα τα 62 χαρακτηριστικά", αναφέρει η έκθεση. "Οι υπόλοιπες υπηρεσίες είχαν τουλάχιστον μία περιοχή που χρειάζεται βελτίωση. Τρεις οργανισμοί δεν είχαν ούτε ένα πρόγραμμα ασφάλειας του κυβερνοχώρου σε ισχύ για έναν τομέα της ασφάλειας και ένα γραφείο δεν είχε έστω ένα πρόγραμμα σε ισχύ για δύο τομείς της ασφάλειας".

Το Γραφείο Διαχείρισης και Προϋπολογισμού χρησιμοποίησε τους τομείς της ανεπάρκειας για να υπολογίσει τις βαθμολογίες των οργανισμών. Έξι οργανισμοί πέτυχαν βαθμολογία πάνω από 90 τοις εκατό της συμμόρφωσης, οκτώ στο 65 με 90 τοις εκατό, και οι υπόλοιποι εννέα πέτυχαν βαθμολογία μικρότερη του 65 τοις εκατό, σύμφωνα με την έκθεση. Η έκθεση διαπίστωσε επίσης ότι πολλοί οργανισμοί δεν εκπαιδεύουν επαρκώς το προσωπικό τους στην ασφάλεια στον κυβερνοχώρο. Εξειδικευμένη εκπαίδευση ασφάλειας στον κυβερνοχώρο για τους χρήστες μιας υπηρεσίας με σημαντικές ευθύνες για την ασφάλεια των πληροφοριών υπολογίστηκε κατά μέσο όρο στο 88 τοις εκατό. Μία υπηρεσία παρείχε μόνο στο 2 τοις εκατό των χρηστών της τέτοιου είδους εκπαίδευση. Για τους νέους εργαζομένους, κατά μέσο όρο 73 τοις εκατό δόθηκε κατάρτιση για την ευαισθητοποίηση σε θέματα ασφάλειας πριν τους επιτραπεί η πρόσβαση στο δίκτυο. Δύο οργανισμοί δεν εκπαίδευσαν κανέναν από τους νέους υπαλλήλους τους, δύο εκπαίδευσαν μεταξύ πέντε και έξι τοις εκατό των υπαλλήλων τους και μία υπηρεσία εκπαίδευσε το ήμισυ περίπου του εισερχόμενου το προσωπικό της κατά 55 τοις εκατό. Η έκθεση του Γραφείου Διαχείρισης και Προϋπολογισμού απηχεί σε παλαιότερες εκθέσεις και αναφορές για την κατάσταση της ασφάλειας του κυβερνοχώρου στην κυβέρνηση. Ο Gregory Wilshusen, διευθυντής του Γραφείου Κυβερνητικής Ευθύνης επί των θεμάτων ασφάλειας των πληροφοριών, δήλωσε πρόσφατα σε υποεπιτροπή εσωτερικής ασφάλειας ότι οι Ηνωμένες Πολιτείες δεν είναι έτοιμη για «εν δυνάμει καταστροφικές» επιθέσεις στον κυβερνοχώρο.

Τον περασμένο μήνα, ο Διευθυντής συστημάτων πληροφορικής του τμήματος Υποθέσεων Απόστρατων Roger Baker πρότεινε τη βελτίωση της ασφάλειας με τη συγκέντρωση της διοίκησης, δηλώνοντας ότι η κυβέρνηση υστερεί σε σχέση με τον ιδιωτικό τομέα σε αυτό το θέμα εξαιτίας της οργάνωσής της. Ο Baker μίλησε σε μια συνάντηση κορυφής για την ασφάλεια του κυβερνοχώρου στην Ουάσιγκτον που φιλοξενήθηκε από το FedScoop. Και τον Ιανουάριο το Εθνικό Ινστιτούτο Ασφάλειας Κυβερνοχώρου απέδωσε τους βαθμούς στην κυβέρνηση Ομπάμα οι οποίοι κυμαίνονταν από Β έως D για τις πολιτικές ασφάλειας του κυβερνοχώρου της. Τα τμήματα που καλύπτονται στην έκθεση περιλαμβάνουν αυτά της Γεωργίας, του Εμπορίου, της Εθνικής Άμυνας, της Παιδείας, της Υγείας και Ανθρωπίνων Υπηρεσιών, της Εσωτερικής Ασφάλειας, του Οικισμού και Αστικής Ανάπτυξης, των Εσωτερικών, της Δικαιοσύνης, της Εργασίας, του Δημοσίου, του Υπουργείου Οικονομικών, των Μεταφορών, των Υποθέσεων Απόστρατων, της Υπηρεσίας Προστασίας του Περιβάλλοντος, της Γενικής Διοίκησης Υπηρεσιών, της Εθνικής Υπηρεσία Αεροναυτικής και Διαστήματος, του Εθνικού Ιδρύματος Επιστημών, της Πυρηνικής Ρυθμιστικής Επιτροπής, του Γραφείου διαχείρισης Προσωπικού, της Διοίκησης Μικρών Επιχειρήσεων, της Διοίκησης Κοινωνικής Ασφάλισης και της Υπηρεσίας Διεθνούς Ανάπτυξης των Ηνωμένων Πολιτειών.

Οι εφαρμογές ιστού, που περιέχουν ευπάθειες (vulnerabilities), είναι στις περισσότερες περιπτώσεις ο αδύναμος κρίκος στην αλυσίδα ασφάλειας των υπολογιστικών συστημάτων επιτρέποντας με αυτόν τον τρόπο να συμβαίνουν παραβιάσεις από κακόβουλους χρήστες (hackers). Οι ευπάθειες αυτές συνήθως υπάρχουν στην αρχιτεκτονική αλλά και στη διαμόρφωση των συστημάτων, στο σχεδιασμό των εφαρμογών, στη διαμόρφωση εγκατάστασης και στη διαχείριση των εφαρμογών. Οι κίνδυνοι από την ύπαρξη αυτών των ευπαθειών μπορεί να είναι πολύ μεγάλοι. Για αυτό το λόγο οι υπεύθυνοι ανάπτυξης των εφαρμογών αλλά και οι υπεύθυνοι ασφαλείας των οργανισμών-επιχειρήσεων που τις χρησιμοποιούν οφείλουν να είναι ικανοί στο να ανιχνεύουν την ύπαρξη και τη σοβαρότητα των ευπαθειών, καθώς και να προτείνουν τα κατάλληλα μέτρα προστασίας των εφαρμογών τους. Αυτό βέβαια προϋποθέτει ότι έχουν το κατάλληλο γνωστικό υπόβαθρο σε θέματα ασφάλειας, ώστε να μπορούν να ανιχνεύουν τις ευπάθειες, αλλά και τα κατάλληλα εργαλεία για να μπορούν να κάνουν τους απαραίτητους ελέγχους γρήγορα και σωστά, χωρίς να επηρεάζεται η εύρυθμη λειτουργία των οργανισμών-επιχειρήσεων τους. Στόχος, λοιπόν αυτού του κεφαλαίου είναι η εξοικείωση με τις σχετικές έννοιες και το κανονιστικό πλαίσιο σε θέματα ασφάλειας, που έχει ήδη ορισθεί από αναγνωρισμένους οργανισμούς, όπως είναι ο CERT και ο OWASP (Open Web Application Security Project – ένας παγκόσμιος οργανισμός που έχει ως σκοπό την βελτίωση της ασφάλειας του λογισμικού εφαρμογών) κ.α., καθώς και η μελέτη των οδηγιών και προτύπων ασφάλειας για την ανάπτυξη και έλεγχο των εφαρμογών ιστού.

1.2 Ορισμοί

1.2.1 Επίθεση

Επιθέσεις (attacks) ονομάζονται οι τεχνικές, που χρησιμοποιούν οι επίδοξοι εισβολείς (intruders) για να εκμεταλλευθούν τις ευπάθειες των διαφόρων εφαρμογών. Οι επιθέσεις αυτές συχνά συγχέονται με

τις ευπάθειες των εφαρμογών. Για το λόγο αυτό οφείλουμε να διευκρινίσουμε ότι επίθεση είναι μια πράξη που ο εισβολέας κάνει σε μια εφαρμογή και δεν είναι μια αδυναμία αυτής.

1.2.2 Απειλή

Απειλή (threat) είναι η ένδειξη του ότι επίκειται κάποιος κίνδυνος ή κάποιο κακό για την εφαρμογή ή το σύστημα γενικότερα. Είναι ο πιθανός κίνδυνος μιας επικείμενης επίθεσης που μπορεί να βλάψει την εφαρμογή. Οποιαδήποτε περίπτωση ή γεγονός με δυνατότητα πρόκλησης ζημιάς σε ένα σύστημα υπό μορφή καταστροφής, κοινοποίησης, τροποποίησης των στοιχείων του, ή/και άρνησης της υπηρεσίας.

1.2.3 Ευπάθεια

Ευπάθεια (vulnerability) ονομάζεται μια «τρύπα» ή μια αδυναμία της εφαρμογής, η οποία μπορεί να οφείλεται σε μια ρωγμή στη σχεδίαση ή ένα σφάλμα υλοποίησης. Αυτή επιτρέπει σε έναν επιτιθέμενο να βλάψει τους ιδιοκτήτες και τους νόμιμους χρήστες της εφαρμογής, ή άλλες οντότητες, που βασίζονται στην εφαρμογή. Ο όρος «ευπάθεια» πολύ συχνά χρησιμοποιείται εσφαλμένα. Πρέπει να διακρίνεται από τους όρους threat (απειλή), attack (επίθεση) και countermeasures (αντίμετρα).

1.2.4 Αντίμετρα

Τα αντίμετρα (countermeasures) περιλαμβάνουν τεχνολογίες ή μοντέλα άμυνας, που χρησιμοποιούνται με σκοπό να ανιχνεύσουν ή να αποτρέψουν επιθέσεις. Τα αναγκαία αντίμετρα σε μια εφαρμογή πρέπει να αναγνωρισθούν με τη χρήση της ανάλυσης κινδύνου, έτσι ώστε να διασφαλιστεί ότι η εφαρμογή προστατεύεται από κοινούς τύπους επιθέσεων. Μια αδυναμία ή μια ρωγμή στη σχεδίαση ενός αντιμέτρου ή η έλλειψη του αναγκαίου αντιμέτρου, έχει ως αποτέλεσμα μια ευπάθεια, η οποία μπορεί να είναι σε θέση να καταστήσει την εφαρμογή ευάλωτη σε επιθέσεις.

1.2.5 Δοκιμή Διείσδυσης

Δοκιμή διείσδυσης (penetration testing) ονομάζεται η μέθοδος εκτίμησης της ασφάλειας ενός υπολογιστικού συστήματος ή δικτύου, με τη μέθοδο της εξομοίωσης μιας επίθεσης. Η δοκιμή διείσδυσης σε μια εφαρμογή ιστού επικεντρώνεται μόνο στην εκτίμηση της ασφάλειας της εφαρμογής ιστού. Η διαδικασία περιλαμβάνει μια ενεργή ανάλυση της εφαρμογής για κάθε είδους αδυναμίες, τεχνικές ρωγμές ή ευπάθειες. Κάθε πρόβλημα ασφάλειας, που ανακαλύπτεται, πρέπει να παρουσιάζεται στον ιδιοκτήτη της εφαρμογής μαζί με τον καθορισμό του αντίκτυπού του και συνήθως με μια πρόταση για τον περιορισμό του ή μια τεχνική λύση αντιμετώπισής του

.

1.2.6 Τι είναι ο έλεγχος (testing);

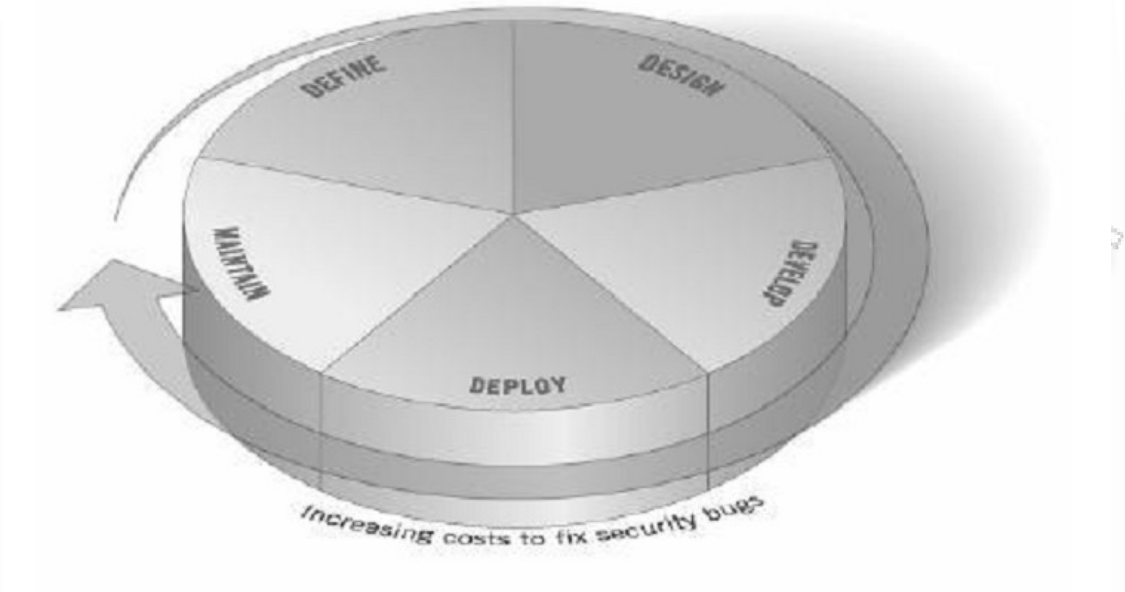
Τι εννοούμε όταν μιλάμε για testing μιας εφαρμογής; Κατά τη διάρκεια ανάπτυξης του κύκλου ζωής μίας web εφαρμογής πολλά πράγματα χρειάζεται να εξεταστούν. Το λεξικό του Meriam-Webster περιγράφει το testing σαν:

- Το να τίθεται κάτι σε δοκιμή ή απόδειξη
- Να υποβάλλεται σε δοκιμή
- Να διατίθεται διαρκώς ή για αξιολόγηση βασισμένη σε δοκιμές

Για τους σκοπούς αυτού του κειμένου, ο έλεγχος(testing) είναι μία διαδικασία σύγκρισης της κατάστασης ενός συστήματος/εφαρμογής απέναντι σε ένα σύνολο κριτηρίων. Στη βιομηχανία της ασφάλειας των δεδομένων, οι άνθρωποι συχνά τεστάρουν σε σχέση με ένα σύνολο διανοητικών κριτηρίων που δεν είναι ούτε καλά ορισμένα ούτε και ολοκληρωμένα. Γι' αυτό το λόγο και άλλους, πολλοί που είναι έξω από το χώρο της ασφάλειας θεωρούν τον έλεγχο της ασφάλειας σαν μαύρη μαγεία. Ο σκοπός αυτού του κειμένου είναι να αλλάξει αυτή η αντίληψη και να γίνει ευκολότερο για ανθρώπους χωρίς γνώση σε βάθος της ασφάλειας να κάνουν τη διαφορά.

1.2.6.1 Πότε κάνουμε έναν έλεγχο?

Οι περισσότεροι άνθρωποι στις μέρες μας δεν ελέγχουν το λογισμικό μέχρι να δημιουργηθεί και να βρίσκεται στην φάση της ανάπτυξης του κύκλου ζωής του (πχ ο κώδικας έχει δημιουργηθεί και αποτελεί μία διαδικτυακή εφαρμογή που βρίσκεται σε λειτουργία). Αυτή γενικά είναι μία πολύ αναποτελεσματική και με απαγορευτικό κόστος πρακτική. Μία από τις καλύτερες μεθόδους ώστε να αποτρέψουμε να εμφανιστούν σφάλματα κατά την παραγωγή εφαρμογών είναι να βελτιώσουμε τον Κύκλο Ζωής Ανάπτυξης του Λογισμικού (Software Development Life Cycle) συμπεριλαμβάνοντας και το κομμάτι της ασφάλειας σε κάθε μία από τις φάσεις του. Ο Κύκλος Ζωής Ανάπτυξης του Λογισμικού (SDLC) είναι μία δομή επιβάλλεται κατά την ανάπτυξη των δημιουργημάτων του λογισμικού. Αν ένας Κύκλος Ζωής Ανάπτυξης του Λογισμικού δεν χρησιμοποιείται προς το παρόν ήρθε η ώρα να διαλέξετε έναν. Η εικόνα που ακολουθεί δείχνει ένα γενικό μοντέλο ενός Κύκλος Ζωής Ανάπτυξης του Λογισμικού καθώς και την (κατ' εκτίμηση) αύξηση του κόστους της διόρθωσης των σφαλμάτων ασφάλειας σε ένα τέτοιο μοντέλο. Οι εταιρείες θα πρέπει να ελέγχουν τον συνολικό Κύκλο Ζωής Ανάπτυξης του Λογισμικού για να εξασφαλίσουν ότι η ασφάλεια αποτελεί αναπόσπαστο κομμάτι της διαδικασίας ανάπτυξης. Οι Κύκλοι Ζωής Ανάπτυξης του Λογισμικού θα πρέπει να περιλαμβάνουν ελέγχους ασφάλειας που να εξασφαλίζουν ότι η ασφάλεια καλύπτεται επαρκώς και πως οι έλεγχοι είναι αποτελεσματικοί καθ' όλη τη διαδικασία ανάπτυξης



Εικόνα : **Ο Κύκλος Ζωής Ανάπτυξης του Λογισμικού.**

1.2.6.2 Τι πρέπει να ελέγξουμε;

Θα ήταν χρήσιμο να σκεφτούμε τη διαδικασία ανάπτυξης λογισμικού σαν ένα συνδυασμό ανθρώπων, διαδικασίας και τεχνολογίας. Αν αυτοί είναι οι παράγοντες που δημιουργούν το λογισμικό, τότε είναι λογικό πως αυτοί είναι οι παράγοντες που πρέπει να ελεγχθούν. Σήμερα οι περισσότεροι άνθρωποι γενικά ελέγχουν την τεχνολογία ή το λογισμικό αυτού καθ' αυτού. Ένας αποτελεσματικός έλεγχος προγράμματος θα πρέπει να περιέχει στοιχεία που ελέγχουν Ανθρώπους - ώστε να διασφαλίσει ότι υπάρχει επαρκής εκπαίδευση και ευαισθητοποίηση – Διαδικασία – για να διαβεβαιώσει πως υπάρχουν οι κατάλληλες πολιτικές και πρότυπα και οι άνθρωποι ξέρουν πώς να ακολουθήσουν αυτές τις πολιτικές – Τεχνολογία – για να διασφαλίσει ότι η διαδικασία έχει υπάρξει αποτελεσματική στην υλοποίησή της. Εξαιρώντας την υιοθέτηση μιας ολιστικής προσέγγισης, ελέγχοντας μόνο την τεχνική υλοποίηση μιας εφαρμογής δεν θα αποκαλυφθούν διοικητικές ή λειτουργικές ευπάθειες που θα μπορούσαν να υπάρχουν. Ελέγχοντας τους ανθρώπους, τις πολιτικές και τις διαδικασίες ένας οργανισμός μπορεί να ανακαλύψει θέματα που θα εμφανιστούν αργότερα σαν ελαττώματα στην τεχνολογία όπως η εξάλειψη των σφαλμάτων από εξ' αρχής και ο προσδιορισμός των βασικών αιτιών των ελαττωμάτων. Ομοίως ελέγχοντας μόνο κάποια από τα τεχνικά θέματα που περιέχονται σε ένα σύστημα θα έχει σαν αποτέλεσμα την ελλιπή και την εσφαλμένη εκτίμηση της κατάστασης της ασφάλειας. Ο Dennis Verdon επικεφαλής της ασφάλειας πληροφοριών του Fidel National Financial παρουσίασε μία εξαιρετική παρομοίωση για αυτή την παρανόηση στο συνέδριο AppSEc 2004 του OWASP στη Νέα Υόρκη. «Εάν τα αυτοκίνητα δημιουργήθηκαν σαν εφαρμογές, οι έλεγχοι ασφάλειας θα αναλάβουν μόνο τη μετωπική σύγκρουση. Τα αυτοκίνητα δε θα ελεγχθούν ως προς την ολίσθησή τους ή για την σταθερότητά τους σε ελιγμούς

έκτακτης ανάγκης, για την αποτελεσματικότητα των φρένων, την πλευρική πρόσκρουση και την αντίσταση στην κλοπή».

1.3 Προβλήματα Ασφάλειας Δικτυακών Εφαρμογών

Ο OWASP (Open Web Application Security Project) που αποτελεί έναν παγκόσμιο οργανισμό που έχει ως σκοπό την βελτίωση της ασφάλειας του λογισμικού εφαρμογών δημοσίευσε για πρώτη φορά τις κατευθυντήριες γραμμές ελέγχου των web εφαρμογών το 2004, οι οποίες στη συνέχεια ενημερώθηκαν το 2007 και το 2010. Οι κατευθυντήριες γραμμές του OWASP χαρακτηρίζονται ως κίνδυνοι A1 έως A10.Τα σημαντικότερα, λοιπόν, προβλήματα που αντιμετωπίζουν οι εφαρμογές ιστού όσον αφορά την ασφάλεια, σύμφωνα με τον οργανισμό OWASP, αναφέρονται συνοπτικά στον παρακάτω πίνακα:

Πίνακας : **Κίνδυνοι ασφάλειας των εφαρμογών ιστού με βάση τον OWASP (2004)**

A1	Μη επικυρωμένα Δεδομένα	Τα δεδομένα που προέρχονται από αιτήματα μέσω δικτύου, δεν επικυρώνονται πριν τη χρήση τους από μια δικτυακή εφαρμογή. Οι επιτιθέμενοι μπορούν να χρησιμοποιήσουν αυτή τη διαρροή ασφαλείας, για να επιτεθούν στα στοιχεία, που την απαρτίζουν, μέσα από μια εφαρμογή
A2	Έλεγχος Πρόσβασης	Οι περιορισμοί σχετικά με τις επιτρεπόμενες ενέργειες των αυθεντικοποιημένων χρηστών δεν επιβάλλονται, όπως πρέπει. Οι επιτιθέμενοι μπορούν να ανακαλύψουν τα κενά αυτά για να αποκτήσουν πρόσβαση σε λογαριασμούς άλλων χρηστών, να δουν ευαίσθητα δεδομένα ή να κάνουν χρήση λειτουργιών, για τις οποίες δεν έχουν δικαιώματα.
A3	Διαχείριση προσβάσεων και συνδέσεων	Οι ιδιότητες των λογαριασμών και συνδέσεις, που έχουν γίνει, δεν προστατεύονται επαρκώς. Επιτιθέμενοι, που μπορούν να χρησιμοποιήσουν κωδικούς, κλειδιά, session cookies, ή άλλα κεκτημένα, μπορούν να ξεπεράσουν τη διαδικασία αυθεντικοποίησης και περιορισμούς, που αυτή επιβάλλει, και να συνδεθούν, προσποιούμενοι άλλους χρήστες.
A4	Διαρροές μέσω Cross Site Scripting (XSS)	Η δικτυακή εφαρμογή μπορεί να χρησιμοποιηθεί σαν ένας μηχανισμός μεταφοράς επιθέσεων στο browser του τελικού χρήστη. Μια επιτυχημένη επίθεση μπορεί να κρατήσει ενεργή τη σύνδεση ενός χρήστη, να μπει στο τοπικό μηχάνημα ή να αλλάξει δεδομένα και περιεχόμενα, για να ξεγελάσει τον χρήστη.
A5	Υπερχείλιση των απομονωτών	Τα στοιχεία μιας δικτυακής εφαρμογής σε ορισμένες γλώσσες, που δεν επικυρώνουν την είσοδο, μπορεί να προκαλέσουν

	(Buffers)	διακοπή της λειτουργίας της εφαρμογής και σε ορισμένες περιπτώσεις να χρησιμοποιηθούν για να αποκτήσει ένας επιτιθέμενος τον έλεγχο της εφαρμογής. Τα στοιχεία αυτά μπορεί να περιλαμβάνουν CGI, βιβλιοθήκες, οδηγούς συσκευών, και στοιχεία δικτυακών εφαρμογών διακομιστών (servers).
A6	Διαρροές μέσω SQL Injection	Οι δικτυακές εφαρμογές περνούν παραμέτρους, όταν παίρνουν πρόσβαση σε εξωτερικά συστήματα όπως ΣΔΒΔ ή λειτουργικά συστήματα. Αν ένας επιτιθέμενος μπορέσει να εγχύσει (inject) κακόβουλες εντολές μέσα σε αυτές τις παραμέτρους, το εξωτερικό σύστημα θα εκτελέσει τις εντολές αυτές για λογαριασμό της εφαρμογής.
A7	Ακατάλληλη διαχείριση λαθών	Οι καταστάσεις λάθους, που προκύπτουν κατά τη διάρκεια κανονικών λειτουργιών, δεν διαχειρίζονται ορθά. Εάν ένας επιτιθέμενος μπορεί να προκαλέσει σφάλματα, για να δώσει την εικόνα ότι η εφαρμογή δεν συμπεριφέρεται σωστά, τότε μπορεί να συλλέξει λεπτομερείς πληροφορίες σχετικά με το σύστημα, να προκαλέσει άρνηση παροχής υπηρεσιών, να θέσει εκτός λειτουργίας μηχανισμούς ασφάλειας ή να προκαλέσει διακοπή της λειτουργίας του server.
A8	Μη ασφαλής Αποθήκευση	Οι δικτυακές εφαρμογές συχνά χρησιμοποιούν εντολές κρυπτογράφησης, για να προστατεύσουν πληροφορίες και πιστοποιητικά. Οι εντολές αυτές και ο κώδικας, που τις ενοποιεί έχει αποδειχθεί ότι είναι δύσκολο να κωδικοποιηθούν σωστά. Το γεγονός αυτό συχνά μας οδηγεί σε αδύναμα μέτρα ασφάλειας.
A9	Άρνηση παροχής Υπηρεσιών	Οι επιτιθέμενοι μπορούν να καταναλώσουν πόρους της δικτυακής εφαρμογής σε σημείο, που άλλοι νόμιμοι χρήστες να μη μπορούν πλέον να έχουν πρόσβαση σε αυτή. Επίσης οι επιτιθέμενοι μπορούν να κλειδώσουν τους λογαριασμούς άλλων χρηστών ή ακόμη να προκαλέσουν την κατάρρευση ολόκληρης της εφαρμογής.
A10	Μη ασφαλής διαχείριση της παραμετροποίησης	Η ύπαρξη ενός ισχυρού προτύπου παραμετροποίησης των διακομιστών αποτελεί σημαντικό στοιχείο για την ασφάλεια των δικτυακών εφαρμογών. Οι διακομιστές αυτοί έχουν πολλές επιλογές παραμετροποίησης, που επηρεάζουν την

		ασφάλεια.

Στον πίνακα που ακολουθεί περιγράφονται αυτές οι υψηλού επιπέδου αλλαγές και ποιες από αυτές καλύπτονται στα αποτελέσματα μεταξύ του 2007 και του 2010:

Πίνακας : **Top 10 των ευπαθειών για το 2007 και το 2010 με βάση τον OWASP**

OWASP Top 10 – 2007		OWASP Top 10 – 2010	
A 2	Injection Flaws	A 1	Injection
A 1	Cross-Site Scripting (XSS)	A 2	Cross-Site Scripting (XSS)
A 7	Broken Authentication and Session Management	A 3	Broken Authentication and Session Management
A 4	Insecure Direct Object Reference	A 4	Insecure Direct Object References
A 5	Cross-Site Request Forgery (CSRF)	A 5	Cross-Site Request Forgery (CSRF)
	Insecure Configuration Management	A 6	Security Mis-configuration
A 8	Insecure Cryptographic Storage	A 7	Insecure Cryptographic Storage
A 10	Failure to Restrict URL Access	A 8	Failure to Restrict URL Access
A 9	Insecure Communications	A 9	Insufficient Transport Layer Protection
	Δεν συμπεριλαμβανόταν στην λίστα Top 10 του 2007	A 10	Invalidated Redirects and Forwards
A 3	Malicious File Execution		Έφυγε από τι λίστα Top 10 του 2010
A 6	Information Leakage and Improper Error Handling		Έφυγε από τι λίστα Top 10 του 2010

Για κάθε έναν από τους Top 10 των κίνδυνους με βάση τον OWASP που προσδιορίστηκαν τόσο για το 2010 όσο και το 2007 δίνονται προτάσεις που καλύπτονται παρακάτω. Κάθε τομέας περιλαμβάνει μια σύντομη συζήτηση για το πώς τα test των διαφόρων προγραμμάτων που αφορούν web εφαρμογές ή οι τεχνικές σάρωσης ευπαθειών μπορούν να χρησιμοποιηθούν για τον εντοπισμό των κινδύνων που όρισε

ο OWASP. Σε επόμενες ενότητες της εργασίας θα αναφερθούμε αναλυτικότερα στο καθένα από αυτά τα προβλήματα σχετικά με την ανίχνευση και την επίλυση τους.

1.4 Βασικές Μέθοδοι Ελέγχου Ασφάλειας των Εφαρμογών Ιστού

Ο συνεχής έλεγχος και η μέτρηση της ασφάλειας των εφαρμογών ιστού είναι απαραίτητη προϋπόθεση για τη διατήρηση της ασφάλειας ενός συστήματος, το οποίο συνδέεται στο Web. Όμως σε έναν ιστότοπο είναι δυνατό να φιλοξενείται ένα μεγάλο πλήθος εφαρμογών ιστού διαφορετικού τύπου με διαφορετικό λογισμικό. Οι εφαρμογές ιστού κατασκευάζονται σε επίπεδα, από προγράμματα και δεδομένα τα οποία φιλοξενούνται σε πολλαπλούς servers(web servers, application servers, database servers). Για το λόγο αυτό υπάρχουν διάφορες μέθοδοι ελέγχου ασφάλειας των εφαρμογών ιστού. Οι σημαντικότερες και ευρέως χρησιμοποιούμενες μέθοδοι είναι οι ακόλουθες:

- Επιθεώρηση Ασφάλειας (security audit):
    - Ένα σύστημα ελέγχεται με βάση ένα σύνολο από λίστες ελέγχου (checklists), οι οποίες διαμορφώνονται με βάση διεθνή πρότυπα σχετικά με την ασφάλεια, καθώς και κατάλληλες πολιτικές ασφάλειας του οργανισμού, που χρησιμοποιεί την εφαρμογή ιστού.
    - Οι ελεγκτές εκτελούν την εργασία τους μέσα από προσωπικές συνεντεύξεις, ανιχνεύσεις αδυναμιών, εξετάσεις των ρυθμίσεων, αναλύσεις των διαμοιρασμένων πόρων δικτύου και μελέτες των ιστορικών στοιχείων (log files).
- Αυτο-αξιολόγηση Ασφάλειας (security self-assessment):
    - Εδώ δεν υπάρχουν συγκεκριμένα standards ως προς τα οποία θα μετρηθεί το σύστημα, αλλά ο στόχος προσδιορίζεται από την περιοχή, που χρειάζεται διερεύνηση και βελτίωση στη θωράκισή της.
    - Ξεπερνά τους πίνακες ελέγχου (checklists) και επεκτείνεται σε ένα πιο λεπτομερή έλεγχο για εντοπισμό αδυναμιών, αλλά και σε συστάσεις για επιδιορθώσεις και βελτιώσεις.
    - Πλεονέκτημά της είναι η δυνατότητα να οριστούν επίπεδα προτεραιότητας σε κάθε συστατικό που αξιολογείται, έτσι ώστε με την ολοκλήρωσή της να δοθεί μια κατάταξη προτεραιοτήτων στην επιδιόρθωση των ευπαθειών που ανιχνεύθηκαν.
- Δοκιμή Διείσδυσης (penetration testing ή "ethical hacking"):
    - Είναι η ελεγχόμενη προσομοίωση μιας επίθεσης προκειμένου να επιτευχθεί ένας προκαθορισμένος στόχος. Επίσης είναι γνωστή και ως εσωτερική επιθεώρηση ασφάλειας (internal security auditing).
    - Σκοπός της είναι να εντοπιστούν συγκεκριμένες πληροφορίες σχετικές με την ύπαρξη γνωστών ευπαθειών και να διερευνηθεί κατά πόσο είναι δυνατόν ένας ξένος, κάνοντας χρήση αυτών των πληροφοριών, να είναι σε θέση να

δημιουργήσει προβλήματα στην εφαρμογή ιστού. Δεν έχει σκοπό να εντοπίσει όλες τις ευπάθειες, αλλά να αποδείξει ότι η ασφάλεια του συστήματος μπορεί να διακυβευτεί.

- Η δοκιμή μπορεί να πραγματοποιηθεί στη βάση μηδενικής γνώσης (zero knowledge) ή με πλήρη γνώση (full knowledge) του συστήματος, που δοκιμάζεται. Χρησιμοποιείται για να καθορίσει την αξιοπιστία και τη δύναμη των μέτρων ασφάλειας, που παίρνουμε.
- Οι "ethical hackers" προσπαθούν να υιοθετήσουν τις τεχνικές επιθέσεων των hackers, ώστε να μπορέσουν να μετρήσουν το επίπεδο ασφάλειας της εφαρμογής.

1.4.1 Σύγκριση των Μεθόδων Μέτρησης Ασφάλειας

Κάνοντας μια σύγκριση των μεθόδων μέτρησης ασφάλειας των εφαρμογών ιστού μπορούμε να πούμε ότι:

- Για τον έλεγχο της ασφάλειας μιας εφαρμογής ιστού με τις μεθόδους της επιθεώρησης ασφάλειας και της αυτο-αξιολόγησης απαιτείται η μετακίνηση μιας μεγάλης ομάδας ειδικών ασφαλείας στον τόπο που λειτουργεί ο οργανισμός, του οποίου η ασφάλεια της εφαρμογής ελέγχεται.
- Η ομάδα αυτή πρέπει να έχει στη διάθεσή της τα κατάλληλα checklists, να έχει υψηλή τεχνογνωσία και να είναι άρτια συντονισμένη.
- Για την ενέργεια των ελέγχων απαιτείται πολύς χρόνος, ώστε να ολοκληρωθούν οι συνεντεύξεις, οι επιθεωρήσεις, οι αξιολογήσεις και οι έρευνες στη διάρκεια των οποίων αποκαλύπτεται και διαταράσσεται η λειτουργία του οργανισμού.

Όλα τα παραπάνω σε συνάρτηση με την ανάγκη για συνεχείς και επαναλαμβανόμενους ελέγχους καθιστούν τη δοκιμή διείσδυσης μονόδρομο. Επιπλέον, η δοκιμή διείσδυσης έχει τα ακόλουθα πλεονεκτήματα:

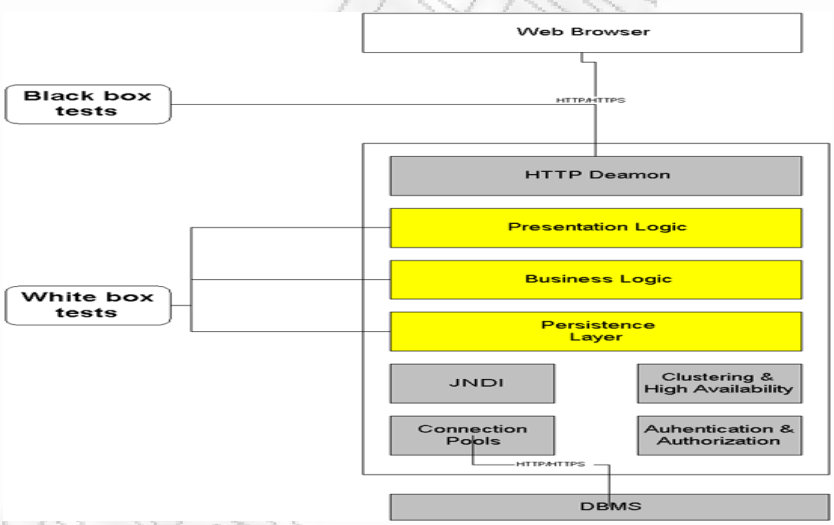
- απαιτείται ελάχιστο προσωπικό και δεν είναι αναγκαία η μετακίνησή του
- παρέχει τη δυνατότητα πλήρους αυτοματοποίησης
- διαρκεί ελάχιστο χρόνο και είναι εύκολα επαναλαμβανόμενη
- δεν απαιτεί τη σε βάθος γνώση της ελεγχόμενης εφαρμογής
- δεν διαταράσσει τη λειτουργία της
- είναι πολύ οικονομικότερη από τις δύο άλλες μεθόδους.

1.4.2 Τεχνικές Δοκιμής Διείσδυσης

Υπάρχουν δύο κύριες προσεγγίσεις τεχνικών ελέγχου ασφάλειας των εφαρμογών ιστού με τη μέθοδο της δοκιμής διείσδυσης:

- **Χειροκίνητη (manual)**, στην οποία όλη η διαδικασία ελέγχου γίνεται βήμα- βήμα χωρίς την ύπαρξη αυτοματισμών επανάληψης παρόμοιων βημάτων.
- **Αυτοματοποιημένη (automated)**, στην οποία με τη χρήση εργαλείων αυτοματοποιούνται μερικοί ή όλοι οι έλεγχοι και οι διαδικασίες ελέγχου. Η αυτοματοποιημένη διακρίνεται σε δύο τεχνικές:
    - **Black Box**: ονομάζεται η εφαρμογή δοκιμαστικών δεδομένων που έχουν προέλθει από καθορισμένες λειτουργικές απαιτήσεις χωρίς να λαμβάνουν υπόψη τη δομή της εφαρμογής, στην οποία εφαρμόζονται. Η εφαρμογή εξετάζεται χρησιμοποιώντας την εξωτερική της διεπαφή, αυτή, που χρησιμοποιούν οι απλοί χρήστες. Μιμούνται την ακολουθία αλληλεπιδράσεων χρήστη-εφαρμογής και κάθε αποτυχία δείχνει ότι ο χρήστης έλαβε ανεπαρκή υπηρεσία.
    - **White Box**: ονομάζεται η τεχνική στην οποία εξετάζεται η δομή της εφαρμογής και στη βάση αυτής καθορίζονται τα δεδομένα της δοκιμής. Εξετάζεται η εσωτερική δομή της εφαρμογής χρησιμοποιώντας τη διεπαφή προγραμματισμού εφαρμογών (Application Programming Interface), η οποία αποτελεί το μέσο επικοινωνίας των εφαρμογών με τον πυρήνα του λειτουργικού συστήματος ή με βιβλιοθήκες τρίτων κατασκευαστών.

Στην εικόνα 3 παρουσιάζεται η διαφορά στη διεπαφή, που χρησιμοποιεί ο ελεγκτής ασφάλειας για την πραγματοποίηση των ελέγχων του με τις δύο αυτές τεχνικές:



Εικόνα : **Σχηματική αναπαράσταση της διεπαφής (interface) των Black box και White box τεχνικών δοκιμής διείσδυσης σε μια εφαρμογή ιστού.**

Κεφάλαιο 2ο: [Παρουσίαση του WebScarab]

Εισαγωγή

Εικόνα ***:*** **Το εικονίδιο του WebScarab**

Το WebScarab είναι ένα εξελισσόμενο project υπό την αιγίδα του Open Web Application Security Project (OWASP). Ο στόχος του WebScarab είναι να παρασχεθεί ένα ελεύθερο εργαλείο στους υπεύθυνους ανάπτυξης και τους κριτικούς εφαρμογών του Web, ώστε να κατανοήσουν τη λειτουργία των εφαρμογών ιστού και να προσδιορίσουν τα πιθανά προβλήματα, τα οποία θα μπορούσαν να προκαλέσουν τη δυσλειτουργία αυτών των εφαρμογών.

Το WebScarab λειτουργεί υπό την άδεια του GNU General Public License v2. Το WebScarab είναι ένα πλαίσιο (framework) για εφαρμογές ιστού, που επικοινωνούν με τη χρησιμοποίηση των πρωτοκόλλων HTTP και HTTPS. Είναι γραμμένο σε γλώσσα Java και για αυτό το λόγο μπορεί να λειτουργήσει σε πολλές πλατφόρμες. Το WebScarab λειτουργεί με τη χρήση διαφόρων plug-ins δηλαδή διάφορα προγράμματα, που προσφέρουν συγκεκριμένες λειτουργίες στο εργαλείο. Στην πιο κοινή χρήση του το WebScarab λειτουργεί ως ένας "intercepting proxy", επιτρέποντας στο χρήστη να ελέγξει και να τροποποιήσει τα αιτήματα, που δημιουργούνται από τον browser του χρήστη, προτού σταλούν στον server, καθώς και να ελέγξει και να τροποποιήσει τις απαντήσεις του server, προτού αυτές παραληφθούν από τον browser. Το WebScarab είναι σε θέση να παρεμποδίσει την επικοινωνία HTTP και HTTPS. Ο χρήστης μπορεί επίσης να εξετάσει τις συνομιλίες, τα αιτήματα και τις απαντήσεις, που έχουν περάσει μέσω του WebScarab. Είναι ένα εργαλείο με σκοπό, πρώτιστα, να χρησιμοποιηθεί από τους ανθρώπους, που έχουν γνώσεις προγραμματισμού. Το WebScarab έχει ως σκοπό να αποκαλύψει τον τρόπο λειτουργίας μιας εφαρμογής HTTP(S) ή να επιτρέπει στον υπεύθυνο ανάπτυξης(developer) να διορθώνει δύσκολα προβλήματα ή να επιτρέπει σε έναν ειδικό ασφάλειας (ελεγκτή) να προσδιορίζει τις αδυναμίες μιας εφαρμογής ανάλογα με τον τρόπο που αυτή έχει σχεδιαστεί ή εφαρμοσθεί.

Το WebScarab παρέχει διάφορα plug-ins, προς το παρόν, που στοχεύουν κυρίως, στη λειτουργία της επιθεώρησης ασφάλειας των εφαρμογών ιστού. Αυτά τα plug-ins, στα οποία να αναφερθούμε αναλυτικότερα στη συνέχεια, περιλαμβάνουν λειτουργίες όπως:

- **Fragments (αποσπάσματα) -** εξάγει Scripts και HTML σχόλια από HTML σελίδες όπως αυτά φαίνονται μέσω του proxy ή άλλων plug-ins.
- **Proxy(πληρεξούσιος)** – παρατηρεί την κυκλοφορία μεταξύ browser και web server. Ο WebScarab proxy είναι σε θέση να παρατηρήσει και το HTTP και την κρυπτογραφημένη κυκλοφορία HTTPS. Για το HTTPS δημιουργείται μια σύνδεση SSL (SSL tunnel) μεταξύ του WebScarab και του browser, αντί απλά να συνδέει τον browser στο server, επιτρέποντας έτσι σε κρυπτογραφημένα δεδομένα να περάσουν μέσω αυτού. Διάφορα proxy plug-ins έχουν αναπτυχθεί επίσης, για να επιτρέπουν στο χρήστη να ελέγχει τα αιτήματα και τις απαντήσεις που περνούν μέσω του proxy.
- **Manual intercept (χειροκίνητη υποκλοπή)** – επιτρέπει στο χρήστη να τροποποιήσει άμεσα τα αιτήματα HTTP/HTTPS και τις απαντήσεις, πριν να φθάσουν στο server ή στον browser αντίστοιχα.
- **Beanshell** – επιτρέπει την εκτέλεση αυθαίρετα σύνθετων διαδικασιών με αιτήματα και απαντήσεις. Οτιδήποτε μπορεί να εκφραστεί στην Java μπορεί να εκτελεσθεί μέσω του plug-in αυτού.
- **Reveal hidden fields(αποκάλυψη κρυμμένων πεδίων)** – μερικές φορές είναι ευκολότερο να τροποποιηθεί ένα κρυμμένο πεδίο στην ίδια τη σελίδα, παρά να υποκλαπεί το αίτημα, αφότου έχει σταλεί. Αυτό το plug-in αλλάζει απλά όλα τα κρυμμένα πεδία, που βρίσκονται στις σελίδες HTML σε πεδία απλού κειμένου (text) καθιστώντας τα έτσι ορατά και επεξεργάσιμα.
- **Προσομοιωτής εύρους ζώνης** – επιτρέπει στο χρήστη να μιμηθεί ένα πιο αργό δίκτυο, προκειμένου να παρατηρηθεί πώς ένας ιστότοπος θα λειτουργεί, όταν για παράδειγμα η πρόσβαση γίνεται μέσω ενός modem.
- **Spider (αράχνη)** – προσδιορίζει νέα URLs στην περιοχή του ιστότοπου- στόχου και τις προσκολλά στην περιοχή εντολών του συγκεκριμένου plug-in.
- **Manual Request (χειροκίνητο αίτημα)** – επιτρέπει επεξεργασία και επανάληψη προηγούμενων αιτημάτων ή δημιουργία εξ' ολοκλήρου νέων αιτημάτων.
- **SessionID analysis (συλλογή αναγνωριστικών συνόδου)** – συλλέγει και αναλύει τα διάφορα cookies (και URL-based παραμέτρους) για να προσδιορισθεί, και οπτικά, ο βαθμός τυχαιότητας και μη προβλεψιμότητας των cookies που δημιουργούνται από την εφαρμογή.
- **Scripted** – μπορεί να χρησιμοποιηθεί η γλώσσα BeanShell για να γραφτούν scripts και να δημιουργούνται με αυτά τα αιτήματα προς το server. Το script μπορεί να εκτελέσει κάποια ανάλυση στις απαντήσεις, που θα λαμβάνονται από το server, χρησιμοποιώντας το μοντέλο αιτημάτων και απαντήσεων του WebScarab, που απλοποιεί τις ενέργειες των χρηστών.

- **Parameter fuzzer** – εκτελεί την αυτοματοποιημένη αντικατάσταση των τιμών παραμέτρων, πράγμα το οποίο είναι πιθανό να αποκαλύψει την ελλιπή επικύρωση παραμέτρων οδηγώντας στην ανακάλυψη ευπαθειών, όπως είναι οι Cross Site Scripting (XSS) και SQL Injection.
- **Search (αναζήτηση)** – επιτρέπει στο χρήστη να επεξεργαστεί αυθαίρετες BeanShell εκφράσεις, για να προσδιορίσει τις συνομιλίες, που επιθυμεί να επεξεργασθεί.
- **Compare (σύγκριση)** – υπολογίζει την "απόσταση επεξεργασίας" μεταξύ του σώματος της απάντησης, και όχι των ετικετών των συνομιλιών που ήδη παρατηρήθηκαν, και μιας επιλεγμένης συνομιλίας που ορίζεται ως βάση. Η απόσταση επεξεργασίας είναι "ο απαιτούμενος αριθμός επεξεργασιών, για να μετασχηματισθεί ένα έγγραφο σε κάποιο άλλο". Για λόγους απόδοσης ο αριθμός επεξεργασιών υπολογίζεται χρησιμοποιώντας τα σημεία της λέξης (word tokens), δηλαδή τα γράμματα και όχι τα bytes.
- **SOAP** – αναλύει το έγγραφο WSDL και παρουσιάζει τις διάφορες λειτουργίες και τις απαραίτητες παραμέτρους επιτρέποντας την επεξεργασία τους, πριν σταλούν στο server.

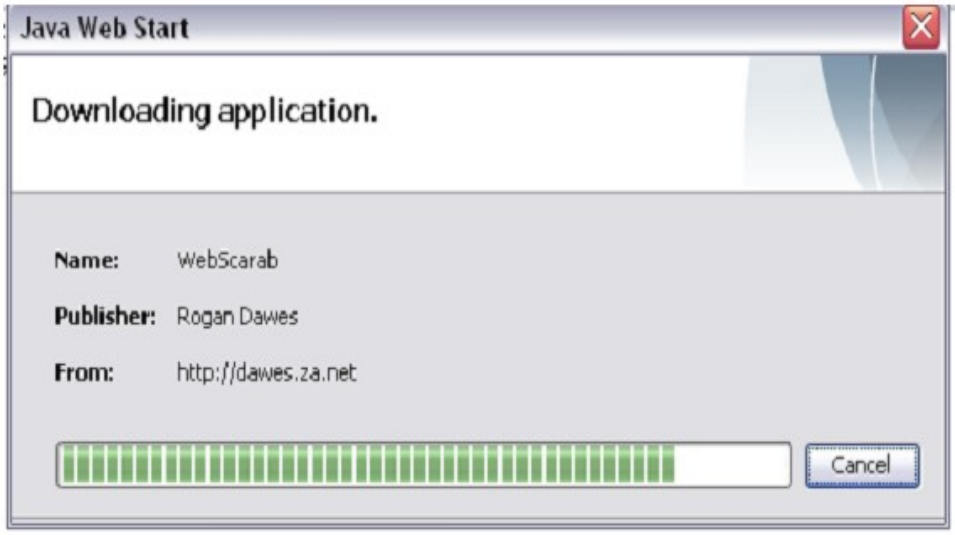
Σαν πλαίσιο (Framework) το WebScarab είναι επεκτάσιμο. Κάθε ένα από τα ανωτέρω συστατικά εφαρμόζεται ως plug-in και μπορεί να αφαιρεθεί ή να αντικατασταθεί. Τα νέα συστατικά του WebScarab, σύμφωνα με τον OWASP θα περιλάβουν πιθανώς:

- Ενίσχυση του SOAP plug-in, που θα περιλαμβάνει τη βελτίωση της υποστήριξης για σύνθετα σχήματα και διαφορετικές κωδικοποιήσεις.
- Συνδυασμό των Search και Compare plug-ins έτσι, ώστε να μπορούμε να συγκρίνουμε συγκεκριμένες απαντήσεις του server.

Εγκατάσταση

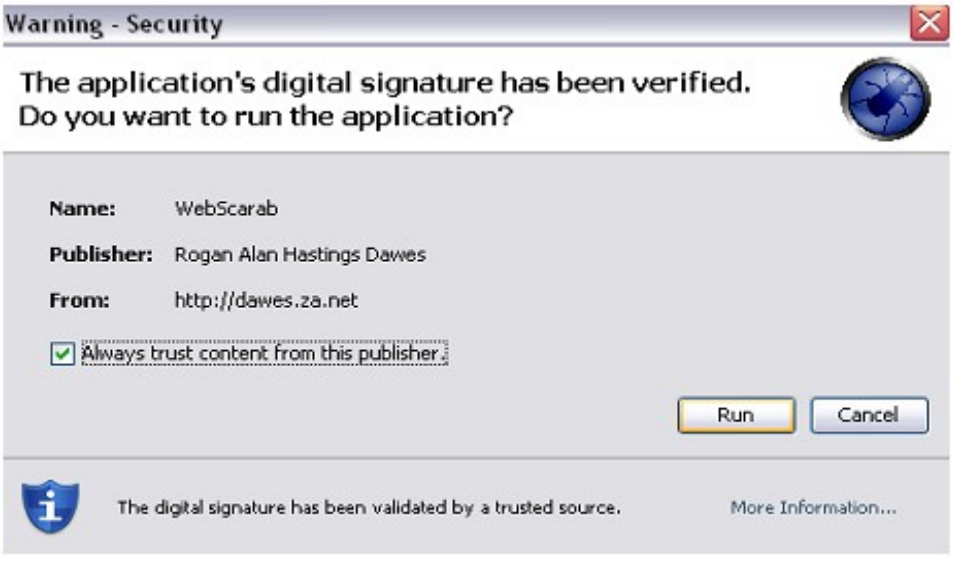
Μπορούμε να "κατεβάσουμε" το WebScarab από το OWASP Source Code Center του Sourceforge. Η ηλεκτρονική διεύθυνση είναι: http://sourceforge.net/project/showfiles.php?group_id=64424&package_id=61823. Επίσης, μια έκδοση για λειτουργικό σύστημα Macintosh μπορεί να βρεθεί στην ιστοσελίδα Corsaire (http://research.corsaire.com/tools/). Επιπλέον μπορούμε να δοκιμάσουμε την έκδοση Java Web Start (http://dawes.za.net/rogan/webscarab/WebScarab.jnlp), η οποία έχει το πλεονέκτημα ότι θα λαμβάνει αυτόματα οποιεσδήποτε νέες εκδόσεις του εργαλείου κυκλοφορούν. Εμείς επιλέξαμε την τρίτη επιλογή, δηλαδή την έκδοση της Java Web Start. Είναι ανάγκη να τονίσουμε ότι το WebScarab είναι μια εφαρμογή σε Java, δηλαδή απαιτεί το περιβάλλον Java Runtime, ώστε να μπορεί να λειτουργεί. Μπορούμε να "κατεβάσουμε" το JRE (Java Runtime Environment) από την ιστοσελίδα της Sun (http://java.sun.com/).

Αφού εγκαταστήσουμε πρώτα το JRE, στη συνέχεια προχωρούμε στην εγκατάσταση του WebScarab. Επιλέγοντας το αρχείο "WebScarab.jnlp", "κατεβάζουμε" την εφαρμογή από την τοποθεσία http://dawes.za.net.



Εικόνα ***:*** **Εγκατάσταση της εφαρμογής WebScarab.**

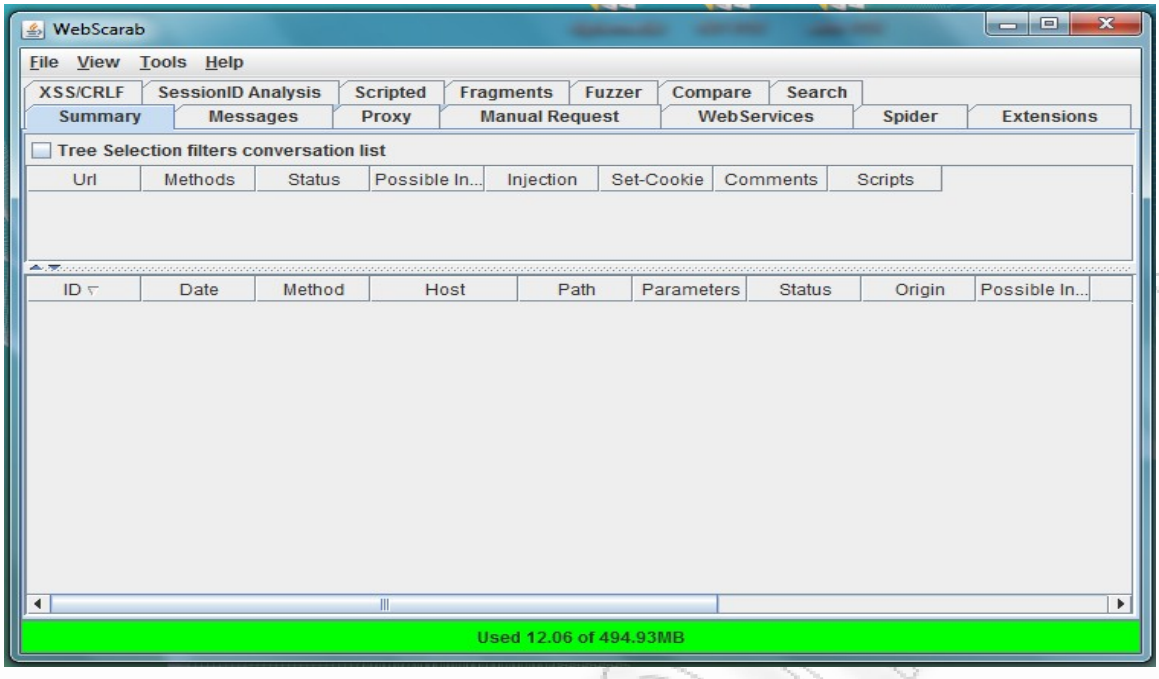
Στη συνέχεια, αφού γίνει αποδεκτή ως έγκυρη η ψηφιακή υπογραφή του προϊόντος (εικόνα 6), μπορούμε να προχωρήσουμε στην εγκατάσταση του. Τέλος, αφού ολοκληρώσουμε με την εγκατάσταση, θα πρέπει στη συνέχεια να προχωρήσουμε σε κάποιες ρυθμίσεις του περιβάλλοντος του WebScarab, όπως θα δούμε παρακάτω.



Εικόνα ***:*** **Μήνυμα επιτυχής εγκατάσταση της εφαρμογής WebScarab.**

Ρυθμίσεις Περιβάλλοντος

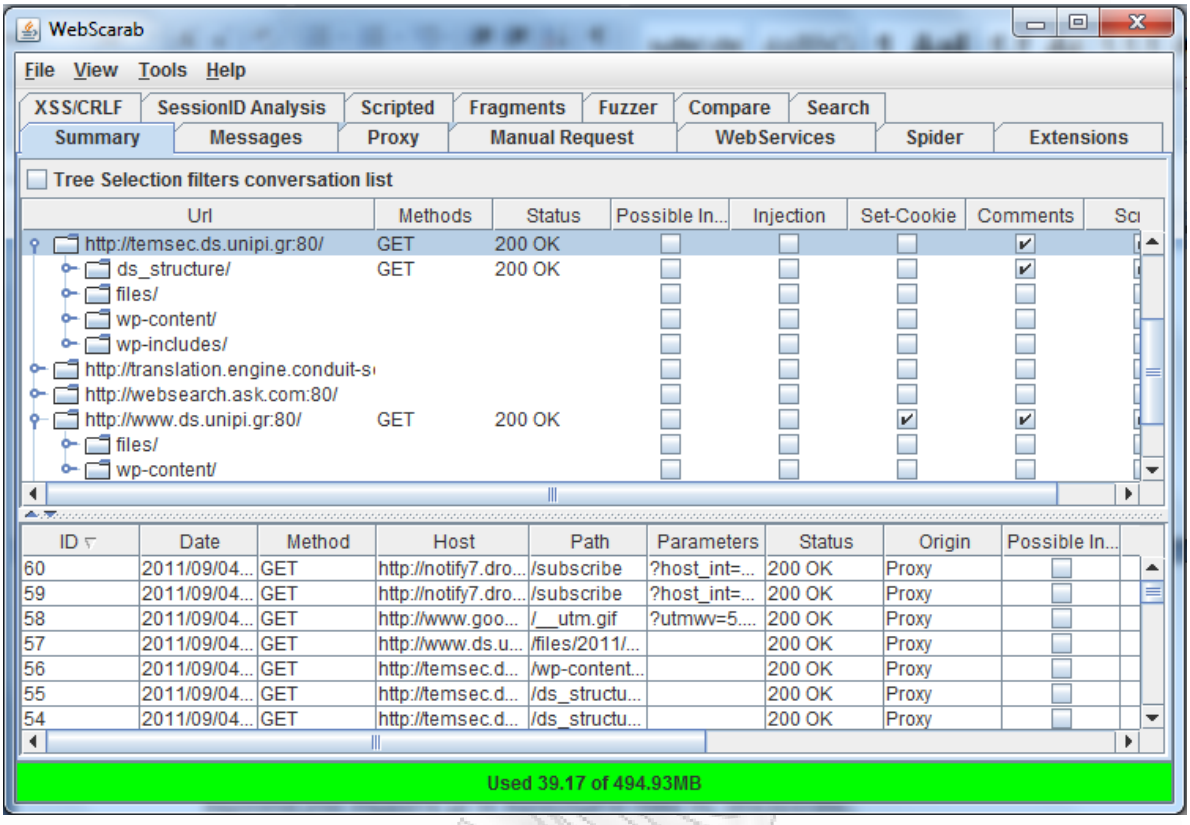
Στην εικόνα 7, που ακολουθεί, βλέπουμε το περιβάλλον του WebScarab:



Εικόνα ***:*** **Το περιβάλλον της εφαρμογής WebScarab.**

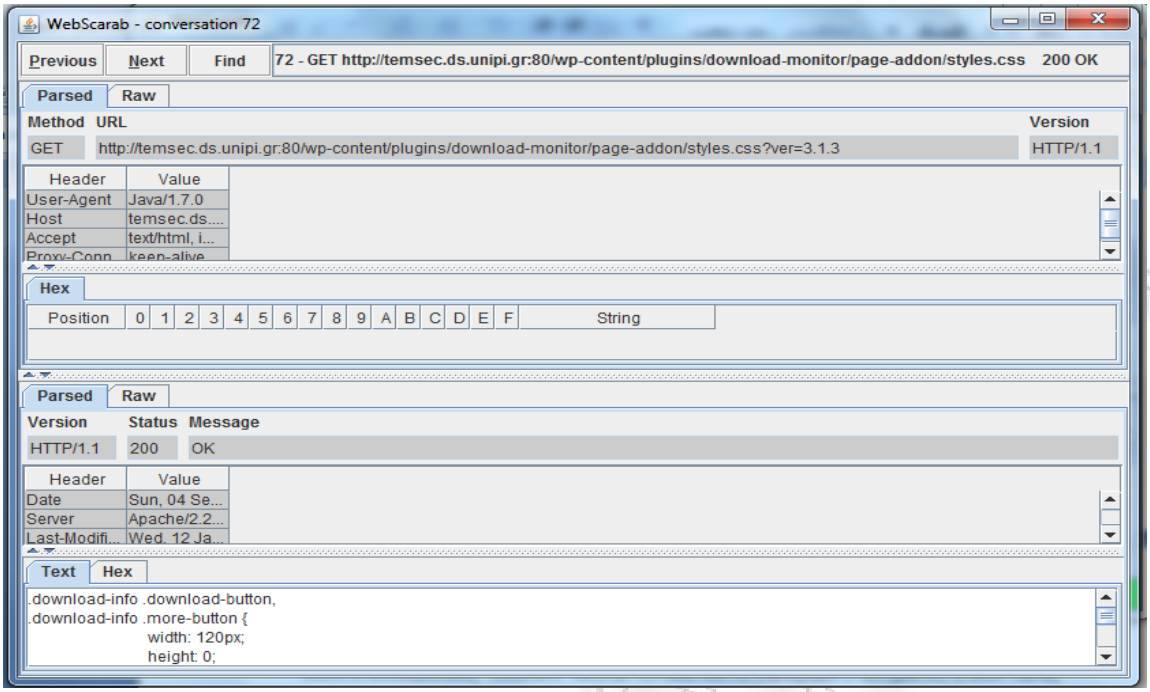
Υπάρχουν μερικές σημαντικές περιοχές που ίσως να χρειαστούν επεξήγηση. Αρχικά η μπάρα εργαλείων παρέχει την πρόσβαση στα διάφορα plug-ins όπως στο Summary παράθυρο (κύρια όψη) και στο παράθυρο Messages (μηνυμάτων - logs).

Το παράθυρο Summary είναι χωρισμένο σε δύο μέρη. Το πάνω μέρος είναι ένας πίνακας σε δεντρική μορφή, και παρουσιάζει το σχεδιάγραμμα των περιοχών που έχουμε επισκεφτεί, και μερικές ιδιότητες των διάφορων URLs, πχ Method, Status κλπ.. Κάτω από αυτό είναι ένας πίνακας και παρουσιάζει όλες τις συνομιλίες, που έχουν "περάσει" μέσα από το WebScarab, ταξινομημένες κατά φθίνουσα σειρά σύμφωνα με την ταυτότητα τους, δηλαδή τον αριθμό που δείχνει τη σειρά με την οποία πραγματοποιήθηκαν, έτσι, ώστε οι πιο πρόσφατες συνομιλίες να είναι στην κορυφή του πίνακα. Η σειρά ταξινόμησης, εάν επιθυμούμε, μπορεί να αλλάξει κάνοντας διπλό κλικ στις επιγραφές των στηλών.



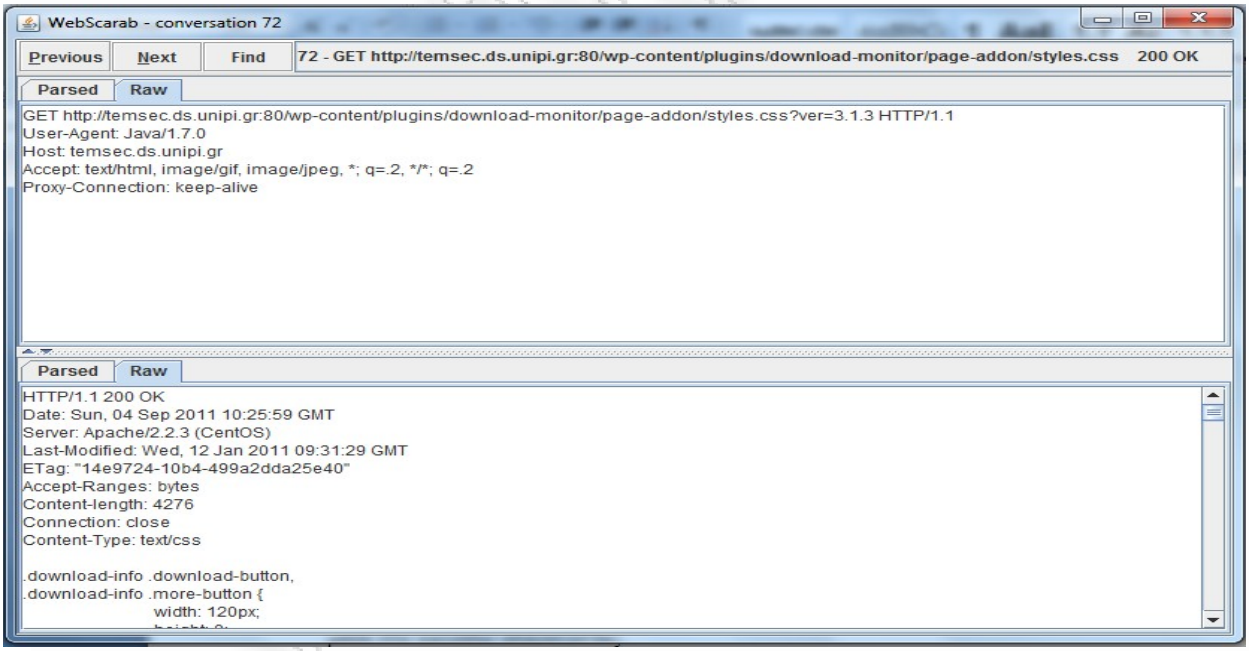
Εικόνα *:* **WebScarab: το παράθυρο Summary.**

Στην εικόνα 8 μπορούμε να δούμε το δέντρο των URLs, το οποίο αντιπροσωπεύει το σχεδιάγραμμα των ιστοτόπων, που έχουμε επισκεφθεί όπως και τις μεμονωμένες συνομιλίες, που έχουν περάσει μέσω του WebScarab. Για να δούμε τις λεπτομέρειες μιας ιδιαίτερης συνομιλίας, μπορούμε να κάνουμε double-click σε μια γραμμή στον πίνακα και τότε ανοίγει ένα παράθυρο, που παρουσιάζει το αίτημα και τις λεπτομέρειες της ληφθείσης απάντησης. Μπορούμε να δούμε το αίτημα και την απάντηση με ποικίλες μορφές. Για τη συνομιλία 72 για παράδειγμα η άποψη, που παρουσιάζεται στην εικόνα 9 είναι η "Parsed", όπου οι επικεφαλίδες "σπάσανε" σε έναν πίνακα και το περιεχόμενο αιτήματος ή απάντησης παρουσιάζεται σύμφωνα με το περιεχόμενο – τύπο της επικεφαλίδας.



Εικόνα ***:* Το αίτημα της συνομιλίας 72 σε μορφή Parsed.**

Μπορούμε επίσης να επιλέξουμε τη μορφή "Raw", όπου το αίτημα ή η απάντηση παρουσιάζεται ακριβώς, όπως θα φαινόταν, αν μπορούσαμε να το δούμε, μέσα στο καλώδιο επικοινωνίας:



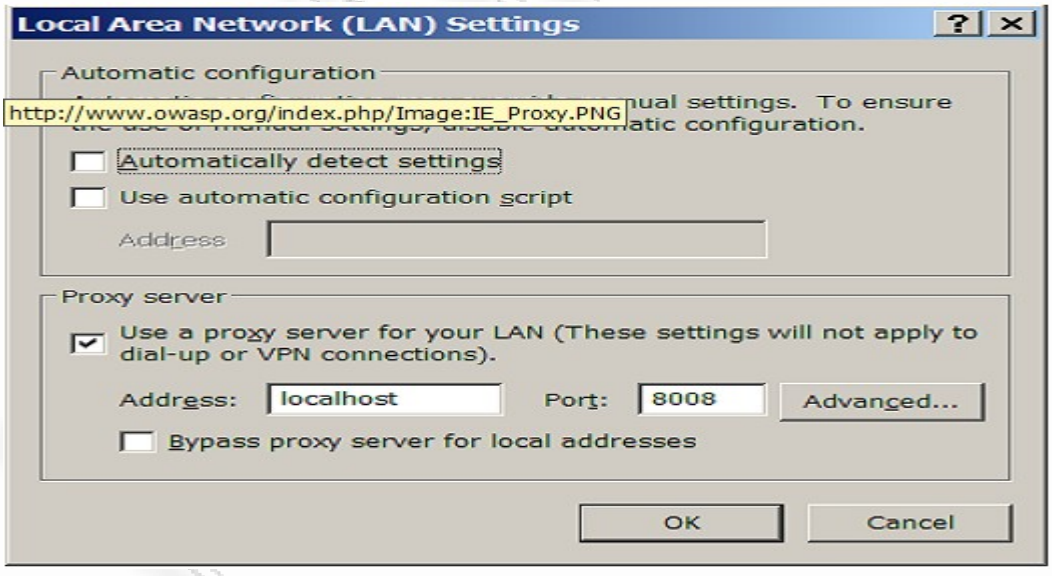
Εικόνα ***:* Το αίτημα της συνομιλίας 72 σε μορφή Raw.**

- **Upstream Proxy**

Εάν πρέπει να χρησιμοποιήσουμε ένα upstream proxy, μπορούμε να κάνουμε τις εξής ρυθμίσεις μέσω της επιλογής Tools -> Proxies menu. Για χρήστες Windows το WebScarab έχει υποστήριξη για αίτηση των ρυθμίσεων Internet Explorer's proxy φορτώνοντας αυτές αυτόματα στα κατάλληλα πλαίσια διαλόγου. Αυτό το χαρακτηριστικό γνώρισμα απαιτεί ένα JNI plug-in/DLL, το οποίο πρέπει να είναι στο PATH. Εάν το αρχείο DLL βρέθηκε επιτυχώς, θα δούμε ένα κουμπί με όνομα "Get IE Settings", όπως βλέπουμε και στην εικόνα 11. Εάν το αρχείο DLL δεν βρίσκεται, αυτό το κουμπί δεν εμφανίζεται. Χρησιμοποιώντας, όμως, το WebScarab installer για την εγκατάσταση αυτό διαμορφώνεται ανάλογα.

Εικόνα ***:* Διαμόρφωση του proxy.**

Εάν δεν χρησιμοποιείται proxy server, χρειάζεται να προχωρήσουμε στις εξής ρυθμίσεις του Internet Explorer: Tools -> Internet Options -> Connections -> LAN Settings για να πάρουμε το παράθυρο proxy configuration, όπως φαίνεται στην εικόνα 12.



Εικόνα ***:* Ρυθμίσεις proxy server του Internet Explorer.**

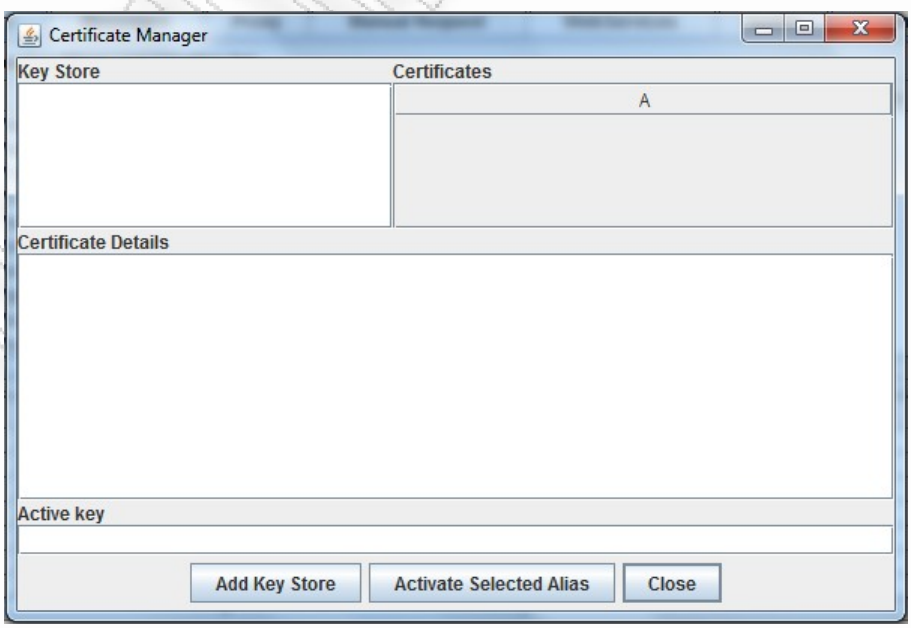
Το WebScarab ως προεπιλογή χρησιμοποιεί την πόρτα 8008 στο localhost για τον proxy. Πρέπει να διαμορφώσουμε τον IE για να αναμεταδίδει τα αιτήματα στο WebScarab, παρά να τα εκτελεί μόνος

του. Πρέπει να σιγουρευτούμε ότι όλα τα checkboxes δεν είναι μαρκαρισμένα εκτός από το "Use a proxy server". Μόλις διαμορφώσουμε τον IE να χρησιμοποιεί τον proxy, επιλέγουμε OK σε όλα τα παράθυρα, για να γυρίσουμε πίσω στον browser. Μπαίνουμε σε ένα non-SSL ιστότοπο και έπειτα ξεκινάμε το WebScarab.

Θα δούμε τότε τις συνομιλίες να περνάνε μέσα από τον proxy του WebScarab. Εάν δεν συμβαίνει αυτό ή παίρνουμε ένα λάθος, ενώ είμαστε στη σελίδα, τότε πρέπει να επιστρέψουμε και να ελέγξουμε τις ρυθμίσεις του proxy στον Internet Explorer, όπως περιγράφεται ανωτέρω. Εάν οι ρυθμίσεις του proxy είναι σωστές, μια πιθανότητα είναι να υπάρχει ήδη ένα άλλο πρόγραμμα, που χρησιμοποιεί την πόρτα 8008 αποτρέποντας έτσι το WebScarab από τη χρησιμοποίησή της. Σε αυτή την περίπτωση, πρέπει να σταματήσουμε το άλλο πρόγραμμα. Θα δούμε επίσης πώς να ρυθμίσουμε το WebScarab, ώστε να χρησιμοποιήσει μια διαφορετική πόρτα λίγο αργότερα.

- **Client-side Certificates**

Το WebScarab έχει υποστήριξη για πιστοποιητικά SSL πελατών. Μπορεί να έχει πρόσβαση στα κλειδιά και τα πιστοποιητικά, που αποθηκεύονται με μορφή αρχείων PKCS#12, καθώς επίσης χρησιμοποιώντας και PKCS#11 στην πλατφόρμα Java 1.5. Αυτή η λειτουργία προσεγγίζεται μέσω του Tools->Certificates menu. Απλά επιλέγουμε την επιλογή "Add Keystore" και επιλέγουμε τον τύπο keystore, που θέλουμε να προστεθεί. Μόλις τα keystores έχουν διαμορφωθεί κατάλληλα, μπορούμε να δούμε τις λεπτομέρειες των πιστοποιητικών. Για να ενεργοποιήσουμε ένα συγκεκριμένο πιστοποιητικό, επιλέγουμε "Activate selected Alias". Μπορεί να μας ζητηθεί να χρησιμοποιήσουμε ένα κωδικό πρόσβασης, για να έχουμε πρόσβαση στο κλειδί. Για να σταματήσουμε τη χρήση κάποιου πιστοποιητικού πελατών, χρειάζεται να σιγουρευτούμε ότι κανένα ψευδώνυμο (alias) δεν είναι επιλεγμένο, και επιλέγουμε " Activate selected Alias ". Μπορούμε να δούμε, ποιο alias είναι ενεργό στο πεδίο κειμένου "Active key".



Εικόνα ***:*** **Δυνατότητα εισαγωγής και αποθήκευσης κλειδιών και πιστοποιητικών πελατών.**

- **Server και Proxy αυθεντικοποίηση**

Το WebScarab παρέχει υποστήριξη για αυθεντικοποίηση proxy και web servers χρησιμοποιώντας βασική, NTLM και Negotiate και όχι Kerberos αυθεντικοποίηση. Την πρώτη φορά, που ο server απαιτεί αυθεντικοποίηση, το WebScarab θα μας προτρέψει να δώσουμε τα διαπιστευτήρια μας για το server. Αυτά τα διαπιστευτήρια θα επαναχρησιμοποιηθούν και για τα επόμενα αιτήματα, σε περίπτωση ανάγκης από τον web server ή proxy.

Διαχείριση Συνόδων

Το WebScarab χρησιμοποιεί την έννοια της συνόδου επικοινωνίας (session), για να συσχετίσει τις διάφορες συνομιλίες (conversations) μεταξύ τους. Είναι υποχρεωτικό να υπάρχει μια σύνοδος κατά τη χρησιμοποίηση του WebScarab, γιατί κάθε συνομιλία καταγράφεται στη σύνοδο, όπως είδαμε στο Summary και μπορούμε να την αναζητήσουμε αργότερα αν χρειαστεί. Προς το παρόν οι σύνοδοι μπορούν να σωθούν μόνο σε ένα κατάλογο του filesystem. Κατά τη δημιουργία μιας νέας συνόδου πρέπει να επιλέξουμε έναν κενό κατάλογο ή να δημιουργήσουμε ένα νέο directory, όπου θα αποθηκευτεί.

Όταν το WebScarab ξεκινά, μας προτρέπει να επιλέξουμε ποιο είδος συνόδου θέλουμε να χρησιμοποιήσουμε. Η προεπιλογή είναι να δημιουργηθεί μια προσωρινή σύνοδος. Αυτή η προσωρινή σύνοδος είναι ένας δημιουργούμενος στον $java.io.tmpdir κατάλογος, που χρησιμοποιεί ένα σχέδιο webscarabnnnnn.tmp. Αυτός ο προσωρινός κατάλογος διαγράφεται, όταν ο χρήστης βγαίνει από το WebScarab. Εάν επιθυμούμε να συντηρήσουμε το αρχείο λογιστικού ελέγχου (audit log) για μετέπειτα έλεγχο, πρέπει να αρχίσουμε μια νέα σύνοδο, πριν δημιουργήσουμε ή παρεμποδίσουμε οποιεσδήποτε συνομιλίες. Εναλλακτικά, εάν πρέπει να σώσουμε μια προσωρινή σύνοδο, που περιέχει ήδη τις συνομιλίες, δεν πρέπει να τερματίσουμε το WebScarab. Πρώτα κάνουμε αλλαγή σε μια νέα σύνοδο ή ανοίγουμε μια υπάρχουσα σύνοδο. Αυτό αναγκάζει το WebScarab να αντιγράψει κάποιες δομές δεδομένων μνήμης στο δίσκο. Κατόπιν αντιγράφουμε ή μετακινούμε τον προσωρινό κατάλογο συνόδου σε μια νέα θέση. Τελειώνοντας και με τις απαραίτητες ρυθμίσεις του WebScarab είμαστε έτοιμοι να ξεκινήσουμε τη λειτουργία των ελέγχων. Πριν από αυτό, όμως, καλό θα ήταν να αναφερθούμε πιο διεξοδικά στα διάφορα plug-ins του.

Επιπρόσθετα

Στη συνέχεια θα δούμε πιο αναλυτικά τις λειτουργίες των διαφόρων plug-ins, που περιλαμβάνονται στο WebScarab. Όπως είπαμε, το WebScarab χρησιμοποιεί την έννοια των plug-ins, τα οποία μπορούν να δημιουργήσουν συνομιλίες και να τις αναλύσουν. Το plug-in, που δημιουργεί συνομιλίες, χρησιμοποιεί μια συγκεκριμένη μέθοδο, για να αποφασίσει ποιους πόρους να ζητήσει από το server, πώς να παραμετροποιήσει τις επικεφαλίδες του αιτήματος, κλπ και στη συνέχεια υποβάλλει το αίτημα στο server. Κατόπιν λαμβάνοντας την απάντηση από το server μπορεί να εκτελεί κάποιο

υπολογισμό στην απάντηση και μπορεί στη συνέχεια να αποφασίσει, εάν είναι ανάγκη να υποβληθεί εκείνη η συνομιλία στο framework. Όλες οι συνομιλίες, που υποβάλλονται στο framework, διανέμονται σε όλα τα plug-ins, που εγκαθίστανται σε αυτό. Κάθε plug-in μπορεί στη συνέχεια να εκτελέσει κάποια ανάλυση σε συγκεκριμένη συνομιλία. Τα plug-ins του WebScarab είναι:

- Proxy
- Manual Request
- Web Services
- Spider
- XSS/CRLF
- SessionID Analysis
- Scripted
- Fragments
- Compare
- Fuzzer
- Search

Proxy Plug-in

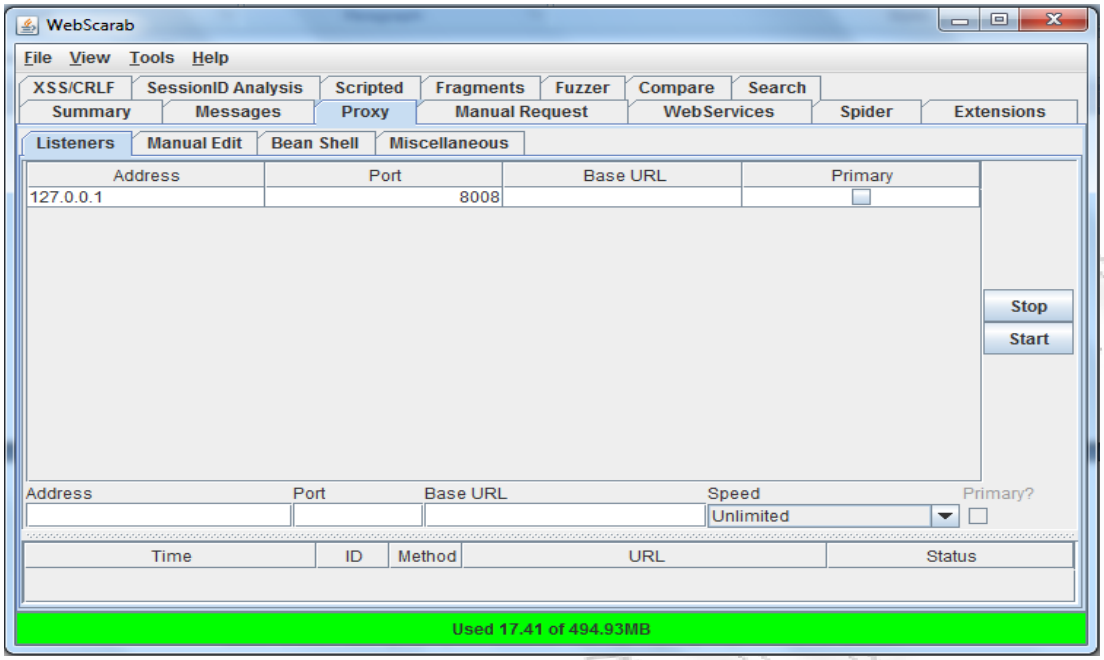
Το proxy plug-in εγκαθιστά έναν HTTP proxy, ο οποίος εξ' ορισμού ακούει στην localhost πόρτα 8008. Προκειμένου να χρησιμοποιηθεί αυτός ο ακροατής, πρέπει να διαμορφώσουμε τον browser έτσι, ώστε να χρησιμοποιεί το WebScarab ως upstream proxy. Μόλις γίνει αυτό, οποιοδήποτε αίτημα κάνει ο browser, θα οδηγηθεί μέσω του WebScarab, θα καταγραφεί και θα αναλυθεί.

Επίσης, το WebScarab έχει την υποστήριξη για παρεμπόδιση των SSL- κρυπτογραφημένων αιτημάτων HTTPS. Εάν ο browser έχει διαμορφωθεί, για να χρησιμοποιεί το WebScarab για αιτήματα SSL, το WebScarab θα χρησιμοποιήσει το δικό του μη έμπιστο SSL πιστοποιητικό server, για να διαπραγματευτεί μια κρυπτογραφημένη σύνοδο με τον browser, προκειμένου να διαβαστεί το αίτημα, που ο browser υποβάλλει στον "ασφαλή" server. Φυσικά από τη στιγμή, που ο browser δεν εμπιστεύεται το πιστοποιητικό, που χρησιμοποιεί το WebScarab, ο browser πρέπει να μας προειδοποιήσει ότι το πιστοποιητικό είναι "un-trusted", και παρέχει την επιλογή, αν θέλουμε να συνεχίσουμε.

- **Proxy listeners**

Το WebScarab υποστηρίζει πολλαπλάσιους ακροατές HTTP. Μπορούμε να καθορίσουμε την IP διεύθυνση και την πόρτα, στην οποία ο ακροατής θα ακούει.

Σημείωση: Εξ ορισμού, το WebScarab ακούει μόνο στο localhost, δηλαδή IP: 127.0.0.1, ώστε να περιορισθεί η δυνατότητα ακρόασης από κάποιο ανώνυμο αναρμόδιο.



Εικόνα ***:* Καθορισμός του Listener και της πόρτας, στην οποία θα ακούει.**

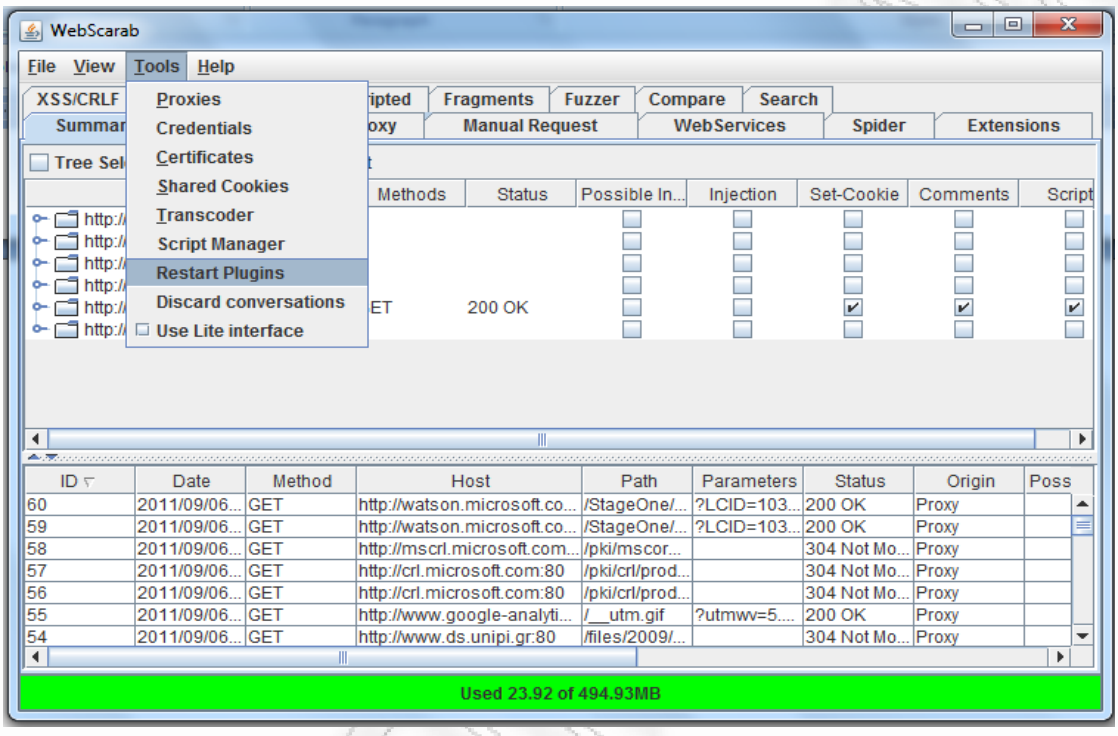
Το WebScarab μας επιτρέπει να καθορίσουμε μια "διεύθυνση βάσεων" για έναν ακροατή. Η διεύθυνση βάσεων καθοδηγεί τον ακροατή να λειτουργήσει ως αντίστροφος proxy. Θα πρέπει δε να δημιουργηθεί ως μια HTTP ή HTTPS URL. Κατ' αυτό τον τρόπο θα ενεργήσει ως web server παρά ως proxy server και θα κατασκευάσει το URL με το συνδυασμό της βάσης URL και της διαδρομής, που εμφανίζεται στη γραμμή αιτήματος. Εάν η βάση URL είναι ένα HTTPS URL, θα διαπραγματευτεί αμέσως μια σήραγγα SSL, πριν από την προσπάθεια για διαβίβαση του αιτήματος από τον browser. Αυτό είναι χρήσιμο στην περίπτωση, που χρησιμοποιούμε ένα συνηθισμένο client βασισμένο στο πρωτόκολλο HTTP, το οποίο δεν υποστηρίζει τη διαμόρφωση ενός upstream proxy. Απλά αλλάζουμε το αρχείο των Hosts στον υπολογιστή, στον οποίο τρέχει ο client, για να δείξουμε τον ιστότοπο μέσα από αίτημα προς τον υπολογιστή, στον οποίο τρέχει το WebScarab. Έτσι το WebScarab θα λάβει τα αιτήματα για τον συγκεκριμένο ιστότοπο.

Δεν μπορούμε σε μια τέτοια περίπτωση, να τρέξουμε το WebScarab και την εφαρμογή πελατών στον ίδιο υπολογιστή, γιατί το WebScarab, κατά την προσπάθειά του να συνδεθεί με τον ιστότοπο, θα πάρει την ίδια διεύθυνση IP από το αρχείο Hosts και θα καταλήξουμε σε ένα φαύλο κύκλο. Επίσης, πρέπει να προσθέσουμε το πιστοποιητικό WebScarab στον κατάλογο αναγνωρισμένων πιστοποιητικών των εφαρμογών των clients, για να αποτρέψουμε με αυτό τον τρόπο τα λάθη από πιστοποιητικά. Εάν δεν

είναι απαραίτητο να κάνουμε κάτι τέτοιο, η εφαρμογή client θα είναι ευπαθής σε μια "man in the middle

attack", γιατί δεν θα ελέγχει κατάλληλα το πιστοποιητικό.

Η επιλογή προσομοιωτή δικτύων επιτρέπει στο χρήστη να περιορίσει το εύρος ζώνης, που είναι διαθέσιμο στον client, και εισάγει τεχνητές λανθασμένες καταστάσεις. Υπάρχει επίσης μια επιλογή για ενεργοποίηση των plug-ins (εικόνα 15). Όταν αυτή δεν επιλέγεται τα διάφορα proxy plug-ins δεν θα επικαλεσθούν όταν φαίνονται τα αιτήματα και οι απαντήσεις. Εάν τα plug-ins δεν φαίνονται ή οι υποκλοπές μας δεν συμβαίνουν, όπως θα θέλαμε, πρέπει να ελέγξουμε, για να σιγουρευτούμε ότι η αντίστοιχη επιλογή έχει γίνει.



Εικόνα **: Επιλογή Restart Plug-ins για ενεργοποίηση των plug-ins κατά τη δημιουργία των αιτημάτων.**

Στην πλατφόρμα των Windows, εάν το αρχείο W32WinInet.dll φορτώνεται από το WebScarab, ο ακροατής που είναι χαρακτηρισμένος ως κύριος ακροατής αυτόματα "θα πειρατέψει" τον Internet Explorer, όταν αρχίζει και θα διαμορφώσει τον Internet Explorer και όλες τις άλλες βασισμένες στο WinInet εφαρμογές έτσι, ώστε να χρησιμοποιεί το WebScarab ως proxy. Όταν ο ακροατής σταματήσει χειροκίνητα ή εάν τερματίσει το WebScarab, οι αρχικές ρυθμίσεις θα αποκατασταθούν. Προφανώς, δεν έχει κανένα νόημα να διαμορφώσουμε πολλαπλούς κύριους ακροατές, δεδομένου ότι θα συγκρουστούν ο ένας με τον άλλον και αυτό θα οδηγήσει πιθανώς σε λανθασμένες ρυθμίσεις του proxy, κατά την αποκατάσταση που θα ακολουθήσει μετά τον τερματισμό του WebScarab.

- **Active Conversations (Ενεργές συνομιλίες)**

Κάτω από την περιοχή της διαμόρφωσης του ακροατή είναι ένας πίνακας, που παρουσιάζει πρόσφατα και τρέχοντα αιτήματα, που έχουν αντιμετωπιστεί από τους διάφορους ακροατές. Αυτό μπορεί

να είναι χρήσιμο κατά την ανίχνευση λαθών συνδεσιμότητας, δεδομένου ότι ο proxy δεν θα αποθηκεύσει τα αποτυχημένα αιτήματα στη γενική περίληψη (Summary).

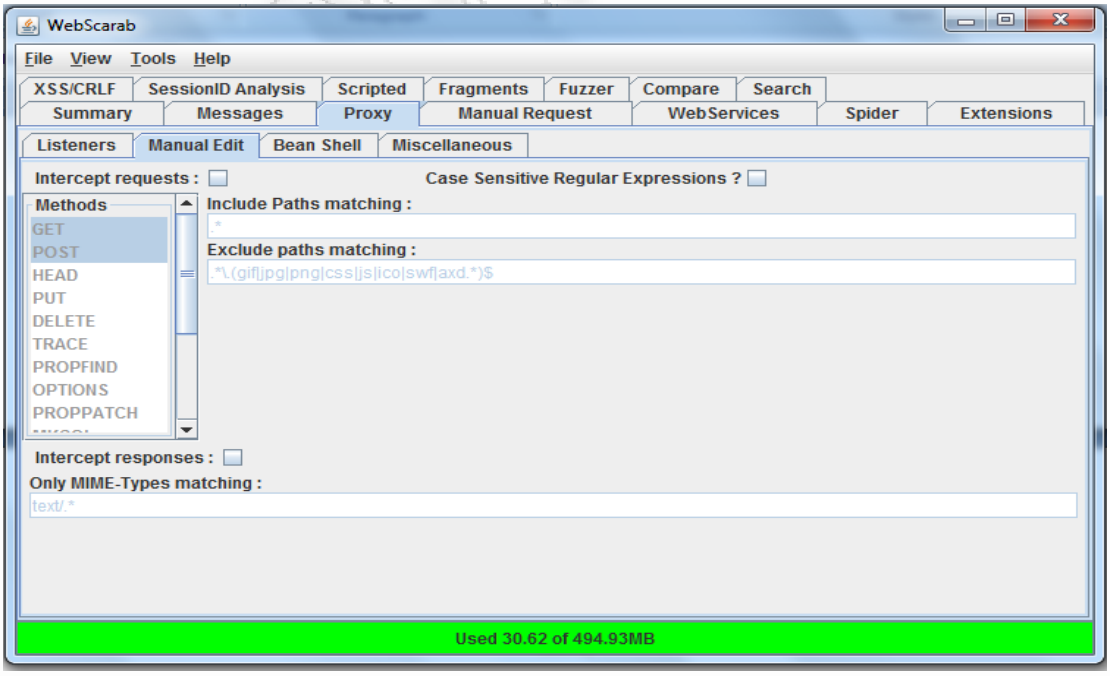
- **Proxy plug-ins**

Είναι δυνατό, εφόσον υπάρχουν βέβαια οι κατάλληλες γνώσεις προγραμματισμού, να γράψουμε proxy plug-ins, τα οποία να εκτελούν τις διάφορες τροποποιήσεις στα αιτήματα και τις απαντήσεις κατά τη μεταφορά αιτημάτων μέσω του proxy. Τα υπάρχοντα proxy plug-ins είναι:

- Manual Intercept (Χειρωνακτική παρεμπόδιση)
- Bean Shell
- Reveal hidden form fields(Αποκάλυψη κρυμμένων πεδίων φορμών)
- Prevent browser caching content (Αποτροπή αποθήκευσης του περιεχόμενου στη μνήμη cache του browser)
- Inject known cookies into requests (Εισαγωγή γνωστών cookies σε αιτήσεις)
- Extract cookies from responses (Εξαγωγή cookies από απαντήσεις)
- Remove NTLM authentication headers (Απομάκρυνση NTLM επικεφαλίδων αυθεντικοποίησης)

- **Manual Intercept**

Το Manual Intercept proxy plug-in επιτρέπει στο χρήστη να υποκλέψει τα αιτήματα από τον browser προς το server και τις απαντήσεις από τον server προς το browser, να τις επιθεωρήσει και να τις τροποποιήσει προαιρετικά πριν από τη μετάδοση τους. Αυτό είναι ιδιαίτερα χρήσιμο, όταν κάποιος θέλει να υποβάλλει μια φόρμα σε ένα web server, αλλά η επικύρωση JavaScript απορρίπτει τις τιμές, που θα επιθυμούσε να υποβάλει. Το αίτημα μπορεί να αλλάξει, αφότου έχει εκτελεσθεί η επικύρωση.



Εικόνα **: Η επιλογή Manual Edit του proxy plug-in.**

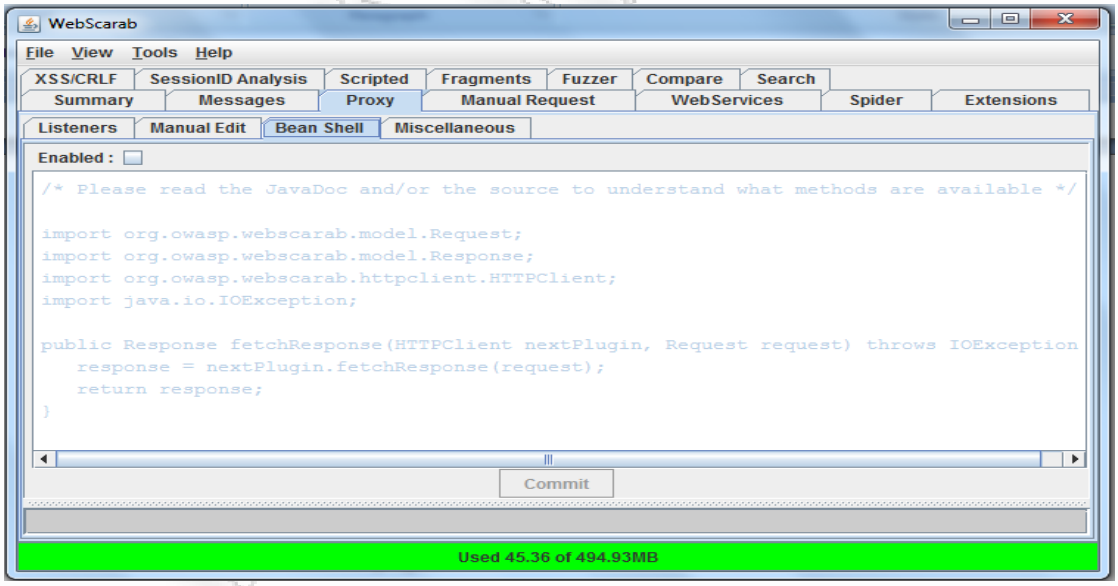
Ο χρήστης έχει την επιλογή να υποκλέψει αιτήματα με συγκεκριμένες μεθόδους ανάλογα με την επιλογή, που θα κάνει από τις μεθόδους του δοθέντος καταλόγου. Επίσης, έχει τη δυνατότητα, να καθορίσει μια κανονική έκφραση της Java, που το URL της πρέπει να ταιριάζει προκειμένου να υποκλαπεί. Έχει επίσης την επιλογή να καθορίσει μια κανονική έκφραση, σύμφωνη με τα URLs που δεν πρέπει να υποκλαπούν. Μπορεί επίσης να καθορίσει, εάν η κανονική έκφραση του, πρέπει να αξιολογηθεί με τη μορφή γραμμάτων, κεφαλαίων και μικρών, παραδείγματος χάριν, όπου η επέκταση αρχείων εικόνας είναι .GIF.

Το WebScarab μπορεί επίσης να υποκλέψει τις απαντήσεις, που προέρχονται από τον server, και να τροποποιήσει εκείνες, πριν να αναμεταδοθούν στον browser. Μπορούμε να καθορίσουμε μια κανονική έκφραση, που η επικεφαλίδα περιεχόμενο-τύπος της απάντησης πρέπει να ταιριάζει, προκειμένου να υποκλαπεί.

- **Bean Shell**

Το Bean Shell proxy plug-in επιτρέπει τις προκαθορισμένες τροποποιήσεις του αιτήματος και της απάντησης. Αυτό είναι ιδιαίτερα χρήσιμο, όταν θέλει να εκτελέσει κάποιος την ίδια τροποποίηση σε διάφορες συνομιλίες, ή εάν η τροποποίηση είναι ιδιαίτερα σύνθετη. Το script, που δημιουργούμε, έχει πλήρη πρόσβαση στο αίτημα, πριν αυτό σταλεί στον server, και στην απάντηση, πριν αυτή σταλεί στο browser. Τα αντικείμενα αιτήματος και απάντησης είναι πλήρως τεκμηριωμένα στο JavaDoc για τις κατηγορίες:

- org.owasp.webscarab.model.Request
- org.owasp.webscarab.model.Response.



Εικόνα ***:* Η επιλογή Bean Shell του proxy plug-in.**

- **Miscellaneous plug-ins**

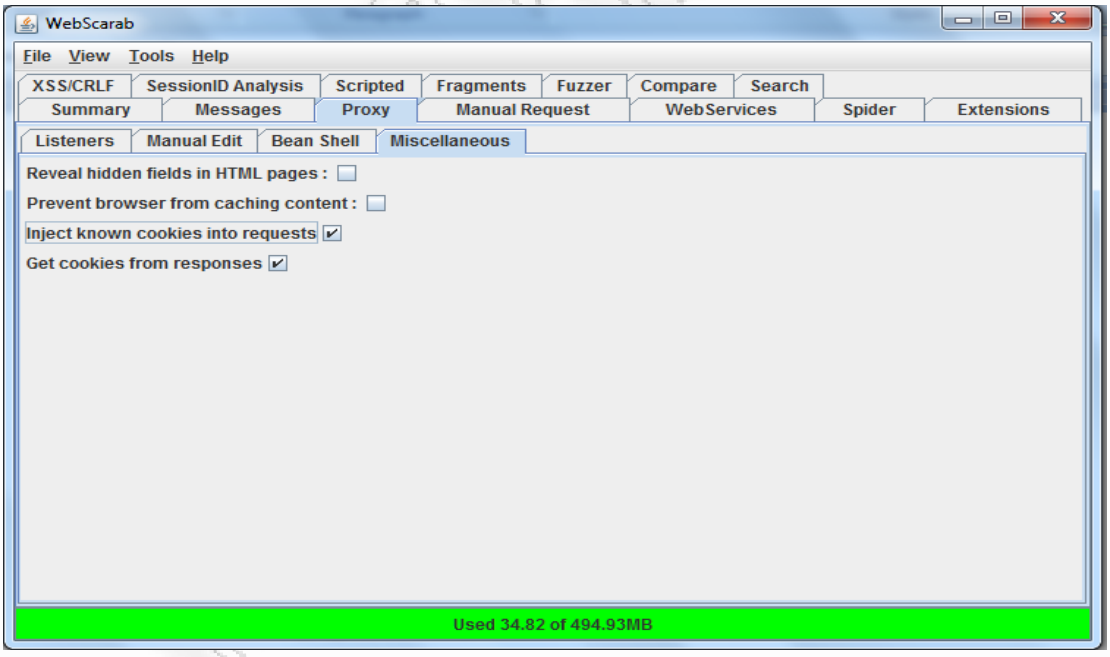
Υπάρχουν διάφορα plug-ins, που μπορούν να λειτουργήσουν επάνω στις συνομιλίες μεταξύ browser και server, όπως φαίνεται και στην ακόλουθη εικόνα 18.

- Το "**Reveal hidden fields**" plug-in αλλάζει οποιαδήποτε πεδία φορμών τύπου "hidden", σε πεδία τύπου "text". Αυτό αναγκάζει τον browser να εμφανίσει αυτά τα πεδία και διευκολύνει έτσι το χρήστη να παρατηρεί πότε χρησιμοποιούνται τέτοια κρυφά πεδία, καθώς και να τα τροποποιεί.

- Το "**prevent caching**" plug-in αφαιρεί κάθε ετικέτα "if-modified-since" ή παρόμοιες ετικέτες από τα αιτήματα, για να εξασφαλιστεί έτσι ότι το WebScarab έχει πάντα ένα αντίγραφο του σώματος της απάντησης, αντί να παρέχει κάθε φορά την άδεια στο browser να χρησιμοποιεί ένα τοπικά αποθηκευμένο αντίγραφο της.

- Το "**Inject cookies**" plug-in επιτρέπει στο WebScarab να αντιγράψει οποιαδήποτε cookies στον browser ή να παρέχει cookies, που ο browser δεν γνωρίζει. Τα cookies ανακτώνται από τη λίστα "Shared cookies " και φιλτράρονται για δυνατότητα χρησιμοποίησης σύμφωνα με τις ιδιότητες του εκάστοτε domain και path.

- Το "**Get cookies**" plug-in εξάγει κάθε επικεφαλίδα "**Set-Cookie**" από τις απαντήσεις, που περνούν μέσω του proxy, και προσθέτει αυτές στη λίστα "**Shared cookies**". Αυτά μπορούν αργότερα να χρησιμοποιηθούν από άλλα plug-ins, εάν επιθυμούμε.



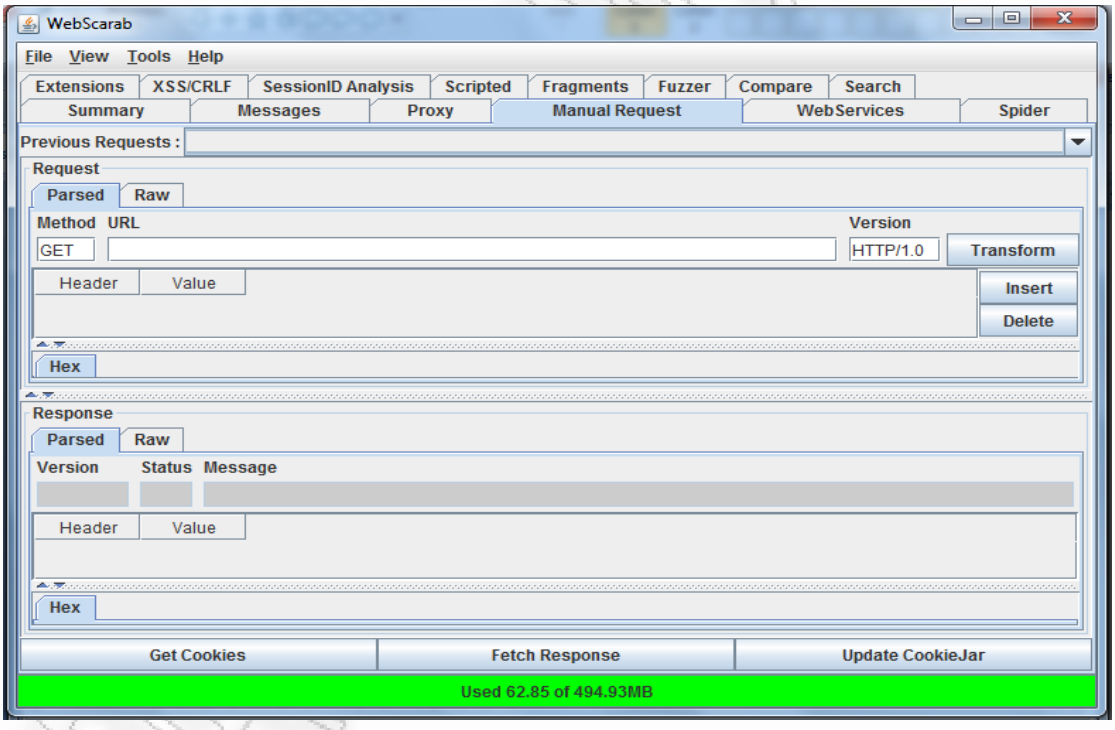
Εικόνα ***: Η επιλογή Miscellaneous του proxy plug-in.***

Σημείωση: είναι αρκετά ενδιαφέρον το ότι το WebScarab είναι δυνατόν να εκτελέσει και SPNEGO (Simple and Protected Negotiation Mechanism) αυθεντικοποίηση μέσω του proxy. Ειδικότερα,

δεδομένου ότι το WebScarab υποστηρίζει τις μόνιμες συνδέσεις, είναι δυνατόν να ελέγξει τις συνομιλίες μέσω ενός server, που χρησιμοποιεί SPNEGO, αν και κάθε τμήμα της διαπραγμάτευσης πρωτοκόλλου μεταξύ του browser και του server, θα καταγραφεί και θα οδηγήσει σε διάφορες απαντήσεις τύπου 401, που καταλήγουν στα log files. Αυτό μπορεί να λειτουργήσει ακόμη και για τις συνδέσεις SSL.

Manual Request Plug-in

Το Manual Request plug-in επιτρέπει στο χρήστη να επαναλάβει χειροκίνητα ένα αίτημα, το οποίο θα σταλεί στο server. Είναι επίσης δυνατόν να επαναληφθεί ένα προηγούμενο αίτημα μας επιλέγοντας το από το drop down μενού επιλογής. Τα προηγούμενα αιτήματα που φορτώνονται στον editor, μπορούν επίσης να επεξεργαστούν, πριν σταλούν στο server. Κάνοντας την επιλογή "Fetch Response", το WebScarab στέλνει το αίτημα στον κατάλληλο server και σώζει τη συνομιλία για τυχόν ανάλυση από άλλα plug-ins. Επιλέγοντας "Get cookies" το WebScarab παίρνει τα σχετικά με το ζητούμενο URL cookies από την λίστα "Shared Cookies" και τα προσθέτει στο αίτημα. Η επιλογή "Update CookieJar" ψάχνει οποιεσδήποτε επικεφαλίδες "Set-Cookie" στην απάντηση, που ανακτήθηκε, και τις προσθέτει στη λίστα "Shared Cookies".



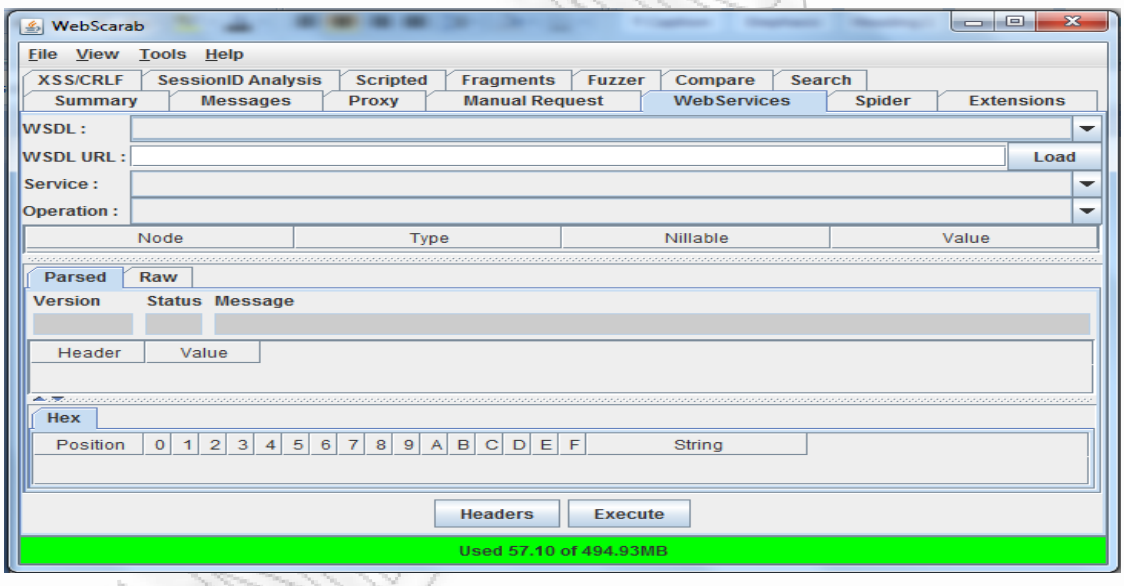
Εικόνα ***:*** **Το Manual Request plug-in.**

Web Services Plug-in

Το Web Services plug-in επιτρέπει στο χρήστη να χρησιμοποιήσει χειροκίνητα μια web υπηρεσία SOAP. Το plug-in αυτό προσδιορίζει αυτόματα τις συνομιλίες, που περιέχουν έγγραφα WSDL (Web Service definition Language) και τις παρουσιάζει σε ένα drop down μενού επιλογής. Το WSDL είναι μια φόρμα XML, που περιγράφει τις υπηρεσίες δικτύου ως σύνολο τερματικών σημείων (θυρών), μέσα σε

μηνύματα, που περιέχουν πληροφορίες βασισμένες σε στοιχεία ή σε διαδικασίες. Οι διαδικασίες και τα μηνύματα περιγράφονται περιληπτικά και συνδέονται στη συνέχεια σε ένα συγκεκριμένο πρωτόκολλο δικτύων και σε ένα σχήμα μηνυμάτων, για να καθορίσουν ένα τερματικό σημείο. Τα σχετιζόμενα αλλά ξεχωριστά τερματικά σημεία, συνδυάζονται στα περιληπτικά τερματικά σημεία (υπηρεσίες). Το WSDL είναι επεκτάσιμο έτσι ώστε να επιτρέπει την περιγραφή των τερματικών σημείων και των μηνυμάτων τους ανεξάρτητα από τη μορφή του μηνύματος ή τα πρωτόκολλα δικτύων, που χρησιμοποιούνται για την επικοινωνία.

Μπορούμε να πληκτρολογήσουμε το URL, για να ανακτηθεί το αρχείο WSDL από το plug-in. Εάν το WSDL δεν είναι προσιτό μέσω ενός HTTP ή ενός HTTPS URL, ο χρήστης μπορεί ακόμη να έχει πρόσβαση σε ένα τοπικό αντίγραφο του WSDL απλά παρέχοντας την διαδρομή προς το τοπικό αντίγραφο στο πεδίο του WSDL URL και κάνοντας την επιλογή “Load”. Το plug-in παρουσιάζει έπειτα τις διαθέσιμες υπηρεσίες και διαδικασίες και δημιουργεί αυτόματα ένα δέντρο, που περιέχει τις παραμέτρους της απαιτούμενης υπηρεσίας. Κατόπιν ο χρήστης μπορεί να επιλέξει "Execute", για να υποβάλει το αίτημα στο server και να λάβει την απάντηση. Προς το παρόν τα plug-ins των υπηρεσιών ιστού είναι ακόμα πολύ "ανώριμα" και είναι πιθανό να παρουσιάζουν κάποια λάθη. Αυτή η αρχική εφαρμογή έχει μόνο την υποστήριξη για RPC (Remote Procedure Call) κωδικοποιημένες μεταφορές.

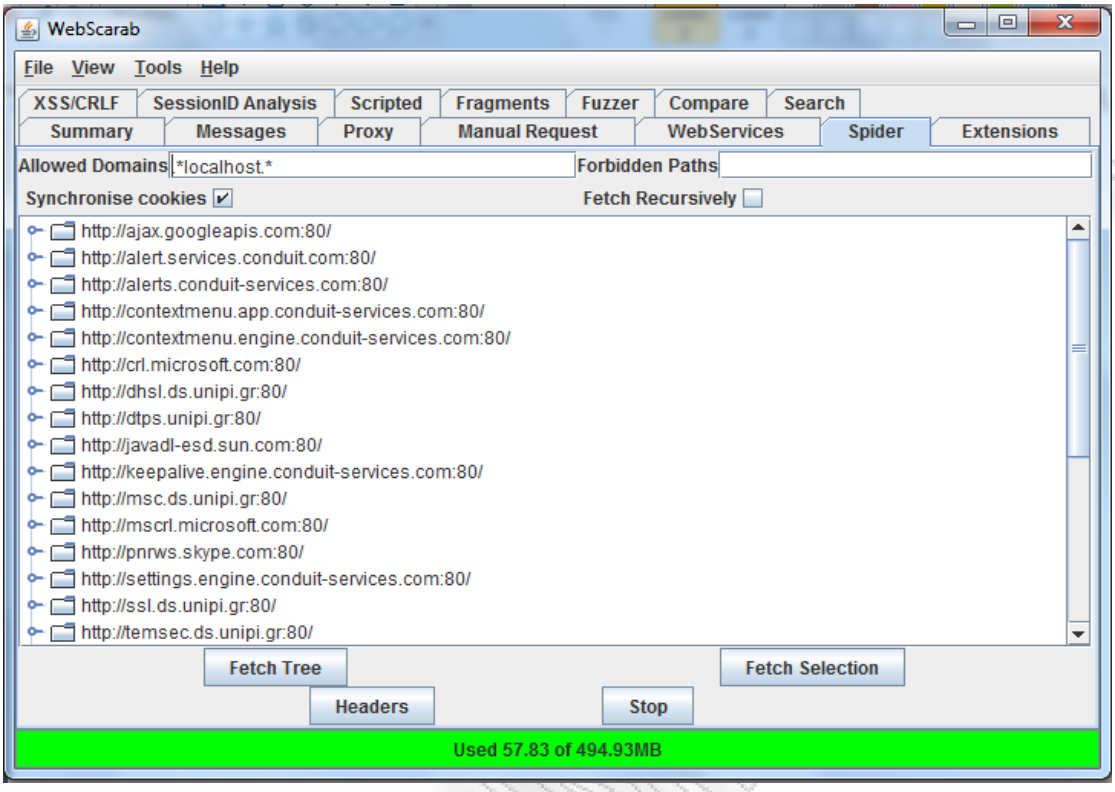


Εικόνα ***:*** **Το WebServices plug-in.**

Spider plug-in

Το Spider plug-in αναλύει τις απαντήσεις, ώστε να προσδιορίσει οποιεσδήποτε συνδέσεις (links) στο σώμα της απάντησης ή στην επικεφαλίδα "Location". Εάν το αντιπροσωπευτικό URL δεν φαίνεται, το URL προστίθεται σε ένα δέντρο και μπορεί να μεταφορτωθεί αυτόματα, όποτε εμείς το επιδιώξουμε. Το WebScarab έχει δύο τρόπους να “πιάνει” τις απαρατήρητες συνδέσεις. Η επιλογή "Fetch Tree" απαριθμεί όλες αυτές τις άγνωστες συνδέσεις κάτω από τον επιλεγμένο κόμβο και τις τοποθετεί στη

σειρά για μετέπειτα ανάκτηση. Η επιλογή "Fetch Selection" τοποθετεί στη σειρά μόνο τους επιλεγμένους κόμβους για ανάκτηση.



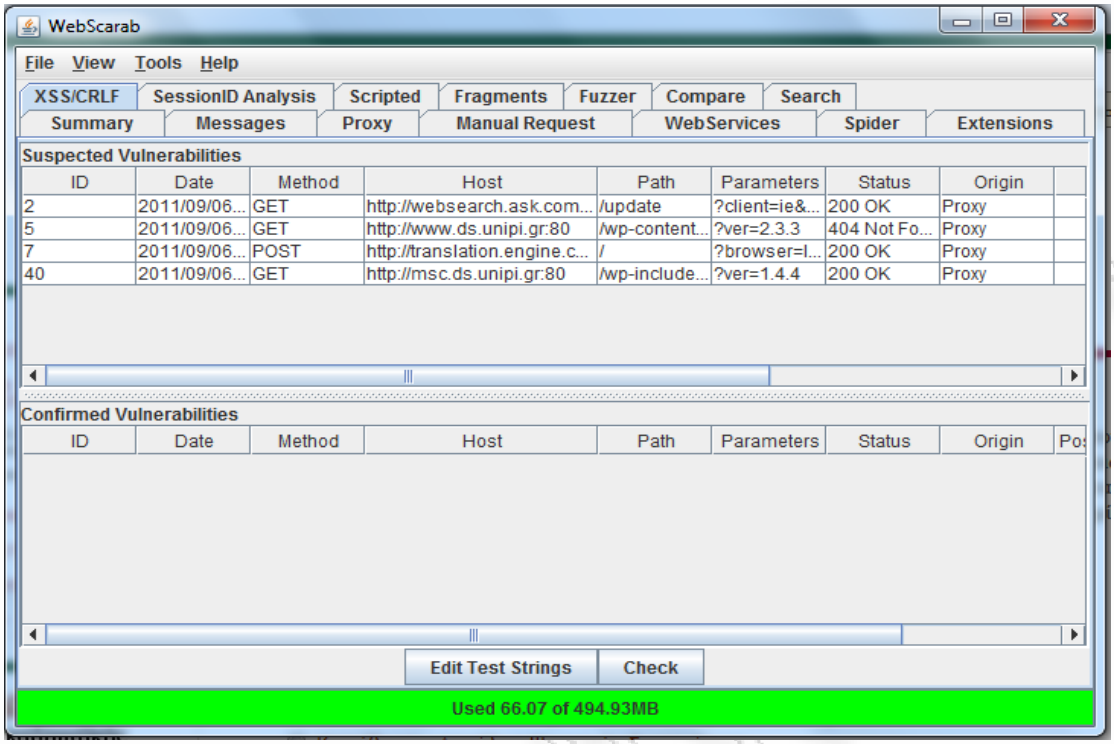
Εικόνα ***:*** **Το Spider plug-in**.

XSS/CRLF plug-in

Το XSS/CRLF plug-in εξετάζει ύποπτα HTTP αιτήματα, για cross-site scripting και CRLF injection (HTTP response splitting) ευπάθειες. Το plug-in αυτό δεν έρπετε στον ιστότοπο με σκοπό να ανακαλύψει ευπαθείς URLs, όπως συμβαίνει με άλλα εργαλεία. Αντίθετα αναλύει παθητικά όλες τις HTTP συνομιλίες που περνούν μέσω του WebScarab. Το plug-in εξετάζει κάθε αίτημα και απάντηση, για να ελέγξει αν:

- Κάθε τιμή των GET-POST παραμέτρων απεικονίζεται στο κυρίως σώμα της HTTP απάντησης, το οποίο δείχνει μια πιθανότητα ύπαρξης XSS ευπάθειας.
- Κάθε τιμή των GET/POST παραμέτρων απεικονίζεται στις HTTP επικεφαλίδες των απαντήσεων, το οποίο δείχνει μια πιθανότητα ύπαρξης CRLF ευπάθειας.

Ο πίνακας στο πάνω μισό μέρος του παραθύρου του plug-in δείχνει όλα τα ύποπτα HTTP αιτήματα για XSS/CRLF injection. Επιλέγοντας το κουμπί "Check" το plug-in προσπαθεί να πραγματοποιήσει ένα έλεγχο, εάν τα strings για XSS και CRLF injection περνούν επιτυχώς. Αν ο έλεγχος σε μια συνομιλία είναι επιτυχής, δηλαδή το string έγινε αποδεκτό ή δημιουργήθηκε μια νέα επικεφαλίδα HTTP, η συνομιλία θα εμφανιστεί στον πίνακα στο κάτω μέρος του παραθύρου. Αν δεν είναι επιτυχής ο πίνακας θα είναι κενός.



Εικόνα ***:*** **Το XSS/CRLF plug-in.**

Σημείωση: ανάλογα με τις ιδιαιτερότητες της ευπάθειας και την εφαρμογή, ο αυτοματοποιημένος έλεγχος ίσως να μην παράγει αξιόπιστα αποτελέσματα. Η εκμετάλλευση XSS είναι ακόμη δυνατή χωρίς τη χρήση χαρακτήρων, για παράδειγμα φιλτράρονται λιγότεροι ή περισσότεροι χαρακτήρες αν υφίσταται XSS χωρίς την ετικέτα "script". Για το XSS το plug-in στέλνει το string, που δίνεται από το χρήστη ως τιμή μιας ευπαθούς παραμέτρου και ελέγχει το κύριο σώμα της απάντησης για την ύπαρξη του string. Αν αυτό το string βρεθεί, τότε σημαίνει ότι ίσως η εφαρμογή είναι ευπαθής στο cross-site scripting. Για το CRLF injection (HTTP response splitting), το plug-in στέλνει επίσης το CRLF string, που δίνεται από το χρήστη, το οποίο υποτίθεται ότι δημιουργεί μια νέα επικεφαλίδα HTTP ως τιμή μιας ευπαθούς παραμέτρου και ελέγχει τις επικεφαλίδες HTTP απαντήσεων, για την ύπαρξη μιας νέας επικεφαλίδας HTTP που δημιουργείται από το CRLF string. Αν υπάρχει η επικεφαλίδα, τότε ίσως η εφαρμογή είναι ευπαθής στο response splitting.

SessionID Analysis

Το SessionID Analysis είναι ένα χρήσιμο plug-in, γιατί μπορεί να καθορίσει πόσο εύκολο είναι για κάποιον επιτιθέμενο να επιτεθεί με τη μέθοδο brute force στο SessionID ενός θύματος και να τα καταφέρει. Η αρχή λειτουργίας του βασίζεται στη συλλογή ενός δείγματος προσδιοριστικών συνόδου πιθανότατα από ένα cookie που τίθεται στον browser, αλλά ενδεχομένως και από ένα, που τίθεται σε ένα κρυφό πεδίο φόρμας στο σώμα της HTML σελίδας. Το WebScarab προσαρτεί σε όλα τα προσδιοριστικά συνόδων την ημερομηνία και το χρόνο, που αυτά συλλέχθηκαν και στη συνέχεια, μετά από την εκτέλεση

κάποιων υπολογισμών στην τιμή του string, για να την μετατρέψει σε έναν αριθμό, σχεδιάζει την τιμή σε σχέση με το χρόνο σε μια γραφική παράσταση. Το ανθρώπινο μάτι είναι πολύ περισσότερο αποδοτικό στον προσδιορισμό των σχεδίων από έναν υπολογιστή. Έτσι με τη χάραξη των τιμών σε μια γραφική παράσταση διευκολύνουμε έναν άνθρωπο να απεικονίσει την ακολουθία.

Ο προεπιλεγμένος τρόπος υπολογισμού προσδιορίζει το πιθανό σύνολο χαρακτήρων για κάθε θέση στο string της ταυτότητας συνόδου. Αποδίδει τη base-n μετατροπή του string σε έναν μεγάλο αριθμό. Αυτό είναι ένας αρκετά αποδοτικός τρόπος, για να μετατρέψει τα αυθαίρετα strings σε έναν αριθμό, δεδομένου ότι δεν κάνει οποιεσδήποτε υποθέσεις για το σύνολο χαρακτήρων και είναι εύλογα αποτελεσματικός στην εξάλειψη των σταθερών strings καθώς μειώνονται σε μια τιμή 0, αφού το σύνολο χαρακτήρων δεν αλλάζει. Είναι δυνατό να χρησιμοποιηθούν εναλλακτικοί τρόποι υπολογισμού, π.χ. ένας Base64 υπολογιστής, που ξέρει τι είναι Base64 σύνολο χαρακτήρων, αλλά δεν υπάρχει καμία υποστήριξη Διεπαφών Χρηστών για επιλογή διαφορετικών υπολογιστών. Γι' αυτό απαιτεί αλλαγές κώδικα για χρήση διαφορετικών υπολογιστών.

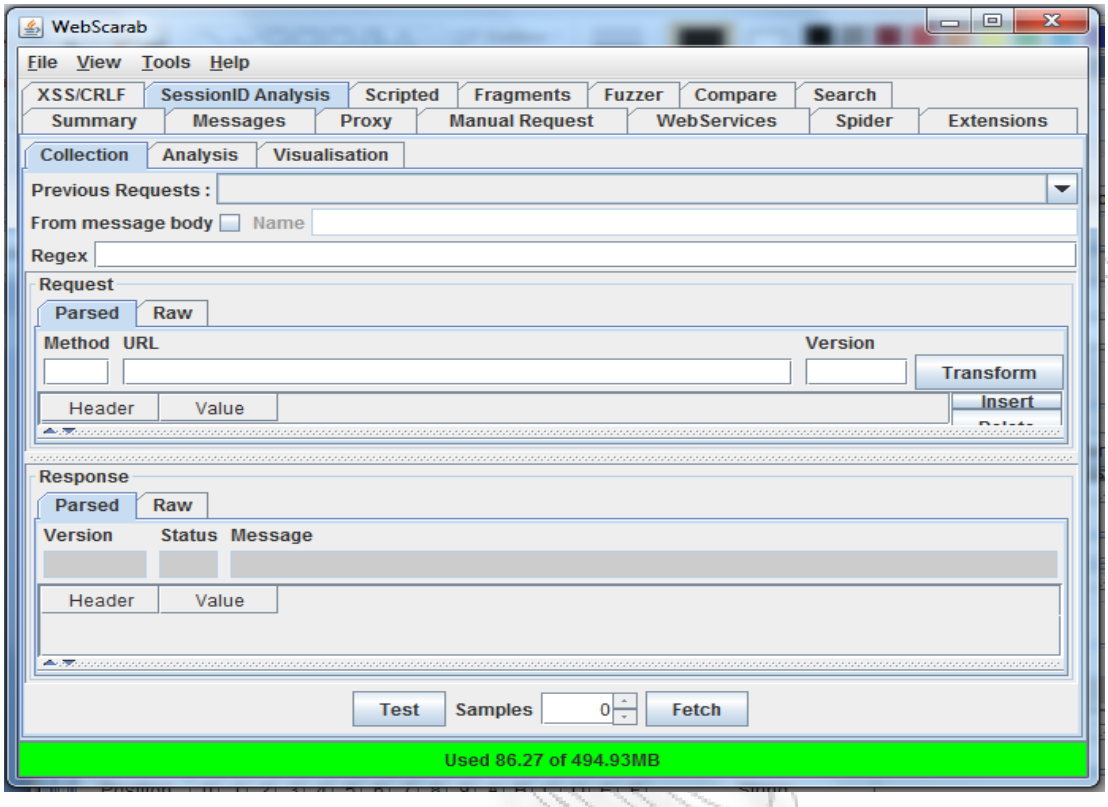
- **Collecting SessionIDs**

Υπάρχουν δύο τρόποι να συλλεχθούν τα SessionIDs. Ο πρώτος είναι από τα cookies στις επιγραφές απάντησης και ο δεύτερος με το ταίριασμα μιας κανονικής έκφρασης με το σώμα της απάντησης. Για να συλλέξουμε SessionIDs από cookies, πρέπει να σιγουρευτούμε ότι η επιλογή "From message body" δεν είναι επιλεγμένη. Για να συλλέξουμε SessionIDs από το σώμα μηνυμάτων, πρέπει να σιγουρευτούμε ότι η επιλογή "From message body" είναι επιλεγμένη. Παρέχουμε ένα όνομα για τη σειρά των SessionIDs το όνομα το οποίο υπολογίζεται αυτόματα για τα cookies, και μια κανονική έκφραση, που μπορεί να χρησιμοποιηθεί, για να προσδιορίσει το προσδιοριστικό από το υπόλοιπο του μηνύματος. Η κανονική έκφραση πρέπει να ταιριάξει με ολόκληρο το σώμα μηνυμάτων, γι' αυτό πιθανώς αρχίζει και τελειώνει με ".*". Το τμήμα του αντιστοιχούμενου κειμένου, που πρέπει να χρησιμοποιηθεί για το προσδιοριστικό συνόδου, πρέπει να περιβληθεί από παρενθέσεις, δηλαδή μια ομάδα. Το WebScarab θα συνθέσει πολλαπλές ομάδες σε ένα ενιαίο προσδιοριστικό, εάν αυτό είναι επιθυμητό.

Παράδειγμα κανονικών εκφράσεων:

· *.*id="(.....)".*: θα εξάγει ένα SessionID, 10 χαρακτήρων, που θα περιβάλλεται από εισαγωγικά και θα έχει το πρόθεμα "id=".*

Μόλις διευκρινίσουμε τι θέλουμε να συλλέξουμε, πρέπει να παρέχουμε ένα αίτημα, το οποίο θα οδηγήσει σε μια απάντηση, που περιέχει ένα νέο SessionID. Ο ευκολότερος τρόπος για να το κάνουμε αυτό, είναι να κοιτάξουμε στο Summary Panel, για να δούμε συνομιλίες, που έχουν μια τιμή στη στήλη "Set-Cookie". Κατόπιν, επιλέγουμε το αίτημα από το μενού επιλογών και χρησιμοποιούμε το συγκεκριμένο αίτημα, για να συλλεχθούν τα νέα SessionIDs. Εναλλακτικά, μπορούμε να δημιουργήσουμε χειρονακτικά το αίτημά μας ενδεχομένως μεταβάλλοντας την επιλογή GET σε HEAD, με σκοπό να μειωθεί το μέγεθος των δεδομένων, που πρέπει να ζητηθεί.



Εικόνα ***:*** **Το SessionID Analysis plug-in.**

Μόλις εισαχθεί το αίτημα, μπορούμε να εξετάσουμε την εγκατάσταση, για να σιγουρευτούμε ότι λειτουργεί. Αυτό μπορεί να γίνει επιλέγοντας το " Test ". Εάν ένα SessionID βρεθεί επιτυχώς, θα παρουσιαστεί ένα πλαίσιο διαλόγου με διάφορες λεπτομέρειες. Μόλις εξαχθούν επιτυχώς τα SessionIDs μέσω της επιλογής "Test", μπορούμε να καθορίσουμε διάφορα SessionIDs, που θα συλλεχτούν στο πεδίο " Samples ", και να επιλέξουμε το " Fetch ", για να αρχίσει η διαδικασία. Εάν καθορίσουμε ένα μεγάλο αριθμό SessionIDs και θέλουμε να σταματήσουμε τη διαδικασία συλλογής, προτού τελειώσει φυσικά, θέτουμε "0" μέσα στο πεδίο "Samples" και επιλέγουμε "Fetch".

- **SessionID Analysis**

Ενώ τα SessionIDs συλλέγονται, μπορούμε να μεταβούμε στην ετικέτα ανάλυσης και να δούμε τις τιμές τους. Χρησιμοποιώντας το μενού "Session Identifier", επιλέγουμε το SessionID, για το οποίο ενδιαφερόμαστε. Τότε, ο πίνακας ακριβώς από κάτω θα γεμίσει με τα SessionIDs, που έχουν συλλεχθεί, και θα επεκταθεί, καθώς θα φαίνονται νέα SessionIDs. Η πρώτη στήλη διευκρινίζει την ημερομηνία-το χρόνο, που το SessionIDs συλλέχθηκε, ενώ στη δεύτερη στήλη φαίνεται η πραγματική τιμή string του SessionID. Η τρίτη στήλη παρουσιάζει την υπολογισμένη τιμή, που μπορεί να αλλάξει κατά τη διάρκεια του χρόνου. Η τελευταία στήλη παρουσιάζει τη διαφορά μεταξύ των επόμενων υπολογισμένων τιμών. Εάν θέλουμε μπορούμε να αντιγράψουμε τα SessionIDs σε ένα πρόγραμμα υπολογισμών με λογιστικό φύλλο, για τυχόν εναλλακτική ανάλυση.

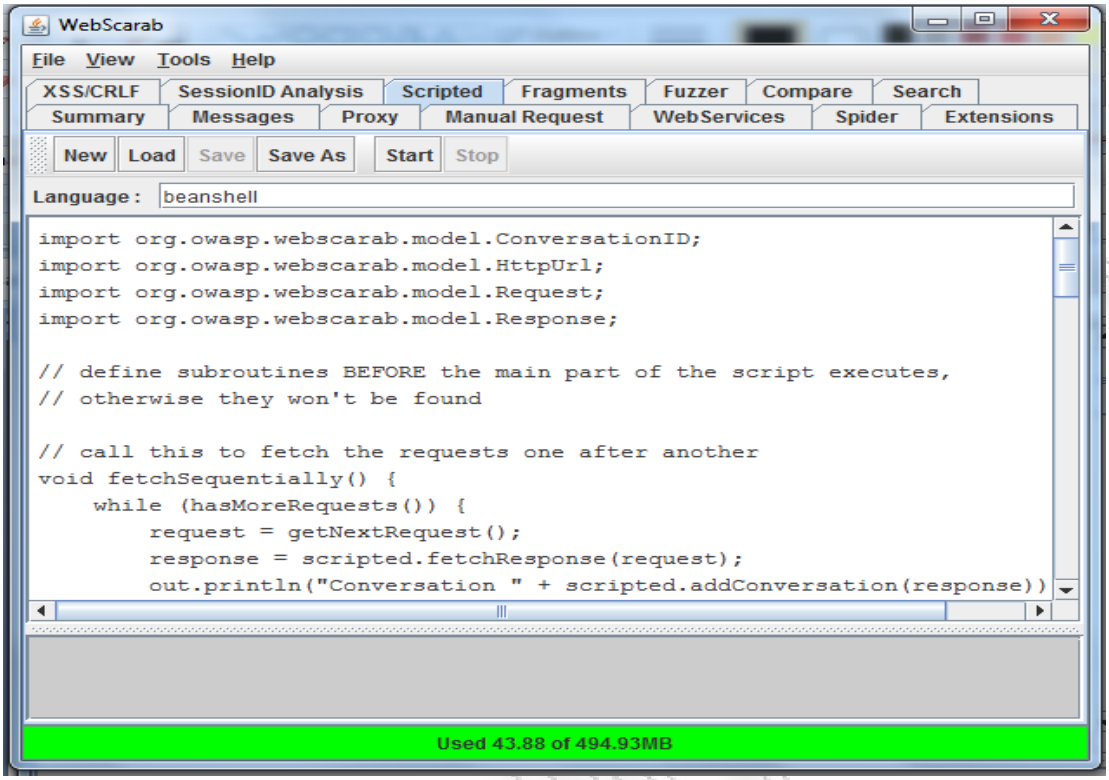
- **SessionID Visualization**

Η ετικέτα "Visualization" περιέχει μια γραφική παράσταση του επιλεγμένου προσδιοριστικού συνόδου. Η γραφική παράσταση δημιουργείται την πρώτη φορά, που η ετικέτα επιλέγεται. Εάν δεν επιλεγεί κανένα προσδιοριστικό συνόδου στην ετικέτα "Analysis", τότε κανένα σημείο δεν θα παρουσιαστεί στη γραφική παράσταση. Χρησιμοποιώντας τη γραφική παράσταση, είναι εύκολο να προσδιοριστούν γραμμικά ή επαναλαμβανόμενα πρότυπα στα SessionIDs, που συλλέχθηκαν.

Scripted plug-in

Αυτό το plug-in δίνει τη δυνατότητα στους χρήστες, χρησιμοποιώντας τις λειτουργίες του WebScarab να δημιουργήσουν scripts δοκιμής. Η υποστήριξη scripting παρέχεται από το Bean Scripting Framework (BSF), το οποίο σημαίνει ότι οποιαδήποτε BSF-υποστηριζόμενη scripting γλώσσα μπορεί να χρησιμοποιηθεί για να γραφτούν τα δοκιμαστικά scripts. Η γλώσσα προεπιλογής, που υποστηρίζεται είναι η Beanshell, η όποια είναι μια χαλαρή γλώσσα σαν τη Java. Για περισσότερες λεπτομέρειες, όσον αφορά τη γλώσσα BeanShell, μπορούμε να επισκεφθούμε την ιστοσελίδα http://www.beanshell.org. Εάν επιθυμούμε να χρησιμοποιήσουμε μια από τις άλλες scripting γλώσσες, όπως παραδείγματος χάριν τη Jython ή τη Groovy, πρέπει να τοποθετήσουμε τις αρμόδιες βιβλιοθήκες υποστήριξης στο ClassPath, πριν να τρέξουμε το WebScarab, και να σιγουρευτούμε ότι τα ονόματα αρχείων των scripts μας τελειώνουν με την κατάλληλη επέκταση για την επιλεγμένη γλώσσα. Η απαραίτητη επέκταση για BeanShell scripts είναι ".bsh".

Το Scripted plug-in εξάγει ένα αντικείμενο Java με τις μεθόδους που μας επιτρέπουν να αλληλεπιδράσουμε με το υπόλοιπο WebScarab, προσκομίζοντας μια απάντηση, ή προσθέτοντας μια συνομιλία στο πλαίσιο WebScarab, για την περαιτέρω ανάλυση από άλλα plug-ins. Είναι πολύ πιθανό ότι αυτό το plug-in θα τροποποιηθεί, δεδομένου ότι είναι ένα αρκετά νέο χαρακτηριστικό γνώρισμα του WebScarab. Επομένως, εάν η τεκμηριωμένη συμπεριφορά δεν ταιριάζει με αυτό που παρατηρούμε, πρέπει να ελέγξουμε τον πηγαίο κώδικα. Το εξαγόμενο αντικείμενο είναι τεκμηριωμένο μέσα στην ιστοσελίδα:



Εικόνα ***:*** **To Scripted plug-in.**

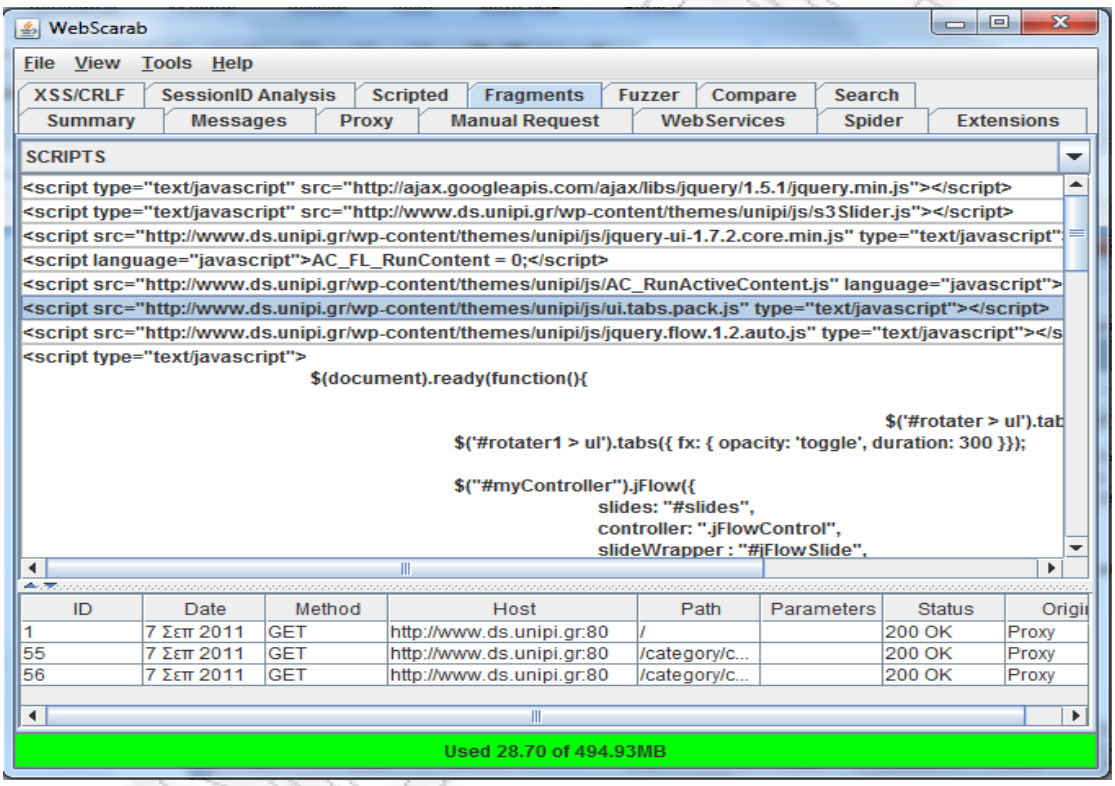
Οι σημαντικότερες μέθοδοι, που θα χρειασθεί πιθανώς να χρησιμοποιήσουμε, είναι οι εξής:

- Request getRequest(int id)
- Response fetchResponse(Request request) throws IOException
- boolean hasAsyncCapacity()
- void submitAsyncRequest(Request request)
- boolean isAsyncBusy()
- boolean hasAsyncResponse()
- Response getAsyncResponse() throws IOException
- ConversationID addConversation(Response response)

Η μέθοδος "async" επιτρέπει την προσκόμιση των απαντήσεων σε πολύ-νηματικές διεργασίες, ενώ η μέθοδος "fetchResponse" χρησιμοποιείτε σε απλές διεργασίες.

Fragments Plug-in

Το Fragments plug-in αναλύει τις απαντήσεις HTML και ψάχνει τα scripts και τα σχόλια. Αυτό μπορεί να είναι πολύ ενδιαφέρον, για να δούμε, πρώτον εάν υπάρχουν οποιεσδήποτε κρυφές συνδέσεις ή άλλος κώδικας διόρθωσης, και επίσης για να καταλάβουμε πώς οι σελίδες υποτίθεται ότι πρέπει να λειτουργούν. Η λίστα επιλογής στην κορυφή μας επιτρέπει να επιλέξουμε μεταξύ των δύο τύπων τμημάτων και ο κατάλογος αμέσως παρακάτω παρουσιάζει τμήματα της κατηγορίας, που έχει προσδιοριστεί. Επιλέγοντας ένα συγκεκριμένο τμήμα από τη λίστα, αυτόματα ανανεώνει τον πίνακα συνομιλιών δείχνοντας τις κυριότερες συνομιλίες, που περιλαμβάνονται στο τμήμα. Το πλαίσιο του Summary δείχνει επίσης ποιες συνομιλίες και ποια URLs έχουν σχόλια και scripts. Κάνοντας δεξί κλικ σε μια συνομιλία στο αίτημα, μπορούμε να επιλέξουμε ένα μενού, που θα δείχνει όλα τα τμήματα του συγκεκριμένου τύπου από τη συγκεκριμένη συνομιλία.

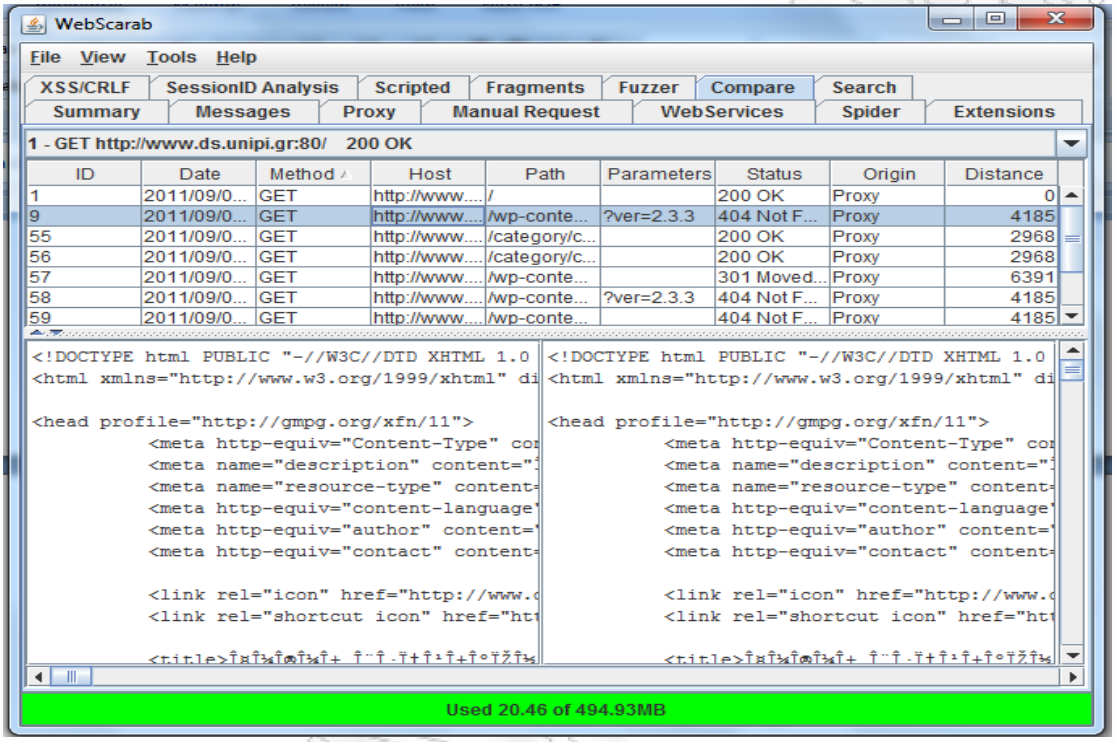


Εικόνα ***: Το Tab 'Scripts' του Fragments plug-in.***

Compare plug-in

Το Compare plug-in επιτρέπει στο χρήστη να κρίνει το βαθμό διαφοράς μεταξύ διάφορων απαντήσεων. Αυτό είναι χρήσιμο στην περίπτωση, που κάνουμε διάφορα αιτήματα για ένα συγκεκριμένο URL, ενδεχομένως μέσω του Scripted plug-in και επιθυμούμε να αξιολογήσουμε τα αποτελέσματα. Προφανώς, εάν μπορούμε να περιορίσουμε τις ομάδες παρόμοιων απαντήσεων, τότε μπορούμε να κερδίσουμε χρόνο. Επιλέγουμε τη συνομιλία "βάσεων", σύμφωνα με την οποία οι άλλες πρέπει να συγκριθούν χρησιμοποιώντας την λίστα από το μενού. Ο πίνακας συνομιλίας θα γεμίσει με τις συνομιλίες και η στήλη διαφοράς θα δείξει τον αριθμό των λέξεων, που διαφέρουν στις απαντήσεις. Ο αριθμός αυτός

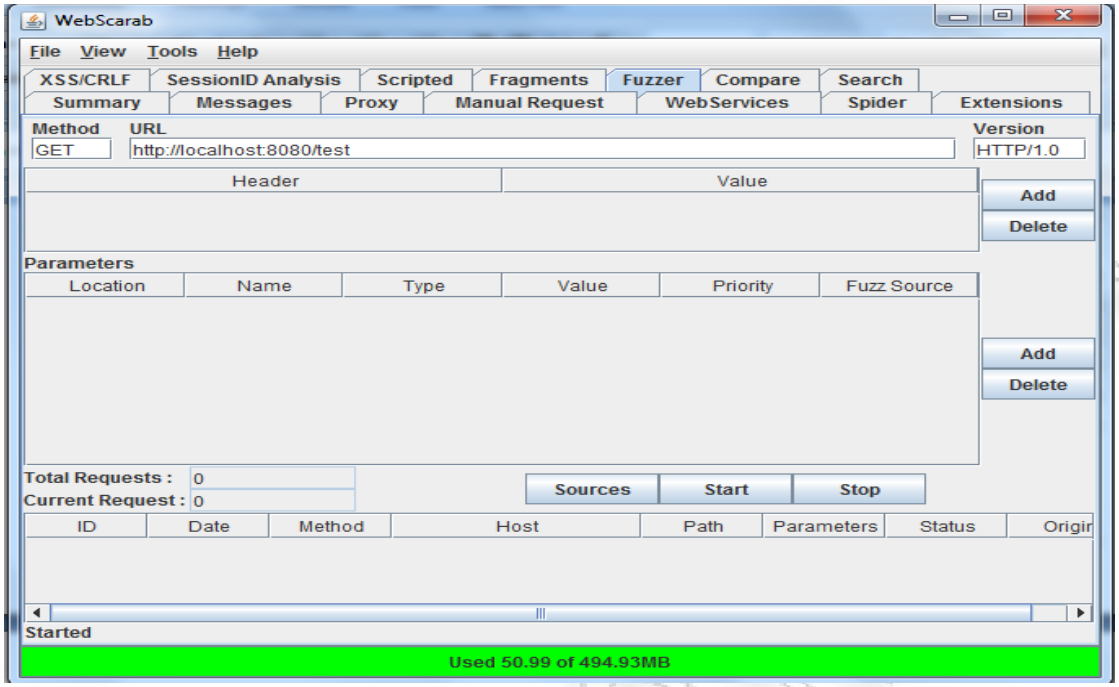
είναι πραγματικά η απόσταση επεξεργασίας Levenshtein (Levenshtein Edit Distance), η οποία παρουσιάζει τον αριθμό επεξεργασιών, εισαγωγές, αλλαγές και διαγραφές, που απαιτούνται για να μετασχηματισθεί η απάντηση "βάσεων" σύμφωνα με τον τρέχοντα στόχο. Για λόγους ταχύτητας το περιεχόμενο απάντησης είναι στις λέξεις, δεδομένου ότι ο αλγόριθμος Levenshtein είναι πολυπλοκότητας O(n*m) και πέφτει γρήγορα για μεγάλους αριθμούς συγκρίσεων. Οι δύο απαντήσεις παρουσιάζονται στη συνέχεια δίπλα-δίπλα στο χαμηλότερο μισό του παραθύρου, για να μπορέσει να γίνει πιο εύκολα η σύγκριση.



Εικόνα ***:* To Compare plug-in.**

Fuzzer plug-in

Το fuzzer plug-in βασικά μας επιτρέπει να δίνουμε ένα συνδυασμό τιμών παραμέτρων στο server. Η βασική ιδέα είναι ότι διαμορφώνουμε τη μέθοδο αιτήματος, το βασικό URL χωρίς παραμέτρους, την έκδοση του αιτήματος, οποιεσδήποτε επικεφαλίδες, π.χ. Host:, επικεφαλίδα εάν χρησιμοποιούμε HTTP/1.1, κ.λπ., και μια λίστα παραμέτρων. Δεν πρέπει να διαμορφώνουμε cookies εδώ. Μια παράμετρος καθορίζεται από τη θέση της (Path, Fragment, Query, Cookie, Body), το όνομά της, τον τύπο της (π.χ. string), την προκαθορισμένη τιμή της, την Fuzz προτεραιότητα και μια fuzz πηγή. Η προκαθορισμένη τιμή είναι η τιμή, που θα υποβληθεί, όταν καμία fuzz πηγή δεν καθορίζεται. Η fuzz προτεραιότητα καθορίζει, πώς συνδυάζονται οι διάφορες fuzz πηγές: εάν όλες οι προτεραιότητες έχουν την ίδια τιμή, ο αριθμός αιτημάτων, που έχουν υποβληθεί, θα είναι ο αριθμός στοιχείων στην πιο σύντομη fuzz πηγή. Εάν έχουν διαφορετικές τιμές, ο αριθμός αιτημάτων, που παράγονται, θα είναι το προϊόν του αριθμού στοιχείων σε κάθε επίπεδο.



Εικόνα ***:* To Fuzzer plug-in.**

Ένα καλό παράδειγμα θα ήταν εάν είχαμε μια λίστα ονομάτων χρήστη και κωδικών πρόσβασης. Εάν οι παράμετροι ονόματος χρήστη και κωδικού πρόσβασης είχαν την ίδια προτεραιότητα, τα ονόματα χρήστη και οι κωδικοί πρόσβασης θα καταναλώνονταν στο βήμα κλειδώματος. Εάν είχαν διαφορετικές προτεραιότητες, κάθε κωδικός πρόσβασης θα δοκιμαζόταν για κάθε όνομα χρήστη. Οι παράμετροι υποβάλλονται σε επεξεργασία με συγκεκριμένη σειρά, πρώτα path, μετά fragment, μετά query, μετά cookie και μετά body.

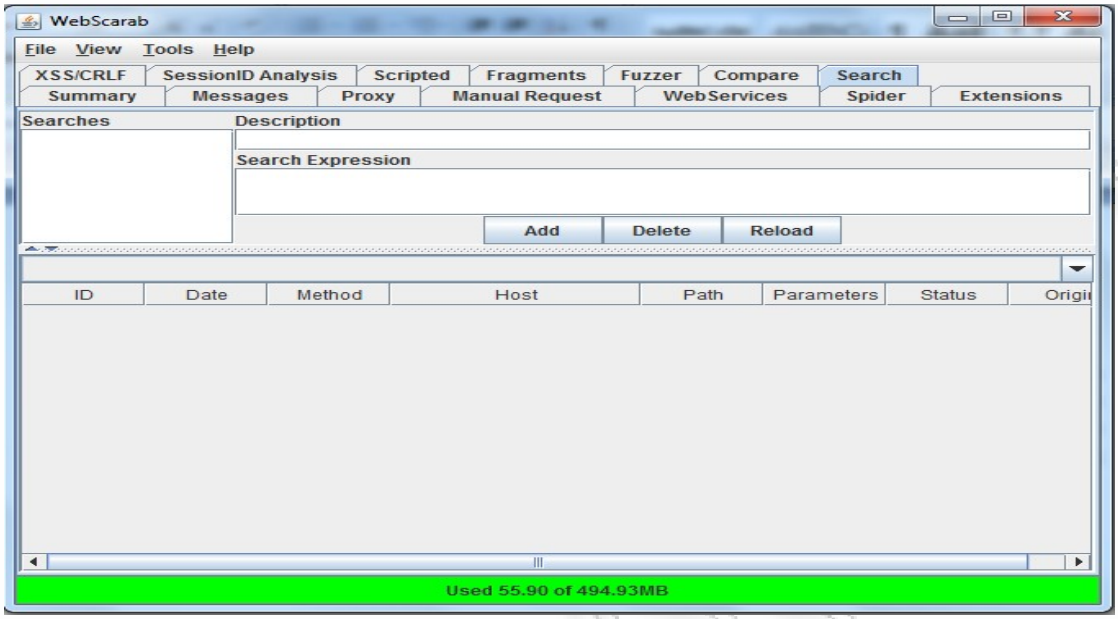
Search plug-in

Το search plug-in μας επιτρέπει να εκτελέσουμε αυθαίρετα beanshell scripts, για να προσδιορίσουμε "ενδιαφέρουσες" συνομιλίες. Μας παρέχεται το αίτημα, η απάντηση και η προέλευση της συνομιλίας και μπορούμε να χρησιμοποιήσουμε τις μεθόδους κλάσεων, για να επιστρέψουμε μια σωστή ή λανθασμένη τιμή. Μια σωστή τιμή δείχνει μια ενδιαφέρουσα συνομιλία, που πρέπει να εμφανισθεί, και μια λανθασμένη τιμή δείχνει ότι η συνομιλία δεν πρέπει να εμφανισθεί. Ένα παράδειγμα είναι κάπως έτσι:

response.getContent() != null && new
String(response.getContent()).matches("(?s).[Ee](rror|xception).*")*

Αυτό βεβαιώνει ότι η απάντηση έχει το περιεχόμενο (byte[ ]), πριν να ελέγξει, για να δει, εάν ένα String, που κατασκευάστηκε από εκείνο το περιεχόμενο, περιέχει κάποιο από τα: Error, error, Exception,

exception. Το (?s) καθοδηγεί τον αλγόριθμο Java Regex να εκτελέσει μια πολλαπλών γραμμών αντιστοιχία, π.χ. επιτρέπει την περίοδο ταιριάσματος ενός χαρακτήρα.



Εικόνα ***:*** **To Search plug-in.**

Για περισσότερες λεπτομέρειες περί των διαθέσιμων μεθόδων κλάσεων, μπορούμε να συμβουλευτούμε το συμπεριλαμβανόμενο στο installer build JavaDocs για org.owasp.webscarab.model. (Request|Response). Η προέλευση είναι απλά ένα String, που ταιριάζει το όνομα του plug-in.

Κεφάλαιο 3ο [Μετρήσεις και Ανάλυση Στοιχείων]

3.1Εισαγωγή

Στα πλαίσια της εργασίας αυτής αναλάβαμε να πραγματοποιήσουμε σε πραγματικές συνθήκες ένα πλήρη έλεγχο ασφάλειας σε έναν ενεργό ιστότοπο και συγκεκριμένα στον ιστότοπο του τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς με βάση το Testing Guide που προτείνει ο OWASP. Τα αποτελέσματα θα πρέπει να αντιπροσωπεύουν την πραγματική κατάσταση της ασφάλειας του ιστότοπου, ώστε να προσδιορισθεί η αποτελεσματικότητα της άμυνάς του σε μια ενδεχόμενη επίθεση. Θα πρέπει να σημειωθεί ότι αυτή η εργασία θα μπορούσε να είναι μια έρευνα του ίδιου του πανεπιστημίου στην απόφασή του να μετρήσει το επίπεδο της ασφάλειάς του, χωρίς να χρειάζεται να απευθύνεται σε εξειδικευμένους αλλά και απαγορευτικού κόστους οργανισμούς ασφάλειας. Μια κίνηση, η οποία είναι απόλυτα αντιπροσωπευτική της ανάγκης, που αντιμετωπίζουν σήμερα όλοι οι οργανισμοί και επιχειρήσεις που δραστηριοποιούνται στο Internet.

Απαραίτητες προϋποθέσεις για τον έλεγχο, ήταν ότι σε καμία περίπτωση δεν θα έπρεπε να παρεμποδισθεί η λειτουργία του ιστότοπου και οπωσδήποτε να μην προκληθεί οποιαδήποτε βλάβη μόνιμη ή προσωρινή σε κάποιο από τα συστατικά του. Επίσης, δεν προϋπήρχαν κανενός είδους πληροφορίες σχετικά με τη δομή και τα συστατικά του ιστότοπου. Γι' αυτό, χρησιμοποιείται η Black Box προσέγγιση της δοκιμής διείσδυσης για τον έλεγχο ασφάλειας του ιστότοπου του τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς.

Τι είναι η μεθοδολογία ελέγχου του OWASP;

Η δοκιμή διείσδυσης δε θα αποτελέσει ποτέ την συγκεκριμένη επιστήμη όπου μία πλήρης λίστα με όλα τα πιθανά θέματα που θα πρέπει να ελεγχθούν θα μπορεί να οριστεί. Πράγματι ο η δοκιμή διείσδυσης είναι μία κατάλληλη τεχνική μόνο για τον έλεγχο της ασφάλειας των διαδικτυακών εφαρμογών κάτω από ορισμένες συνθήκες. Ο στόχος είναι να συλλέξει όλες τις πιθανές τεχνικές ελέγχου, να τις εξηγήσει και να διατηρήσει τον οδηγό ενημερωμένο. Η μέθοδος της δοκιμής διείσδυσης των διαδικτυακών εφαρμογών του OWASP βασίζεται στην black-box προσέγγιση. Ο ελεγκτής δε γνωρίζει τίποτα ή έχει πολύ λίγες πληροφορίες αναφορικά με την εφαρμογή που θα ελεγχθεί.

<u>Το μοντέλο ελέγχου αποτελείται από:</u>

- Τον Ελεγκτή: Αυτός που πραγματοποιεί τις δραστηριότητες του έλεγχου.
- Τα εργαλεία και τη μεθοδολογία: Ο πυρήνας αυτού του έργου Οδηγού Ελέγχου.
- Εφαρμογή: Το «μαύρο κουτί» που θα ελέγξουμε.

<u>Η δοκιμή διαιρείται σε δύο φάσεις:</u>

- ***Παθητική κατάσταση:*** Κατά την παθητική κατάσταση, ο ελεγκτής προσπαθεί να καταλάβει τη λογική της εφαρμογής και «παίζει» με την εφαρμογή. Εργαλεία μπορούν να χρησιμοποιηθούν για τη συλλογή πληροφοριών, για παράδειγμα, ένας HTTP proxy να εξυπηρετήσει όλα τα HTTP αιτήματα και απαντήσεις. Στο τέλος αυτής της φάσης, ο ελεγκτής θα πρέπει να καταλαβαίνει όλα τα σημεία πρόσβασης (πύλες) της εφαρμογής (πχ, HTTP επικεφαλίδες, παράμετροι και cookies). Το κομμάτι της Συλλογής των Πληροφοριών (Information Gathering) εξηγεί πώς πρέπει να προετοιμαστεί ένας έλεγχος παθητικής κατάστασης (passive mode). Για παράδειγμα ο ελεγκτής θα μπορούσε να βρει το ακόλουθο:
  https://www.example.com/login/Authentic_Form.html
  Αυτό μπορεί να υποδεικνύει μία φόρμα αυθεντικοποίησης κατά την οποία η εφαρμογή ζητά ένα όνομα χρήστη(username) και ένα συνθηματικό (password). Οι παράμετροι που ακολουθούν αντιπροσωπεύουν δύο σημεία πρόσβασης (πύλες) στην εφαρμογή:
  http://www.example.com/Appx.jsp?a=1&b=1
  Σε αυτήν την περίπτωση, η εφαρμογή δείχνει δύο πύλες (παράμετροι a και b). Όλες οι πύλες που τυχόν θα βρεθούν σε αυτή την φάση αποτελούν ένα σημείο ελέγχου. Ένα λογιστικό φύλλο με το δέντρο των directories της εφαρμογής και όλα τα σημεία πρόσβασης θα ήταν χρήσιμα για την δεύτερη φάση.

- ***Ενεργητική κατάσταση:*** Σε αυτή τη φάση, ο ελεγκτής ξεκινά να τους ελέγχους χρησιμοποιώντας τη μεθοδολογία που περιγράφεται παρακάτω:
  - Έλεγχος Διαμόρφωσης Διαχείρισης (Configuration Management Testing)
  - Έλεγχος της Επιχειρηματικής Λογικής (Business Logic Testing)
  - Έλεγχος Αυθεντικοποίησης (Authentication Testing)
  - Έλεγχος Εξουσιοδότησης (Authorization Testing)
  - Έλεγχος Διαχείρισης Συνόδου (Session Management Testing)
  - Έλεγχος Επικύρωσης Δεδομένων (Data Validation Testing)
  - Έλεγχος Άρνησης Παροχής Υπηρεσίας (Denial of Service Testing)
  - Έλεγχος Διαδικτυακών Υπηρεσιών (Web Services Testing)
  - Δοκιμές Ajax

Παρακάτω δίνεται ο πίνακας με βάση την κατηγοριοποίηση που κάνει ο οργανισμός OWASP και αφορά τους ελέγχους που πρέπει να πραγματοποιούνται με τη μέθοδο της δοκιμής διείσδυσης (penetration testing) από τους ελεγκτές ασφαλείας εφαρμογών ιστού. Ο πίνακας αυτός προκύπτει από την έκδοση OWASP Testing Guide 3.0.

Πίνακας : **Λίστα κατηγοριοποίησης ελέγχων του OWASP, με τη μέθοδο του Penetration testing.**

Category	Reference Number	Test Name	Vulnerability

Information Gathering	OWASP-IG-001	Spiders, Robots and Crawlers	N.A.
	OWASP-IG-002	Search Engine Discovery/Reconnaissance	N.A.
	OWASP-IG-003	Identify application entry points	N.A.
	OWASP-IG-004	Testing for Web Application Fingerprint	N.A.
	OWASP-IG-005.	Application Discovery	N.A
	OWASP-IG-006	Analysis of Error Codes	Information Disclosure
Configuration Management Testing	OWASP-CM-001	SSL/TLS Testing (SSL Version, Algorithms, Key length, Digital Cert. Validity)	SSL Weakness
	OWASP-CM-002	DB Listener Testing	DB Listener weak
	OWASP-CM-003	Infrastructure Configuration Management Testing	Infrastructure Configuration management weakness
	OWASP-CM-004	Application Configuration Management Testing	Application Configuration management weakness
	OWASP-CM-005	Testing for File Extensions Handling	File extensions handling
	OWASP-CM-006	Old, backup and unreferenced files	Old, backup and unreferenced files
	OWASP-CM-007	Infrastructure and Application Admin Interfaces	Access to Admin interfaces
	OWASP-CM-008	Testing for HTTP Methods and XST	HTTP Methods enabled, XST permitted, HTTP Verb
Authentication Testing	OWASP-AT-001	Credentials transport over an encrypted channel	Credentials transport over an encrypted channel
	OWASP-AT-002	Testing for user enumeration	User enumeration
	OWASP-AT-003	Testing for Guessable (Dictionary) User Account	Guessable user account
	OWASP-AT-004	Brute Force Testing	Credentials Brute forcing
	OWASP-AT-005	Testing for bypassing authentication schema	Bypassing authentication schema

	OWASP-AT-006	Testing for vulnerable remember password and pwd reset	Vulnerable remember password, weak pwd reset
	OWASP-AT-007	Testing for Logout and Browser Cache Management	Logout function not properly implemented, browser cache weakness
	OWASP-AT-008	Testing for CAPTCHA Weak	Captcha implementation
	OWASP-AT-009	Testing Multiple Factors Authentication	Weak Multiple Facto-Authentication
	OWASP-AT-010	Testing for Race Conditions	Race Conditions vulnerability
Session Management	OWASP-SM-001	Testing for Session Management Schema	Bypassing Session Management Schema, Weak Session Token
	OWASP-SM-002	Testing for Cookies attributes	Cookies are set not 'HTTP Only', 'Secure', and no time validity
	OWASP-SM-003	Testing for Session Fixation	Session Fixation
	OWASP-SM-004	Testing for Exposed Session Variables	Exposed sensitive session variables
	OWASP-SM-005	Testing for CSRF	CSRF
Authorization Testing	OWASP-AZ-001	Testing for Path Traversal	Path Traversal
	OWASP-AZ-002	Testing for bypassing authorization schema	Bypassing authorization schema
	OWASP-AZ-003	Testing for Privilege Escalation	Privilege Escalation
Business logic testing	OWASP-BL-001	Testing for business logic	Bypassable business logic
Data Validation Testing	OWASP-DV-001	Testing for Reflected Cross Site Scripting	Reflected XSS
	OWASP-DV-002	Testing for Stored Cross Site Scripting	Stored XSS
	OWASP-DV-003	Testing for DOM based Cross Site Scripting	DOM XSS
	OWASP-DV-004	Testing for Cross Site Flashing	Cross Site Flashing
	OWASP-DV-005	SQL Injection	SQL Injection
	OWASP-DV-006	LDAP Injection	LDAP Injection
	OWASP-DV-007	ORM Injection	ORM Injection
	OWASP-DV-008	XML Injection	XML Injection
	OWASP-DV-009	SSI Injection	SSI Injection
	OWASP-DV-010	XPath Injection	XPath Injection

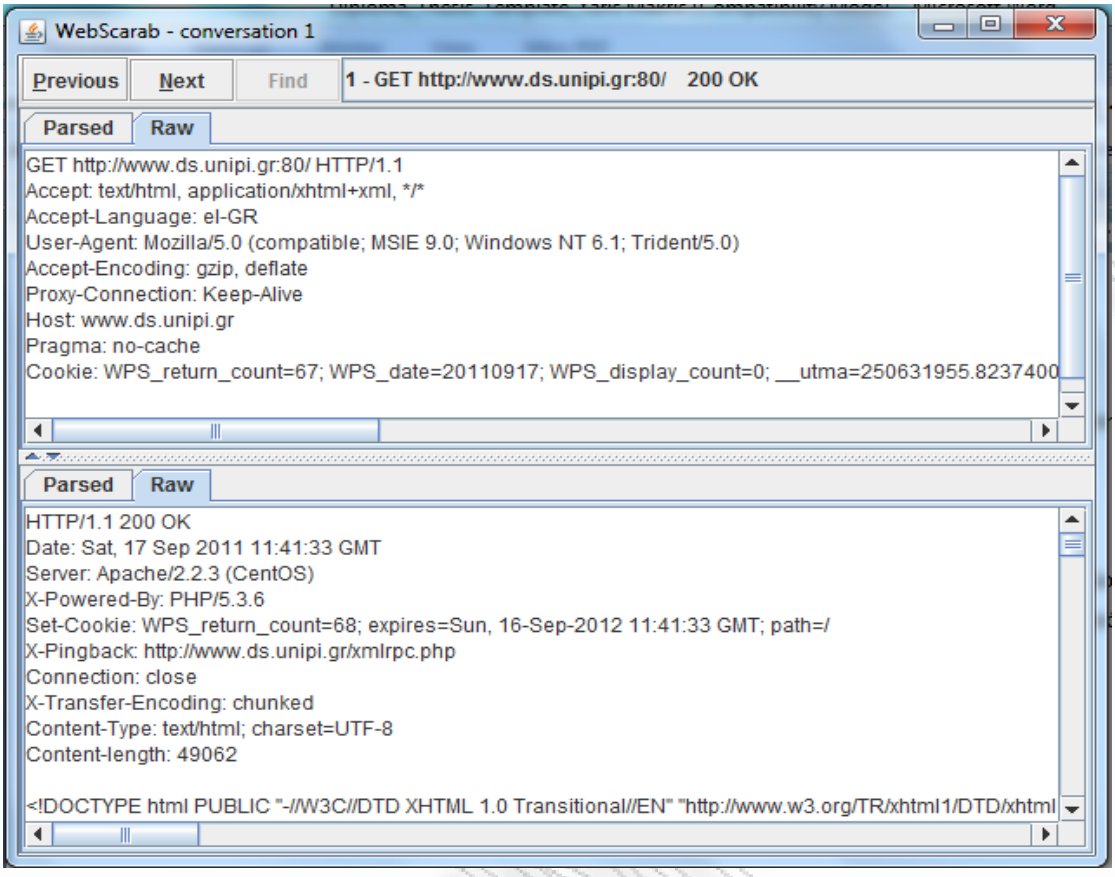
	OWASP-DV-011	IMAP/SMTP Injection	IMAP/SMTP Injection
	OWASP-DV-012	Code Injection	Code Injection
	OWASP-DV-013	OS Commanding	OS Commanding
	OWASP-DV-014	Buffer overflow	Buffer overflow
	OWASP-DV-015	Incubated vulnerability Testing	Incubated vulnerability
	OWASP-DV-016	Testing for HTTP Splitting/Smuggling	HTTP Splitting, Smuggling
Denial of Service Testing	OWASP-DS-001	Testing for SQL Wildcard Attacks	SQL Wildcard vulnerability
	OWASP-DS-002	Locking Customer Accounts	Locking Customer Accounts
	OWASP-DS-003	Buffer Overflows	Buffer Overflows
	OWASP-DS-004	User Specified Object Allocation	User Specified Object Allocation
	OWASP-DS-005	User Input as a Loop Counter	User Input as a Loop Counter
	OWASP-DS-006	Writing User Provided Data to Disk	Writing User Provided Data to Disk
	OWASP-DS-007	Failure to Release Resources	Failure to Release Resources
	OWASP-DS-008	Storing too Much Data in Session	Storing too Much Data in Session
Web Services Testing	OWASP-WS-001	WS Information Gathering	N.A.
	OWASP-WS-002	Testing WSDL	WSDL Weakness
	OWASP-WS-003	XML Structural Testing	Weak XML Structure
	OWASP-WS-004	XML content-level Testing	XML content-level
	OWASP-WS-005	HTTP GET parameters/REST Testing	WS HTTP GET parameters/REST
	OWASP-WS-006	Naughty SOAP attachments	WS Naughty SOAP attachments
	OWASP-WS-007	Replay Testing	WS Replay Testing
AJAX Testing	OWASP-AJ-001	AJAX Vulnerabilities	N.A
	OWASP-AJ-002	AJAX Testing	AJAX weakness

Στην παρούσα μεταπτυχιακή διπλωματική εργασία μελετήσαμε τις κατηγορίες ελέγχων που αφορούν το Information Gathering, Configuration Management Testing, Authentication Testing, Authorization και Denial of Service. Έγινε προσπάθεια να πραγματοποιηθούν οι περισσότεροι από τους παραπάνω ελέγχους και μάλιστα, για λόγους πληρότητας της παρουσίασης, σε κάθε στάδιο της μεθόδου έγινε προσπάθεια ο έλεγχος να γίνεται και με τις δύο τεχνικές ελέγχου, τη χειροκίνητη και την

αυτοματοποιημένη, όσο αυτό είναι εφικτό λόγω της περιορισμένης τεχνογνωσίας και εμπειρίας σε διαδικασίες ελέγχου ασφάλειας.

3.2 Συλλογή κι ανάλυση Πληροφοριών (Information Gathering)

Η πρώτη φάση στην εκτίμηση του επιπέδου ασφάλειας επικεντρώνεται στην συλλογή όσο το δυνατών περισσότερων πληροφοριών για την εφαρμογή(ιστότοπο)-στόχο και στη συνέχεια στην ανάλυσή τους. Έτσι, ο πηγαίος κώδικας των ιστοσελίδων, εάν είναι διαθέσιμος, μπορεί να δώσει ενδείξεις ως προς το περιβάλλον υπόστρωμα της εφαρμογής. Πρέπει να σημειωθεί βέβαια, ότι τώρα πλέον αυτό το φαινόμενο είναι αρκετά σπάνιο, δηλαδή να αφήνονται τέτοιες πληροφορίες στον πηγαίο κώδικα των HTML σελίδων. Η συλλογή των πληροφοριών, λοιπόν, μπορεί να πραγματοποιηθεί με διάφορους τρόπους. Χρησιμοποιώντας εργαλεία (μηχανές αναζήτησης), scanners, στέλνοντας HTTP αιτήματα, ή ειδικά διαμορφωμένα αιτήματα είναι δυνατόν να αναγκάσει την εφαρμογή να έχει κάποια διαρροή πληροφοριών, πχ να αποκαλύψει κάποια μηνύματα λάθους ή να αποκαλύψει τις εκδόσεις και τις τεχνολογίες που χρησιμοποιούνται. . Η συλλογή των πληροφοριών είναι ένα απαραίτητο βήμα για μία δοκιμή διείσδυσης (penetration testing). Πραγματικά, όπως φαίνεται στην εικόνα 29 που αφορά τον πηγαίο κώδικα του ιστότοπου του πανεπιστημίου, δίνονται κάποιες πληροφορίες σχετικά με τα προγραμματιστικά εργαλεία, που έχουν χρησιμοποιηθεί από τους προγραμματιστές της εφαρμογής. Από τα στοιχεία των διαφόρων συνδέσμων του πηγαίου κώδικα της HTML σελίδας συμπεραίνουμε ότι ο Web server της εφαρμογής είναι ένας Apache της έκδοσης 2.2.3 που λειτουργεί σε λειτουργεί σε λειτουργικό σύστημα Linux και συγκεκριμένα «τρέχει» λειτουργικό CentOS και η εφαρμογή έχει αναπτυχθεί σε PHP της έκδοσης 5.3.6. Γενικά θεωρείται αδυναμία της εφαρμογής να αναφέρεται η έκδοσή της και το είδος του λειτουργικού που χρησιμοποιεί ο server καθώς και η γλώσσα με βάση την οποία αναπτύχθηκε γιατί αποτελούν στοιχεία που μπορεί εύκολα να εκμεταλλευτεί κάποιος κακόβουλος χρήστης στοχεύοντας σε γνωστές ευπάθειες των συγκεκριμένων εκδόσεων.



Εικόνα ***:*** **Παθητικός προσδιορισμός πληροφοριών με την εξέταση του πηγαίου κώδικα του ιστότοπου.**

3.2.1 Spiders, Robots and Crawlers

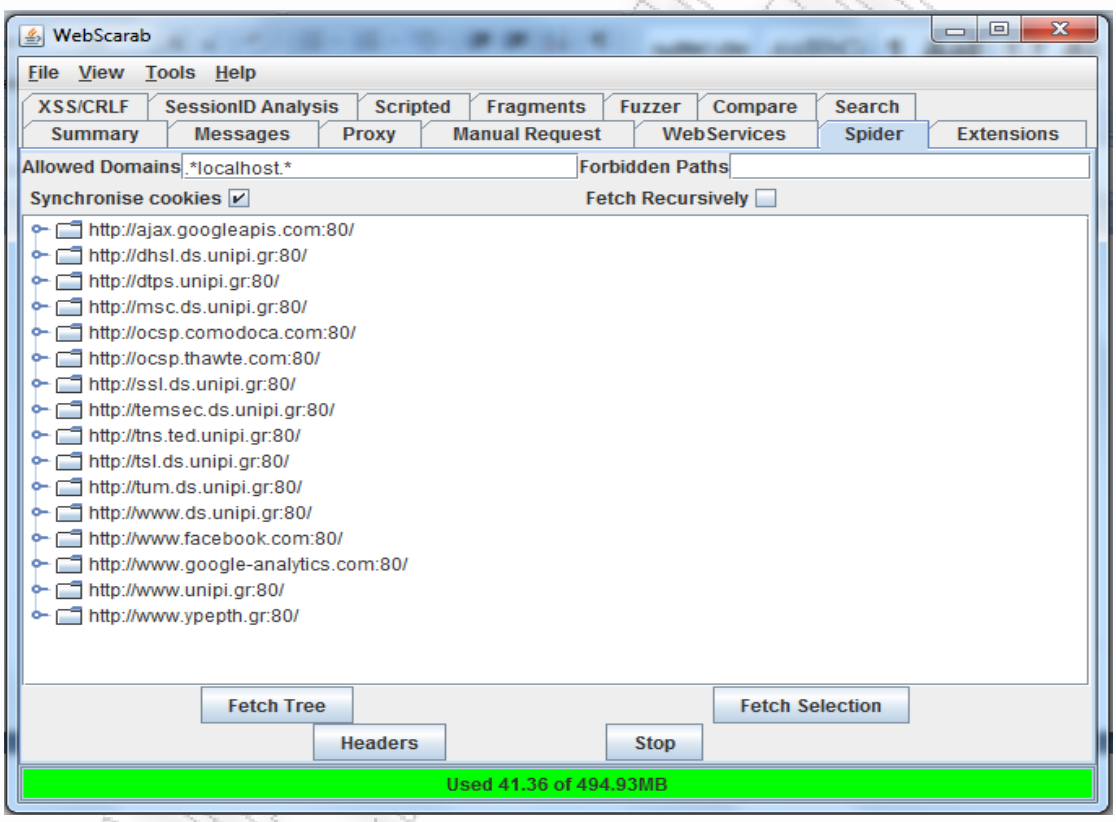
Αυτή η φάση της συλλογής πληροφοριών η διαδικασία περιλαμβάνει περιήγηση και καταγραφή των πόρων που σχετίζονται με την υπό εξέταση εφαρμογή. Τα διαδικτυακά spiders/robots/crawlers ανακτούν μια ιστοσελίδα και στη συνέχεια διαπερνούν αναδρομικά τους συνδέσμους(links) προκειμένου να ανακτήσουν περισσότερο διαδικτυακό περιεχόμενο. Για παράδειγμα το αρχείο robots.txt από το http://www.ds.unipi.gr/robots.gr το οποίο το πήραμε στις 19 Σεπτεμβρίου του 2011 παρατίθεται παρακάτω:

User-agent: *
Disallow:

Η οδηγία User-Agent αναφέρεται στο προκείμενο διαδικτυακό spider/robot/crawler. Για παράδειγμα το User-Agent: Googlebot αναφέρεται στον Crawler GoogleBot ενώ το User-Agent: * στο παράδειγμα που δώσαμε από την ιστοσελίδα του Τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς παραπάνω αναφέρεται σε όλα τα διαδικτυακά spiders/robots/crawlers. Η οδηγία Disallow ορίζει ποιοι πόροι είναι απαγορευμένοι (μη-προσπελάσιμοι) από spiders/robots/crawlers. Τα διαδικτυακά

spiders/robots/crawlers μπορούν εσκεμμένα να αγνοήσουν την οδηγία Disallow που ορίζεται στο αρχείο robots.txt. Ως εκ τούτου το robots.txt δε θα πρέπει να θεωρείται ένας μηχανισμός για την επιβολή περιορισμών σχετικά με το διαδικτυακό περιεχόμενο γίνεται προσπελάσιμο, αποθηκεύεται ή αναδημοσιεύεται από τρίτους.

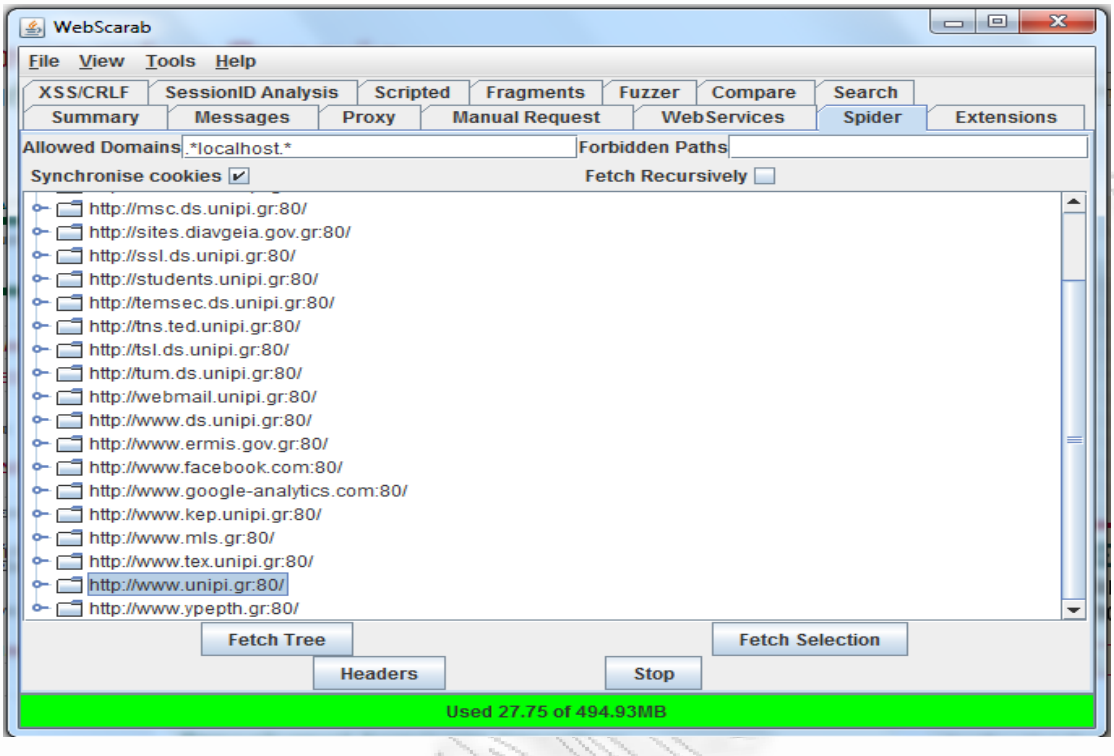
Όπως αναφέραμε και στο θεωρητικό κομμάτι, ο στόχος μας είναι να δημιουργήσουμε έναν χάρτη της εφαρμογής με όλα τα σημεία πρόσβασης (gates) σε αυτήν. Αυτό θα είναι πολύ χρήσιμο για τη δεύτερη φάση της δοκιμής διείσδυσης. Μπαίνοντας, λοιπόν, στην αρχική σελίδα του ιστότοπου του πανεπιστημίου και επιλέγοντας το Spider plug-in του WebScarab, βλέπουμε ένα παράθυρο σαν και αυτό της εικόνας 30.



Εικόνα : **Οι αναγνωρισμένες από το Spider plug-in συνδέσεις (links) στον ιστότοπο του Τμήματος Ψηφιακών Συστημάτων (www.ds.unipi.gr).**

Εδώ μπορούμε να δούμε όλες τις συνδέσεις, που υπάρχουν στην αρχική σελίδα. Αν επιλέξουμε τη σύνδεση http://www.unipi.gr:80 και την επιλογή “Fetch Selection”, διαπιστώνουμε ότι αναγνωρίζονται και άλλες ενδιαφέρουσες συνδέσεις, τις οποίες θέσει στην σειρά για να ανακτηθούν, όποτε εμείς θελήσουμε, όπως φαίνεται στην εικόνα 31. Αν επιλέξουμε "Fetch Tree”, τότε αναφέρονται κατά

αλφαβητική σειρά όλες οι αναγνωρισμένες συνδέσεις στην αρχική σύνδεση, που επιλέξαμε, και αυτόματα τις θέτει στην σειρά για ανάκτηση.



Εικόνα **: Οι αναγνωρισμένες από το Spider plug-in συνδέσεις (links) στον ιστότοπο του Τμήματος Ψηφιακών Συστημάτων κατά αλφαβητική σειρά (www.ds.unipi.gr).**

Συμπερασματικά, όμως, μπορούμε να πούμε πως η ιστοσελίδα που εξετάζουμε με βάση τις παραμέτρους που είδαμε πως ισχύουν στο αρχείο robots.txt καθώς επίσης και μετά το "spidering" που πραγματοποιήσαμε μέσω του Web Scarab είναι «ανοιχτή» σε spiders/robots/crawlers χωρίς να ορίζει προστατευμένους πόρους.

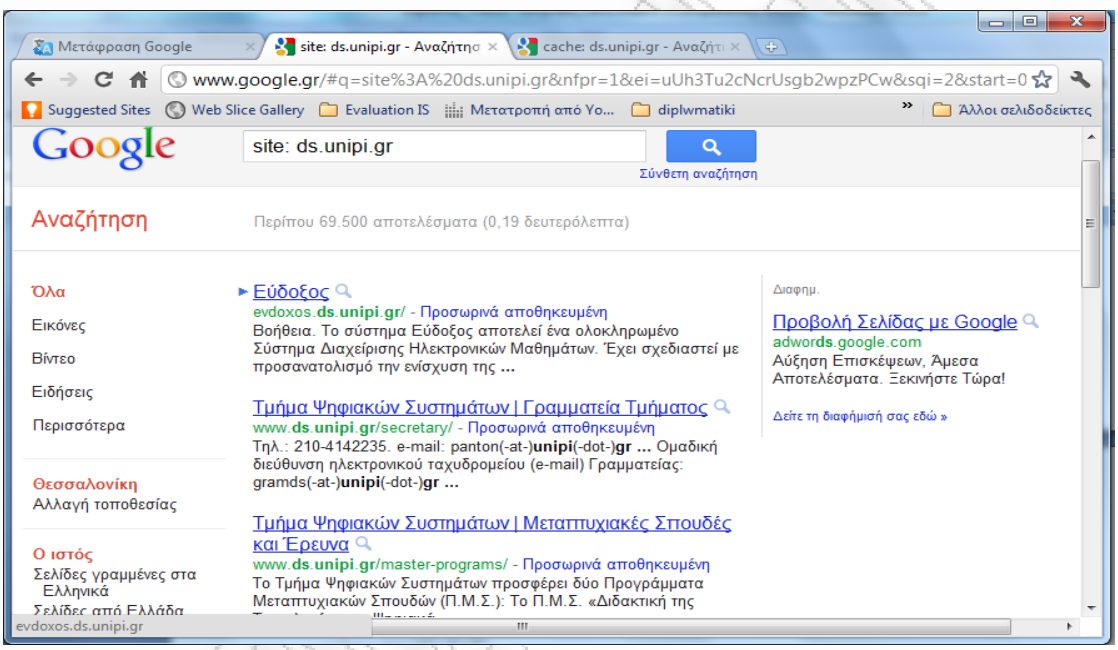
3.2.2 Search Engine Discovery/Reconnaissance

Αυτή η ενότητα περιγράφει πώς να κάνουμε αναζήτηση στο ευρετήριο του Google και ανασύρουμε το σχετικό περιεχόμενο του Παγκόσμιου Ιστού από τη μνήμη cache του Google Μόλις το GoogleBot έχει ολοκληρώσει το crawling, ξεκινά να κατατάσσει σε ευρετήριο την ιστοσελίδα με βάση τις ετικέτες και τις συναφείς ιδιότητες, όπως το <TITLE>, προκειμένου να επιστρέψει τα συναφή αποτελέσματα αναζήτησης. Αν το αρχείο robots.txt δεν έχει ενημερωθεί κατά τη διάρκεια ζωής του δικτυακού τόπου, τότε είναι πιθανό το περιεχόμενο των ιστοσελίδων να μην περιληφθεί στα αποτελέσματα αναζήτησης της Google έτσι ώστε να επιστραφεί. Ως εκ τούτου, πρέπει να αφαιρεθεί από τη μνήμη cache του Google.

BLACK BOX TESTING

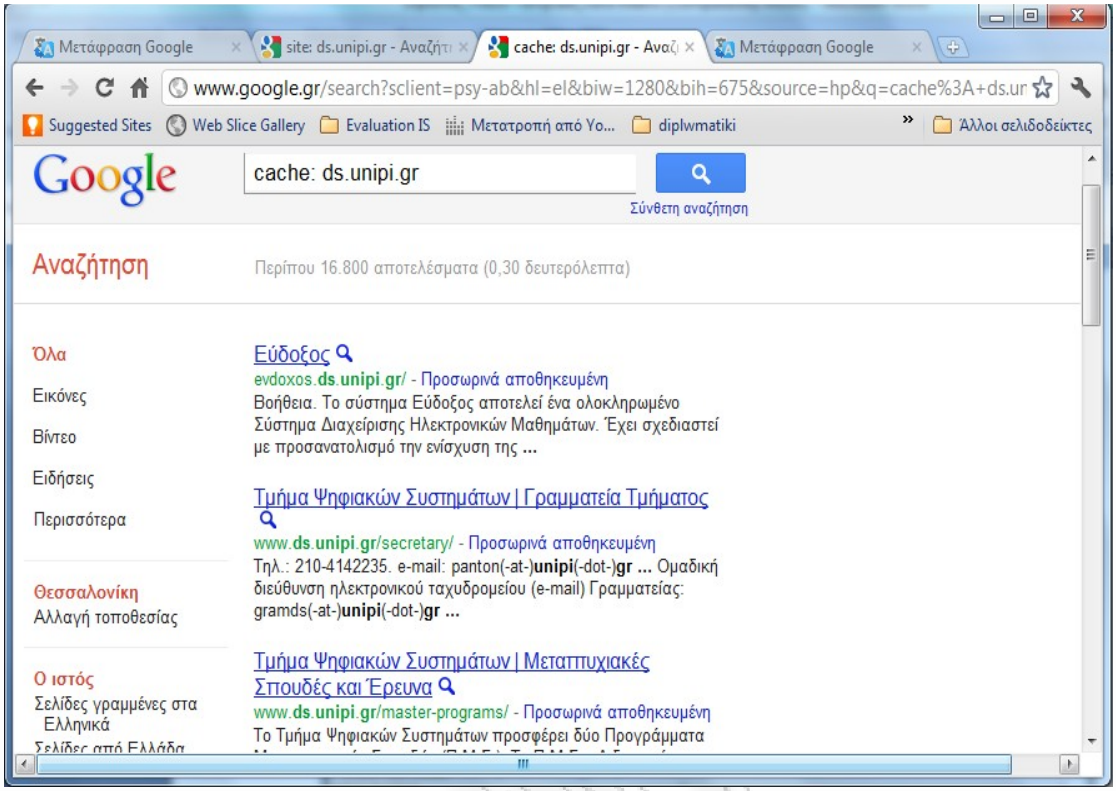
Χρησιμοποιώντας τον προηγμένο διαχειριστή αναζήτησης “site:” είναι δυνατόν να περιορίσουμε τα αποτελέσματα της αναζήτησης σε έναν συγκεκριμένο τομέα. Η Google παρέχει τον προηγμένο χειριστή αναζήτησης “cache:” αλλά αυτό ισοδυναμεί με το «κλικ» στο “Cached” δίπλα σε κάθε αποτέλεσμα της αναζήτησης μέσω της Google. Ως εκ τούτου είναι προτιμότερη η χρήση του προηγμένο διαχειριστή αναζήτησης “site:” και έπειτα το «κλικ» στο “Cached”. Η Google SOAP Search API υποστηρίζει την doGetCachedPage και τα σχετικά μηνύματα SOAP doGetCachedPageResponse [3], για να συνδράμει με την ανάκτηση cachedσελίδων. Η εφαρμογή αυτή βρίσκεται υπό ανάπτυξη από το OWASP "Google Hacking" Project.

Για να βρούμε τι περιέχει το διαδίκτυο για τον ιστότοπο ds.unipi.gr στο ευρετήριο του Google Cache θα δώσουμε το παρακάτω Google αίτημα αναζήτησης: ***site: ds.unipi.gr***



Εικόνα : **Τα αποτελέσματα της αναζήτησης πληροφοριών στο διαδίκτυο (googling) για το ds.unipi.gr.**

Για να εμφανίσουμε το index.html του ds.unipi.gr που αποθηκεύεται προσωρινά από την Google πρέπει να κάνουμε το παρακάτω αίτημα αναζήτησης: ***cache: ds.unipi.gr***



Εικόνα **: Τα αποτελέσματα της αναζήτησης πληροφοριών στο διαδίκτυο (googling) για το ds.unipi.gr με τη χρήση του διαχειριστή αναζήτησης "cache".**

Με βάση λοιπόν τα αποτελέσματα του googling και που κάναμε και στις δύο παραπάνω περιπτώσεις διαπιστώνουμε πως η «σημαντική» πληροφορία που παίρνουμε για τον ιστότοπο μας είναι σχετικά με το ποιές εφαρμογές που υποστηρίζονται στον server ο οποίος φιλοξενεί τον υπό εξέταση ιστότοπο όπως πχ κάποια site εργαστηρίων.

3.2.3 Identify Application Entry Points (Προσδιορισμός των Σημείων Εισόδου της Εφαρμογής)

Η απαρίθμηση της εφαρμογής καθώς και το μέγεθος της περιοχής που καλύπτει η επίθεση είναι ένας πρόδρομος «κλειδί» προτού οποιοσδήποτε ενδελεχής έλεγχος ξεκινήσει, καθώς επιτρέπει σε αυτόν που κάνει τον έλεγχο να εντοπίσει πιθανές περιοχές αδυναμίας. Η ενότητα αυτή έχει ως στόχο να βοηθήσει στον εντοπισμό και την χαρτογράφηση των περιοχών εντός της εφαρμογής που θα πρέπει να διερευνηθούν αμέσως μόλις η αρίθμηση και η χαρτογράφηση έχουν ολοκληρωθεί.

Πριν ξεκινήσει οποιοσδήποτε έλεγχος, θα πρέπει πάντα να έχουμε μία καλή κατανόηση της εφαρμογής και με ποιον τρόπο ο χρήστης/browser επικοινωνεί με αυτή. Καθώς πλοηγούμαστε μέσα στην εφαρμογή θα πρέπει να δώσουμε ιδιαίτερη σημασία σε όλα τα HTTP αιτήματα (GET και POST μέθοδοι, επίσης γνωστές ως «ρήματα»), καθώς και σε κάθε παράμετρο και φόρμα που «περνιούνται» στην εφαρμογή. Επίσης πρέπει να δώσουμε προσοχή όταν χρησιμοποιούνται GET αιτήματα και επίσης όταν

αιτήματα POST χρησιμοποιούνται προκειμένου να εισαχθούν παράμετροι στην εφαρμογή. Είναι πολύ κοινό να χρησιμοποιούνται τα αιτήματα GET, όμως όταν εισάγεται ευαίσθητη πληροφορία είναι συχνό φαινόμενο να γίνεται αυτό μέσα στο «σώμα» ενός αιτήματος POST. Πρέπει να σημειώσουμε πως για να δούμε τις παραμέτρους που στέλνονται σε ένα POST αίτημα θα χρειαστεί να χρησιμοποιήσουμε έναν ενδιάμεσο proxy, όπως για παράδειγμα τον WebScarab του OWASP που χρησιμοποιούμε στην περίπτωσή μας ή ένα plug-in για τον περιηγητή browser.

Σε κάθε αίτημα POST, επίσης πρέπει να κάνουμε μία ειδική σημείωση για οποιοδήποτε κρυφό πεδίο της οποιασδήποτε φόρμας εισάγεται στην εφαρμογή, τα οποία συνήθως περιέχουν ευαίσθητη πληροφορία όπως πληροφορίες κατάστασης, την ποσότητα των στοιχείων, την τιμή των στοιχείων, που ο δημιουργός της εφαρμογής ποτέ δεν σκόπευε να δούμε ή να αλλάξουμε. Για την εμπειρία του συγγραφέα θα ήταν πολύ χρήσιμο να χρησιμοποιήσουμε έναν ενδιάμεσο proxy και ένα λογιστικό φύλλο για αυτό το στάδιο των δοκιμών. Ο proxy θα κρατά ίχνος από οποιοδήποτε αίτημα και θα αντίδραση από εμάς και την εφαρμογή όσο διαπερνούμε την εφαρμογή. Επιπλέον σε αυτό το σημείο, οι ελεγκτές συνήθως παγιδεύουν κάθε αίτημα και αντίδραση έτσι ώστε να μπορούν να δουν με ακρίβεια οποιαδήποτε επικεφαλίδα(header), παράμετρο κλπ που εισάγεται στην εφαρμογή καθώς και τι επιστρέφεται. Αυτό μπορεί να είναι αρκετά κουραστικό μερικές φορές, ιδιαίτερα στις μεγάλες διαδραστικές ιστοσελίδες(σκεφτείτε μία τραπεζική εφαρμογή).

Ωστόσο, η εμπειρία θα μας διδάξει τι πρέπει να αναζητήσουμε και που να στοχεύουμε, ως εκ τούτου, η φάση αυτή μπορεί να μειωθεί σημαντικά. Καθώς πλοηγούμαστε στην εφαρμογή, πρέπει να σημειώσουμε οποιαδήποτε ενδιαφέρουσα παράμετρο στο URL, τις προσαρμοσμένες επικεφαλίδες ή στο σώμα των αιτημάτων/αποκρίσεων και να τις «σώσουμε» στο λογιστικό μας φύλλο. Το λογιστικό φύλλο θα πρέπει να περιλαμβάνει τη σελίδα που αιτηθήκαμε (ίσως είναι χρήσιμο να προσθέσουμε και τον αριθμό αιτήματος από τον proxy, για μελλοντική αναφορά), τις ενδιαφέρουσες παραμέτρους, τους τύπους των αιτημάτων (POST/GET), εάν η πρόσβαση είναι αυθεντικοποιημένη ή όχι, εάν χρησιμοποιείται πρωτόκολλο SSL, εάν είναι μέρος μιας διαδικασίας πολλών βημάτων και άλλες σχετικές σημειώσεις. Μόλις έχουμε κάθε περιοχή της εφαρμογής χαρτογραφημένη τότε μπορούμε να προχωρήσουμε με την εφαρμογή και να ελέγξουμε καθεμία από τις περιοχές που έχουμε εντοπίσει και να κρατήσουμε σημειώσεις για το τι δούλεψε και τι όχι. Παρακάτω θα προσδιορίσουμε πως μπορεί να ελεγχθεί κάθε μία από τις περιοχές που παρουσιάζουν ενδιαφέρον αλλά το κομμάτι αυτό πρέπει να γίνεται πριν αρχίσει οποιοσδήποτε πραγματικός έλεγχος.

Παρακάτω είναι κάποια σημεία ενδιαφέροντος για όλα τα αιτήματα και τις απαντήσεις. Μέσα στο τμήμα των αιτημάτων προσέχουμε τις GET και POST μεθόδους όπως αυτές εμφανίζονται στην πλειοψηφία των αιτημάτων. Να σημειώσουμε πως άλλες μέθοδοι όπως η PUT και DELETE, μπορούν επίσης να χρησιμοποιηθούν. Συχνά, αυτά τα πιο σπάνια αιτήματα, εάν επιτραπούν μπορούν να προκαλέσουν ευπάθειες. Υπάρχει ένα ειδικό κομμάτι του κεφαλαίου που είναι αφιερωμένο στις δοκιμές των HTTP μεθόδων.

<u>Αιτήματα:</u>

- Αναγνωρίστε που χρησιμοποιούνται αιτήματα τύπου GET και που τύπου POST.
- Αναγνωρίστε όλες τις παραμέτρους που χρησιμοποιούνται σε ένα αίτημα POST (βρίσκονται στο σώμα του αιτήματος).
- Εντός του αιτήματος POST, δώστε ιδιαίτερη προσοχή σε τυχόν κρυμμένες παραμέτρους. Όταν ένα αίτημα POST έχει σταλεί, όλα τα πεδία της φόρμας (συμπεριλαμβανομένων και των κρυμμένων παραμέτρων) θα σταλούν μέσα στο σώμα του HTTP μηνύματος στην εφαρμογή. Αυτά τυπικά δεν είναι ορατά εκτός και αν χρησιμοποιούμε έναν proxy ή βλέπουμε τον πηγαίο κώδικα της HTML. Επιπλέον, η επόμενη σελίδα που βλέπετε αποτελεί δεδομένα και η πρόσβαση μπορεί να είναι διαφορετική ανάλογα με την τιμή των κρυμμένων παραμέτρων.
- Αναγνωρίστε όλες τις παραμέτρους που χρησιμοποιούνται σε ένα αίτημα GET (πχ URL) , ιδιαίτερα το query string (συνήθως μετά από ένα σύμβολο '?').
- Αναγνωρίστε όλες τις παραμέτρους ενός query string. Αυτά είναι στη μορφή ζευγαριού όπως το foo=bar. Επίσης σημειώστε πως πολλές παράμετροι μπορεί να είναι αίτημα string και να χωρίζονται με ένα '&','~' ή οποιοδήποτε άλλο ειδικό χαρακτήρα.
- Μια ειδική σημείωση που πρέπει να γίνει, όταν πρόκειται για τον εντοπισμό πολλαπλών παραμέτρων σε ένα string ή ένα αίτημα POST, είναι ότι ορισμένες ή όλες οι παράμετροι θα χρειαστούν για να εκτελέσουν τις επιθέσεις μας. Χρειάζεται να εντοπίσουμε όλες τις παραμέτρους (ακόμη και αν είναι κωδικοποιημένες ή κρυπτογραφημένες) και να αναγνωρίσουμε ποιες από αυτές είναι αντικείμενο επεξεργασίας της εφαρμογής. Αργότερα θα προσδιορίσουμε τον τρόπο με τον οποίο ελέγχουμε αυτές τις παραμέτρους, σε αυτό το σημείο απλώς πρέπει να είμαστε σίγουροι ότι αναγνωρίζουμε την καθεμία από αυτές.
- Επίσης πρέπει να δώσουμε σημασία σε τυχόν συμπληρωματικές ή προσαρμοσμένες επικεφαλίδες που τυπικά δεν είναι εμφανείς(όπως debug = False).

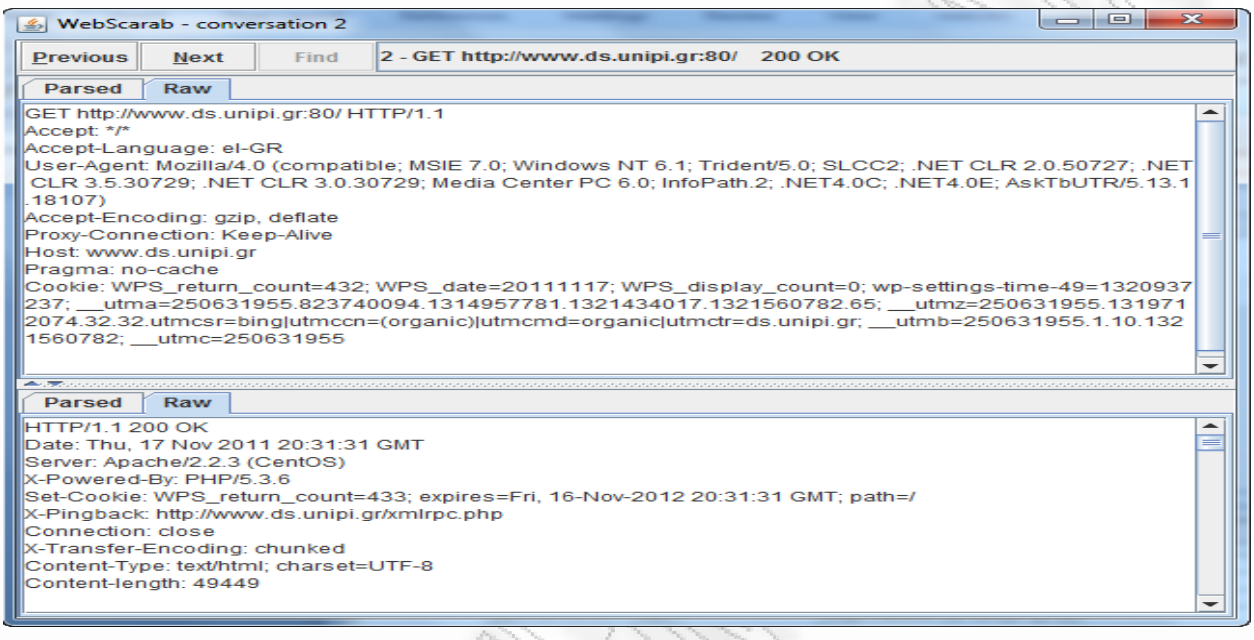
<u>Απαντήσεις:</u>

- Αναγνωρίστε που έχουν οριστεί καινούρια cookies (Set-Cookie header), τροποποιήθηκαν ή προστέθηκαν.
- Αναγνωρίστε που υπάρχουν τυχόν ανακατευθύνσεις(300 HTTP status code), status code 400, ιδιαίτερα 403 Forbidden και 500 interval server errors κατά τη διάρκεια κανονικών απαντήσεων(πχ μη-τροποποιημένα αιτήματα).
- Επίσης σημειώστε οποιαδήποτε ενδιαφέρουσα που χρησιμοποιείται. Για παράδειγμα,"Server: BIG-IP" υποδεικνύει ότι μία ιστοσελίδα έχει ισορροπημένο φορτίο. Έτσι αν μία τέτοια ιστοσελίδα και ένας server είναι παραμετροποιημένοι εσφαλμένα τότε ίσως χρειαστεί να στείλουμε πολλαπλά αιτήματα για πρόσβαση στον ευάλωτο server, ανάλογα με τον τύπο του ισορροπημένου φορτίου που χρησιμοποιείται.

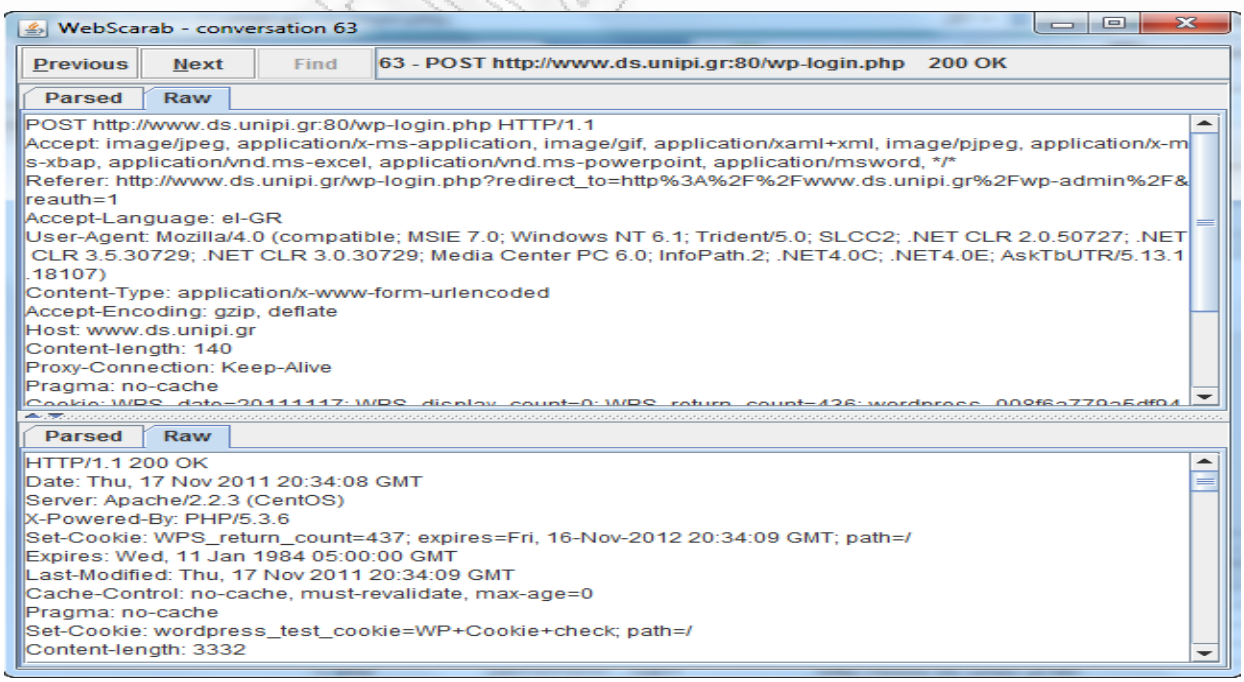
<u>BLACK BOX TESTING</u>

Εφαρμόζοντας τον έλεγχο αυτό στον ιστότοπο που εξετάζουμε διαπιστώνουμε τα παρακάτω αποτελέσματα:

- Σχετικά με τα αιτήματα(requests) που χρησιμοποιούνται κατά την περιήγησή μας στον ιστότοπο αυτά είναι ως επί το πλείστον αιτήματα τύπου GET και σε ελάχιστες περιπτώσεις τύπου POST. Αυτό έχει να κάνει και με τον τρόπο που έχει αναπτυχθεί η εφαρμογή μας. Επιπλέον στις εικόνες παρακάτω φαίνονται ενδεικτικά ένα αίτημα GET και ένα αίτημα POST:



Εικόνα : **Αίτημα GET από τον ιστότοπο www.ds.unipi.gr.**



Εικόνα *:* **Αίτημα POST της εφαρμογής www.ds.unipi.gr**

Όπως λοιπόν είναι φανερό μέσω του εργαλείου WebScarab, χρησιμοποιούνται κάποιες παράμετροι στα διάφορα αιτήματα τα οποία αποτελούν entry points για την εφαρμογή μας. Συγκεκριμένα βρήκαμε τις παρακάτω παραμέτρους να χρησιμοποιούνται κατά την περιήγησή μας στο www.ds.unipi.gr.

1. Παράμετρος: ?ver=2.3.3 σε περιπτώσεις απάντησης "Not Found" ή "Not Modified" από τον server.
2. Παράμετρος: ?ver=2.4.5 σε περιπτώσεις απάντησης "Not Found" ή "Not Modified" από τον server.
3. Παράμετρος: ?ver=2.3.3 σε περιπτώσεις απάντησης "Not Modified" από τον server.
4. Παράμετρος: ?ver=1.4.2 σε περιπτώσεις απάντησης "Not Modified" από τον server.
5. Παράμετρος: ?ver=3.0.5 σε περιπτώσεις απάντησης "Not Modified" από τον server.

Αναφορικά με τα αιτήματα POST που βρήκαμε στην δική μας περίπτωση έχουν να κάνουν με την προσπέλαση της διαχειριστικής σελίδας εισόδου της εφαρμογής που συγκεκριμένα είναι η www.ds.unipi.gr/wp-admin. Στην εικόνα που δείξαμε πιο πάνω με το POST αίτημα όπως εμφανίζεται στο WebScarab φαίνονται και οι τυχόν κρυμμένοι παράμετροι που χρησιμοποιούνται σε αυτά τα αιτήματα. Γενικά δεν χρησιμοποιούνται μέθοδοι POST κατά την περιήγησή μας στον ιστότοπο που εξετάζουμε.

- Σχετικά με τις απαντήσεις που λαμβάνουμε από τον server κατά την περιήγησή μας στον ιστότοπο προκύπτουν τα εξής αποτελέσματα:
    - Τα cookies που χρησιμοποιούνται δεν αλλάζουν τιμές. Δεν προστίθενται καινούρια και η μόνη τροποποίηση έχει να κάνει με τον counter που αναμενόμενα αυξάνει περιοδικά κατά την περιήγησή μας στον ιστότοπο.
    - Δεν παρατηρούμε περιπτώσεις στο status του server που να έχουμε τιμές κωδικών 300,400,403 Forbidden,500 Interval Error.

3.2.4 Testing for Web Application Fingerprint (Ελεγχος για Αποτύπωμα της Web Εφαρμογής)

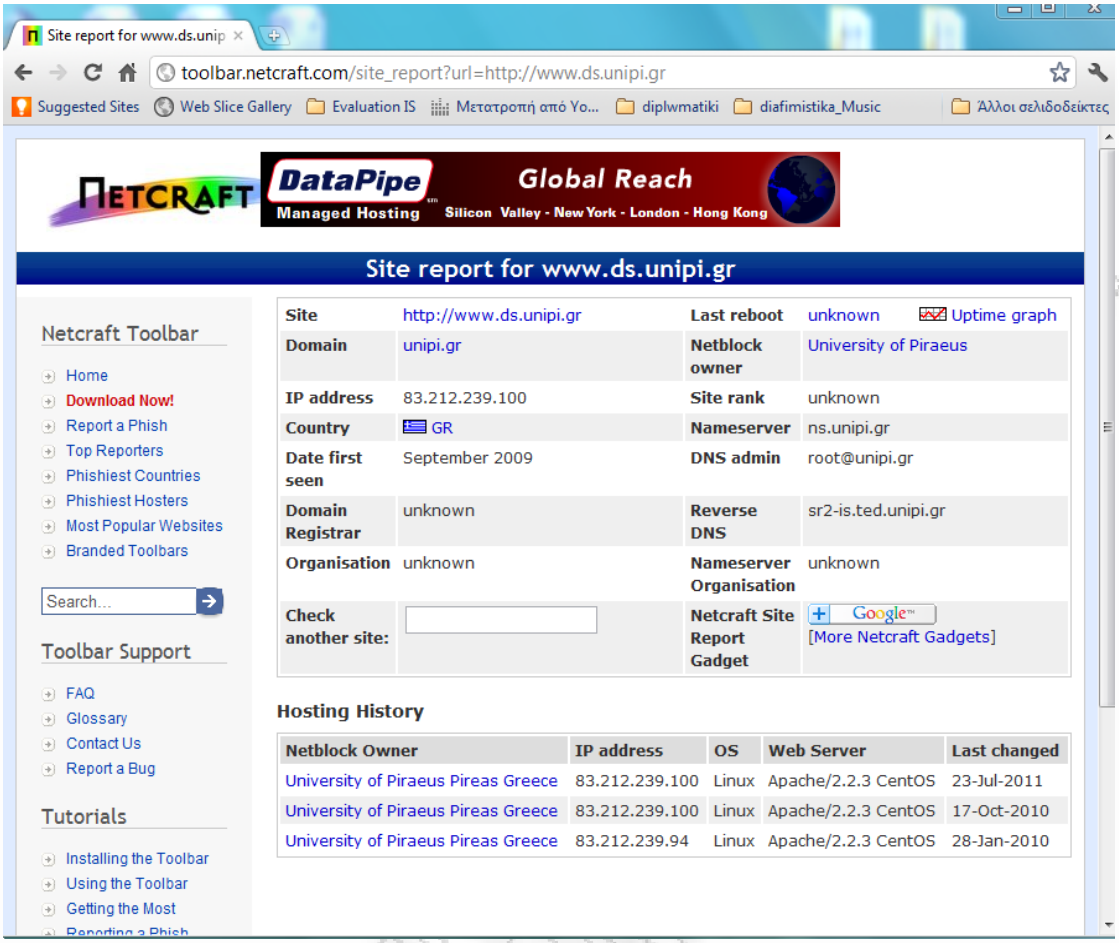
Η εύρεση και η ανάλυση του αποτυπώματος (fingerprint) της εφαρμογής, όπως είπαμε, έγκειται ουσιαστικά στη γνώση της έκδοσης (version) και του τύπου του server δικτύου. Μέσα από αυτές τις πληροφορίες θα μπορέσουμε ίσως να βρούμε στη συνέχεια γνωστές ευπάθειες των συστατικών της εφαρμογής.

Υπάρχουν διάφοροι προμηθευτές και εκδόσεις των web servers στην αγορά σήμερα. Γνωρίζοντας τον τύπο του web server στον οποίο κάνουμε τις δοκιμές βοηθά σημαντικά τη διαδικασία ελέγχου και θα αλλάξει επίσης τη διάρκεια της δοκιμής. Αυτές οι πληροφορίες μπορούν να προκύψουν στέλνοντας στον

web server εντολές κι αναλύοντας τις απαντήσεις μιας και κάθε διαφορετική έκδοση web server απαντά διαφορετικά στις εντολές αυτές. Γνωρίζοντας πως ο κάθε τύπος των web server απαντά στις εξειδικευμένες αυτές εντολές και κρατώντας τις πληροφορίες αυτές σε μία διαδικτυακή βάση δεδομένων δακτυλικών αποτυπωμάτων, ένας ελεγκτής διείσδυσης μπορεί να στείλει αυτές τις εντολές στον web server, να αναλύσει την απάντηση και να την συγκρίνει με τη βάση δεδομένων των γνωστών υπογραφών. Ας σημειώσουμε πως παίρνει συνήθως πολλές διαφορετικές εντολές για να προσδιορίσουμε με ακρίβεια τον web server, καθώς διαφορετικές εκδόσεις μπορεί να αντιδράσουν με τον ίδιο τρόπο στην ίδια εντολή. Σπάνια, ωστόσο, διαφορετικές εκδόσεις αντιδρούν το ίδιο σε όλες τις HTTP εντολές. Έτσι, στέλνοντας πολλές διαφορετικές εντολές θα αυξήσουμε την ακρίβεια των προβλέψεών μας.

BLACK BOX TESTING

Με τη βοήθεια του WebScarab επιλέγοντας τη συνομιλία 2 (εικόνα 35 σελ. 64), βλέπουμε διάφορες πληροφορίες, που αφορούν το λογισμικό του Web Server του τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς. Είναι ένας Apache Server και ξέρουμε ακριβώς την έκδοσή του, όχι όμως τις μεθόδους HTTP (GET, HEAD, POST, κλπ.), που επιτρέπονται, γνωρίζουμε ωστόσο το λειτουργικό χρησιμοποιούμενο σύστημα και το εργαλείο-πλατφόρμα με την οποία αναπτύχθηκε η εφαρμογή κάτι που συμπεράναμε εύκολα από την ονοματολογία των cookies(Wordpress) . Βλέπουμε επίσης, ότι το scripting περιβάλλον, που μπορεί να χρησιμοποιηθεί για δημιουργία και εκτέλεση προγραμμάτων στην πλευρά του server, είναι X-Beanshell. Δοκιμάσαμε στη συνέχεια και ένα άλλο εργαλείο, το οποίο είναι o-line και ονομάζεται Netcraft. Το Netcraft παρέχει πολλές πληροφορίες για τους web servers. Με αυτό το εργαλείο μπορούμε να αποκτήσουμε πληροφορίες για τα λειτουργικά συστήματα τα οποία ο συγκεκριμένος web sever έχει χρησιμοποιήσει κατά καιρούς, για το χρονικό διάστημα που βρίσκεται σε λειτουργία, (Server Uptime) το NetBlock Owner, την IP του sever το ιστορικό των αλλαγών λειτουργικών συστημάτων του web server και διάφορες άλλες χρήσιμες πληροφορίες που αφορούν χαρακτηριστικά του server. Σ’ αυτή την περίπτωση, σε αντίθεση με το WebScarab οι δοκιμές είχαν θετικά αποτελέσματα και οι πληροφορίες που λάβαμε για τον web server του Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς φαίνονται αναλυτικά στην παρακάτω εικόνα:



Εικόνα :**Αποτελέσματα από τη χρησιμοποίηση του on-line εργαλείου Netcraft για Fingerprinting στο web server του τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς.**

Από τα αποτελέσματα, λοιπόν, οι πληροφορίες που παίρνουμε είναι αυτές που αναμέναμε όμως διαπιστώνουμε πως η έκδοση του Apache Server όπως και το είδος του λειτουργικού όπως επίσης και άλλες πληροφορίες που έχουν να κάνουν με τον ιδιοκτήτη (owner) της εφαρμογής, με το πότε ξεκίνησε η διαδικτυακή λειτουργία της εφαρμογής, τον DNS admin, τον DNS reserve σε ποιο domain ανήκει αυτή η εφαρμογή που εξετάζουμε και την IP του web server. Αποτελούν όπως είναι φυσικό αδυναμία για την εφαρμογή μας όλα τα παραπάνω μιας και κάποιος θα μπορούσε να εκμεταλλευτεί αυτές τις πληροφορίες και να βασιστεί πάνω σε αυτά τα στοιχεία για να πραγματοποιήσει κάποια επίθεση.

3.2.5 Application Discovery (Εντοπισμός Εφαρμογών Ιστού)

Η ανακάλυψη εφαρμογών είναι μια δραστηριότητα που προσανατολίζεται στον εντοπισμό των εφαρμογών ιστού που φιλοξενούνται στον web server/application sever. Αυτή η ανάλυση είναι σημαντική επειδή συχνά δεν υπάρχει απευθείας σύνδεση που να συνδέει την κύρια εφαρμογή προς τα πίσω. Η

ανακάλυψη εφαρμογών μπορεί να είναι πολύ χρήσιμη στο να αποκαλύψει λεπτομέρειες όπως δικτυακές εφαρμογές που χρησιμοποιούνται για διαχειριστικούς σκοπούς. Επιπλέον, μπορεί να αποκαλύψει παλιές εκδόσεις των αρχείων ή αντικείμενα, όπως μη διαγραμμένα κι απαρχαιωμένα scripts, δημιουργημένα κατά τη φάση της δοκιμής / ανάπτυξης είτε ως αποτέλεσμα της συντήρησης.

Πολλές εφαρμογές έχουν γνωστά τρωτά σημεία και γνωστές στρατηγικές επίθεσης μπορούν να τα αξιοποιήσουν προκειμένου να αποκτήσουν τον απομακρυσμένο έλεγχο ή να αποκτήσουν δεδομένα. Επιπλέον, πολλές εφαρμογές είναι λάθος διαμορφωμένες ή δεν είναι ενημερωμένες λόγω της εντύπωσης ότι χρησιμοποιούνται μόνο "εσωτερικά" και ως εκ τούτου δεν υπάρχει κίνδυνος. Με τον πολλαπλασιασμό των virtual(εικονικών) web server, η παραδοσιακή 1-προς-1 σχέση μεταξύ της διεύθυνσης IP και του web server χάνει μεγάλο μέρος της αρχικής της σημασίας. Δεν είναι ασυνήθιστο να έχουμε πολλαπλές ιστοσελίδες/εφαρμογές ιστού των οποίων η τα συμβολικά ονόματα αναλύονται στην ίδια διεύθυνση IP (αυτό το σενάριο δεν περιορίζεται σε hosting περιβάλλοντα αλλά ισχύει επίσης και για τα κοινά εταιρικά περιβάλλοντα). Κάποιες φορές μας δίνεται ένα σύνολο διευθύνσεων(ή, ενδεχομένως, μόνο ένα) σαν στόχο προς εξέταση. Θα μπορούσαμε να πούμε πως αυτό το σενάριο μοιάζει περισσότερο με μία δοκιμή διείσδυσης, αλλά σε κάθε περίπτωση αναμένεται από μία τέτοια ανάθεση να ελεγχθούν όλες οι εφαρμογές ιστού που είναι προσπελάσιμες μέσω αυτού του στόχου. Το πρόβλημα είναι ότι η συγκεκριμένη διεύθυνση IP φιλοξενεί την υπηρεσία HTTP στη θύρα 80, αλλά αν την προσπελάσουμε καθορίζοντας τη διεύθυνση IP (το οποίο είναι το μόνο που γνωρίζουμε) αναφέρει «δεν υπάρχει web server που να έχει ρυθμιστεί σε αυτή τη διεύθυνση» ή ένα παρόμοιο μήνυμα.

Όμως αυτό το σύστημα θα μπορούσε να «κρύψει» μια σειρά web εφαρμογών, που συνδέονται με μη συσχετιζόμενα συμβολικά (DNS) ονόματα. Προφανώς η έκταση της ανάλυσής μας έχει επηρεαστεί σε βάθος από το γεγονός ότι ελέγχουμε τις εφαρμογές ή όχι επειδή δεν τις προσέχουμε ή προσέχουμε κάποιες από αυτές. Μερικές φορές, οι προδιαγραφές του στόχου είναι πιο πλούσιες - ίσως να μας έχει δοθεί μια λίστα με τις διευθύνσεις IP και τα αντίστοιχα συμβολικά τους ονόματα. Παρ' όλα αυτά, αυτή η λίστα μπορεί να μεταφέρει μερικές πληροφορίες δηλαδή θα μπορούσε να παραλείψει ορισμένα συμβολικά ονόματα - και ο client να μην μπορεί να το γνωρίζει καν αυτό (αυτό είναι πιο πιθανό να συμβεί σε μεγάλους οργανισμούς). Άλλα ζητήματα που επηρεάζουν το σκοπό της αξιολόγησης της εφαρμογής παρουσιάζονται από τις εφαρμογές ιστού και δημοσιεύονται σε μη προφανή URL. (πχ http://www.example.com/some-strange-URL), οι οποίες δεν αναφέρονται αλλού. Αυτό μπορεί να συμβεί είτε από λάθος είτε (λόγω κακής ρύθμισης), είτε εσκεμμένα (για παράδειγμα προβάλλονταν διαχειριστικά interfaces). Για να αντιμετωπιστούν αυτά τα ζητήματα είναι απαραίτητο να εκτελεστεί η αναζήτηση εφαρμογών ιστού.

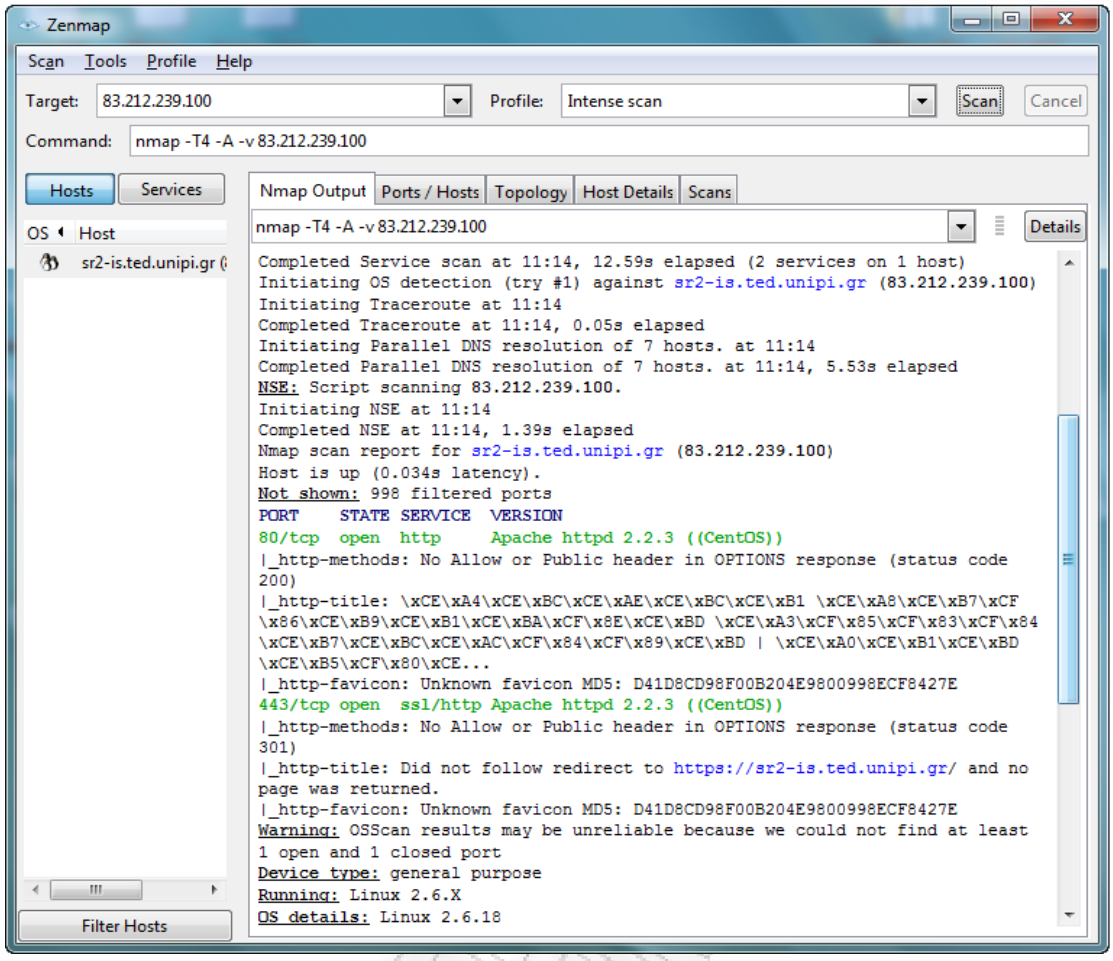
BLACK BOX TESTING

Στον έλεγχο αυτό προσπαθούμε να βρούμε τις εφαρμογές, οι οποίες υπάρχουν στον ιστότοπο του τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς. Χρησιμοποιώντας την υπηρεσία Netcraft Search DNS του εργαλείου Netcraft και βάζοντας απλά το DNS (.ds.unipi.gr) μπορούμε να

δούμε τους servers που υποστηρίζουν την εφαρμογή, καθώς επίσης και τα λειτουργικά συστήματα, που αυτοί χρησιμοποιούν. Τα αποτελέσματα που πήραμε από τον έλεγχο φαίνονται στην εικόνα 37.

Εικόνα : **Αποτελέσματα από τη χρησιμοποίηση του on-line εργαλείου Netcraft για την ανακάλυψη των servers του τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς.**

Διαπιστώσαμε μετά από τον παραπάνω έλεγχο πως μας εμφανίζεται μόνο μία εφαρμογή αποτέλεσμα που δεν ανταποκρίνεται στην πραγματικότητα καθώς γνωρίζουμε εκ των προτέρων πως λειτουργούν πολύ περισσότερες εφαρμογές κάτω από το hosting του συγκεκριμένου ιστότοπου. Αυτό το αποτέλεσμα δείχνει πως ο server που εξετάζουμε κρύβει προς τα «έξω» πληροφορίες αυτού του είδους. Ακόμη, σε αυτό τον έλεγχο θα πρέπει να προσδιορίσει ο ελεγκτής τις ποικίλες εφαρμογές, που λειτουργούν, και συγκεκριμένα σε ποιες πόρτες του server. Χρησιμοποιήσαμε το εργαλείο Zenmap, με το οποίο διενεργήσαμε αυτό τον έλεγχο. Με το εργαλείο Zenmap, λοιπόν, ανακαλύψαμε όλες τις ανοικτές πόρτες, τα πρωτόκολλα, που τρέχουν σε αυτές, καθώς επίσης και την έκδοση του πρωτοκόλλου.



Εικόνα **: Χρησιμοποίηση του εργαλείου Nmap για τον προσδιορισμό των ανοικτών ports του server του τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς.**

Εδώ παρατηρούμε πως μετά τον έλεγχο του εργαλείου Nmap ανακαλύψαμε 2 συγκεκριμένες πόρτες ανοικτές (80, 443) σε συγκεκριμένα services (http,ssl) και μάλιστα είναι διαθέσιμες και οι εκδόσεις του Apache server που χρησιμοποιείται. Αυτό, που αξίζει σ' αυτό το σημείο να παρατηρήσουμε, είναι ότι το εργαλείο WebScarab δεν παρέχει τέτοιου είδους δυνατότητες όπως τα εργαλεία, που χρησιμοποιήθηκαν παραπάνω για τους συγκεκριμένους ελέγχους, κάτι που είναι πολύ χρήσιμο για τον ελεγκτή και τη συνέχιση των ελέγχων του.

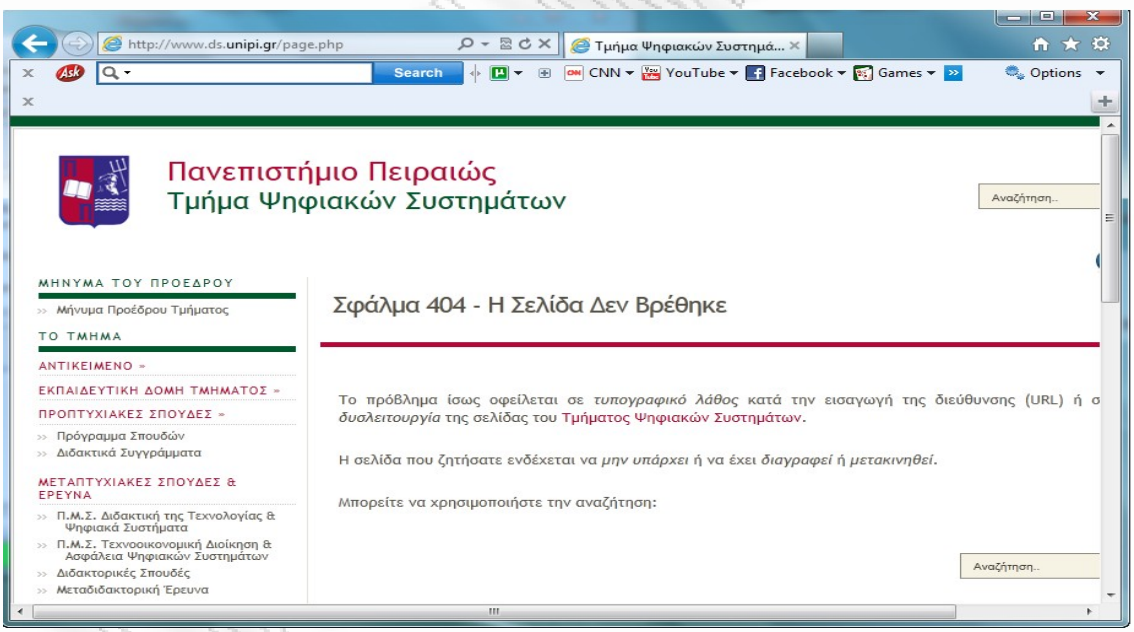
3.2.6 Analysis of Error Codes (Ανάλυση Κωδικών Σφαλμάτων)

Συχνά κατά τη διάρκεια μιας δοκιμής διείσδυσης σε εφαρμογές ιστού ερχόμαστε αντιμέτωποι με πολλούς κώδικες σφαλμάτων που δημιουργούνται από τις εφαρμογές ή τους web servers. Είναι πιθανόν την εμφάνιση αυτών των σφαλμάτων να την προκαλεί ένα συγκεκριμένο αίτημα είτε ειδικά κατασκευασμένο με κάποιο εργαλείο είτε χειροκίνητα. Αυτοί οι κώδικες είναι πάρα πολύ χρήσιμοι στους ελεγκτές των δοκιμών κατά τη διάρκεια των δραστηριοτήτων τους καθώς αποκαλύπτουν αρκετή πληροφορία για τις βάσεις δεδομένων, τα λάθη και άλλα τεχνολογικά στοιχεία που συσχετίζονται

απευθείας με την εφαρμογή. Κατά τη διάρκεια μιας δοκιμής διείσδυσης, οι εφαρμογές είναι δυνατόν να αποκαλύψουν πληροφορίες που δεν προβλέπεται να είναι ορατές από τον τελικό χρήστη. Πληροφορίες όπως οι κωδικοί σφάλματος μπορούν να ενημερώσουν τον ελεγκτή για τεχνολογίες και προϊόντα που χρησιμοποιούνται από την εφαρμογή. Σε πολλές περιπτώσεις, οι κωδικοί σφάλματος μπορούν εύκολα να ανακληθούν χωρίς χρειάζονται ειδικές δεξιότητες ή εργαλεία, λόγω κακών εξαιρέσεων κατά τον σχεδιασμό και την ανάπτυξη του κώδικα. Σαφώς, εστιάζοντας μόνο στην εφαρμογή ιστού δε θα είναι μια εξαντλητική δοκιμασία. Δε μπορεί να είναι τόσο πλήρης όσο οι πληροφορίες που ενδεχομένως συλλέγονται από την εκτέλεση μιας ευρύτερης ανάλυσης της υποδομής.

BLACK BOX TESTING

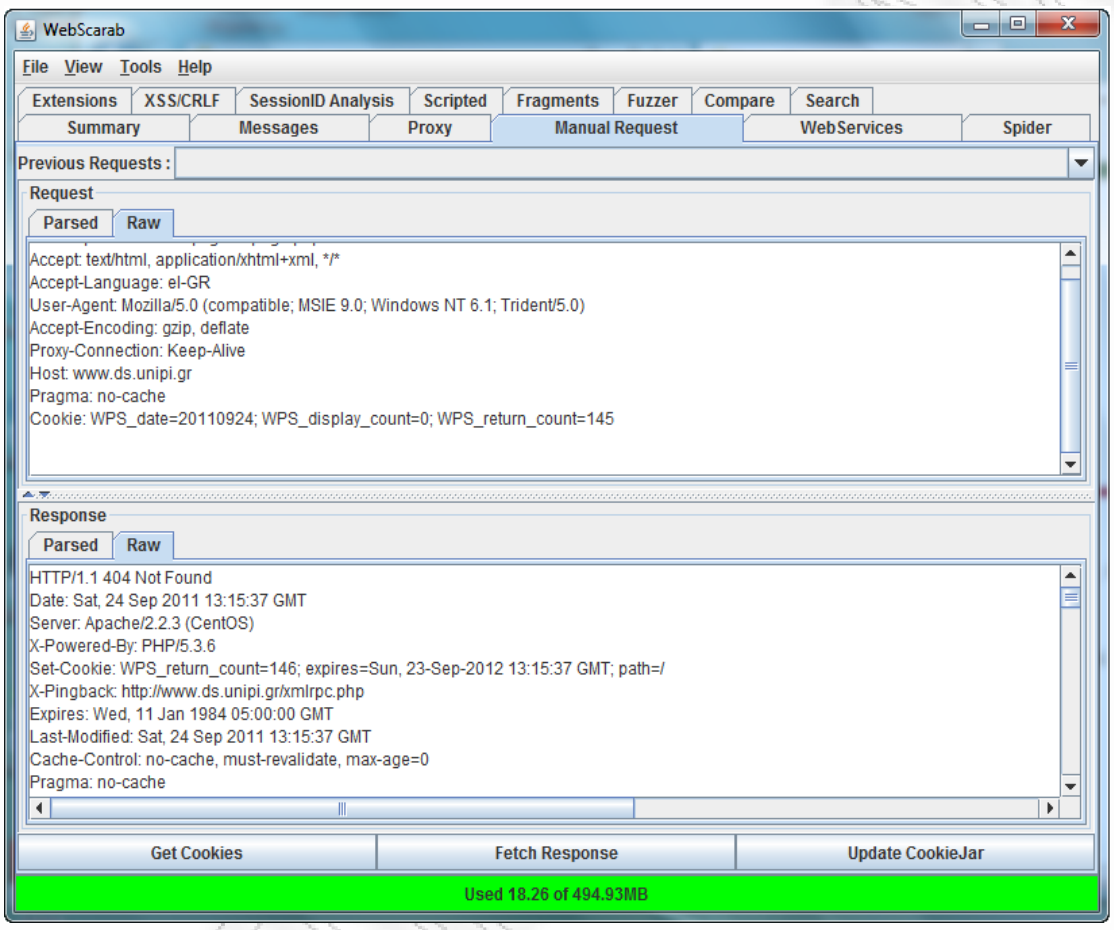
Σ' αυτό το σημείο θα αναλύσουμε τους πιο συχνούς κώδικες (μηνύματα λάθους) και θα φέρουμε στο προσκήνιο τα βήματα ανακάλυψης της ευπάθειας μέσα από αυτά. Η πιο σημαντική πτυχή αυτής της δραστηριότητας είναι να επικεντρώσουμε την προσοχή κάποιου σε αυτά τα σφάλματα, βλέποντάς τα σαν μία συλλογή πληροφορίας που θα έχει θα βοηθήσει στα επόμενα βήματα της ανάλυσής μας. Μία καλή συλλογή μπορεί να διευκολύνει την αποτελεσματικότητα της αξιολόγησης μειώνοντας τον συνολικό χρόνο της εκτέλεσης της δοκιμής διείσδυσης. Για τον συγκεκριμένο έλεγχο αιτούμε μια ανύπαρκτη σελίδα. Στην προκειμένη περίπτωση το αίτημα είναι: GET http://www.ds.unipi.gr/page.php. Στην εικόνα 39 μπορούμε να δούμε τον browser όταν αιτούμαστε μία σελίδα που δεν υπάρχει.



Εικόνα : **Το μήνυμα λάθους που εμφανίζεται ως αποτέλεσμα στον browser όταν γίνεται η αίτηση ανύπαρκτης σελίδας**

Στο σημείο αυτό πρέπει να αναφέρουμε πως ένας κακόβουλος χρήστης ή, ακόμη χειρότερα για την εφαρμογή, ένας hacker χρησιμοποιεί τα μηνύματα λάθους και τις ανύπαρκτες σελίδες για να φτάσει εκμεταλλευόμενος τις διάφορες ευπάθειες και «τρύπες» του συστήματος ακόμη και ως τη βάση

δεδομένων της εφαρμογής με ότι αυτό μπορεί να συνεπάγεται. Η συλλογή πληροφοριών στις εφαρμογές ιστού με τεχνολογία client-server είναι δύσκολη, αλλά οι πληροφορίες, οι οποίες ανακαλύπτονται, μπορούν να είναι πολύ χρήσιμες για τη σωστή εκτέλεση μιας απόπειρας επίθεσης, (π.χ. επιθέσεις SQL injection ή Cross Site Scripting) από τους ίδιους τους διαχειριστές του συστήματος, η οποία είναι σε θέση να μειώσει αποτελεσματικά τα λάθη σχεδιασμού και διαχείρισης της εφαρμογής. Έχουμε την απάντηση, που λάβαμε από τον server στην εικόνα 40.



Εικόνα : **Αποτελέσματα του WebScarab της απάντησης του server με ένα μήνυμα λάθους σε μια αίτηση ανύπαρκτης σελίδας και οι πληροφορίες που επιστρέφει αυτός.**

Υπάρχει το μήνυμα λάθους "HTTP/1.1 404 Not Found" και οι υπόλοιπες πληροφορίες για το περιβάλλον του server είναι αρκετές και μάλιστα μας δείχνουν πληροφορίες που θα ήταν πολύ χρήσιμες για κάποιο hacker ο οποίος εύκολα θα μπορούσε να εκμεταλλευτεί τον κώδικα που μας επιστρέφει η εφαρμογή μετά την αίτηση μίας ανύπαρκτης σελίδας. Τα αποτελέσματα που επιστρέφει ο server έχουν να κάνουν τόσο με την ταυτότητα του «ιδιοκτήτη» της εφαρμογής δηλαδή αυτού που την ανέπτυξε όσο και με τα στοιχεία της πλατφόρμας (όνομα, έκδοση) στην οποία αναπτύχθηκε ο ιστότοπος που εξετάζουμε. Επίσης το αίτημα της ανύπαρκτης σελίδας μας επέστρεψε αρκετό κώδικα ο οποίος αναφέρεται και σε εφαρμογές οι οποίες φιλοξενούνται στον server τον οποίο εξετάζουμε.

3.3 Configuration Management Testing (Ελεγχος Διαχείρισης Διαμόρφωσης)

Συχνά η ανάλυση της υποδομής και της αρχιτεκτονικής της τοπολογίας μπορούν να αποκαλύψουν πολλά για μία εφαρμογή ιστού. Πληροφορίες όπως ο πηγαίος κώδικας, HTTP μέθοδοι που επιτρέπονται, διαχειριστική λειτουργικότητα, μέθοδοι αυθεντικοποίησης και διαμόρφωση υποδομών μπορούν να εύκολα να ανακτηθούν.

3.3.1 SSL/TLS Testing (SSL Version, Algorithms, Key length, Digital Cert, Validity)

Το SSL και τα oTLS είναι δύο πρωτόκολλα τα οποία παρέχουν, με την υποστήριξη της κρυπτογραφίας, ασφαλή κανάλια για την προστασία, την εμπιστευτικότητα και την αυθεντικοποίηση της πληροφορίας που μεταφέρεται. Λαμβάνοντας υπ' όψιν την κρισιμότητα αυτών των εφαρμογών ασφάλειας, είναι σημαντικό να ελέγξουμε την χρήση ενός ισχυρού αλγόριθμου κρυπτογράφησης και την ορθή εφαρμογή του. Λόγω των ιστορικών περιορισμών εξαγωγής της υψηλού βαθμού της κρυπτογραφίας, οι νέοι web servers είναι ικανοί να χειριστούν μια αδύναμη υποστήριξη κρυπτογράφησης. Ακόμη και αν χρησιμοποιούνται και εγκαθίστανται υψηλού βαθμού κρυπτογραφικοί αλγόριθμοι, κάποιο σφάλμα στην εγκατάσταση του server θα μπορούσε να χρησιμοποιηθεί ώστε να αναγκάσει τη χρήση ενός πιο αδύναμου κρυπτογραφικού αλγόριθμου να αποκτήσει πρόσβαση στο υποτιθέμενο ασφαλές κανάλι επικοινωνίας.

Το HTTP clear-text πρωτόκολλο προσφέρεται φυσιολογικά μέσω ενός SSL ή ενός TLS καναλιού, που αναλύεται σε κίνηση HTTPS. Εκτός από την παροχή κρυπτογράφησης κατά της μεταφορά των δεδομένων, το https επιτρέπει την αναγνώριση των server (και εναλλακτικά των clients) με την έννοια των ψηφιακών πιστοποιητικών. Ιστορικά έχουν τεθεί οριοθετήσεις από την κυβέρνηση των Ηνωμένων Πολιτειών ώστε να επιτρέπουν κρυπτοσυστήματα να εξάγονται με μήκος κλειδιού το πολύ 40-bits, ένα τέτοιο μήκος κλειδιού που θα μπορούσε να «σπάσει» και να επιτρέψει την αποκρυπτογράφηση της επικοινωνίας. Από τότε οι κρυπτογραφικοί κανονισμοί εξαγωγών είναι χαλαρή, αν και μερικοί περιορισμοί εξακολουθούν να αντέχουν, ωστόσο είναι σημαντικό να ελέγχουμε την SSL διαμόρφωση που χρησιμοποιείται ώστε να αποφευχθεί να χρησιμοποιηθεί μία κρυπτογραφική υποστήριξη που είναι εύκολο να νικηθεί. Οι υπηρεσίες βασισμένες σε SSL πρωτόκολλο δε θα πρέπει να προσφέρουν τη δυνατότητα να διαλέξεις αδύναμο κρυπτογραφικό αλγόριθμο.

Τεχνικά, ο προσδιορισμός του κρυπτογραφικού αλγόριθμου γίνεται ως εξής. Στην αρχική φάση της εγκατάστασης μίας SSL σύνδεσης, ο client στέλνει στον server ένα λεπτομερές μήνυμα "Client Hello" μαζί με άλλες πληροφορίες που ο κρυπταλγόριθμος είναι σε θέση να χειριστεί. Ο client είναι συνήθως ένας web browser, ο πλέον γνωστός SSL client στις μέρες μας, αλλά όχι απαραίτητα από τη στιγμή που θα μπορούσε να είναι οποιαδήποτε εφαρμογή που έχει εγκαταστήσει SSL πρωτόκολλο. Το ίδιο ισχύει και για τον server, ο οποίος δε χρειάζεται να είναι ένας web server, παρόλο ότι είναι η πιο κοινή περίπτωση. Για παράδειγμα, μια αξιοσημείωτη κατηγορία SSL client είναι οι SSL proxies όπως ο

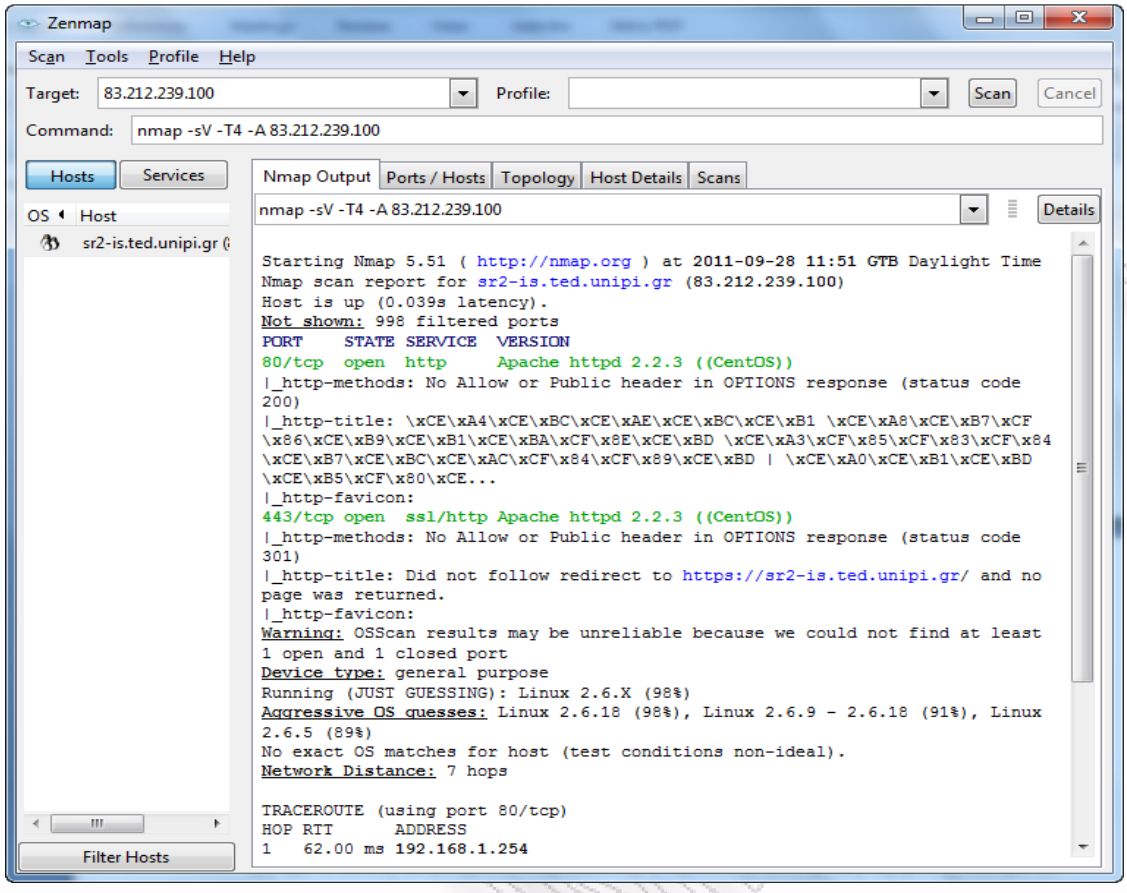
stunnel (www.stunnel.org) όπου μπορεί να χρησιμοποιηθεί επιτρέποντας εργαλεία που δεν υποστηρίζουν SSL πρωτόκολλο να επικοινωνήσουν με SSL υπηρεσίες. Ένα πρόγραμμα κρυπτογράφησης προσδιορίζεται από ένα πρωτόκολλο κρυπτογράφησης (DES, RC4, AES), το μήκος του κλειδιού κρυπτογράφησης(40, 56, 128 bits) και έναν αλγόριθμο κατακερματισμού (hash) (SHA, MD5) που χρησιμοποιείται για τον έλεγχο ακεραιότητας. Με τη λήψη του μηνύματος "Client Hello", ο server αποφασίζει ποιό κρυπτογραφικό πρόγραμμα θα χρησιμοποιήσει σε αυτή την περίπτωση. Είναι δυνατόν, για παράδειγμα μέσω διαμόρφωσης οδηγιών, να ορίσουμε ποιο κρυπτογραφικό πρόγραμμα θα επιλέξει ο server. Με αυτόν τον τρόπο μπορούμε να ελέγξουμε, για παράδειγμα, πότε οι συνομιλίες με τους clients θα υποστηρίζουν συνομιλίες 40-bit μόνο και πότε όχι.

BLACK BOX TESTING

Με σκοπό την ανακάλυψη τυχόν υποστήριξης αδύναμων αλγορίθμων κρυπτογράφησης , οι πόρτες που σχετίζονται με τα πρωτόκολλα SSL/TLS πρέπει να προσδιοριστούν. Αυτές τυπικά περιλαμβάνουν την πόρτα 443 η οποία είναι η προκαθορισμένη https πόρτα, ωστόσο αυτό μπορεί να αλλάζει γιατί:

a) οι υπηρεσίες https μπορεί να είναι διαμορφωμένες να τρέχουν σε μη-προκαθορισμένες θύρες και
b) μπορεί να υπάρχουν επιπρόσθετες SSL/TLS υπηρεσίες που σχετίζονται με την εφαρμογή ιστού. Γενικά στην ανακάλυψη υπηρεσιών απαιτείται η αναγνωρίσουμε αυτές τις πόρτες.

Ο ZenMap scanner μέσω της scan επιλογής "-sV" είναι σε θέση να ανακαλύψει τις υπηρεσίες SSL. Οι Scanners ευπαθειών, εκτός από την εκτέλεση της υπηρεσίας ανακάλυψης, μπορεί να περιλαμβάνουν ελέγχους ενάντια στους αδύναμους αλγόριθμους κρυπτογράφησης. Στην παρακάτω εικόνα φαίνονται τα αποτελέσματα χρήσης του εργαλείου ZenMap.



Εικόνα : **Αποτελέσματα που εμφανίζει το εργαλείο ZenMap σχετικά με τον SSL/TLS έλεγχο.**

Τα αποτελέσματα που μας δίνει το ZenMap δείχνουν πως τόσο η θύρα 80 όσο και η 443 που αφορούν τις επιθέσεις μέσω των πρωτοκόλλων SSL/TLS που εξετάζουμε είναι «ανοιχτές» παρουσιάζεται δε και η έκδοση του Apache Server καθώς και το είδος του λειτουργικού που χρησιμοποιείται, πληροφορίες που η ανακάλυψη τους αποτελεί ευπάθεια που μπορεί εύκολα να εκμεταλλευτεί κάποιος κακόβουλος χρήστης.

3.3.2 DB Listener Testing (Έλεγχος του Ακροατή της Βάσης Δεδομένων)

Κατά τη διάρκεια διαμόρφωσης ενός server βάσης δεδομένων πολλοί διαχειριστές των βάσεων αυτών δεν παίρνουν υπ' όψιν τους επαρκώς την ασφάλεια του στοιχείου του ακροατή (listener) της βάσης δεδομένων. Ο ακροατής μπορεί να αποκαλύψει ευαίσθητα δεδομένα καθώς και ρυθμίσεις διαμόρφωσης ή περιόδους λειτουργίας της βάσης δεδομένων εάν έχει διαμορφωθεί μη ασφαλώς και διερευνάται με χειροκίνητες ή αυτοματοποιημένες μεθόδους. Η πληροφορία που αποκαλύπτεται συχνά θα φανεί χρήσιμη σε έναν ελεγκτή που θα τη χρησιμοποιήσει ως είσοδο σε δοκιμές συνέχειας πιο αποδοτικές. Ο ακροατής της βάσης δεδομένων είναι ένας ανεξάρτητος διαδικτυακός δαίμονας στις βάσεις δεδομένων. Περιμένει για αιτήματα συνδέσεις και απομακρυσμένους clients. Αυτός ο δαίμονας μπορεί να τεθεί υπό κίνδυνο και ως εκ τούτου να επηρεάσει και τη διαθεσιμότητα της βάσης δεδομένων. Ο ακροατής της βάσης δεδομένων είναι ένα σημείο εισόδου για απομακρυσμένες συνδέσεις σε μια βάση δεδομένων της.

«Ακούει» για αιτήματα σύνδεσης και τα διαχειρίζεται αναλόγως. Ο έλεγχος αυτός μπορεί να συμβεί εάν ο ελεγκτής έχει πρόσβαση στην υπηρεσία –ο έλεγχος πρέπει να πρέπει να γίνει από το Intranet.

Ο ακροατής, εξ' ορισμού, ακούει στην πόρτα 1521(η πόρτα 2483 είναι επίσημα η καινούρια εγκατεστημένη πόρτα για τον TNS ακροατή και η 2484 θύρα για τον TNS ακροατή που χρησιμοποιεί το πρωτόκολλο SSL). Είναι καλή πρακτική να αλλάξουμε την πόρτα που συνήθως ακούει ο ακροατής και να χρησιμοποιήσουμε μία άλλη αυθαίρετη πόρτα. Εάν αυτός ο ακροατής είναι «κλειστός η απομακρυσμένη πρόσβαση δεν είναι δυνατή. Αν αυτό συμβαίνει η εφαρμογή κάποιου θα αποτύχει δημιουργώντας επίσης άρνηση παροχής της υπηρεσίας.

Σε πιθανές περιοχές της επίθεσης:

- Σταματήστε τον ακροατή - μπορεί να δημιουργηθεί μία επίθεση άρνησης παροχής υπηρεσιών
- Ορίστε ένα password και αποτρέψτε άλλους απ' το να χειρίζονται τον ακροατή – μπορεί να υποκλαπεί η βάση δεδομένων.
- Καταγράψτε τα ίχνη και τα log files σε ένα αρχείο προσβάσιμο στον ιδιοκτήτη της όλης διαδικασίας (συνήθως η Oracle) – πιθανή διαρροή πληροφοριών
- Αποκτήστε λεπτομερείς πληροφορίες για τον ακροατή, τη βάση δεδομένων και τη διαμόρφωση της εφαρμογής.

BLACK BOX TESTING

Μετά από αναζήτηση μεθόδου που να εξεταστεί ο DB Listener της Βάσης Δεδομένων που αφορά την δική μας εφαρμογή δηλαδή το site www.ds.unipi.gr διαπιστώσαμε πως δεν έχουμε τρόπο να πραγματοποιήσουμε με την τεχνική black box αυτόν τον έλεγχο. Η Βάση Δεδομένων που χρησιμοποιεί η συγκεκριμένη εφαρμογή είναι PHP / MySQL και δεν μπορούμε να εφαρμόσουμε τις πληροφορίες και συμβουλές που αναφέρονται παραπάνω καθώς αφορούν βάσεις δεδομένων της Oracle. Όμως θα δοκιμάσουμε με το εργαλείο που προτείνει ο OWASP να ελέγξουμε τι αποτελέσματα παίρνουμε και στη δική μας περίπτωση.

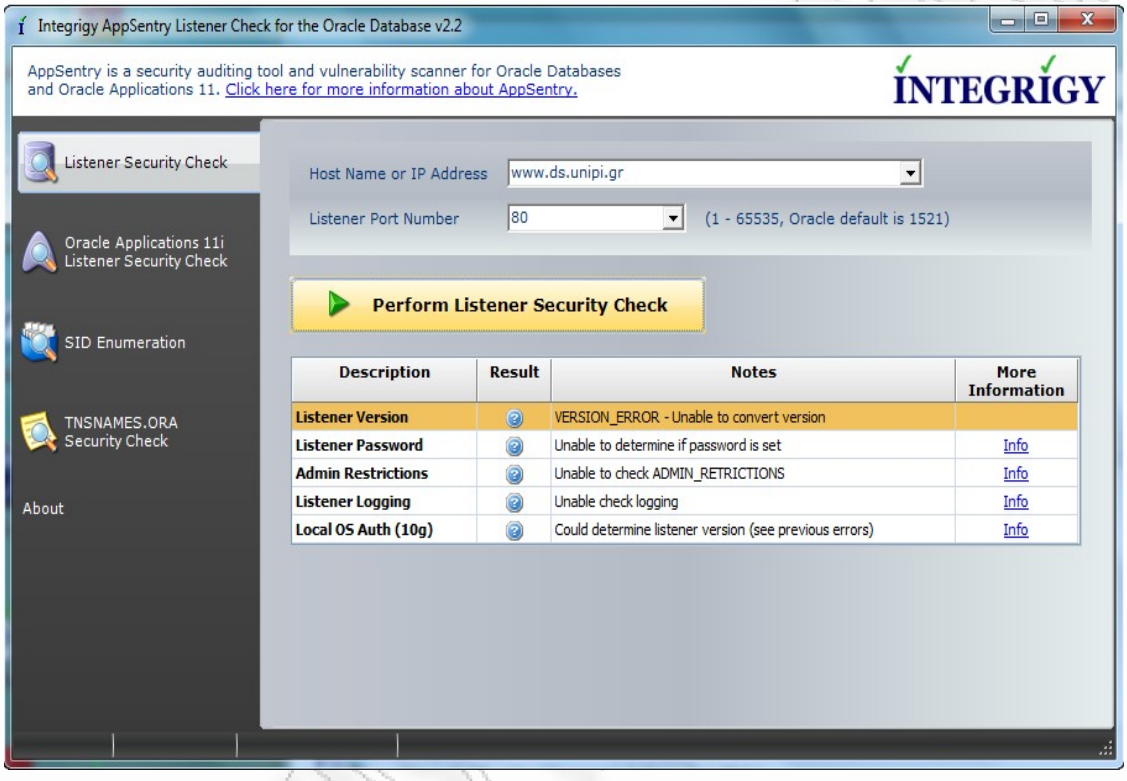
Μετά την ανακάλυψη της θύρας στην οποία "ακούει" ο listener της Βάσης Δεδομένων μας, μπορεί κάποιος να τον αξιολογήσει κάνοντας χρήση ενός εργαλείου που αναπτύχθηκε από την Integrigy. Το εργαλείο ελέγχει τα παρακάτω:

- Κωδικός πρόσβασης του Listener: Σε πολλά συστήματα Βάσης Δεδομένων ο κωδικός πρόσβασης του Ακροατή(Listener) μπορεί να μην έχει οριστεί με κάποια τιμή. Εάν ο κωδικός πρόσβασης, λοιπόν, δεν έχει οριστεί θα μπορούσε ένας επιτιθέμενος να ορίσει εκείνος έναν και να επισκιάσει τον Listener, αν και ο κωδικός πρόσβασης μπορεί να αφαιρεθεί με τοπική επεξεργασία του αρχείου Listener.ora
- Ενεργοποίηση των αρχείων καταγραφής: Το εργαλείο που αναφέρθηκε παραπάνω ελέγχει επίσης να δει εάν η καταγραφή των γεγονότων που συμβαίνουν στη βάσης μας έχει οριστεί. Εάν δεν έχει γίνει κάτι τέτοιο, τότε δε θα μπορεί κάποιος να ανιχνεύσει οποιαδήποτε αλλαγή στον Listener ή

να έχει κάποιο ιστορικό του. Επίσης δε θα μπορούσε να ελεγχθεί ούτε η περίπτωση της brute force επίθεσης στον Listener.

- Περιορισμοί Διαχείρισης: Εάν δεν είναι ενεργοποιημένοι οι περιορισμοί του διαχειριστή είναι δυνατόν να γίνει η χρήση της εντολής "SET" εξ αποστάσεως.

Μετά λοιπόν τη χρήση του εργαλείου τα αποτελέσματα που πήραμε φαίνονται στην εικόνα που ακολουθεί:



Εικόνα : **Αποτελέσματα του εργαλείου Integrigy για το DB Listener Testing.**

Όπως μπορούμε να δούμε δεν είμαστε σε θέση να ξέρουμε εάν έχει οριστεί ο κωδικός πρόσβασης όπως επίσης ούτε μπορούμε να αντλήσουμε πληροφορίες για την έκδοση του Listener που χρησιμοποιείται στην εφαρμογή μας. Ακόμη οι περιορισμοί του διαχειριστή είναι μη ορατοί ως προς εμάς όπως επίσης δεν έχουμε τη δυνατότητα να ελέγξουμε τα αρχεία καταγραφής. Όλα τα παραπάνω αποτελέσματα δείχνουν πως η εφαρμογή μας σε σχέση με τον έλεγχο που διενεργήσαμε δεν δείχνει ευάλωτη σε τέτοιου είδους επιθέσεις.

3.3.3 Infrastructure Configuration Management Testing

Η εγγενής πολυπλοκότητα των διασυνδεδεμένων και ετερογενών υποδομών των web server, η οποία μπορεί να μετρά εκατοντάδες διαδικτυακές εφαρμογές, καθιστά τη διαμόρφωση της διαχείρισης και την επανεξέτασή της ένα θεμελιώδες βήμα για τη δοκιμή και την ανάπτυξη κάθε εφαρμογής. Στην πραγματικότητα αρκεί και μόνο μια ευπάθεια για να υπομονεύσει την ασφάλεια ολόκληρης της υποδομής, ακόμη και τα μικρά και σχεδόν ασήμαντα προβλήματα μπορούν να εξελιχθούν σε σοβαρούς κινδύνους για μια άλλη εφαρμογή στον ίδιο server. Για την αντιμετώπιση αυτών των προβλημάτων, είναι υψίστης σημασίας η εκτέλεση μίας σε βάθος αναθεώρησης των ρυθμίσεων και των γνωστών ζητημάτων ασφάλειας.

Η σωστή διαμόρφωση της διοίκησης της υποδομής του web server είναι πολύ σημαντική προκειμένου να διατηρηθεί η ασφάλεια της ίδιας της εφαρμογής. Εάν στοιχεία όπως το λογισμικό του web server, οι back-end servers της βάσης δεδομένων ή οι server αυθεντικοποίησης δεν έχουν επανεξεταστεί και δεν είναι ασφαλείς, θα μπορούσαν να εισάγουν ανεπιθύμητους κινδύνους ή καινούριες ευπάθειες που θα μπορούσαν να θέσουν σε κίνδυνο την ίδια την εφαρμογή. Για παράδειγμα, ένα τρωτό σημείο ενός web server θα μπορούσε να επιτρέψει σε έναν απομακρυσμένο επιτιθέμενο να αποκαλύψει τον πηγαίο κώδικα της εφαρμογής (μια ευπάθεια που έχει ανακύψει πολλές φορές τόσο σε web servers όσο και σε server εφαρμογών) θα μπορούσε να θέσει σε κίνδυνο την εφαρμογή, όπως οι ανώνυμοι χρήστες θα μπορούσαν να χρησιμοποιήσουν τις πληροφορίες που γίνονται γνωστές στον πηγαίο κώδικα σε ισχυρές επιθέσεις απέναντι στην εφαρμογή ή τους χρήστες της.

Για να ελέγξουμε την διαμόρφωση της διοίκησης της υποδομής θα πρέπει να ληφθούν τα ακόλουθα βήματα:

- Τα διάφορα στοιχεία που συνθέτουν την υποδομή πρέπει να καθοριστούν έτσι ώστε να καταλάβουμε πως αλληλεπιδρούν με την web εφαρμογή και πως μπορούν να επηρεάσουν την ασφάλειά της.
- Όλα τα στοιχεία της εφαρμογής πρέπει να επανεξεταστούν, ώστε να βεβαιωθούμε ότι δεν διατηρούν στις τάξεις τους κάποια από τις γνωστές αδυναμίες.
- Μία επανεξέταση πρέπει να γίνει από τα διαχειριστικά εργαλεία που χρησιμοποιούνται για τη συντήρηση όλων των διαφορετικών στοιχείων.
- Τα συστήματα αυθεντικοποίησης , εάν υπάρχουν, πρέπει να επανεξεταστούν προκειμένου να διασφαλίσουμε ότι υπηρετούν τις ανάγκες της εφαρμογής και δεν έχουν παραποιηθεί από εξωτερικούς χρήστες με σκοπό να αποκτήσουν πρόσβαση.
- Μία λίστα ων προκαθορισμένων θυρών οι οποίες απαιτούνται από την εφαρμογή θα πρέπει να συντηρείται και να διατηρείται υπό έλεγχο αλλαγής.

BLACK BOX TESTING

Επανεξέταση της αρχιτεκτονικής της εφαρμογής

Η αρχιτεκτονική της εφαρμογής θα πρέπει να επανεξεταστεί μέσα από τον έλεγχο για να καθοριστούν ποια είναι τα διαφορετικά συστατικά που χρησιμοποιούνται για την κατασκευή της web εφαρμογής. Σε μικρού μεγέθους setups, όπως μια απλή εφαρμογή τύπου CGI, ένας μόνο server θα μπορούσε να χρησιμοποιηθεί που θα «τρέχει» τον web server, ο οποίος εκτελεί την γλώσσα C, Perl, οι εφαρμογές τύπου Shell CGI και ίσως και έναν μηχανισμό αυθεντικοποίησης. Σε πιο πολύπλοκα setups, όπως ένα τραπεζικό σύστημα, πολλαπλοί server μπορεί να συμπεριλαμβάνονται περιέχοντας: Έναν reverse proxy, ένα frontend server, έναν server εφαρμογής και έναν server βάσης δεδομένων ή LDAP server, Κάθε ένας από αυτούς τους server θα χρησιμοποιηθεί για διαφορετικούς σκοπούς και ίσως να είναι χωρισμένοι σε διαφορετικά δίκτυα με συσκευές firewall μεταξύ τους, δημιουργώντας διαφορετικές περιμετρικές ζώνες με έτσι ώστε η πρόσβαση στον web server να μην χορηγήσει σε κανένα απομακρυσμένο χρήστη πρόσβαση στο μηχανισμό αυθεντικοποίησης και το να εκτεθούν κάποια διαφορετικά στοιχεία της εφαρμογής σε κίνδυνο να μπορεί να απομονωθεί με τέτοιον τρόπο που δε θα τίθεται σε κίνδυνο ολόκληρη η εφαρμογή.

Η κατανόηση της αρχιτεκτονικής της εφαρμογής μπορεί να είναι εύκολη καθώς αυτή η πληροφορία παρέχεται στην ομάδα που κάνει τους ελέγχους από αυτούς που αναπτύσσουν την εφαρμογή σε φόρμα εγγράφου ή μέσω συνέντευξης, αλλά μπορεί να αποδειχθεί ότι είναι πολύ δύσκολη εάν κάνουμε μία δοκιμή διείσδυσης στα τυφλά. Στην τελευταία περίπτωση, ένας ελεγκτής θα ξεκινήσει πρώτα με την παραδοχή ότι υπάρχει μια απλή εγκατάσταση(setup) και θα αντλήσει, μέσα από τις πληροφορίες που θα αποσπάσει από άλλους ελέγχους, διαφορετικά στοιχεία και εξετάζει την υπόθεση ότι η αρχιτεκτονική θα πρέπει να επεκταθεί. Ο ελεγκτής θα ξεκινήσει κάνοντας το απλό ερώτημα όπως: «Υπάρχει τοίχος ασφαλείας(firewall) που να προστατεύει τον web server;» το οποίο θα πρέπει απαντηθεί με βάση τα αποτελέσματα από τις σαρώσεις του δικτύου με στόχο τον web server και την ανάλυση κατά πόσο οι θύρες δικτύου του web server φιλτράρονται στην άκρη του δικτύου ή (μήνυμα καμίας απάντησης ή ICMP μηνύματα μη πρόσβασης λαμβάνονται) ή αν ο server είναι άμεσα συνδεδεμένος στο διαδίκτυο (πχ επιστρέφουν RST πακέτα από όλες τις πόρτες που δεν «ακούν»).

Αυτή η ανάλυση μπορεί να ληφθεί υπ όψιν προκειμένου να καθοριστεί ο τύπος του firewall που χρησιμοποιείται και βασίζεται στον έλεγχο των πακέτων:

- Είναι ένα στατικό τοίχος προστασίας(firewall) ή αποτελεί ένα φίλτρο με λίστα πρόσβασης σε ένα δρομολογητή(router);
- Πως είναι διαμορφωμένο;
- Μπορεί να παρακαμφθεί;

Η ανίχνευση ενός reserve proxy μπροστά από τον web server πρέπει να γίνει με την ανάλυση του web server banner, η οποία μπορεί να κάνει γνωστή την ύπαρξη ενός reserve proxy(πχ αν ο WebSEAL είναι ενεργός). Μπορεί επίσης να καθοριστεί η ύπαρξή του με τη λήψη των απαντήσεων που δόθηκαν από τον web server σε κάποια αιτήματα και συγκρίνοντάς τα με τις αναμενόμενες απαντήσεις. Για παράδειγμα, ορισμένοι reverse proxies λειτουργούν ως "συστήματα πρόληψης εισβολών" (ή ασπίδες ιστού) μπλοκάροντας κάποιες άγνωστες επιθέσεις που στοχεύουν στον web server. Αν ο server είναι

γνωστός ώστε να απαντήσει με ένα μήνυμα 404 σε ένα αίτημα το οποίο στοχεύει σε μια μη διαθέσιμη σελίδα και επιστρέφει ένα διαφορετικό μήνυμα λάθους για κάποιες κοινές επιθέσεις του διαδικτύου όπως αυτές που γίνονται από CGI σαρωτές, μπορεί να είναι ένδειξη για την ύπαρξη ενός reserve proxy (ή μια εφαρμογή επιπέδου firewall) το οποίο φιλτράρει τα αιτήματα και επιστρέφει διαφορετική σελίδα λάθους από αυτή που αναμένουμε. Ένα άλλο παράδειγμα: Αν ο web server επιστρέφει ένα σύνολο από διαθέσιμες HTTP μεθόδους (συμπεριλαμβάνοντας την TRACE) αλλά οι αναμενόμενες μέθοδοι επιστρέφουν σφάλματα τότε υπάρχει κάτι μεταξύ τους και τις εμποδίζει. Σε ορισμένες περιπτώσεις ακόμη και το ίδιο σύστημα προστασίας αποτρέπει τον εαυτό του.

GET / web-console/ServerInfo.jsp%00 HTTP/1.0
HTTP/1.0 200 Pragma: no-cache Cache-Control: no-cache Content-Type: text/html Content-Length: 83

<TITLE>Error</TITLE>
<BODY> <H1>Error</H1>
FW-1 at XXXXXX: Access denied.</BODY>

Παράδειγμα του server ασφαλείας του σημείου ελέγχου του Firewall-1 NG AI που προστατεύει έναν web server.

Οι reserve proxies μπορούν επίσης να εισαχθούν ως proxy "κρύπτες" για να επιταχύνουν την απόδοση των back-end server εφαρμογής. Η ανίχνευση αυτών των proxies μπορεί να γίνει με βάση, και πάλι, την επικεφαλίδα του server ή με αιτήματα χρόνου που θα πρέπει να αποθηκευτούν προσωρινά στον server και συγκριθεί ο χρόνος που χρειάζεται ο server για το πρώτο αίτημα με τα μεταγενέστερα αιτήματα. Ένα άλλο στοιχείο που μπορεί να ανιχνευθεί είναι οι εξ-ισορροπιστές του φορτίου του δικτύου. Τυπικά αυτά τα συστήματα θα εξισορροπήσουν μια TCP/IP θύρα που θα τους δοθεί σε πολλαπλούς servers βασισμένα σε διαφορετικούς αλγόριθμους (round-robin, φορτίο του web server, αριθμός των αιτημάτων, κλπ) Έτσι, η ανίχνευση αυτού του στοιχείου της αρχιτεκτονικής θα πρέπει να γίνει εξετάζοντας πολλαπλά αιτήματα και συγκρίνοντας τα αποτελέσματα προκειμένου να διαπιστώσουμε εάν τα αιτήματα πηγαίνουν στον ίδιο ή σε διαφορετικούς web server. Για παράδειγμα, με βάση την ημερομηνία. Η επικεφαλίδα, εάν δεν είναι συγχρονισμένα τα ρολόγια του server. Σε ορισμένες περιπτώσεις η διαδικασία εξισορρόπησης του φορτίου του δικτύου μπορεί να δώσει νέες πληροφορίες στις επικεφαλίδες που τις κάνει να ξεχωρίζουν ευδιάκριτα, όπως το AlteonP cookie που προτάθηκε από την εταιρεία εξισορρόπησης φορτίου Nortel's Alteon WebSystems.

Οι web server εφαρμογής είναι εύκολο να ανιχνευθούν. Το αίτημα για διάφορους πόρους ελέγχεται από την ίδια την εφαρμογή (όχι τον web server) και η επικεφαλίδα της απάντησης θα διαφέρει σημαντικά (συμπεριλαμβάνοντας διαφορετικές ή πρόσθετες τιμές στην επικεφαλίδα της απάντησης). Ένας άλλος τρόπος να ανιχνεύσουμε τα παραπάνω είναι να ελέγξουμε εάν ο web server προσπαθεί να θέσει cookies κάτι που είναι ενδεικτικό ότι ένας web server χρησιμοποιείται (όπως ο JSESSIONID που

παρέχεται από κάποιους J2EE servers) ή να ξαναγράψουμε τα URLs αυτόματα για να κάνουμε session ανίχνευση.

Η αυθεντικοποίηση των back-ends (όπως οι LDAP κατάλογοι, σχεσιακές βάσεις δεδομένων ή RADIUS servers) ωστόσο, δεν είναι εύκολο να ανιχνευθούν από μία πιο αποστασιοποιημένη οπτική γωνία με άμεσο τρόπο, από τη στιγμή που θα κρύβονται από την ίδια την εφαρμογή. Η χρήση μιας backend βάσης δεδομένων μπορεί να προσδιοριστεί μόνο με την πλοήγηση στην εφαρμογή. Εάν υπάρχει πολύ δυναμικό περιεχόμενο που έχει δημιουργηθεί ακαριαία είναι πιθανόν να προέρχονται από ένα είδος βάσης δεδομένων της ίδιας της εφαρμογής. Μερικές φορές ο τρόπος που οι πληροφορίες ζητούνται θα μπορούσε να δώσει εικόνα για την ύπαρξη μιας βάσης δεδομένων back-end. Για παράδειγμα, μια on-line εφαρμογή αγορών που χρησιμοποιεί αριθμητικά αναγνωριστικά (id) κατά την περιήγηση διαφορετικών αντικειμένων στο κατάστημα. Ωστόσο, όταν κάνουμε έναν τυφλό έλεγχο εφαρμογής, η γνώση των υποκείμενων βάσεων δεδομένων είναι συνήθως διαθέσιμη μόνο όταν μια ευπάθεια βγαίνει στην επιφάνεια αναφορικά με την εφαρμογή, όπως ο κακός χειρισμός των εξαιρέσεων ή ευαισθησία σε SQL Injection.

Γνωστές ευπάθειες των server

Ευπάθειες που βρέθηκαν σε διάφορα στοιχεία που συνθέτουν την αρχιτεκτονική της εφαρμογής, είτε πρόκειται για τον web server ή την backend βάση δεδομένων, μπορεί να θέσουν σε σοβαρό κίνδυνο την ίδια την εφαρμογή. Για παράδειγμα, ας σκεφτούμε μία ευπάθεια ενός server που επιτρέπει έναν απομακρυσμένο, μη-αυθεντικοποιημένο χρήστη να ανεβάζει αρχεία στον web server ή ακόμη και να αντικαθιστά αρχεία. Αυτή η ευπάθεια θα μπορούσε να προκαλέσει μεγάλο κίνδυνο στην εφαρμογή, δεδομένου ότι ένας αδίστακτος χρήστη θα μπορούσε να αντικαταστήσει ολόκληρη την εφαρμογή ή να εισάγει έναν κωδικό που θα μπορούσε να επηρεάσει τους backend servers, όπως έναν κωδικό εφαρμογής που θα έτρεχε όπως οποιαδήποτε άλλη εφαρμογή.

Η επανεξέταση των τρωτών σημείων του server μπορεί να αποδειχθεί δύσκολη εάν οι έλεγχοι πρέπει να γίνουν μέσω μιας τυφλής δοκιμής διείσδυσης. Σε αυτές τις περιπτώσεις, οι ευπάθειες πρέπει να δοκιμαστούν από ένα απομακρυσμένο ιστότοπο, συνήθως χρησιμοποιώντας ένα αυτοματοποιημένο εργαλείο, ωστόσο η δοκιμή ορισμένων τρωτών σημείων μπορεί να έχει απρόβλεπτα αποτελέσματα για τον web server και για άλλα (όπως αυτά που εμπλέκονται άμεσα με επιθέσεις άρνησης υπηρεσιών) η δοκιμή να μην είναι δυνατή εξαιτίας του χρόνου διακοπής της λειτουργίας της υπηρεσίας που εμπλέκεται εφόσον η δοκιμή ήταν επιτυχημένη. Επίσης κάποια αυτοματοποιημένα εργαλεία θα επισημάνουν κάποιες ευπάθειες βασισμένες στην έκδοση του web serverr. Αυτό οδηγεί σε δύο ψευδώς θετικά και δύο ψευδώς αρνητικά αποτελέσματα: Από τη μία, αν η έκδοση του web server έχει αφαιρεθεί ή αποκρύπτεται από τον τοπικό ιστότοπο του διαχειριστή, το εργαλείο σάρωσης δεν θα σημειώσει τον server σαν ευπαθή ακόμη κι αν όντως είναι. Από την άλλη, αν ο πωλητής που παρέχει το λογισμικό δεν ενημερώσει(update) την έκδοση του server όταν τα τρωτά του σημεία έχουν διορθωθεί σε αυτή το εργαλείο σάρωσης θα σημειώσει ευπάθειες που δεν υπάρχουν.

Ειδικά η τελευταία περίπτωση είναι πολύ κοινή σε κάποιους προμηθευτές λειτουργικών συστημάτων που ενώ τοποθετούν τα back-port patches των ευπαθειών ασφάλειας στο λογισμικό το οποίο παρέχουν στα λειτουργικά συστήματα αλλά δεν κάνουν μια πλήρη μεταφορά(upload) στην τελευταία έκδοση του λογισμικού. Αυτό συμβαίνει στις περισσότερες GNU/Linux διανομές όπως οι Debian, Red Hat ή SuSE. Στις περισσότερες περιπτώσεις, η σάρωση των ευπαθειών της αρχιτεκτονικής μιας εφαρμογής εντοπίζει μόνο ευπάθειες που σχετίζονται με τα στοιχεία της αρχιτεκτονικής που είναι εκτεθειμένα (όπως ο web server) και είναι δεν συνήθως ικανή να βρει τις ευπάθειες των στοιχείων που σχετίζονται με στοιχεία που δεν είναι άμεσα εκτεθειμένα, όπως οι back-ends αυθεντικοποίησης, οι back-ends βάσεων δεδομένων ή οι reserve proxies που χρησιμοποιούνται.

Τέλος, δεν θα αποκαλύψουν όλοι οι προμηθευτές τις ευπάθειες δημόσια και ως εκ τούτου αυτές οι αδυναμίες δεν θα γίνουν γνωστές με τη ευρύτερη έννοια των ευπαθειών βάσεων δεδομένων. Αυτές οι πληροφορίες είναι διαθέσιμες μόνο σε πελάτες ή δημοσιεύονται μέσω διορθώσεων που δεν παρέχουν συνοδευτικές συμβουλές. Αυτό μειώνει την χρησιμότητα των εργαλείων σάρωσης ευπαθειών. Τυπικά, η κάλυψη των ευπαθειών αυτών των εργαλείων θα είναι μεγάλη για ευρέως χρησιμοποιούμενα προϊόντα (όπως ο Apache server, ο server πληροφοριών ιστού της Microsoft ή ο Lotus Domino της IBM) αλλά υστερεί σε λιγότερο γνωστά προϊόντα. Αυτός είναι ο λόγος που η επανεξέταση των ευπαθειών γίνεται με τον καλύτερο τρόπο όταν ο ελεγκτής είναι εφοδιασμένος με εσωτερικές πληροφορίες του λογισμικού που χρησιμοποιείται, συμπεριλαμβανομένων των εκδόσεων και των patches που εφαρμόζονται στο λογισμικό. Με αυτή την πληροφόρηση ο ελεγκτής μπορεί να ανακτήσει πληροφορίες ο ίδιος από τον προμηθευτή και να αναλύσει τι τρωτά σημεία εμφανίζονται στην αρχιτεκτονική και πως μπορούν να επηρεάσουν την ίδια την εφαρμογή. Όταν είναι δυνατόν, αυτά τα τρωτά σημεία μπορούν να ελεγχθούν με σκοπό να καθοριστούν οι πραγματικές επιπτώσεις τους και να εντοπιστεί αν υπάρχει οποιοδήποτε εξωτερικό στοιχείο (όπως η ανίχνευση εισβολών ή συστήματα πρόληψης) που μπορεί να περιορισθεί ή να εξαλειφθεί η πιθανότητα επιτυχούς αξιοποίησής τους. Οι ελεγκτές μπορούν ακόμη να προσδιορίσουν, μέσω μιας επανεξέτασης της διαμόρφωσης, ότι η ευπάθεια δεν υπάρχει καν αφού επηρεάζει ένα στοιχείο του λογισμικού που δεν χρησιμοποιείται.

Αξίζει επίσης να σημειωθεί ότι οι προμηθευτές μερικές φορές θα διορθώσουν σιωπηρά τις ευπάθειες και θα τις κάνουν διαθέσιμες μέσω νέων εκδόσεων λογισμικού. Διαφορετικοί προμηθευτές θα έχουν διαφορετικούς κύκλους εκδόσεων και θα καθορίσουν την υποστήριξη που θα παρέχουν για παλιότερες εκδόσεις. Ένας ελεγκτής με λεπτομερείς πληροφορίες για τις εκδόσεις λογισμικού που χρησιμοποιεί η αρχιτεκτονική, μπορεί να αναλύσει το κίνδυνο που σχετίζεται με τη χρήση παλιότερων εκδόσεων λογισμικού που μπορεί να μην υποστηρίζονται είτε τη στιγμή που γίνεται ο έλεγχος είτε σε λίγο χρονικό διάστημα. Αυτό είναι κρίσιμο, αφού αν η ευπάθεια που έχει έρθει στην επιφάνεια μιας παλιάς έκδοσης πλέον δεν υποστηρίζεται το προσωπικό των συστημάτων μπορεί να μην την γνωρίζει άμεσα. Κανένα patch δε θα είναι ποτέ διαθέσιμο για αυτή και η έκδοση αυτή δε θα σημειωθεί και στην λίστα σαν ευπαθής (αφού είναι μη-υποστηριζόμενη). Ακόμη και στην περίπτωση που είναι που γίνει γνωστό ότι η ευπάθεια είναι παρούσα και το σύστημα είναι, πράγματι, ευπαθές θα χρειαστεί να γίνει μια πλήρης αναβάθμιση σε μια νέα έκδοση λογισμικού το οποίο μπορεί να επιφέρει σημαντικές διακοπές

στην αρχιτεκτονική της εφαρμογής ή μπορεί να αναγκάσει την εφαρμογή να επανεξετάσει τον κώδικά της λόγω ασυμβατότητας με την πιο πρόσφατη έκδοση λογισμικού.

Διαχειριστικά Εργαλεία

Κάθε υποδομή του web server απαιτεί την ύπαρξη διαχειριστικών εργαλείων για την συντήρηση(maintain) και την ενημέρωση(update) των πληροφοριών που χρησιμοποιούνται από την εφαρμογή: στατικό περιεχόμενο (ιστοσελίδες, αρχεία γραφικών), πηγαίος κώδικας εφαρμογής, βάσεις δεδομένων αυθεντικοποίησης χρηστών κλπ. Ανάλογα με τη με τον ιστότοπο, την τεχνολογία ή το λογισμικό που χρησιμοποιείται, τα διαχειριστικά εργαλεία θα διαφέρουν. Για παράδειγμα, κάποιοι web servers θα διαχειρίζονται χρησιμοποιώντας διαχειριστικές διεπαφές οι οποίες είναι, οι ίδιες, web servers (όπως ο iPlanet web server) ή θα διαχειρίζονται από διαχειριστικά αρχεία απλού κειμένου (στην περίπτωση του Apache) ή θα χρησιμοποιούν GUI εργαλεία λειτουργικών συστημάτων (όταν χρησιμοποιούν τον Microsoft IIS server ή ASP.Net). Στις περισσότερες περιπτώσεις, ωστόσο, ο χειρισμός της διαμόρφωσης του server θα γίνεται χρησιμοποιώντας διαφορετικά εργαλεία από ότι η συντήρηση των αρχείων που γίνεται από τον web server, τα οποία διαχειρίζονται μέσω FTP servers, WebDAV, συστήματα αρχείων δικτύου (NFS, CIFS) ή άλλους μηχανισμούς. Προφανώς, το λειτουργικό σύστημα των στοιχείων, Που συνθέτουν την αρχιτεκτονική της εφαρμογής θα διαχειρίζονται επίσης από άλλα εργαλεία. Οι εφαρμογές μπορούν επίσης να έχουν διαχειριστικές διεπαφές ενσωματωμένες μέσα σε αυτές που χρησιμοποιούνται για να διαχειριστούμε τα δεδομένα της ίδιας της εφαρμογής (χρήστες, περιεχόμενο, κλπ)

Η επανεξέταση των διαχειριστικών διεπαφών που χρησιμοποιούνται για να διαχειριστούμε τα διαφορετικά μέρη της αρχιτεκτονικής είναι πολύ σημαντική, δεδομένου ότι αν ένας επιτιθέμενος αποκτήσει την πρόσβαση σε ένα από αυτά μπορεί να θέσει σε κίνδυνο ή να προκαλέσει ζημιά στην αρχιτεκτονική της εφαρμογής. Έτσι είναι σημαντικό να:

- Συγκεντρώσουμε σε λίστα όλες τις πιθανές διαχειριστικές διεπαφές
- Προσδιορίσουμε εάν οι διαχειριστικές διεπαφές είναι διαθέσιμες από εσωτερικό δίκτυο ή είναι επίσης διαθέσιμες από το διαδίκτυο.
- Εάν οι διεπαφές αυτές είναι διαθέσιμες μέσω διαδικτύου να προσδιοριστούν οι μηχανισμοί που ελέγχουν την πρόσβαση σε αυτές τις διεπαφές και οι συναφείς ευαισθησίες τους.
- Αλλάξουμε τον προ-επιλέγμενο όνομα χρήστη και τον κωδικό πρόσβασης

Ορισμένες εταιρείες επιλέγουν να μη διαχειριστούν όλες τις πτυχές των web server των εφαρμογών, αλλά μπορεί να έχουν άλλες ομάδες(εταιρείες) για τη διαχείριση του περιεχομένου του web server. Αυτές μπορούν είτε να παρέχουν μόνο μέρη του περιεχομένου (νέες ενημερώσεις ή προωθήσεις) ή μπορεί να διαχειρίζονται πλήρως τον web server (συμπεριλαμβανομένου του περιεχομένου και του πηγαίου κώδικα).Είναι συνηθισμένο να βρίσκουμε διαχειριστικές διεπαφές διαθέσιμες μέσω του διαδικτύου σε αυτές τις περιπτώσεις, δεδομένου ότι είναι φθηνότερο από το να παρέχουμε μια ειδική γραμμή που συνδέει την εξωτερική εταιρεία με την αρχιτεκτονική της εφαρμογής μέσω μίας

διαχειριστικής διεπαφής. Σε αυτή την κατάσταση είναι πολύ σημαντικό να ελέγξουμε αν οι διαχειριστικές διεπαφές μπορούν να είναι ευάλωτες σε επιθέσεις.

3.3.4 Application Configuration Management Testing

Οι Web εφαρμογές αποκρύπτουν κάποιες πληροφορίες που συνήθως δε λαμβάνονται υπόψιν κατά την ανάπτυξη ή τη διαμόρφωση της εφαρμογής. Αυτά τα δεδομένα μπορούν να ανακαλυφθούν στον πηγαίο κώδικα, στα αρχεία καταγραφής(log files) ή στους προ-επιλεγμένους κωδικούς σφαλμάτων των web servers. Μια σωστή προσέγγιση για το θέμα αυτό έχει θεμελιώδη σημασία κατά τη διάρκεια μιας αξιολόγησης ασφάλειας. Η επανεξέταση και ο έλεγχος της διαμόρφωσης είναι ένα κρίσιμο θέμα για τη δημιουργία και συντήρηση της αρχιτεκτονικής από τη στιγμή που πολλά διαφορετικά συστήματα θα παρέχονται συνήθως με γενική διαμόρφωση που μπορεί να μην είναι κατάλληλη για το έργο που εκτελούν στο συγκεκριμένο ιστότοπο που έχουν εγκατασταθεί. Ενώ η τυπική εγκατάσταση του web server και του server της εφαρμογής θα σημειώσει πολλές λειτουργίες (όπως παραδείγματα εφαρμογών, τεκμηρίωση, δοκιμαστικές σελίδες), οτιδήποτε δεν είναι απαραίτητο θα πρέπει να αφαιρεθεί για να αποφύγουμε την εκμετάλλευση της μετεγκατάστασης.

BLACK BOX TESTING

Αρχεία δειγμάτων και γνωστά αρχεία και κατάλογοι

Πολλοί web servers και servers εφαρμογών παρέχουν, μια προκαθορισμένη εγκατάσταση, δοκιμαστικές εφαρμογές και αρχεία που παρέχονται προς όφελος αυτού που αναπτύσσει την εφαρμογή και με σκοπό να ελέγξουμε εάν ο server λειτουργεί αμέσως μετά την εγκατάσταση. Ωστόσο, πολλές προκαθορισμένες εφαρμογές των web server έγιναν αργότερα γνωστές ως ευπαθείς. Αυτό συνέβη, για παράδειγμα, στον CVE1999-0449 (Άρνηση παροχής υπηρεσιών σε IIS όταν το δείγμα του Exair ιστότοπου είχε εγκαταστάθηκε), στον CAN-2002-1744 (ευπάθεια διασταύρωσης καταλόγων στο CodeBrws.asp του Microsoft IIS 5.0), στον CAN-2002-1630 (χρήση του sendmail.jsp στον Oracle 9iAS) ή στον CAN-2003-1172 (διασταύρωση καταλόγων στο δείγμα εμφάνισης κώδικα στον Apache Cocoon). Οι CGI σαρωτές περιέχουν μία λεπτομερή λίστα με γνωστά αρχεία και καταλόγους δειγμάτων που παρέχονται από διαφορετικούς web server ή server εφαρμογής και μπορεί να είναι ένας εύκολος τρόπος να αποφασίσουμε εάν υπάρχουν αυτά τα αρχεία. Ωστόσο, ο μόνος τρόπος να είμαστε πράγματι σίγουροι είναι να κάνουμε μια πλήρη επανεξέταση των περιεχομένων του web server και/ή του server εφαρμογής και να αποφασίσουμε κατά πόσο έχουν σχέση με την ίδια την εφαρμογή ή όχι.

Επανεξέταση των Σχόλιων της ανάπτυξης

Είναι πολύ κοινό, και συνίσταται ακόμη, για τους προγραμματιστές να συμπεριλάβουν τα σχόλιά τους στον πηγαίο κώδικα προκειμένου να δώσουν την ευκαιρία σε άλλους προγραμματιστές να κατανοήσουν καλύτερα γιατί μια συγκεκριμένη απόφαση ελήφθη για την κωδικοποίηση μιας συγκεκριμένης λειτουργίας. Οι προγραμματιστές συνήθως το κάνουν όταν αναπτύσσουν μεγάλες

εφαρμογές ιστού. Ωστόσο, τα σχόλια που περιλαμβάνονται στον ενσωματωμένο κώδικα HTML θα μπορούσαν να αποκαλύψουν σε έναν επιτιθέμενο εσωτερικές πληροφορίες που δε θα έπρεπε να είναι διαθέσιμες σε αυτόν. Κάποιες φορές ακόμη και ο πηγαίος κώδικας είναι σε σχόλια και αφορά μια λειτουργία που πλέον δεν είναι απαραίτητη αλλά αυτό το σχόλιο έχει διαρρεύσει στις HTML σελίδες επιστρέφοντας στους χρήστες ακούσια. Σχόλια επανεξέτασης θα πρέπει να γίνονται προκειμένου να διαπιστωθεί εάν κάποιες πληροφορίες διέρρευσαν από τα σχόλια. Αυτή η επανεξέταση μπορεί να γίνει μόνο μέσω μιας ανάλυσης του στατικού και δυναμικού περιεχομένου του web server και μέσα από αναζητήσεις αρχείων. Θα ήταν χρήσιμο, ωστόσο, να περιηγηθούμε στον ιστότοπο είτε αυτόματα είτε χρησιμοποιώντας έναν οδηγό(guide) και να αποθηκεύσουμε όλο το περιεχόμενο που θα ανακτηθεί. Το περιεχόμενο που ανακτάται μπορεί στη συνέχεια να ερευνηθεί προκειμένου να αναλύσουμε τα σχόλια της HTML, εάν υπάρχουν, στον κώδικα.

3.3.5 Testing for File Extensions Handling

Οι επεκτάσεις των αρχείων χρησιμοποιούνται στους web servers για να προσδιοριστούν εύκολα ποιές τεχνολογίες/γλώσσες/plug-ins πρέπει να χρησιμοποιήσουμε για να εκπληρώσουμε το αίτημα του web server. Ενώ αυτή η διαδικασία είναι συνεπής με τα RFC's και στα πρότυπα του παγκόσμιου ιστού, η χρήση των πρότυπων επεκτάσεων των αρχείων παρέχει στον ελεγκτή χρήσιμες πληροφορίες για τις υπάρχουσες τεχνολογίες που χρησιμοποιούνται σε μια δικτυακή συσκευή και σε μεγάλο βαθμό απλοποιεί το έργο του προσδιορισμού του σεναρίου επίθεσης που θα χρησιμοποιηθούν σε ιδιαίτερες τεχνολογίες. Επιπλέον, αυτή η λανθασμένη διαμόρφωση στους web servers μπορεί εύκολα να αποκαλύψει εμπιστευτικές πληροφορίες σχετικά με στοιχεία πρόσβασης**.**

Η απόφαση για το πως οι web servers θα χειρίζονται τα αιτήματα που αντιστοιχούν σε προγράμματα που έχουν διαφορετικές εκδόσεις, μπορεί να βοηθήσει στο να καταλάβουμε τη συμπεριφορά των web server ανάλογα με το είδος των αρχείων που προσπαθούμε να προσπελάσουμε. Για παράδειγμα, μπορεί να βοηθήσει να καταλάβουμε ποιες επεκτάσεις αρχείων επιστρέφονται σε κάποιο κείμενο σε σχέση με εκείνες που προκαλούν εκτέλεση στην πλευρά του web server. Οι τελευταίες είναι ενδεικτικές των τεχνολογιών/γλωσσών/plugins που χρησιμοποιούνται από τους web servers ή servers εφαρμογών και μπορεί να παρέχουν πρόσθετες γνώσεις σχετικά με το πως η διαδικτυακή είναι σχεδιασμένη. Για παράδειγμα μία επέκταση ".pl" σχετίζεται συνήθως με την πλευρά του server και την υποστήριξη της Perl (αν και η επέκταση του αρχείου από μόνη της μπορεί να είναι παραπλανητική και όχι απολύτως πειστική, πχ οι πόροι του Perl server μπορεί να έχουν μετονομαστεί για να αποκρύψουν το γεγονός ότι σχετίζονται με την γλώσσα Perl).

BLACK BOX TESTING

Ακολουθώντας παραδείγματα black box για τον χειρισμό των επεκτάσεων των αρχείων θα ήταν να υποβάλουμε http[s] αιτήματα που περιλαμβάνουν διαφορετικές επεκτάσεις αρχείων και να

επαληθεύσουμε τον τρόπο που αντιμετωπίζονται. Οι επαληθεύσεις αυτές θα πρέπει να είναι για κάθε περίπτωση χωριστά σε έναν δικτυακό κατάλογο. Πρέπει να βεβαιωθούμε για τους καταλόγους που επιτρέπουν την εκτέλεση scripts. Οι κατάλογοι των web server μπορούν να προσδιοριστούν από σαρωτές ευπαθειών, που αναζητούν την παρουσία γνωστών καταλόγων. Επιπλέον, η αντανάκλαση της αρχιτεκτονικής του web site επιτρέπει την ανακατασκευή του δέντρου των καταλόγων ιστού που χρησιμοποιούνται από την εφαρμογή. Σε περίπτωση που η αρχιτεκτονική της web εφαρμογής είναι load-balanced είναι σημαντικό να αξιολογηθεί το σύνολο των web server. Αυτό μπορεί να είναι εύκολο ή όχι ανάλογα με τη διαμόρφωση της υποδομής εξισορρόπησης. Σε μια υποδομή με πλεονάζοντα στοιχεία μπορεί να υπάρχουν μικρές διαφοροποιήσεις στη διαμόρφωση ενός ξεχωριστού web server ή web εφαρμογής. Αυτό μπορεί να συμβεί για παράδειγμα εάν η web αρχιτεκτονική απασχολεί ετερογενείς τεχνολογίες (σκεφτείτε ένα σύνολο από IIS και Apache web servers σε μια load-balancing διαμόρφωση, το οποίο μπορεί να εισάγει μια μικρή ασύμμετρη συμπεριφορά μεταξύ τους και πιθανώς διαφορετικές ευπάθειες).

Παράδειγμα: Έχουμε αναγνωρίσει την ύπαρξη ενός αρχείου που ονομάζεται connection.inc. Επιχειρώντας να αποκτήσουμε άμεση πρόσβαση σε αυτό επιστρέφει το περιεχόμενο του το οποίο είναι:

```
<?    mysql_connect("127.0.0.1", "root", "")      or die("Could not connect");   ?>
```

Έχουμε διαπιστώσει την ύπαρξη ενός back-end συστήματος βάσης δεδομένων της MySQL και τα αδύναμα credentials (όνομα χρήστη /κωδικός πρόσβασης) που χρησιμοποιούνται από την εφαρμογή ιστού για την πρόσβαση σε αυτή. Αυτό το παράδειγμα (το οποίο συνέβη σε πραγματικό σενάριο αποτίμησης ασφάλειας) δείχνει πόσο επικίνδυνη μπορεί να είναι η πρόσβαση σε κάποια είδη αρχείων. Οι επεκτάσεις/καταλήξεις των αρχείων που ακολουθούν παρακάτω δεν πρέπει ποτέ να επιστρέφονται από έναν web server, δεδομένου ότι σχετίζονται με αρχεία που περιέχουν ευαίσθητες πληροφορίες ή με αρχεία για τα οποία δεν υπάρχει λόγος να είναι διαθέσιμα.

- *.asa*
- *.inc*

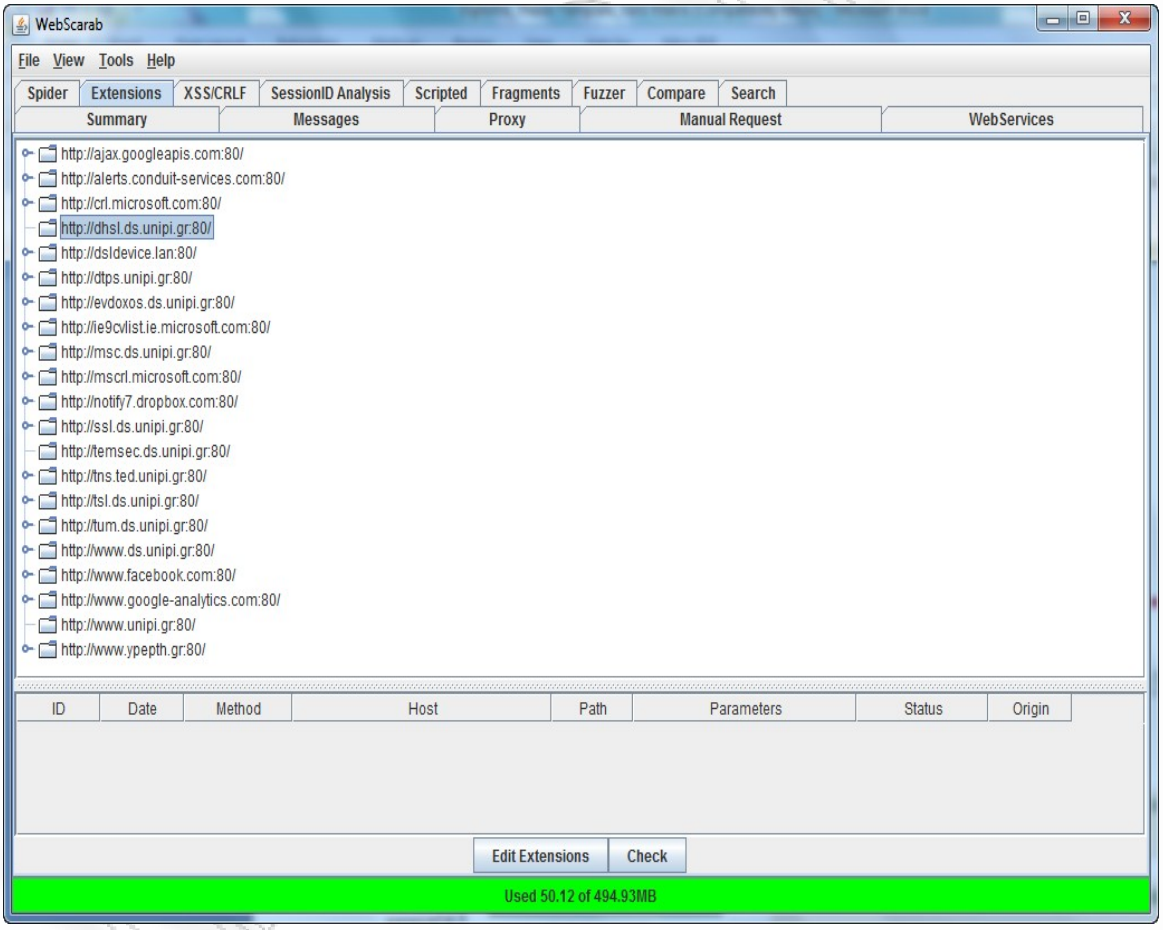
Οι επεκτάσεις αρχείων που ακολουθούν σχετίζονται με αρχεία που, όταν προσπελάζονται, είτε εμφανίζουν το περιεχόμενό τους είτε "κατεβαίνουν"(download) από τον browser. Ως εκ τούτου, τα αρχεία με αυτές τις καταλήξεις θα πρέπει να ελέγχονται για να επαληθεύσουμε ότι πράγματι πρέπει να εξυπηρετούνται από τον web server (δεν είναι αρχεία ξεχασμένα) και δεν περιέχουν ευαίσθητες πληροφορίες.

- *.zip, .tar, .gz, .tgz, .rar: Συμπιεσμένα αρχεία*
- *.java: Δεν υπάρχει λόγος να παρέχουμε πρόσβαση σε αρχεία πηγαίου κώδικα Java.*
- *.txt: Αρχεία κειμένου*
- *.pdf: Αρχεία PDF*
- *.doc, .rtf, .xls, .ppt: Αρχεία του Office.*

- *.bak, .old and και άλλες επεκτάσεις που είναι ενδεικτικές για αρχεία backup.*

Η παραπάνω λίστα δίνει λεπτομερή περιγραφεί για μερικά παραδείγματα μιας και οι επεκτάσεις των αρχείων είναι πάρα πολλές για να περιγραφούν ολοκληρωτικά σε αυτή την εργασία. Μπορούμε να περιηγηθούμε στον ιστότοπο http://filext.com/ για μια μεγαλύτερη βάση δεδομένων με επεκτάσεις αρχείων. Ολοκληρώνοντας, προκειμένου να αναγνωρίσουμε αρχεία που έχουν κάποιες συγκεκριμένες καταλήξεις μια πλειάδα τεχνικών μπορεί να ακολουθηθούν που περιλαμβάνει: σαρωτές ευπαθειών, εργαλεία spidering και mirroring, χειροκίνητος έλεγχος της εφαρμογής(αυτό ξεπερνά τα όρια του αυτόματου spidering), αιτήματα σε μηχανές αναζήτησης.

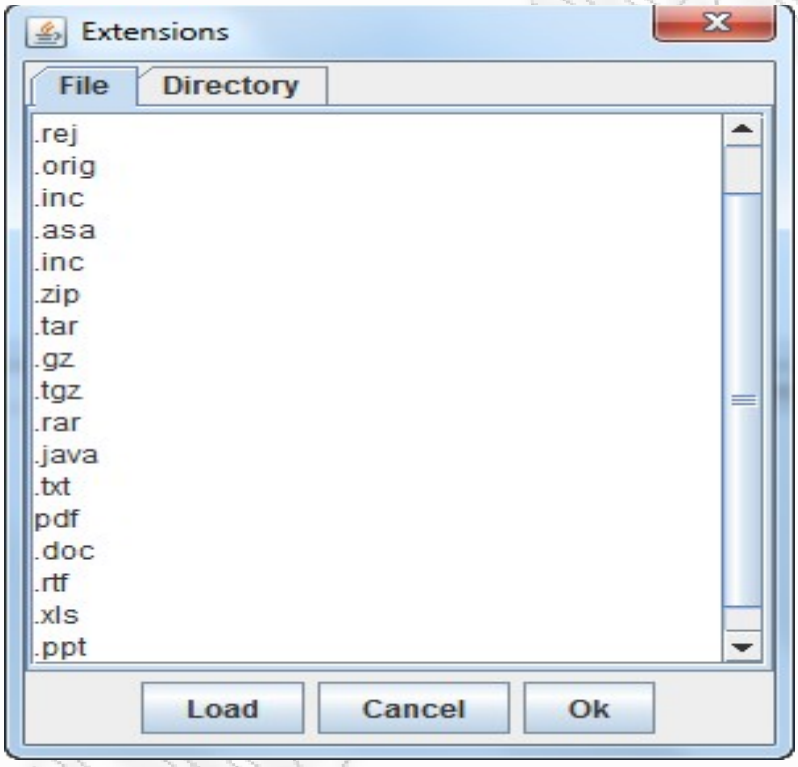
Στην περίπτωση του ιστότοπου που εξετάζουμε στην παρούσα εργασία θα χρησιμοποιήσουμε το εργαλείο WebScarab για να μας βοηθήσει να ανακαλύψουμε ποιες καταλήξεις επιστρέφονται από τον server της εφαρμογής. Συγκεκριμένα χρησιμοποιούμε το plug-in "Extensions" η εικόνα του οποίου φαίνεται παρακάτω.



Εικόνα : **Το plug-in "Extensions" για το www.ds.unipi.gr.**

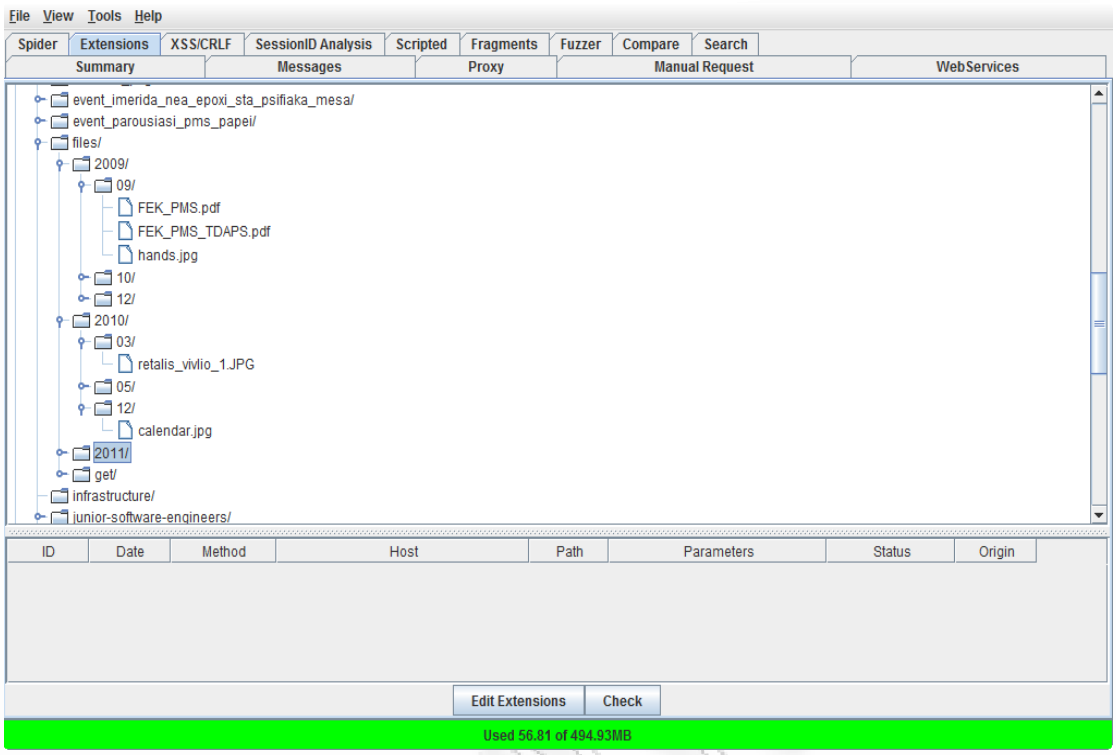
Στο plug-in αυτό έχουμε τη δυνατότητα με τη χρήση του κουμπιού "Edit Extensions" να αναζητήσουμε συγκεκριμένες καταλήξεις αρχείων που εμείς θα ορίσουμε. Ορίσαμε λοιπόν τις παρακάτω κατηγορίες καταλήξεων όπως φαίνεται και στην εικόνα που ακολουθεί.

- .asa
- .inc
- .zip, .tar, .gz, .tgz, .rar
- .java
- .txt
- .pdf
- .doc, .rtf, .xls, .ppt
- .bak, .old



Εικόνα : **Ορισμός καταλήξεων στο Plug-in "Extensions" του εργαλείου WebScarab.**

Αφού ανατρέξαμε, λοιπόν όλο το δικτυακό δέντρο που αφορά το περιεχόμενο του ιστότοπου που εξετάζουμε διαπιστώσαμε πως δεν επιστρέφονται από την εφαρμογή καταλήξεις ευπαθείς για την ίδια την εφαρμογή με την έννοια ότι μπορεί κάποιος να τις εκμεταλλευτεί. Εξαίρεση βέβαια αποτελούν κάποιες περιπτώσεις αρχείων με κατάληξη .pdf που ανακαλύψαμε ότι επιστρέφονται από τον server. Στην εικόνα που ακολουθεί φαίνεται το δικτυακό δέντρο όπως το ανατρέχουμε ψάχνοντας για συγκεκριμένες καταλήξεις αρχείων μέσα στην εφαρμογή.



Εικόνα : **Αναζήτηση αρχείων με συγκεκριμένες καταλήξεις που επιστρέφονται από τον server.**

3.3.6 Testing for old, backup and unreferenced files

Ενώ οι περισσότεροι web servers χειρίζονται απευθείας όλα τα αρχεία τους οι ίδιοι, δεν είναι ασυνήθιστο να συναντήσουμε μη αναφερόμενα και/ή ξεχασμένα αρχεία τα οποία μπορούν να χρησιμοποιηθούν για να ανακτήσουμε χρήσιμες πληροφορίες είτε για την υποδομή είτε για τα συνθηματικά(credentials). Το πιο συνηθισμένο σενάριο περιλαμβάνει την ύπαρξη μετονομασμένων ή παλιότερων εκδόσεων τροποποιημένων αρχείων, αρχείων περίληψης που φορτώνονται στη γλώσσα της επιλογής και μπορούν να κατέβουν σαν πηγαίος κώδικας ή ακόμα και αυτόματα ή χειροκίνητα backups στη μορφή συμπιεσμένων αρχείων. Όλα αυτά τα αρχεία μπορούν να χορηγήσουν πρόσβαση στον ελεγκτή στις εσωτερικές λειτουργίες, πίσω από τις πόρτες(θύρες), τις διεπαφές του διαχειριστή ή ακόμα και σε credentials ώστε να συνδεθεί στη διεπαφή του διαχειριστή ή τον server της βάσης δεδομένων.

Μια σημαντική πηγή ευπαθειών υφίσταται στα αρχεία που δεν έχουν να κάνουν με την εφαρμογή, αλλά δημιουργήθηκαν ως συνέπεια της επεξεργασίας των αρχείων της εφαρμογής ή μετά από τη ακαριαία δημιουργία αρχείων backup ή αφήνοντας στο διαδικτυακό δέντρο παλιά αρχεία ή μη αναφερόμενα αρχεία. Η εκτέλεση της επί τόπου επεξεργασίας ή άλλων διαχειριστικών πράξεων της παραγωγής των web server μπορεί, σαν συνέπεια, να αφήσει κατά λάθος αντίγραφα backup (είτε που δημιουργούνται αυτόματα από τον επεξεργαστή κατά την επεξεργασία των αρχείων ή από τον διαχειριστή που συμπιέζει ένα σύνολο αρχείων για να δημιουργήσει ένα backup).

Είναι ιδιαίτερα εύκολο να ξεχάσουμε αυτά τα αρχεία και αυτό μπορεί να αποτελέσει μια σοβαρή απειλή ασφάλειας για την εφαρμογή. Αυτό συμβαίνει επειδή τα αντίγραφα μπορούν να δημιουργηθούν με

τις επεκτάσεις των αρχείων να διαφέρουν από αυτές που έχουν τα πραγματικά αρχεία. Μια κατάληξη .tar, .zip or .gz ενός αρχείου που δημιουργούμε (και ξεχνάμε) έχει μια διαφορετική κατάληξη και το ίδιο συμβαίνει με τα αυτόματα αντίγραφα που δημιουργούν οι επεξεργαστές. Η δημιουργία αντιγράφου με το χέρι μπορεί να έχει επίσης το ίδιο αποτέλεσμα(πχ αντιγραφή του αρχείου file σε file.old). Ως αποτέλεσμα, αυτές οι ενέργειες δημιουργούν αρχεία που:

- δεν χρειάζονται στην εφαρμογή
- μπορεί ο server να τα χειρίζεται διαφορετικά από ότι πραγματικό αρχείο.

Για παράδειγμα, εάν κάνουμε ένα αντίγραφο του login.asp και το ονομάσουμε login.asp.old, επιτρέπουμε στους χρήστες να κατεβάσουν τον πηγαίο κώδικα του login.asp. Αυτό συμβαίνει γιατί εξαιτίας της επέκτασης το login.asp.old θα προσφέρεται σε ένα αρχείο κειμένου παρά θα εκτελεστεί. Με άλλα λόγια, η πρόσβαση στο login.asp προκαλεί την εκτέλεση του κώδικα του login.asp στην πλευρά του server, ενώ η πρόσβαση στο login.asp.old προκαλεί την επιστροφή του περιεχομένου του αρχείου login.asp.old (το οποίο και αυτό είναι κώδικας που εκτελείται στην πλευρά του server στο χρήστη) και εμφανίζονται στον browser. Αυτό μπορεί να ενέχει κινδύνους ασφάλειας από τη στιγμή που αποκαλύπτεται ευαίσθητη πληροφορία. Γενικά, το να εκτίθεται κώδικας από τη πλευρά του server είναι μια άσχημη κατάσταση. Δεν εκτίθεται μόνο άσκοπα η επιχειρηματική λογική που ακολουθείται αλλά μπορεί εν αγνοία μας να αποκαλύπτονται πληροφορίες που σχετίζονται με την εφαρμογή το οποίο μπορεί να βοηθήσει έναν επιτιθέμενο(ονόματα μονοπατιών, δομές δεδομένων κλπ) χωρίς να αναφέρουμε το γεγονός ότι συμπεριλαμβάνονται πολλά scripts με username/password σε αρχεία κειμένου (το οποίο είναι μια απρόσεκτη και επικίνδυνη πρακτική).

Άλλες αιτίες για μη-αναφερόμενα αρχεία οφείλονται στο σχεδιασμό ή τις επιλογές διαμόρφωσης όταν επιτρέπουν διαφορετικού είδους αρχεία που σχετίζονται με την εφαρμογή όπως αρχεία δεδομένων, αρχεία διαμόρφωσης, αρχεία καταγραφής(log files) να αποθηκεύονται σε καταλόγους ενός συστήματος αρχείων που μπορεί να προσπελαστεί από τον web server. Αυτά τα αρχεία λογικά δεν έχουν κανένα λόγο να βρίσκονται στον χώρο ενός συστήματος αρχείου το οποίο μπορεί να προσπελαστεί μέσω του διαδικτύου, εφόσον θα πρέπει να είναι προσπελάσιμα μόνο σε επίπεδο εφαρμογής από την ίδια την εφαρμογή (και όχι από τον κοινό χρήστη που μια περιήγηση μέσω ενός browser).

Απειλές

Τα παλιά, τα μη αναφερόμενα αρχεία και τα αρχεία backup παρουσιάζουν διάφορες απειλές για την ασφάλεια μιας εφαρμογής ιστού.

- Μη αναφερόμενα αρχεία μπορούν να αποκαλύψουν ευαίσθητες πληροφορίες που μπορούν να διευκολύνουν μια επίθεση που στοχεύει στην εφαρμογή. Για παράδειγμα μπορεί να περιλαμβάνουν credentials για τη βάση δεδομένων, αρχεία διαμόρφωσης που περιλαμβάνουν αναφορές σε άλλο κρυμμένο περιεχόμενο, ακριβή μονοπάτια κλπ.

- Μη αναφερόμενες σελίδες μπορεί να περιέχουν ισχυρή λειτουργικότητα η οποία μπορεί να χρησιμοποιηθεί έτσι ώστε να επιτεθούμε στην εφαρμογή. Για παράδειγμα μια σελίδα διαχείρισης που δεν είναι συνδεδεμένη από το δημοσιευμένο περιεχόμενο αλλά μπορεί να προσπελαστεί από οποιονδήποτε χρήστη που ξέρει που μπορεί να τη βρει.
- Παλιά αρχεία και αρχεία backup μπορεί να περιέχουν ευπάθειες που έχουν διορθωθεί σε πρόσφατες εκδόσεις. Για παράδειγμα το αρχείο viewdoc.old.jsp μπορεί να περιέχει μια ευπάθεια διάσχισης καταλόγων η οποία έχει διορθωθεί στο αρχείο viewdoc.jsp όμως μπορεί ακόμη να αξιοποιηθεί από οποιονδήποτε βρίσκει την παλιά έκδοση.
- Τα αρχεία backup μπορούν να αποκαλύψουν τον πηγαίο κώδικα σελίδων που δεν σχεδιάστηκαν να εκτελούνται στον server. Για παράδειγμα, όταν κάνουμε ένα αίτημα για το αρχείο viewdoc.bak αυτό μπορεί να μας επιστρέψει τον πηγαίο κώδικα για το viewdoc.jsp, το οποίο μπορεί να έχει επανεξεταστεί για ευπάθειες που είναι δύσκολο να τις βρούμε στέλνοντας «τυφλά» αιτήματα σε μια εκτελέσιμη σελίδα. Ενώ αυτή η απειλή προφανώς ισχύει για γλώσσες σεναρίου όπως η Perl, η Php, η ASP, τα scripts φλοιού, η JSP κλπ δεν περιορίζεται σε αυτές όπως φαίνεται και στο παράδειγμα που δίνεται παρακάτω.

- Τα backup αρχεία μπορεί να περιέχουν αντίγραφα όλων των αρχείων του web-root. Αυτό επιτρέπει έναν επιτιθέμενο να απαριθμήσει ολόκληρη την εφαρμογή, συμπεριλαμβάνοντας τις μη αναφερόμενες σελίδες, τον πηγαίο κώδικα κλπ Για παράδειγμα, αν ξεχάσουμε ένα αρχείο που ονομάζεται myservlets.jar.old που περιέχει ένα backup αντίγραφο των servlet κλάσεων υλοποίησης τότε εκθέτουμε αρκετές ευαίσθητες πληροφορίες που είναι ευάλωτες στη μεταγλώττιση και την τεχνική της αντιστροφής.
- Σε κάποιες περιπτώσεις η αντιγραφή ή η επεξεργασία αρχείων δεν τροποποιεί την κατάληξη του αρχείου, αλλά το όνομα αρχείου. Αυτό συμβαίνει για παράδειγμα στα παραθυρικά περιβάλλοντα(Windows) όπου η λειτουργία της αντιγραφής αρχείων δημιουργεί ονόματα αρχείων του τύπου “Copy of” ή τοπικές εκδόσεις αυτής της συμβολοσειράς. Από τη στιγμή που η κατάληξη του αρχείου δεν έχει τροποποιηθεί δεν αποτελεί μια περίπτωση όπου ένα εκτελέσιμο αρχείο επιστρέφει ένα αρχείο κειμένου από τον web server και ως εκ τούτου δεν αποτελεί περίπτωση αποκάλυψης του πηγαίου κώδικα. Ωστόσο αυτά τα αρχεία είναι πολύ επικίνδυνα καθώς διότι υπάρχει μια πιθανότητα να περιλαμβάνουν απαρχαιωμένη και εσφαλμένη λογική η οποία όταν επικαλείται θα μπορούσε να προκαλέσει λάθη εφαρμογής που ενδέχεται να αποφέρουν πολύτιμες πληροφορίες για έναν επιτιθέμενο, εάν η οθόνη των διαγνωστικών μηνυμάτων είναι ενεργοποιημένη.
- Τα αρχεία καταγραφής (log files) μπορεί να περιέχουν ευαίσθητη πληροφορία για τις ενέργειες των χρηστών της εφαρμογής, για παράδειγμα ευαίσθητα δεδομένα που περνιούνται στις URL παραμέτρους, στα session id’s, τα URLs που επισκέπτονται (το οποίο μπορεί να κάνει γνωστό πρόσθετο μη αναφερόμενο περιεχόμενο) κλπ. Άλλα αρχεία καταγραφής (πχ ftp logs) μπορεί να περιέχουν ευαίσθητες πληροφορίες για την συντήρηση της εφαρμογής από τους διαχειριστές.

Αντίμετρα

Για την εξασφάλιση μιας αποτελεσματικής στρατηγικής προστασίας, η δοκιμή θα πρέπει να συνδυάζεται με μια πολιτική ασφάλειας η οποία απαγορεύει ρητά επικίνδυνες πρακτικές όπως:

- Επεξεργασία αρχείων στο σύστημα αρχείων της πλευράς του web server/server εφαρμογής. Αυτή είναι μια συγκεκριμένη κακή συνήθεια, δεδομένου ότι είναι πιθανόν να προκαλέσουν άθελά τους τη δημιουργία αρχείων backup από τους συντάκτες. Είναι εκπληκτικό να δούμε το πόσο συχνά συμβαίνει, ακόμα και σε μεγάλους οργανισμούς. Αν πραγματικά υπάρχει η ανάγκη να επεξεργαστούμε αρχεία σε ένα σύστημα παραγωγής, πρέπει να βεβαιωθούμε ότι δεν αφήνουμε τίποτε πίσω που να δεν αποσκοπεί ρητά κάπου και να θεωρήσουμε ότι το κάνουμε με δική μας ευθύνη.
- Ελέγξτε προσεκτικά οποιαδήποτε άλλη δραστηριότητα σε συστήματα αρχείων που εκτίθενται από τον web server, όπως οι δραστηριότητες διαχείρισης. Για παράδειγμα εάν περιστασιακά χρειαστεί να πάρουμε ένα στιγμιότυπο από ένα ζεύγος καταλόγων (κάτι το οποίο δε θα έπρεπε να κάνουμε σε ένα σύστημα παραγωγής), μπορούμε να τούς συμπιέσουμε πρώτα. Πρέπει να προσέξουμε να μην ξεχάσουμε πίσω απο αυτούς αρχεία.
- Οι κατάλληλες πολιτικές διαχείρισης της διαμόρφωσης πρέπει να βοηθούν ώστε να μην αφήνουν απαρχαιωμένα ή μη αναφερόμενα αρχεία.
- Οι εφαρμογές θα πρέπει να σχεδιάζονται έτσι ώστε να μην δημιουργούν (ή να βασίζονται σε) αρχεία που είναι αποθηκευμένα κάτω από τα δέντρα των καταλόγων του web που προσφέρει ο web server. Αρχεία δεδομένων, αρχεία καταγραφής, αρχεία διαμόρφωσης κλπ θα πρέπει να αποθηκεύονται σε καταλόγους μη προσπελάσιμους από τον web server, ώστε να αντιμετωπίζεται το ενδεχόμενο της αποκάλυψης πληροφοριών (για να μην αναφέρουμε τα δεδομένα τροποποίησης αν τα δικαιώματα του web καταλόγου επιτρέπουν το γράψιμο).

BLACK BOX TESTING

Ο έλεγχος για μη αναφερόμενα αρχεία χρησιμοποιεί τόσο αυτοματοποιημένες όσο και χειροκίνητες τεχνικές και τυπικά περιλαμβάνει έναν συνδυασμό από τα ακόλουθα:

1. Συμπέρασμα από το σχήμα ονομασίας που χρησιμοποιείται για το δημοσιευμένο περιεχόμενο: Εάν δεν έχει ήδη γίνει, απαριθμήστε όλες τις σελίδες της εφαρμογής και τη λειτουργικότητα. Αυτό μπορεί να γίνει χειροκίνητα χρησιμοποιώντας έναν browser ή χρησιμοποιώντας ένα εργαλείο spidering της εφαρμογής. Οι περισσότερες εφαρμογές χρησιμοποιούν ένα αναγνωρίσιμο σχήμα ονομασίας και οργανώνουν τους πόρους σε σελίδες και καταλόγους που χρησιμοποιούν λέξεις που περιγράφουν τη λειτουργικότητά τους. Από το σχήμα ονομασίας που χρησιμοποιείται για το δημοσιευμένο περιεχόμενο είναι συχνά δυνατόν να συμπεράνουμε το όνομα και την τοποθεσία του μη αναφερόμενου αρχείου. Για παράδειγμα, αν μια σελίδα viewuser.asp βρεθεί, μετά κοιτάξτε επίσης για τις edituser.asp, adduser.asp και deleteuser.asp. Αν ένας κατάλογος /app/user βρεθεί τότε ψάξτε επίσης για /app/admin και /app/manager.

2. Άλλες ενδείξεις σε δημοσιευμένο περιεχόμενο: Πολλές εφαρμογές ιστού αφήνουν κάποιες ενδείξεις στο δημοσιευμένο περιεχόμενο που μπορούν να οδηγήσουν στην ανακάλυψη κρυφών σελίδων και λειτουργικότητας. Αυτές οι ενδείξεις συνήθως εμφανίζονται στον πηγαίο κώδικα της HTML και στα JavaScript αρχεία. Ο πηγαίος κώδικας για όλο το δημοσιευμένο περιεχόμενο θα πρέπει να επανεξεταστεί χειροκίνητα για να αναγνωρίσει τις ενδείξεις για άλλες σελίδες και λειτουργικότητα. Για παράδειγμα: Τα σχόλια των προγραμματιστών και τα τμήματα του πηγαίου κώδικα που έχουν σχόλια μπορεί να αναφέρονται στο κρυμμένο περιεχόμενο:

```
<!-- <A HREF="uploadfile.jsp">Upload a document to the server</A> -->
<!-- Link removed while bugs in uploadfile.jsp are fixed        -->
```

Η γλώσσα Javascript μπορεί να περιέχει συνδέσμους σελίδων που προσφέρονται με το GUI του χρήστη κάτω από ορισμένες συνθήκες:

```
 var adminUser=false;
:
 if (adminUser) menu.add (new menuItem ("Maintain users", "/admin/useradmin.jsp"));
HTML pages may contain FORMs that have been hidden by disabling the SUBMIT element:
<FORM action="forgotPassword.jsp" method="post">    <INPUT type="hidden" name="userID"
value="123">    <!-- <INPUT type="submit" value="Forgot Password"> -->
</FORM>
```

Μια άλλη πηγή ενδείξεων για μη αναφερόμενους καταλόγους αποτελεί το αρχείο /robots.txt που χρησιμοποιείται για να παρέχει οδηγίες για τα robots ιστού:

```
User-agent: *
Disallow: /Admin
Disallow: /uploads
Disallow: /backup
Disallow: /~jbloggs
Disallow: /include
```

3. Τυφλή εικασία: Στην απλούστερη μορφή του, περιλαμβάνει το τρέξιμο ενός καταλόγου κοινών ονομάτων αρχείων μέσω μιας μηχανής αιτημάτων σε μια προσπάθεια να μαντέψουμε αρχεία και καταλόγους που υπάρχουν στον server. Το ακόλουθο netcat wrapper script θα διαβάσει μια λίστα από stdin και θα εκτελέσει μια βασική επίθεση εικασίας:

```
#!/bin/bash
```

server=www.targetapp.com

port=80

while read url

do

echo -ne "$url\t"

echo -e "GET /$url HTTP/1.0\nHost: $server\n" | netcat $server $port | head -1 done | tee outputfile

Ανάλογα με τον server, το GET μπορεί να αντικατασταθεί με το HEAD για πιο γρήγορα αποτελέσματα. Το προκαθορισμένο αρχείο εξόδου μπορεί να γίνει grep αναζητώντας ενδιαφέροντες κωδικούς απαντήσεων. Ο κωδικός απάντησης 200 (OK) συνήθως δείχνει ότι βρέθηκε ένας έγκυρος πόρος (με την προϋπόθεση ότι δεν έχει επιστρέψει ο server ένα μήνυμα για μια σελίδα που δεν βρέθηκε (not found) χρησιμοποιώντας τον κωδικό 200). Αλλά επίσης πρέπει να προσέξουμε και για τους κωδικούς 301 (Moved), 302 (Found), 401 (Unauthorized), 403 (Forbidden) και 500 (Internal Error), που μπορούν επίσης να υποδείξουν πόρους ή καταλόγους που αξίζουν περαιτέρω διερεύνηση.

Η βασική επίθεση εικασίας θα μπορούσε να "τρέξει" ενάντια στο webroot και επίσης ενάντια σε όλους τους καταλόγους που έχουν αναγνωριστεί μέσω άλλων τεχνικών απαρίθμησης. Οι πιο προχωρημένες/αποτελεσματικές επιθέσεις εικασίας μπορούν να πραγματοποιηθούν ως εξής:

- Αναγνωρίζουμε τις καταλήξεις των αρχείων που χρησιμοποιούνται μέσα σε γνωστές περιοχές της εφαρμογής (πχ jsp, aspx, html) και χρησιμοποιούμε μια βασική λίστα που επισυνάπτεται σε κάθε μία από αυτές τις καταλήξεις (ή χρησιμοποιούμε μια μεγαλύτερη λίστα από κοινές καταλήξεις αν οι πόροι το επιτρέπουν).
- Για κάθε αρχείο που αναγνωρίζεται μέσω μιας τεχνικής απαρίθμησης, δημιουργούμε μια λίστα χρηστών που προκύπτει από αυτό το όνομα αρχείου. Συγκεντρώνουμε μια λίστα με τις κοινές καταλήξεις των αρχείων (συμπεριλαμβάνοντας ~, bak, txt, src, dev, old, inv, origin, copy, tmp, κλπ) και χρησιμοποιούμε κάθε κατάληξη αρχείου πριν, μετά και στη θέση της κατάληξης του πραγματικού αρχείου.

Σημείωση: Οι λειτουργίες αντιγραφής αρχείων των Windows δημιουργούν ονόματα αρχείων με το πρόθεμα «Copy of" ή τοπικές εκδόσεις αυτής της συμβολοσειράς και ως εκ τούτου δεν αλλάζουν κατάληξη αρχείου. Ενώ τα αρχεία "Copy of" τυπικά δεν αποκαλύπτουν πηγαίο κώδικα όταν προσπελάζονται, θα μπορούσαν να αποφέρουν πολύτιμες πληροφορίες στην περίπτωση που προκαλούν σφάλματα όταν καλούνται προς χρήση.

4. Οι πληροφορίες που ανακτώνται μέσω των ευπαθειών και λανθασμένης διαμόρφωσης: Ο πιο προφανής τρόπος με τον οποίο ένας λανθασμένα διαμορφωμένος server μπορεί να αποκαλύψει μη αναφερόμενες σελίδες μέσω μιας λίστας καταλόγων. Αιτούμαστε όλους τους αριθμημένους καταλόγους για να αναγνωρίσουμε οποιονδήποτε παρέχει μια λίστα καταλόγου. Πολλές

ευπάθειες έχουν βρεθεί σε επιμέρους web server οι οποίοι επιτρέπουν σε έναν επιτιθέμενο να απαριθμήσει το μη αναφερόμενο περιεχόμενο, για παράδειγμα:

- Ο ?M=D κατάλογος λίστας ευπαθειών του Apache.
- Διάφορα IIS πηγαίων scripts που αποκαλύπτουν ευπάθειες
- IIS WebDAV τρωτά σημεία λιστών καταλόγου

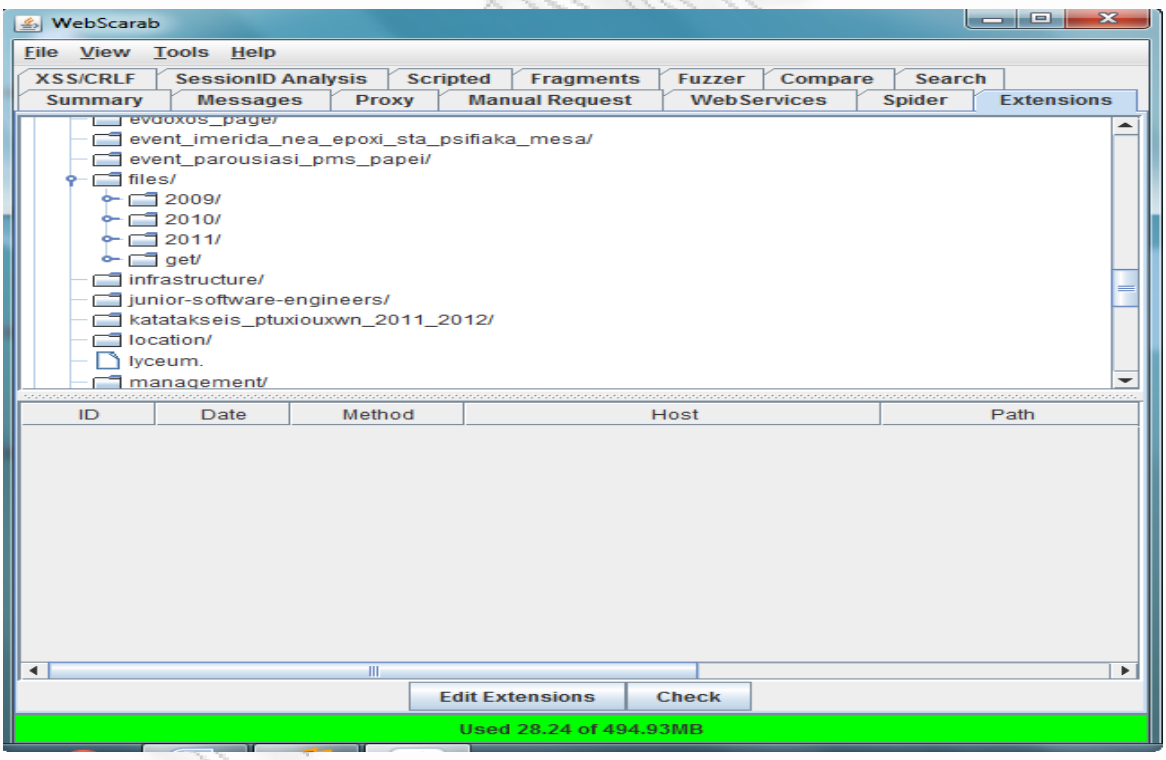
5. Χρήση της δημόσιων διαθέσιμων πληροφοριών: Οι σελίδες και η λειτουργικότητα στις εφαρμογές ιστού που αντιμετωπίζονται διαδικτυακά δεν αναφέρονται μέσα από την ίδια την εφαρμογή μπορεί όμως να υπάρχουν αναφορές από άλλες πηγές domain. Παρακάτω υπάρχουν διάφορες τέτοιες αναφορές:

- Σελίδες που χρησιμοποιούνται για να γίνεται αναφορά μπορεί να εξακολουθούν να εμφανίζονται στα αρχεία των μηχανών αναζήτησης. Για παράδειγμα, το αρχείο 1998results.asp μπορεί να μην συνδέεται μέσω της ιστοσελίδας μιας επιχείρησης, αλλά μπορεί να παραμένει στον server και στις βάσεις δεδομένων της μηχανής αναζήτησης. Αυτό το παλιό script μπορεί να περιέχει ευπάθειες που θα μπορούσαν να χρησιμοποιηθούν για να τεθεί σε κίνδυνο ολόκληρη η ιστοσελίδα. Ο χειριστής αναζήτησης της Google "The site:" μπορεί να χρησιμοποιηθεί ώστε να εκτελέσει ένα αίτημα αναζήτησης μόνο σε σχέση με το domain της επιλογής μας, όπως για παράδειγμα: the site: www.ds.unipi.gr. Η χρησιμοποίηση των μηχανών αναζήτησης κατά αυτόν τον τρόπο έχει οδηγήσει σε ένα ευρύ φάσμα τεχνικών οι οποίες μπορεί να μας φανούν χρήσιμες. Μπορούμε να τις ελέγξουμε ώστε να δούμε τις δυνατότητες που έχουμε μέσω του Google. Τα backup αρχεία δεν αναφέρονται συνήθως από άλλα αρχεία και γι' αυτό δεν περιλαμβάνονται στα αποτελέσματα της Google, αλλά αν βρίσκονται σε καταλόγους που μπορούν να βρεθούν μέσω ενός browser τότε η μηχανή αναζήτησης θα μπορούσε να τα βρει.
- Επιπλέον, η Google και η Yahoo διατηρούν αποθηκευμένες εκδόσεις των σελίδων που βρέθηκαν από τα robots τους. Ακόμη και αν το 1998results.asp έχει αφαιρεθεί από τον server στόχο, μια έκδοση της εξόδου του μπορεί να παραμένει αποθηκευμένη από τις μηχανές αναζήτησης. Οι αποθηκευμένη έκδοση μπορεί να περιέχει αναφορές σε, ή ενδείξεις για, πρόσθετο κρυφό περιεχόμενο το οποίο παραμένει στον server.
- Το περιεχόμενο που δεν αναφέρεται μέσα από μια εφαρμογή στόχο μπορεί να συνδέεται με τους δικτυακούς τόπους τρίτων οντοτήτων. Για παράδειγμα μια εφαρμογή που επεξεργάζεται τις online πληρωμές για λογαριασμό τρίτων εμπορικών οντοτήτων μπορεί να περιέχει μια ποικιλία λειτουργικότητας παραγγελιών που μπορούν να βρεθούν μόνο ακολουθώντας τους συνδέσμους εντός των δικτυακών τόπων των πελατών της.

Στην προσπάθειά μας να μελετήσουμε τι συμβαίνει με τις περιπτώσεις των παλιών και των μη-αναφερόμενων αρχείων καθώς και των backup αρχείων χρησιμοποιήσαμε τόσο το proxy εργαλείο μας (WebScarab), τον ειδικό τελεστή αναζήτησης “the site:” στην μηχανή αναζήτησης καθώς επίσης

εκμεταλλευτήκαμε και κάποια εσωτερική πληροφόρηση για κάποιες πολιτικές που εφαρμόζονται από τους administrator της εφαρμογής.

Εξετάσαμε, λοιπόν, αρχικά μέσω της μηχανής αναζήτησης της google να ελέγξουμε τα αποτελέσματα της αναζήτησης με τη χρήση του τελεστή “the site:”. Διαπιστώνουμε πως δεν επιστρέφονται αρχεία που υπάρχουν στον server παρά μόνο κάποιες σελίδες που αναφέρονται στον ιστότοπο του τμήματος και αφορούν κάποια forums είτε κάποιες άλλες σελίδες πανεπιστημιακού ή εκπαιδευτικού γενικότερα ενδιαφέροντος. Συνεχίζουμε τον έλεγχο χρησιμοποιώντας το WebScarab και ελέγχουμε το διαδικτυακό δέντρο που εμφανίζεται στο plug-in extensions με σκοπό να ανακαλύψουμε καταλήξεις που να αφορούν τέτοια είδη αρχείων που μας ενδιαφέρουν στον έλεγχο αυτό. Από τα αποτελέσματα που προέκυψαν διαπιστώσαμε πως τα παλιά αρχεία του ιστότοπου διατηρούνται και μάλιστα σε αρχεία ξεχωριστά μέσα στον server και είναι πολύ εύκολα αναγνωρίσιμα. Βέβαια το είδος των αρχείων που βρήκαμε δεν είναι ενδιαφέροντα με την έννοια ότι είναι καταλήξεων .jpg ή .pdf όμως φαίνεται πως δεν υπάρχει πολιτική να σβήνονται είτε να διατηρούνται κάπου αλλού τα παλιά αρχεία κάτι που πρέπει να επισημανθεί καθώς στο μέλλον δεν αποκλείεται η αποθήκευση κάποιων αρχείων με μεγαλύτερο “ενδιαφέρον” για τον έλεγχο αυτό. Στην εικόνα που ακολουθεί φαίνεται το δυαδικό δέντρο αναζήτησης και οι φάκελοι με τα παλιά αρχεία.



Εικόνα : **Το δυαδικό δέντρο αναζήτησης του ιστότοπου www.ds.unipi.gr.**

Επιπρόσθετα με την ενημέρωση από τους administrators του ιστότοπου που εξετάζουμε ενημερωθήκαμε πως τα back-up αρχεία διατηρούνται σε ξεχωριστό μέρος και όχι στο server που φιλοξενείται και η εφαρμογή γεγονός που δηλώνει μια σωστή πολιτική back-up.

3.3.7 Infrastructure and Application Admin Interfaces

Πολλές εφαρμογές χρησιμοποιούν ένα κοινό μονοπάτι για τις διεπαφές του διαχειριστή οι οποίες μπορούν να χρησιμοποιηθούν για να μαντέψουμε passwords ή να τα αποκτήσουμε μέσω brute force(επίθεση ωμής βίας). Αυτός ο έλεγχος έχει την τάση να βρίσκει τις διεπαφές του διαχειριστή και να καταλαβαίνει αν είναι δυνατόν να την εκμεταλλευτεί για να αποκτήσει πρόσβαση στη λειτουργικότητα που έχει ο διαχειριστής. Οι διαχειριστικές διεπαφές μπορούν να υπάρχουν στην εφαρμογή ή στον server της εφαρμογής ώστε να επιτρέπει σε ορισμένους χρήστες να αναλάβουν κάποιες εξουσιοδοτημένες λειτουργίες στον ιστότοπο. Οι δοκιμές θα πρέπει να γίνουν για να αποκαλύψουν εάν και πως αυτές οι εξουσιοδοτημένες λειτουργίες μπορούν να προσπελαστούν από έναν μη εξουσιοδοτημένο ή έναν τυπικό χρήστη.

Μια εφαρμογή μπορεί να απαιτήσει μια διαχειριστική διεπαφή για να καταστεί δυνατή η εξουσιοδοτημένη χρήση της πρόσβασης στις λειτουργίες που μπορούν να κάνουν αλλαγές στο πως ο ιστότοπος λειτουργεί. Αυτές οι αλλαγές περιλαμβάνουν:

- Πρόβλεψη λογαριασμού χρήστη
- Σχεδιασμός και διάταξη του ιστότοπου
- Χειρισμός δεδομένων
- Αλλαγές διαμόρφωσης

Σε πολλές περιπτώσεις, τέτοιες διεπαφές υλοποιούνται συνήθως με τη σκέψη πως να τις χωρίσουμε από τους κανονικούς χρήστες της ιστοσελίδας. Ο έλεγχος έχει ως στόχο την ανακάλυψη αυτών των διαχειριστικών διεπαφών και την πρόσβαση στη λειτουργικότητα που έχει να κάνει με τους εξουσιοδοτημένους χρήστες.

BLACK BOX TESTING

Τα παρακάτω περιγράφουν πολιτικές που μπορούν να χρησιμοποιηθούν για να ελέγξουν την παρουσία διαχειριστικών διεπαφών. Αυτές οι τεχνικές μπορούν επίσης να χρησιμοποιηθούν για τον έλεγχο σχετικών θεμάτων που περιλαμβάνουν την κλιμάκωση των δικαιωμάτων:

- Καταμέτρηση καταλόγων και αρχείων: Μια διαχειριστική διεπαφή μπορεί να υπάρχει αλλά να μην είναι εμφανώς διαθέσιμη στον ελεγκτή. Η προσπάθεια να μαντέψουμε το μονοπάτι της διαχειριστικής διεπαφής μπορεί να είναι τόσο εύκολη όσο η διαδικασία αποστολής αιτημάτων /admin ή /administrator κλπ. Ένας ελεγκτής μπορεί να πρέπει επίσης να αναγνωρίσει το όνομα του αρχείου της διαχειριστικής σελίδας. Η βίαιη περιήγηση στην αναγνωρισμένη σελίδα μπορεί να παρέχει πρόσβαση στην εφαρμογή.
- Σχόλια και συνδέσεις στην πηγή: Πολλές ιστοσελίδες χρησιμοποιούν κοινό κώδικα ο οποίος φορτώνεται για τους χρήστες όλων των ιστοσελίδων. Εξετάζοντας όλες τις πηγές που στέλνονται

στον client, μπορεί να ανακαλύψουμε διαχειριστικές λειτουργίες οι οποίες πρέπει να διερευνηθούν.

- Επανεξέταση της τεκμηρίωσης του server και της εφαρμογής: Εάν ο server της εφαρμογής ή η εφαρμογή έχουν αναπτυχθεί στην προκαθορισμένη διαμόρφωση μπορεί να είναι δυνατόν να αποκτήσουμε πρόσβαση στις διαχειριστικές διεπαφές χρησιμοποιώντας τις πληροφορίες που περιγράφονται στην διαμόρφωση ή στην τεκμηρίωση. Θα πρέπει να συμβουλευόμαστε τις λίστες με τα προκαθορισμένα password εάν βρεθεί μια διαχειριστική διεπαφή και ζητηθούν credentials.
- Εναλλακτική θύρα για τον server: Οι διαχειριστικές διεπαφές μπορεί να φαίνονται σε μια διαφορετική θύρα στον κεντρικό υπολογιστή απ' ότι η κύρια εφαρμογή. Για παράδειγμα, η διαχειριστική σελίδα του Apache Tomcat συχνά φαίνεται στην θύρα 8080.
- Παρέμβαση παραμέτρων: Μπορεί να ζητηθεί σε μια παράμετρο GET ή POST ή μια μεταβλητή cookie να επιτρέψει τη διαχειριστική λειτουργία. Για παράδειγμα:

include the presence of hidden fields such as:

<input type="hidden" name="admin" value="no"> or in a cookie: Cookie: session_cookie; useradmin=0

Μόλις ανακαλυφθεί μια διαχειριστική διεπαφή, ένας συνδυασμός όλων των παραπάνω τεχνικών μπορεί να χρησιμοποιηθεί για να προσπαθήσουμε να παρακάμψουμε την αυθεντικοποίηση.

Προκειμένου να πραγματοποιήσουμε, λοιπόν, αυτόν τον έλεγχο μεταφερθήκαμε στην διαχειριστική σελίδα του ιστότοπου που εξετάζουμε. Εφαρμόσαμε την τεχνική χρήσης των προκαθορισμένων (default) account και διαπιστώσαμε πως ισχύουν κάποια προκαθορισμένα ονόματα χρηστών όπως το “admin” ¨όχι όμως και προκαθορισμένοι κωδικοί πρόσβασης. Βέβαια η εφαρμογή εμφανίζει διαφορετικά μηνύματα για λάθος κωδικό χρήστη και διαφορετικά για λάθος κωδικό πρόσβασης δηλαδή μας αποκαλύπτει έμμεσα τα ονόματα των χρηστών. Οι κωδικοί που χρησιμοποιούνται δε θεωρούνται αδύναμοι (weak) ούτε διαμοιράζονται (shared). Επίσης δεν χρησιμοποιούνται προκαθορισμένοι(default) λογαριασμοί ούτε για testing και monitoring. Ακόμη η διαχειριστική σελίδα «φαίνεται» στο ίδιο port όπως και η εφαρμογή που εξετάζουμε.

3.3.8 Testing for HTTP Methods and XST

Σε αυτή τη δοκιμή ελέγχουμε εάν ο server είναι ρυθμισμένος έτσι ώστε να επιτρέπει HTTP μεθόδους που μπορούν δυνητικά να αποδειχθούν επικίνδυνες και κατά πόσο το Cross Site Tracing(XST) είναι δυνατό να πραγματοποιηθεί. Πολλές από αυτές τις μεθόδους σχεδιάζονται για να βοηθήσουν τους developers να αναπτύξουν και να ελέγξουν HTTP εφαρμογές. Αυτές οι HTTP μέθοδοι μπορούν να χρησιμοποιηθούν για φαύλους σκοπούς αν ο web server δεν είναι σωστά διαμορφωμένος. Επιπλέον, εξετάζεται το Cross Site Tracing, μια μορφή Cross site scripting που χρησιμοποιεί τους servers με τη μέθοδο HTTP TRACE. Αν και η GET και η POST είναι μακράν οι πιο συνηθισμένες μέθοδοι που χρησιμοποιούνται για την πρόσβαση σε πληροφορίες που παρέχονται από έναν web server, το

πρωτόκολλο Hypertext Transfer (HTTP) επιτρέπει κι άλλες (κάπως λιγότερες γνωστές) μεθόδους. Το RFC 2616 (το οποίο περιγράφει την έκδοση 1.1 του HTTP που είναι το σημερινό πρότυπο) ορίζει τις εξής οκτώ μεθόδους:

- HEAD
- GET
- POST
- PUT
- DELETE
- TRACE
- OPTIONS
- CONNECT

Κάποιες από αυτές τις μεθόδους μπορούν υπό περιπτώσεις να ενέχουν κίνδυνο για την ασφάλεια της εφαρμογής ιστού, καθώς επιτρέπουν σε έναν επιτιθέμενο να τροποποιήσει τα αρχεία που αποθηκεύονται στον web server και, σε ορισμένες περιπτώσεις, να κλέψουν τα credentials των νόμιμων χρηστών. Πιο συγκεκριμένα, οι μέθοδοι που θα έπρεπε να είναι απενεργοποιημένες είναι οι εξής:

- PUT: Αυτή η μέθοδος επιτρέπει στον client να ανεβάσει νέα αρχεία στον web server. Ένας επιτιθέμενος μπορεί να την εκμεταλλευτεί ανεβάζοντας κακόβουλα αρχεία (πχ ένα αρχείο .asp που εκτελεί εντολές επικαλούμενο το cmd.exe), ή χρησιμοποιώντας απλά τον server "θύμα" ως αποθήκη αρχείων.
- DELETE: Αυτή η μέθοδος επιτρέπει στον client να διαγράψει ένα αρχείου που βρίσκεται στον web server. Ένας επιτιθέμενος μπορεί να την εκμεταλλευτεί με ένα πολύ απλό και άμεσο τρόπο και να καταστρέψει μια ιστοσελίδα ή να εξαπολύσει μια επίθεση άρνησης παροχής της υπηρεσίας.
- CONNECT: Η μέθοδος αυτή μπορούσε να επιτρέψει σε έναν client να χρησιμοποιήσει τον web server σαν έναν proxy.
- TRACE: Αυτή η μέθοδος απλά επιστρέφει στον client οτιδήποτε έχει σταλεί στον server και χρησιμοποιείται κυρίως για σκοπούς debugging. Η μέθοδος αυτή, που αρχικά θεωρούνταν αβλαβής, μπορεί να χρησιμοποιηθεί ώστε να γίνει μια επίθεση γνωστή ως Cross Site Tracing, που έχει ανακαλυφθεί από τον Jeremiah Grossman.

Αν κάποια εφαρμογή χρειάζεται μια ή περισσότερες από αυτές τις μεθόδους, όπως η REST Web Services (η οποία μπορεί να απαιτεί την PUT ή την DELETE), είναι σημαντικό να ελεγχθεί ότι η χρήση τους είναι περιορισμένη σωστά στους έμπιστους χρήστες και σε ασφαλείς συνθήκες.

Αυθαίρετοι μέθοδοι HTTP

Ο Arshan Dabirsiaghi ανακάλυψε ότι πολλά πλαίσια εφαρμογών ιστού επέτρεπαν σε σωστά επιλεγμένες ή/και αυθαίρετες HTTP μεθόδους να παρακάμψουν ένα επίπεδο της δοκιμής ελέγχου πρόσβασης:

- Πολλά πλαίσια(frameworks) και γλώσσες αντιμετωπίζουν ένα "HEAD" αίτημα σαν ένα "GET". Αν υπήρχε ένας περιορισμός στα "GET" αιτήματα έτσι ώστε μόνο οι "αυθεντικοποιημένοι χρήστες" να έχουν πρόσβαση σε "GET" αιτήματα για ένα συγκεκριμένο servlet ή πόρο, θα είχε παρακαμφθεί για την εκδοχή του "HEAD" αιτήματος. Αυτή επέτρεπε την τυφλή μη εξουσιοδοτημένη υποβολή των εξουσιοδοτημένων GET αιτημάτων.
- Κάποια πλαίσια(frameworks) επέτρεπαν αυθαίρετες HTTP μεθόδους όπως οι "JEFF" ή οι "CATS" να χρησιμοποιούνται χωρίς περιορισμό. Αυτές συμπεριφέρονταν σαν να εκδόθηκε μια μέθοδος "GET" και διαπιστώθηκε ότι δεν υπόκεινται σε ελέγχους πρόσβασης με βάση το ρόλο της μεθόδου, σε πολλές γλώσσες και πλαίσια, και επέτρεπαν και πάλι την τυφλή μη εξουσιοδοτημένη υποβολή των "νόμιμων" GET αιτημάτων.

Σε πολλές περιπτώσεις, ο κώδικας που ελέγχεται ρητά για μια μέθοδο "GET" ή "POST" θα ήταν ασφαλής.

BLACK BOX TESTING

Ανακάλυψη των υποστηριζόμενων μεθόδων

Για να πραγματοποιήσουμε αυτό τον έλεγχο, θα πρέπει με κάποιο τρόπο να καταλάβουμε ποιες μέθοδοι HTTP υποστηρίζονται από τον web server που εξετάζουμε. Η μέθοδος OPTIONS HTTP μας παρέχει τον πιο άμεσο και αποτελεσματικό τρόπο για να το κάνουμε αυτό. Το RFC 2616 δηλώνει ότι "Η μέθοδος OPTIONS αντιπροσωπεύει ένα αίτημα για πληροφορίες σχετικά με τις επιλογές επικοινωνίας που είναι διαθέσιμες στην αλυσίδα αίτημα/απάντηση και προσδιορίζονται από το REQUEST-URI". Η μέθοδος δοκιμής είναι εξαιρετικά απλή και θα πρέπει μόνο να ενεργοποιήσουμε το netcat (ή το telnet):

icesurfer@nightblade ~ $ nc www.victim.com 80
OPTIONS / HTTP/1.1
Host: www.victim.com

HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Tue, 31 Oct 2006 08:00:29 GMT
Connection: close
Allow: GET, HEAD, POST, TRACE, OPTIONS
Content-Length: 0

icesurfer@nightblade ~ $

Όπως βλέπουμε στο παράδειγμα, η OPTIONS παρέχει μια λίστα των μεθόδων που υποστηρίζονται από τον web server και στην περίπτωση αυτή μπορούμε να δούμε, για παράδειγμα, ότι η μέθοδος TRACE είναι ενεργοποιημένη. Ο κίνδυνος που τίθεται από αυτή τη μέθοδο παρουσιάζεται παρακάτω.

Εφαρμόζουμε την παραπάνω μέθοδο ανίχνευσης των υποστηριζόμενων HTTP μεθόδων στον ιστότοπο που εξετάζουμε και συγκεκριμένα προσπαθούμε μέσω της μεθόδου TRACE να ανακαλύψουμε τις μεθόδους αυτές:

nc -v www.ds.unipi.gr 80
--sr2-is.ted.unipi.gr [83.212.239.100] 80 (http) open
OPTIONS / HTTP/1.1

Απάντηση από τον server:

HTTP/1.1 400 Bad Request
Date: Sat, 19 Nov 2011 19:47:13 GMT
Server: Apache/2.2.3 (CentOS)
Content-Length: 306
Connection: close
Content-Type: text/html; charset=iso-8859-1

Άρα συμπεραίνουμε πως η μέθοδος OPTIONS δεν υποστηρίζεται στην έκδοση 1.1 επομένως επιχειρούμε να καλέσουμε απευθείας τις μεθόδους που μας ενδιαφέρουν στους παρακάτω ελέγχους.

Έλεγχος δυναμικού XST

Σημείωση: Προκειμένου να κατανοήσουμε τη λογική και τους στόχους αυτής της επίθεσης θα πρέπει να είμαστε εξοικειωμένοι με επιθέσεις Cross Site Scripting.

Η μέθοδος TRACE, ενώ φαινομενικά δείχνει ακίνδυνη, μπορεί να αξιοποιηθεί με επιτυχία σε κάποια σενάρια έτσι ώστε να κλέψει τα credentials των νόμιμων χρηστών. Η τεχνική της επίθεσης ανακαλύφθηκε από τον Jeremiah Grossman το 2003 σε μια προσπάθεια να παρακάμψει την ετικέτα HTTPOnly που η Microsoft είχε εισάγει στον Internet Explorer 6 με service pack 1 για να προστατέψει την πρόσβαση στα cookies από την JavaScript. Στην πραγματικότητα, ένα από τα πιο επαναλαμβανόμενα μοτίβα στο Cross Site Scripting είναι η πρόσβαση στο αντικείμενο document.cookie και η αποστολή του στον web server ελεγχόμενο πια από τον επιτιθέμενο και έτσι να μπορεί να υποκλέψει το session του θύματος. Σημειώνοντας ένα cookie σαν httpOnly απαγορεύει στην JavaScript να έχει πρόσβαση σε αυτό, προστατεύοντάς το από το να σταλεί σε μια τρίτη οντότητα. Ωστόσο, η μέθοδος TRACE μπορεί να

χρησιμοποιηθεί για να παρακαμφθεί αυτή η προστασία και να υπάρξει πρόσβαση στο cookie ακόμη και σε αυτό το σενάριο.

Όπως προαναφέρθηκε, η TRACE απλά επιστρέφει οποιαδήποτε συμβολοσειρά(string) αποστέλλεται στον web server. Προκειμένου να επιβεβαιωθεί η παρουσία της (ή να ελέγξουμε δυο φορές τα αποτελέσματα του OPTION αιτήματος που φαίνεται παρακάτω), μπορούμε να προχωρήσουμε όπως φαίνεται στο παράδειγμα που ακολουθεί:

icesurfer@nightblade ~ $ nc www.victim.com 80
TRACE / HTTP/1.1
Host: www.victim.com
HTTP/1.1 200 OK
Server: Microsoft-IIS/5.0
Date: Tue, 31 Oct 2006 08:01:48 GMT
Connection: close
Content-Type: message/http
Content-Length: 39

TRACE / HTTP/1.1
Host: www.victim.com

Όπως μπορούμε να δούμε, το σώμα της απάντησης είναι ακριβώς ένα αντίγραφο του αρχικού αιτήματος μας, πράγμα που σημαίνει ότι ο στόχος μας επιτρέπει αυτή τη μέθοδο. Τώρα όμως πρέπει να προσδιορίσουμε το που ελλοχεύει ο κίνδυνος. Αν έχουμε αναθέσει σε έναν browser να υποβάλλει ένα αίτημα TRACE στον web server, και αυτός ο browser έχει ένα cookie για αυτό το domain, το cookie θα συμπεριληφθεί αυτόματα στην επικεφαλίδα του αιτήματος και συνεπώς θα εμφανιστεί στα αποτελέσματα της απάντησης. Στο σημείο αυτό, η συμβολοσειρά(string) του cookie θα είναι προσβάσιμη από τη JavaScript και θα είναι τελικά δυνατό να σταλεί σε ια τρίτη οντότητα ακόμη και αν το cookie είναι σημειωμένο ως httpOnly.

Υπάρχουν πολλοί τρόποι να κάνουμε έναν browser να έκδοση ένα TRACE αίτημα όπως ο έλεγχος XMLHTTP ActiveX στον Internet Explorer και ο XMLDOM στον Mozilla και στον Netscape. Ωστόσο, για λόγους ασφαλείας, ο browser επιτρέπεται να ξεκινήσει μια σύνδεση μόνο στο domain που βρίσκεται το εχθρικό script. Αυτός είναι ένας ελαφρυντικός παράγοντας, καθώς ο επιτιθέμενος πρέπει να συνδυάσει την μέθοδο TRACE με μια άλλη ευπάθεια προκειμένου να εξαπολύσει επίθεση. Βασικά, ένας επιτιθέμενος έχει δύο τρόπους για να ξεκινήσει με επιτυχία μια επίθεση Cross Site Scripting:

1. Αξιοποιώντας μια άλλη ευπάθεια στην πλευρά του server: ο επιτιθέμενος εγχέει το εχθρικό απόσπασμα Javascript, που περιέχει το TRACE αίτημα, στην ευπαθή εφαρμογή όπως στην κανονική επίθεση Cross Site Scripting.

2. Αξιοποιώντας μια ευπάθεια στην πλευρά του client: ο επιτιθέμενος δημιουργεί μια κακόβουλη ιστοσελίδα που περιέχει το εχθρικό απόσπασμα Javascript και εκμεταλλεύεται κάποια ευπάθεια μεταξύ των domain του browser του θύματος, προκειμένου ο Javascript κώδικας να εκτελέσει επιτυχώς μια σύνδεση με την ιστοσελίδα που υποστηρίζει την μέθοδο TRACE και προέρχεται από το cookie που ο επιτιθέμενος προσπαθεί να κλέψει.

Πραγματοποιούμε τον έλεγχο για το δυναμικό XST στον υπό εξέταση ιστότοπο www.ds.unipi.gr και υποβάλλουμε στο εργαλείο netcat τα παρακάτω αιτήματα:

nc -v www.ds.unipi.gr 80
--sr2-is.ted.unipi.gr [83.212.239.100] 80 (http) open
TRACE / HTTP/1.1

Απάντηση από τον server:
HTTP/1.1 400 Bad Request
Date: Sat, 19 Nov 2011 19:47:13 GMT
Server: Apache/2.2.3 (CentOS)
Content-Length: 306
Connection: close
Content-Type: text/html; charset=iso-8859-1

Διαπιστώνουμε πως ούτε το άμεσο αίτημα για την μέθοδο TRACE δουλεύει όσον αφορά την έκδοση 1.1 στον συγκεκριμένο server που φιλοξενεί την εφαρμογή που εξετάζουμε. Θα προσπαθήσουμε τώρα να αιτηθούμε την ίδια μέθοδο στην έκδοση 1.0 και να ελέγξουμε κατά πόσο περνάει το αίτημά μας αυτό:

nc -v www.ds.unipi.gr 80
--sr2-is.ted.unipi.gr [83.212.239.100] 80 (http) open
TRACE / HTTP/1.0

Απάντηση από τον server:
HTTP/1.1 403 Forbidden
Date: Sat, 19 Nov 2011 19:11:38 GMT
Server: Apache/2.2.3 (CentOS)
Content-Length: 282
Connection: close
Content-Type: text/html; charset=iso-8859-1

Από την απάντηση του server μας στο αίτημα που υποβάλλαμε είναι εμφανές πως η μέθοδος TRACE είναι απαγορευμένη στην έκδοση HTTP/1.0.

Black Box δοκιμή των HTTP μεθόδων αλλοίωσης

Οι δοκιμές για τις HTTP μεθόδους αλλοίωσης είναι ουσιαστικά οι ίδιες με τις δοκιμές για XST.

Δοκιμές για αυθαίρετες μεθόδους HTTP

Βρείτε μια σελίδα που θέλετε να επισκεφθείτε που έχει έναν περιορισμό ασφάλειας τέτοιον που φυσιολογικά θα προκαλέσει μια 302 ανακατεύθυνση σε μια σελίδα σύνδεσης ή θα προκαλέσει την απευθείας σύνδεση. Η δοκιμή του URL στο παράδειγμα αυτό λειτουργεί με αυτόν τον τρόπο όπως και κάνουν πολλές εφαρμογές ιστού. Ωστόσο, εάν λάβετε μια "200" απάντηση αιτήματος που δεν είναι μια σελίδα σύνδεσης (login) είναι δυνατόν να παρακαμφθεί η αυθεντικοποίηση και έτσι η εξουσιοδότηση:

[rapidoffenseunit:~] vanderaj% nc www.example.com 80
JEFF / HTTP/1.1
Host: www.example.com

HTTP/1.1 200 OK
Date: Mon, 18 Aug 2008 22:38:40 GMT
Server: Apache Set-Cookie: PHPSESSID=K53QW...

Αν το πλαίσιό σας ή το firewall ή η εφαρμογή δεν υποστηρίζει μια μέθοδο “JEFF”, θα πρέπει να εκδίδεται μια σελίδα σφάλματος (ή καλύτερα μια σελίδα 405 “Not Allowed” ή 501 “Not Implemented”). Αν υπηρετεί αυτό το αίτημα τότε είναι ευπαθές σε αυτό το θέμα. Αν νιώθετε ότι το σύστημα είναι ευπαθές στο ζήτημα αυτό, εισάγεται επιθέσεις τύπου CRSF για να εκμεταλλευτείτε το θέμα εκτενέστερα:

- *FOOBAR /admin/createUser.php?member=myAdmin*
- *JEFF /admin/changePw.php?member=myAdmin&passwd=foo123&confirm=foo123*
- *CATS /admin/groupEdit.php?group=Admins&member=myAdmin&action=add*

Με λίγη τύχη χρησιμοποιώντας τις παραπάνω τρεις εντολές, μεταβάλλονται για να ταιριάζουν με την υπό δοκιμή εφαρμογή και τις απαιτήσεις της δοκιμής, ένας νέος χρήστης θα δημιουργηθεί, ένας κωδικός πρόσβασης θα του εκχωρηθεί και θα γίνει διαχειριστής.

Εφαρμόζουμε τώρα το αίτημα αυτό στην δική μας περίπτωση και βλέπουμε τα αποτελέσματα που ακολουθούν παρακάτω:

nc -v www.ds.unipi.gr 80
--sr2-is.ted.unipi.gr [83.212.239.100] 80 (http) open
JEFF / HTTP/1.1

Απάντηση από τον server:

HTTP/1.1 400 Bad Request
Date: Sat, 19 Nov 2011 19:47:13 GMT
Server: Apache/2.2.3 (CentOS)
Content-Length: 306
Connection: close
Content-Type: text/html; charset=iso-8859-1

Η απάντηση και σε αυτή την περίπτωση δείχνει ξεκάθαρα πως πρόκειται για ένα αίτημα που δεν υποστηρίζεται. Δοκιμάζουμε τώρα το ίδιο αίτημα για HTTP/1.0:

nc -v www.ds.unipi.gr 80
--sr2-is.ted.unipi.gr [83.212.239.100] 80 (http) open
*JEFF / HTTP/*1.0

Στο αίτημά μας αυτό η απάντηση που πήραμε είναι μεν ένα αίτημα *200 OK* όμως το γεγονός ότι επιστρέφεται και σώμα μαζί με την απάντηση αυτή συμπεραίνουμε πως η εφαρμογή δεν μπορεί να επεξεργαστεί το αίτημα χωρίς αυθεντικοποίηση ή εξουσιοδότηση.

Δοκιμές για παράκαμψη του HEAD ελέγχου πρόσβασης

Βρείτε μια σελίδα που θέλετε να επισκεφθείτε που έχει έναν περιορισμό ασφάλειας τέτοιον που φυσιολογικά θα προκαλέσει μια 302 ανακατεύθυνση σε μια σελίδα σύνδεσης ή θα προκαλέσει την απευθείας σύνδεση. Η δοκιμή του URL στο παράδειγμα αυτό λειτουργεί με αυτόν τον τρόπο όπως και κάνουν πολλές εφαρμογές ιστού. Ωστόσο, εάν λάβετε μια "200" απάντηση αιτήματος που δεν είναι μια σελίδα σύνδεσης (login) είναι δυνατόν να παρακαμφθεί η αυθεντικοποίηση και με αυτόν τον τρόπο και η εξουσιοδότηση.

[rapidoffenseunit:~] vanderaj% nc www.example.com 80
HEAD /admin HTTP/1.1
Host: www.example.com

HTTP/1.1 200 OK
Date: Mon, 18 Aug 2008 22:44:11 GMT
Server: Apache
Set-Cookie: PHPSESSID=pKi...; path=/; HttpOnly
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0

Pragma: no-cache

Set-Cookie: adminOnlyCookie1=...; expires=Tue, 18-Aug-2009 22:44:31 GMT; domain=www.example.com

Set-Cookie: adminOnlyCookie2=...; expires=Mon, 18-Aug-2008 22:54:31 GMT; domain=www.example.com

Set-Cookie: adminOnlyCookie3=...; expires=Sun, 19-Aug-2007 22:44:30 GMT; domain=www.example.com

Content-Language: EN

Connection: close

Content-Type: text/html; charset=ISO-8859-1

Εάν λάβουμε ένα μήνυμα "405 Method not allowed" ή "501 Method Unimplemented", η εφαρμογή/πλαίσιο/γλώσσα/σύστημα/firewall δουλεύει σωστά. Αν ένας κωδικός αιτήματος "200" επιστραφεί και η απάντηση δεν περιέχει σώμα, είναι πιθανόν η εφαρμογή να έχει επεξεργαστεί το αίτημα χωρίς αυθεντικοποίηση ή εξουσιοδότηση και ο περαιτέρω έλεγχος είναι δικαιολογημένος.

Εάν αισθάνεστε ότι το σύστημα είναι ευπαθές σ' αυτό το θέμα, εισάγεται επιθέσεις τύπου CSRF για να εκμεταλλευτείτε το θέμα εκτενέστερα:

- *HEAD /admin/createUser.php?member=myAdmin*
- *HEAD/admin/changePw.php?member=myAdmin&passwd=foo123&confirm=foo13*
- *HEAD /admin/groupEdit.php?group=Admins&member=myAdmin&action=add*

Με λίγη τύχη χρησιμοποιώντας τις παραπάνω τρεις εντολές, μεταβάλλονται για να ταιριάζουν με την υπό δοκιμή εφαρμογή και τις απαιτήσεις της δοκιμής, ένας νέος χρήστης θα δημιουργηθεί, ένας κωδικός πρόσβασης θα του εκχωρηθεί και θα γίνει διαχειριστής, όλα χρησιμοποιώντας τυφλή υποβολή αιτημάτων.

Δοκιμάζουμε, λοιπόν και στον ιστότοπο που εξετάζουμε να παρακάμψουμε το HEAD ελέγχου πρόσβασης:

nc -v www.ds.unipi.gr 80

--sr2-is.ted.unipi.gr [83.212.239.100] 80 (http) open

HEAD /admin HTTP/1.1

Απάντηση από τον server:

HTTP/1.1 400 Bad Request

Date: Sat, 19 Nov 2011 19:47:13 GMT

Server: Apache/2.2.3 (CentOS)

Content-Length: 306

Connection: close

Content-Type: text/html; charset=iso-8859-1

Η απάντηση και σε αυτή την περίπτωση δείχνει ξεκάθαρα πως πρόκειται για ένα αίτημα που δεν υποστηρίζεται. Δοκιμάζουμε τώρα το ίδιο αίτημα για HTTP/1.0:

nc -v www.ds.unipi.gr 80
--sr2-is.ted.unipi.gr [83.212.239.100] 80 (http) open
HEAD /admin HTTP/1.0

Στο αίτημά μας αυτό η απάντηση που πήραμε είναι μεν ένα αίτημα *200 OK* όμως το γεγονός ότι επιστρέφεται και σώμα μαζί με την απάντηση αυτή συμπεραίνουμε πως η εφαρμογή δεν μπορεί να επεξεργαστεί το αίτημα χωρίς αυθεντικοποίηση ή εξουσιοδότηση.

3.4 Authentication Testing

Αυθεντικοποίηση ονομάζεται η πράξη της καθιέρωσης ή της επιβεβαίωσης κάποιου ως αυθεντικού, δηλαδή ότι οι αξιώσεις, που γίνονται από/ή για το συγκεκριμένο πράγμα είναι αληθινές. Η αυθεντικοποίηση ενός αντικειμένου μπορεί να δηλώσει την προέλευσή του, ενώ η αυθεντικοποίηση ενός προσώπου αποτελείται συχνά από την επαλήθευση της ταυτότητάς του. Η αυθεντικοποίηση εξαρτάται από έναν ή περισσότερους παράγοντες αυθεντικοποίησης. Στην ασφάλεια υπολογιστών η αυθεντικοποίηση είναι η διαδικασία του ελέγχου της ψηφιακής ταυτότητας του αποστολέα σε μια επικοινωνία. Ένα σύνηθες παράδειγμα μιας τέτοιας διαδικασίας είναι η διαδικασία σύνδεσης (login). Η δοκιμή του σχήματος αυθεντικοποίησης σημαίνει την κατανόηση του πώς η διαδικασία αυθεντικοποίησης λειτουργεί και χρησιμοποιώντας αυτές τις πληροφορίες να μπορέσουμε να παρακάμψουμε το μηχανισμό αυθεντικοποίησης.

3.4.1 Credentials transport over encrypted channel

Έλεγχος για τα credentials της μεταφοράς σημαίνει να επιβεβαιώσουμε ότι τα στοιχεία αυθεντικοποίησης του χρήστη μεταφέρονται μέσω ενός κρυπτογραφημένου καναλιού προκειμένου να αποφευχθεί το να υποκλαπούν από κακόβουλους χρήστες. Η ανάλυση εστιάζει μόνο στην προσπάθεια να καταλάβουμε αν τα δεδομένα ταξιδεύουν μη κρυπτογραφημένα από τον web browser στον server, ή αν η εφαρμογή ιστού παίρνει τα κατάλληλα μέτρα ασφαλείας με τη χρήση πρωτοκόλλου όπως το HTTPS. Το πρωτόκολλο HTTPS έχει δομηθεί σε TLS/SSL ώστε να κρυπτογραφεί δεδομένα που μεταδίδονται και να εξασφαλιστεί ότι ο χρήστης ανακατευθύνεται στην επιθυμητή ιστοσελίδα. Σαφώς, το γεγονός ότι η κίνηση είναι κρυπτογραφημένη δεν σημαίνει απαραίτητα πως είναι και πλήρως ασφαλής. Η ασφάλεια επίσης εξαρτάται από τον αλγόριθμο κρυπτογράφησης που χρησιμοποιείται και το πόσο σφριγηλά είναι τα κλειδιά που η εφαρμογή χρησιμοποιεί, αλλά αυτό το συγκεκριμένο θέμα δεν θα αναλυθεί σε αυτή την ενότητα. Εδώ ο ελεγκτής πρέπει να προσπαθήσει να κατανοήσει εάν τα δεδομένα που οι χρήστες χρησιμοποιούν σε web φόρμες, για παράδειγμα για να συνδεθούν με μια ιστοσελίδα του web, μεταδίδονται χρησιμοποιώντας ασφαλή πρωτόκολλα που τα προστατεύουν από έναν επιτιθέμενο ή όχι.

Σήμερα, το πιο κοινό παράδειγμα αυτού του θέματος είναι η αρχικής σελίδα εισαγωγής μιας εφαρμογής ιστού. Ο ελεγκτής πρέπει να επιβεβαιώσει ότι τα credentials του χρήστη μεταδίδονται μέσω ενός κρυπτογραφημένου καναλιού. Για να συνδεθεί σε μια σελίδα, συνήθως, ο χρήστης πρέπει να συμπληρώσει μια απλή φόρμα που μεταδίδει τα στοιχεία που εισήχθησαν μέσω της μεθόδου POST. Αυτό που είναι λιγότερο εμφανές είναι ότι τα δεδομένα αυτά μπορούν να περάσουν χρησιμοποιώντας το πρωτόκολλο HTTP, που σημαίνει μη ασφαλής δρόμος, ή χρησιμοποιώντας το HTTPS που κρυπτογραφεί τα δεδομένα. Κι αν θέλουμε να περιπλέξουμε τα πράγματα ακόμη περισσότερο, υπάρχει η πιθανότητα η ιστοσελίδα να έχει την αρχική σελίδα σύνδεσης προσπελάσιμη μέσω HTTP (κάνοντάς μας να πιστέψουμε πως η μετάδοση είναι μη-ασφαλής), αλλά στη συνέχεια στέλνει πράγματι δεδομένα μέσω HTTPS. Αυτός ο έλεγχος πρέπει να πραγματοποιηθεί για να είμαστε ότι ένας επιτιθέμενος δεν μπορεί να ανακτήσει ευαίσθητες πληροφορίες με κάνοντας sniffing το δίκτυο με ένα εργαλείο(sniffer).

BLACK BOX TESTING

Στο παράδειγμα που ακολουθεί θα χρησιμοποιήσουμε το WebScarab προκειμένου να συλλάβουμε πακέτα επικεφαλίδων και να τα επιθεωρήσουμε. Μπορούμε να χρησιμοποιήσουμε όποιον proxy θέλουμε.

Μελέτη περίπτωσης: Αποστολή δεδομένων με τη μέθοδο POST μέσω HTTP

Ας υποθέσουμε ότι η σελίδα σύνδεσης παρουσιάζει μια φόρμα με πεδία User, Pass και το κουμπί 'Υποβολή' για να αυθεντικοποιήσει το χρήστη και να δώσει πρόσβαση στην εφαρμογή. Εάν κοιτάξουμε την επικεφαλίδα του αιτήματος μας με το WebScarab, θα πάρουμε κάτι σαν το αυτό:

POST http://www.example.com/AuthenticationServlet HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; it; rv:1.8.1.14) Gecko/2008040
Accept: text/xml,application/xml,application/xhtml+xml
Accept-Language: it-it,it;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,;q=0.7*
Keep-Alive: 300
Connection: keep-alive
Referer: http://www.example.com/index.jsp
Cookie:JSESSIONID=LVrRRQQXgwyWpW7QMnS49vtW1yBdqn98CGlkP4jTvVCGdyPkmn3S!
Content-Type: application/x-www-form-urlencoded
Content-length: 64

delegated_service=218&User=test&Pass=test&Submit=SUBMIT

Από αυτό το παράδειγμα ο ελεγκτής μπορεί να καταλάβει ότι η POST στέλνει δεδομένα στη σελίδα 'www.example.com/AuthenticationServlet' εύκολα χρησιμοποιώντας HTTP. Έτσι σε αυτή την περίπτωση, τα δεδομένα μεταδίδονται χωρίς κρυπτογράφηση και ένας κακόβουλος χρήστης θα μπορούσε να διαβάζει το όνομα χρήστη και τον κωδικό πρόσβασης κάνοντας απλά sniffing το δίκτυο με ένα εργαλείο όπως το Wireshark.

Μελέτη περίπτωσης: Αποστολή δεδομένων με τη μέθοδο POST μέσω HTTPS

Υποθέστε ότι η εφαρμογή ιστού χρησιμοποιεί το HTTPS πρωτόκολλο για να κρυπτογραφήσει τα δεδομένα που στέλνουμε (ή τουλάχιστον για αυτά που σχετίζονται με την αυθεντικοποίηση). Σε αυτή την περίπτωση, προσπαθώντας να αποκτήσουνε πρόσβαση στη σελίδα σύνδεσης και να αυθεντικοποιηθούμε, η επικεφαλίδα του POST αιτήματός μας θα ήταν παρόμοια με την παρακάτω:

POST https://www.example.com:443/cgi-bin/login.cgi HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; it; rv:1.8.1.14) Gecko/20080404
Accept: text/xml,application/xml,application/xhtml+xml,text/html
Accept-Language: it-it,it;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,;q=0.7*
Keep-Alive: 300 Connection: keep-alive
Referer: https://www.example.com/cgi-bin/login.cgi
Cookie: language=English;
Content-Type: application/x-www-form-urlencoded
Content-length: 50

Command=Login&User=test&Pass=test

Μπορούμε να δούμε ότι το αίτημα απευθύνεται στη διεύθυνση www.example.com:443/cgi-bin/login.cgi χρησιμοποιώντας το πρωτόκολλο HTTPS. Αυτό εξασφαλίζει ότι τα δεδομένα μας θα αποσταλούν μέσω ενός κρυπτογραφημένου καναλιού που δεν μπορεί να αναγνωστεί από άλλους.

Μελέτη Περίπτωσης: Αποστολή δεδομένων με την μέθοδο POST σε μια σελίδα προσπελάσιμη μέσω HTTP

Τώρα ας υποθέσουμε πως έχουμε μια ιστοσελίδα προσπελάσιμη μέσω HTTP και στη συνέχεια μόνο τα δεδομένα της φόρμας αυθεντικοποίησης αποστέλλονται μέσω του HTTPS. Αυτό σημαίνει πως τα δεδομένα μας μεταδίδονται με έναν ασφαλή τρόπο μέσω κρυπτογραφίας. Αυτό συμβαίνει, για παράδειγμα, όταν είμαστε σε μια πύλη μιας μεγάλης εταιρείας που προσφέρει διάφορες πληροφορίες και υπηρεσίες διαθέσιμες στο κοινό, χωρίς ταυτοποίηση, αλλά έχει επίσης και ένα ιδιωτικό τμήμα

προσβάσιμο από την αρχική σελίδα μέσω μιας σύνδεσης. Έτσι όταν προσπαθούμε να συνδεθούμε, η επικεφαλίδα του αιτήματός ας θα είναι όπως το ακόλουθο παράδειγμα:

POST https://www.example.com:443/login.do HTTP/1.1
Host: www.example.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; it; rv:1.8.1.14) Gecko/20080404
Accept: text/xml,application/xml,application/xhtml+xml,text/html
Accept-Language: it-it,it;q=0.8,en-us;q=0.5,en;q=0.3
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,;q=0.7*
Keep-Alive: 300
Connection: keep-alive
Referer: http://www.example.com/homepage.do
Cookie: SERVTIMSESSIONID=s2JyLkvDJ9ZhX3yr5BJ3DFLkdphH0QNSJ3VQB6pLhjkW6F
Content-Type: application/x-www-form-urlencoded
Content-length: 45

User=test&Pass=test&portal=ExamplePortal

Μπορούμε δούμε ότι το αίτημά μας απευθύνεται στη διεύθυνση www.example.com:443/login.do χρησιμοποιώντας HTTPS. Αλλά εάν ρίξουμε μια ματιά στο πεδίο 'referrer' της επικεφαλίδας (τη σελίδα από την οποία ήρθαμε), είναι www.example.com/homepage.do και είναι προσπελάσιμη μέσω HTTP. Έτσι, σε αυτή την περίπτωση, δεν έχουμε καμία κλειδαριά να εμφανίζεται στο παράθυρο του browser που να μας λέει πως χρησιμοποιούμε μια ασφαλή σύνδεση όμως, στην πραγματικότητα, στέλνουμε δεδομένα μέσω HTTPS. Αυτό εξασφαλίζει πως κανείς άλλος δεν μπορεί να διαβάσει τα δεδομένα που στέλνουμε.

Μελέτη περίπτωσης: Αποστολή δεδομένων με την μέθοδο GET μέσω HTTPS

Σε αυτό το τελευταίο παράδειγμα, ας υποθέσουμε πως η εφαρμογή μεταφέρει δεδομένα χρησιμοποιώντας την μέθοδο GET. Αυτή η μέθοδος δε θα πρέπει ποτέ να χρησιμοποιείται σε μια φόρμα που μεταδίδει ευαίσθητες πληροφορίες όπως το όνομα χρήστη και ο κωδικός πρόσβασης., επειδή εμφανίζονται ξεκάθαρα στο URL και αυτό συνεπάγεται ένα ολόκληρο σύνολο άλλων ζητημάτων. Επομένως αυτό το παράδειγμα είναι καθαρά επίδειξης αλλά στην πραγματικότητα προτείνεται έντονα η χρήση της μεθόδου POST στη θέση του. Αυτό συμβαίνει γιατί όταν χρησιμοποιείται η μέθοδος GET το URL που πραγματοποιεί το αίτημα είναι εύκολα διαθέσιμο, για παράδειγμα, από τα αρχεία καταγραφής(logs) του server εκθέτοντας έτσι τις ευαίσθητα δεδομένα σε διαρροή.

GET https://www.example.com/success.html?user=test&pass=test HTTP/1.1

Host: www.example.com

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; it; rv:1.8.1.14) Gecko/20080404

Accept: text/xml,application/xml,application/xhtml+xml,text/html

Accept-Language: it-it,it;q=0.8,en-us;q=0.5,en;q=0.3

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-1,utf-8;q=0.7,;q=0.7*

Keep-Alive: 300

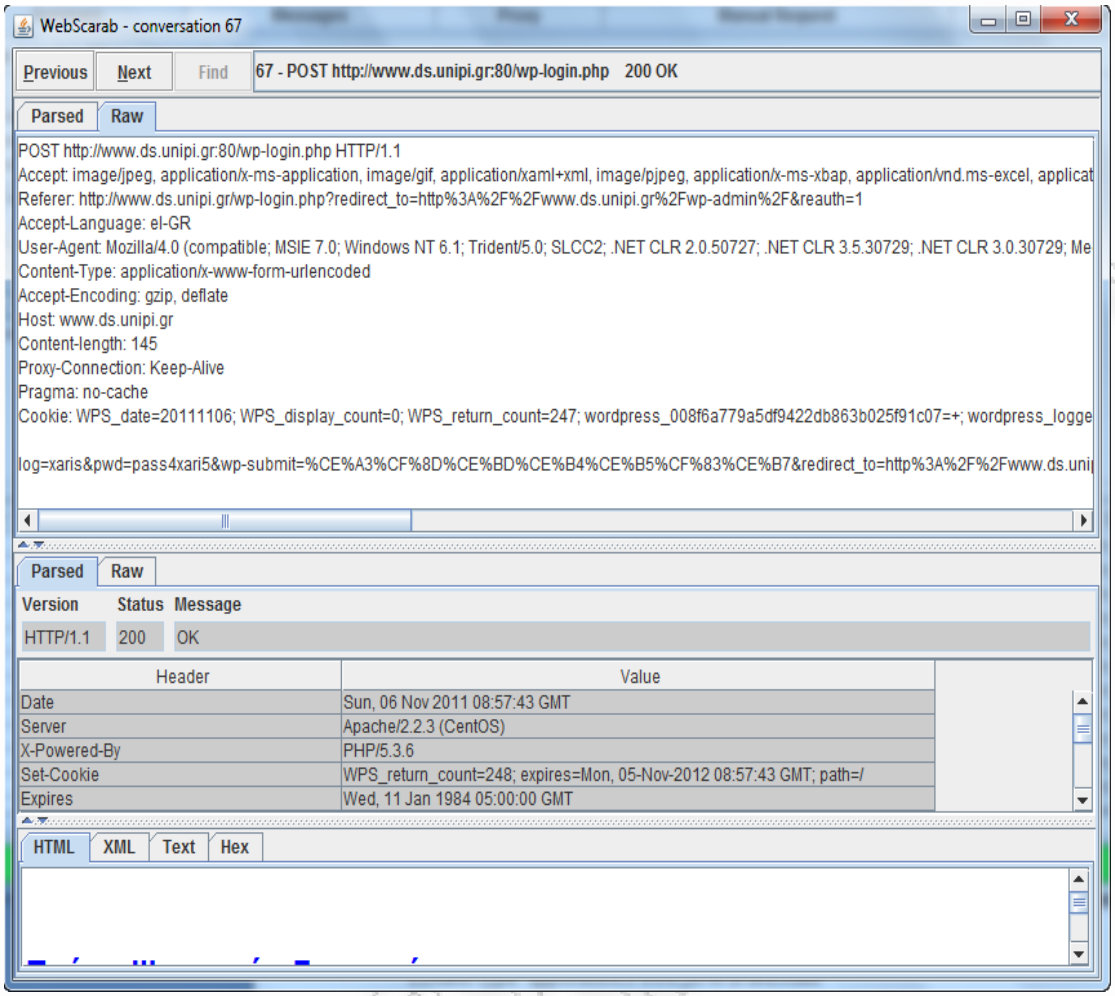
Connection: keep-alive Referer: https://www.example.com/form.html

If-Modified-Since: Mon, 30 Jun 2008 07:55:11 GMT

If-None-Match: "43a01-5b-4868915f"

Βλέπουμε πως τα δεδομένα μεταφέρονται σε απλό κείμενα στο URL και όχι στο σώμα το μηνύματος όπως συνέβαινε προηγουμένως. Αλλά θα πρέπει να θεωρήσουμε πως το TLS/SSL είναι ένα πρωτόκολλο επιπέδου 5, χαμηλότερου επιπέδου από το HTTP, επομένως ολόκληρο το HTTP πακέτο είναι ακόμη κρυπτογραφημένο και το URL είναι μη αναγνώσιμο σε οποιονδήποτε επιτιθέμενο. Δεν είναι καλή πρακτική να χρησιμοποιούμε την μέθοδο GET σε αυτές τις περιπτώσεις, επειδή η πληροφορία που περιέχεται στο URL μπορεί να αποθηκευτεί σε πολλούς servers, όπως server και proxy, επιτρέποντας τη διαρροή των συνθηματικών(credentials) του χρήστη.

Στην περίπτωση του ιστότοπου του που εξετάζουμε στην εργασία μας θα προσπαθήσουμε να ανακαλύψουμε τον τρόπο με τον οποίο τα συνθηματικά(credentials) μεταφέρονται στον server κατά την προσπάθεια σύνδεσης με τον server της εφαρμογής. Χρησιμοποιώντας το WebScarab θα ανατρέξουμε στην συνομιλία που αναφέρεται στη διαδικασία login στον server. Στη συνομιλία, λοιπόν, αυτή φαίνεται από τον κώδικα πως χρησιμοποιείται HTTP επικοινωνία όπως περιγράψαμε στην πρώτη μελέτη περίπτωσης στα παραπάνω γενικότερα παραδείγματα. Συμπεραίνουμε επομένως πως η επικοινωνία είναι ευπαθής καθώς τα συνθηματικά (credentials) μεταφέρονται μη-κρυπτογραφημένα στον server και είναι δυνατόν για κάποιον κακόβουλο χρήστη να τα ανακαλύψει πολύ εύκολα με μια διαδικασία sniffing. Η εικόνα που ακολουθεί δείχνει λεπτομερώς την επικοινωνία μεταξύ client-server κατά τη διαδικασία αυθεντικοποίησης και αποστολής των credentials στον server.



Εικόνα : **Αίτημα σύνδεσης με την εφαρμογή www.ds.unipi.gr μέσω της σελίδας διαχείρισης της εφαρμογής.**

3.4.2 Testing for user enumeration

Ο σκοπός αυτού του ελέγχου είναι να εξακριβώσει αν είναι δυνατόν να συλλέξουμε ένα σύνολο έγκυρων ονομάτων χρηστών αλληλεπιδρώντας με τον μηχανισμό αυθεντικοποίησης της εφαρμογής. Αυτός ο έλεγχος είναι χρήσιμος και για τον έλεγχο επιθέσεων brute force επιθέσεων, στον οποίο εξακριβώνουμε εάν, δεδομένου ενός ονόματος χρήστη, είναι δυνατόν να βρούμε τον αντίστοιχο κωδικό πρόσβασης. Συχνά, οι εφαρμογές ιστού αποκαλύπτουν πότε ένα όνομα χρήστη υπάρχει στο σύστημα είτε ως συνέπεια ενός λάθους διαμόρφωσης είτε σαν απόφαση σχεδιασμού της εφαρμογής. Για παράδειγμα, κάποιες φορές όταν εισάγουμε λανθασμένα συνθηματικά(credentials) παίρνουμε ένα μήνυμα που δηλώνει είτε ότι το όνομα χρήστη υπάρχει στο σύστημα είτε ότι ο χρησιμοποιούμενος κωδικός πρόσβασης είναι λάθος. Οι πληροφορίες που ανακτώνται μπορούν να χρησιμοποιηθούν από έναν επιτιθέμενο ώστε να αποκτήσει μια λίστα των χρηστών του συστήματος. Αυτές οι πληροφορίες μπορούν να χρησιμοποιηθούν για κάποια επίθεση στην εφαρμογή, για παράδειγμα, μέσω brute force επίθεσης ή επίθεσης προκαθορισμένου ονόματος χρήστη/κωδικού πρόσβασης.

Ο χρήστης θα πρέπει να αλληλεπιδράσει με τον μηχανισμό αυθεντικοποίησης της εφαρμογής για να καταλάβει αν η αποστολή συγκεκριμένων αιτημάτων προκαλεί την εφαρμογή να απαντήσει με διαφορετικούς τρόπους. Αυτό το ζήτημα υφίσταται γιατί οι πληροφορίες προκύπτουν από τον server εφαρμογής ή τον web server, όταν δώσουμε ένα έγκυρο όνομα χρήστη είναι διαφορετικές από ότι όταν δώσουμε ένα μη έγκυρο. Σε κάποιες περιπτώσεις λαμβάνουμε ένα μήνυμα το οποίο αποκαλύπτει εάν τα συνθηματικά που χρησιμοποιήσαμε είναι εσφαλμένα ποιο από το όνομα χρήστη ή τον κωδικό πρόσβασης είναι λάθος. Μερικές φορές μπορούμε να αριθμήσουμε τους χρήστες που υπάρχουν στέλνοντας ένα όνομα χρήστη και ένα κενό κωδικό πρόσβασης.

BLACK BOX TESTING

Σε μια black box δοκιμή, δεν γνωρίζουμε τίποτα για τα για τη συγκεκριμένη εφαρμογή, το όνομα χρήστη, τη λογική της εφαρμογής και τα μηνύματα λάθους στην αρχική σελίδα εισαγωγής ή τις διαδικασίες ανάκτησης του κωδικού πρόσβασης. Αν η εφαρμογή είναι ευπαθής, λαμβάνουμε ένα μήνυμα που αποκαλύπτει άμεσα ή έμμεσα κάποιες πληροφορίες χρήσιμες για την καταμέτρηση των χρηστών.

Μήνυμα HTTP απάντησης

- **Έλεγχος για έγκυρο χρήστη/ σωστό κωδικό πρόσβασης**

Καταγράφουμε την απάντηση του server όταν υποβάλλουμε ένα έγκυρο UserID και έναν έγκυρο κωδικό πρόσβασης.

Αναμενόμενο Αποτέλεσμα:

Χρησιμοποιώντας το WebScarab προσέχουμε τις πληροφορίες που λαμβάνονται από την επιτυχή αυθεντικοποίηση (HTTP απάντηση 200, μήκος απάντησης).

- **Έλεγχος για έγκυρο όνομα χρήστη/λάθος κωδικό πρόσβασης**

Ο ελεγκτής τώρα θα πρέπει να δοκιμάσει να εισάγει ένα έγκυρο userID και έναν λάθος κωδικό πρόσβασης και να καταγράψει το μήνυμα λάθους που εμφανίζει η εφαρμογή.

Αναμενόμενο Αποτέλεσμα:

Από τον browser περιμένουμε ένα μήνυμα παρόμοιο με το παρακάτω:

Εικόνα : **Μήνυμα αποτυχίας αυθεντικοποίησης**

ή κάτι όπως αυτό:

Εικόνα : **Μήνυμα λάθους που επιστρέφεται από τον server.**

Αντίθετα οποιοδήποτε μήνυμα που αποκαλύπτει την ύπαρξη ενός χρήση είναι παρόμοιο με αυτό που ακολουθεί:

Login for User foo: invalid password

Χρησιμοποιώντας το WebScarab, πρέπει να προσέξουμε τις πληροφορίες που ανακτώνται από μια ανεπιτυχή προσπάθεια αυθεντικοποίησης (HTTP απάντηση 200, μήκος απάντησης)

- **Έλεγχος για όνομα χρήστη που δεν υπάρχει**

Εδώ ο ελεγκτής θα πρέπει να δοκιμάσει να εισάγει ένα μη έγκυρο όνομα χρήστη και έναν εσφαλμένο κωδικό πρόσβασης και να καταγράψει την απάντηση του server (θα πρέπει να είμαστε σίγουροι πως το όνομα χρήστη είναι μη έγκυρο στην εφαρμογή μας).

Αναμενόμενο αποτέλεσμα:

Αν εισάγουμε ένα userID που δεν υπάρχει θα λάβουμε ένα μήνυμα παρόμοιο με το παρακάτω:

Εικόνα : **Μήνυμα μη ενεργού χρήστη**

ή ένα μήνυμα σαν αυτό:

Login failed for User foo: invalid Account

Γενικά η εφαρμογή θα πρέπει να ανταποκριθεί με το ίδιο μήνυμα λάθους και μήκος μηνύματος στα διαφορετικά εσφαλμένα αιτήματα. Αν παρατηρήσουμε ότι οι απαντήσεις δεν είναι οι ίδιες θα πρέπει να ερευνήσουμε και να ανακαλύψουμε το κλειδί που δημιουργεί τη διαφορά ανάμεσα στις δύο απαντήσεις. Για παράδειγμα:

- Αίτημα client: Έγκυρο όνομα χρήστη/λάθος κωδικός πρόσβασης
  Απάντηση server: «ο κωδικός πρόσβασης είναι εσφαλμένος»
- Αίτημα client: Λάθος χρήστης/ λάθος κωδικός πρόσβασης
  Απάντηση server: «Ο χρήστης δεν αναγνωρίζεται»

Οι παραπάνω απαντήσεις κάνουν τον client να καταλάβει ότι για το πρώτο αίτημα έχουνε ένα έγκυρο όνομα χρήστη. Έτσι μπορούμε να αλληλεπιδράσουμε με την εφαρμογή αιτούμενοι ένα σύνολο πιθανών userIDs παρακολουθώντας την απάντηση. Βλέποντας τη δεύτερη απάντηση, καταλαβαίνουμε με τον ίδιο τρόπο ότι κατέχουμε ένα έγκυρο όνομα χρήστη. Άρα μπορούμε να αλληλεπιδράσουμε με τον ίδιο τρόπο και να δημιουργήσουμε μια λίστα έγκυρων userID κοιτάζοντας στις απαντήσεις του server.

Άλλοι τρόποι για να απαριθμήσουμε τους χρήστες

Μπορούμε να απαριθμήσουμε τους χρήστες με διάφορους τρόπους όπως:

- **Ανάλυση του κώδικα λάθους που λαμβάνουμε στις αρχικές σελίδες σύνδεσης:**
  Κάποιες εφαρμογές ιστού επιστρέφουν ένα συγκεκριμένο κωδικό σφάλματος που μπορούμε να τον αναλύσουμε.
- **Ανάλυση και ανακατεύθυνση των URL:**
  Για παράδειγμα:
  http://www.foo.com/err.jsp?User=baduser&Error=0 http://www.foo.com/err.jsp?User=gooduser&Error=2

Όπως μπορούμε να δούμε παραπάνω όταν δώσουμε ένα userID και ένα κωδικό χρήστη στην εφαρμογή ιστού βλέπουμε μια ένδειξη μηνύματος ότι ένας σφάλμα εμφανίστηκε στο URL. Στην πρώτη περίπτωση δώσαμε ένα κακό userID και έναν κακό κωδικό πρόσβασης. Στην δεύτερη, έναν καλό χρήστη και έναν κακό κωδικό πρόσβασης έτσι μπορούμε να αναγνωρίσουμε τον έγκυρο χρήστη.

- **Διερεύνηση του URI**
  Μερικές φορές ένας web server απαντά διαφορετικά αν λαμβάνει ένα αίτημα για την ύπαρξη ενός καταλόγου ή όχι. Για παράδειγμα σε κάποιες πύλες(portals) όπου κάθε χρήστης σχετίζεται με έναν κατάλογο καν προσπαθήσουμε να προσπελάσουμε έναν κατάλογο που υπάρχει μπορεί να πάρουμε ένα σφάλμα του web server. Ένα πολύ κοινό σφάλμα που παίρνουμε από τον web server είναι:

  403 Forbidden error code και 404 Not found error code

Παράδειγμα

http://www.foo.com/account1- από τον web server: 403 Forbidden

http://www.foo.com/account2 - παίρνουμε από τον web server: 404 file Not Found

Στην πρώτη περίπτωση ο χρήστης υπάρχει, αλλά δεν μπορούμε να δούμε την ιστοσελίδα, στη δεύτερη περίπτωση το όνομα χρήστη account2 δεν υπάρχει. Η συλλογή αυτών των πληροφοριών μπορεί να απαριθμήσει τους χρήστες.

- **Ανάλυση των τίτλων των web σελίδων**

Μπορούμε να πάρουμε σημαντικές πληροφορίες από τον τίτλο της web σελίδας, όπου μπορούμε να πετύχουμε ένα συγκεκριμένο κωδικό σφάλματος ή ένα μήνυμα λάθους που αποκαλύπτει εάν το πρόβλημα έχει να κάνει με το όνομα χρήστη ή τον κωδικό πρόσβασης. Για παράδειγμα, αν δεν μπορούμε να αυθεντικοποιηθούμε σε μια εφαρμογή και λαμβάνουμε μια web σελίδα που ο τίτλος της είναι παρόμοιος με:

Invalid user

Invalid authentication

- **Ανάλυση του μηνύματος που λαμβάνεται από τις διαδικασίες ανάκτησης**

Όταν χρησιμοποιούμε μια διαδικασία ανάκτησης, η εφαρμογή που είναι ευπαθής μπορεί να επιστρέψει ένα μήνυμα το οποίο αποκαλύπτει αν ένα όνομα χρήστη υπάρχει ή όχι. Για παράδειγμα ένα μήνυμα όπως το παρακάτω:

Invalid username: email address is not valid or the specified user was not found.

Valid username: Your recovery password has been successfully sent.

- **«Φιλικό» Μήνυμα Σφάλματος 404**

Όταν στέλνουμε αίτημα για ένα χρήστη σε έναν κατάλογο που δεν υπάρχει, δε λαμβάνουμε πάντα κωδικό σφάλματος 404, ίσως λάβουμε '200 OK' με μια εικόνα οπότε μπορούμε να θεωρήσουμε ότι όταν λαμβάνουμε τη συγκεκριμένη εικόνα ο χρήστης δεν υπάρχει. Αυτή η λογική μπορεί να εφαρμοστεί και σε άλλες απαντήσεις των web server. Το τέχνασμα είναι μια καλή ανάλυση των μηνυμάτων των web server και των server εφαρμογής.

- **Προσπάθεια να «μαντέψουμε» χρήστες**

Σε κάποιες περιπτώσεις τα userIDs δημιουργούνται με κάποιες συγκεκριμένες πολιτικές του διαχειριστή ή της εταιρείας. Για παράδειγμα μπορούμε να δούμε ένα userID να έχει δημιουργηθεί με την εξής ακολουθιακή σειρά:

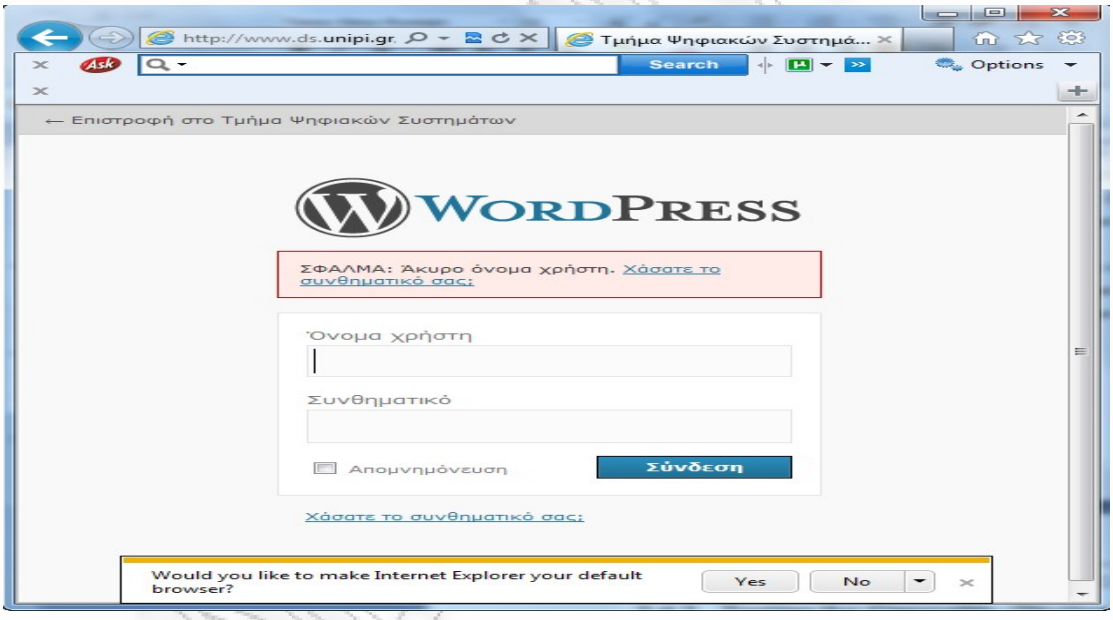
CN000100 CN000101 ….

Κάποιες φορές τα ονόματα των χρηστών δημιουργούνται με ένα REALM ψευδώνυμο και μετά έναν αύξοντα αριθμό:

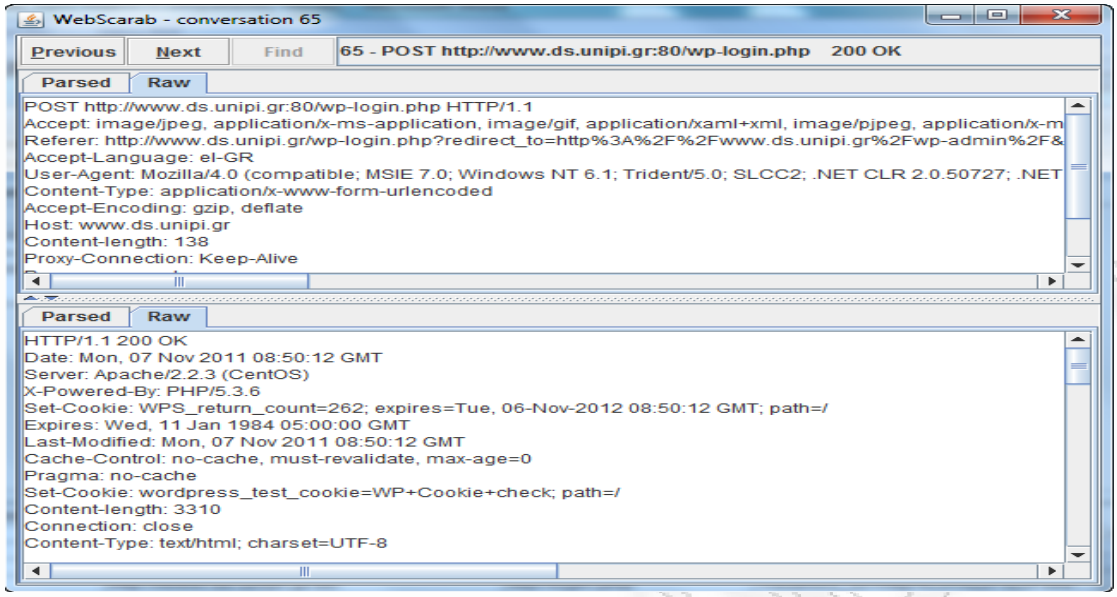
R1001 – user 001 for REALM1 R2001 – user 001 for REALM2

Άλλες περιπτώσεις είναι τα userIDs που σχετίζονται με τους αριθμούς πιστωτικών καρτών ή γενικά αριθμοί που βασίζονται σε ένα κάποιο πρότυπο. Στο παραπάνω παράδειγμα μπορούμε να δημιουργήσουμε shell scripts τα οποία συνθέτουν userIDs και υποβάλλουν κάποιο αίτημα μέσω ενός εργαλείου όπως πχ το wget για να αυτοματοποιήσουμε ένα web ερώτημα ώστε να διακρίνουμε τα έγκυρα userIDs. Ξανά, βέβαια, μπορούμε να μαντέψουμε ένα όνομα χρήστη μέσω των πληροφοριών που παίρνουμε από ένα LDAP ερώτημα ή από πληροφορίες μέσω της google αναζήτησης για παράδειγμα για ένα συγκεκριμένο domain. Η αναζήτηση μέσω του google μπορεί να βοηθήσει ώστε να βρούμε τους χρήστες κάποιου domain μέσω συγκεκριμένων ερωτημάτων ή μέσω ενός απλού shell script ή εργαλείου.

Εξετάζουμε τώρα κατά πόσο είναι ευπαθής η δική μας εφαρμογή σε μια επίθεση τύπου καταμέτρησης χρηστών (user enumeration). Θα ξεκινήσουμε τον έλεγχό μας προσπαθώντας να εισάγουμε ένα λάθος όνομα χρήστη και έναν λάθος κωδικό πρόσβασης και να δούμε την απάντηση του server.

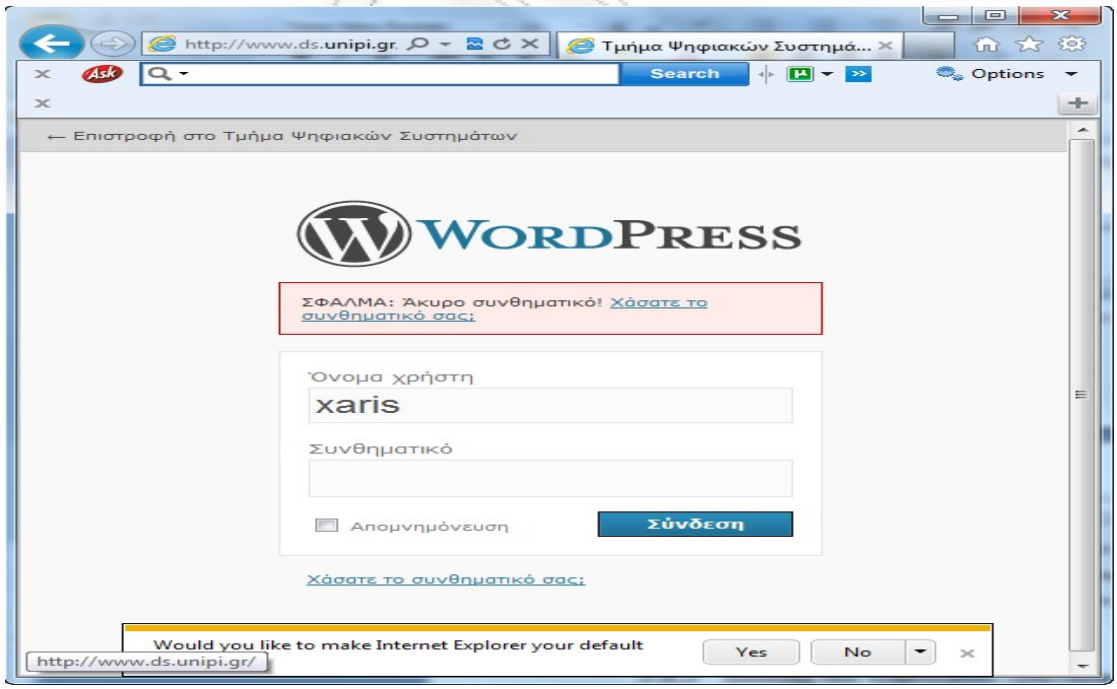


Εικόνα **: Η απάντηση του server για μη έγκυρο όνομα χρήστη και μη έγκυρο κωδικό πρόσβασης**



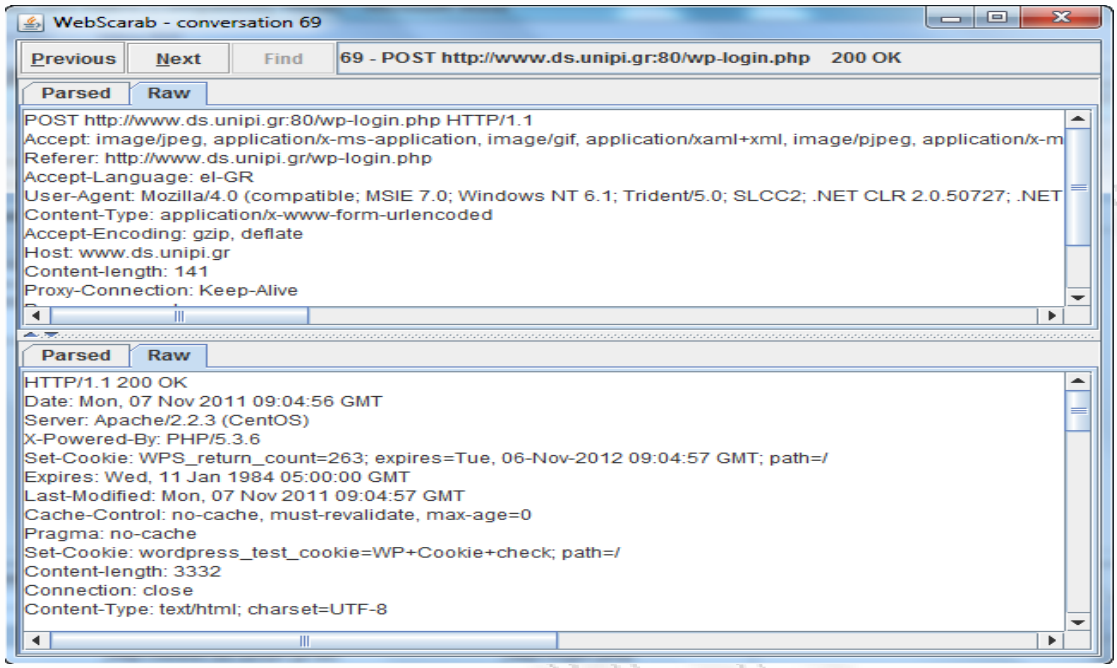
Εικόνα : **Το μήνυμα της απάντησης του server όπως εμφανίζεται στο WebScarab.**

Δοκιμάζουμε τώρα να κάνουμε εισαγωγή στο σύστημα της εφαρμογής μας χρησιμοποιώντας ένα όνομα χρήστη, που προκαταβολικά ξέρουμε ότι υπάρχει, και έναν λάθος κωδικό πρόσβασης για να δούμε πως ανταποκρίνεται ο server.



Εικόνα : **Η απάντηση που λαμβάνουμε από τον server κατά την εισαγωγή έγκυρο ονόματος χρήστη και εσφαλμένου κωδικού πρόσβασης.**

Στον proxy εμφανίζεται όπως φαίνεται στην παρακάτω εικόνα:



Εικόνα : **Το μήνυμα της απάντησης του server όπως εμφανίζεται στο WebScarab.**

Μετά την δοκιμή των διάφορων τρόπων αυθεντικοποίησης στον server της εφαρμογής διαπιστώνουμε πως ο server δεν απαντά με τον ίδιο τρόπο στους διάφορους αυτούς τρόπους προσπαθειών εισαγωγής στο σύστημα. Αυτό είναι κάτι που δείχνει εύκολα την αδυναμία του να αποκρύπτει τα ονόματα των χρηστών του και τον κάνει ιδιαίτερα ευπαθή σε επιθέσεις brute force με κάποιο λεξικό ονομάτων χρηστών είτε χρησιμοποιώντας διάφορα άλλα ονόματα χρηστών που μπορεί εύκολα κάποιος να ανακαλύψει ότι ισχύουν για την συγκεκριμένη εφαρμογή δεδομένης του συστήματος με το οποίο αναπτύχθηκε (Wordpress). Συνεπώς είναι εύκολη η καταγραφή μιας λίστας με έγκυρα ονόματα χρηστών.

3.4.3 Testing for Guessable (Dictionary) User Account

Οι εφαρμογές ιστού σήμερα, λειτουργούν πάνω σε δημοφιλή software, ανοιχτού ή εμπορικού λογισμικού, που εγκαθίστανται στους servers και απαιτούν διαμόρφωση από τον εκάστοτε διαχειριστή τους. Επιπλέον οι περισσότερες hardware συσκευές όπως routers δικτύου, servers βάσεων δεδομένων, κλπ., προσφέρουν βασισμένες στο web διαμορφώσεις ή διεπαφές διαχειριστών. Συχνά, αυτές οι εφαρμογές ιστού δεν διαμορφώνονται κατάλληλα και τα προεπιλεγμένα (default) πιστοποιητικά, που παρέχονται για την αυθεντικοποίηση, δεν ενημερώνονται ποτέ. Αυτοί οι προεπιλεγμένοι συνδυασμοί ονόματος χρήστη-κωδικού πρόσβασης είναι γνωστοί ευρέως στους ελεγκτές διείσδυσης και τους κακόβουλους hackers, τους οποίους συνδυασμούς μπορούν να χρησιμοποιήσουν, για να αποκτήσουν πρόσβαση στην εσωτερική υποδομή δικτύων ή και να αποκτήσουν προνόμια και να υποκλέψουν

δεδομένα. Αυτό το πρόβλημα ισχύει για το λογισμικό ή και τις συσκευές οι οποίες παρέχουν ενσωματωμένους λογαριασμούς, που δεν μπορούν να διαγραφούν (non-removable) και σε ορισμένες περιπτώσεις χρησιμοποιούν κενούς (blank) κωδικούς πρόσβασης ως προεπιλεγμένα πιστοποιητικά.

Πηγές αυτού του προβλήματος είναι το άπειρο τεχνικό προσωπικό, το οποίο είναι απληροφόρητο για την σημασία της αλλαγής των προεπιλεγμένων κωδικών πρόσβασης σε εγκατεστημένα στοιχεία του συστήματος, πχ οι προγραμματιστές, που αφήνουν ανοιχτές πόρτες με σκοπό την εύκολη και γρήγορη πρόσβαση στις εφαρμογές και μετά ξεχνούν να τις κλείσουν, οι διαχειριστές συστήματος και οι χρήστες, που επιλέγουν ένα εύκολο όνομα και κωδικό πρόσβασης, καθώς και οι εφαρμογές με ενσωματωμένους και μόνιμους λογαριασμούς χρήστη με προεπιλεγμένα ονόματα χρηστών και κωδικούς πρόσβασης. Ένα ακόμη πρόβλημα είναι οι κενοί κωδικοί πρόσβασης, το οποίο είναι αποτέλεσμα της άγνοιας της ασφάλειας, καθώς επίσης και της επιθυμίας για διευκόλυνση της διαχείρισης του συστήματος.

BLACK BOX TESTING

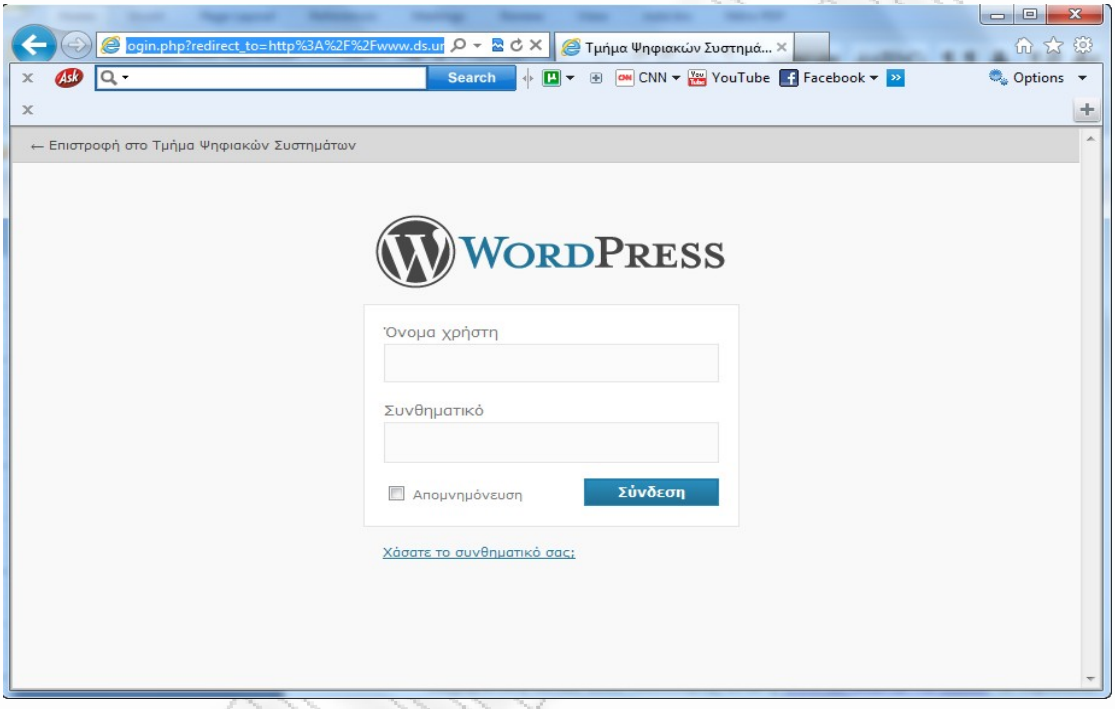
Στην black box τεχνική ελέγχου δεν γνωρίζουμε τίποτα για την εφαρμογή, την τεχνική υποδομή και την υπάρχουσα πολιτική ασφάλειας, που αφορά τα ονόματα χρηστών και τους κωδικούς πρόσβασης. Όταν ελέγχουμε μια γνωστή διεπαφή (interface) εφαρμογής, όπως είναι για παράδειγμα ένα Cisco router web interface, πρέπει να ελέγξουμε τα γνωστά ονόματα χρήστη και τους κωδικούς πρόσβασης για αυτές τις συσκευές. Όταν έχουμε να αντιμετωπίσουμε μια απλή εφαρμογή, στην οποία δεν έχουμε έναν κατάλογο προεπιλεγμένων και κοινών λογαριασμών χρηστών, χρειάζεται να την εξετάσουμε χειροκίνητα ακολουθώντας τα παρακάτω βήματα:

- Δοκιμάζουμε τα ακόλουθα ονόματα χρήστη - "admin", "administrator", "root", "system", ή "super". Αυτά είναι δημοφιλή μεταξύ των διαχειριστών συστημάτων και χρησιμοποιούνται πολύ συχνά. Επιπλέον θα μπορούσαμε να δοκιμάσουμε τα "qa", "test", "test1", "testing", και παρόμοια ονόματα. Απαιτείται να δοκιμάσουμε οποιοδήποτε συνδυασμό των ανωτέρω και στο όνομα χρήστη και στο πεδίο κωδικού πρόσβασης. Εάν η εφαρμογή είναι ευάλωτη στη διερεύνηση ονόματος χρήστη και κατορθώσουμε επιτυχώς να προσδιορίσουμε οποιοδήποτε από τα ανωτέρω ονόματα χρήστη, τότε είναι ανάγκη κατά παρόμοιο τρόπο να προσπαθήσουμε για τους κωδικούς πρόσβασης.
- Οι χρήστες διαχείρισης της εφαρμογής συχνά ονομάζονται μετά από τη δημιουργία της εφαρμογής. Αυτό σημαίνει ότι εάν εξετάζουμε μια εφαρμογή η οποία ονομάζεται "Obscurity", οφείλουμε να δοκιμάσουμε να χρησιμοποιήσουμε τα obscurity-obscurity ως όνομα χρήστη και κωδικό πρόσβασης.
- Κατά την εκτέλεση μιας δοκιμής για έναν πελάτη, χρειάζεται να προσπαθήσουμε να χρησιμοποιήσουμε τα ονόματα των επαφών που έχουμε ως ονόματα χρήστη.
- Η εξέταση της σελίδας εγγραφής χρηστών (User Registration) ενδέχεται να βοηθήσει στον καθορισμό του αναμενόμενου σχήματος και μήκους των ονομάτων χρήστη και των κωδικών πρόσβασης της εφαρμογής. Εάν δεν υπάρχει σελίδα εγγραφής χρηστών, χρειάζεται να

προσδιορίσουμε, αν ο οργανισμός χρησιμοποιεί μια τυποποιημένη σύμβαση για τα ονόματα χρηστών.

- Είναι αναγκαίο να προσπαθήσουμε να χρησιμοποιήσουμε όλα τα ανωτέρω ονόματα χρήστη με κενούς κωδικούς πρόσβασης.

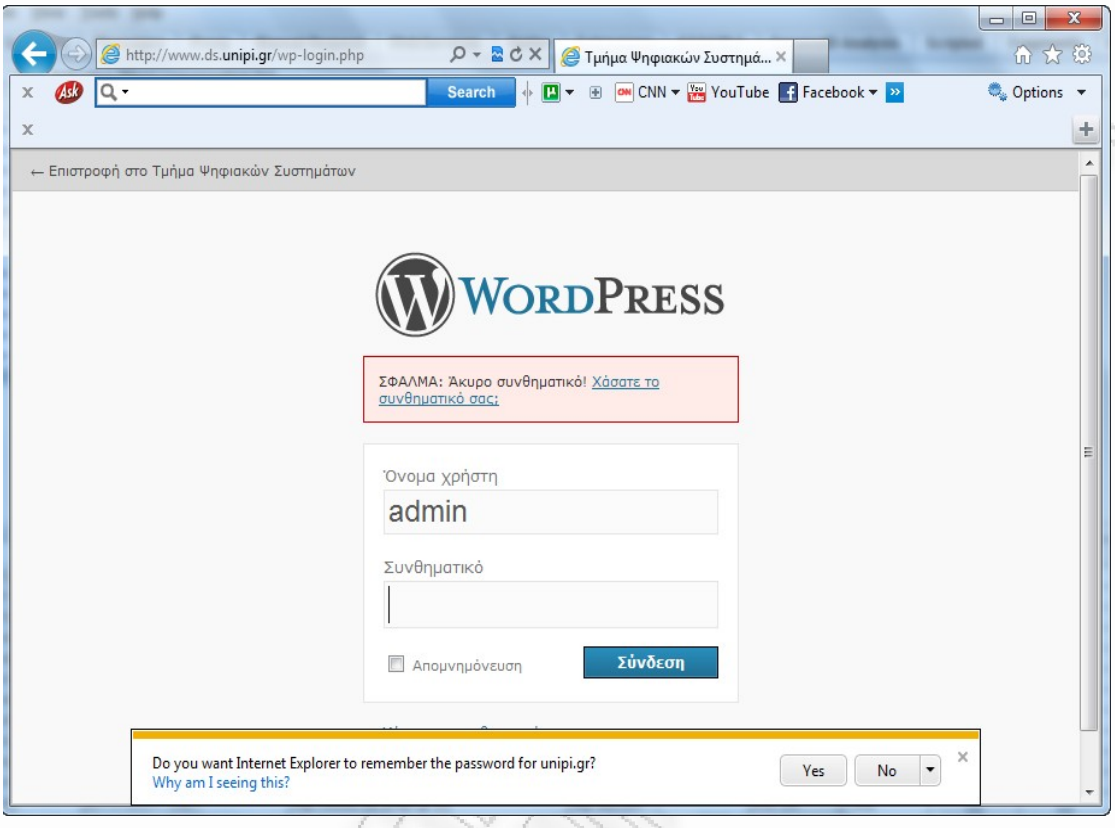
Από τον έλεγχο spidering που πραγματοποιήσαμε στο κομμάτι του Information gathering διαπιστώσαμε πως η σελίδα αυθεντικοποίησης του διαχειριστή της εφαρμογής είναι η www.ds.unipi.gr/wp-admin. Αυτό μπορούμε εύκολα να το βρούμε επίσης μέσω της αναζήτησης που αφορά τη σελίδα διαχείρισης για εφαρμογές που έχουν αναπτυχθεί σε wordpress. Η παρακάτω εικόνα δείχνει τη σελίδα αυθεντικοποίησης.



Εικόνα : **Η σελίδα αυθεντικοποίησης του διαχειριστή της εφαρμογής http://www.ds.unipi.gr**

Μετά τη δοκιμή όλων των προκαθορισμένων(default) συνθηματικών διαπιστώσαμε πως αναφορικά με το όνομα χρήστη χρησιμοποιείται ένα από τα «κοινά» ονόματα χρήστη συγκεκριμένα το "admin" όμως δεν ισχύει κάτι τέτοιο και για τον κωδικό πρόσβασης κάτι που θα ήταν καταστροφικό για την εφαρμογή που εξετάζουμε. Σε περίπτωση που γίνουν και τα δύο συνθηματικά αυτής της σελίδας αυθεντικοποίησης του διαχειριστή γνωστά τότε η εφαρμογή μπορεί πολύ εύκολα να πάψει να είναι υπό την πλήρη κατοχή και διαχείρισή μας καθώς ο επιτιθέμενος έχει τη δυνατότητα να αλλάξει τα συνθηματικά μιας και θα έχει πλήρη δικαιώματα και να μας «κλειδώσει απ' έξω». Για την εφαρμογή που εξετάζουμε δεν ισχύει κάτι τέτοιο όμως και μόνο η χρήση ενός τόσο τετριμμένου, για τα δεδομένα της ασφάλειας πληροφοριών, ονόματος χρήστη θεωρείται ευπάθεια για μια εφαρμογή ιστού ιδιαιτέρως αν συνδυάσουμε το γεγονός αυτό με άλλους ελέγχους που είδαμε προηγουμένως κατά την εργασία αυτή.

Παρακάτω φαίνεται η εικόνα με το μήνυμα που εμφανίζει η σελίδα αυθεντικοποίησης του διαχειριστή κατά την ορθή εισαγωγή του ονόματος χρήστη “admin” και λάθους κωδικού πρόσβασης.



Εικόνα : **Η σελίδα αυθεντικοποίησης του διαχειριστή με σωστό όνομα χρήστη και λάθος κωδικό πρόσβασης.**

3.4.4 Brute Force Testing

Η μέθοδος Brute force αποτελείται από τη συστηματική διερεύνηση όλων των πιθανών υποψηφίων λύσεων. Στον έλεγχο των εφαρμογών ιστού το πρόβλημα, που συχνότερα ενδέχεται να αντιμετωπίσουμε συνδέεται με την ανάγκη ενός έγκυρου λογαριασμού χρήστη, για να έχουμε πρόσβαση στο εσωτερικό μέρος της εφαρμογής. Επομένως, πρόκειται να ελέγξουμε τους διαφορετικούς τύπους σχημάτων αυθεντικοποίησης και την αποτελεσματικότητα των διαφορετικών επιθέσεων brute force.

Μια μεγάλη πλειοψηφία εφαρμογών παρέχει κάποιο τρόπο αυθεντικοποίησης των χρηστών τους. Γνωρίζοντας η εφαρμογή την ταυτότητα του χρήστη είναι δυνατό να δημιουργεί προστατευόμενες ζώνες ή γενικότερα είναι δυνατό η εφαρμογή να συμπεριφέρεται διαφορετικά ανάλογα με τη σύνδεση διαφορετικών χρηστών. Στην πραγματικότητα υπάρχουν διάφορες μέθοδοι αυθεντικοποίησης χρήστη όπως είναι τα πιστοποιητικά, οι βιομετρικές συσκευές, οι κωδικοί πρόσβασης μιας χρήσης (OTP tokens – One Time Password tokens), αλλά στις εφαρμογές ιστού βρίσκουμε συνήθως ένα συνδυασμό της ταυτότητας χρηστών και του κωδικού πρόσβασης τους. Επομένως, είναι δυνατό να πραγματοποιηθεί μια επίθεση για να ανακτηθεί ένας έγκυρος λογαριασμός και ένας κωδικός πρόσβασης χρήστη προσπαθώντας

να απαριθμήσουμε πολλούς λογαριασμούς π.χ. επίθεση λεξικών ή ολόκληρο το διάστημα των πιθανών υποψήφιων λογαριασμών.

Μετά από μια επιτυχημένη επίθεση brute force, ένας κακόβουλος χρήστης θα μπορούσε να έχει πρόσβαση σε:

- Εμπιστευτικές πληροφορίες / δεδομένα
    - Ιδιωτικά τμήματα μιας εφαρμογής ιστού θα μπορούσαν να αποκαλύψουν εμπιστευτικά έγγραφα, δεδομένα των προφίλ των χρηστών, οικονομική κατάσταση, τραπεζικά στοιχεία, σχέσεις χρηστών, κλπ.
- Πίνακες διαχείρισης
    - Αυτά τα τμήματα χρησιμοποιούνται από τους webmasters για να διαχειριστούν το περιεχόμενο της εφαρμογής (τροποποίηση, διαγραφή, εισαγωγή), τις ρυθμίσεις των χρηστών, τα δικαιώματα των χρηστών, κλπ.
- Διαθεσιμότητα περαιτέρω φορέων επίθεσης
    - Ιδιωτικά τμήματα μιας εφαρμογής ιστού θα μπορούσαν να κρύψουν επικίνδυνες ευπάθειες και να περιέχουν υψηλότερου επιπέδου λειτουργίες που δεν είναι διαθέσιμες στους κοινούς χρήστες.

Για να εφαρμόσουμε διαφορετικές επιθέσεις brute force, είναι σημαντικό να ανακαλυφθεί ο τύπος της μεθόδου αυθεντικοποίησης, που χρησιμοποιείται από την εφαρμογή, γιατί οι τεχνικές και τα εργαλεία, που χρησιμοποιούνται, ποικίλουν ανάλογα με τη μέθοδο αυθεντικοποίησης.

2 Ανακάλυψη των Μεθόδων Αυθεντικοποίησης

Οι δύο συνηθέστερες μέθοδοι, που χρησιμοποιούνται στη web αυθεντικοποίηση, είναι οι ακόλουθες:

- HTTP Αυθεντικοποίηση: η οποία διακρίνεται περαιτέρω στη

o Βασική αυθεντικοποίηση πρόσβασης (Basicc Access Authentication)

o Περιληπτική αυθεντικοποίηση πρόσβασης (Digest Access Authentication)

- Αυθεντικοποίηση βασισμένη σε φόρμα HTML

Οι ακόλουθες ενότητες παρέχουν κάποιες πληροφορίες για τον προσδιορισμό του μηχανισμού αυθεντικοποίησης που χρησιμοποιείται κατά τη διάρκεια μιας δοκιμής διείσδυσης. Στη συνέχεια θα εξετάσουμε πιο αναλυτικά τις δύο συνηθέστερες μεθόδους αυθεντικοποίησης και τα χαρακτηριστικά τους.

3 HTTP Αυθεντικοποίηση

Όπως αναφέραμε, υπάρχουν δύο εγγενή σχέδια HTTP αυθεντικοποίησης, τα οποία είναι το Basic και το Digest:

- Βασική αυθεντικοποίηση πρόσβασης (Basic)

Η Βασική αυθεντικοποίηση πρόσβασης υποθέτει ότι ο πελάτης θα προσδιοριστεί με ένα όνομα σύνδεσης (π.χ. "username") και ένα κωδικό πρόσβασης (π.χ. "password"). Όταν ο browser του πελάτη έχει πρόσβαση αρχικά σε μια ιστοσελίδα χρησιμοποιώντας αυτό το σχήμα, ο web server θα απαντήσει με μια απάντηση 401, που περιέχει την ετικέτα "WWW-Authenticate" συμπεριλαμβανομένης μιας τιμής " Basic " και το όνομα του προστατευμένου τομέα (π.χ. WWW-Authenticate: Basic realm = "wwwProtectedSite"). Έπειτα ο browser του πελάτη θα προτρέψει το χρήστη για το όνομα και τον κωδικό πρόσβασης σύνδεσής του. Αμέσως μετά ο browser απαντά στο web server με μια ετικέτα "Authorization", συμπεριλαμβανομένης της τιμής " Basic " και του συνδυασμού (base64-encoded) του ονόματος χρήστη και του κωδικού πρόσβασης (π.χ. Authorization: Basic b3dhc3A6cGFzc3dvcmQ=). Δυστυχώς η απάντηση αυθεντικοποίησης μπορεί να αποκωδικοποιηθεί εύκολα, εάν ένας επιτιθέμενος χρησιμοποιήσει εργαλεία για sniffing στη μετάδοση.

Δοκιμή αίτησης και απάντησης:

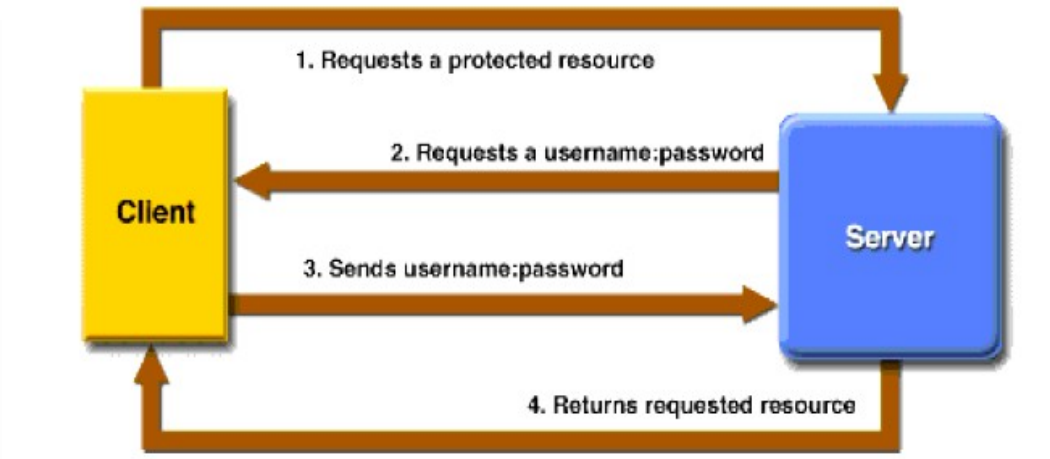
1. Ο πελάτης στέλνει το τυποποιημένο HTTP αίτημα για κάποιο πόρο:
   GET /members/docs/file.pdf HTTP/1.1
   Host: target
2. Ο web server δηλώνει ότι ο ζητούμενος πόρος βρίσκεται σε ένα προστατευμένο κατάλογο.
3. Ο serverr στέλνει την απάντηση με την έγκριση HTTP 401, που απαιτείται:

HTTP/1.1 401 Authorization RequiredDate: Sat, 04 Nov 2006 12:52:40 GMTWWW-Authenticate: Basic realm="User Realm"Content-Length: 401Keep-Alive: timeout=15, max=100Connection: Keep-AliveContent-Type: text/html; charset=iso-8859-1

4. Ο browser προβάλει ένα pop-up παράθυρο και προτρέπει για εισαγωγή του ονόματος χρήστη και του κωδικού πρόσβασης.
5. Ο πελάτης υποβάλλει εκ νέου το αίτημα HTTP συμπεριλαμβανομένων των διαπιστευτηρίων (credentials) του:
   GET /members/docs/file.pdf HTTP/1.1
   Host: target
   Authorization: Basic b3dhc3A6cGFzc3dvcmQ=
6. Ο server συγκρίνει τις πληροφορίες πελάτη με τον κατάλογο διαπιστευτηρίων, που διατηρεί.
7. Εάν τα διαπιστευτήρια ισχύουν, ο server στέλνει το ζητούμενο περιεχόμενο.

Εάν η αυθεντικοποίηση αποτύχει, ο server στέλνει εκ νέου τον κώδικα κατάστασης HTTP 401 στην ετικέτα απάντησης. Εάν ο χρήστης επιλέξει ακύρωση, τότε ο browser θα δείξει πιθανώς ένα μήνυμα λάθους.

Εάν ένας επιτιθέμενος είναι σε θέση να παρεμποδίσει το αίτημα από το βήμα 5, η συμβολοσειρά "b3dhc3A6cGFzc3dvcmQ=", θα μπορούσε να αποκωδικοποιηθεί ως ακολούθως (Base64 αποκωδικοποίηση): username: password



Εικόνα : **Μηχανισμός HTTP Basic Αυθεντικοποίησης.**

- Περιληπτική αυθεντικοποίηση πρόσβασης (Digest)

Η Περιληπτική αυθεντικοποίηση πρόσβασης αποτελεί επέκταση της Βασικής αυθεντικοποίησης πρόσβασης με τη χρησιμοποίηση ενός μονόδρομου κρυπτογραφικού hashing αλγορίθμου (MD5) για να κρυπτογραφηθούν τα στοιχεία αυθεντικοποίησης και την πρόσθεση μιας τιμής μίας χρήσεως (nonce), που τίθεται από τον web server. Αυτή η τιμή χρησιμοποιείται από το browser του πελάτη στον υπολογισμό μιας απάντησης κωδικού πρόσβασης. Ενώ ο κωδικός πρόσβασης αποκρύπτεται με την χρήση κρυπτογραφικού hashing και η χρήση της τιμής nonce αποκλείει την απειλή μιας επίθεσης επανάληψης, το όνομα σύνδεσης υποβάλλεται με τη μορφή απλού κειμένου.

Δοκιμή αίτησης και απάντησης:

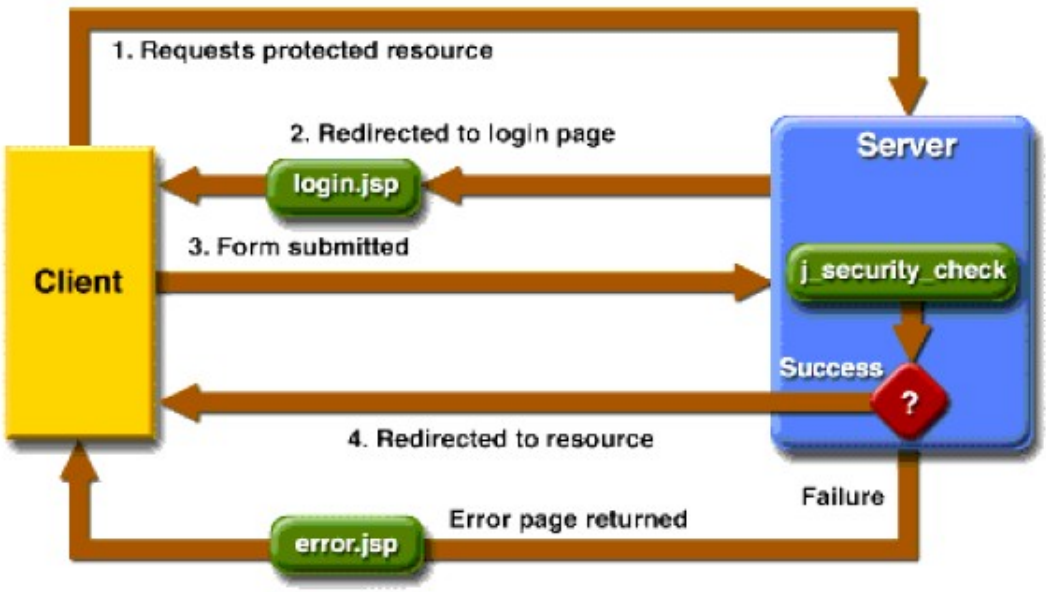
1. Εδώ είναι ένα παράδειγμα της αρχικής επικεφαλίδας απάντησης κατά το χειρισμό ενός HTTP Digest στόχου:
   HTTP/1.1 401 Unauthorized
   WWW-Authenticate: Digest realm="OwaspSample",
   nonce="Ny8yLzIwMDIgMzoyNjoyNCBQTQ",
   opaque="0000000000000000", \
   stale=false,
   algorithm=MD5,
   qop="auth"
2. Οι επόμενες επικεφαλίδες απάντησης με τα έγκυρα διαπιστευτήρια είναι, όπως παρουσιάζονται παρακάτω:
   GET /example/owasp/test.asmx HTTP/1.1
   Accept: */*
   Authorization: Digest username="username",

realm="OwaspSample",
qop="auth",
algorithm="MD5",
uri="/example/owasp/test.asmx",
nonce="Ny8yLzIwMDIgMzoyNjoyNCBQTQ",
nc=00000001,
cnonce="c51b5139556f939768f770dab8e5277a",
opaque="0000000000000000",
response="2275a9ca7b2dadf252afc79923cd3823"

4 Αυθεντικοποίηση Βασισμένη σε Φόρμα HTML

Ενώ και τα δύο σχήματα HTTP αυθεντικοποίησης μπορούν να εμφανιστούν κατάλληλα για εμπορική χρήση μέσω του Internet ιδιαίτερα, όταν χρησιμοποιούντα πάνω σε μια κρυπτογραφημένη SSL σύνοδο, πολλοί οργανισμοί ή επιχειρήσεις έχουν επιλέξει να χρησιμοποιούν HTML και διαδικασίες αυθεντικοποίησης επιπέδου εφαρμογής HTML, προκειμένου να παρασχεθεί μια πιο περίπλοκη διαδικασία αυθεντικοποίησης. Ο πηγαίος κώδικας, που λαμβάνεται από μια φόρμα HTML, είναι ως εξής:

```
<form method="POST" action="login">
    <input type="text" name"username">
    <input type="password" name="password">
    </form>
```



Εικόνα : **Μηχανισμός Form-Based Αυθεντικοποίησης.**

5 Επιθέσεις με τη Μέθοδο Brute force

Αφού έχουμε απαριθμήσει τους διαφορετικούς τύπους μεθόδων αυθεντικοποίησης για μια εφαρμογή ιστού, θα εξηγήσουμε τώρα διάφορους τύπους επιθέσεων brute force.

- Επίθεση λεξικών (Dictionary attacks)

Οι επιθέσεις λεξικών αποτελούνται από αυτοματοποιημένα scripts και εργαλεία, που προσπαθούν να μαντέψουν το όνομα χρήστη και τον κωδικό πρόσβασης από ένα αρχείο λεξικού. Ένα αρχείο λεξικού μπορεί να συνταχθεί, για να καλύψει τις λέξεις, που χρησιμοποιούνται πιθανώς από τον ιδιοκτήτη του λογαριασμού, στον οποίο ένας κακόβουλος χρήστης πρόκειται να επιτεθεί. Ο επιτιθέμενος ενδέχεται να συγκεντρώσει πληροφορίες (μέσω ενεργητικής ή παθητικής αναγνώρισης, ανταγωνιστικής νοημοσύνης, κοινωνικής εφαρμοσμένης μηχανικής), ώστε να καταλάβει το χρήστη ή να δημιουργήσει ένα κατάλογο όλων των μοναδικών λέξεων, που είναι διαθέσιμες στον ιστοχώρο.

- Επιθέσεις αναζήτησης (Search attacks)

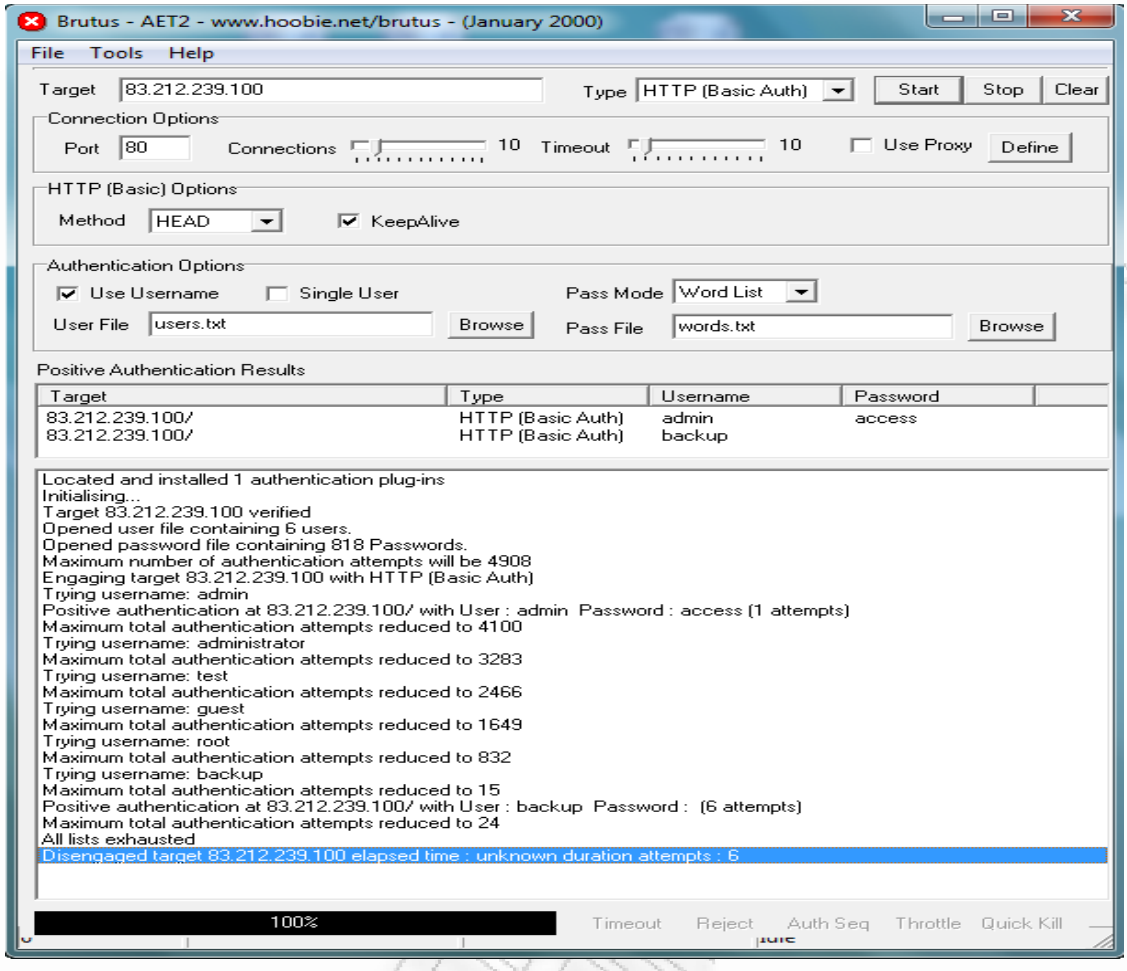
Οι επιθέσεις αναζήτησης θα προσπαθήσουν να καλύψουν όλους τους πιθανούς συνδυασμούς ενός δεδομένου συνόλου χαρακτήρων και ενός δεδομένου μήκους του κωδικού πρόσβασης. Αυτό το είδος επίθεσης είναι πολύ αργό, γιατί το διάστημα των πιθανών υποψηφίων είναι αρκετά μεγάλο. Αν λάβουμε υπόψιν για παράδειγμα μια γνωστή ταυτότητα χρηστών, ο συνολικός αριθμός κωδικών πρόσβασης μέχρι 8 χαρακτήρες στο μήκος, οι οποίοι χρειάζεται να δοκιμαστούν, είναι ίσος με 26(8!), δηλαδή περισσότεροι από 200 δισεκατομμύρια πιθανοί κωδικοί πρόσβασης.

- Επιθέσεις βασισμένες στους κανόνες αναζήτησης

Για να αυξήσουμε την κάλυψη του διαστήματος των συνδυασμών χωρίς μεγάλη επιβράδυνση της διαδικασίας, είναι προτιμότερο να δημιουργηθούν κάποιοι καλοί κανόνες, ώστε να παραχθούν οι υποψήφιοι κωδικοί. Το εργαλείο "John the Ripper" για παράδειγμα μπορεί να παράγει παραλλαγές κωδικού πρόσβασης από μέρος του ονόματος χρήστη ή να το τροποποιήσει μέσω μιας προ-διαμορφωμένης μάσκας λέξεων στην εισαγωγή (π.χ. 1ο βήμα: "pen" --> 2ο βήμα: "p3n" --> 3ο βήμα: "p3np3n").

BLACK BOX TESTING

Είναι πολύ σημαντικό το γεγονός να ανακαλύψουμε την μέθοδο αυθεντικοποίησης, που χρησιμοποιείται από την εφαρμογή που εξετάζουμε. Παραπάνω είδαμε τρόπους να ανακαλύψουμε τον τρόπο αυθεντικοποίησης που όμως αφορούσαν περιβάλλον Unix. Εμείς χρησιμοποιήσαμε αυτοματοποιημένο εργαλείο στη δική μας περίπτωση. Το WebScarab δεν προσφέρει τη δυνατότητα για κάτι τέτοιο έτσι χρησιμοποιήσαμε το εργαλείο Brutus ώστε να μας βοηθήσει στο να πραγματοποιήσουμε μια επίθεση brute force στην σελίδα διαχείρισης της ιστοσελίδας του Τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς. Στην παρακάτω εικόνα φαίνεται το εργαλείο με τα αποτελέσματα που αυτό εμφανίζει για το συγκεκριμένο έλεγχο.



Εικόνα : **Χρήση του εργαλείου Brutus για την ανακάλυψη της χρησιμοποιούμενης μεθόδου και την εφαρμογή μιας brute force επίθεσης στην σελίδα διαχείρισης της ιστοσελίδας του Τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς.**

Όπως βλέπουμε η εφαρμογή που εξετάζουμε χρησιμοποιεί HTTP αυθεντικοποίηση. Εφαρμόζοντας μια brute force επίθεση, βασισμένη σε λεξικό, με το εργαλείο αυτό δεν καταφέρνουμε να έχουμε πρόσβαση στην εφαρμογή παρά μόνο να ανακαλύψουνε έναν παραπάνω χρήστη αυτής. Συγκεκριμένα δοκιμάσαμε στη σελίδα που ελέγχουμε να βάλουμε τα credentials που υποτίθεται πως θα μας έδιναν πλήρη πρόσβαση στην εφαρμογή και κάτι τέτοιο δεν ισχύει παρά μόνο επιβεβαιώνεται η ορθή ανακάλυψη δυο ονομάτων χρηστών αυτής του "admin" και "backup".

3.4.5 Testing for by-pass authentication schema

Παρόλο που οι περισσότερες εφαρμογές απαιτούν αυθεντικοποίηση για να αποκτήσουν πρόσβαση στις ιδιωτικές πληροφορίες ή για να εκτελέσουν διάφορες εργασίες, δεν είναι όλες οι μέθοδοι αυθεντικοποίησης επαρκείς όσον αφορά την ασφάλεια, την οποία οφείλουν να παρέχουν. Η αμέλεια, η άγνοια ή απλά η μειωμένη σημασία στις απειλές ασφάλειας οδηγούν συχνά σε σχήματα

αυθεντικοποίησης, που μπορούν να παρακαμφθούν υπερπηδώντας απλά τη σελίδα σύνδεσης (bypassing authentication page) και καλώντας έτσι άμεσα μια εσωτερική σελίδα, που υποτίθεται ότι μπορεί να προσεγγιστεί μόνο, αφού έχει εκτελεσθεί η αυθεντικοποίηση. Πέραν τούτου είναι συχνά δυνατό να παρακαμφθούν τα μέτρα αυθεντικοποίησης με ανάλογη αλλαγή των αιτημάτων και εξαπατώντας με αυτόν τον τρόπο την εφαρμογή, η οποία υποθέτει ότι έχει γίνει ήδη η αυθεντικοποίηση. Αυτό μπορεί να πραγματοποιηθεί, είτε με την τροποποίηση της δεδομένης παραμέτρου URL, είτε με κατάλληλο χειρισμό της φόρμας, είτε με την πλαστογράφηση των συνόδων.

Τα προβλήματα σχετικά με το σχήμα αυθεντικοποίησης, θα μπορούσαν να βρεθούν στα διαφορετικά στάδια του κύκλου ζωής ανάπτυξης λογισμικού (Software Development Life Cycle - SDLC) όπως στη φάση σχεδίασης, ανάπτυξης και επέκτασης. Τα παραδείγματα των λαθών σχεδίασης περιλαμβάνουν ένα λανθασμένο καθορισμό των μερών της εφαρμογής που πρέπει να προστατεύονται, δηλαδή την επιλογή της εφαρμογής μη ισχυρών πρωτοκόλλων κρυπτογράφησης για την εξασφάλιση της ανταλλαγής στοιχείων αυθεντικοποίησης και πολλά άλλα. Τα προβλήματα στη φάση ανάπτυξης είναι για παράδειγμα η ανακριβής εφαρμογή των λειτουργιών επικύρωσης εισαγωγής ή η μη χρησιμοποίηση των καλύτερων πρακτικών ασφάλειας για τη συγκεκριμένη γλώσσα. Επιπλέον υπάρχουν ζητήματα κατά τη διάρκεια της εγκατάστασης της εφαρμογής όπως οι δραστηριότητες εγκαταστάσεων και διαμόρφωσης, λόγω έλλειψης απαιτούμενων τεχνικών δεξιοτήτων ή λόγω της ανεπαρκούς τεκμηρίωσης.

BLACK BOX TESTING

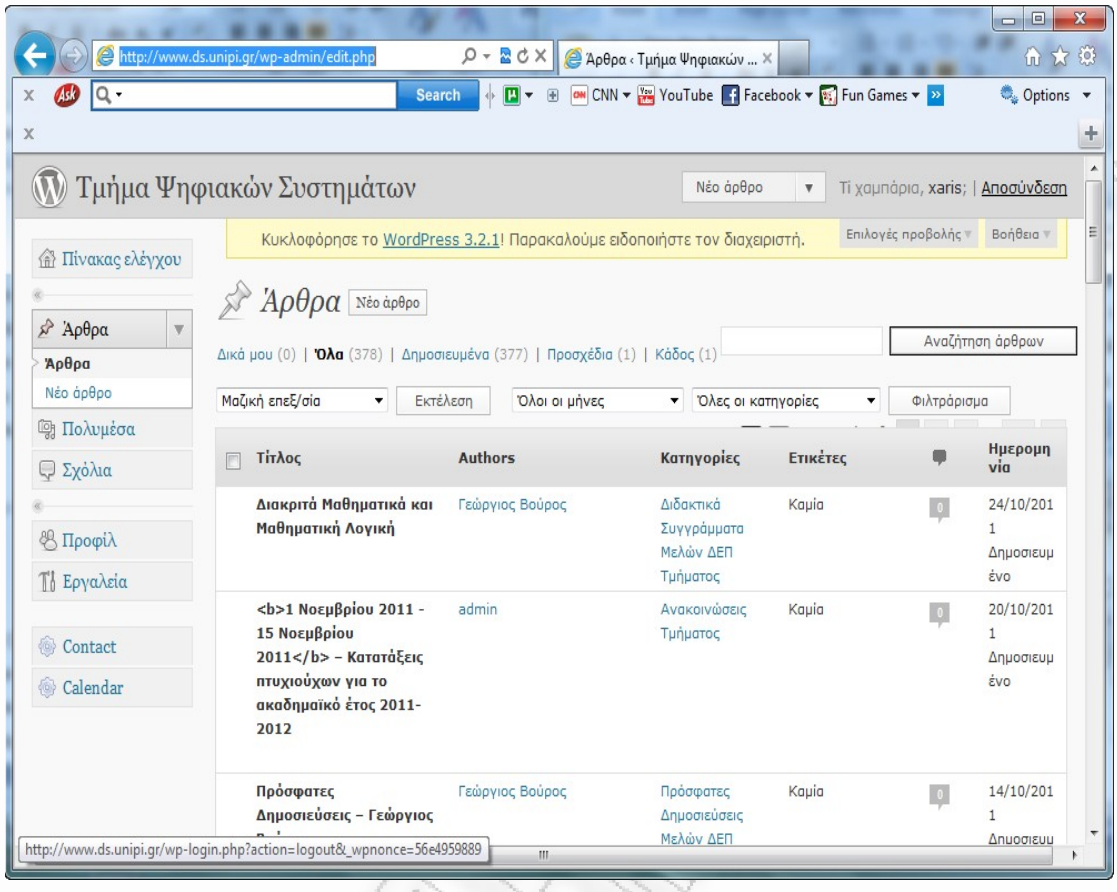
Υπάρχουν σε χρήση διάφορες μέθοδοι παράκαμψης του σχήματος αυθεντικοποίησης σε μια εφαρμογή ιστού:

a) Άμεσο αίτημα σελίδων (Direct page request)
b) Τροποποίηση παραμέτρου (Parameter Modification)
c) Πρόβλεψη ταυτότητας συνόδου (Session ID Prediction)
d) Έγχυση εντολών SQL (Sql Injection)

6 Άμεσο Αίτημα Σελίδων

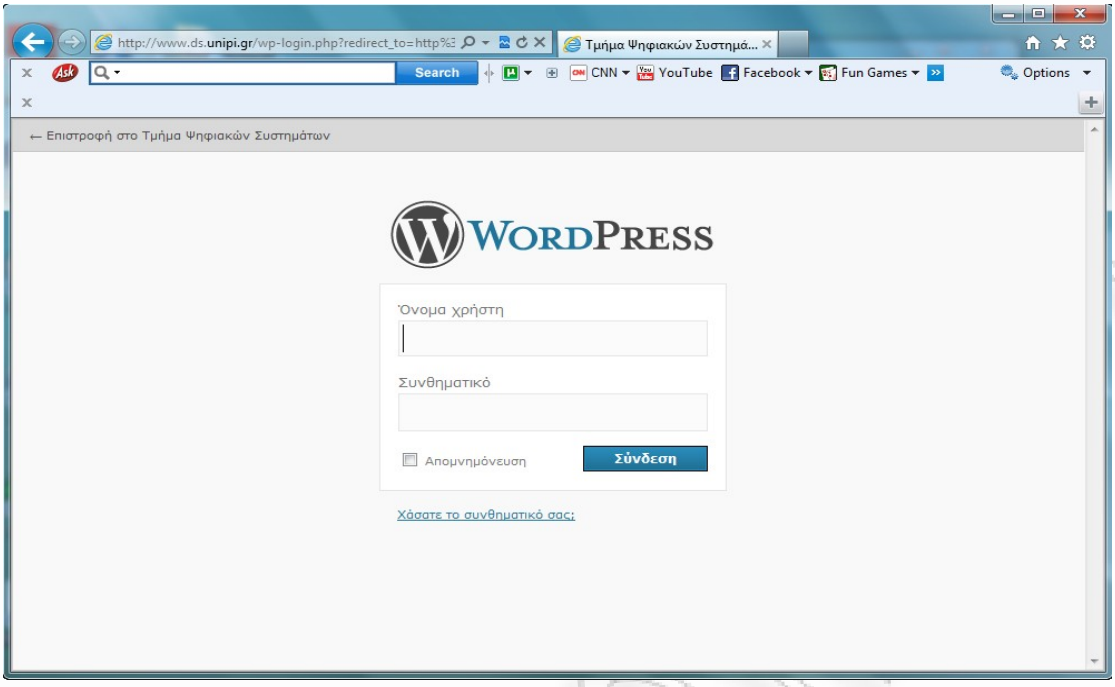
Εάν μια εφαρμογή ιστού εφαρμόζει τον έλεγχο πρόσβασης μόνο στη σελίδα σύνδεσης (login page), το σχήμα αυθεντικοποίησης θα μπορούσε να παρακαμφθεί. Εάν, για παράδειγμα, ένας χρήστης ζητά άμεσα μια επόμενη σελίδα από τη σελίδα αυθεντικοποίησης μέσω του browser, εκείνη η σελίδα μπορεί να μην ελέγξει τα πιστοποιητικά του χρήστη, πριν χορηγήσει την πρόσβαση. Πρέπει να προσεγγίσουμε άμεσα μια προστατευμένη σελίδα μέσω της μπάρας διευθύνσεων στον browser, για να εξετάσουμε τη χρησιμοποίηση αυτής της μεθόδου. Έτσι κάνουμε κανονικά login με τον κωδικό μας στο server που φιλοξενεί την ιστοσελίδα που εξετάζουμε και κάνουμε κλικ να δούμε τα «άρθρα» που ανεβαίνουν στο site του Τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου από τους καθηγητές του τμήματος. Αντιγράφουμε, λοιπόν, το URL που έχει η συγκεκριμένη σελίδα, http://www.ds.unipi.gr/wp-

admin/edit.php και κατόπιν αποσυνδεόμαστε από τη σελίδα διαχείρισης του ιστότοπου που εξετάζουμε. Οι ενέργειες που περιγράψαμε φαίνονται στην εικόνα.



Εικόνα : **Αυθεντικοποιημένη περιήγηση στην διαχειριστική σελίδα του www.ds.unipi.gr.**

Κατόπιν προσπαθούμε να μεταφερθούμε και πάλι στη σελίδα με τα «άρθρα» που είδαμε προηγουμένως πληκτρολογώντας το URL της σελίδας που είχαμε αντιγράψει ενώ ήμασταν αυθεντικοποιημένοι νόμιμα στην πλατφόρμα διαχείρισης του ιστότοπου. Το αποτέλεσμα φαίνεται στην παρακάτω εικόνα η οποία δείχνει πως πολύ σωστά για την ασφάλεια της εφαρμογής μας δεν μπορεί να γίνει υπερπήδηση του σχήματος αυθεντικοποίησης μέσω άμεσου αιτήματος σελίδας.



Εικόνα **: Αποτυχημένη προσπάθεια υπερπήδησης της σελίδας αυθεντικοποίησης μέσω ανάκτησης μιας σελίδας με δεδομένη διεύθυνση URL.**

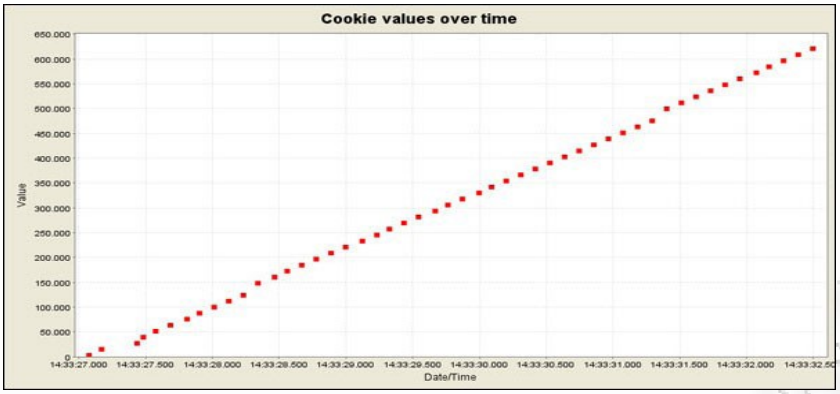
7 Τροποποίηση Παραμέτρου

Όπως αναφέραμε στο θεωρητικό μέρος της εργασίας, υφίσταται ένα πρόβλημα, σχετικά με το σχήμα αυθεντικοποίησης όταν η εφαρμογή ελέγχει μια επιτυχή σύνδεση βασιζόμενη στις σταθερές τιμές παραμέτρου. Ένας χρήστης μπορεί να τροποποιήσει αυτές τις παραμέτρους και να αποκτήσει πρόσβαση στις προστατευόμενες σελίδες, χωρίς να απαιτείται η παροχή των έγκυρων πιστοποιητικών. Κάτι τέτοιο όμως δεν συμβαίνει στην περίπτωση του ιστότοπου του Τμήματος Ψηφιακών Συστημάτων του πανεπιστημίου Πειραιώς, διότι δεν υπάρχουν σταθερές παράμετροι, που αλλάζοντάς τους να έχουμε τη δυνατότητα να αποκτήσουμε πρόσβαση στο server. Γι' αυτό η μέθοδος της τροποποίησης παραμέτρου δεν μπορεί να χρησιμοποιηθεί στην προσπάθεια δοκιμής διείσδυσης στην ιστοσελίδα που εξετάζουμε.

8 Πρόβλεψη Ταυτότητας Συνόδου

Πολλές εφαρμογές ιστού διαχειρίζονται την αυθεντικοποίηση χρησιμοποιώντας τις τιμές προσδιορισμού συνόδου (Session IDs). Επομένως εάν η παραγωγή του Session ID είναι προβλέψιμη, ένας κακόβουλος χρήστης θα είναι σε θέση να "ανακαλύψει" μια έγκυρη ταυτότητα συνόδου και να «κερδίσει» την αναρμόδια πρόσβασή του στην εφαρμογή υποδυόμενος έναν αυθεντικοποιημένο χρήστη.

Στην ακόλουθη εικόνα για παράδειγμα, παρατηρούμε ότι οι τιμές στα cookies σε σχέση με το χρόνο αυξάνονται γραμμικά, έτσι γι' αυτό θα μπορούσε να είναι εύκολο για έναν επιτιθέμενο να διαπιστώσει τον τρόπο δημιουργίας των cookies και να μπορέσει στη συνέχεια να υποθέσει μια έγκυρη ταυτότητα συνόδου χρήστη.



Εικόνα : **Γραφική παρουσίαση της γραμμικής αύξησης των Session IDs μέσω του WebScarab.**

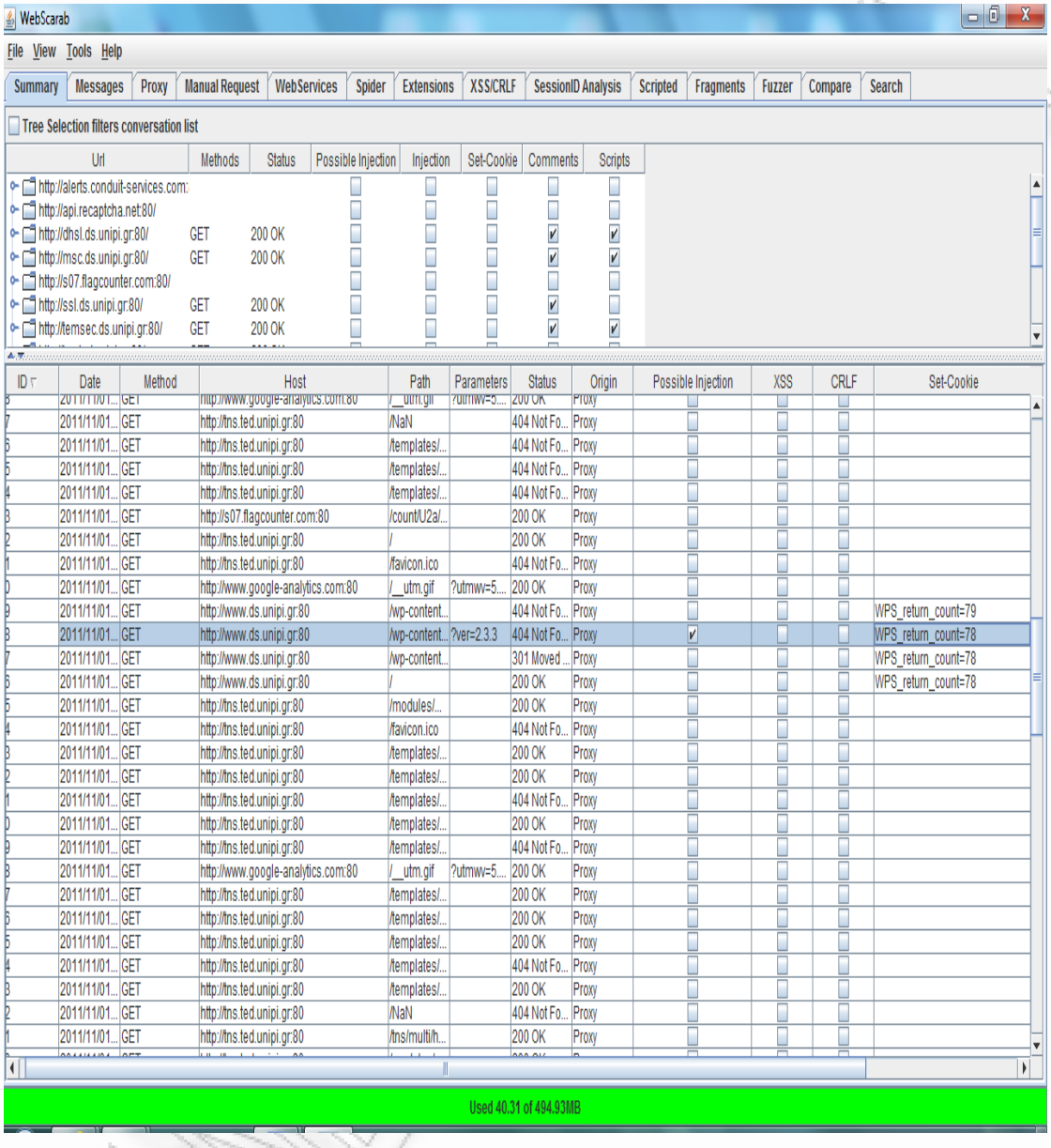
Επίσης στην ακόλουθη εικόνα παρατηρούμε ότι οι τιμές στα cookies αλλάζουν μόνο μερικώς, έτσι είναι δυνατόν κάποιος επιτιθέμενος που θα προσπαθήσει να υποθέσει/υπολογίσει μια έγκυρη τιμή cookie, μέσω μιας επίθεσης brute force, να περιοριστεί μόνο στους καθορισμένους τομείς τιμών που παρουσιάζονται παρακάτω.

Session Identifier :	127.0.0.1/WebGoat WEAKID
Date	Value
2006/11/11 14:33:27	1243011632520007028
2006/11/11 14:33:27	1243111632520007138
2006/11/11 14:33:27	1243211632520007247
2006/11/11 14:33:27	1243311632520007435
2006/11/11 14:33:27	1243411632520007544
2006/11/11 14:33:27	1243511632520007653
2006/11/11 14:33:27	1243611632520007763
2006/11/11 14:33:27	1243711632520007872
2006/11/11 14:33:28	1243811632520007982
2006/11/11 14:33:28	1243911632520008091
2006/11/11 14:33:28	1244011632520008200
2006/11/11 14:33:28	1244211632520008310
2006/11/11 14:33:28	1244311632520008419
2006/11/11 14:33:28	1244411632520008528
2006/11/11 14:33:28	1244511632520008638
2006/11/11 14:33:28	1244611632520008747
2006/11/11 14:33:28	1244711632520008857
2006/11/11 14:33:28	1244811632520008966
2006/11/11 14:33:29	1244911632520009075

Εικόνα : **Οι τιμές των cookies αλλάζουν μερικώς, κάτι που μπορεί να κάνει την πρόβλεψη τους εύκολη.**

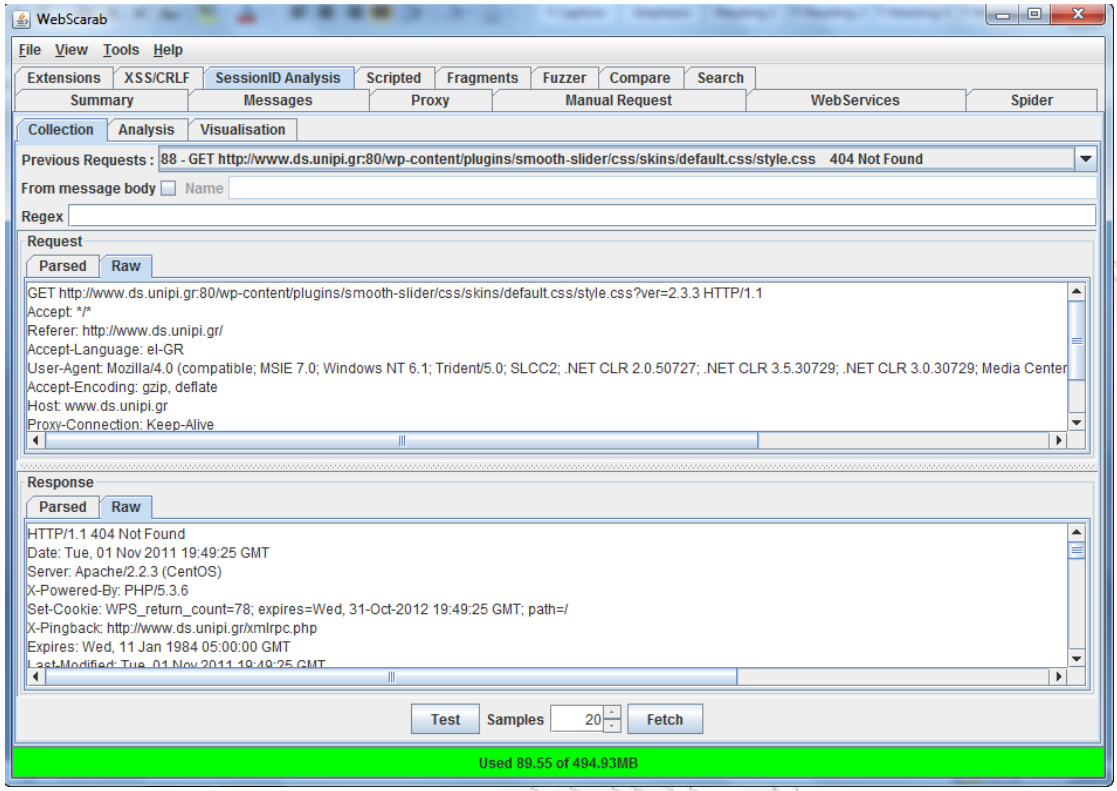
Ερχόμαστε τώρα να ελέγξουμε τον ιστότοπο του τμήματος Ψηφιακών Συστημάτων που εξετάζουμε συλλέγοντας Session IDs από τις ποικίλες συνομιλίες μας. Αν η παραγωγή του Session ID είναι προβλέψιμη, ένας κακόβουλος χρήστης μπορεί να "ανακαλύψει" μια έγκυρη ταυτότητα συνόδου και

να εξασφαλίσει αναρμόδια πρόσβαση στην εφαρμογή υποδυόμενος προηγουμένως έναν αυθεντικοποιημένο χρήστη. Συλλέγουμε λοιπόν τα Session IDs από τα cookies και πιο συγκεκριμένα από τις συνομιλίες, που έχουν συμπληρωμένη τη στήλη Set-Cookie, όπως φαίνεται στην παρακάτω εικόνα.



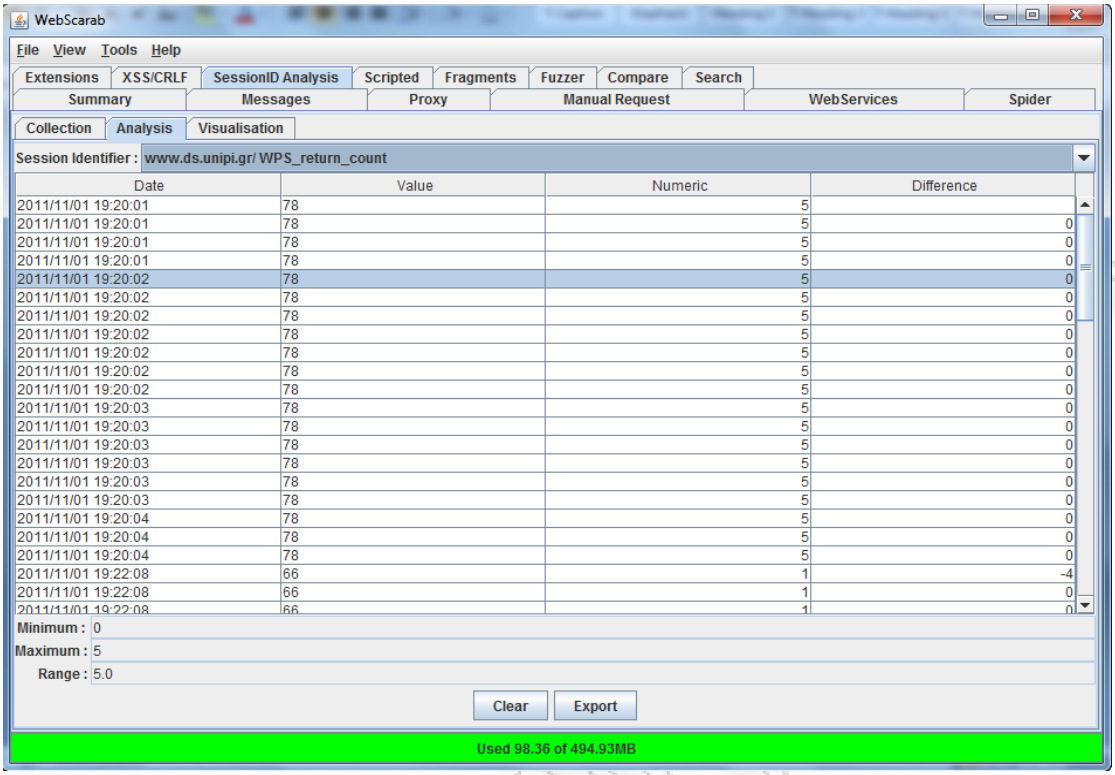
Εικόνα : **Συλλογή των cookies από τις συνομιλίες μέσω του WebScarab.**

Εντοπίζουμε λοιπόν μια συνομιλία, η οποία χρησιμοποιεί cookies, που στην περίπτωσή μας είναι μία μόνο συνομιλία κάθε φορά, και κατευθυνόμαστε στο plug-in SessionID Analysis, στην καρτέλα Collection (εικόνα 65). Εκεί στο πεδίο των αιτημάτων, βρίσκουμε τη συγκεκριμένη συνομιλία, όπως φαίνεται αναλυτικά στις παρακάτω εικόνες. Επιλέγουμε τον αριθμό των δειγμάτων, που θέλουμε, (π.χ. Samples:20) και επιλέγουμε “Test”. Το Session ID εξάγεται από τη συνομιλία και επιλέγοντας "Fetch" μπορούμε να το χρησιμοποιήσουμε στις επόμενες ετικέτες-εργαλεία του plug-in.



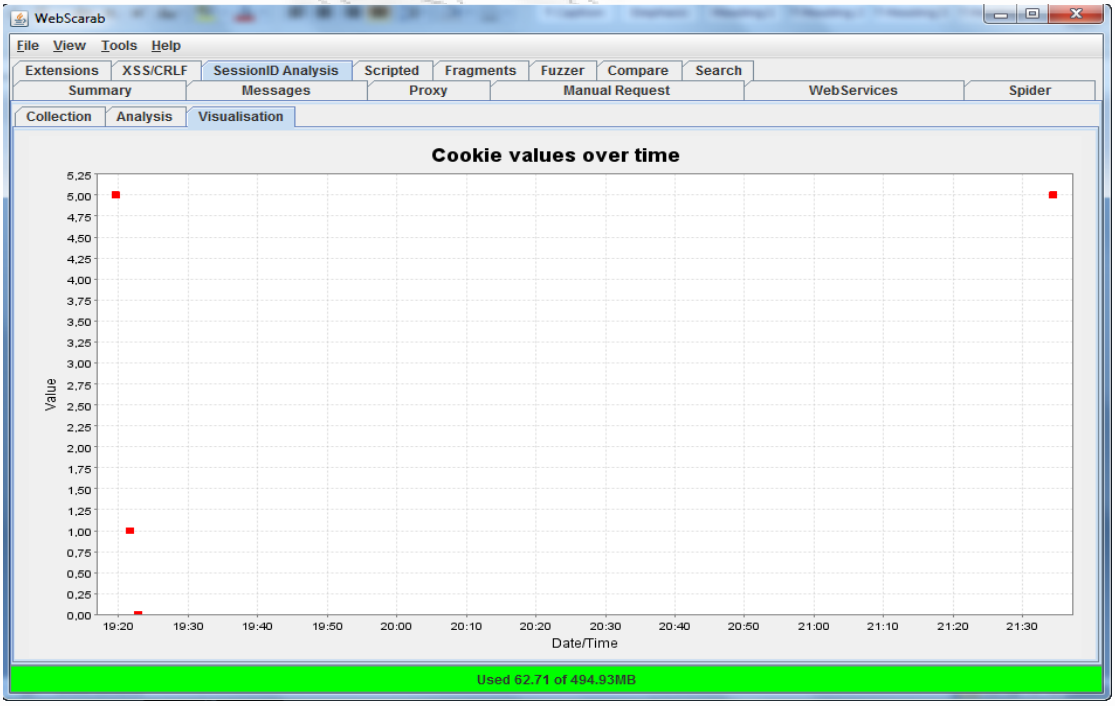
Εικόνα : **Συλλογή των Session IDs από συγκεκριμένες συνομιλίες.**

Στην ετικέτα Analysiss του ίδιου plug-in (εικόνα 66) μπορούμε να δούμε τις τιμές των cookies, που έχουν συλλεχθεί, και συγκεκριμένα την ημερομηνία συλλογής, την πραγματική τιμή τους, την αριθμητική τιμή τους και τη διαφορά μεταξύ τους, όπως φαίνεται στην εικόνα που ακολουθεί. Παρατηρούμε ότι δεν υπάρχει κάποια γραμμική σχέση μεταξύ των τιμών αυτών σε αντίθεση με το παράδειγμα, που αναφέραμε στο προηγουμένως, γεγονός που κάνει πιο δύσκολη την προσπάθεια ενός επιτιθέμενου να βρει ή να μαντέψει ένα έγκυρο Session ID ή να χρησιμοποιήσει μια επιτυχημένη brute force επίθεση.



Εικόνα : **Ανάλυση των Session Ids που συλλέχθηκαν.**

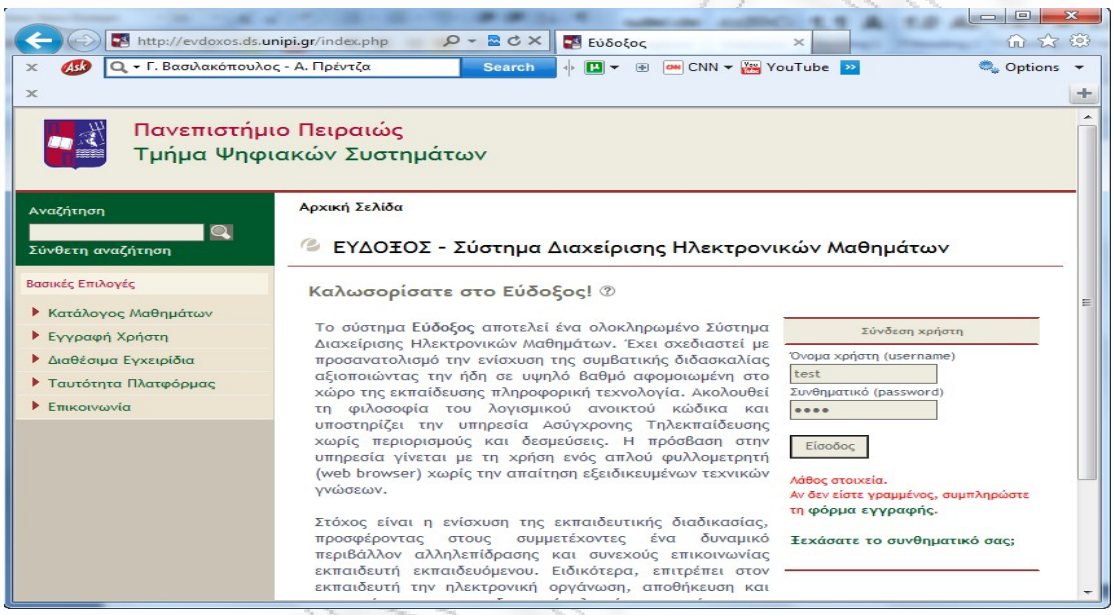
Η επόμενη ετικέτα του plug-in (Visualization) μας δίνει μια γραφική άποψη του επιλεγμένου Session ID (εικόνα 67), στην οποία μπορούμε να διαπιστώσουμε και οπτικά, αν όντως υπάρχει μια γραμμική ή επαναληπτική σχέση, το οποίο δεν ισχύει στην προκειμένη περίπτωση.



Εικόνα : **Γραφική παράσταση της τιμής του Session ID σε σχέση με το χρόνο.**

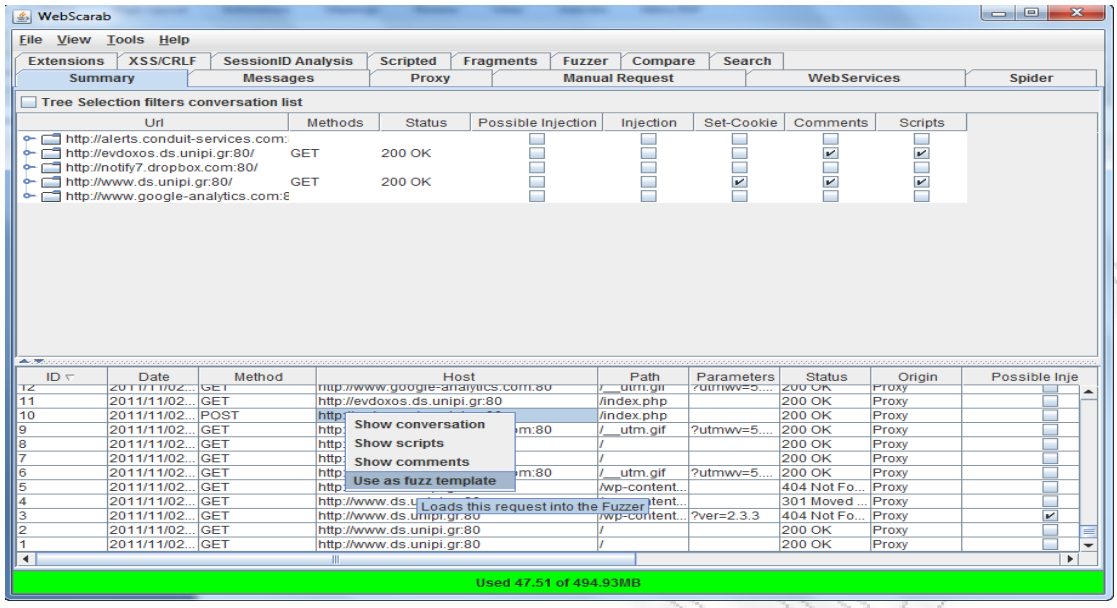
9 Έγχυση SQL σε HTML Φόρμα Αυθεντικοποίησης

Η έγχυση SQL (SQL Injection) είναι μια ευρέως γνωστή τεχνική επίθεσης. Απλά στον έλεγχο του μηχανισμού αυθεντικοποίησης μιας εφαρμογής ιστού, θα πρέπει οπωσδήποτε να ελέγξουμε, αν με την SQL Injection μπορούμε να παρακάμψουμε την HTML φόρμα αυθεντικοποίησης και να αποκτήσουμε πρόσβαση στην εφαρμογή. Θα προσπαθήσουμε να εφαρμόσουμε τον έλεγχο αυτό στον ιστότοπο του e-class του πανεπιστημιακού τμήματος που εξετάζουμε, οποίος λειτουργεί στον ίδιο server με τον κεντρικό ιστότοπο του τμήματος, και ακολουθεί αυθεντικοποίηση HTML φόρμας. Εισάγουμε δυο τυχαίες τιμές στα πεδία που μας δίνονται κατά τη σελίδα εισαγωγής στο σύστημα.



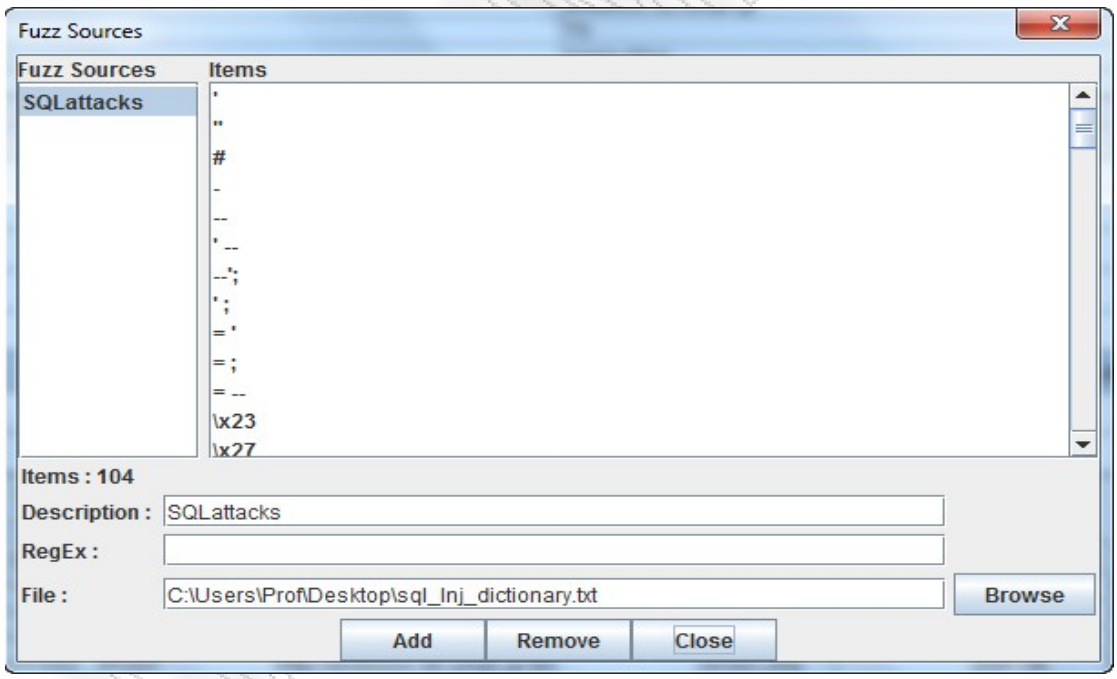
Εικόνα : **Προσπάθεια για σύνδεση στον ιστότοπο www.evdoxos.ds.unipi.gr.**

Αυτή η διαδικασία δεν θα μας εισάγει στο σύστημα όμως το εργαλείο WebScarab θα εμφανίσει τη διαδικασία σύνδεσης στην καρτέλα “summary”. Με το που συμβεί αυτό βρίσκουμε τη συνομιλία αυτήστην παραπάνω καρτέλα και την μεταφέρουμε με δεξί ‘κλικ’ στην καρτέλα fuzzer:



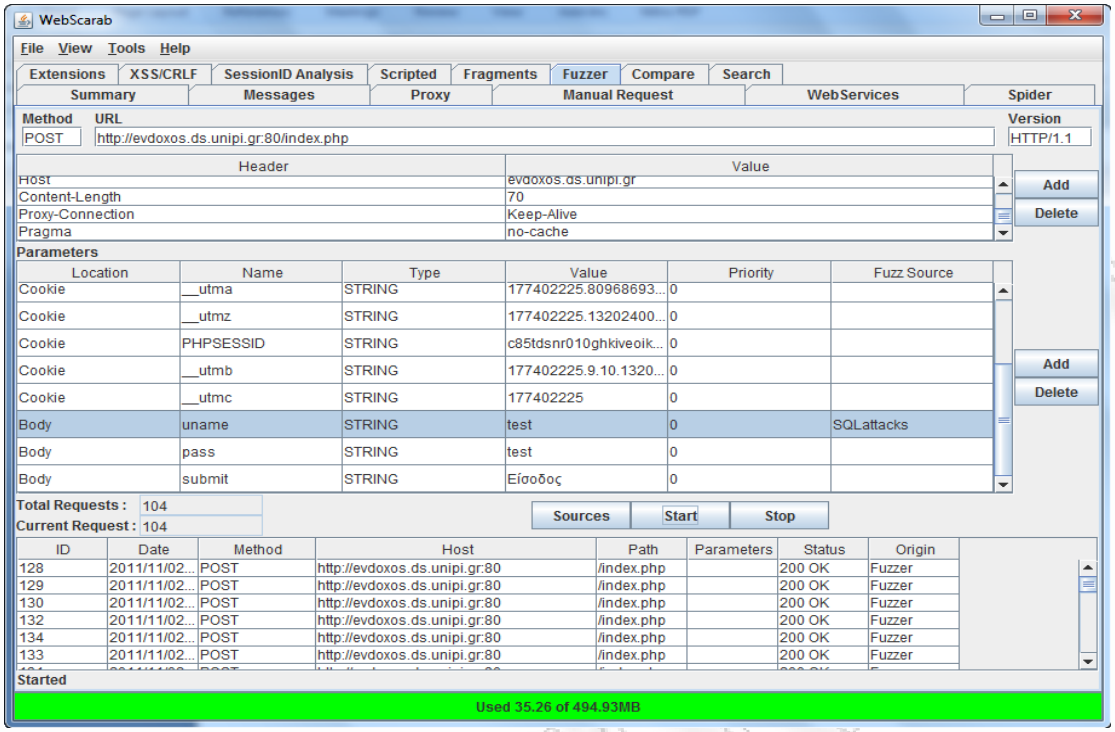
Εικόνα : **Εύρεση της συνομιλίας στην καρτέλα "Summary" και μεταφορά της στην καρτέλα "Fuzzer".**

Στην παρακάτω εικόνα βλέπουμε το λεξικό που χρησιμοποιήσαμε για την επίθεση SQL Injection.



Εικόνα : **Το λεξικό που χρησιμοποιήθηκε για το SQL Injection.**

Βλέποντας λοιπόν τα αποτελέσματα που εμφανίζει το εργαλείο μας εδώ συμπεραίνουμε πως η εφαρμογή που εξετάζουμε δεν είναι ευπαθής σε SQL Injection αναφορικά με διαδικασία αυθεντικοποίησης που εφαρμόζει. Φαίνεται στην εικόνα που ακολουθεί στο πεδίο "status" πως όλες οι συνομιλίες που εξετάστηκαν μέσω του λεξικού εμφάνισαν μη-ευπαθές αποτέλεσμα.



Εικόνα : **Η εκτέλεση του fuzzer του εργαλείου WebScarab.**

3.4.6 Testing for vulnerable remember password and password reset

Οι περισσότερες εφαρμογές ιστού επιτρέπουν στους χρήστες να επαναπροσδιορίσουν τον κωδικό πρόσβασής τους, εάν τον έχουν ξεχάσει. Αυτό συνήθως γίνεται με την αποστολή ενός μηνύματος επαναπροσδιορισμού κωδικού πρόσβασης μέσω ηλεκτρονικού ταχυδρομείου ή και με το να τους ζητείται να απαντήσουν μια ή περισσότερες "ερωτήσεις ασφάλειας". Εδώ ελέγχουμε ότι αυτή η λειτουργία εφαρμόζεται κατάλληλα και ότι δεν εισάγει οποιαδήποτε ρωγμή στο σχήμα αυθεντικοποίησης. Επίσης ελέγχουμε, εάν η εφαρμογή επιτρέπει στο χρήστη να αποθηκεύσει τον κωδικό πρόσβασης του στον browser δηλαδή εκτελούμε τη λειτουργία "απομνημόνευση κωδικού πρόσβασης".

Μια μεγάλη πλειοψηφία εφαρμογών παρέχει έναν τρόπο ανάκτησης του κωδικού πρόσβασης από τους χρήστες τους σε περίπτωση, που τον έχουν ξεχάσει. Η ακριβής διαδικασία ποικίλλει αρκετά μεταξύ των διαφόρων εφαρμογών, ανάλογα με το απαραίτητο επίπεδο ασφάλειας. Η προσέγγιση μπορεί σε πολλές περιπτώσεις να χρησιμοποιηθεί ως ένας εναλλακτικός τρόπος επικύρωσης της ταυτότητας του χρήστη. Μια από τις απλούστερες και πιο συνηθισμένες προσεγγίσεις είναι να ρωτηθεί ο χρήστης για τη διεύθυνση ηλεκτρονικού ταχυδρομείου του και να σταλεί ο παλαιός κωδικός πρόσβασης ή ένας νέος σε εκείνη την διεύθυνση. Αυτό το σχήμα είναι βασισμένο στην υπόθεση ότι το ηλεκτρονικό ταχυδρομείο του χρήστη δεν έχει εκτεθεί και ότι είναι αρκετά ασφαλές για αυτό τον σκοπό. Εναλλακτικά η εφαρμογή θα μπορούσε να ζητήσει από το χρήστη να απαντήσει σε μια ή περισσότερες "μυστικές ερωτήσεις", οι οποίες επιλέγονται συνήθως από το χρήστη μεταξύ ενός συνόλου ερωτήσεων. Η ασφάλεια αυτού του σχήματος βρίσκεται στη δυνατότητα να υπάρχει ένας τρόπος να προσδιορίζεται κάποιος χρήστης από το σύστημα σύμφωνα με τις απαντήσεις του σε ερωτήσεις, που δεν είναι εύκολο να απαντηθούν από τυχόν

υποκλοπή προσωπικών πληροφοριών σχετικές με το χρήστη. Για παράδειγμα μια πολύ επισφαλής ερώτηση θα ήταν "το όνομα της μητέρας σας" δεδομένου ότι αυτό είναι μια πληροφορία, που ένας επιτιθέμενος θα μπορούσε να ανακαλύψει χωρίς μεγάλη προσπάθεια. Ένα παράδειγμα καλύτερης ερώτησης θα ήταν "ο αγαπημένος σας σχολικός δάσκαλος", δεδομένου ότι αυτό θα ήταν ένα δυσκολότερο θέμα έρευνας για ένα πρόσωπο, του οποίου η ταυτότητα έχει κλαπεί.

Ένα άλλο κοινό χαρακτηριστικό γνώρισμα το οποίο οι εφαρμογές χρησιμοποιούν για να παρέχουν στους χρήστες μια ευκολία, είναι να αποθηκευθεί ο κωδικός πρόσβασης τοπικά στον browser και να δακτυλογραφείται αυτόματα σε όλες τις επόμενες προσπάθειες πρόσβασης τους. Ενώ αυτό το χαρακτηριστικό γνώρισμα, είναι εξαιρετικά φιλικό για το μέσο χρήστη, εισάγοντας συγχρόνως μια ρωγμή ασφάλειας, καθώς ο λογαριασμός χρήστη μπορεί να γίνει εύκολα προσιτός από οποιονδήποτε άλλο χρήστη χρησιμοποιεί το ίδιο υπολογιστικό μηχάνημα.

BLACK BOX TESTING

Password Reset

Το πρώτο βήμα είναι να ελέγξουμε κατά πόσο χρησιμοποιούνται οι μυστικές ερωτήσεις. Η αποστολή του κωδικού πρόσβασης (ή του link επαναφοράς του κωδικού πρόσβασης) στην διεύθυνση του ηλεκτρονικού ταχυδρομείου του χρήστη χωρίς πρώτα να γίνει μια μυστική ερώτηση συνεπάγεται πως στηριζόμαστε 100% στην ασφάλεια αυτής της διεύθυνσης του ηλεκτρονικού ταχυδρομείου, το οποίο δεν είναι το κατάλληλο εάν η εφαρμογή χρειάζεται ένα υψηλό επίπεδο ασφάλειας. Από την άλλη, εάν χρησιμοποιείται μια μυστική ερώτηση, το επόμενο βήμα είναι να αξιολογήσουμε την ισχύ της. Σαν πρώτο σημείο θα πρέπει να αξιολογήσουμε πόσες ερωτήσεις πρέπει να απαντηθούν πριν γίνει η επαναφορά του κωδικού πρόσβασης. Στην πλειοψηφία των εφαρμογών χρειάζεται ο χρήστης χρειάζεται να απαντήσει σε μία ερώτηση όμως σε κάποιες κρίσιμες εφαρμογές απαιτείται ο χρήστης να απαντήσει σωστά σε δύο ή περισσότερες ερωτήσεις. Σαν δεύτερο βήμα, πρέπει να αναλύσουμε τις ίδιες τις ερωτήσεις. Συχνά ένα σύστημα που αυτο-επαναφέρεται προσφέρει την επιλογή για πολλαπλές ερωτήσεις. Αυτό είναι καλό σημάδι για τον επίδοξο επιτιθέμενο καθώς τον παρουσιάζει με επιλογές. Θα πρέπει να αναρωτηθούμε αν θα μπορούσαμε να λάβουμε απαντήσεις σε κάποια ή σε όλες τις ερωτήσεις μέσω μιας απλής αναζήτησης στο google ή μέσω μιας επίθεσης με χρήση social engineering. Παρακάτω παρουσιάζεται μια βήμα-προς-βήμα διαδικασία αξιολόγησης των εργαλείων επαναφοράς του κωδικού πρόσβασης.

- Προσφέρονται πολλαπλές ερωτήσεις;
    - Εάν ναι, προσπαθήστε να διαλέξετε μια ερώτηση που έχουν μια πιο «κοινή» απάντηση, για παράδειγμα κάτι που με μια απλή αναζήτηση στο Google θα το βρούμε εύκολα
    - Πάντα διαλέγουμε ερωτήσεις που έχουν πραγματική απάντηση όπως «το πρώτο μου σχολείο» ή άλλα γεγονότα στα οποία μπορούμε να ανατρέξουμε
    - Αναζητούμε ερωτήσεις που έχουν λίγες πιθανές απαντήσεις όπως «πιο ήταν το πρώτο αυτοκίνητό σας». Αυτή η ερώτηση παρουσιάζεται στον επιτιθέμενο έχοντας λίγες

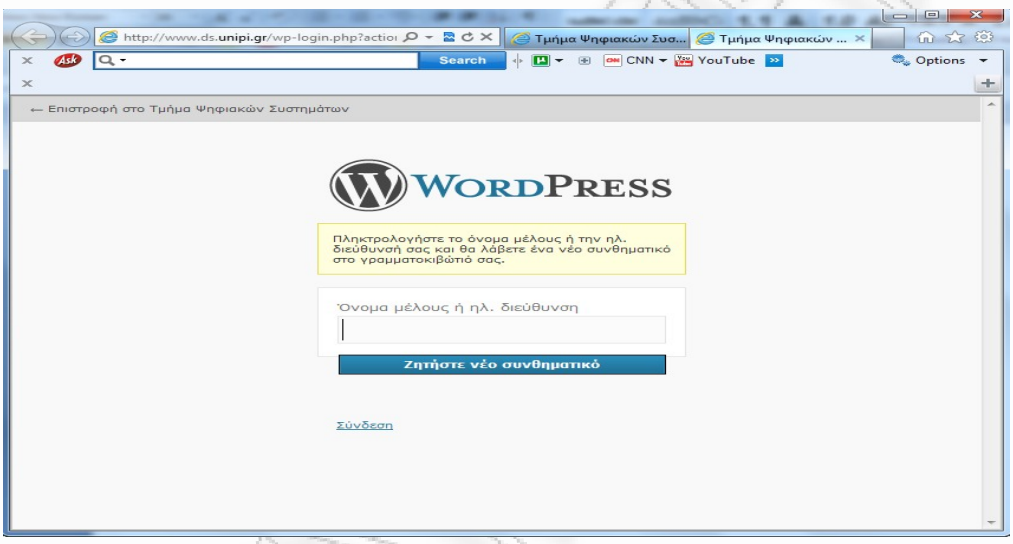
απαντήσεις να μαντέψει και στηρίζεται σε στατιστικά στοιχεία που θα πρέπει να βαθμολογήσει από το περισσότερο στο λιγότερο πιθανό.

- Καθορίστε πόσες φορές για εικασίες έχετε (αν είναι δυνατόν)
    - Η διαδικασία επαναφοράς του κωδικού πρόσβασης έχει απεριόριστες προσπάθειες;
    - Υπάρχει κάποια περίοδος κλειδώματος για Χ λάθος απαντήσεις; Πρέπει να έχουμε στο μυαλό μας ότι ένα σύστημα κλειδώματος μπορεί να είναι ένα πρόβλημα ασφάλειας από μόνο του καθώς μπορεί να το εκμεταλλευτεί ένας επιτιθέμενος ώστε να προκαλέσει άρνηση παροχής της υπηρεσίας στους χρήστες.
- Επιλέξτε την κατάλληλη ερώτηση βασιζόμενοι στην ανάλυση από τα παραπάνω στοιχεία και πραγματοποιείστε έρευνα για να προσδιορίσετε την πιο πιθανή απάντηση
- Πως συμπεριφέρεται το εργαλείο επαναφοράς του κωδικού πρόσβασης;
    - Επιτρέπει την άμεση αλλαγή του κωδικού πρόσβασης;
    - Εμφανίζει τον παλιό κωδικό πρόσβασης;
    - Αποστέλλει τον κωδικό πρόσβασης μέσω email στην προκαθορισμένη διεύθυνση ηλεκτρονικού ταχυδρομείου;
    - Το πιο μη ασφαλές σενάριο είναι εάν το εργαλείο επαναφοράς του κωδικού πρόσβασης μας εμφανίζει τον κωδικό αυτόν. Αυτό δίνει τη δυνατότητα στον επιτιθέμενο να μπει στον λογαριασμό και μόνο εάν η εφαρμογή παρέχει πληροφορίες για την τελευταία είσοδο (login) το θύμα θα καταλάβει πως ο λογαριασμός του έχει παραβιαστεί.
    - Ένα λιγότερο μη ασφαλές σενάριο είναι αν το εργαλείο επαναφοράς του κωδικού πρόσβασης επιβάλλει στον χρήστη να αλλάξει αμέσως τον κωδικό του. Όσο αυτός είναι αποδεκτός σαν κωδικός όπως στην πρώτη περίπτωση, επιτρέπει στον επιτιθέμενο να αποκτήσει πρόσβαση στο λογαριασμό και να κλειδώσει τον πραγματικό χρήστη «απ' έξω».
    - Η καλύτερη ασφάλεια επιτυγχάνεται όταν η επαναφορά του κωδικού πρόσβασης γίνεται μέσω email στην διεύθυνση του ηλεκτρονικού ταχυδρομείου που ο χρήστης αρχικά είχε εγγραφεί, ή σε κάποια άλλη διεύθυνση. Αυτό αναγκάζει τον επιτιθέμενο όχι μόνο να μαντέψει σε ποια διεύθυνση του ηλεκτρονικού ταχυδρομείου έχει σταλεί ο κωδικός πρόσβασης μετά την επαναφορά του (εκτός και αν το λέει η εφαρμογή) αλλά επίσης θέσει σε κίνδυνο τον λογαριασμό προκειμένου να πάρει τον έλεγχο της πρόσβασης του θύματος στην εφαρμογή.

Το κλειδί για την επιτυχή εκμετάλλευση και παράκαμψη του αυτόματης επαναφοράς του κωδικού πρόσβασης είναι να βρούμε μια ερώτηση ή ένα σύνολο ερωτήσεων οι οποίες δίνουν τη δυνατότητα για εύκολη ανάκτηση των απαντήσεων. Πάντα πρέπει να ψάχνουμε για ερωτήσεις που μας δίνουν στατιστικά τη μεγαλύτερη δυνατότητα να μαντέψουμε τη σωστή απάντηση εάν είμαστε απολύτως βέβαιοι για κάποιες από τις απαντήσεις. Στο τέλος, ένα εργαλείο αυτο-επαναφοράς του κωδικού

πρόσβασης είναι τόσο δυνατό όσο η πιο αδύναμη ερώτηση. Σαν δευτερεύουσα σημείωση, αν η εφαρμογή στέλνει/απεικονίζει τον παλιό κωδικό πρόσβασης σε ένα απλό κείμενο σημαίνει ότι οι κωδικοί δεν αποθηκεύονται σε κατακερματισμένη (hashed) μορφή, το οποίο είναι ένα ζήτημα ασφάλειας από μόνο του.

Στην περίπτωση του ιστότοπου που εξετάζουμε δεν χρησιμοποιούνται καθόλου μυστικές ερωτήσεις κατά τη διαδικασία επανάκτησης του κωδικού πρόσβασης. Από την άλλη χρησιμοποιείται η δυνατότητα έκδοσης νέου κωδικού πρόσβασης με τη συμπλήρωση είτε του ονόματος χρήστη είτε της διεύθυνσης του ηλεκτρονικού ταχυδρομείου αυτού και την αποστολή ηλεκτρονικού μηνύματος με το νέο κωδικό πρόσβασης. Άρα η εφαρμογή μας σε αυτό το σημείο στηρίζεται στην ασφάλεια που προσφέρει το ηλεκτρονικό ταχυδρομείο του χρήστη. Παρακάτω στην εικόνα φαίνεται το αποτέλεσμα της επιλογής «χάσατε το συνθηματικό σας» που υπάρχει στη σελίδα αυθεντικοποίησης της εφαρμογής.



Εικόνα : **Διαδικασία επανάκτησης του κωδικού πρόσβασης.**

Απομνημόνευση Κωδικού Πρόσβασης

Ο μηχανισμός "απομνημόνευση κωδικού πρόσβασης" μπορεί να υλοποιηθεί με μια από τις ακόλουθες μεθόδους:

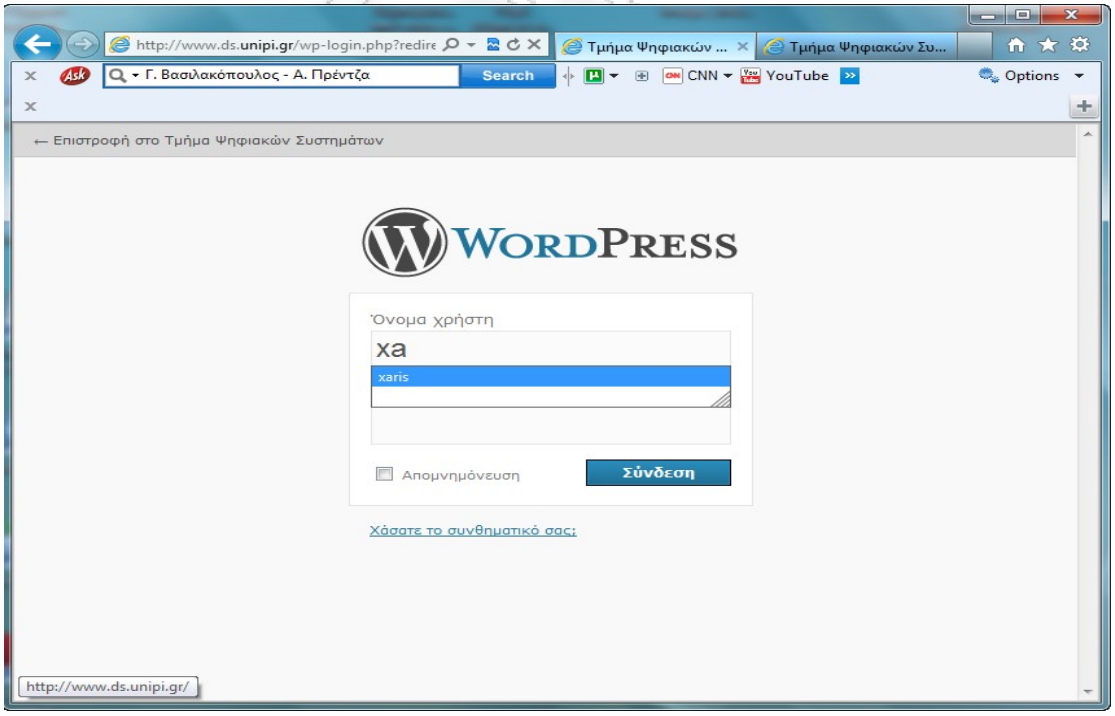
- Επιτρέποντας το χαρακτηριστικό "cache password" στους web browsers. Αν και δεν είναι άμεσα ένας μηχανισμός της εφαρμογής, μπορεί και πρέπει να τεθεί εκτός λειτουργίας.
- Αποθηκεύοντας τον κωδικό πρόσβασης σε ένα μόνιμο cookie. Ο κωδικός πρόσβασης πρέπει να κατακερματιστεί (hashed) και να κρυπτογραφηθεί, ώστε να μην στέλνεται με μορφή απλού κειμένου.

Για την πρώτη μέθοδο πρέπει να ελέγξουμε τον κώδικα HTML της σελίδας σύνδεσης, για να δούμε, εάν η αποθήκευση στον browser των κωδικών πρόσβασης είναι εκτός λειτουργίας. Ο κώδικας για αυτό είναι συνήθως:

<INPUT TYPE="password" AUTOCOMPLETE="off">

Η αυτόματη συμπλήρωση του κωδικού πρόσβασης πρέπει πάντα να είναι εκτός λειτουργίας και ειδικά σε ευαίσθητες εφαρμογές, αφού ένας επιτιθέμενος, που είναι ικανός να αποκτήσει πρόσβαση στην μνήμη (cache) του browser, θα μπορούσε εύκολα να λάβει τον κωδικό πρόσβασης σε μορφή απλού κειμένου οι υπολογιστές δημόσιας χρήσης είναι ένα πολύ χαρακτηριστικό παράδειγμα αυτής της επίθεσης. Για να ελέγξουμε την δεύτερη μέθοδο εφαρμογής, οφείλουμε να εξετάσουμε το cookie, που αποθηκεύεται από την εφαρμογή. Πρέπει να ελέγξουμε ότι τα πιστοποιητικά δεν αποθηκεύονται σε απλό κείμενο, αλλά κατακερματίζονται (hashing). Να εξετάσουμε το μηχανισμό hashing: εάν είναι ένας κοινός μηχανισμός, πρέπει να ελέγξουμε τη δύναμή του. Στις κοινές hash συναρτήσεις, πρέπει να προσπαθήσουμε να δοκιμάσουμε διάφορα ονόματα χρήστη για να ελέγξουμε εάν η hash συνάρτηση είναι εύκολα προβλέψιμη. Επιπλέον, πρέπει να ελέγξουμε ότι τα διαπιστευτήρια στέλνονται μόνο κατά τη διάρκεια της φάσης σύνδεσης και δεν στέλνονται μαζί με κάθε αίτημα του χρήστη στην εφαρμογή.

Στον ιστότοπο μας γίνεται χρήση της δυνατότητας απομνημόνευσης του κωδικού πρόσβασης και μάλιστα είναι εμφανές αυτό από την επιλογή «Απομνημόνευση» που προσφέρεται στο περιεχόμενο της σελίδας σύνδεσης της εφαρμογής. Αυτό αποτελεί μια σημαντική ευπάθεια για την αυθεντικοποίηση της εφαρμογής μας για όλους τους λόγους που είδαμε παραπάνω. Δε θα πρέπει σε καμία περίπτωση να αφήνουμε περιθώρια σε έναν κακόβουλο χρήστη να ανακτήσει τον κωδικό πρόσβασης της σελίδας διαχείρισης της εφαρμογής μας διότι τότε μπορούμε πολύ εύκολα να χάσουμε τον πλήρη έλεγχο της εφαρμογής μας και αν αλλάξει ο επιτιθέμενος τον κωδικό να μας «κλειδώσει απ' έξω».



Εικόνα : **Η διαχειριστική σελίδα με την επιλογή «απομνημόνευση» του κωδικού πρόσβασης καθώς και του ονόματος χρήστη.**

3.4.7 Testing for Logout and Browser Cache Management

Σε αυτό τον έλεγχο εξετάζουμε, αν η λειτουργία αποσύνδεσης (logout) εφαρμόζεται κατάλληλα και αν δεν είναι δυνατή η "επαναχρησιμοποίηση" μιας συνόδου χρήστη μετά την αποσύνδεση του. Επίσης ελέγχουμε ότι η εφαρμογή αποσυνδέει αυτόματα έναν χρήστη, όταν εκείνος δεν κάνει κάποια ενέργεια για ένα ορισμένο χρονικό διάστημα και ότι καμία ευαίσθητη πληροφορία δεν παραμένει αποθηκευμένη στην προσωρινή μνήμη (cache) του browser. Το τέλος μιας web συνόδου προκαλείται συνήθως με ένα από τα ακόλουθα δύο γεγονότα:

- Ο χρήστης αποσυνδέεται
- Ο χρήστης παραμένει ανενεργός για ένα ορισμένο χρονικό διάστημα και η εφαρμογή τον αποσυνδέει αυτόματα

Και οι δύο περιπτώσεις πρέπει να εφαρμοστούν προσεκτικά, προκειμένου να αποφύγουμε τις αδυναμίες, οι οποίες θα προκύψουν και οι οποίες θα μπορούσαν να χρησιμοποιηθούν από έναν επιτιθέμενο για να κερδίσει την αναρμόδια πρόσβαση του στην εφαρμογή. Πιο συγκεκριμένα η λειτουργία αποσύνδεσης πρέπει να εξασφαλίσει ότι όλα τα σημεία συνόδου (π.χ.cookies) καταστρέφονται κατάλληλα ή γίνονται ακατάλληλα προς χρήση και ότι οι κατάλληλοι έλεγχοι επιβάλλονται στην πλευρά των web servers, για να απαγορεύσουν την επαναχρησιμοποίηση τους. Το πιο σημαντικό για την εφαρμογή είναι να ακυρωθεί η σύνοδος χρήστη στην πλευρά των web servers. Αυτό σημαίνει ότι ο κώδικας πρέπει να επικαλεσθεί την κατάλληλη μέθοδο (π.χ. "HttpSession.invalidate()" στην Java ή "Session.abandon()" στη .NET). Ο καθαρισμός των cookies από τον browser είναι μια καλή ενέργεια αλλά δεν είναι αυστηρά απαραίτητη δεδομένου ότι, εάν η σύνοδος ακυρώνεται κατάλληλα στο server, η κατοχή του cookie στο browser δεν θα βοηθήσει έναν επιτιθέμενο.

Εάν αυτές οι ενέργειες δεν πραγματοποιούνται κατάλληλα, ένας επιτιθέμενος θα μπορούσε να επαναλάβει αυτά τα σημεία συνόδου, προκειμένου "να ανασυσταθεί" η σύνοδος ενός νόμιμου χρήστη και ουσιαστικά να υποδυθεί αυτόν. Αυτή η επίθεση είναι συνήθως γνωστή ως "επανάληψη cookie". Φυσικά ένας παράγοντας, που μετριάζει αυτό το φαινόμενο, είναι ότι ο επιτιθέμενος πρέπει να είναι σε θέση να έχει πρόσβαση σε εκείνα τα αποθηκευμένα στο PC του χρήστη σημεία, αλλά σε μερικές περιπτώσεις ίσως να μην είναι πάρα πολύ δύσκολο κάτι τέτοιο. Το πιο κοινό σενάριο για αυτό το είδος επίθεσης είναι ένας δημόσιος υπολογιστής, για παράδειγμα σε ένα internet cafe, ο οποίος χρησιμοποιείται για πρόσβαση σε κάποιες ιδιωτικές πληροφορίες π.χ. ηλεκτρονικό ταχυδρομείο, online τραπεζικός λογαριασμός, κλπ. Όταν ο χρήστης έχει τελειώσει με την εφαρμογή και αποσυνδέεται, εάν η διαδικασία αποσύνδεσης δεν επιβάλλεται κατάλληλα από τον web server, ο επόμενος χρήστης που θα χρησιμοποιήσει τον υπολογιστή θα μπορούσε ίσως να έχει πρόσβαση στον ίδιο λογαριασμό. Αυτό θα μπορούσε να συμβεί απλά πιέζοντας το κουμπί "back" του browser. Ένα άλλο σενάριο το οποίο ενδέχεται να προκύψει είναι από μια Cross Site Scripting ευπάθεια της εφαρμογής ή από μια σύνδεση που δεν είναι προστατευμένη από SSL: μια

ευάλωτη λειτουργία αποσύνδεσης θα καθιστούσε τα κλεμμένα cookies για πολύ μεγαλύτερο χρονικό διάστημα στη διάθεση κάποιου επιτιθέμενου, πράγμα που καθιστά το έργο του πολύ ευκολότερο. Ένα τελευταίο σενάριο είναι η εφαρμογή να μην απαγορεύει στον browser την αποθήκευση ευαίσθητων δεδομένων, τα οποία θα έθεταν πάλι σε κίνδυνο ένα χρήστη, ο οποίος έχει πρόσβαση στην εφαρμογή από ένα δημόσιο υπολογιστή.

BLACK BOX TESTING

Λειτουργία Logout

Το πρώτο βήμα ελέγχου είναι να εξεταστεί η ύπαρξη της συνάρτησης αποσύνδεσης. Πρέπει να γίνει έλεγχος ότι η εφαρμογή παρέχει μια επιλογή (κουμπί) αποσύνδεσης και ότι αυτό το κουμπί είναι παρόν και καλά ορατό σε όλες τις σελίδες που απαιτούν αυθεντικοποίηση. Ένα κουμπί αποσύνδεσης, που δεν είναι σαφώς ορατό ή που είναι παρόν μόνο σε ορισμένες σελίδες, θέτει ένα κίνδυνο ασφάλειας, αφού ο χρήστης μπορεί να ξεχάσει να το χρησιμοποιήσει στο τέλος της συνόδου του.

Το δεύτερο βήμα συνίσταται στον έλεγχο του τι συμβαίνει στα σημεία συνόδου όταν επικαλείται η συνάρτηση αποσύνδεσης. Παραδείγματος χάριν στην περίπτωση, που χρησιμοποιούνται cookies, η κατάλληλη συμπεριφορά της εφαρμογής θα ήταν να διαγραφούν όλα τα cookies συνόδου με την έκδοση μιας νέας εντολής Set-Cookie, η οποία θα θέτει την τιμή τους σε μη-έγκυρη (π.χ. “NULL” ή μια παρόμοια τιμή) και αν το cookie είναι μόνιμο, να θέτει την ημερομηνία λήξης του σε μια ημερομηνία στο παρελθόν, το οποίο δηλώνει στο browser ότι αυτό έχει λήξει και πρέπει να απορριφθεί. Έτσι για παράδειγμα, εάν η σελίδα αυθεντικοποίησης θέτει αρχικά ένα cookie με τον ακόλουθο τρόπο:

Set-Cookie: SessionID=sjdhqwoy938eh1q; expires=Sun, 29-Oct-2008 12:20:00 GMT; path=/; domain=victim.com

τότε η λειτουργία αποσύνδεσης πρέπει να προκαλέσει μια απάντηση, που μοιάζει με την ακόλουθη:

Set-Cookie: SessionID=noauth; expires=Sat, 01-Jan-2000 00:00:00 GMT; path=/; domain=victim.com

Η πρώτη λοιπόν δοκιμή συνίσταται στο να αποσυνδεθούμε και έπειτα να επιλέξουμε το κουμπί "back" του browser, για να ελέγξουμε εάν είμαστε ακόμη αυθεντικοποιημένοι. Εάν όντως είμαστε, σημαίνει ότι η συνάρτηση αποσύνδεσης δεν έχει εφαρμοστεί κατάλληλα, με αποτέλεσμα να μην καταστρέφει τα χαρακτηριστικά συνόδου (Session IDs). Αυτό μερικές φορές συμβαίνει με τις εφαρμογές, που χρησιμοποιούν όχι μόνιμα (non-persistent) cookies και που απαιτούν από το χρήστη να κλείσει το browser του, προκειμένου να σβηστούν αποτελεσματικά τα cookies από τη μνήμη του. Μερικές από αυτές τις εφαρμογές παρέχουν μια προειδοποίηση στο χρήστη, που του προτείνουν να κλείσει το browser του, αλλά αυτή η λύση στηρίζεται εντελώς στη συμπεριφορά των χρηστών και οδηγεί σε ένα χαμηλότερο επίπεδο ασφάλειας έναντι της καταστροφής των cookies. Άλλες εφαρμογές προσπαθούν να κλείσουν το

browser χρησιμοποιώντας κώδικα JavaScript, αλλά αυτή είναι πάλι μια λύση η οποία στηρίζεται στη συμπεριφορά των χρηστών, η οποία είναι πραγματικά λιγότερο ασφαλής, αφού ο browser του χρήστη μπορεί να έχει διαμορφωθεί έτσι, ώστε να περιορίζει την εκτέλεση των scripts. Σε αυτήν την περίπτωση μια διαμόρφωση που είχε ως στόχο την αύξηση της ασφάλειας, θα κατέληγε στη μείωση αυτής. Επιπλέον η αποτελεσματικότητα αυτής της λύσης θα εξαρτιόταν από τον εκάστοτε προμηθευτή, την έκδοση και τις ρυθμίσεις του browser. Για παράδειγμα ο κώδικας JavaScript ίσως επιτυχημένα να έκλεινε μια σελίδα του Internet Explorer, αλλά να αποτύγχανε σε μια σελίδα του Firefox.

Το δεύτερο βήμα ελέγχου αφορά στον έλεγχο αποθήκευσης δεδομένων στη μνήμη του browser. Εάν με την επιλογή του κουμπιού "back" μπορούμε να έχουμε πρόσβαση στις προηγούμενες σελίδες, αλλά δεν μπορούμε να έχουμε πρόσβαση σε νέες, τότε απλά έχουμε πρόσβαση στην μνήμη του browser. Σε περίπτωση που αυτές οι σελίδες περιέχουν ευαίσθητα στοιχεία, σημαίνει ότι η εφαρμογή δεν απαγόρευσε στο browser να τα αποθηκεύσει δηλαδή δεν έγινε ρύθμιση της ετικέτας "Cache-Control". Πρέπει να σημειωθεί ότι αυτή η δοκιμή ισχύει μόνο για τα cookies συνόδου και ότι ένα μόνιμο cookie, που αποθηκεύει στοιχεία μόνο για μερικές ασήμαντες προτιμήσεις χρηστών όπως ο τρόπος εμφάνισης των ιστοσελίδων και που δεν διαγράφεται, όταν ο χρήστης αποσυνδέεται, δεν πρόκειται να θεωρηθεί ως κίνδυνος ασφάλειας.

Είναι ανάγκη λοιπόν να εξετάσουμε κάτι πιο περίπλοκο: μπορούμε να ρυθμίσουμε πάλι το cookie στην αρχική του τιμή και να ελέγξουμε, εάν μπορούμε ακόμα να έχουμε πρόσβαση στην εφαρμογή σε μια φόρμα αυθεντικοποίησης. Εάν όντως έχουμε ακόμη πρόσβαση, σημαίνει, πως δεν υπάρχει ένας μηχανισμός στην πλευρά του server, που να παρακολουθεί τα ενεργά και μη cookies, αλλά ότι η ακρίβεια των πληροφοριών, που αποθηκεύονται στο cookie, είναι αρκετή για να χορηγήσει την πρόσβαση στην εφαρμογή. Για να καθορίσουμε μια επιθυμητή τιμή στο cookie μπορούμε να χρησιμοποιήσουμε εργαλεία όπως είναι το WebScarab, τα οποία λειτουργούν σαν proxy και παρεμβάλλοντας μια απάντηση της εφαρμογής και να εισάγουμε μια ετικέτα Set-Cookie με τις επιθυμητές τιμές.

Ένα ξεχωριστό παράδειγμα ενός σχεδιασμού, στο οποίο δεν υπάρχει κανένας έλεγχος στην πλευρά του web server σχετικά με τα cookies που ανήκουν σε χρήστες, οι οποίοι έχουν ήδη αποσυνδεθεί, είναι το ASP.NET FormsAuthentication class, στο οποίο το cookie είναι βασικά μια κρυπτογραφημένη και αυθεντικοποιημένη έκδοση των πληροφοριών του χρήστη, οι οποίες έχουν αποκρυπτογραφηθεί και ελεγχθεί από την πλευρά του server. Ενώ αυτό είναι πολύ αποτελεσματικό στην παρεμπόδιση αλλοίωσης των cookies, το γεγονός ότι ο server δεν διατηρεί ένα εσωτερικό αρχείο της κατάστασης συνόδου, σημαίνει ότι είναι δυνατό να πραγματοποιηθεί μια επίθεση επανάληψης cookies μετά την αποσύνδεση του νόμιμου χρήστη, υπό τον όρο βέβαια ότι το cookie δεν έχει λήξει ακόμη.

10

11 Αποσύνδεση Λόγω Λήξης Χρόνου (Timeout Logout)

Η ίδια προσέγγιση με την προηγούμενη ενότητα μπορεί να εφαρμοστεί κατά τον έλεγχο της αποσύνδεσης λόγω λήξης χρόνου (timeout logout). Ο καταλληλότερος χρόνος αποσύνδεσης πρέπει να ισορροπεί μεταξύ της ασφάλειας ο οποίος είναι σύντομος χρόνος αποσύνδεσης και της ευκολίας χρήσης ο οποίος είναι μεγάλος χρόνος αποσύνδεσης και εξαρτάται σημαντικά από το πόσο κρίσιμα είναι τα

στοιχεία, που χειρίζονται από την εφαρμογή. Ένας χρόνος αποσύνδεσης 60 λεπτών για ένα δημόσιο φόρουμ μπορεί να είναι αποδεκτός, αλλά ένας τέτοιος χρόνος θα ήταν πάρα πολύ μεγάλος για μια εφαρμογή τραπεζικών εργασιών. Γενικά οποιαδήποτε εφαρμογή δεν επιβάλλει μια αποσύνδεση λόγω λήξης χρόνου, πρέπει να θεωρείται μη ασφαλής, εκτός αν απαιτείται μια τέτοια συμπεριφορά λόγω των λειτουργικών απαιτήσεων της.

Η μεθοδολογία εξέτασης είναι παρόμοια με αυτήν που περιγράφεται στην προηγούμενη παράγραφο. Πρώτα πρέπει να ελέγξουμε εάν υπάρχει χρόνος λήξης, κάνοντας μια σύνδεση και έπειτα αφήνοντας κάποιο χρόνο χωρίς να κάνουμε κάποια ενέργεια, περιμένοντας να προκληθεί η αποσύνδεση λόγω λήξης χρόνου. Όπως συμβαίνει στη συνάρτηση αποσύνδεσης, έτσι και εδώ, αφού έχει περάσει το συγκεκριμένο χρονικό διάστημα, όλα τα σημεία συνόδου πρέπει να καταστραφούν ή να γίνουν ακατάλληλα προς χρήση. Πρέπει επίσης να διαπιστώσουμε, εάν το timeout επιβάλλεται από τον χρήστη ή από το server (ή και τους δύο). Επιστρέφοντας στο παράδειγμα μας των cookies, εάν το cookie συνόδου δεν είναι μόνιμο ή γενικότερα το σημείο συνόδου δεν αποθηκεύει οποιαδήποτε στοιχεία σχετικά με το χρόνο, μπορούμε να είμαστε βέβαιοι ότι το timeout επιβάλλεται από το server. Εάν το σημείο συνόδου περιέχει μερικά στοιχεία σχετικά με το χρόνο όπως ο χρόνος σύνδεσης ή ο τελευταίος χρόνος πρόσβασης ή η ημερομηνία λήξης ενός μόνιμου cookie τότε ξέρουμε ότι ο χρήστης εμπλέκεται στην επιβολή του timeout. Σε αυτήν την περίπτωση πρέπει να τροποποιήσουμε το σημείο συνόδου εάν δεν προστατεύεται με κρυπτογράφηση και να δούμε τι συμβαίνει στη σύνοδό μας. Μπορούμε να θέσουμε για παράδειγμα την ημερομηνία λήξης των cookies στο μακρινό μέλλον και να εξετάσουμε, εάν η σύνοδός μας μπορεί να παραταθεί. Κατά κανόνα όλα πρέπει να είναι ελεγχόμενα από την πλευρά του server και δεν πρέπει να είναι δυνατή η ρύθμιση των cookies σε προηγούμενες τιμές, ώστε να υπάρχει ξανά η δυνατότητα πρόσβασης στην εφαρμογή.

12

13 Αποθηκευμένες Σελίδες

Με την αποσύνδεση από μια εφαρμογή, προφανώς δεν καθαρίζεται η μνήμη του browser από οποιεσδήποτε ευαίσθητες πληροφορίες που έχουν αποθηκευτεί. Επομένως μια άλλη δοκιμή που πρέπει να εκτελεσθεί είναι να ελέγξουμε ότι με την αίτησή μας δεν αποθηκεύονται οποιαδήποτε κρίσιμα στοιχεία στην μνήμη του browser. Προκειμένου να γίνει αυτό, μπορούμε να χρησιμοποιήσουμε εργαλεία όπως το WebScarab και να αναζητήσουμε τις απαντήσεις του server κατά τη διάρκεια της συνόδου μας, ελέγχοντας αν για κάθε σελίδα που περιέχει ευαίσθητες πληροφορίες ο server έδωσε εντολή στο browser να μην αποθηκεύσει οποιαδήποτε στοιχεία. Μια τέτοια εντολή μπορεί να καθοριστεί στις HTTP ετικέτες απάντησης:

HTTP/1.1:
Cache-Control: no-cache
HTTP/1.0:
Pragma: no-cache
Expires: <past date or illegal value (e.g.: 0)>

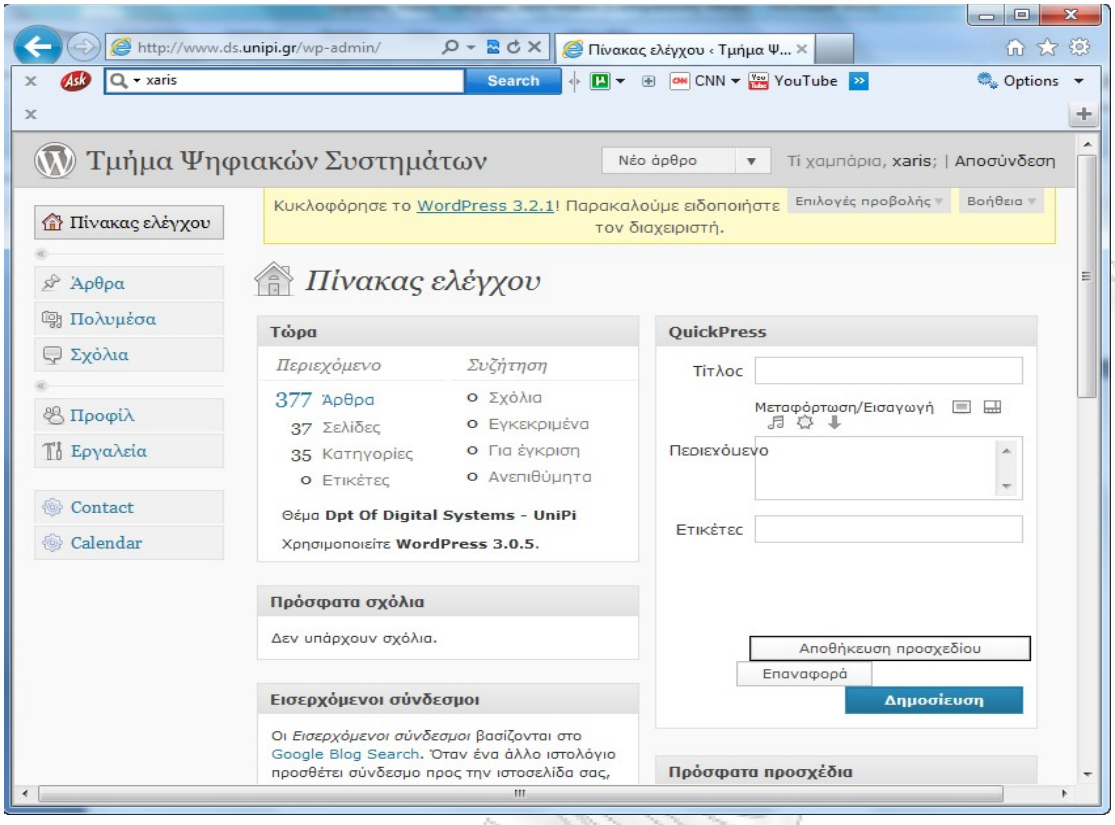
Εναλλακτικά το ίδιο μπορεί να καθοριστεί άμεσα σε επίπεδο HTML περιλαμβάνοντας σε κάθε σελίδα, που περιέχει ευαίσθητα στοιχεία, τον ακόλουθο κώδικα:

HTTP/1.1:
<META HTTP-EQUIV="Cache-Control" CONTENT="no-cache">
HTTP/1.0:
<META HTTP-EQUIV="Pragma" CONTENT="no-cache">
<META HTTP-EQUIV=”Expires” CONTENT=”Sat, 01-Jan-2000 00:00:00 GMT”>

Εάν εξετάζουμε για παράδειγμα μια εφαρμογή ηλεκτρονικού εμπορίου (ecommerce), οφείλουμε να ψάξουμε όλες τις σελίδες, που περιέχουν κωδικό πιστωτικών καρτών ή κάποιες άλλες οικονομικές πληροφορίες και να ελέγξουμε, αν όλες αυτές οι σελίδες επιβάλλουν την εντολή no-cache. Αντίθετα, εάν βρούμε σελίδες, που περιέχουν κρίσιμες πληροφορίες αλλά αποτυγχάνουν να δώσουν εντολή στο browser να μην αποθηκεύσει το περιεχόμενό τους, τότε ξέρουμε ότι αυτές οι ευαίσθητες πληροφορίες θα αποθηκευτούν στο δίσκο και μπορούμε να το ελέγξουμε αυτό απλά με τον έλεγχο της μνήμης του browser. Η ακριβής θέση αποθήκευσης αυτών των πληροφοριών εξαρτάται από το λειτουργικό σύστημα του χρήστη και από το browser, που έχει χρησιμοποιηθεί. Ένα παράδειγμα θέσης αποθήκευσης για τον Internet Explorer δίνεται παρακάτω:

C:\Documents and Settings\<user_name>\Local Settings\Temporary Internet Files>

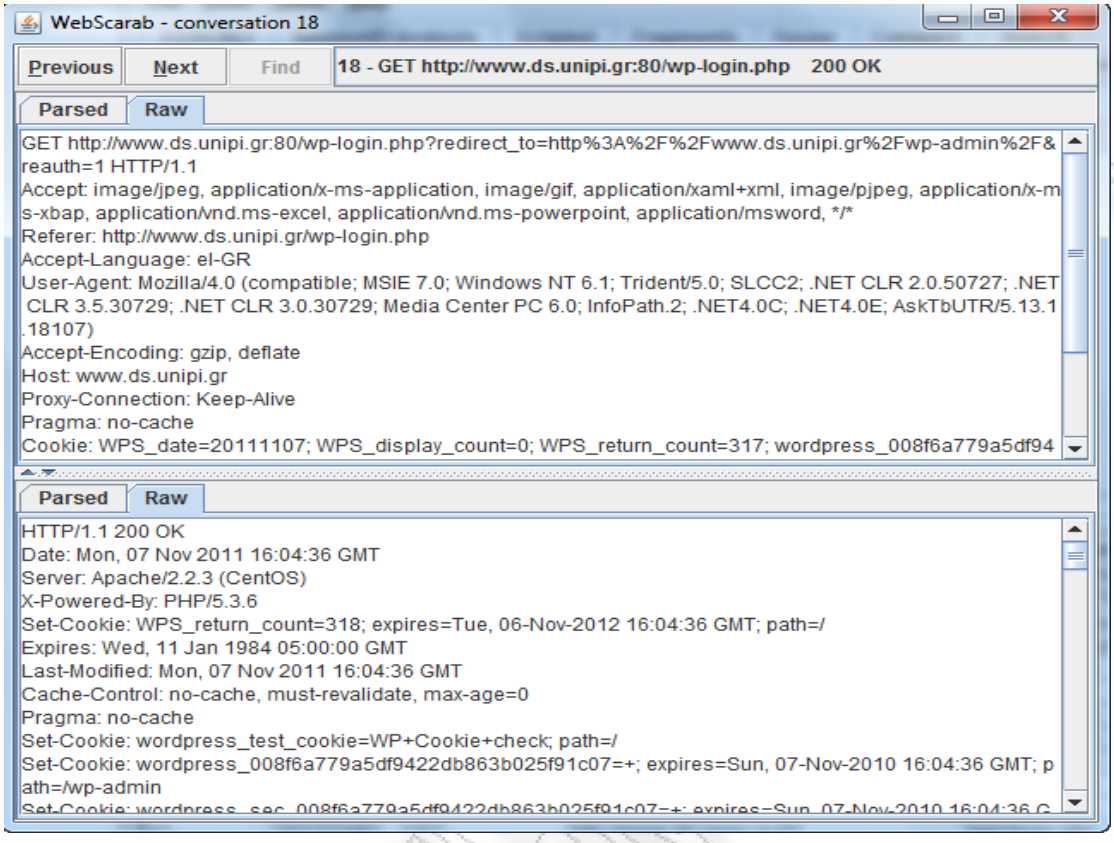
Σχετικά με τη λειτουργία Logout στην δική μας περίπτωση και συγκεκριμένα στον ιστότοπο που εξετάζουμε διαπιστώσαμε πως υπάρχει εμφανές κουμπί αποσύνδεσης κάτι που φαίνεται και στην εικόνα που ακολουθεί.



Εικόνα : **Η σελίδα διαχείρισης του www.ds.unipi.gr με την επιλογή "Αποσύνδεση" να εμφανίζεται πάνω δεξιά.**

Επιλέγοντας να αποσυνδεθούμε και με τη χρήση του κουμπιού "Back" του browser να δούμε τι συμβαίνει διαπιστώνουμε πως έχουμε πρόσβαση στις σελίδες που είχαμε επισκεφθεί ενώ ήμασταν κανονικά συνδεδεμένοι όμως δεν έχουμε σε καινούριες. Αυτό πρακτικά για την εφαρμογή που εξετάζουμε σημαίνει πως ναι μεν δεν παραμένουμε συνδεδεμένοι όμως έχουμε πρόσβαση στην μνήμη του browser κάτι που σημαίνει πως αν οι σελίδες που επισκεφτήκαμε περιείχαν ευαίσθητα δεδομένα θα μπορούσαν αυτά να γίνουν διαθέσιμα σε οποιονδήποτε θα έκανε χρήση αυτού του κουμπιού. Επίσης δοκιμάσαμε υποβάλλοντας ένα χειροκίνητο αίτημα μέσω του proxy και με τη χρήση μιας παλιάς τιμής του cookie να επιστρέψουμε στην σελίδα που αναφέρεται σε αυτό το cookie και διαπιστώσαμε πως δεν μπορεί να συμβεί αυτό καθώς η εφαρμογή δεν αποθηκεύει τα cookies των προηγούμενων συνόδων.

Όσον αφορά το Timeout Logout των σελίδων της εφαρμογής μας διαπιστώσαμε πως δεν υπάρχει ορισμένο κάποιο χρονικό διάστημα μικρό. Συγκεκριμένα ο χρόνος λήξης των cookies που χρησιμοποιούνται είναι περίπου ένα χρόνο μετά την ημερομηνία εφαρμογής τους άρα πρακτικά υπάρχει άπειρος χρόνος για την περίπτωση του Timeout Logout. Επίσης ελέγχοντας το σώμα των απαντήσεων server που εμφανίζονται στον proxy χρησιμοποιούμε (WebScarab) βλέπουμε πως γίνεται χρήση της ετικέτας "Cache-Control: no-cache" κάτι που δείχνει πως δεν επιτρέπεται η αποθήκευση των σελίδων. Στην εικόνα που ακολουθεί φαίνονται οι παραπάνω παρατηρήσεις:



Εικόνα : **Απάντηση του server σε αίτημα όπου φαίνονται οι περίοδος διάρκειας των cookies και η χρήση της ετικέτας "Cache-Control".**

3.4.8 Testing for CAPTCHA Weak

Ο CAPTCHA ("Completely Automated Public Turing test to tell Computers and Humans Apart") είναι ένα είδος ελέγχου πρόκλησης-απόκρισης που χρησιμοποιείται από πολλές εφαρμογές ιστού για να εξασφαλίσουν ότι η απάντηση δεν έχει δημιουργηθεί από κάποιον Η/Υ. Οι υλοποιήσεις CAPTCHA είναι συχνά ευπαθείς διάφορα είδη επιθέσεων ακόμη και αν ο CAPTCHA που δημιουργήθηκε είναι αδιάσπαστος. Αυτή η ενότητα θα μας βοηθήσει στο να αναγνωρίζουμε αυτά τα είδη των επιθέσεων. Παρόλο που ο CAPTCHA δεν είναι ένας έλεγχος αυθεντικοποίησης, η χρήση του μπορεί να είναι πολύ αποτελεσματική απέναντι σε:

- Επιθέσεις καταμέτρησης (σύνδεση, εγγραφή ή επαναφορά κωδικού πρόσβασης είναι συχνά ευπαθή σε επιθέσεις καταμέτρησης. Χωρίς έλεγχο CAPTCHA ο επιτιθέμενος μπορεί να αποκτήσει έγκυρα ονόματα χρηστών, αριθμούς τηλεφώνων ή οποιαδήποτε άλλη ευαίσθητη πληροφορία σε πολύ μικρό χρονικό διάστημα)

- Αυτόματη αποστολή πολλών GET/POST αιτημάτων σε λίγο χρονικό διάστημα εκεί όπου είναι ανεπιθύμητα (πχ υπερχείλιση από SMS/MMS/email). Ο CAPTCHA παρέχει περιορισμένου ρυθμό λειτουργίας.
- Αυτοματοποιημένη δημιουργία/χρήση του λογαριασμού που θα πρέπει να χρησιμοποιείται μόνο από ανθρώπους (πχ δημιουργία Web λογαριασμών, διακοπή του spamming)
- Αυτοματοποιημένο posting σε blogs, forums και wikis ακόμη και σαν αποτέλεσμα εμπορικής προώθησης, παρενόχλησης ή βανδαλισμού.
- Οποιαδήποτε αυτοματοποιημένη επίθεση που μαζικά αποκτά ή κάνει κακή χρήση των ευαίσθητων πληροφοριών της εφαρμογής.

Η χρήση του CAPTCHA σαν μια CSRF προστασία δεν συστήνεται επειδή υπάρχουν ισχυρότερα CSRF αντίμετρα. Αυτές οι ευπάθειες είναι αρκετά κοινές σε πολλές CAPTCHA υλοποιήσεις:

- Η CAPTCHA εικόνα που δημιουργείται είναι αδύναμη κάτι που μπορεί να αναγνωριστεί μόνο από απλή σύγκριση με ήδη «σπασμένα» CAPTCHAs.
- Οι CAPTCHA ερωτήσεις που δημιουργούνται έχουν ένα περιορισμένο σύνολο πιθανών απαντήσεων.
- Η τιμή των αποκωδικοποιημένων CAPTCHA αποστέλλεται από τον client σαν μια GET παράμετρος ή σαν ένα κρυφό πεδίο σε μια φόρμα POST. Αυτή η τιμή είναι συχνά:
    - Κρυπτογραφημένη από έναν απλό αλγόριθμο και μπορεί εύκολα να αποκρυπτογραφηθεί ακολουθώντας του πολλαπλές CAPTCHA τιμές της αποκρυπτογράφησης.
    - Κατακερματισμένη από μια αδύναμη συνάρτηση κατακερματισμού (πχ MD5) που μπορεί να σπάσει χρησιμοποιώντας έναν rainbow πίνακα.
- Πιθανότητα επαναλαμβανόμενων επιθέσεων:
    - Η εφαρμογή δεν κρατά ίχνη αναφορικά με ποιο ID της CAPTCHA εικόνας στέλνεται στο χρήστη. Ως εκ τούτου, ο επιτιθέμενος μπορεί απλά να αποκτήσει μια κατάλληλη CAPTCHA εικόνα και το ID του να το λύσει και να στείλει την τιμή του αποκρυπτογραφημένου CAPTCHA με το αντίστοιχο ID (το ID ενός CAPTCHA θα μπορούσε να είναι ένας πίνακας κατακερματισμού των αποκρυπτογραφημένων CAPTCHA ή ένας μοναδικός προσδιοριστής)
    - Η εφαρμογή δεν καταστρέφει τη σύνοδο (session) όταν εισάγονται σωστές φράσεις. Με την επαναχρησιμοποίηση του ID της συνόδου ενός ήδη γνωστού CAPTCHA είναι δυνατόν να παρακάμψουμε την προστατευόμενη CAPTCHA σελίδα.

BLACK BOX TESTING

Μπορούμε να χρησιμοποιήσουμε έναν ενδιάμεσο proxy (πχ WebScarab) για να:

- Προσδιορίσουμε όλες τις παραμέτρους που αποστέλλονται εκτός από την αποκωδικοποιημένη τιμή του CAPTCHA από τον client στον server (αυτές οι παράμετροι μπορεί να περιέχουν κρυπτογραφημένες ή κατακερματισμένες τιμές από αποκρυπτογραφημένα CAPTCHA ή CAPTCHA ID αριθμούς)
- Προσπαθήσουμε να στείλουμε μια παλιότερη αποκωδικοποιημένη τιμή CAPTCHA μαζί με ένα παλιό CAPTCHA ID (εάν η εφαρμογή τις δέχεται καθώς είναι ευπαθείς σε επαναλαμβανόμενες επιθέσεις)
- Προσπαθήσουμε να στείλουμε μια παλιότερη αποκωδικοποιημένη τιμή CAPTCHA μαζί με ένα παλιό ID συνόδου (εάν η εφαρμογή τις δέχεται καθώς είναι ευπαθείς σε επαναλαμβανόμενες επιθέσεις)

Καλό θα ήταν να προσπαθήσουμε να βρούμε παρόμοια CAPTCHAs που έχουν ήδη "σπάσει". Η επιβεβαίωση του συνόλου των απαντήσεων για CAPTCHA είναι περιορισμένη και μπορεί εύκολα να προσδιοριστεί.

Στον ιστότοπο που εξετάζουμε στην παρούσα εργασία, www.ds.unipi.gr , δεν χρησιμοποιούνται CAPTCHAs οπότε δεν μπορούμε να πραγματοποιήσουμε τον έλεγχο αυτής της κατηγορίας.

3.4.9 Testing for Multiple Factors Authentication

Η αξιολόγηση της ισχύς ενός «Multiple Factors Authentication System»(Συστήματος Αυθεντικοποίησης Πολλαπλών Παραγόντων» είναι ένα κρίσιμο ζήτημα για τον ελεγκτή της δοκιμής διείσδυσης. Οι τράπεζες και άλλα χρηματοπιστωτικά ιδρύματα σκοπεύουν να ξοδέψουν σημαντικά ποσά για ακριβά MFAS. Ως εκ τούτου η εκτέλεση ελέγχων με ακρίβεια πριν από την υιοθέτηση της συγκεκριμένης λύσεις είναι προτείνεται ανεπιφύλακτα. Επιπλέον, μια περαιτέρω ευθύνη του ελεγκτή είναι η να αναγνωρίσει την εάν το ήδη υιοθετημένο MFAS είναι αποτελεσματικά ικανό να υπερασπιστεί τα περιουσιακά στοιχεία του οργανισμού από τις απειλές που οδηγούν γενικά στην υιοθέτηση ενός MFAS.

Γενικά ο στόχος του συστήματος αυθεντικοποίησης δύο παραγόντων είναι να ενισχύσει την ισχύ της διαδικασίας αυθεντικοποίησης. Αυτός ο στόχος μπορεί να επιτευχθεί ελέγχοντας έναν επιπρόσθετο παράγοντα, ή «κάτι που έχουμε» όπως επίσης κάτι που γνωρίζουμε, εξασφαλίζοντας με σιγουριά πως ο χρήστης κατέχει μια hardware συσκευή κάποιου είδους εκτός από τον κωδικό πρόσβασης. Η hardware συσκευή που παρέχεται στον χρήστη μπορεί να είναι σε θέση να επικοινωνήσει απευθείας και ανεξάρτητα με την αρχιτεκτονική αυθεντικοποίησης χρησιμοποιώντας ένα επιπρόσθετο κανάλι επικοινωνίας. Αυτό το συγκεκριμένο χαρακτηριστικό είναι κάτι που είναι γνωστό ως «διαχωρισμός των καναλιών». Ο Bruce Schneier το 2005 παρατήρησε ότι πριν μερικά χρόνια «οι απειλές ήταν πιο παθητικές: Υποκλοπές και διαδικασίες να μαντέψει κάποιος τον κωδικό πρόσβασης. Σήμερα, οι απειλές είναι πιο ενεργητικές: Phishing και Δούρειοι Ίπποι (Trojan horses)».

Στην πραγματικότητα οι συνηθισμένες απειλές που ένα MFAS θα πρέπει να αντιμετωπίσει σωστά σε ένα διαδικτυακό περιβάλλον περιλαμβάνουν:

1. Κλοπή Συνθηματικών
2. Αδύναμα συνθηματικά
3. Επιθέσεις που βασίζονται στις συνόδους
4. Επιθέσεις Δουρείων Ίππων και κακόβουλες επιθέσεις
5. Επαναχρησιμοποίηση κωδικών

Η βέλτιστη λύση θα πρέπει να είναι σε θέση να αντιμετωπίσει όλες τις πιθανές επιθέσεις που σχετίζονται με τις 5 παραπάνω κατηγορίες. Δεδομένου ότι η ισχύς μιας λύσης αυθεντικοποίησης γενικά ταξινομείται ανάλογα με το πόσοι παράγοντες αυθεντικοποίησης ελέγχονται μέχρι ο χρήστης να έρθει σε επαφή με το υπολογιστικό σύστημα, η τυπική συμβουλή του IT ειδικού είναι: "Αν δεν είστε ευχαριστημένοι με την ισχύουσα λύση αυθεντικοποίησης, απλά προσθέστε έναν ακόμη παράγοντα αυθεντικοποίησης και όλα θα είναι μια χαρά". Δυστυχώς, ο κίνδυνος που σχετίζεται με τις επιθέσεις που εκτελούνται από υποκινούνται από τους επιτιθέμενους δεν μπορούν να εξαλειφθούν πλήρως. Επιπλέον κάποιες MFAS λύσεις είναι πιο ευέλικτες και ασφαλείς σε σύγκριση με κάποιες άλλες. Θεωρώντας τις 5 απειλές που είδαμε παραπάνω θα μπορούσαμε να αναλύσουμε την ισχύ μιας MFAS λύσης δεδομένου ότι η λύση είναι σε θέση να αντιμετωπίσει, περιορίσει ή να μην αντικαταστήσει αυτή τη συγκεκριμένη διαδικτυακή επίθεση.

Αναφορικά με τον έλεγχο πολλαπλών παραγόντων αυθεντικοποίησης, το testing guide του OWASP που ακολουθούμε για την πραγματοποίηση αυτών των ελέγχων δεν προτείνει κάποιον τρόπο για black-box testing οπότε και δεν θα ασχοληθούμε περαιτέρω στην παρούσα εργασία με το είδος των ελέγχων αυτών.

3.4.10 Testing for Race Conditions

Μια συνθήκη "αγώνα δρόμου" είναι ένα ελάττωμα που παράγει απρόσμενα αποτελέσματα όταν χρονική στιγμή κάποιων ενεργειών επηρεάζουν άλλες ενέργειες. Μπορούμε να δούμε ένα παράδειγμα σε μια πολύ-νηματική εφαρμογή όπου οι διάφορες ενέργειες διεξάγονται στα ίδια δεδομένα. Οι συνθήκες "αγώνα δρόμου" είναι από τη φύση τους δύσκολο να ελεγχθούν.

Οι συνθήκες "αγώνα δρόμου" μπορούν να εμφανιστούν όταν μια διαδικασία εξαρτάται απρόσμενα ή με τρόπο κρίσιμο από τη διαδοχή ή την χρονική στιγμή άλλων γεγονότων. Σε ένα περιβάλλον μιας εφαρμογής ιστού, όπου πολλαπλά αιτήματα μπορούν να επεξεργάζονται σε μια δεδομένη στιγμή, οι προγραμματιστές πρέπει να αφήσει τα θέματα συγχρονισμού να τα χειριστεί το πλαίσιο(framework), ο server ή η γλώσσα προγραμματισμού. Το απλοποιημένο παράδειγμα που ακολουθεί αποτυπώνει ένα πιθανό πρόβλημα συγχρονισμού σε μια web εφαρμογή συναλλαγών και σχετίζεται με έναν κοινό λογαριασμού καταθέσεων στον οποία δυο χρήστες εισάγονται στον ίδιο λογαριασμό και επιχειρούν μια μεταφορά:

Ο λογαριασμός Α έχει 100 ευρώ

Ο λογαριασμός Β έχει 100 ευρώ

Οι χρήστες 1 και 2 θέλουν να μεταφέρουν 10 ευρώ από τον λογαριασμό Α στον Β.
Εάν η μεταφορά ήταν σωστή το αποτέλεσμα θα ήταν:

Ο λογαριασμός Α έχει 80 ευρώ
Ο λογαριασμός Α έχει 120 ευρώ

Ωστόσο, λόγω ζητημάτων συγχρονισμού, θα μπορούσαμε πάρουμε το παρακάτω αποτέλεσμα:

Ο χρήστης 1 ελέγχει το υπόλοιπο του λογαριασμού Α (=100 ευρώ)
Ο χρήστης 2 ελέγχει το υπόλοιπο του λογαριασμού Α (=100 ευρώ)
Ο χρήστης 2 παίρνει 10 ευρώ από τον λογαριασμό Α (=90 ευρώ) και τα τοποθετεί στον λογαριασμό Β(=110 ευρώ).
Ο χρήστης 1 παίρνει 10 ευρώ από τον λογαριασμό Α (=που πιστεύει ότι έχει 100 ευρώ)(=90 ευρώ) και τα τοποθετεί στον λογαριασμό Β(=120 ευρώ).

<u>Αποτέλεσμα:</u> Ο λογαριασμός Α έχει 90 ευρώ.
Ο λογαριασμός Β έχει 120 ευρώ.

Όλα τα αποτελέσματα του παραπάνω του παραδείγματος συμβαίνουν εξαιτίας της ανταλλαγής δεδομένων μεταξύ της στιγμής του ελέγχου και στης στιγμής της χρήσης.

<u>BLACK BOX TESTING</u>

Ο έλεγχος για συνθήκες «αγώνα δρόμου» είναι προβληματικός εξαιτίας της φύσης του ελέγχου και των εξωτερικών επιδράσεων στις δοκιμές που γίνονται συμπεριλαμβανομένων του φορτίου του server, την αναποτελεσματικότητα του δικτύου, κλπ που θα παίξουν ρόλο στην ισχύουσα κατάσταση αλλά και στην ανακάλυψη κάποιας άλλης. Ωστόσο, ο έλεγχος μπορεί να εστιάσει σε συγκεκριμένες περιοχές συναλλαγών της εφαρμογής, όπου η χρονική στιγμή της αντίληψης και η χρονική στιγμή της χρήσης συγκεκριμένων μεταβλητών των δεδομένων θα μπορούσε να επηρεαστεί από ζητήματα συγχρονισμού.

Η Black Box τεχνική ελέγχου προσπαθεί να δείξει πως μια συνθήκη «αγώνα δρόμου» μπορεί να περιλαμβάνει την ικανότητα να πραγματοποιεί πολλαπλά ταυτόχρονα αιτήματα καθώς παρατηρεί το αποτέλεσμα μιας απροσδόκητης συμπεριφοράς. Οι ελεγκτές θα πρέπει να γνωρίζουν τις συνέπειες ,αναφορικά με την ασφάλεια, των συνθηκών «αγώνων δρόμου» και τους παράγοντες που περιβάλλουν τη δυσκολία του ελέγχου αυτού. Γι' αυτό προτείνεται ότι κάτω από ορισμένες συνθήκες είναι δυνατό να:

- Δημιουργηθούν πολλαπλοί λογαριασμοί χρηστών με το ίδιο όνομα χρήστη.
- Παρακαμφθούν τα κλειδώματα των λογαριασμών ενάντια σε επιθέσεις brute force(ωμής βίας).

Στην περίπτωση του ιστότοπου που εξετάζουμε στην παρούσα μεταπτυχιακή διπλωματική εργασία η εφαρμογή που τρέχει στον ιστότοπο αυτόν δεν επιτρέπει τη δημιουργία πολλαπλών λογαριασμών χρηστών με το ίδιο όνομα άρα καταστάσεις συνθηκών αγώνα δρόμου δεν μπορούν να δημιουργηθούν εξαιτίας αυτής της αιτίας παρόλο που δεν ισχύει ο μηχανισμός κλειδώματος λογαριασμών σε περιπτώσεις επιθέσεων brute force. Θα πρέπει επίσης να λάβουμε υπ' όψιν πως ο συγκεκριμένος ιστότοπος δεν δίνει τη δυνατότητα στο να δημιουργηθούν χρήστες που να μπορούν να υποβάλλουν δεδομένα που να αποθηκεύονται στη βάση δεδομένων άρα οι συνθήκες αγώνα δρόμου αφορούν μόνο τις περιπτώσεις όπου δύο ή περισσότερα φυσικά πρόσωπα προσπαθήσουν ταυτόχρονα ως διαχειριστές να προκαλέσουν κάποια αλλαγή στο περιεχόμενο του ιστότοπου και έτσι να δημιουργηθεί κάποια τέτοια συνθήκη. Στην περίπτωση αυτή ελέγχοντας περιπτώσεις ταυτόχρονης υποβολής αλλαγών σε δεδομένα του ιστότοπου δεν παρουσιάστηκαν περιπτώσεις συνθηκών αγώνων δρόμου.

3.5 Authorization Testing

Η εξουσιοδότηση (authorization) είναι η λειτουργία που επιτρέπει την πρόσβαση σε προστατευόμενους πόρους, μόνο σε αυτούς που επιτρέπεται να τους χρησιμοποιήσουν. Ο έλεγχος εξουσιοδότησης(Authorization Testing) είναι η κατανόηση της διαδικασίας της εξουσιοδότησης και η χρήση της γνώσης αυτής, για την παράκαμψη του μηχανισμού εξουσιοδότησης. Η εξουσιοδότηση είναι η διαδικασία που λαμβάνει χώρα κατόπιν επιτυχούς αυθεντικοποίησης και έτσι ο ελεγκτής από αυτό το σημείο και μετά, θα επιβεβαιώσει την διαδικασία ενώ έχει έγκυρα στοιχεία εισόδου συνδεδεμένα με ένα καλά ορισμένο σύνολο από ρόλους και δικαιώματα. Κατά την διάρκεια αυτού του είδους της εκτίμησης, πρέπει να διευκρινισθεί αν είναι δυνατόν να παρακαμφθεί ο μηχανισμός εξουσιοδότησης, να βρεθεί κάποια αδυναμία στην προσβασιμότητα των αρχείων ή κάποιος άλλος τρόπος να αποκτήσει ο ελεγκτής δικαιώματα που δεν του ανήκουν.

3.5.1 Testing for Path Traversal

Πολλές web εφαρμογές χρησιμοποιούν και διαχειρίζονται αρχεία ως μέρος της καθημερινή τους λειτουργίας. Με τη χρήση μεθόδων εισόδου που δεν έχουν σχεδιαστεί ή υλοποιηθεί σωστά, ο επιτιθέμενος θα μπορούσε να εκμεταλλευτεί το σύστημα με σκοπό να αποκτήσει δικαιώματα ανάγνωσης ή εγγραφής σε αρχεία που δεν θα έπρεπε να ήταν προσβάσιμα. Σε συγκεκριμένες περιπτώσεις, είναι δυνατό να εκτελεσθεί κώδικας ή ακόμη και εντολές του συστήματος.

Παραδοσιακά η εξυπηρετητές και οι web εφαρμογές υλοποιούν κάποιο μηχανισμό αυθεντικοποίησης για τον έλεγχο της πρόσβασης σε αρχεία και πόρους. Οι εξυπηρετητές προσπαθούν να αποκλείσουν τους χρήστες σε ένα φάκελο που αναπαριστά μια φυσική θέση στο σύστημα αρχείων. Οι χρήστες θεωρούν αυτόν το φάκελο ως τη βάση της ιεραρχίας αρχείων της web εφαρμογής. Ο ορισμός των δικαιωμάτων γίνεται με τη χρήση Λιστών Ελέγχου Πρόσβασης (Access Control Lists) που καθορίζουν ποιοι χρήστες ή ομάδες χρηστών μπορούν να προσπελάσουν, να επεξεργαστούν ή να εκτελέσουν ένα αρχείο στον εξυπηρετητή. Αυτοί οι μηχανισμοί, είναι σχεδιασμένοι, να εμποδίζουν την

πρόσβαση σε ευαίσθητα αρχεία από κακόβουλους χρήστες (για παράδειγμα το αρχείο /etc/passwd σε ένα Unix based σύστημα) και την εκτέλεση εντολών του συστήματος.

Πολλές web εφαρμογές χρησιμοποιούν κομμάτια κώδικα εκτελέσιμα από τον εξυπηρετητή με σκοπό την διαχείριση διαφορετικών ειδών αρχείων. Αυτή η μέθοδος είναι ιδιαίτερα κοινή στην διαχείριση γραφικών, αρχείων template κτλ. Δυστυχώς αυτές οι εφαρμογές αποκαλύπτουν αρκετές αδυναμίες αν οι παράμετροι εισόδου δεν επικυρώνονται σωστά (πχ παράμετροι σε φόρμες, cookies). Σε web servers και εφαρμογές, αυτού του τύπου τα προβλήματα αποκαλύπτονται σε επιθέσεις τύπου διάσχισης μονοπατιού. Με την εκμετάλλευση αυτού του τύπου των αδυναμιών, ο επιτιθέμενος μπορεί να αναγνώσει φακέλους ή αρχεία που δεν θα έπρεπε, να προσπελάσει αρχεία έξω από τον βασικό φάκελο του web, ή να συμπεριλάβει κομμάτια κώδικα ή αρχεία από τρίτα sites. Για τους σκοπούς της εργασίας θα λάβουμε υπόψη τις απειλές προς web εφαρμογές.

Αυτού του τύπου η επίθεση είναι επίσης γνωστή ως επίθεση dot-dot-slash(../), directory traversal, directory climbing, ή backtracking.Κατά την διάρκεια της εκτίμησης, με σκοπό να ανακαλύψουμε διάσχιση μονοπατιού ή αδυναμίες στα αρχεία, πρέπει να εκτελέσουμε μια διαδικασία δύο σταδίων:

- Αναγνώριση και Εκτίμηση των σημείων εισόδου δεδομένων
- Τεχνικές δοκιμής (μια μεθοδική εκτίμηση κάθε τεχνικής επίθεσης που μπορεί να χρησιμοποιηθεί από το επιτιθέμενο για να εκμεταλλευτεί μια αδυναμία).

BLACK BOX TESTING

A. **Αναγνώριση και Εκτίμηση των σημείων εισόδου δεδομένων**

Με σκοπό να καθοριστούν ποια σημεία της εφαρμογής είναι εκτεθειμένα σε τεχνικές παράκαμψης της διαδικασίας επικύρωσης, των δεδομένων εισόδου, πρέπει να απαριθμηθούν όλα τα σημεία της εφαρμογές στα οποία δέχεται είσοδο από τον χρήστη. Αυτό συμπεριλαμβάνει και αιτήματα http GET και POST καθώς και κοινές επιλογές όπως τον ανέβασμα αρχείων και οι html φόρμες. Μερικά παραδείγματα ελέγχων που διενεργούνται σε αυτό το στάδιο:

- Υπάρχουν παράμετροι αιτημάτων που μπορούν να χρησιμοποιηθούν σε διαδικασίες που εμπλέκουν αρχεία;
- Υπάρχουν ασυνήθιστες καταλήξεις αρχείων;
- Υπάρχουν ενδιαφέροντα ονόματα μεταβλητών;
  - *http://example.com/getUserProfile.jsp?item=ikki.html*
  - *http://example.com/index.php?file=content*
  - *http://example.com/main.cgi?home=index.htm*

Είναι δυνατό να αναγνωριστούν cookies της web εφαρμογής που χρησιμοποιούνται για την δυναμική παραγωγή σελίδων;

Cookie:
ID=d9ccd3f4f9f18cc1:TM=2166255468:LM=1162655568:S=3cFpqbJgMSSPKVMV:TEMPLATE=flower

Cookie: USER=1826cc8f:PSTYLE=GreenDotRed

B. **Τεχνικές Δοκιμής**

Το επόμενο στάδιο δοκιμών είναι η ανάλυση των διαδικασιών εισαγωγής δεδομένων της web εφαρμογής. Με χρήση των προηγούμενων παραδειγμάτων, μια δυναμική σελίδα με όνομα getUserProfile.jsp φορτώνει δεδομένα από ένα αρχείο, προβάλλοντας την πληροφορία στο χρήστη. Ο επιτιθέμενος μπορεί να εισάγει μια κακόβουλη συμβολοσειρά, όπως το «../../../../etc/passwd» για να αποκτήσει πρόσβαση στο αρχείο κωδικών ενός Unix based/like. Προφανώς κάτι τέτοιο είναι δυνατό μόνο αν η επικύρωση των δεδομένων εισόδου αποτύχει. Σύμφωνα με τα δικαιώματα πρόσβασης στο σύστημα αρχείων, η εφαρμογή πρέπει να έχει τη δυνατότητα να αναγνώσει το αρχείο. Για εξετάσουμε με επιτυχία για την συγκεκριμένη αδυναμία, ο ελεγκτής πρέπει να έχει γνώση του συστήματος που δοκιμάζεται και της τοποθεσίας που βρίσκονται τα αρχεία που γίνεται προσπάθεια να προσπελαστούν. Δεν υπάρχει νόημα στην προσπάθεια ανάκτησης του αρχείου /etc/passwd σε εξυπηρετητή IIS.

http://example.com/getUserProfile.jsp?item=../../../../etc/passwd

Για το παράδειγμα των cookies:

Cookie: USER=1826cc8f:PSTYLE=../../../../etc/passwd

Είναι επίσης δυνατό να συμπεριλαμβάνει αρχεία και κώδικα που φιλοξενούνται σε εξωτερικούς ιστότοπους:

http://example.com/index.php?file=http://www.owasp.org/malicioustxt

Το επόμενο παράδειγμα δείχνει πως είναι δυνατό να προβάλουμε τον πηγαίο κώδικα ενός CGI script, χωρίς την χρήση χαρακτήρων διάσχισης μονοπατιού:

http://example.com/main.cgi?home=main.cgi

Το αρχείο main.cgi βρίσκεται στον ίδιο φάκελο με τα στατικά html αρχεία που χρησιμοποιεί την εφαρμογή. Σε μερικές περιπτώσεις ο ελεγκτής πρέπει να συμπεριλάβει στα αιτήματά του ειδικούς χαρακτήρες (like the "." dot, "%00" null, ...) με στόχο να παρακαμφθεί το σύστημα αρχείων ή να εμποδιστεί η εκτέλεση κώδικα. Είναι σύνηθες λάθος από τους προγραμματιστές να κάνουν ελέγχους μόνο για βασικό περιεχόμενο και όχι για κάθε δυνατή περίπτωση κωδικοποίησης. Αν η πρώτη δοκιμή δεν είναι πετυχημένη, δοκιμάστε ένα διαφορετικό σχήμα κωδικοποίησης. Κάθε λειτουργικό σύστημα χρησιμοποιεί διαφορετικούς χαρακτήρες ως διαχωριστικούς:

Unix-like OS:

root directory: "/"

directory separator: "/"

Windows OS:

root directory: "<drive letter>:\"

directory separator: "\" but also "/"

(Στα windows η διάσχιση μονοπατιού περιορίζεται σε ένα μόνο partition.)

***Classic Mac OS**:*

root directory: "<drive letter>:"

directory separator: ":"

Επίσης πρέπει να συμπεριλάβουμε του παρακάτω κωδικοποιημένους χαρακτήρες:

- Κωδικοποίηση URL και διπλή κωδικοποίηση URL

  %2e%2e%2f αναπαριστούν το '/'

  %2e%2e/ αναπαριστούν το '/'

  ..%2f αναπαριστούν το '/'

  %2e%2e%5c αναπαριστούν το '\'

  %2e%2e αναπαριστούν το '\'

  ..%5c αναπαριστούν το '\'

  %252e%252e%255c αναπαριστούν το '\'

  ..%255c αναπαριστούν το '\'και ούτω καθεξής.

- Κωδικοποίηση Unicode/UTF-8

  ..%c0%af αναπαριστούν το '/'

  ..%c1%9c αναπαριστούν το '\'

Εφαρμόσαμε στον ιστότοπο του Τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς κάποιες από τις τεχνικές περιγράφηκαν παρακάτω και διαπιστώσαμε πως η εφαρμογή που εξετάζουμε είναι ευπαθής σε επιθέσεις Path Traversal καθώς ανακαλύψαμε κάποια ενδιαφέρονται ονόματα που ισχύουν όπως το παρακάτω:

http://www.ds.unipi.gr/?file=content

Από την άλλη βέβαια δεν παρατηρήθηκαν ασυνήθιστες καταλήξεις αρχείων στον server της εφαρμογής ούτε κάποιοι παράμετροι σε αιτήματα που μπορούν να χρησιμοποιηθούν ώστε να παρακαμφθεί η επικύρωση των σελίδων.

Ακόμη σχετικά με τις τεχνικές δοκιμής που αναφέρθηκαν βρήκαμε πως η εφαρμογή μας εμφανίζει αδυναμία σε σχέση με κάποιες. Τα URLs που ακολουθούν δοκιμάστηκαν και επιστρέφουν την αιτούμενη σελίδα όταν αυτά χρησιμοποιούνται:

http://www.ds.unipi.gr/index.php?file=http://www.owasp.org/malicioustxt

http://www.ds.unipi.gr/?file=http://www.owasp.org%2e%2e%2fmalicioustxt

Όπως καταλαβαίνουμε από το δεύτερο link ισχύουν τα URLs και αναπαράσταση με διαφορετική κωδικοποίηση. Άρα συμπεραίνουμε πως η εφαρμογή μας είναι ευπαθής σε αυτόν τον τύπο ελέγχου δηλαδή σε Path Traversal.

3.5.2 Testing for by passing authorization schema

Αυτού του είδους ο έλεγχος έχει ως στόχο να ερευνήσει τον τρόπο με τον οποίο έχει υλοποιηθεί το σύστημα εξουσιοδότησης για κάθε ρόλο ή δικαίωμα, ώστε να προσπελάζονται πόροι και διαδικασίες. Για κάθε συγκεκριμένο ο ρόλο που έχει ο ελεγκτής, κατά τη διάρκεια της εκτίμησης καθώς και κάθε διαδικασία ή αίτημα που εκτελεί η εφαρμογή στην φάση μετά την εξουσιοδότηση, είναι αναγκαίο να ελεγχθεί αν:

- Είναι δυνατή η πρόσβαση σε ένα πόρο όταν ο χρήστης δεν έχει εξουσιοδοτηθεί;
- Είναι δυνατή η πρόσβαση σε ένα πόρο μετά την έξοδο του χρήστη
- Είναι δυνατή η πρόσβαση σε διαδικασίες και πόρους που θα έπρεπε να είναι προσβάσιμες σε ένα χρήστη με διαφορετικά δικαιώματα ή ρόλους
- Μετά την καταγραφή όλων των διαχειριστικών διαδικασιών, πρέπει να δοκιμαστεί αν είναι δυνατό να υπάρξει πρόσβαση σε αυτές, χωρίς ο εγγεγραμμένος χρήστης να έχει διαχειριστικά δικαιώματα.
- Είναι δυνατό κάποια από αυτές τις δυνατότητες να χρησιμοποιηθεί από χρήστη με διαφορετικό ρόλο, στον οποίο η πρόσβαση θα έπρεπε να απαγορεύεται.

BLACK BOX TESTING

Έλεγχος για Διαχειριστικές Διεργασίες

Ας υποθέσουμε για παράδειγμα την διαδικασία AddUser.jsp που είναι μέρος του διαχειριστικού μενού της εφαρμογής, και είναι προσβάσιμη από το ακόλουθο URL:

https://www.example.com/admin/addUser.jsp

Το παρακάτω http αίτημα παράγεται όταν καλείται η διαδικασία AddUser:

POST /admin/addUser.jsp HTTP/1.1
Host: www.example.com
[other HTTP headers]
userID=fakeuser&role=3&group=grp001

Τι θα συμβεί αν ένας χρήστης χωρίς διαχειριστικά δικαιώματα, δοκιμάσει να εκτελέσει το αίτημα; Θα δημιουργηθεί νέος χρήστης; Και αν ναι, ο νέος χρήστης θα κληρονομήσει τα δικαιώματα της διαδικασίας;

Έλεγχος πρόσβασης σε πόρους που έχουν ανατεθεί σε άλλους ρόλους

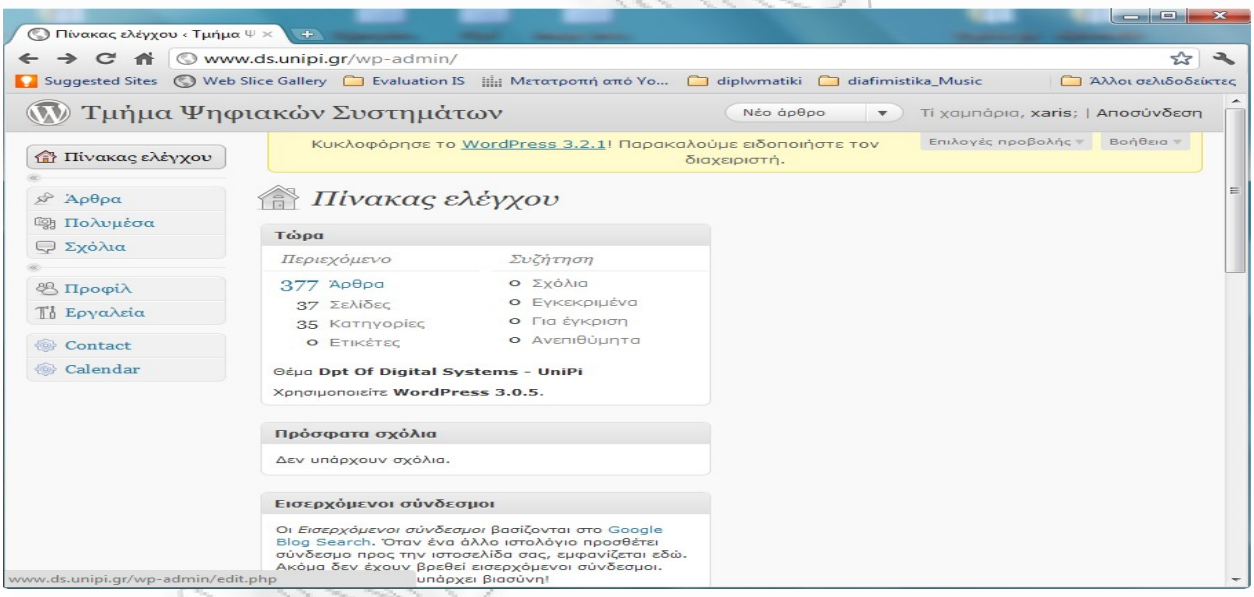
Ας αναλύσουμε για παράδειγμα, μια εφαρμογή που χρησιμοποιεί ένα διαμοιραζόμενο φάκελο για να αποθηκεύει προσωρινά pdf αρχεία για διαφορετικούς χρήστες. Ας υποθέσουμε ότι το αρχείο

documentABC.pdf πρέπει να είναι προσβάσιμο μόνο από τον χρήστη test1 με ρόλο roleA. Πρέπει να ελεγχθεί αν ο χρήστης test2 με ρόλο roleB μπορεί να προσπελάσει το αρχείο.

Αναμενόμενα αποτελέσματα:

Προσπαθήστε να εκτελέσετε διαχειριστικές διαδικασίες ή να προσπελάσετε διαχειριστικούς πόρους σαν απλός χρήστης.

Εξετάζουμε τώρα τον ιστότοπο που μελετάμε σχετικά με τον έλεγχο αυτό και τη δυνατότητα ή όχι να παρακάμψουμε το σχήμα εξουσιοδότησης (authorization). Στην εικόνα που ακολουθεί βλέπουμε τη σελίδα διαχείρισης της εφαρμογής που αφορά τον λογαριασμό ενός απλού χρήστη και όχι του admin που έχει πλήρη δικαιώματα. Όπως βλέπουμε μπορεί να κάνει συγκεκριμένες λειτουργίες που απορρέουν από το είδος των δικαιωμάτων που έχει ως απλός χρήστης και δεν μπορεί να προβεί σε διαχειριστικές λειτουργίες όπως για παράδειγμα η δημιουργία κάποιου χρήστη. Ακόμη δεν έχει τη δυνατότητα ένας απλός χρήστης να επεξεργαστεί δεδομένα που κάποιοι άλλοι απλοί χρήστες έχουν εισάγει στην εφαρμογή. Γενικότερα το authorization schema που χρησιμοποιείται στην εφαρμογή διατηρεί τους ρόλους και τα δικαιώματα αυτών και δεν υπάρχει η δυνατότητα παράκαμψής του.



Εικόνα : **Η διαχειριστική σελίδα όπως αυτή φαίνεται μέσω ενός λογαριασμού απλού χρήστη.**

3.5.3 Testing for privilege escalation

Σε αυτό τον τομέα περιγράφεται το ζήτημα της κλιμάκωσης δικαιωμάτων από ένα επίπεδο σε κάποιο υψηλότερο. Κατά την διάρκεια αυτής της φάσης. Ο ελεγκτής πρέπει να επιβεβαιώσει ότι δεν είναι δυνατό για ένα χρήστη, να τροποποιήσει τα δικαιώματά του ή τους ρόλους του στην εφαρμογή με τρόπο τέτοιο που να γίνουν δυνατές επιθέσεις τύπου πολλαπλασιασμού δικαιωμάτων.

Η κλιμάκωση δικαιωμάτων συμβαίνει όταν ένας χρήστης αποκτά πρόσβαση σε περισσότερους πόρους ή διαδικασίες από ότι κανονικά επιτρέπεται και τέτοιες αλλαγές θα έπρεπε να αποτρέπονται από την εφαρμογή. Αυτό συμβαίνει συνήθως εξαιτίας κάποιας ατέλειας στην εφαρμογή. Αποτέλεσμα είναι η εφαρμογή να εκτελεί διαδικασίες με περισσότερα δικαιώματα από ότι ήταν η πρόθεση του προγραμματιστή ή του διαχειριστή. Ο βαθμός κλιμάκωσης των δικαιωμάτων εξαρτάται από το ποια δικαιώματα έχει ο επιτιθέμενος και ποια καταφέρνει να αποκτήσει εκμεταλλευόμενος κάποια αδυναμία του συστήματος. Για παράδειγμα ένα προγραμματιστικό λάθος που επιτρέπει στο χρήστη να αποκτήσει περισσότερα δικαιώματα μετά από επιτυχή εξουσιοδότηση, μειώνει το βαθμός κλιμάκωσης γιατί ο χρήστης ήδη έχει κάποια δικαιώματα. Παρόμοια ένας απομακρυσμένος επιτιθέμενος, που αποκτά διαχειριστικά δικαιώματα, χωρίς εξουσιοδότηση παρουσιάζει μεγαλύτερο βαθμός κλιμάκωσης δικαιωμάτων. Συνήθως αναφερόμαστε σε κάθετη κλιμάκωση όταν είναι δυνατή η πρόσβαση σε ποιο διακεκριμένα δικαιώματα (δικαίωμα διαχείρισης της εφαρμογής) και οριζόντια κλιμάκωση όταν αποκτούνται δικαιώματα πρόσβαση σε πόρους που έχουν ανατεθεί σε λογαριασμούς παρόμοιου επιπέδου. (πχ πρόσβαση σε online banking σαν κάποιος άλλος χρήστης).

BLACK BOX TESTING

Έλεγχος για τροποποίηση ρόλου/δικαιωμάτων

Σε οποιοδήποτε σημείο μιας εφαρμογής όπου ο χρήστης μπορεί να δημιουργήσει δεδομένα σε μια βάση (πχ. Πραγματοποίηση μια πληρωμής, προσθήκη επαφής, αποστολή μηνύματος), να λάβει δεδομένα (κατάσταση λογαριασμού, λεπτομέρειες παραγγελίας) ή να διαγράψει δεδομένα (διαγραφή χρηστών, μηνυμάτων), είναι αναγκαίο να καταγράφεται αυτή η λειτουργία. Ο ελεγκτής πρέπει να δοκιμάσει να προσπελάσει τέτοιες διαδικασίες σαν άλλος χρήστης με σκοπό να ελέγξει αν για παράδειγμα είναι δυνατό να προσπελάσει μια διαδικασία που δεν του επιτρέπουν τα δικαιώματά/ρόλος του (αλλά μπορεί να επιτρέπονται σε άλλους χρήστες).

Για παράδειγμα το επόμενο http post επιτρέπει στον χρήστη που ανήκει στο group grp001 να προσπελάσει την παραγγελία #0001:

POST /user/viewOrder.jsp HTTP/1.1
Host: www.example.com
...
gruppoID=grp001&ordineID=0001

Πρέπει να ελεγχθεί αν ο χρήστης που ανήκει στο grp001 μπορεί να τροποποιήσει την τιμή των παραμέτρων gruppoID και ordineID για να αποκτήσει πρόσβαση σε αυτά τα ιδιωτικά δεδομένα. Για παράδειγμα η απάντηση του ακόλουθου εξυπηρετητή εμφανίζει ένα κρυφό πεδίο, που επιστρέφεται στον χρήστη όταν αυτός επιτυγχάνει στην εξουσιοδότηση.

HTTP/1.1 200 OK

Server: Netscape-Enterprise/6.0

Date: Wed, 1 Apr 2006 13:51:20 GMT

Set-Cookie: USER=aW78ryrGrTWs4MnOd32Fs51yDqp; path=/; domain=www.example.com

Set-Cookie: SESSION=k+KmKeHXTgDi1J5fT7Zz; path=/; domain= www.example.com

Cache-Control: no-cache

Pragma: No-cache

Content-length: 247

Content-Type: text/html

Expires: Thu, 01 Jan 1970 00:00:00 GMT

Connection: close

<form name="autoriz" method="POST" action = "visual.jsp">

<input type="hidden" name="profilo" value="SistemiInf1">

<body onload="document.forms.autoriz.submit()">

</td>

</tr>

Τι θα γινόταν αν ο ελεγκτής τροποποιήσει την τιμή της παραμέτρου σε profile σε SistemiInf9; Είναι δυνατό να γίνει διαχειριστής;

Για παράδειγμα:

Σε ένα περιβάλλον που ο εξυπηρετητής στέλνει μήνυμα λάθους που συμπεριλαμβάνεται σε μια συγκεκριμένη παράμετρο από ένα σύνολο απαντήσεων όπως το ακόλουθο:

@0`1`3`3``0`UC`1`Status`OK`SEC`5`1`0`ResultSet`0`PVValido`-1`0`0`

Notifications`0`0`3`Command

Manager`0`0`0` StateToolsBar`0`0`0`

StateExecToolBar`0`0`0`FlagsToolBar`0

Ο εξυπηρετητής εμπιστεύεται εκ των προτέρων τον χρήστη. Πιστεύει ότι ο χρήστης θα απαντήσει με το παραπάνω τρόπο κλείνοντας την συνεδρία. Σε αυτή την κατάσταση, επιβεβαιώνουμε ότι δεν είναι δυνατή η κλιμάκωση των δικαιωμάτων με τροποποίηση των παραμέτρων. Στο συγκεκριμένο παράδειγμα, με την τροποποίηση της τιμής PVValido από -1 σε 0 (κανένα λάθος), μπορεί να είναι δυνατό να αυθεντικοποιηθούμε σαν διαχειριστές στον εξυπηρετητή.

Αναμενόμενο αποτέλεσμα:

Ο ελεγκτής πρέπει να επιβεβαίωση την εκτέλεση επιτυχημένων προσπαθειών κλιμάκωσης δικαιωμάτων.

Στην περίπτωση του ιστότοπου του Τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιώς δεν βρέθηκαν περιπτώσεις που παραβιάζεται η κλιμακούμενη ανάθεση δικαιωμάτων που ισχύει για τους χρήστες της εφαρμογής επομένως το Privilege Escalation δεν αποτελεί ευπάθεια για τον ιστότοπο αυτόν.

3.6 Denial of Service Testing

Το πιο κοινό είδος επίθεσης Άρνησης Παροχής Υπηρεσίας χρησιμοποιείται σε ένα δίκτυο για να καταστήσουν ένα εξυπηρετητή μη προσβάσιμο σε έγκυρους χρήστες. Η βασική ιδέα πίσω από μια επίθεση Άρνηση Εξυπηρέτησης είναι ένας κακόβουλος χρήστης που προκαλεί υπερβολική κίνηση σε ένα στόχο, καθιστώντας τον ανίκανο να επεξεργαστεί όλα τα αιτήματα. Όταν πολλοί κακόβουλοι χρήστες η μηχανήματα συνασπίζονται εναντίον ενός στόχου τότε έχουμε κατανεμημένη επίθεση άρνησης εξυπηρέτησης. Αυτού του τύπου οι επιθέσεις είναι εκτός πεδίου του προγραμματιστή της εφαρμογής καθώς αντιμετωπίζονται μέσω κατάλληλων λύσεων στην αρχιτεκτονική δικτύων. Υπάρχουν ωστόσο κάποιοι τύποι αδυναμιών που μπορούν να επιτρέψουν στους κακόβουλους χρήστες να καταστήσουν κάποια διαδικασία ή και ολόκληρο ιστότοπο μη διαθέσιμο. Αυτά τα προβλήματα προκαλούνται από ατέλειες της εφαρμογής στην είσοδο δεδομένων. Αυτό το κεφάλαιο θα επικεντρωθεί σε επιθέσεις εναντίον μια εφαρμογής από έναν κακόβουλο χρήστη ή μηχάνημα. Οι σχετικοί τύποι επίθεσης Άρνησης Εξυπηρέτησης είναι:

1. Δοκιμή για επιθέσεις με ειδικούς χαρακτήρες SQL
2. Κλείδωμα λογαριασμών χρηστών
3. Υπερχείλιση μνήμης (buffer overflow)
4. Ανάθεση αντικειμένων στον χρήστη
5. Είσοδος χρήστης σαν μετρητής βρόγχου
6. Εγγραφή δεδομένων σε δίσκο
7. Αποτυχία αποδέσμευσης πόρων
8. Αποθήκευση υπερβολικού όγκου σε μια συνεδρία

3.6.1 Testing for SQL Wildcard Attacks

Οι επιθέσεις αυτού του τύπου έχουν ως στόχο τους να εξαναγκάσουν την βάση δεδομένων να εκτελεί απαιτητικές σε υπολογιστική ισχύ εργασίες, με τη χρήση διάφορων χαρακτήρων. Αυτή η αδυναμία υπάρχει γενικά σε διαδικασίες αναζήτησης των web εφαρμογών. Επιτυχής εκμετάλλευση τους προκαλεί άρνηση εξυπηρέτησης. Οι επιθέσεις αυτού του τύπου μπορούν να επηρεάσουν όλες τις πλατφόρμες βάσεων δεδομένων αλλά κυρίων τον SQL SERVER καθώς σε αυτόν ο τελεστής LIKE υποστηρίζει έξτρα χαρακτήρες όπως "[]","[^]","_" και "%". Σε μια τυπική web εφαρμογή, αν αναζητήσετε τον όρο “foo”, το αντίστοιχο SQL αίτημα θα ήταν το:

*SELECT * FROM Article WHERE Content LIKE '%foo%'*

Σε μια αξιοπρεπή βάση δεδομένων με 1-100000 εγγραφές, το παραπάνω αίτημα απαιτεί λιγότερο από ένα δευτερόλεπτο για να εκτελεστεί. Το επόμενο όμως αίτημα στην ίδια βάση χρειάζεται γύρω στα 6 δευτερόλεπτα για την ολοκλήρωσή του σε μόλις 2600 εγγραφές:

*SELECT TOP 10 * FROM Article WHERE Content LIKE*

'%_[^!_%/%a?F%_D)_(F%)_%([)({}%){()}£$&N%_)$£()$*R"_)][%](%[x])%a][$*"£$-9]_%'*

Έτσι ένα ο ελεγκτής ήθελε να απασχολήσει τον επεξεργαστεί για 6 δευτερόλεπτα θα μπορούσε να εισάγει το εξής στο πεδίο αναζήτησης:

_[^!_%/%a?F%_D)_(F%)_%([)({}%){()}£$&N%_)$£()$*R"_)][%](%[x])%a][$*"£$-9]_*

BLACK BOX TESTING

Δοκιμές για επιθέσεις με ειδικούς χαρακτήρες SQL

Δημιουργήστε ένα αίτημα που δεν θα επιστρέψει κάποιο αποτέλεσμα και περιέχει πολλούς ειδικούς χαρακτήρες. Μπορείτε να χρησιμοποιήσετε ένα αίτημα από τα παρακάτω. Στείλτε τα δεδομένα μέσα από την αναζήτηση μια εφαρμογής και αν η εφαρμογή καθυστερεί παραπάνω από το φυσιολογικό να απαντήσει, τότε είναι ευάλωτη.

Παράδειγμα αιτήματος

- *'%_[^!_%/%a?F%_D)_(F%)_%([)({}%){()}£$&N%_)$*£()$*R"_)][%](%[x])%a][$*"£$-9]_%'*
- *'%64_[^!_%65/%aa?F%64_D)_(F%64)_%36([)({}%33){()}£$&N%55_)$*£()$*R"_)][%55](%66[x])%ba*
- *][$*"£$-9]_%54' bypasses modsecurity*
- *_[r/a)_ _(r/b)_ _(r-d)_*
- *%n[^n]y[^j]l[^k]d[^l]h[^z]t[^k]b[^q]t[^q][^n]!%*
- *%_[aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa[! -z]@$!_%*

  ...

Αναμενόμενο αποτέλεσμα

Αν η εφαρμογή είναι ευάλωτη, τότε η απάντηση θα καθυστερήσει περισσότερο από το σύνηθες.

Πώς να δημιουργήσετε κατάλληλα αιτήματα για δοκιμές:

- Τα αιτήματα πρέπει να επιστρέφουν όσο το δυνατό λιγότερα αποτελέσματα ή καθόλου. Σε αυτή την περίπτωση είμαστε σίγουροι ότι αναγκάζουμε την βάση να ψάξει όλες τις εγγραφές.
- Κατά την διάρκεια των συνδυασμών με το λογικό τελεστή OR θα πρέπει να κάθε μέρος της λογικής πράξης να είναι διαφορετικό, ώστε να μην βελτιστοποιηθεί από την βάση.
- Στον SQL Server κάθε χαρακτήρας μετά από αγκύλη «[» προκαλεί μεγάλο χρόνο εκτέλεση. Αυτό μπορεί να χρησιμοποιηθεί για αύξηση το συνολικό χρόνο.
  - *LIKE '%_[a[! -z]@$!_% - 1050 ms.*
  - *LIKE '%_[aaaaaaaaa[! -z]@$!_%' - 1600 ms.*

- *LIKE '%_[aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa[! -z]@$!_%' - 3700 ms.*

- Μακρύτερα αιτήματα συνήθως απαιτούν μεγαλύτερο χρόνο εκτέλεσης.
- Ξεκινώντας και τελειώνοντας ένα αίτημα με τους χαρακτήρες % προκαλούμε μεγαλύτερους χρόνους εκτέλεσης.
- Κάποιες υλοποιήσεις αναζητήσεις μπορεί να αποθηκεύουν αποτελέσματα αναζήτησης. Κατά την διάρκεια των δοκιμών πρέπει κάθε αίτημα που εκτελείται να είναι ελάχιστα διαφορετικό από το προηγούμενο, ώστε να μην επανέρχονται τα παλιότερα αποτελέσματα.
- Δοκιμάστε διαφορετικούς συνδυασμούς για να βρείτε τους πλέον απαιτητικούς σε υπολογιστικούς πόρους.

3.6.2 Locking Customer Accounts

Σε αυτή τη δοκιμή ελέγχουμε εάν ο επιτιθέμενος μπορεί να κλειδώσει λογαριασμού έγκυρων χρηστών με επαναλαμβανόμενες προσπάθειες εισόδου με λανθασμένο κωδικό. Στο σενάριο Άρνησης Εξυπηρέτησης, το σύστημα αυθεντικοποίησης μπορεί να αποτελεί στόχο της επίθεσης. Μια κοινή τεχνική άμυνας απέναντι στην αποκάλυψη κωδικών με συνεχείς δοκιμές, είναι το κλείδωμα του λογαριασμού, μετά από τρεις με πέντε προσπάθειες εισόδου. Αυτό σημαίνει πως ο έγκυρος χρήστης ακόμα και αν διαθέτει τον σωστό κωδικό δε θα μπορεί να μπει στο σύστημα, έως ότου ο λογαριασμός του ξεκλειδωθεί. Αυτός ο μηχανισμός άμυνας μπορεί να έχει αποτελέσματα Άρνησης Εξυπηρέτησης εάν δεν υπάρχει τρόπος να προβλεφτούν οι έγκυροι λογαριασμοί. Υπάρχει εδώ μια σχέση ισορροπίας μεταξύ λειτουργικότητας και ασφάλειας η οποία πρέπει να διατηρείται ανάλογα και με το περιβάλλον εκτέλεσης της εφαρμογής. Υπάρχουν θετικά και αρνητικά στο κλείδωμα λογαριασμών, στην δυνατότητα να επιλέγουν οι χρήστες τα ονόματα λογαριασμών με τη χρήση συστημάτων όπως το CAPTCHA ή παρόμοια. Κάθε οργανισμός πρέπει να αξιολογεί τους κινδύνους και τα πλεονεκτήματα από την υιοθέτηση μια πολιτικής ασφάλειας από αυτές που περιγράφονται εδώ.

BLACK BOX TESTING

Η πρώτη δοκιμή που πρέπει να διενεργηθεί είναι αν ένας λογαριασμός, όντως κλειδώνει μετά από συγκεκριμένο αριθμό αποτυχημένων προσπαθειών εισόδου. Αν γνωρίζεται ήδη ένα έγκυρο όνομα λογαριασμού τότε χρησιμοποιήστε το, για να επιβεβαιώσετε ότι το σύστημα όντως κλειδώνει το λογαριασμό μετά από τουλάχιστον 15 αποτυχημένες προσπάθειες. Αν ο λογαριασμός δεν κλειδώσει στις 15 αποτυχημένες προσπάθειες είναι απίθανο, να κλειδώσει. Πολλές φορές οι εφαρμογές προειδοποιούν τους χρήστες ότι εξαντλούν τις προσπάθειες τους για επιτυχή σύνδεση και ότι θα κλειδωθεί ο λογαριασμός τους. Αυτό μπορεί να βοηθήσει τον ελεγκτή καθώς το κλείδωμα λογαριασμών είναι αρνητικό από διαχειριστικής άποψης. Αν δεν υπάρχει όνομα λογαριασμού διαθέσιμο κατά την διάρκεια της δοκιμής, ο ελεγκτής μπορεί να χρησιμοποιήσει κάποια από τις παρακάτω μεθόδους για να ανακαλύψει ένα έγκυρο όνομα χρήστη. Για τον προσδιορισμό έγκυρων ονομάτων λογαριασμών, ο

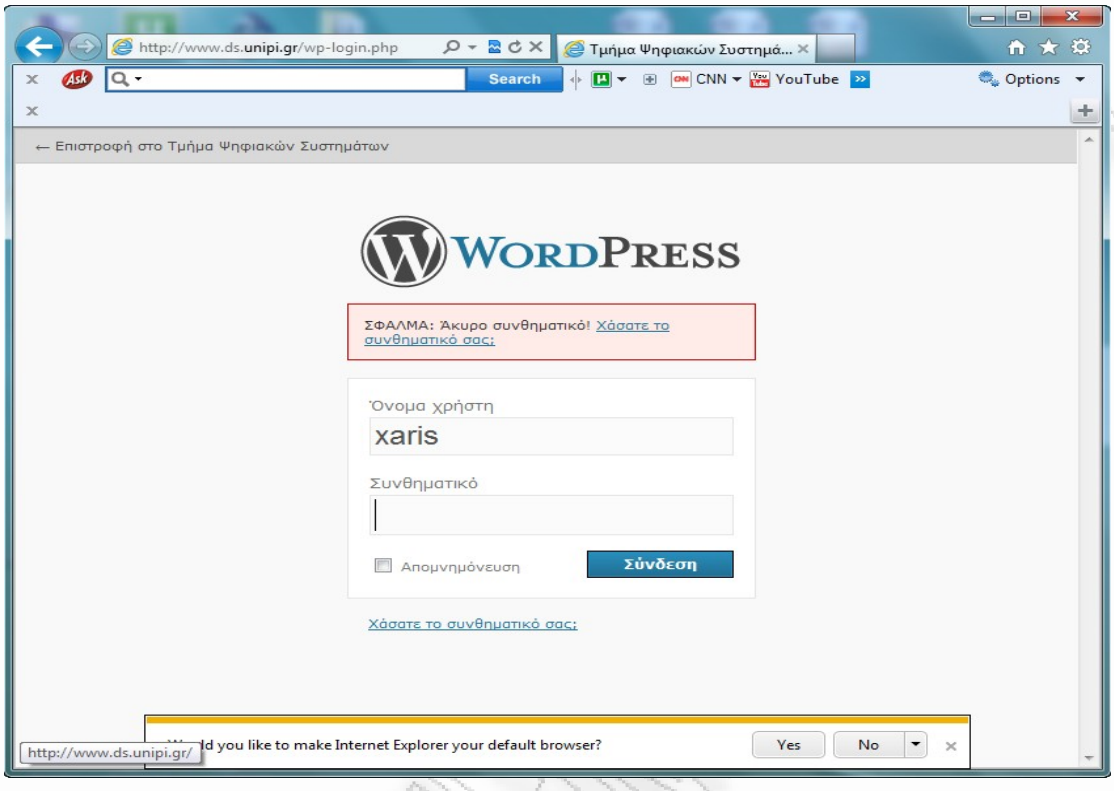
ελεγκτής πρέπει να ερευνήσει σε περιοχές όπου η εφαρμογή διαχωρίζει τα έγκυρα από τα άκυρα στοιχεία εισόδου. Τέτοιες περιοχές είναι :

1. Η σελίδα εισόδου. Χρησιμοποιώντας ένα κοινό όνομα με ένα άκυρο κωδικό ελέγχουμε το μήνυμα λάθους . Στέλνουμε ένα ακόμη αίτημα εισόδου με όνομα που είναι καταφανώς μη υπαρκτό και ελέγχουμε το νέο μήνυμα λάθους ερευνώντας για διαφορές με το προηγούμενο. Αν όντως υπάρχουν διαφορές, μπορούν να χρησιμοποιηθούν για τον εντοπισμό έγκυρων ονομάτων λογαριασμών. Μερικές φορές η διαφορά μεταξύ των απαντήσεων είναι τόσο μικρή που δεν είναι αμέσως εμφανείς. Για παράδειγμα το μήνυμα μπορεί να είναι ακριβώς το ίδιο αλλά ο χρόνος απάντησης να διαφέρει. Ένας άλλος τρόπος να ελέγξουμε για διαφορές είναι με την σύγκριση των κωδικών κατακερματισμού κάθε μηνύματος λάθους. Εκτός και αν ο εξυπηρετητής μεταβάλλει κάθε απάντηση, ο τρόπος με την σύγκριση των κωδικών κατακερματισμού είναι ο βέλτιστος για τον εντοπισμό διαφορών.
2. Στην σελίδα δημιουργίας νέου λογαριασμού, αν η εφαρμογή επιτρέπει στους χρήστες να δημιουργούν νέους λογαριασμούς, με τη δυνατότητα να επιλέγουν όνομα χρήστη, μπορεί να είναι δυνατό να ανακαλυφθούν έγκυρα ονόματα λογαριασμών. Όταν προσπαθείτε να δημιουργήσετε νέο λογαριασμό με όνομα το οποίο χρησιμοποιείται, τότε το σύστημα επιστρέφει μήνυμα λάθους, ώστε να επιλέξετε νέο όνομα.
3. Στην σελίδα επαναφοράς κωδικού. Αν η εφαρμογή διαθέτει λειτουργία, επαναφοράς κωδικού για τους χρήστες, τότε είναι δυνατό η διαδικασία αυτή να επιστρέφει διαφορετικά μηνύματα όταν γίνεται προσπάθεια επαναφοράς κωδικού υπαρκτού ονόματος από ότι όταν γίνεται για ανύπαρκτο όνομα.

Όταν ο επιτιθέμενος έχει την ικανότητα να ανακαλύψει έγκυρα ονόματα χρηστών, ή τα ονόματα έχουν ένα προβλέψιμο μορφότυπο τότε είναι αρκετά εύκολη η δημιουργία μιας αυτοματοποιημένης διαδικασίας που θα στέλνει 3 με 5 άκυρους κωδικούς για κάθε λογαριασμό ώστε να τους κλειδώσει. Αν ο επιτιθέμενος ανακαλύψει πολλά ονόματα χρηστών είναι πιθανό να κλειδώσει μεγάλο κομμάτι της βάσης.

Στη περίπτωση του δικού μας εξεταζόμενου ιστότοπου προσπαθήσαμε να δούμε εάν εφαρμόζεται το κλείδωμα των χρηστών στη διαχειριστική σελίδα της εφαρμογής. Γνωρίζαμε την ύπαρξη ενός χρήστη, μιας και αποτελεί το προσωπικό μου όνομα χρήστη για αυτή τη σελίδα, και προσπαθήσαμε πάνω από 30 φορές να εισέλθουμε με λανθασμένο κωδικό πρόσβασης παρέχουμε το σωστό κωδικό και αποκτούμε πρόσβαση στην εφαρμογή. Το ίδιο αποτέλεσμα είχαμε τόσο στη σελίδα δημιουργίας νέου λογαριασμού όσο και στη σελίδα επαναφοράς του κωδικού. Με αυτόν τον έλεγχο διαπιστώνουμε ότι η εφαρμογή δεν εφαρμόζει μηχανισμό κλειδώματος των λογαριασμών των χρηστών παρά το μεγάλο αριθμό αποτυχημένων προσπαθειών σύνδεσής τους. Αυτό έχει ως αποτέλεσμα να αποφεύγεται η άρνηση παροχής των υπηρεσιών σε νόμιμους χρήστες, των οποίων οι λογαριασμοί ενδέχεται να γίνουν γνωστοί σε κακόβουλους χρήστες. Στην εικόνα που ακολουθεί βλέπουμε το αποτέλεσμα που λαμβάναμε κατά τις

επανειλημμένες προσπάθειές μας να αποκτήσουμε πρόσβαση με έγκυρο όνομα χρήστη και μη-έγκυρο κωδικό πρόσβασης προκειμένου να πετύχουμε το κλείδωμα του λογαριασμού:



Εικόνα : **Το αποτέλεσμα της προσπάθειας κλειδώματος ενός έγκυρου λογαριασμού χρήστη.**

3.6.3 Buffer Overflows

Σε αυτό τον έλεγχο δοκιμάζεται το αν είναι δυνατό να προκληθεί μια κατάσταση άρνησης εξυπηρέτησης με την υπερχείλιση μιας ή περισσοτέρων δομών δεδομένων στην εφαρμογή στόχο. Σε κάθε γλώσσα προγραμματισμού ο προγραμματιστής έχει την ευθύνη για την διαχείριση της μνήμης, ειδικά στην C και την C++ υπάρχει η πιθανότητα υπερχείλισης της μνήμης. Ενώ ο πιο σημαντικός κίνδυνος που σχετίζεται με την υπερχείλιση μνήμης είναι η δυνατότητα εκτέλεσης κακόβουλου κώδικα στον εξυπηρετητή, το πρώτο πρόβλημα είναι η άρνηση εξυπηρέτησης που μπορεί να συμβεί εάν η εφαρμογή τερματιστεί. Το παρακάτω είναι ένα παράδειγμα προβληματικό κώδικα σε C:

```
void overflow (char *str) { char buffer[10]; strcpy(buffer, str); // Dangerous!
}
int main () { char *str = "This is a string that is larger than the buffer of 10"; overflow(str);
}
```

Το παραπάνω τμήμα εάν εκτελεστεί, θα προκαλέσει λάθος και θα τερματιστεί. Ο λόγος είναι ότι η εντολή strcpy προσπαθεί να αντιγράψει 53 χαρακτήρες σε ένα πίνακα δέκα θέσεων, αντικαθιστώντας έτσι θέσεις

μνήμης με άλλα δεδομένα. Παρότι η περίπτωση αυτή είναι υπεραπλουστευμένη, η πραγματικότητα σε μια web εφαρμογή είναι ότι μπορεί να υπάρχει κάποιο σημείο εισόδου δεδομένων που δεν ελέγχεται σωστά και μπορεί να προκαλέσει κάτι παρόμοιο.

BLACK BOX TESTING

Το κεφάλαιο σχετικά με τις δοκιμές Υπερχείλισης της Μνήμης δείχνει πως μπορεί κανείς να υποβάλλει διάφορα μήκη δεδομένων και που σε μια εφαρμογή. Όπως και στις δοκιμές Άρνησης Εξυπηρέτησης αν λάβετε απάντηση που σας κάνει να πιστεύεται ότι δημιουργήθηκε η υπερπλήρωση τότε δοκιμάστε άλλο ένα αίτημα στον εξυπηρετητή για να δείτε εάν αποκρίνεται.

3.6.4 User Specified Object Allocation

Σε αυτούς τους ελέγχους εξετάζεται εάν είναι δυνατό να εξαντληθούν οι πόροι του συστήματος με την ανάθεση πολλών αντικειμένων σε ένα χρήστη. Αν οι χρήστες μπορούν να εισάγουν άμεσα ή έμμεσα μια τιμή που θα καθορίζει πόσα αντικείμενα θα δημιουργήσει ο εξυπηρετητής και αν ο δεύτερος δεν έχει ένα άνω όριο για τον αριθμό αυτό τότε είναι δυνατό να καταναλωθεί όλη η διαθέσιμη μνήμη. Ο εξυπηρετητής μπορεί να ξεκινήσει να αναθέτει τον αναγκαίο αριθμό αντικειμένων αλλά αν αυτός είναι πολύ μεγάλος, τότε μπορεί να προκαλέσει σημαντικά προβλήματα στο εξυπηρετητή με το γέμισμα της μνήμης και τελικά την αρνητική επιρροή στην απόδοση. Το επόμενο είναι ένα απλό παράδειγμα ελαττωματικού κώδικα σε Java:

String TotalObjects = request.getParameter("numberofobjects"); int NumOfObjects = Integer.parseInt(TotalObjects); ComplexObject[] anArray = new ComplexObject[NumOfObjects]; // wrong!

BLACK BOX TESTING

Ο ελεγκτής ψάχνει για σημεία που είναι δυνατή η εισαγωγή αριθμών με τρόπο παρόμοιο του παραπάνω παραδείγματος. Προσπαθεί να θέσει μια πολύ υψηλή τιμή και να δει εάν ο εξυπηρετητής εξακολουθεί να αποκρίνεται. Μπορεί να χρειαστεί να περάσει λίγος χρόνος, μέχρι η απόδοση να αρχίσει να μειώνεται, καθώς ο εξυπηρετητής συνεχίζει την ανάθεση αντικειμένων. Στο παραπάνω παράδειγμα, με την αποστολή ενός μεγάλου αριθμού στη μεταβλητή "numberofobjects", ο εξυπηρετητής θα προσπαθήσει να δημιουργήσει τόσο πολύπλοκα αντικείμενα. Ενώ οι περισσότερες εφαρμογές δεν επιτρέπουν στους χρήστες να θέτουν τέτοιες τιμές, περιπτώσεις αυτού του τύπου αδυναμιών, μπορούν να βρεθούν σε ένα κρυφό πεδίο, η μια τιμή που υπολογίζεται μέσω JavaScript όταν μια φόρμα υποβάλλεται. Αν η εφαρμογή δεν διαθέτει κανένα αριθμητικό πεδίο που μπορεί να χρησιμοποιηθεί με τέτοιο τρόπο, το ίδιο αποτέλεσμα μπορεί να επιτευχθεί ανάθεση αντικειμένων σειριακά. Ένα ενδιαφέρον παράδειγμα υπάρχει στα ηλεκτρονικά καταστήματα. Αν η εφαρμογή δεν θέτει ένα άνω όριο στον αριθμό των

αντικειμένων που μπορούν να μπουν στο καλάθι, τότε μπορεί να γραφτεί μια αυτοματοποιημένη διαδικασία που να προσθέτει αντικείμενα μέχρι να εξαντληθεί η μνήμη.

3.6.5 User Input as a Loop Counter

Σε αυτό τον έλεγχο, εξετάζεται εάν είναι δυνατό να επιβληθεί στην εφαρμογή να εκτελεί επαναληπτικά ένα τμήμα κώδικα με υψηλές υπολογιστικές απαιτήσεις, ώστε να μειωθεί η συνολική απόδοση. Όπως και με την ανάθεση αντικειμένων, εάν ο χρήστης μπορεί να θέσει μια τιμή που θα χρησιμοποιηθεί ως μετρητής για μια επαναληπτική διαδικασία, τότε μπορεί να προκληθεί πρόβλημα. Το παρακάτω αποτελεί παράδειγμα ελαττωματικού κώδικα σε java:

public class MyServlet extends ActionServlet { public void doPost(HttpServletRequest request, HttpServletResponse response)

throws ServletException, IOException {

String [] values = request.getParameterValues("CheckboxField"); // Process the data without length check for reasonable range – wrong!

for (int i=0; i<values.length; i++) { // lots of logic to process the request

} .. .

} . ..

}

.. .

Όπως βλέπουμε σε αυτό το απλό παράδειγμα, ο χρήστης έχει τον έλεγχο στον μετρητή επαναλήψεων. Αν ο κώδικας μέσα στο βρόγχο επανάληψης είναι απαιτητικός σε υπολογιστικούς πόρους και ο επιτιθέμενος υποχρεώσει τον εξυπηρετητή να τον εκτελέσει πολλαπλές φορές, τότε μπορεί να μειωθεί σημαντικά η απόδοση του εξυπηρετητή συνολικά, προκαλώντας κατάσταση άρνησης εξυπηρέτησης.

BLACK BOX TESTING

Αν αποσταλεί ένα αίτημα στον εξυπηρετητή, που ορίζει για παράδειγμα τον αριθμό των επαναλήψεων και ο εξυπηρετητής δεν έχει άνω όριο στον αριθμό επαναλήψεων τότε μπορεί να προκληθεί εκτέλεση της επανάληψης για μεγάλα χρονικά διαστήματα. Ένα άλλο πρόβλημα είναι όταν ο κακόβουλος χρήστης στέλνει μεγάλο αριθμό από τιμές κατευθείαν στον εξυπηρετητή. Ενώ η εφαρμογή δεν μπορεί να αποτρέψει την επεξεργασία άμεσα, δε θα πρέπει να επιτρέπεται στον εξυπηρετητή να εκτελεί επαναλήψεις χωρίς κάποιο όριο σχετικό μετά δεδομένα αυτά. Για παράδειγμα πολλαπλές τιμές μπορούν να υποβληθούν από τον ελεγκτή με το ίδιο όνομα αλλά διαφορετικές τιμές. Έτσι κοιτάζοντας την τιμή ενός συγκεκριμένου ζεύγους ονόματος/τιμής θα επιστραφεί ολόκληρος ο πίνακας τιμών που υποβλήθηκε. Αν υπάρχει υποψία για τέτοιο λάθος, ο ελεγκτής πρέπει να υποβάλλει μεγάλο αριθμό από επαναλαμβανόμενα ζεύγη ονομάτων/τιμών στο αίτημα με ένα script. Αν προκληθεί ανιχνεύσιμη διαφορά

στους χρόνους αποκρίσεις τότε μπορεί να υπάρχει τέτοιο πρόβλημα. Οι κρυφές τιμές περνούν στην εφαρμογή ενώ μπορεί και να παίζουν ρόλο στην εκτέλεση κώδικα.

3.6.6 Writing User Provided Data to Disk

Σε αυτό τον έλεγχο θα εξεταστεί η δυνατότητα πρόκληση άρνησης εξυπηρέτησης με το γέμισμα των δίσκων στόχων με δεδομένα log. Ο στόχος αυτού του τύπου της επίθεσης είναι να προκαλέσει την εφαρμογή να καταγράφει δεδομένα τεραστίου μεγέθους με στόχο να γεμίσουν οι τοπικοί δίσκοι. Αυτή η επίθεση μπορεί να γίνει με δύο τρόπους:

1. Ο ελεγκτής υποβάλλει μια πολύ μεγάλη τιμή στον εξυπηρετητή και η εφαρμογή καταγράφει την τιμή αυτή χωρίς έλεγχο και έτσι προκαλείται το εν λόγω πρόβλημα.
2. Η εφαρμογή μπορεί να διαθέτει κάποιο μηχανισμό ελέγχου για το μήκος των δεδομένων εισόδου αλλά και πάλι να καταγράφει τα δεδομένα για λόγους αρχείου.

Αν η εφαρμογή δεν έχει κάποιο άνω όριο στο επιτρεπτό μέγεθος για κάθε log εγγραφή και καταναλωθεί όλος ο διαθέσιμος χώρος, τότε είναι ευάλωτη σε αυτή την επίθεση. Αυτό ισχύει ειδικά όταν δεν υπάρχει ξεχωριστό αρχείο για τα logs και τα αρχεία μεγαλώνουν έως ότου όλες οι άλλες διεργασίες γίνουν αδύνατες. Ωστόσο μπορεί να είναι δύσκολο να επιβεβαιωθεί η επιτυχία αυτής της επίθεσης, εκτός και αν ο ελεγκτής έχει πρόσβαση στα αρχεία log.

BLACK BOX TESTING

Σε αυτό τον έλεγχο είναι πολύ δύσκολο να εκτελεστεί ένα σενάριο black box χωρίς τύχη και υπομονή. Ο καθορισμός μια τιμής που υποβάλλεται χωρίς έλεγχο μεγέθους ή με πολύ μεγάλο όριο είναι το πρώτο βήμα. Περιοχές κειμένου είναι συνήθως τέτοια πεδία αλλά μπορεί να μην καταγράφονται . Η αυτοματοποίηση της διαδικασίας με ένα script και η εκτέλεση του για επαρκή χρόνο μπορεί να προκαλέσει τελικά το πρόβλημα με τον εξυπηρετητή να αναφέρει λάθη όταν προσπαθεί να γράψει στο σύστημα αρχείων.

3.6.7 Failure to Release Recourses

Με αυτό τον έλεγχο εξετάζεται αν η εφαρμογή απελευθερώνει τους πόρους (αρχεία και μνήμη) μετά την χρήση τους. Αν προκύψει ένα λάθος στην εφαρμογή που εμποδίζει την απελευθέρωση ενός πόρου σε χρήση, τότε αυτός μπορεί να παραμείνει μη διαθέσιμος και μελλοντικά. Μερικά παραδείγματα:

- Μια εφαρμογή κλειδώνει ένα αρχείο για εγγραφή και εξαιτίας κάποιας εξαίρεσης δεν το απελευθερώνει
- Σε γλώσσες που η διαχείριση μνήμης γίνεται από το χρήστη (C , C++). Στην περίπτωση που κάποιο λάθος προκαλέσει ατέρμονα βρόγχο, η μνήμη που ανατέθηκε στην διεργασία μπορεί να βρεθεί σε κατάσταση τέτοια που η αποκομιδή απορριμμάτων να μην μπορεί να καθορίσει αν χρησιμοποιείται ή όχι.

- Χρήση αντικειμένων συνδέσεων με βάση δεδομένων που δεν απελευθερώνονται. Ένας αριθμός από αιτήματα που προκαλεί αυτές τις συνδέσεις μπορεί να καταναλώσει όλες τις διαθέσιμες συνδέσεις, καθώς ο κώδικας δεν θα απελευθερώνει τα αντικείμενα που έχουν ήδη χρησιμοποιηθεί.

  Το παρακάτω παράδειγμα επιδεικνύει την αδυναμία αυτή σε Java.

  public class AccountDAO {

  public void createAccount(AccountInfo acct) throws AcctCreationException {

  ..... try {

  Connection conn = DAOFactory.getConnection(); CallableStatement calStmt = conn.prepareCall(...);

  ..... calStmt.executeUpdate();

  calStmt.close(); conn.close();

  } catch (java.sql.SQLException e) { throw AcctCreationException (...);

  } } }

BLACK BOX TESTING

Γενικά είναι ιδιαίτερα δύσκολη η παρακολούθηση αυτού του είδους των διαρροών σε με την τεχνική black box. Αν βρείτε ένα αίτημα που πιστεύεται ότι εκτελεί κάποια διεργασία στη βάση η οποία θα προκαλέσει στον server να επιστρέψει κάποιο λάθος το οποίο θα είναι μια εξαίρεση που δε θα μπορεί να διαχειριστεί, τότε μπορείτε με μια αυτοματοποιημένη διαδικασία να υποβάλετε εκατοντάδες τέτοια αιτήματα πολύ γρήγορα. Παρακολουθήστε εάν υπάρχει μείωση στην απόδοση του εξυπηρετητή ή νέα μηνύματα λάθους από την εφαρμογή κατά τη διάρκεια κανονικής και νόμιμης χρήσης της.

3.6.8 Storing

Σε αυτό τον έλεγχο, εξετάζεται εάν είναι δυνατό να αποθηκευτεί μεγάλος όγκος δεδομένων, στην συνεδρία του χρήστη, ώστε να εξαντληθούν οι πόροι μνήμης του εξυπηρετητή. Πρέπει να μην επιτρέπεται να αποθηκεύονται πολλά δεδομένα στην συνεδρία του χρήστη. Πολλά δεδομένα , όπως για παράδειγμα αποτελέσματα αιτημάτων στην βάση δεδομένων, μπορούν να προκαλέσουν ζητήματα άρνησης εξυπηρέτησης. Το πρόβλημα μπορεί να δημιουργηθεί και πριν το login και έτσι ο επιτιθέμενος μπορεί να το προκαλέσει και χωρίς να διαθέτει έγκυρο λογαριασμό.

BLACK BOX TESTING

Πρόκειται και πάλι για μια δύσκολη περίπτωση ελέγχου μέσω black box. Τα πιθανότερα σημεία είναι αυτά που μεγάλος αριθμός εγγραφών επιστρέφεται με βάση δεδομένα εισόδου του χρήστη. Επίσης διαδικασίες σχετικές με την προβολή μεγάλων σετ δεδομένων σε συγκεκριμένο χρόνο. Ο προγραμματιστής πρέπει να έχει επιλέξει να αποθηκεύει τα αποτελέσματα στην συνεδρία. Αν κάτι τέτοιο

συμβαίνει τότε με μια αυτοματοποιημένη διαδικασία δημιουργούμε πολλές συνεδρίες και εκτελούμε το εν λόγο αίτημα. Μετά από λίγο χρόνο θα παρατηρηθεί μειωμένη απόκριση εάν η επίθεση είναι επιτυχής.

Κεφάλαιο 4ο [Αποτελέσματα και Συμπεράσματα από τους ελέγχους]

Εισαγωγή

Κατά την εξέταση του επιπέδου ασφάλειας της web εφαρμογής μας υπήρξαν δύο βασικοί παράγοντες που έπαιξαν τον κυριότερο ρόλο στον τρόπο με τον οποίο διενεργήσαμε τους ελέγχους μας και αποτελούν τους δύο κύριους άξονες με βάση πάνω στους οποίους βασιστήκαμε προκειμένου να ολοκληρώσουμε το σύνολο των ελέγχων που περιελάμβανε η παρούσα διπλωματική μεταπτυχιακή εργασία. Οι παράγοντες αυτοί παρουσιάζονται παρακάτω:

1. **Επιλογή της κατάλληλης μεθόδου ελέγχου ασφάλειας εφαρμογών ιστού**

Μια από τις σημαντικότερες διαδικασίες κατά τις φάσεις της ανάπτυξης και εγκατάστασης είναι ο έλεγχος ασφάλειας των εφαρμογών ιστού. Υπάρχουν πολλοί μέθοδοι ελέγχου ασφάλειας των εφαρμογών ιστού. Ένας από αυτούς και ίσως ο πιο αποτελεσματικός είναι η επιθεώρηση ασφάλειας του κώδικα της εφαρμογής. Αυτή αποτελεί ένα λεπτομερή και μεθοδικό έλεγχο, ο οποίος, όμως, απαιτεί μεγάλη χειρωνακτική προσπάθεια, σημαντικό υπόβαθρο γνώσεων και άφθονο χρόνο στη διάθεση του προσωπικού ασφάλειας, που θα πραγματοποιήσει τον έλεγχο. Για τους λόγους αυτούς, που αποτελούν και μειονεκτήματα της μεθόδου αυτής, συνήθως αναλαμβάνεται από εξειδικευμένο οργανισμό. Το γεγονός αυτό, όμως, αυξάνει το κόστος της, που ίσως είναι και το πιο σημαντικό μειονέκτημα, το οποίο και καθιστά την επιλογή της μεθόδου αυτής απαγορευτική. Κατά συνέπεια οι περισσότεροι οργανισμοί-επιχειρήσεις καταφεύγουν στη λύση του ελέγχου ασφάλειας με τη μέθοδο της δοκιμής διείσδυσης, κάτι το οποίο μπορεί να πραγματοποιηθεί από προσωπικό του οργανισμού-επιχείρησης. Διαπιστώνουμε ότι η μέθοδος της δοκιμής διείσδυσης με την black box τεχνική είναι η καλύτερη μέθοδος για τον έλεγχο των εφαρμογών ιστού. Η μέθοδος αυτή αφορά τον έλεγχο ασφάλειας της εφαρμογής μέσω της εξωτερικής της διεπαφής, δηλαδή αυτή που χρησιμοποιείται από τους χρήστες και χωρίς γνώση της εσωτερικής δομής της εφαρμογής. Για να πραγματοποιηθεί ένας αποτελεσματικός έλεγχος μιας εφαρμογής ιστού με τη μέθοδο αυτή, θα πρέπει ο ελεγκτής να «παίξει» πολύ με την εφαρμογή, να προσδιορίσει τι αποτελεί «αντικανονική εισαγωγή» δεδομένων για το λογισμικό της εφαρμογής παρατηρώντας καλά κάθε λειτουργία της και να προσπαθήσει να εισάγει τα αντικανονικά δεδομένα. Αυτό βέβαια είναι πολύ

χρονοβόρα διαδικασία και απαιτεί βαθιά γνώση σε θέματα ασφάλειας, ώστε να είναι εφικτός ο προσδιορισμός των ευπαθειών, που πιθανόν να υπάρξουν. Το όφελος βέβαια είναι, ότι κατά αυτό τον τρόπο εξετάζεται και η λειτουργικότητα της εφαρμογής ιστού, αφού ο ελεγκτής συμπεριφέρεται σαν ένας απλός εξωτερικός χρήστης της.

2. **Επιλογή του κατάλληλου εργαλείου ελέγχου**

Επειδή, όπως είπαμε, είναι δύσκολη η χειροκίνητη μέθοδος της δοκιμής διείσδυσης με την black box τεχνική, χρησιμοποιούνται αυτοματοποιημένα εργαλεία. Τα εργαλεία αυτά ενδέχεται να είναι εργαλεία, που αυτοματοποιούν το σύνολο των ελέγχων ή να αυτοματοποιούν ορισμένες μόνο από τις λειτουργίες ελέγχων. Υπάρχουν και εργαλεία, που είναι πολύ εξειδικευμένα και χρησιμοποιούνται μόνο από ειδικούς ασφάλειας σε χειροκίνητους ελέγχους. Όλα αυτά τα εργαλεία διατηρούν βάσεις δεδομένων με γνωστές ευπάθειες και σύμφωνα με αυτές κάνουν τους ελέγχους. Το ζήτημα, όμως, είναι κατά πόσο αυτά τα εργαλεία έχουν ενημερωμένες βάσεις δεδομένων, κατά πόσο κάνουν σωστά τους ελέγχους, που επιθυμούμε, και συνεπώς σε ποιο βαθμό μπορούμε να τα εμπιστευόμαστε. Χρησιμοποιώντας διάφορα εργαλεία και με τους ελέγχους που διενεργήσαμε, διαπιστώσαμε ότι δεν παίρνουμε τα ίδια αποτελέσματα. Αυτό αποτελεί μια ένδειξη του ότι δεν είναι όλα τα εργαλεία αποτελεσματικά, καθώς επίσης ότι δεν έχουν όλα τις ίδιες δυνατότητες ελέγχου ασχέτως, αν στις δυνατότητες, που διαφημίζουν, αναφέρονται. Αυτό το φαινόμενο πιθανότατα είναι πιο έντονο στα εμπορικά εργαλεία, όπου ο κώδικας τους είναι κλειστός και δεν είναι εφικτό κάποιος να ελέγξει την αποτελεσματικότητα τους. Μάλιστα οι περισσότερες demo εκδόσεις τέτοιων εργαλείων, λειτουργούν μόνο με ιστότοπους, που περιέχουν εφαρμογές ιστού με πολλές ευπάθειες προσπαθώντας να πείσουν τους πιθανούς πελάτες τους για την αποτελεσματικότητα τους. Σε αυτές τις περιπτώσεις είναι ανάγκη να δίνεται ιδιαίτερη βαρύτητα στην εταιρεία, που αναπτύσσει το κάθε εργαλείο. Τα ανοικτού λογισμικού εργαλεία υπερέχουν κατά κάποιο τρόπο σε αυτό το σημείο, διότι μπορεί κάποιος να ελέγξει τον κώδικά τους, να βρει πληροφορίες για τη λειτουργία τους από διάφορους σχετικούς ιστότοπους, καθώς επίσης μπορεί ακόμη και να ζητήσει πληροφορίες από τους υπεύθυνους ανάπτυξης τους. Εμείς χρησιμοποιήσαμε για την έρευνά μας διάφορα εργαλεία, άλλα ανοικτού λογισμικού και άλλα εμπορικά, προσπαθώντας να γίνει όσο ήταν δυνατόν, και μια σύγκριση μεταξύ των αποτελεσμάτων τους. Τα συμπεράσματα, που εξάγονται από τη χρήση αυτών, είναι ότι τα εμπορικά εργαλεία είναι φιλικά προς το χρήστη με καλή τεκμηρίωση και οδηγίες για τη χρήση τους (wizards) και γενικά μπορούν να χρησιμοποιηθούν εύκολα από χρήστες, που δεν έχουν εξειδικευμένες γνώσεις και εμπειρία, στον έλεγχο της ασφάλειας εφαρμογών ιστού. Δίνουν και τη δυνατότητα εκτός από πλήρεις αυτοματοποιημένους ελέγχους να εκτελεσθούν και χειροκίνητοι έλεγχοι για κάποιους πιο έμπειρους και με προγραμματιστικές γνώσεις χρήστες. Παρέχουν τα αποτελέσματα ανίχνευσης (scan reports) μαζί με αναλυτικές πληροφορίες για τις ευπάθειες, που έχουν εντοπισθεί, καθώς επίσης και συμβουλές για την επίλυση των προβλημάτων. Όσον αφορά τα ανοικτού λογισμικού ή ελεύθερα εργαλεία, τα περισσότερα δεν έχουν καλή τεκμηρίωση των λειτουργιών τους και δεν παρέχουν αρκετές οδηγίες για τη χρήση τους. Ακόμη, τα περισσότερα εργαλεία αυτού του είδους παρέχουν κάποιες συγκεκριμένες λειτουργίες ή αυτοματοποιούν ένα ή περισσότερα τεστ ελέγχου. Εξαίρεση αυτών αποτελεί

το WebScarab, το οποίο παρέχει καλή τεκμηρίωση, βασικές οδηγίες, για να ξεκινήσει κάποιος με βασικούς ελέγχους, καθώς επίσης υπάρχει η δυνατότητα υποστήριξης και βοήθειας στη χρήση του μέσω e-mail από τους υπεύθυνους ανάπτυξης του εργαλείου. Το WebScarab αποτελείται από διάφορα plug-ins, τα οποία παρέχουν πλήρεις ελέγχους των εφαρμογών ιστού. Θα λέγαμε ότι το μόνο μειονέκτημα του εργαλείου αυτού, είναι η απαίτηση βασικών γνώσεων προγραμματισμού από τον χρήστη.

Αποτελέσματα από τον έλεγχο ασφάλειας του ιστότοπου του Τμήματος Ψηφιακών Συστημάτων

Στα πλαίσια της παρούσας διπλωματικής εργασίας πραγματοποιήθηκαν οι παρακάτω έλεγχοι με βάση το Testing Guide v.3 του OWASP:

1. **Information Gathering**
    a) **Spiders, Robots**
    b) **Search Engine Discovery/Reconnaissance**
    c) **Identify application entry points**
    d) **Testing for Web Application Fingerprint**
    e) **Application Discovery**
    f) **Analysis of Error Codes**
2. **Configuration Management**
    a) **SSL/TLS Testing (SSL Version, Algorithms, Key length, Digital Cert. Validity)**
    b) **DB Listener Testing**
    c) **Infrastructure Configuration Management Testing**
    d) **Application Configuration Management Testing**
    e) **Testing for File Extensions Handling**
    f) **Old, backup and unreferenced files**
    g) **Infrastructure and Application Admin Interfaces**
    h) **Testing for HTTP Methods and XST**
3. **Authentication Testing**
    a) **Credentials transport over an encrypted channel**
    b) **Testing for user enumeration**
    c) **Testing for Guessable (Dictionary) User Account**
    d) **Brute Force Testing**
    e) **Testing for bypassing authentication schema**
    f) **Testing for vulnerable remember password and pwd reset**
    g) **Testing for Logout and Browser Cache Management**
    h) **Testing for CAPTCHA Weak**
    i) **Testing Multiple Factors Authentication**
    j) **Testing for Race Conditions**

4. **Authorization Testing**
    a) **Testing for Path Traversal**
    b) **Testing for bypassing authorization schema**
    c) **Testing for Privilege Escalation**
5. **Denial of Service**
    a) **Testing for SQL Wildcard Attacks**
    b) **Locking Customer Accounts**
    c) **Testing for Denial of Service (DoS)**
    d) **User Specified Object Allocation**
    e) **User Input as a Loop Counter**
    f) **Writing User Provided Data to Disk**
    g) **Failure to Release Resources**
    h) **Storing too Much Data in Session**

Για τους ελέγχους αυτούς ακολουθήθηκε η μέθοδος black box μιας και θέλαμε να εξετάσουμε ποια είναι η εικόνα που παρουσιάζει προς τα "έξω" η εφαρμογή που εξετάζουμε σε σχέση με το επίπεδο ασφάλειάς της. Οι περισσότεροι από τους ελέγχους αυτούς πραγματοποιήθηκαν χρησιμοποιώντας κάποια εργαλεία που προτείνονται για τις περιπτώσεις τους και κάποιοι άλλοι, σε επίπεδο τεχνικής black box, αποτελούν χρήσιμες συμβουλές και εμπεριστατωμένους τρόπους για τους διαχειριστές και τους developers της εφαρμογής σχετικά με το πώς πρέπει να διατηρηθούν κάποια στοιχεία της εφαρμογής και πως πρέπει να πραγματοποιηθούν αυτοί οι έλεγχοι με τη συμβολή τους. Κάποιοι από τους έλεγχους αυτούς δεν μπορούν να πραγματοποιηθούν με την black box τεχνική που επιλέξαμε ως μέθοδο στην διπλωματική εργασία αυτή διότι αφορά ελέγχους που για να διενεργηθούν απαιτείται άμεση παρέμβαση τοπικά στον server της εφαρμογής και πραγματοποίηση του έλεγχου τοπικά στην εφαρμογή. Γενικά κατά τη διάρκεια διενέργειας των ελέγχων λάβαμε υπ' όψιν την κρισιμότητα της εφαρμογής μιας και αποτελεί τον ιστότοπο τμήματος ενός πανεπιστημιακού ιδρύματος που αποτελεί φορέα γνώσης και εκπαίδευσης και σε καμία περίπτωση δε θα έπρεπε να δημιουργηθεί πρόβλημα ασφάλειας της εφαρμογής εξαιτίας των σκοπών της εργασίας. Στους τέσσερις πίνακες που ακολουθούν παρουσιάζουμε τα κυριότερα αποτελέσματα από τις κατηγορίες ελέγχων που πραγματοποιήσουμε:

Πίνακας ***:*** **Αποτελέσματα από το Information Gathering**

Information Gathering

- Ανακαλύφθηκαν:

    1. Ο τύπος και η έκδοση του Web Server (Apache Server 2.2.3)
    2. Το είδος του λειτουργικού συστήματος στο οποίο «τρέχει» ο server (Linux – CentOS)
    3. Η γλώσσα με την οποία αναπτύχθηκε η εφαρμογή και η έκδοσή της (PHP 5.3.6)
    4. Όλες οι εφαρμογές που φιλοξενούνται στο site της εφαρμογής και τρέχουν στον web server της.
    5. Η εφαρμογή είναι «ανοιχτή» σε spiders/robots/crawlers χωρίς να ορίζει προστατευόμενους πόρους.
    6. Γίνεται χρήση κυρίως GET αιτημάτων λόγω του τρόπου ανάπτυξης της εφαρμογής
    7. Χρησιμοποιούνται παράμετροι οι οποίες αποτελούν σημεία εισόδου (entry points) που μπορούν να οδηγήσουν στην κακόβουλη χρήση της εφαρμογής.
    8. Το scripting περιβάλλον για τη δημιουργία και εκτέλεση προγραμμάτων στην πλευρά του server (X-BeanShell).
    9. Ο ιδιοκτήτης της εφαρμογής (Πανεπιστήμιο Πειραιώς).
    10. Πότε ξεκίνησε η διαδικτυακή λειτουργία της εφαρμογής (Σεπτέμβριος 2009).
    11. Ο DNS Admin (root@unipi.gr).
    12. Ο DNS Reserve (sr2-is.ted.unipi.gr)
    13. Το Domain που ανήκει η εφαρμογή (unipi.gr)
    14. Η IP της εφαρμογής στην οποία γίνεται το name resolution καθώς και μια δεύτερη IP που πιθανώς πρόκειται για την real IP του server στον οποίο τρέχει η εφαρμογή.
    15. Το εργαλείο-πλατφόρμα στην οποία αναπτύχθηκε η εφαρμογή (Wordpress)
    16. Οι θύρες που είναι ανοιχτές και οι υπηρεσίες που επιτρέπουν αυτές (Ports:80,443 – Services: http, ssl)
    17. Η πλήρης ταυτότητα του developer της εφαρμογής (Ονοματεπώνυμο και e-mail)
    18. Πηγαίος κώδικας της εφαρμογής που αναφέρεται και σε άλλες εφαρμογές που φιλοξενούνται στον ίδιο ιστότοπο και «τρέχουν» στον ίδιο server.

Πίνακας ***:*** **Αποτελέσματα από το Configuration Management Testing**

Configuration Management Testing
• <u>Προέκυψε ότι:</u> 1. Οι θύρες 80 και 443 είναι ανοιχτές για SSL/TLS επιθέσεις 2. Η έκδοση και το είδος του Web Server (Apache Server 2.2.3) 3. <u>Σχετικά με τον DB Listener:</u> Δεν επιστρέφονται πληροφορίες για τον κωδικό πρόσβασης, την έκδοση του, τους περιορισμούς που έχει ορίσει ο admin και δεν υπάρχει δυνατότητα καταγραφής των log files. 4. <u>Αναφορικά με την Αρχιτεκτονική:</u> Πρέπει να γίνει έλεγχος για το configuration της αρχιτεκτονικής, της ρυθμίσεις του firewall, το φορτίο του δικτύου της εφαρμογής, να δημιουργηθεί μια λίστα με τα ports που επιτρέπεται να είναι ανοιχτά που να ελέγχεται συνεχώς και επίσης να γίνεται πάντα update των νεότερων εκδόσεων (version). 5. Είναι απαραίτητη η εξέταση των σχόλιων του πηγαίου κώδικα για τυχόν ευπάθειες που προκύπτουν από αυτά. 6. Θα ήταν πολύ χρήσιμη η χρήση CGI σαρωτών για την πλήρη επανεξέταση των περιεχομένων του web server. 7. Δεν επιστρέφονται ευπαθείς καταλήξεις αρχείων με εξαίρεση κάποια .pdf αρχεία. 8. Υπάρχει πολιτική διατήρησης των back-up αρχείων σε ξεχωριστή τοποθεσία από τον web server όμως διατηρούνται πολλά αχρείαστα και μη αναφερόμενα αρχεία σε αυτόν. 9. Ισχύουν default ονόματα χρηστών και χρησιμοποιούνται διαφορετικά μηνύματα λάθους για λάθος κωδικό πρόσβασης και λάθος όνομα χρήστη αποκαλύπτοντας έμμεσα τους έγκυρους χρήστες. 10. Οι κωδικοί δεν θεωρούνται αδύναμοι ούτε και διαμοιράζονται. 11. Δεν γίνεται χρήση default λογαριασμών για monitoring και testing. 12. Η σελίδα του admin «φαίνεται» στο ίδιο port όπως και η εφαρμογή. 13. Δεν επιτρέπεται το XST, αυθαίρετες μέθοδοι HTTP, μέθοδοι αλλοίωσης ή παράκαμψη

των μεθόδων ελέγχου πρόσβασης. _

Πίνακας ***:*** **Αποτελέσματα από το Authentication Testing.**

Authentication Testing

- Προέκυψαν τα παρακάτω αποτελέσματα:

    1. Χρησιμοποιείται επικοινωνία HTTP χωρίς κρυπτογράφηση που είναι ευπαθής σε επιθέσεις sniffing.

    2. Γίνεται χρήση διαφορετικών μηνυμάτων λάθους για περιπτώσεις λάθους ονόματος χρήστη και λάθους κωδικού πρόσβασης με αποτέλεσμα να αποκαλύπτονται έμμεσα οι έγκυροι χρήστες και να είναι εύκολη η διατήρηση μιας λίστας χρηστών.

    3. Ευπάθεια σε brute force επίθεση με χρήση λεξικού και χρήση default συνθηματικών για δεδομένο σύστημα ανάπτυξης της εφαρμογής (wordpress).

    4. Γίνεται χρήση του default ονόματος χρήστη ”admin”.

    5. Με brute force επίθεση βασισμένη σε λεξικό καταφέρνουμε να αποκαλύψουμε κάποιους έγκυρους χρήστες.

    6. Δεν μπορεί να παρακαμφθεί το σχήμα αυθεντικοποίησης με άμεσο αίτημα σελίδων ή τροποποίηση παραμέτρων.

    7. Δεν υπάρχει γραμμική/επαναληπτική σχέση των SessionIDs άρα είναι δύσκολο να μαντέψει κάποιος ένα SessionID.

    8. Η εφαρμογή δεν είναι ευπαθής σε SQL injection με χρήση λεξικού.

    9. Δε γίνεται χρήση μυστικών ερωτήσεων κατά την επανάκτηση του κωδικού πρόσβασης και η εφαρμογή στηρίζεται στο επίπεδο ασφάλειας του εκάστοτε ηλεκτρονικού ταχυδρομείου του χρήστη.

    10. Υπάρχει δυνατότητα απομνημόνευσης του κωδικού πρόσβασης στη σελίδα αυθεντικοποίησης της εφαρμογής.

    11. Υπάρχει η δυνατότητα αποσύνδεσης με τη χρήση κουμπιού σε εμφανές σημείο.

    12. Με τη χρήση του κουμπιού “back” μπορούμε να επιστρέψουμε σε σελίδες που επισκεφθήκαμε όσο ήμασταν αυθεντικοποιημένοι στο σύστημα κατάσταση επικίνδυνη για σελίδες με κρίσιμα δεδομένα.

    13. Δεν επιστρέφονται σελίδες με τη χρήση κάποιου cookie έγκυρου χρήστη.

    14. Δε ακολουθείται πολιτική Timeout Logout και τα cookies πρακτικά να διαρκούν για άπειρο χρόνο.

    15. Δεν υπάρχουν CAPTCHAs στην εφαρμογή μας.

16. Δεν επιτρέπονται πολλαπλοί λογαριασμοί χρηστών με το ίδιο όνομα άρα δεν προκύπτουν συνθήκες αγώνα δρόμου από τέτοιες περιπτώσεις.

Πίνακας : **Αποτελέσματα από το Authorization και Denial of Service Testing.**

Authorization & Denial of Service Testing

- <u>Οι έλεγχοι έδειξαν ότι:</u>

    1. Η εφαρμογή μας είναι ευπαθής σε Path Traversal
    2. Το Authorization schema της εφαρμογής διατηρεί τους ρόλους και τα δικαιώματα αυτών χωρίς να υπάρχει δυνατότητα παράκαμψής του.
    3. Η εφαρμογή μας σχετικά με το Privilege Escalation δεν παρουσιάζει κάποια ευπάθεια αφού διατηρείται η κλιμακούμενη ανάθεση δικαιωμάτων.
    4. Δεν εφαρμόζεται μηχανισμός κλειδώματος χρηστών κατά την επαναλαμβανόμενη εισαγωγή στοιχείων στις διάφορες σελίδες αυθεντικοποίησης χρηστών.

Σύνοψη

Σήμερα υπάρχει ανάγκη από υψηλής ποιότητας εφαρμογές ιστού με μεγάλη λειτουργικότητα και αποδοτικότητα, το οποίο καθιστά πολύ μεγαλύτερη την ανάγκη της ασφάλειάς τους από κακόβουλες πράξεις. Η μέτρηση της ασφάλειας δεν είναι όμως μια απλή και τυπική διαδικασία, καθόσον απαιτεί κατάλληλο γνωστικό υπόβαθρο. Αυτό αφορά τόσο τους υπεύθυνους ανάπτυξης των εφαρμογών ιστού, δηλαδή τους προγραμματιστές, όσο και τους υπεύθυνους ασφάλειας των πληροφοριακών συστημάτων των οργανισμών-επιχειρήσεων, που χρησιμοποιούν τις εφαρμογές αυτές. Το λάθος, το οποίο γίνεται στις περισσότερες περιπτώσεις και αφορά κυρίως τη σχεδίαση των εφαρμογών ιστού, είναι η μεγάλη σημασία, που δίνεται στη λειτουργικότητα τους, καθώς επίσης και στη γρήγορη ανάπτυξη και εγκατάσταση τους, υποβαθμίζοντας μ' αυτό τον τρόπο τη σημασία της ασφάλειας. Αυτό οφείλεται και στη μη χρήση ενός αναγνωρισμένου προτύπου ανάπτυξης ασφαλών εφαρμογών. Χρειάζεται, όσοι ασχολούνται με το θέμα της ασφάλειας εφαρμογών, να έχουν συνεχή ενημέρωση σχετικά με τα ζητήματα ασφάλειας και τις τρέχουσες εξελίξεις στο χώρο. Τα ζητήματα στο χώρο της ασφάλειας είναι τόσο ρευστά, γι' αυτό και επιβάλλεται η ανάγκη συνεχούς εκπαίδευσης πάνω σε νέες και καλύτερες πρακτικές για την ανάπτυξη και λειτουργία ασφαλών εφαρμογών ιστού, είτε αυτό αφορά το υλικό είτε το λογισμικό, το οποίο χρησιμοποιείται. Τελικά, η ασφάλεια θα πρέπει να είναι πάντα το πρώτο μέλημα κατά την ανάπτυξη νέων εφαρμογών, η οποία είναι ανάγκη να βασίζεται σε πρότυπα ασφάλειας.

Συνοψίζοντας, λοιπόν, είμαστε σε θέση να συμπεράνουμε ότι ο έλεγχος της ασφάλειας εφαρμογών ιστού, δεν είναι κάτι το ανέφικτο, εάν επιλεγούν οι κατάλληλες μέθοδοι και τα κατάλληλα εργαλεία ελέγχου. Οφείλουν να υποστηρίζονται πάντα από τις απαραίτητες γνώσεις του ελεγκτή και τη συνεχή ενημέρωσή του στο αντικείμενο της ασφάλειας των εφαρμογών ιστού. Σαφέστατα, οι έλεγχοι απαιτείται να γίνονται συνεχώς και να λαμβάνονται τα κατάλληλα μέτρα για την εξάλειψη των ευπαθειών, οι οποίες εντοπίζονται, ώστε να υπάρχει συνεχής βελτίωση των εφαρμογών ιστού στο θέμα της ασφάλειας.

Βιβλιογραφία

1. Χρήστος Παπακρίβος (2008), «Μέτρηση Ασφάλειας Εφαρμογών Ιστού», Διπλωματική εργασία, Τμήμα Εφαρμοσμένης Πληροφορικής, Πανεπιστήμιο Μακεδονίας, Θεσσαλονίκη.
2. Thomas Wilhelm (August 2009), Professional Penetration Testing – Creating and Operating a formal hacking Lab.
3. OWASP, "A Guide to Building Secure Web Application", www.owasp.org
4. OWASP, "OWASP Testing Guide 2008 V3.0", https://www.owasp.org/images/5/56/OWASP_Testing_Guide_v3.pdf
5. OWASP_Testing_Guide_Part2.pdf
6. OWASP_Testing_Guide_Part3.pdf
7. https://www.owasp.org/index.php/OWASP_Testing_Project
8. Καζακώνης Αναστάσιος (Ιούνιος 2005), OWASP, The Open Web Application Security, Application Security FAQ – OWASP_faq_Greek.pdf
9. Ron Gula, Michel Arboi (June 24, 2011 – Revision 6), «Performing PCI DSS and OWASP Web Application Audits with Nessus» http://www.nessus.org/sites/drupal.dmz.tenablesecurity.com/files/uploads/documents/whitepapers/nesssus-web-based-auditing_0.pdf
10. CERT The Open Web Application Project - http://www.cert.org
11. CERT The Open Web Application Project - http://www.cert.org/stats/
12. www.internet2.edu
13. http://sectools.org/
14. Nmap - http://nmap.org/
15. Brutus - http://www.hoobie.net/brutus/
16. Netcraft - http://www.netcraft.com
17. Integrigy http://www.integrigy.com/
18. Gunter Ollmann: "Web Based Session Management"- http://www.technicalinfo.net/papers/WebBasedSessionManagement.html
19. Johnny Long, "Google Hacking", http://johnny.ihackstuff.com
20. RFC 2965 - http://www.ietf.org/rfc/rfc2965
21. WebScarab - http://www.owasp.org/index.php/OWASP_WebScarab_Project
22. http://ha.ckers.org/xss.html
23. *http://www.onjava.com/pub/a/onjava/2003/05/07/blackboxwebtest.html*)
24. *http://java.sun.com/j2ee/1.4/docs/tutorial/doc/Security5.html)*

25. *http://java.sun.com/j2ee/1.4/docs/tutorial/doc/Security5.html)*
26. Kathleen Hickey (March 24, 2011), GCN The online authority for government IT professionals, «Cyberattacks on agencies increase as preparedness lags» - http://gcn.com/articles/2011/03/24/omb-reports-spike-in-agency-cyber-attacks.aspx