Χαροκόπειο Πανεπιστήμιο

Τμήμα Πληροφορικής και Τηλεματικής

Πτυχιακή Εργασία

Ανάπτυξη Firefox add-on για την προστασία της ιδιωτικότητας των δεδομένων του χρήστη και των Web Servers

Γιώργος Παούρης

ΑΜ:20923

Επιβλέπων Καθηγητής

Κωνσταντίνος Τσερπές

Μέλης της εξεταστικής επιτροπής

Μάρα Νικολαΐδου

Χρύσα Σοφιανοπούλου

Αθήνα, Σεπτέμβριος 2013

Πρόλογος

Αρχικά θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της πτυχιακής μου εργασίας τον κο. Κωνσταντίνο Τσερπέ για την απεριόριστη υπομονή που έδειξε καθώς επίσης και για την βοήθεια και τις γνώσεις που μου πρόσφερε σε όλη την διάρκεια εκπόνησης της εργασίας μου.

Εν συνέχεια θα ήθελα να ευχαριστήσω και τα υπόλοιπα μέλη της επιτροπής την κα. Μάρα Νικολαΐδου και την κα. Χρύσα Σοφιανοπούλου για τον ενδιαφέρον που μου έδειξαν και που μου αξιολόγησαν την προσπάθειά μου.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου για την μεγάλη κατανόηση και υπομονή που έδειξε στα φοιτητικά μου χρόνια και σε όλη την διάρκεια την πτυχιακής εργασίας. Επίσης θα ήθελα να ευχαριστήσω συγγενείς και φίλους για την ηθική τους συμπαράσταση.

Περιεχόμενα

Περίληψη

Η πρόσβαση σε μεγάλο όγκο δεδομένων του Διαδικτύου γίνεται συνήθως μέσω του HTTP (Hypertext Transfer Protocol) το οποίο εχεί εδραιωθεί ως το παγκόσμιο πρωτόκολλο μεταφοράς δεδομένων. Αυτό έχει ως αποτέλεσμα, το πρωτόκολλο να αποτελεί ένα κοινό μέσο επιθέσεων σε Web Servers αλλα και σε χρήστες για την εξαγωγή πολύτιμων ιδιωτικών στοιχείων και πληροφοριών. Για να αποφευχθούν λοιπόν με επιτυχία αυτές οι επιθέσεις είναι σημαντικό να κατασκευαστεί ένα ακριβές σύστημα ανίχνευσης.

Η παρούσα εργασία αποσκοπεί στην δημιουργία ενός τέτοιου συστήματος, ικανό να προστατεύει τον χρήστη και τους Web Servers από κακόβουλες επιθέσεις που γίνονται αξιοποιώντας το HTTP πρωτοκόλλο. Στο σύστημα αυτό προστέθηκαν λειτουργίες για την καταπολέμηση επιθέσεων που προέρχονται από τα αυτόματα αναδυόμενα παράθυρα διαφημίσεων στους browsers, καθώς επίσης προστέθηκαν και λειτουργίες που ελέγχουν μέσω κάποιων μοντέλων ανώμαλης συμπεριφοράς κάθε HTTP request που φεύγει από τον browser του χρήστη για κακόβουλες επιθέσεις.

Προκειμένου να εφαρμοστούν τα παραπάνω, το ακριβές σύστημα ανίχνευσης υλοποιήθηκε υπό την μορφή ενός extension, ειδικά κατασκευασμένο να λειτουργεί για τον περιηγητή Firefox. Το extension παρέχει στους Web Servers και στον χρήστη προστασία από κακόβουλες επιθέσεις προσφέροντάς τους και επιπλέον επιλογές ασφάλειας.

Abstract

Access to high-volume Internet data is typically done via the HTTP (Hypertext Transfer Protocol) which has been established as the global data transfer protocol. This has as result, the protocol to be a common means of attacks on Web Servers and to other users to extract valuable data and private information. So to avoid successfully these attacks is essential to construct an accurate detection system.

This work aims to create such a system, capable of protecting the user and the Web Servers from malicious attacks made through the HTTP protocol. This system functions not only to prevent attacks from pop-ups ads in browsers but it also checks each HTTP request leaving the user's browser for any anomaly behavior through some detection models.

To implement the above, the exact detection system was implemented in the form of an extension, specially made for Firefox browser. The extension provides to the Web Servers and to the users protection by offering additional security options.

Λέξεις Κλειδιά

Επέκταση Firefox, μοντέλα ανίχνευσης ανώμαλης συμπεριφοράς, κακόβουλες επιθέσεις, προστασία ιδιωτικότητας του χρήστη, προστασία ιδιωτικότητας των Web Servers, HTTP πρωτόκολλο, Javascript, XUL, Mozilla, add-ons.

Key Words

Firefox Extension, anomaly detection models, malicious attacks , user’s privacy protection, Web Serves privacy protection, HTTP protocol, Javascript, XUL, Mozilla, add-ons.

1 Εισαγωγή

Το Διαδίκτυο αποτελεί πλέον ένα αναπόσπαστο κομμάτι του 21ου αιώνα. Από την εποχή που υπήρχαν οι υπολογιστές το Διαδίκτυο δεν σταμάτησε ποτέ να μεγαλώνει και να εξελίσσεται. Είναι γεγονός ότι αποτέλεσε ένα απαραίτητο εργαλείο για την ανάπτυξη σημαντικών εταιρειών και οργανισμών όπως είναι οι τράπεζες και ο στρατός. Η ολοένα και περισσότερη ανάπτυξή του κατάφερε να φέρει σε επαφή χρήστες από τα διάφορα μέρη του κόσμου. Ωστόσο πολλοί από τους χρήστες παρουσιάζονται να μη έχουν καλές προθέσεις όταν κάνουν χρήση του Διαδικτύου. Είναι ικανοί να διαβάζουν και να διαγράφουν πολύτιμες πληροφορίες οι οποίες όμως δεν τους ανήκουν διαταράσσοντας με αυτόν τον τρόπο ολόκληρο το σύστημα.

"Από τη στιγμή που το Διαδίκτυο είναι ένα δίκτυο συνδεδεμένων υπολογιστών, κάθε χρήστης έχει την δυνατότητα να μοιραστεί πληροφορίες με άλλους χρήστες γενόμενος, πολλές φορές, ο ίδιος δημιουργός και πάροχος των πληροφοριών αυτών. Ο όγκος της πληροφορίας στο Διαδίκτυο είναι πράγματι μεγάλος. Παρ' όλα αυτά, υπάρχουν πληροφορίες ευκολότερα και δυσκολότερα προσβάσιμες από τον χρήστη" (**wikipedia, 2004**) [1].

Αφού το Διαδίκτυο αποτελεί τον πυρήνα για την ανταλλαγή της πληροφορίας, καθιστάται αυτομάτως ευάλωτο στις επιθέσεις των χρηστών που θέλουν να αποκτήσουν αυτές τις πληροφοριές. Τα κακόβουλα λογισμικά όπως τα trojan horses και κάθε είδους adware και spyware που διακινούνται μέσω του Διαδικτύου αποτελούν μερικά παραδείγματα παραβιάσης των προσωπικών δεδομένων του χρήστη και των Web Servers. Τα περισσότερα από αυτά αποκομίζονται με την πρόσβαση του χρήστη σε κακόφημες ιστοσελίδες. Ωστόσο υπάρχουν και άλλοι τρόποι παραβίασης των προσωπικών δεδομένων όπως το ***ClickJacking,*** που προκαλεί τα αυτόματα αναδυόμενα παράθυρα γνωστά και ως pop-ups. Σε αυτήν την περίπτωση ο χρήστης εντελώς ανυποψίαστος κάνει κλικ σε ένα αντικείμενο μέσα στην ιστοσελίδα (π.χ έναν σύνδεσμο) ή και στο παράθυρο του browser προκαλώντας την ενεργοποίηση ενός παραθύρου.

[1] http://el.wikipedia.org/wiki/%CE%94%CE%B9%CE%B1%CE%B4%CE%AF%CE%BA%CF%84%CF%85%CE%BF

Αυτοί μηχανισμοί χρησιμοποιούνται κυρίως για την προβολή διαφημίσεων και αποτελούν το στόχαστρο για διαδικτυακές επιθέσεις. Ένας άλλος τρόπος παραβίασης δεδομένων που είναι εξίσου δημοφιλής με όσους αναφέρθηκαν προηγουμένως είναι μέσω των HTTP requests. Κακόβουλοι χρήστες προσπαθούν να βρίσκουν αδυναμίες στα HTTP requests για την εισβολή κακόβουλων λογισμικών προκαλώντας ζημιά στους Web Servers και στους χρήστες. Προκειμένου λοιπόν να αποφευχθούν τέτοιου είδους επιθέσεις είναι συνετό να κατασκευαστούν συστήματα ανίχνευσης εισβολών (IDS).

Ένα σύστημα IDS (**intrusion detection system**) επιτρέπει στα υπολογιστικά συτήματα να ανιχνεύουν αυτομάτως συμπεριφορές επίθεσης. Τα συστήματα αυτά για πολυ καιρό υπήρξαν το κέντρο μελέτης και ανάπτυξης και καθιερώθηκαν όταν εταιρείες άρχισαν να ανεβάζουν διαδικτυακά σημαντικά δεδομένα. Η όλη ιδέα σε ένα τέτοιο σύστημα είναι να ανιχνεύει συμπεριφορές που ίσως να αποκλίνουν από την φυσιολογική, αποτρέπωντας με αυτόν τον τρόπο την εισβολή.

1.1 Αντικείμενο πτυχιακής

Στο πλαίσιο της ερευνητικής δραστηριότητας του τμήματος Πληροφορικής και Τηλεματικής του Χαροκόπειου Πανεπιστημίου έχει αναπτυχθεί ένα σύστημα το οποίο είναι ικανό να ανιχνεύει και να αποτρέπει από επιθέσεις που προέρχονται από το Διαδίκτυο μέσω του HTTP πρωτοκόλλου.

Το σύστημα ανίχνευσης παρέχει ένα μηχανισμό ο οποίος αποτρέπει τα αυτόματα αναδυόμενα παράθυρα, προστατεύοντας τον χρήστη από τυχόν κακόβουλες επιθέσεις. Επίσης το σύστημα δημιουργεί ένα μοντέλο φυσιολογικής συμπεριφοράς από τα δεδομένα εκπαίδευσης και το χρησιμοποιεί για να εντοπίσει επιθέσεις εις βάρος των Web Servers. Το σύστημα στο μοντέλο αυτό έχει συσχετίσει κάθε server-side πρόγραμμα (path) με όλες τις πιθανές παραμέτρους του, που μπορούν να βρίσκονται στο URL του κάθε HTTP request. Έτσι το κάθε πρόγραμμα γνωρίζει τα φυσιολογικά χαρακτηριστικά των παραμέτρων του και μέσω της χρήσης μοντέλων ανίχνευσης μπορούμε να διαπιστώσουμε πότε αυτά αποκλίνουν από τα φυσιολογικά επίπεδα. Η ακρίβεια του συστήματος η οποία καθορίζει σε σημαντικό βαθμό την αποτελεσματικότητα του εξαρτάται από τη φάση εκπαίδευσης. Η φάση της

εκπαίδευσης περιέχει αρχεία με ένα καθορισμένο όγκο δεδομένων, τα οποία είναι καθαρά από επιθέσεις του Διαδικτύου και βοηθούν το σύστημα να φτιάξει την φυσιολογική συμπεριφορά για την προστατευόμενη εφαρμογή. Η ύπαρξη καλών μοντέλων στο σύστημα διαδραματίζει σημαντικό ρόλο στην αποτελεσματική «απόκρουση» των επιθέσεων.

Αντικείμενο της παρούσας πτυχιακής είναι η δημιουργία ενός extension με τις παραπάνω λειτουργίες που να προστατεύει τον χρήστη και τους Web Servers από κακόβουλες επιθέσεις. Για τον σκοπό αυτό, πρώτα μελετήθηκαν οι γλώσσες με τις οποίες μπορεί κάποιος να φτιάξει ένα extension. Η Javascript που αποτελεί ένα είδος αντικειμεστραφούς Java σε επίπεδο διαδικτύου και η XUL που θεωρείται μια καλύτερη έκδοση της γλώσσας HTML είναι αρκετές για να μπορέσει κάποιος να φτιάξει ένα extension. Στην συνέχεια μελετήθηκε το documentation για τα extension του Firefox, το οποίο περιέχει σημαντικές οδηγιές για την χρήση των διάφορων υπηρεσιών που παρέχει ο περιηγητής. Ως τελικό βήμα μελετήθηκαν οι τεχνικές παραβίασης των δεδομένων του χρήστη και των Web Servers και δημιουργήθηκαν πρακτικοί μηχανισμοί και μοντέλα τα οποία επιλύουν σε ικανοποιητικό βαθμό το πρόβλημα της ιδιωτικότητας.

1.2 Δομή πτυχιακής

Η εργασία είναι δομημένη ως εξής:

Το 2^{ο} κεφάλαιο αποτελεί το υπόβαθρο της πτυχιακής εργασίας και περιλαμβάνει μια ιστορική αναδρομή, την αποσαφήνιση εννοιών και την περιγραφή εργαλείων χρήσιμων για την υλοποιήση της παρούσας εργασίας. Πιο συγκεκριμένα γίνεται μια ιστορική αναδρομή του οργανισμού Mozilla και περιγράφεται το HTTP πρωτόκολλο. Ακόμα, το κεφάλαιο πραγματοποιεί μια επισκόπηση στις γλώσσες Javascript και XUL. Συγκεκριμένα δίνονται ορισμοί και ορισμένα παραδείγματα χρήσης των γλωσσών.

Το 3^{ο} κεφάλαιο περιλαμβάνει όλη την σημαντική έρευνα που έγινε στα πλαίσια της παρούσας πτυχιακής. Συγκεκριμένα αναφέρονται οι μελέτες που έγιναν με στόχο την την προστασία της ιδιωτικότητας του χρήστη και των Web Servers. Αρχικά δίνεται

ένας ορισμός για τα μοντέλα και περιγράφεται ο ρόλος τους στην ανίχνευση των ανώμαλων συμπεριφορών. Εν συνέχεια περιγράφονται μοντέλα και μηχανισμοί που αποσκοπούν να ανιχνεύουν ανώμαλες συμπεριφορές που προέρχονται από επιθέσεις του Διαδικτύου.

Το 4^{ο} κεφάλαιο περιλαμβάνει την υλοποίηση μιας πιλοτικής εφαρμογής που πραγματοποιήθηκε με βάση τα μοντέλα που αναφέρθηκαν στο 3^{ο} κεφάλαιο. Αρχικά αναφέρεται ο ρόλος της εφαρμογής και δυσκολίες που παρουσιάστηκαν κατά την υλοποίηση της. Στην συνέχεια δίνεται μια αναλυτική περιγραφή για την διαδικασία εγκατάστασης των μερών την εφαρμογής απαραίτητων για την ομαλή λειτουργία της και περιγράφεται αναλυτικά ο ρόλος τους. Ακόμα μέσω σχεδιαγραμμάτων παρουσιάζονται και εξηγούνται τα σημεία αλληλεπίδρασης της εφαρμογής με τον χρήστη και περιγράφονται αναλυτικά οι λειτουργίες τους και η ροή εκτέλεσής τους. Τέλος, στο παρόν κεφάλαιο περιγράφεται ένα πείραμα που πραγματοποιήθηκε με στόχο την επιβεβαίωση της λειτουργίας των παραπάνων λειτουργιών. Στο πείραμα αναφέρονται τα μοντέλα που χρησιμοποιήθηκαν και αναλύονται οι παραδοχές που λήφθηκαν υπόψη.

Στο 5^{ο} κεφάλαιο αναφέρονται τα συμπεράσματα που προέκυψαν από την υλοποίηση της παρούσας εργασίας. Δίνεται μια σύνοψη της εργασίας και μια ερμηνεία που εξηγεί αν τελικά λειτουργίες της εφαρμογής δουλεύουν σωστά ή όχι. Ακόμα, αναφέρονται οι πιθανές χρήσεις της εφαρμογής και ο τελικός της στόχος. Τέλος, αποτυπώνονται ορισμένες μελλοντικές κατευθύνσεις που θα μπορούσαν να εφαρμοστούν στην συγκεκριμένη εφαρμογή.

2 Υπόβαθρο

Στα επόμενα τμήματα αυτού του κεφαλαίου παρουσιάζονται έννοιες και εργαλεία που κρίθηκαν απαραίτητα για την διεκπαιρέωση του θέματος της πτυχιακής εργασίας. Αρχικά αναφέρεται η ιστορία εξέλιξης του οργανισμού Mozilla και στην συνέχεια ανφέρονται έννοιες που περιγράφουν το HTTP πρωτόκολλο και τις λειτουργίες του. Τέλος, περιγράφονται εργαλεία όπως οι γλώσσες προγραμματισμού Javascript και XUL.

2.1 Mozilla

2.1.1 Η εξέλιξη του Mozilla

Το έργο του Mozilla δημιουργήθηκε το 1998 με την κυκλοφορία του ανοικτού πηγαίου κώδικα του Netscape περιηγητή. Σκόπος αυτής της έκδοσης ήταν να αξιοποιηθεί η δημιουργικότητα και η εμπειρία πολλών προγραμματιστών ώστε να επιφέρουν επαναστατικές καινοτομίες στον κόσμο των περιηγητών. Από τον πρώτο χρόνο κιόλας, καινούργια μέλη από όλο τον κόσμο πρόσθεσαν λειτουργίες, ενίσχυσαν χαρακτηριστικά και έμειναν αφοσιωμένοι στην διαχείριση του έργου.

Με την δημιουργία λοιπόν μιας ανοικτής κοινότητας, ο Mozilla μπόρεσε να γίνει το μεγαλύτερο έργο που υπήρχε εκείνη την περίοδο. Οι προγραμματιστές του Mozilla κατάφεραν όχι μόνο να πετύχουν τον αρχικό τους στόχο που ήταν να σχεδιάσουν και να δημιουργήσουν τον επόμενο περιηγητή αλλά μπόρεσαν να επεκταθούν και σε μεγαλύτερη κλίμακα σχεδιάζοντας και φτιάχνοντας άλλα projects, εργαλεία προγραμματισμού και άλλου είδους περιηγητές. Οι άνθρωποι που συνεισέφεραν στο έργο του Mozilla ήταν παθιασμένοι στο γεγονός ότι η δημιουργία ελεύθερων λογισμικών θα μπορούσε να αλλάξει την αντίληψη των ανθρώπων για το Διαδίκτυο.

Το 2002 δημιουργήθηκε η πρώτη σημαντική έκδοση του Mozilla, η Mozilla 1.0. Η έκδοση αυτή περιελάμβανε πολλά καινούργια χαρακτηριστικά για τον περιηγητή όπως email client και άλλες εφαρμογές. Ωστόσο το 2002, το 90% των χρηστών του

πληθυσμού που χρησιμοποιούσαν τον Internet Explorer, δεν είχε αντιληφθεί ότι εκείνη την χρονιά ο περιηγητής Phoenix (που αργότερα ονομάστηκε Firefox) προερχόταν από τα μέλη της οργάνωσης του Mozilla που στόχος τους ήταν να παρέχουν την καλύτερη δυνατή εμπειρία περιήγησης σε όλους τους ανθρώπους. Το 2003 δημιουργήθηκε μια μη κερδοσκοπική οργάνωση από χορηγούς και άλλες εταιρείες εντελώς ανεξάρτητη από την οργάνωση του Mozilla. Η μη κερδοσκοπική οργάνωση συνέχισε να στηρίζει το έργο του Mozilla αλλά και να προωθεί καινοτομίες στον κόσμο του Διαδικτύου. Το τελευταίο, το πέτυχε με την έκδοση των ελεύθερων λογισμικών του Firefox και του Thunderbird.

Το 2004 εκδόθηκε η πρώτη έκδοση του Firefox, η Firefox 1.0. Ο Firefox είχε τεράστια επιτυχία και το λογισμικό του υπολογίζεται ότι το κατέβασαν πάνω από εκατό εκατομμύρια χρήστες του Διαδικτύου. Επειδή ο Firefox ήταν πολύ πετυχημένος πειρηγητής έβγαζε κάθε χρόνο και από μια έκδοση (από 1.0 μέχρι την 5.0 που είναι η τωρινή) καθιστώντας τον πρώτο στις προτιμήσεις των χρηστών.

Τα επόμενα χρόνια τα μέλη της οργάνωσης του Mozilla συνέχισαν να εργάζονται με σκοπό να επεκταθούν ακόμα περισσότερο σε καινούργιες περιοχές όπως για παράδειγμα να παρέχουν βελτιώσεις για την είσοδό των χρηστών στο Διαδίκτυο αλλά και για να συνεχίσουν να παρέχουν την καλύτερη εμπειρία περιήγησης σε αυτό.

2.1.2 Χαρακτηριστικά του Mozilla (Firefox)

“Ο Mozilla αποτελεί την καλύτερη μέχρις στιγμής υποστήριξη για τα εργαλεία HTML 4.0, XML, CSS1, DOM1 και RDF. Επίσης παρέχει υποστήριξη για τα χαρακτηριστικά του CSS2 (π.χ την τοποθέτηση περιεχομένου μιας ιστοσελίδας) και για τα χαρακτηριστικά του DOM2 (π.χ διεπιφάνεια του CSS)“ (**Jessica Halida Harjono, 2001**). Αυτό σημαίνει ότι θα μπορούσε να είναι δυνατή η δημιουργία μιας ολόκληρης διεπιφάνειας του χρήστη σε desktop εφαρμογές χρησιμοποιώντας αυτά τα εργαλεία. Η HTML και η XML θα μπορούσε να δομεί την διεπιφάνεια του χρήστη ενώ το CSS να την μορφοποιεί. Το W3C DOM θα μπορούσε να διαχειρίζεται τα διάφορα γεγονότα στην διεπιφάνεια του χρήστη όπως το πάτημα ενός κουμπιού και το RDF θα μπορούσε να αποτελέσει μια συλλογή πόρων. Συνδετικός κρίκος των παραπάνω εργαλείων θα μπορούσε να είναι η Javascript.

Εφόσον λοιπόν ο Mozilla υποστηρίζει XML, ενσωματώνει και άλλα πρότυπα τα οποία δημιουργούν προυποθέσεις για την ανταλλαγή δεδομένων και για την επεκτασιμότητα του ίδιου του περιηγητή. Η XML σταδιακά αναδεικνύεται ως το πρότυπο για την δημιουργία εγγράφων που υποστηρίζονται σε διάφορα μέσα ενημέρωσης. Επιπλέον περιλαμβάνει την XUL (XML-based user interface language) και το DTD (document type declaration). Η XUL είναι αυτή που καθορίζει με τις διάφορες ετικέτες της, τα κουμπιά και γενικά ολόκληρη την διεπιφάνεια του browser αφήνωντας με αυτόν τον τρόπο τους προγραμματιστές να τον «διακοσμούν» με την φαντασία τους.

Ουσιαστικά παρέχει μια γλώσσα η οποία περιγράφει την διεπιφάνεια του χρήστη με περισσότερα εργαλεία (widgets) από ότι μπορεί να παρέχει η HTML. Τέτοια εργαλεία μπορούν να είναι τα δέντρα ελέγχου, τα scrollbars και οι διαχωριστές (splitters). Η περιγραφή της διεπιφάνειας του χρήστη περιέχει διάφορα στοιχεία που ελέγχουν την εμφάνιση και την συμπεριφορά της διεπιφάνειας. Αυτά είναι:

- Περιεχόμενο: Περιγραφή των widgets σε μια διεπιφάνεια.
- Εμφάνιση: Το CSS δίνει την εμφάνιση στα widgets.
- Συμπεριφορά: Καθορίζεται είτε από την Javascript είτε μέσω της C++.
- Locale[2]: διεπιφάνεια με localize πληροφορία.

“Η χρήση της XUL έχει αρκετά πλεονεκτήματα. Μπορεί να υποστήρίζεται σε πολλές πλατφόρμες και έχει την δυνατότητα να ορίζει υποστηριζόμενες συσκευές διεπαφανειών. Επίσης έχει την δυνατότητα να «εκδημοκρατίσει» τον σχεδιασμό της διεπιφάνειας της εφαρμογής φέρνοντάς την στα λογίκα πλαίσια του προγραμματιστή όπως είναι η HTML και η Javascript” (**Jessica Halida Harjono, 2001**). Η XUL μπορεί να χρησιμοποιηθεί για την δημιουργία μικρών εφαρμογών αξιοποιώντας τις δυνατότητες του browser και μπορεί να προσθέτει UI (User Interface) στοιχεία που η HTML δεν υποστηρίζει.

[2] http://en.wikipedia.org/wiki/Locale

2.2 XUL

2.2.1 Εισαγωγή στην XUL

"XUL σημαίνει επεκτάσιμη γλώσσα της διεπιφάνειας του χρήστη. Είναι μια γλώσσα βασισμένη στην XML για την περιγραφή των περιεχομένων των αναδυόμενων παραθύρων και των διαλόγων. Επίσης διαθέτει υποδομές για όλους του σχετικούς ελέγχους των διαλόγων και των widgets όπως είναι τα menu αλληλεπίδρασης με τον χρήστη, τα scrollbars, τα progress bars κ.α" (**Jessica Halida Harjono, 2001**).

Η XUL είναι μια γλώσσα βασισμένη στην γραμματική της XML για τον σχεδιασμό στατικού GUI (Graphical User Interface) και όλη η ιδέα για τον προγραμματισμό της βασίζεται στην εύκολη σχεδίαση/δημιουργία γραφικών διεπιφανειών η οποία καθιστά ευχάριστη την ανάπτυξη τέτοιου είδους εφαρμογών.

Το κύριο μέλημα του προγραμματιστή όταν ξεκινάει να γράφει σε XUL είναι να αρχικά να δημιουργήσει τα επιθυμητά παράθυρα και τους διαλόγους μέσα από μια περιγραφή σε XML. Αυτή η περιγραφή έχει την δυνατότητα να δώσει στον προγραμματιστή μεγαλύτερο έλεγχο πάνω στα UI αντικείμενα κάτι το οποίο η HTML δεν μπορεί να κάνει. Άν η HTML μπορεί να περιγράψει ολόκληρο το έγγραφο μιας ιστοσελίδας, η XUL μπορεί να περιγραψεί ολόκληρο το παράθυρο του browser. Η XUL είναι τύπου case-sensitive[3] όπως και η XML. Ένα XUL αρχείο μπορεί να περιέχει XML αντικείμενα, HTML αντικείμενα όπως επίσης και ειδικά αντικείμενα της XUL, τα XUL αντικείμενα.

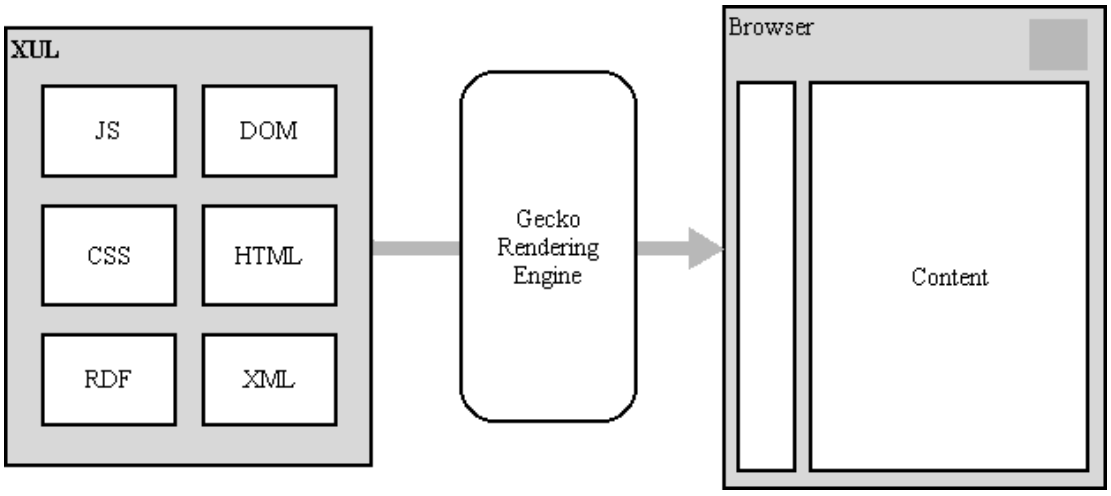
Επιτρέποντας λοιπόν στις γραφικές διεπιφάνειες να κατασκευάζονται εξ' ολοκλήρου από την XML και την Javascript, διευκολύνει σε μεγάλο βαθμό τους UI προγραμματιστές να τις διαχειρίζονται καλύτερα. Ακόμα η XUL τους δίνει την δυνατότητα να δημιουργούν menus και κουμπιά με τα δικά τους χρώματα και σχέδια επιβάλλοντάς τους και συγκεκριμένους κανόνες.

[3] http://en.wikipedia.org/wiki/Case_sensitivity

Γράφοντας ένα XUL αρχείο είναι σχεδόν το ίδιο γράφοντας και ένα XML. Η XUL έχει αρκετά style-sheet[4] αρχεία τα οποία περιγράφουν την συμπεριφορά των μερών που την απαρτίζουν. Επίσης οι λεπτομέρειες που περιγράφουν μια εφαρμογή σε ένα XML αρχείο όπως η σύνταξη για την κατασκευή ενός scrollbar μένουν σε ξεχωριστά αρχεία. Μια XUL γραφική διεπιφάνεια λοιπόν, είναι μια συλλογή από μη συνδεδεμένα widgets εώς ότου αυτά προγραμματιστούν.

Η Javascript αποτελεί μια βοήθεια για την ένωση των widget δίνοντάς τους ενίοτε και παραπάνω λειτουργικότητα ή και ακόμα συνδέοντάς τα ίσως με κάποια συνάρτηση που έχει δημιουργηθεί σε C++.

Η XUL μπορεί να περιέχει και DTD αρχεία. Τα αρχεία αυτά ορίζουν σταθερές με συγκεκριμένες τιμές οι οποίες μπορούν να χρησιμοποιηθούν σε όλα τα XUL αρχεία. Η ιδιότητα αυτή καθιστά την XUL «τακτοποιημένη» και οργανωμένη. Οι σταθερές των DTD αρχείων τοποθετούνται ως τιμές σε παραμέτρους οντοτήτων των XUL αρχείων. Με αυτόν τον τρόπο όταν αλλάξει η τιμή μιας σταθεράς στο DTD αρχείο θα αλλάξει αντίστοιχα και η τιμή της παραμέτρου. Έτσι λοιπόν ολόκληρο το σύστημα μπορεί να απαρτίζεται από πολλά DTD αρχεία και ανάλογα από με τις locale ρυθμίσεις του κάθε XML αρχείου θα καλείται και το σωστό DTD αρχείο.



Εικόνα 2.2.1.1: Διαδικασία εκτέλεσης της μηχανής Mozilla (Jessica Halida Harjono, 2001)

[4] Αρχεία που περιγράφουν το βασικό σχεδιασμό της εφαρμογής (π.χ χρώματα,εικόνες)

Όλη η XML χρειάζεται κάποιον parser[5] ή κάποιου είδους μηχανή για την τρέξει στο τέλος. Αυτή την δουλειά την κάνει η Gecko. Η Gecko είναι μια rendering[6] μηχανή η οποία γνωρίζει την XUL και ξέρει πως να την εμφανίζει σε pixels στην οθόνη.

2.2.2 Πλεονεκτήματα XUL

Οι προγραμματιστές των online εφαρμογών και των τοπικών εφαρμογών (εκτελούνται τοπικά στον υπολογιστή) έχουν βιώσει την χρησιμότητα της XUL διότι:

- Είναι εύκολο να κατασκευαστεί για πολλές πλατφόρμες.
- Μπορεί να κατασκευάσει μικρές,δυνατές και γρήγορες για download εφαρμογές αξιοποιώντας τις δυνατότητες του browser.

Ως καταναλωτές:

- Οι XUL εφαρμογές είναι μικρές και γρήγορες για download γλιτώνοντας χρόνο για τον χρήστη.
- Οι XUL εφαρμογές μπορούν να τρέχουν σε πολλές πλατφόρμες δίνοντας στον χρήστη την ελευθερία να επιλέξει αυτές που επιθυμεί.
- Η ευκολία της «διακόσμισης» της γραφικής διεπιφάνειας αυξάνει την ικανότητα της προσαρμογής της εμφάνισης της στις προτιμήσεις του χρήστη.

2.2.3 Δομή XUL

Όπως προαναφέρθηκε και νωρίτερα οι XUL εφαρμογές αποτελούνται απο XML αρχεία με επέκταση τύπου .xul τα οποία καθορίζουν το περιεχόμενο της εφαρμογής. Τα CSS αρχεία παρέχουν ένα είδος μορφοποίησης της εξωτερικής εμφάνισης (skin) και κάποια συμπεριφορά στην εφαρμογή. Τα αρχεία Javascript παρέχουν προγραμματιστική συμπεριφορά. Τέλος, για περισσότερη πληροφορία στην διεπιφάνεια του χρηστή ίσως χρησιμοποιηθούν κάποιες PNG εικόνες ή κάποια αρχεία ακουστικού και οπτικού περιεχομένου.

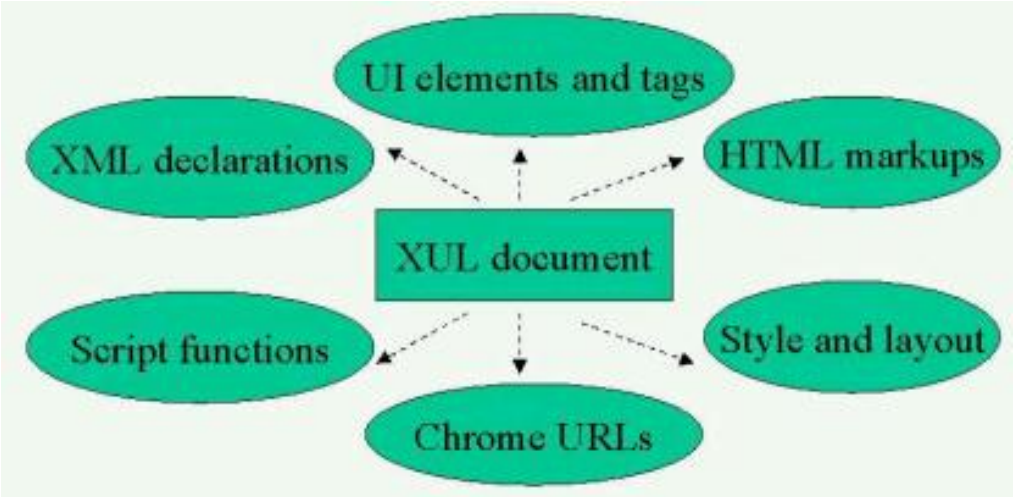
[5] http://en.wikipedia.org/wiki/Parsing

[6] http://en.wikipedia.org/wiki/Rendering_%28computer_graphics%29

Εικόνα 2.2.3.1: Ο ρόλος κάθε τεχνολογίας στον Firefox (Πηγή: developer.mozilla.org)

Ο ρόλος του XPCOM αναφέρεται στο 4^{ο} κεφάλαιο.

"Όλοι οι τύποι αρχείων βρίσκονται στις προδιαγραφές που συνιστώνται από το W3C και όλα μαζί αναφέρονται ως "chrome" της XUL εφαρμογής. Ο chrome αποτελεί το περιεχόμενο, τη συμπεριφορά, και την εμφάνιση της διεπαφής του χρήστη" (**Jessica Halida Harjono, 2001**).



Εικόνα 2.2.3.2: Μέρη που απαρτίζουν ένα XUL έγγραφο (Jessica Halida Harjono, 2001)

Τα ακόλουθα αποτελούν μέρη της δομής της XUL.

2.2.3.1 Chrome

“Ό Mozilla browser αποτελεί από μόνος του μια XUL εφαρμογή. Μαζί με τον Navigator στον κύριο φάκελό τους περιέχουν υποφακέλους με το όνομα chrome. Μέσα στον chrome φάκελο περιέχονται ξεχωριστές XUL εφαρμογές σε ξεχωριστούς υποφακέλους. Μέσα σε κάθε φάκελο εφαρμογής υπάρχουν υποφάκελοι που χωρίζουν την περαιτέρω εφαρμογή σε περιεχόμενο (XUL, JS), εξωτερική εμφάνιση (CSS) και locale (DTD) “ (**Jessica Halida Harjono, 2001**).

Ο chrome αποτελεί εκείνο το μέρος της εφαρμογής που βρίσκεται έξω από το κύριο περιεχόμενο του παραθύρου. Τα scrollbars, τα menu bars, τα progress bars και οι τίτλοι των παραθύρων αποτελούν μερικά από τα αντικείμενα που είναι μέρος του chrome. Ο chrome έχει τέσσερις τύπους παρόχων: πάροχοι περιεχομένου, πάροχοι εξωτερικής εμφάνισης, πάροχοι πλατφόρμας και οι localization πάροχοι.

- Ο πάροχος εξωτερικής εμφάνισης είναι υπεύθυνος για την εξωτερική μορφοποίηση της εξωτερικής εμφάνισης του chrome. Ουσιαστικά είναι αυτός που παρέχει τα CSS αρχεία και τις εικόνες.
- Ο πάροχος περιεχομένου παρέχει όλα τα απαραίτητα αρχεία τα οποία αποτελούν την δομή του chrome (π.χ menu,scrollbars).
- Ο πάροχος πλατφόρμας παρέχει πληροφορίες εξωτερικής εμφάνισης και δομής.
- Ο localization πάροχος παρέχει τις συμβολοσειρές που χρησιμοποιούνται από αντικείμενα στην XUL.

Τα κείμενα επάνω στους διαλόγους, οι εικόνες στα κουμπιά και γενικά όλο το περιεχόμενο του παραθύρου του περιηγητή είναι αποτέλεσμα των τεσσάρων παρόχων. Ο πάροχος περιεχομένου είναι αυτός που περιγράφει το κύριο περιεχόμενο του παραθύρου και θα μπορούσε να είναι οποιοσδήποτε τύπος αρχείου αναγνώσιμος από τον Mozilla. Ωστόσο θα πρέπει να είναι αρχείο XUL γιατί έχει σχεδιαστεί για να περιγράφει καλύτερα τα περιεχόμενα παραθύρων και διαλόγων.

Ένα χαρακτηριστικό της XUL είναι το μητρώο του chrome (Registry Chrome). Ο Mozilla διατηρεί έναν πίνακα που είναι γνωστός ως registry chrome με τον οποίο κρατάει αντιστοιχίσεις για κάθε έναν από τους τέσσερις παρόχους χρησιμοποιώντας

είτε κάποια αναγνωριστική συμβολοσειρά είτε κάποια άλλη σχετική πληροφορία. Για κάθε τύπο παρόχου δίνεται ένα μοναδικό όνομα, ορίζεται ένας βασικός κατάλογος για όλα τα αρχεία που αντιπροσωπεύουν και ορίζεται και μια τοποθεσία για το κύριο αρχείο του παρόχου. Επιπλέον ένα URL μπορεί να οριστεί για ένα αρχείο.

“Το Chrome URL αποτελεί μια έννοια του chrome ως ένα ολοκληρωμένο, δυναμικό αντικείμενο, κατά κάποιον τρόπο διαζευγμένο από τον «πυρήνα» της εφαρμογής το οποίο χρησιμοποιείται για να δείχνει σε κομμάτια της XUL και στα αντίστοιχα αρχεία τους” (**Jessica Halida Harjono, 2001**). Παρακάτω ακολουθεί ένα παράδειγμα του τέτοιου URL.

```
<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>
```

Ο παραπάνω κώδικας απλά δηλώνει ότι η μηχανή Gecko θα πρέπει να φορτώσει το chrome. Στην περίπτωση αυτή ο chrome είναι ένα αρχείο για την εξωτερική εμφάνιση της εφαρμογής ο οποίος φορτώνεται στο XUL αρχείο.

Θα πρέπει να ληφθεί υπόψη ότι αν δεν δηλωθεί στο τέλος του path του URL κάποιο αρχείο, το σύστημα θα ψάξει για ένα αρχείο με όνομα ίδιο με το πακέτο του URL. Στο πρώτο παράδειγμα το σύστημα θα αναζητήσει το αρχείο global.css.

Για να γίνει ακόμα πιο κατανοητή η ιεραρχία παρακάτω ακολουθεί ένα παράδειγμα της δομής του Mozilla.

```
[Mozilla]
  [chrome]
    [navigator]
         [skin]
           [default]
             navigatorSkin.jar
             ... other files bundled with navigator ...
           ... other skins used by users for navigator ...
         [content]
           [default]
             navigatorContent.jar
             ... other files bundled with navigator ...
          [platform]
            [default]
             ... optional JAR file and other files ...
         [locale]
           [default]
             ... optional JAR file and other files ...
    ... subdirectories for other specific window types ...
```

Εικόνα 2.2.3.1.1: Δομή αρχείων του Mozilla (Jessica Halida Harjono, 2001)

2.2.3.2 Widgets

Όλοι οι διαφορετικοί τομείς της XUL εφαρμογής περιέχονται σε widgets τα οποία αποκαλούνται και ως «κουτιά». Τα κουτιά από μόνα τους δεν έχουν κάποια εξωτερική εμφάνιση αλλά σκοπός τους είναι να συνδέονται με άλλα κουτιά και να σχηματίζουν μια ενιαία εφαρμογή στην οποία οι χρήστες θα μπορούν να πλοηγηθούν με ευκολία. Τα κουτιά επίσης παρέχουν και ένα συγκεκριμένο σχέδιο τοποθέτησης (layout) για την συμμετρική κατανομή του περιεχομένου. Αυτό επιτυγχάνεται με κάποια ειδικά γνωρίσματα (attributes) όπως το orient ή το width που αλλάζει το μήκος του κουτιού.

Μερικά widgets:

- Menu Bars and Menus
- Toolbars and Toolboxes

- Titled Buttons
- Tree Widget
- Tab Widget
- Sliders and Scrollbars
- The Splitter Widget
- Progress Meter
- Checkbox

2.2.3.3 Namespaces

“Ενα XML Namespace είναι μια συλλογή από στοιχεία τύπου και χαρακτηριστικά ονόματα. Τα namespaces αναγνωρίζονται από ένα μοναδικό όνομα, το οποίο είναι ένα URI. Επομένως, οποιοδήποτε στοιχείο τύπου ή χαρακτηριστικό όνομα σε ένα XML namespace μπορεί να αναγνωριστεί μοναδικά από ένα όνομα δύο μερών: το όνομα του XML namespace του και του τοπικού του ονόματος. Αυτό το σύστημα διμερούς ονομασίας είναι η μόνη λειτουργία των XML namespaces

Τα XML namespaces δηλώνονται με ένα στοιχείο xmlns, το οποίο μπορεί να συνδέσει ένα πρόθημα με το namespace. Η δήλωση βρίσκεται μέσα στα πλαίσια εφαρμογής των στοιχείων που περιέχουν τα χαρακτηριστικά και όλους τους απογόνους τους. Ο σκοπός των XML Namespaces είναι να διακρίνουν τα πανομοιότυπα στοιχεία τύπου από τα χαρακτηριστικά ονόματα. ” (**UNESCO, 1999**).

2.2.3.4 Scripting

Όπως είχαμε αναφέρει μια XUL γραφική διεπιφάνεια, είναι μια συλλογή από μη συνδεδεμένα widgets εώς ότου αυτά προγραμματιστούν. Η Javascript αποτελεί μια βοήθεια για την ένωση των widget δίνοντάς τους ενίοτε και παραπάνω λειτουργικότητα ή και ακόμα συνδέοντάς τα ίσως με κάποια συνάρτηση που έχει δημιουργηθεί σε C++.

Η XUL μπορεί να περιέχει τόσο HTML περιεχόμενο όσο και Javascript. Η Javascript ορίζεται με τρόπο παρόμοιο με αυτόν της HTML. Στην XUL επειδή δεν υπάρχουν <header> tags για να οριστεί η Javascipt, απλά τοποθετείται ανάμεσα στα άλλα περιεχόμενα ορίζοντας το <script> tag. Τέλος, μπορούμε να αναφερθούμε στην Javascript με παρόμοιο τρόπο όπως στα HTML έγγραφα.

"Η Javascript είναι καλύτερο να φυλάσσεται σε χωριστά αρχεία και απλά να τα συμπεριλαμβάνεις στο XUL αρχείο γιατί με αυτό τον τρόπο δεν υπερφορτώνεις τον XML parser να διαβάζει javascript σε XML περιεχόμενο" (**Jessica Halida Harjono, 2001**).

2.2.3.5 Διακόσμιση και σχεδιασμός εμφάνισης

Η διακόσμιση και ο σχεδιασμός είναι δύο στοιχεία που διαμορφώνουν την εμφάνιση της εφαρμογής. Είναι δυνατόν να μπορεί να αλλάξει ολόκληρη η εμφάνιση της εφαρμογής αλλά μόνο μέχρι το σημείο που την περιορίζει το CSS αρχείο που βρίσκεται στο πακέτο global. Οι προγραμματιστές μπορούν να φτιάχνουν και δικά τους στυλιστικά σχέδια δημιουργώντας άλλα CSS αρχεία, εμποδίζοντας με αυτόν τον τρόπο την Gecko να «στολίζει» την εφαρμογή με την παραδοσιακή εξωτερική εμφάνιση.

Όταν ένα URL του τύπου chrome://file/skin/ ορίζεται, τότε καλείται και το αντίστοιχο CSS αρχείο αυτού του file από τον πάροχο εξωτερικής εμφάνισης. Οποιοδήποτε CSS αρχείο μπορεί να χρησιμοποιηθεί αρκεί να υπάρχει το κατάλληλο URL προς αυτό, θεωρώντας πάντα ότι το chrome://file/skin/ αποτελεί το αρχικό σημείο για τον πάροχο εξωτερικής εμφάνισης.

2.3 Javascript

2.3.1 Εισαγωγή στην Javascript

Η Javascipt είναι μια αντικειμενοστραφής (object-oriented) γλώσσα προγραμματισμού. Μοιάζει πολύ με την Java ως προς τις λειτουργίες της αλλά πρόκειται για δύο εντελώς ανεξάρτητες γλώσσες, υλοποιημένες για συγκεκριμένους σκοπούς. "Η Javascript αποτελεί μια από τις δυνατότερες γλώσσες προγραμματισμού και για την εξήγησή της απαιτούνται τουλάχιστον 1000 σελίδες. Προς το παρόν σε αυτό το κεφαλαίο θα περιγράψουμε με συνοπτίκο τρόπο τις λειτουργίες της Javascript" (**Jennifer Golbeck, 2003**).

Η Javascript είναι μια scripting[7] γλώσσα που χρησιμοποιείται κυρίως για τον προγραμματισμό ιστοσελίδων. Αντί να υπάρχουν στατικές σελίδες, η Javascript μπορεί να αλλάζει δυναμικά το περιεχόμενο τους (π.χ εικόνες, κείμενα) ανάλογα με τις πράξεις των χρηστών. Σε αντίθεση με την HTML που δημιουργεί σελίδες που φαίνονται ίδιες σε όλους τους χρήστες, η Javascript μπορεί να αλληλεπιδρά με τον χρήστη μέσω διάφορων γεγονότων (π.χ πάτημα ενός κουμπιού) και να αλλάζει το περιεχόμενο ανάλογα με τις προτιμήσεις του. Η χρήση των cookies και άλλων παρόμοιων λειτουργιών μπορούν να προσθέσουν επιπλέον δυναμικότητα στην σελίδα του χρήστη.

Όπως αναφέρθηκε και στο προηγούμενο κεφάλαιο η Javascript εκτός από την κατασκευή ιστοσελίδων μπορεί να χρησιμοποιηθεί και για τον προγραμματισμό των μερών της XUL. Σε αντίθεση με την HTML όπου η Javascript έχει τον έλεγχο μόνο του HTML εγγράφου, στην XUL μπορεί να έχει τον έλεγχο ολόκληρου του παραθύρου του browser δίνοντας επιπλέον δυνατότητες στον προγραμματιστή.

Η Javascript έχει υλοποιηθεί για να τρέχει σχεδόν σε όλους τους υπάρχοντες browsers όπως τον Mozilla, τον Netscape, τον Safari, τον MAC OS X, Opera κτλ. Η πιο πρόσφατη έκδοση της Javascript είναι η 1.9. Γενικά όλα τα χαρακτηριστικά και λειτουργίες της γλώσσας που θα αναφερθούν υποστηρίζονται από παλιούς και καινούργιους browsers. Αλλα όταν ένας προγραμματιστής αναλαμβάνει να γράψει σε Javascript θα πρέπει να λάβει υπόψη του ποιές εκδόσεις περιηγητών το κοινό του θα χρησιμοποιεί ώστε να τοποθετήσει τις απαραίτητες λειτουργίες.

2.3.2 Basic Javascript

[7] http://en.wikipedia.org/wiki/Scripting_language

2.3.2.1 Ενσωματωμένες συναρτήσεις

Υπάρχουν δύο σημαντικές συναρτήσεις στην Javascript. Η μια είναι η συνάρτηση alert και η άλλη είναι η συνάρτηση document.write(). Η alert λέγεται συνάρτηση ακριβώς επειδή κάνει πολύπλοκες διαδικασίες στο παρασκήνιο. Όταν την εκτελούμε δεν μας ρωτά τι μέγεθος θέλουμε να είναι αναδυόμενο παράθυρο ή τι είκονα θέλουμε να βάλουμε στο παράθυρο. Όλα αυτά τα κάνει αυτόματα για εμάς.

Το μόνο που κάνουμε είναι να περνάμε μια συμβολοσειρά ως όρισμα της συνάρτησης alert. Μια συμβολοσειρά (string) είναι ένα σύνολο χαρακτήρων τοποθετημένων μαζί ανάμεσα σε διπλά ή μονά εισαγωγικά.

Η δεύτερη χρήσιμη συνάρτηση είναι η document.write(). Παρομοίως με την alert η η document.write() παίρνει και αυτή ως όρισμα ένα string. Σε αντίθεση με την alert που δείχνει το μήνυμα σε ένα παράθυρο η document.write() το εμφανίζει απευθείας στην σελίδα.

Η εικόνα 2.3.2.1.1 δείχνει δύο κείμενα που έχουν παραχθεί με την συνάρτηση document.write(). Η συνάρτηση είναι ιδιαίτερα χρήσιμη για την δημιουργία σελίδων που θα δείχνουν διαφορετικές για τον κάθε χρήστη. Ωστόσο χρειάζεται μεγάλη προσοχή όσον αναφορά την χρήση της. Η document.write() τοποθετείται συνήθως ανάμεσα στο σώμα του εγγράφου και χρησιμοποιείται συνήθως για να δείχνει περιεχόμενο που μόλις φόρτωσε η σελίδα. Οποιαδήποτε χρήση της μετά από αυτό το σημείο θα σβήσει όλα τα περιεχόμενα της και θα την ξαναφορτώσει με το καινούργιο μήνυμα.

Εικόνα 2.3.2.1.1: Κείμενα παραγμένα από την document.write() (Jennifer Golbeck, 2003)

2.3.2.2 Τύποι Δεδομένων

Όπως η Java έτσι και η Javascript έχει τύπους για να περιγράψει τα διάφορα δεδομένα της. Οι συνηθέστεροι τύποι δεδομένων είναι string, integers, boolean κτλ. Η διαδικασία δήλωσης μιας μεταβλητής είναι αρκετά διαφορετική από αυτήν που συναντάμε στις γλώσσες Java και C++. Η Javascirpt χρησιμοποιεί μια σταθέρα που λέγεται **var** (variable) και είναι αρκετά έξυπνη ώστε να αποθηκεύει στην μεταβλητή τον κατάλληλο τύπο που αντιπροσωπεύει η δεξιά μεριά της ανάθεσης.

Παράδειγμα ανάθεσης σε μεταβλητή στην Javascript:

```
var package=3;
var hello="world";
```

Στην πρώτη ανάθεση η μεταβλητή package είναι τύπου ακεραίου και στην δεύτερη η μεταβλητή hello είναι τύπου string.

2.3.2.3 Αντικείμενα (Objects)

Ένα αντικείμενο είναι μια τοποθεσία σε μια θέση μνήμης με μια συγκεκριμένη τιμή και μπορούμε να αναφερθούμε σε αυτό μέσω ενός αναγνωριστικού (π.χ μεταβλητή). Ένα αντικείμενο μπορεί να είναι μια μεταβλητή, μια συνάρτηση η μια δομή δεδομένων. Στον αντικειμενοστραφή προγραμματισμό τα αντικείμενα συνήθως

χρησιμοποιούνται για να αντιπροσωπεύουν στιγμιότυπα μιας κλάσης. Μια κλάση είναι μια δομή δεδομένων η οποία περιέχει γνωρίσματα και συναρτήσεις οι οποίες ορίζουν την συμπεριφορά της κλάσης. Παρόλο που η Javascript είναι μια αντικειμενοστραφής γλώσσα δεν έχει κλάσεις. Ωστόσο χρησιμοποιεί **constructors** και είναι αρκετά έξυπνη ώστε να ορίζει μέσα από αυτούς τα αντικείμενα. Οι constructors ουσιαστικά είναι συναρτήσεις οι οποίες παίρνουν ως παραμέτρους τις τιμές που θέλουμε να ορίσουμε για ένα αντικείμενο.

Παράδειγμα δημιουργίας αντικειμένου στην Javascript:

```
function person(firstname,lastname)
{
this.firstname=firstname;
this.lastname=lastname;
}
```

Με την παραπάνω σύνταξη δημιουργούμε ένα αντικείμενο person που έχει για γωρίσματα το firstname και το lastname. Το **this** είναι μια ειδική λέξη των αντικειμενοστραφών γλωσσών και αντιπροσωπεύει το ίδιο το αντικείμενο. Τέλος, η αρχικοποίηση ενός αντικειμένου και η ανάθεση του σε μια μεταβλητή γίνεται ως εξής:

```
var myFather=new person("John","Doe");
```

Όπου John και Doe είναι οι τιμές των γνωρισμάτων.

2.3.3 Advanced Javascript

2.3.3.1 Διαχείρηση γεγονότων

Ένα γεγονός (event) είναι μια δράση (action) που λαμβάνει χώρα στον browser και στην οποία αντιστοιχεί ένα κομμάτι του κώδικα Javascript ή σε ένα HTML script. Ένα event μπορεί να ενεργοποιηθεί όταν ο χρήστης πατήσει το αριστερό κλικ του ποντικιού κατα την συμπλήρωση μιας φόρμας ή και ακόμα όταν σύρουμε το ποντίκι πάνω σε ένα αντικείμενο του browser. Ένας διαχειριστής γεγονότος (event hadler) είναι ένας κώδικας σε Javascript ο οποίος είναι συνδεδεμένος με ένα συγκεκριμένο event που συμβαίνει σε συγκεκριμένα σημεία του browser. Για παράδειγμα ένα event

handler συνδεδεμένο με το ποντίκι μπορεί να ανοίξει ένα καινούργιο παράθυρο στον browser όταν το ποντίκι πατηθεί.

Τα ονόματα των events έχουν δοθεί με τρόπο ώστε να περιγράφουν τις ενέργειες των χρηστών. Παραδείγματα αποτελούν το click, mouseover και submit. Ωστόσο υπάρχουν και events με λιγότερο περιγραφικό όνομα όπως το focus το οποίο ενεργοποιείται όταν ένα αντικείμενο γίνει ανενεργό.

Τα events δεν περιορίζονται μόνο σε αυτά που συμβαίνουν από τις πράξεις των χρηστών. Events όπως το resize που ενεργοποιείται όταν το παράθυρο αλλάζει μέγεθος και το load που ενεργοποιείται όταν το έγγραφο ανανεώνεται, βασίζονται στην συμπεριφορά του browser.

Οι browsers παρέχουν όλες τις απαραίτητες πληροφορίες για κάθε event που συμβαίνει μέσω του αντικειμένου **Event**. Το αντικείμενο αυτό δίνεται ως παράμετρος στον event handler και περιέχει πληροφορίες όπως τις συντεταγμένες της οθόνης στην οποία πατήθηκε το ποντίκι και πολλές άλλες.

Ο ορισμός ενός event σε Javascript μπορεί να είναι ο εξής:

```
window.onload=init;
```

Με την γραμμή αυτή λέμε στο παράθυρο του browser να εκτελέσει την συνάρτηση init όταν το έγγραφο ανανεωθεί.

Η συνάρτηση init είναι η εξής:

```
function init(){
 alert("hello");
}
```

Όταν το παράθυρο ανανεωθεί θα εμφανιστεί το μήνυμα hello.

2.3.3.2 Regular expressions

Μια κανονική έκφραση (regular expression) χρησιμοποιείται για να προσδιορίσει μοτίβα (patterns) για χαρακτήρες. Για παράδειγμα μπορεί να φτιάξει μοτίβα για το email που αποτελείται από το όνομα, το σύμβολο @, τον διακομιστή, μια τελεία και

την χώρα. Η κανονική έκφραση επιτρέπει ακόμα να προσδιορίζει μοτίβα για την εύρεση χαρακτήρων που επαναλαμβάνονται πολλές φορές μέσα σε ένα string ή μοτίβα για την εύρεση μιας ομάδας χαρακτήρων σε συγκεκριμένο σημείο του string. Η κανονική έκφραση συγκρίνεται με ένα string για να εξακριβωθεί εάν αυτό ακολουθεί τους κανόνες ή τα μοτίβα που έχουν δηλωθεί στην έκφραση. Για παράδειγμα μια κανονική έκφραση προσδιορίζει μοτίβα για τους αρθμούς του τηλεφώνου οι οποίοι πρέπει να είναι έγκυροι και μοτίβα για το email. Όταν ο χρήστης συμπληρώσει μια φόρμα και πατήσει submit τα στοιχεία που έδωσε θα ελεγχθούν με τα patterns. Αν κάποιο από αυτά τα στοιχεία δεν ανταποκρίνεται στα μοτίβα των κανονικών εκφράσεων τότε εμφανίζεται ένα μήνυμα λάθους στον χρήστη.

Παράδειγμα δημιουργίας μιας κανονικής έκφρασης:

```
var ex=/http/;

ex.exec("hellohttp");
```

Στην πρώτη γραμμή δημιουργείται μια κανονική έκφραση και δηλώνει ότι αν η συμβολοσειρά http δεν βρεθεί στο string με το οποίο πρόκειται να συγκριθεί θα επιστραφεί false ειδάλλως true. Στην δεύτερη γραμμή συγκρίνεται η κανονική έκφραση με το hellohttp. Η σύγκριση θα επιστρέψει true επειδή το http υπάρχει μέσα στο string.

2.3.3.3 Συναρτήσεις παραλληλίας

Η παραλληλία στις γλώσσες προγραμματισμού είναι αρκετά σημαντική και χρήσιμη διότι με αυτήν επιτυγχάνεται περισσότερη εκτέλεση εργασιών σε λιγότερο χρόνο. Χωρίς την παραλληλία οι Web Servers θα εξυπηρετούσαν τους χρήστες τον έναν μετά τον άλλον προκαλώντας μεγάλη συμφόρηση στο Διαδίκτυο. Έτσι λοιπόν η Javascript έχει υλοποιηθεί ώστε να υποστηρίζει και αυτή παραλληλία παρέχοντας ακόμα μεγαλύτερη ευλεξία στην δημιουργία ιστοσελίδων. Η παραλληλία μπορεί να επιτευχθεί με δύο συναρτήσεις: την setInterval και την setTimeout.

SetTimeout: Η συνάρτηση setTimeout παίρνει δύο παραμέτρους. Η πρώτη παράμετρος είναι η συνάρτηση η οποία πρόκειται να εκτελεσθεί και η δεύτερη παράμετρος είναι ένας ακέραιος που προσδιορίζει το πότε θα εκτελεσθεί η συνάρτηση.

SetInterval: Η συνάρτηση setInterval είναι ακριβώς ίδια με την setTimeout με την μόνη διαφορά ότι η setTimeout θα εκτελεσθεί μια φορά ενώ η setInterval θα εκτελείται για πάντα (εκτός και αν τη διακόψουμε).

Και στην δύο περιπτώσεις όταν το κεντρικό νήμα εκτέλεσης ενεργοποιήσει τις μεθόδους θα συνεχίσει την εκτέλεση του χωρίς να περιμένει πότε θα τερματίσουν. Τέλος, οι μέθοδοι clearTimeout και clearInterval χρησιμοποιούνται για να τερματίσουν τις μεθόδους setTimeout και setInterval αντίστοιχα.

Παράδειγμα εκτέλεσης της setInterval:

```
var myVar=setInterval(function(){myTimer()},1000);

function myTimer()
{
 alert("hello");
}
```

Η setInterval μετά από κάθε ένα δευτερόλεπτο θα εκτελεί την συνάρτηση myTimer. Ιδιαίτερη προσοχή χρειάζεται στην δεύτερη παράμετρο η οποία δίνεται σε milliseconds.

2.3.4 Σχόλια

Τα σχόλια είναι το πιο σημαντικό χαρακτηριστικό που βοηθούν τον προγραμματιστή να κινείται με ευκολία στο αρχείο που γράφει. Τα σχόλια δεν εκτελούνται σαν τον κώδικα, αλλά αποτελούν ένα είδος σημείωσης που περιγράφουν τι γίνεται μέσα σε αυτόν. Επίσης μπορούν να τον χρησιμοποιήσουν για να παρακάμψουν ένα κομμάτι του κώδικα βοηθώντας τον προγραμματιστή στην διαδικασία του debugging[8].

Μιας γραμμής σχόλιο μπορεί να γίνει με δύο μπροστινές καθέτους (/). Οτιδήποτε υπάρχει μετά από αυτές αγνοείται κατά την εκτέλεση.

Παράδειγμα:

```
//This next line prints text into the document
document.write("This line came from some JavaScript");

alert("The text has been printed");
```

[8] http://en.wikipedia.org/wiki/Debugging

Τέλος, είναι δυνατόν να αγνοηθούν περισσότερες της μιας γραμμής ταυτόχρονα. Αυτό επιτυγχάνεται τοποθετώντας στην αρχή μια προστινή κάθετο και έναν αστερισκό και στο τέλος πρώτα έναν αστερίσκο και έπειτα μια προστινή κάθετο. Οτιδήποτε ανάμεσα στα /* .....*/ αγνοείται.
Παράδειγμα:

```
alert("The text has been printed"); /*this line shows an
alert so we know
things worked */
```

2.4 HyperText Transfer Protocol

2.4.1 Εισαγωγή

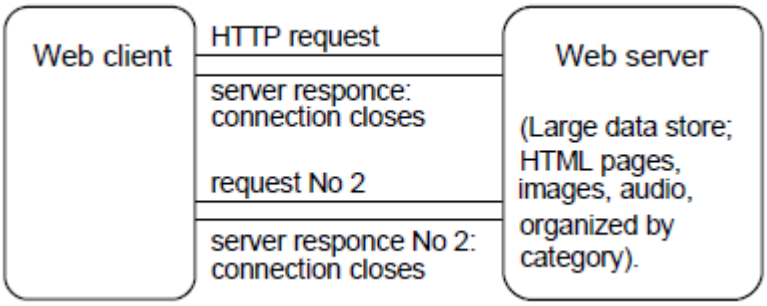
Ασχέτως της γραφικής διεπιφάνειας του χρήστη που παρουσιάζουν οι browsers, το Διαδίκτυο και τα πρωτόκολλα είναι αυτά που διακινούν την πληροφορία στους απομακρυσμένους διακομιστές, οι οποίοι επεξεργάζονται τα requests και επιστρέφουν τα αντίστοιχα μέσα ενημέρωσης. Το HTTP πρωτόκολλο υλοποιήθηκε από τον Tim Berners-Lee το 1990 στον ερευνητικό κέντρο CERN και αποτελεί τον πυρήνα του παγκόσμιου ιστότοπου.

Το HTTP "αποτελεί το βασικό πρωτόκολλο για την ανταλλαγή πληροφορίας στο πλαίσιο του WWW. Είναι ένα ιδιαίτερα ευέλικτο πρωτόκολλο επιπέδου εφαρμογής (application level) που καθορίζει απλές δοσοληψίες μεταξύ του WWW browser και ενός HTTP server. Βασικός στόχος του HTTP είναι η επίτευξη χαμηλών χρόνων απόκρισης (response times). Προς αυτή την κατεύθυνση το HTTP αναπτύχθηκε σαν πρωτόκολλο χωρίς μνήμη (stateless protocol) δηλ. Δεν διατηρεί καμία πληροφορία για μία σύνδεση μετά από την διεκπεραίωση μίας σχετικής αίτησης. Η διατήρηση πληροφορίας κατάστασης μπορεί να επιτευχθεί εκτός από τον ίδιο τον HTTP server μέσω εξωτερικών προγραμμάτων που ακολουθούν το πρωτόκολλο CGI ή βάσεων δεδομένων. Τέλος, το HTTP χαρακτηρίζεται αντικειμενοστρεφές (object oriented protocol). Μπορεί να εφαρμοστεί, με μικρές μετατροπές στις υποστηριζόμενες μεθόδους, σε name servers και κατανεμημένα συστήματα διαχείρισης αντικειμένων" (**D. Martakos, 1999**).

“Πιο ειδικά αναφέρουμε ότι, όπως όλες οι υπηρεσίες του Internet έτσι και η υπηρεσία WWW στηρίζεται στο μοντέλο πελάτη / διανομέα client/server. Αυτό σημαίνει, πως το σύνολο των πληροφοριών βρίσκεται σε κάποιον υπολογιστή που εξυπηρετεί κλήσεις ανάσυρσης. Το πρόγραμμα εξυπηρέτησης των κλήσεων ονομάζεται server, ενώ το πρόγραμμα το οποίο στέλνει τις κλήσεις στον server ονομάζεται client. Έτσι, λοιπόν, στην υπηρεσία WWW ο server ονομάζεται Web server και ο client ονομάζεται Web client ή Web browser” (**ΜΑΡΙΑΝΝΑ ΣΤΟΥΓΙΑΝΝΙΔΟΥ, 2012**).

Το HTTP πρωτόκολλο είναι κατάλληλα διαμορφωμένο για συλλογικά και κατανεμημένα συστήματα υπερμέσων. Οι πληροφορίες που διανέμει μπορεί να είναι ένα απλό κείμενο, εικόνες και υπερκείμενα.

Τα μηνύματα του HTTP είναι παρόμοια με αυτά του FTP (File Transfer Protocol) πρωτοκόλλου και του NNTP (Network News) πρωτοκόλλου. Η βασική τους διαφορά έγκειται στο γεγονός ότι το HTTP πρόκειται για ένα stateless πρωτόκολλο. Ένα stateless πρωτόκολλο σημαίνει με λίγα λόγια ότι δεν μπορεί θυμάται προηγούμενες καταστάσεις, πράγμα το οποίο συμφέρει αφού ο browser του χρήστη επιστρέφει πληροφορίες σύμφωνα με URLS και όχι με προηγούμενες ενέργειες.



Εικόνα 2.4.1.1: Stateless HTTP (D. Martakos, 1999)

“Το πρωτόκολλο HTTP (Hypertext Transfer Protocol) λοιπόν, είναι το σύνολο των κανόνων για την μεταφορά του υπερκειμένου(hypertext) (το οποίο μπορεί να αντιστοιχεί σε αρχεία κειμένου, γραφικών, εικόνας, ήχου, video ή οποιουδήποτε multimedia αρχείου) μέσα στον Παγκόσμιο Ιστό (World Wide Web). Αμέσως μόλις ο χρήστης του Web ανοίξει τον δικό του Web browser, κάνει χρήση του πρωτοκόλλου HTTP. Το HTTP είναι ένα πρωτόκολλο σε επίπεδο εφαρμογής, όπως προαναφέραμε, το οποίο δουλεύει πάνω από το TCP/IP (το θεμελιώδες σύστημα πρωτοκόλλων για το Internet) “(**ΜΑΡΙΑΝΝΑ ΣΤΟΥΓΙΑΝΝΙΔΟΥ, 2012**).

2.4.2 Πεδία επικεφαλίδας HTTP πρωτοκόλλου

Κατά την συναλλαγή μεταξύ του client και του server ανταλλάσσονται κάποια πακέτα πληροφορίας. Τα πακέτα αυτά συνήθως περιέχουν επικεφαλίδες (headers) και άλλες πληροφορίες χωρισμένα με μια γραμμή. Τα headers διαθέτουν πληροφορίες όπως για παράδειγμα κάποιους κωδικούς που περιγράφουν την κατάσταση της απάντησης από έναν server ή παρέχουν πληροφορίες σχετικά με το μηχάνημα του χρήστη. Οι πληροφορίες είναι αυτές είναι χωρισμένες σε πεδία και δίνουν μεγάλη ευελιξία στο HTTP πρωτόκολλο. Για παράδειγμα μπορούν να αναγνωρίζουν τους χρήστες ή να επιτρέπουν την κρυπτογράφηση. "Τα headers αποτελούν μια ομάδα δεδομένων που προηγείται από την πραγματική πληροφορία και συχνά αναφέρονται και ως μεταδεδομένα επειδή έχουν πληροφορίες για την περιγραφή πληροφοριών" (**John Yannakopoulos, 2003**). Τα πεδία ενός header επισυνάπτονται παρακάτω και είναι τα εξής:

- **HTTP ACCEPT (ή Accept header)**

  Πρόκειται για τους MIME (**Multipurpose Internet Mail Extensions**) τύπους που θα δεκτεί ο client σύμφωνα με αυτούς που ορίζει ο header. Κάθε αντικείμενο στην λίστα πρέπει να χωρίζεται με κόμμα.

- **HTTP USER AGENT (ή User-Agent header)**

  Ο browser που χρησιμοποιεί ο client για να στείλει το request

  Ο server απαντάει με τα εξής:

  - Με έναν κωδικό κατάστασης. Οι κωδικοί που δηλώνουν μια κατάσταση είναι προκαθορισμένοι και μπορεί να επιστρέψουν ότι η σελίδα που ζητήθηκε δεν βρέθηκε ή ότι χρειάζεται κάποιο είδος αυθεντικοποίησης για να μπεί ο χρήστης σε μια σελίδα.

  - Με τα πραγματικά δεδομένα που μπορούν να είναι εικόνες, έγγραφα, κείμενα κ.α

  - Με κάποιες πληροφορίες σχετικά με το αντικείμενο που γύρισε από τον server.

- **Content-Type**

  Το πεδίο αυτό χρησιμοποιείται για να περιγράψει το τύπο των δεδομένων που θα δεκτεί ο παραλήπτης. Είναι ιδιαίτερα χρήσιμο για να γνωρίζουν οι browser πως να επεξεργαστούν τα δεδομένα.

- **Date**

  Χρησιμοποιείται για να δηλώσει την ημερομηνία που παράχθηκε το μήνυμα.

- **Expires**

  "Το πεδίο αυτό προσδιορίζει την ημερομηνία/ώρα ύστερα από την οποία η υποδεικνυόμενη οντότητα θα πρέπει να θεωρηθεί παρωχημένη (stale) και μη έγκυρη. Οι εφαρμογές δεν θα πρέπει να εφαρμόζουν caching σε παρόμοια στοιχεία μετά την ημερομηνία αυτή. Η παρουσία του πεδίου δεν σημαίνει κατά ανάγκη ότι η αρχική οντότητα θα πάψει να υπάρχει ή θα αλλάξει μετά την προσδιοριζόμενη ημερομηνία. Η κύρια λειτουργικότητα του πεδίου αφορά στον μηχανισμό caching. Επίσης επιτρέπει στον παροχέα της πληροφορίας να προσδιορίσει το ευμετάβλητο (volatility) των πόρων" (**D. Martakos, 1999**).

- **If-Modified-Since**

  Το πεδίο αυτό χρησιμοποιείται με την μέθοδο GET και δηλώνει ότι εάν η ημερομηνία του πόρου που ζητήθηκε δεν είναι πιο πρόσφατη από την ημερομηνία που έχει καθοριστεί, το request θα ακυρωθεί επιστρέφοντας τον κωδικό κατάστασης 304.

- **Last-Modified**

  "Το πεδίο αυτό υποδεικνύει την ημέρα και ώρα στην οποία ο server πιστεύει ότι υπήρξε μεταβολή του περιεχομένου του πόρου. Ο client αποδέκτης της πληροφορίας την συγκρίνει με το αντίγραφο που ενδεχομένως διατηρεί στην cache του και ενεργεί κατάλληλα" (**D. Martakos, 1999**).

- **Referer**

  Περιέχει την τοποθεσία του ζητούμενου πόρου μέσω ενός URL. Αυτό επιτρέπει στον server διαμορφώσει καταλόγους με αυτά τα URLS διευκολύνοντας του μηχανισμούς της cache.

2.4.3 Μέθοδοι HTTP πρωτοκόλλου

Το HTTP πρωτόκολλο συνολικά υποστηρίζει 8 μεθόδους για την πραγματοποίηση των αιτήσεων. Οι μέθοδοι GET, POST και HEAD που χρησιμοποιούνται και πιο συχνά υποστηρίζονται από τις παλαιότερες εκδόσεις του πρωτοκόλλου. Οι υπόλοιπες καθιερώθηκαν μαζί με την έκδοση 1.1. Οι μέθοδοι παρουσιάζονται παρακάτω.

- **GET**

  Η μέθοδος GET χρησιμοποιείται κυρίως για την αίτηση οποιασδήποτε πληροφορίας που έχει να κάνει με τα μέσα ενημέρωσης. Αυτά μπορεί να είναι είτε ένα κείμενο, είτε εικόνες, βίντεο κτλ. Συνηθισμένες οντότητες μέσα στο HTML έγγραφο που μπορούν να ενεργοποιούν τέτοιου είδους αιτήσεις είναι οι υπερσύνδεσμοι (links). Χαρακτηριστικό αυτής της μεθόδου είναι ότι οι παράμετροι της είναι εμφανίσιμοι. Με λίγα λόγια η χρήση της μεθόδου δεν συνιστάται όταν προσωπικές πληροφορίες πρόκειται να διανεμηθούν (π.χ συμπλήρωση μια φόρμας).

  Παράδειγμα GET μεθόδου:

  http://www.youtube.com/watch?v=XMqdNzCNl5Q. Ζητάμε από τον διακομιστή youtube να μας επιστρέψει το βίντεο με την παράμετρο v=XMqdNzCNl5Q.

  Εκτός από την κανονική μέθοδο GET υπάρχει και η υποσυνθήκη μέθοδος, αν στο header του request συμπεριλαμβάνεται και το πεδίο **If-Modified-Since** . Όπως αναφέραμε και προηγουμένως η αίτηση θα γίνει επιτυχώς μόνο αν η ημερομηνία τροποποίησης του πόρου είναι πιο πρόσφατη από

την ημερομηνία που αναγράφεται στην επικεφαλίδα του αιτήματος. "Η υποσυνθήκη μέθοδος είναι χρήσιμη για να μειώνει τον φόρτο του διαδικτύου και να επιτρέπει σε cached οντότητες να ανανεώνονται χωρίς να χρειάζονται πολλαπλά requests ή να μεταφέρονται άσκοπα μη χρήσιμα δεδομένα" (**John Yannakopoulos, 2003**).

- **HEAD**

Η μέθοδος HEAD είναι παρόμοια με την GET. Η διαφορά τους έγκειται στο γεγονός ότι η μέθοδος HEAD δεν επιστρέφει τα πραγματικά δεδομένα για τα οποία και έγινε η αίτηση παρά μόνο την πληροφορία για αυτά. Η τελευταία λέγεται και ως meta-πληροφορία επειδή περιγράφει άλλη πληροφορία (τα πραγματικά δεδομένα). Το γεγονός ότι δεν επιστρέφει τα πραγματικά δεδομένα την κάνει αρκετά πιο γρήγορη από την GET. Η μέθοδος HEAD χρησιμοποιείται κυρίως για να εξακριβωθεί αν ένας πόρος είναι ελεύθερος και δεν χρειάζεται κάποια αυθεντικοποίηση ή αν ένας πόρος έχει υποστεί μεταβολές πρόσφατα.

- **POST**

Η μέθοδος POST χρησιμοποιείται για την υποβολή δεδομένων πληροφορίας σε μια βάση δεδομένων. Παραδείγματα μπορεί να αποτελέσουν:

- Αποστολή μηνύματος μέσω του facebook ή mail μέσα από το ταχυδρομείο.
- Το submit μετά της συμπλήρωσης μια φόρμας.

Γενικά οτιδήποτε που μπορεί να προσθέσει δεδομένα σε μια βάση δεδομένων χρησιμοποιεί την μέθοδο POST. Σε αντίθεση με την μέθοδο GET που αναφέραμε προηγουμένως η POST δεν φανερώνει τις παραμέτρους της πάνω στο URL αλλά τις τοποθετεί στο σώμα του πακέτου που φεύγει από τον client, προστατεύοντας τις προσωπικές του πληροφορίες.

Παράδειγμα POST μεθόδου

POST /~ιτ20923/cgi-bin/post-query HTTP/1.0

Accept: text/html,video/mpeg,image/gif,application/postscript
User-Agent: Lynx/2.8.4 libwww/5.4.0
From: it20923@hua.gr
Content-Type: application/x-www-form-urlencoded
Content-Length: 150
* a blank line *
org=Distributed%20Systems&professor=george&browsers=lynx

Η τελευταία γραμμή είναι η πληροφορία που περνάει στον server για να την διαχειριστεί.

"Οι αιτήσεις POST μπορεί να μην έχουν σαν αποτέλεσμα πόρους που είναι προσπελάσιμοι σε μελλοντική αναφορά (αντιπροσωπεύονται από κάποιο URI). Το αποτέλεσμα των POST αιτήσεων (αναφορικά με τους πόρους που ενδεχομένως διαμορφώθηκαν) περιγράφεται στα success/error codes τα οποία επιστρέφονται από τον server" (**D. Martakos, 1999**).

Οι επόμενες μέθοδοι χρησιμοποιούνται πιο σπάνια και υποστηρίζονται από την 1.1 του HTTP πρωτοκόλλου. Είναι οι εξής:

- **PUT**

  Είναι παρόμοια με την μέθοδο POST και χρησιμοποιείται για να ανεβάζει και αυτή δεδομένα σε κάποιο server. Η διαφορά τους έγκειται στην χρήση του URL. Η μέθοδος POST χρησιμοποιεί το URL για να υποδείξει τον πόρο που θα χειριστεί τα δεδομένα ενώ η μέθοδος PUT χρησιμοποιεί το URL για να υποδείξει σε ποιό πόρο θα ανεβούν (upload) τα δεδομένα.

- **DELETE**

  Πραγματοποιεί την διαγραφή ενός περιεχομένου (π.χ αρχείου) από κάποιον server (συνήθως μέσω ftp).

- **TRACE**

  Πληροφορεί τον client σχετικά με του διακομιστές που επεξεργάστηκαν το request πριν αυτό φτάσει στον τελικό προορισμό του.

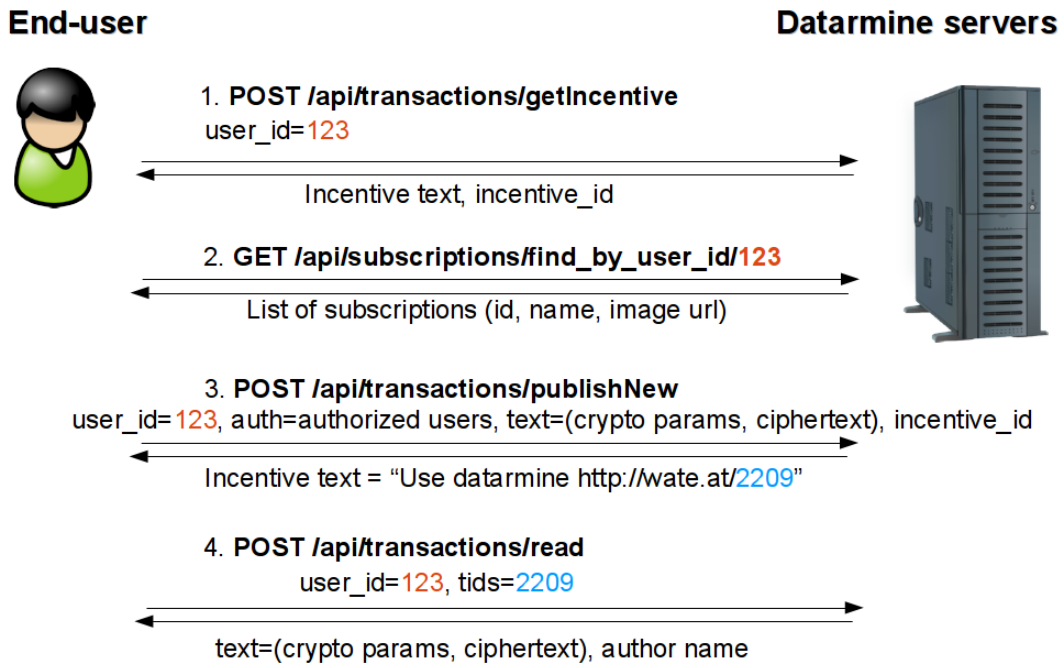
- **OPTIONS**

  Γυρίζει όλες τις HTTP μεθόδους που υποστηρίζει ο απομακρυσμένος διακομιστής.

- **CONNECT**

  Χρησιμοποιείται μόνο για proxy που υποστηρίζει SSL tunnel.

Στην εικόνα 2.4.3.1 δίνεται ένα παράδειγμα λειτουργίας των μεθόδων GET και POST.



Εικόνα 2.4.3.1: Παράδειγμα λειτουργίας των μεθόδων GET και POST (Πηγή: fortinet)

2.4.4 Λειτουργία HTTP πρωτοκόλλου

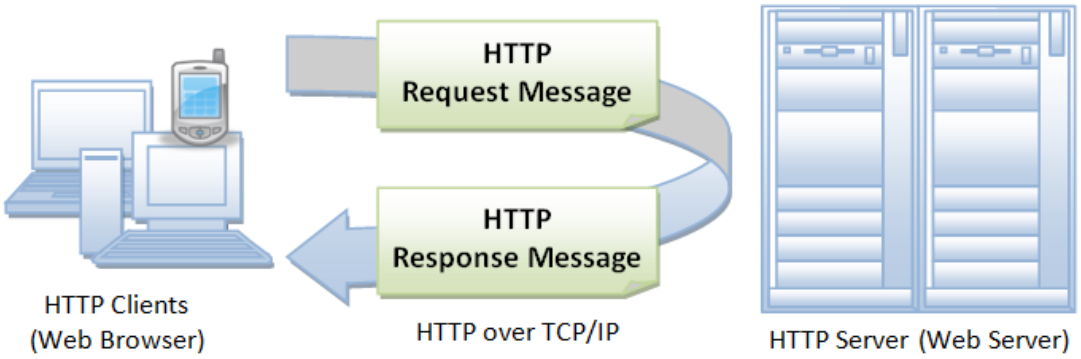
Όπως αναφέραμε και προηγουμένως το HTTP πρωτόκολλο στηρίζεται στην σχέση μεταξύ ενός client και server. Στην πιο απλή μορφή του πρωτοκόλλου εγκαθιστάται μια σύνδεση μεταξύ των δύο αυτών οντοτήτων (μέσω TCP πρωτοκόλλου) και ανταλλάσσονται requests/responses αντίστοιχα.

Ο client στέλνει αιτήματα που περιέχουν τα εξής:

- Την μέθοδο που προσδιορίζει τον τρόπο με τον οποίο ο client θέλει να αλληλεπιδράσει με τον server. "Η χρήση του όρου μέθοδος οφείλεται στον αντικειμενοστρεφή προσανατολισμό του πρωτοκόλλου" (**D. Martakos, 1999**).
- Ένα Universal Resource Identifier (URI) που προσδιορίζει τον πόρο με τον οποίο ο client θέλει να αλληλεπιδράσει.
- Την έκδοση του πρωτοκόλλου.
- MIME τύπους που περιέχουν πληροφορίες σχετικά με τον client κ.α.

Ο server στέλνει απαντήσεις που περιέχουν τα εξής:

- Ένα κωδικό κατάστασης που δηλώνει αν η αίτηση ενός client έγινε με επιτυχία.
- Μεταπληροφορία
- Τα πραγματικά δεδομένα.

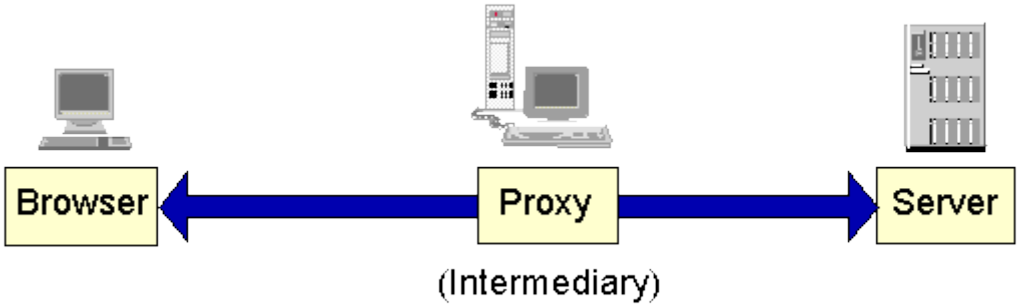


Εικόνα 2.4.4.1: Ανταλλαγή αιτημάτων μεταξύ Client και Server (Πηγή: ntu.edu.sg)

Σε μια περισσότερο σύνθετη μορφή η HTTP επικοινωνία περιλαμβάνει περισσότερους από έναν server ανάμεσα στον client και στον τελικό server οι οποίοι ονομάζονται ενδιάμεσοι (**intermediaries**). Αυτοί εμφανίζονται σε τρεις μορφές: proxy,gateway και tunnel.

"Ενας proxy ενδιάμεσος αποτελεί έναν πράκτορα προώθησης (forwarding agent) ο οποίος δέχεται αιτήσεις για κάποιο URI σε απόλυτη μορφή (absolute form), ανασκευάζει τα σχετικά μηνύματα μεταβάλλοντας όλα τα συστατικά τμήματα τους και τα προωθεί στον server ο οποίος προσδιορίζεται από το URI. Ενας gateway

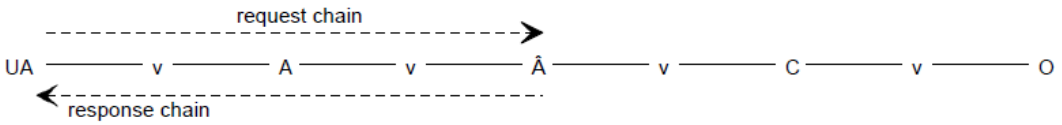
ενδιάμεσος αποτελεί ένα πράκτορα παραλαβής (receiving agent) ο οποίος τοποθετείται στο αμέσως υψηλότερο επίπεδο από ορισμένους servers και μεταφράζει τις αιτήσεις στο πρωτόκολλο που οι servers αυτοί αντιλαμβάνονται και μπορούν να ερμηνεύσουν. Ενας tunnel ενδιάμεσος λειτουργεί ως σημείο μεταγωγής (relay point) μεταξύ δύο συνδέσεων χωρίς να παρεμβαίνει στο περιεχόμενο των μηνυμάτων. Tunnels χρησιμοποιούνται όταν η επικοινωνία HTTP θα πρέπει να διέλθει από ενδιάμεσους όπως π.χ. firewalls" (**D. Martakos, 1999**).



Εικόνα 2.4.4.2: Παράδειγμα ενδιάμεσου server (Πηγή: wwwconference.org)

Στην εικόνα 2.4.4.2 βλέπουμε ένα παράδειγμα ενδιαμέσου. Ο proxy αποτελεί έναν ενδιάμεσο ο οποίος μεταφέρει το αίτημα στον origin server. Ο ενδιάμεσος μπορεί να δέχεται ταυτόχρονα και άλλες αιτήσεις από άλλους client και ταυτόχρονα να τις αποστέλνει σε άλλους origin servers.

Οποιοσδήποτε ενδιάμεσος που δεν λειτουργεί ως tunnel μπορεί να αναπτύξει μια εσωτερική cache μνήμη για την επεξεργασία των αιτήσεων. Η μνήμη αυτή είναι ιδιαίτερα χρήσιμη για δίνει γρήγορες απαντήσεις σε αιτήματα τα οποία έχουν ήδη απαντηθεί, μειώνοντας την ουρά των requests και των responses. Απαραίτητη προυπόθεση για την λειτουργία αυτή είναι οι cached απαντήσεις να μην έχουν αλλάξει από τον origin server. Σε αυτή την περίπτωση το αίτημα θα φτάσει μέχρι τον origin server για να πάρει την απάντηση.



Εικόνα 2.4.4.3: HTTP επικοινωνία με ενδιάμεσους (D. Martakos, 1999)

Η εικόνα 2.4.4.3 αποτελεί παράδειγμα cached απάνησης. Ο ενδιάμεσος Β διαθέτει την απάντηση που θέλει ο client και την γυρνάει πίσω.

Όπως επισημάνθηκε και προηγουμένως, η επικοινωνία του HTTP πρωτοκόλλου στηρίζεται στην TCP/IP σύνδεση. Η TCP σύνδεση συνήθως ακούει στην θύρα 80 αλλά δεν αποκλείται να χρησιμοποιηθούν και άλλες. Επίσης η επικοινωνία του HTTP μπορεί να επιτευχθεί πάνω από άλλα πρωτόκολλα του διαδικτύου. Τέλος. το HTTP διακρίνεται για την αξιόπιστη μεταφορά δεδομένων.

3 ΜΟΝΤΕΛΟΠΟΙΗΣΗ ΑΠΕΙΛΩΝ ΙΔΙΩΤΙΚΌΤΗΤΑΣ ΧΡΉΣΤΗ ΚΑΙ ΕΠΙΘΕΣΕΩΝ ΣΕ SERVERS

Κύριος στόχος της παρούσας εργασίας είναι η μελέτη και η εύρεση κατάλληλων μηχανισμών για την προστασία της ιδιωτικότητας του χρήστη και των Web Servers. Στον παρόν κεφάλαιο παρουσιάζονται ορισμένοι μηχανισμοί και μοντέλα που επιδιώκουν την προστασία της ιδιωτικότητας. Τα μοντέλα αυτά έχουν δημιουργηθεί και λειτουργούν με βάση την γνώση του προστατευμένου πόρου. Ωστόσο αυτή η γνώση δεν είναι πάντοτε επαρκής για την πρόβλεψη όλων των πιθανών ανωμαλιών στις μελλοντικές συμπεριφορές του υπολογιστικών πόρων.

Για να γίνει ακόμα περισσότερο κατανοητός ο τρόπος που λειτουργούν τα μοντέλα δίνεται ο παρακάτω ορισμός:

Μοντέλο: Ένα μοντέλο είναι ένα σύνολο από διαδικασίες που χρησιμοποιούνται για να αποδίδουν μια πιθανότητα στην τιμή ενός χαρακτηριστικού. Η πιθανότητα αυτή αντιπροσωπεύει την φυσιολογική «συμπεριφορά» του χαρακτηριστικού (στην φάση της εκπαίδευσης). Όταν μια πιθανότητα είναι κοντά στο μηδέν, υποδηλώνει πιθανή ανώμαλη συμπεριφορά (στην φάση της ανίχνευσης). Ένα μοντέλο χωρίζεται συνήθως σε δύο φάσεις: φάση της εκπαίδευσης (training phase) και φάση της ανίχνευσης (detection phase). Κατα την φάση της εκπαίδευσης συγκεντρώνονται όλες οι απαραίτητες πληροφορίες που αφορούν την φυσιολογική «συμπεριφορά» των χαρακτηριστών και εφαρμόζονται κατώφλια για να ξεχωρίζει από την ανώμαλη συμπεριφορά. Τέλος, κατά την φάση της ανίχνευσης γίνεται σύγκριση μεταξύ των φυσιολογικών κατωφλίων που συγκεντρώθηκαν στην φάση της εκπαίδευσης με τις τιμές που προέκυψαν σε αυτήν την φάση. Σε περίπτωση που οι τιμές ξεπερνούν τα φυσιολογικά κατώφλια πρόκειται για πιθανή ανώμαλη δραστηριότητα.

3.1 Μηχανισμοί & Μοντέλα

Όπως αναφέραμε σε προηγούμενα κεφάλαια το HTTP πρωτόκολλο αποτελεί το στόχαστρο πολλών online επθέσεων. Σύμφωνα με τους Christopher Kruegel & Giovanni Vigna (2003) είναι δυνατόν να αποφευχθούν τέτοιου είδους επιθέσεις με την εφαρμογή μοντέλων ανίχνευσης ανώμαλης συμπεριφοράς. Η έρευνά τους βασίστηκε κυρίως σε HTTP requests που χρησιμοποιούν την μέθοδο GET και σε requests που έχουν παραμέτρους με τιμές. Οι Christopher Kruegel & Giovanni Vigna (2003) κατάφεραν να αποδείξουν ότι με την εφαρμογή των μοντέλων σε μια ή σε περισσότερες παραμέτρους ενός HTTP request είναι δυνατόν να βρεθούν συμπεριφορές οι οποίες να αποκλίνουν από τις φυσιολογικές. Για να πετύχουν αυτές τις αποκλίσεις χρησιμοποίησαν τον παρακάτω τύπο:

$$\text{Anomaly Score} = \sum_{m \in \text{Models}} w_m * (1 - p_m) \quad (1)$$

Όπου p_m είναι η πιθανότητα που επιστρέφει ένα μοντέλο, όπου w_m είναι το βάρος του μοντέλου και όπου m είναι ο αριθμός των μοντέλων που εφαρμοστήκαν σε μια ή σε περισσότερες παραμέτρους ενός request. Τέλος, προστίθενται όλα τα scores για να προκύψει ένα συνολικό score που αντιπροσωπεύει είτε την κάθε παράμετρο ξεχωριστά είτε ολόκληρο το request μαζί. Στον τύπο (1) όσο το p_m είναι πιο κοντά στο μηδέν τόσο αυξάνεται και η πιθανότητα μια παράμετρος (ή όλο το request) να παρουσιάζει ανώμαλη συμπεριφορά.

Οι Christopher Kruegel & Giovanni Vigna (2003) παρουσιάζουν τα εξής μοντέλα για την ανίχνευση ανώμαλης συμπεριφοράς:

- **Attribute Length**

  Το μοντέλο αυτό χρησιμοποιείται για να ανιχνεύει καταστάσεις στις οποίες ο επιτιθέμενος προσπαθεί να γεμίσει τον buffer της εφαρμογής στην οποία κάνει την επίθεση. Το μοντέλο «μαθαίνει» τα φυσιολογικά μεγέθη των παραμέτρων κατά την φάση της εκπαίδευσης και τα συγκρίνει με αυτά που θα προκύψουν κατά την φάση της ανίχνευσης. Πιο συγκεκριμένα, υπολογίζεται ο μέσος όρος (mean) μ και η διακύμανση (variance) σ^2 των τιμών των παραμέτρων των requests που μαζεύτηκαν στην φάση της εκπαίδευσης. Στην συνέχεια εφαρμόζοντας τον τύπο:

$$p(|x-\mu| > |l-\mu|) < p(l) = \frac{\sigma^2}{(l-\mu)^2} \quad (2)$$

βγάζουμε τις πιθανότητες που εκπροσωπούν τις τιμές των παραμέτρων. Όπου l είναι το μέγεθος (length) της τιμής μιας παραμέτρου.

Στην φάση της ανίχνευσης, σε κάθε request που ελέγχεται τοποθετείται το μέγεθος της τιμής της κάθε παραμέτρου στον τύπο (2) με μέση τιμή και διακύμανση που υπολογίστηκε στην φάση της εκπαίδευσης. Αν ο τύπος (2) επιστρέφει πιθανότητα κοντά στο μηδέν τότε σημαίνει ότι έχουμε πιθανή ανώμαλη συμπεριφορά.

- **Attribute Presence or Absence**

Ο ρόλος του μοντέλου είναι να ανιχνεύει για παραμέτρους που ενδεχομένως να λείπουν από ένα request στο οποίο έπρεπε να υπάρχουν και το ίδιο ισχύει αντίθετα. Πιο συγκεκριμένα κατά την φάση της εκπαίδευσης το μοντέλο αρχίζει και μαθάινει για κάθε πρόγραμμα (π.χ το login.php) ξεχωριστά όλες τις πιθανές παραμέτρους που μπορεί να έχει καθώς και τον συνολικό αριθμός τους. Με αυτόν τον τρόπο θα μπορεί να ανιχνεύσει αν κάποιο request ξεφεύγει από τα φυσιολογικά πλαίσια. Το μοντέλο αυτό επειδή αναφέρεται σε όλες τις παραμέτρους μαζί σε ένα request επιστρέφει μηδέν αν πρόκειται για ανώμαλη συμπεριφορά και ένα σε περίπτωση φυσιολογικής. Αν και το μοντελό αυτό φαίνεται να αποδίδει καλά ωστόσο δεν μπορεί να αντιμετωπίσει ορισμένες επιθέσεις.

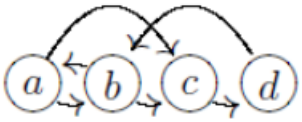
Για παράδειγμα έστω ότι έχουμε το παρακάτω request:

http://www.otinanai.com/home.php?a=12&a=32&a=54

και ας υποθέσουμε ότι το μοντέλο έχει βρει ότι οι συνολικές παραμέτρους για αυτό το πρόγραμμα είναι τρείς και η μια από αυτές είναι η παράμετρος a. Το μοντελό δεν θα ανιχνεύσει καμία ανώμαλη συμπεριφορά επειδή καταλαβαίνει ότι ο αριθμός των παραμέτρων είναι ο σωστός και ότι η παράμετρος a ανήκει σε αυτό το πρόγραμμα.

- **Attribute Order**

  Το μοντέλο αυτό έρχεται να καλύψει το παραπάνω πρόβλημα. Ο ρόλος του είναι να ελέγχει την σειρά των παραμέτρων. Το μοντέλο κατά την φάση της εκπαίδευσης «μαθαίνει» ποιές παράμετροι συνδέονται μεταξύ τους δημιουργώντας κάτι σαν μια αλυσίδα ή ένα γραφήμα.



Εικόνα 3.1.1: Παράδειγμα αλυσίδας παραμέτρων (Gustavo Miguel Barroso Assis do Nascimento, 2010)

  Σε περίπτωση που το γράφημα ξεφεύγει από το φυσιολογικό που υπολογίστηκε στην φάση της εκπαίδευσης επιστρέφεται μηδέν ειδάλλως επιστρέφεται ένα. Συνήθως τα μοντέλα **Attribute Presence or Absence** και **Attribute Order** χρησιμοποιούνται μαζί για προφανείς λόγους.

Οι Christopher Kruegel & Giovanni Vigna έστρεψαν την έρευνά τους σε επιθέσεις που γίνονται αυτή την φορά μέσω του κώδικα της Javascript. Σε συνεργασία με τον Marco Cova (2010) υποστήριξαν ότι ελέγχοντας τα φυσιολογικά χαρακτηριστικά του κώδικα της Javascript (όπως για παράδειγμα την τιμή μιας παραμέτρου σε ένα tag που περιέχει ένα URL προς ένα plugin) μέσω μοντέλων ανίχνευσης ανώμαλης συμπεριφοράς είναι δυνατόν ένα σύστημα να αναγνωρίζει κακόβουλες επιθέσεις. Οι Christopher Kruegel, Giovanni Vigna & Marco Cova (2010) χρησιμοποίησαν ακριβώς τα ίδια μοντέλα που αναφέρθηκαν προηγουμένως προσθέτοντας ένα επιπλέον.

Το επιπλέον μοντέλο ακολουθεί την ίδια ακριβώς λογική με τα προηγούμενα και είναι το εξής:

- **Type Learner**

  Το μοντέλο αυτό χρησιμοποιείται για να εξακριβώσει αν οι τύποι ορισμένων παραμέτρων είναι αυτοί που είχαν εξακριβωθεί και στην φάση της εκπαίδευσης. Για παράδειγμα μια παράμετρος που έχει για τιμή ένα

URL και στην φάση της ανίχνευσης βρεθεί ότι έχει τιμή έναν ακέραιο το μοντέλο θα το σημειώσει ως μια ανώμαλη συμπεριφορά. Για την καλύτερη λειτουργία του μοντέλου, όλοι οι πιθανοί τύποι που το μοντέλο μπορεί να ανιχνεύσει στην φάση της εκπαίδευσης χωρίζονται σε πέντε κατηγορίες:

1. Ακεραίους μικρότερους η ίσους από το 1024.
2. Μεγάλου ακεραίους.
3. String που περιέχουν path σε αρχείο
4. Αναγνώσιμους χαρακτήρες.
5. URLS.

Σε περίπτωση που ανιχνευτεί ένας τύπος που δεν ταιριάζει με αυτόν που ανιχνεύθηκε στην εκπαίδευση το μοντέλο επιστρέφει μηδέν ειδάλλως επιστρέφει ένα.

Να σημειώθει ότι και εδώ όλα τα scores από τα μοντέλα προστίθενται στον τύπο (1) για να βρεθεί το συνολικό anomaly score μιας ιστοσελίδας.

Ένας άλλος τρόπος παραβιάσης της ιδιωτικότητας των χρηστών και των servers είναι μέσω του λεγόμενου ***script injection***. Σύμφωνα με τους Trevor Jim, Nikhil Swamy & Michael Hicks (2007) κομμάτια κώδικα σε Javascript με ανώμαλη συμπεριφορά τοποθετούνται κυρίως σε ιστοσελίδες που έχουν πλούσιο περιεχόμενο (π.χ εικόνες κτλ). Όταν ο χρήστης μπεί σε μια τέτοια σελίδα το κακόβουλο script θα φορτωθεί και θα εκτελεστεί στον browser του. Τα script αυτά είναι ικανά διαβάζουν προσωπικές πληροφορίες του χρήστη μέσω των cookies και οποιαδήποτε άλλη πληροφορία σημαντική για την ιστοσελίδα. Επίσης μπορούν να απενεργοποιούν μηχανισμούς του browser που υπάρχουν για να προστατεύουν τον χρήστη από άλλες online επιθέσεις.

Προκειμένου λοιπόν να αποφεύγονται τέτοιου είδους καταστάσεις οι Trevor Jim, Nikhil Swamy & Michael Hicks (2007) εφάρμοσαν τις Browser-Enforced Embedded Policies (BEEP). Οι πολιτικές αυτές εφαρμόζονται μέσω ενός ***security hook.*** Το security hook είναι μια συνάρτηση σε κώδικα Javascript και είναι ενσωματωμένη σε κάθε σελίδα που περιλαμβάνει ένα Web Site.

Γενικά όταν ο browser φορτώνει μια ιστοσελίδα γωρίζει πολύ καλά πότε πρόκειται να εκτελέστει ένα script. Το security hook δέχεται από τον browser ένα script και ελέγχει αν είναι κακόβουλο ή όχι σύμφωνα με κάποιες πολιτίκες. Προκειμένου να είναι δυνατή η επικοινωνία μεταξύ του security hook και του browser, ο τελευταίος έχει τροποποιηθεί ειδικά για τον σκοπό αυτόν.

Απαραίτητο για την λειτουργία του security hook είναι να μην έχει προηγηθεί η εκτέλεση άλλων script πριν αυτό προλάβει να τα ελέγξει. Για τον λόγο αυτόν τοποθετείται πάντα πρώτο στο tag <header>. Αυτή η τεχνική διασφαλίζει ότι ακόμα και τα κακόβουλα scripts που προσπαθούν να αλλάξουν το hook θα εκτελεστούν αφού θα έχει εκτελεστεί πρώτα αυτό.

Η συνάρτηση που υλοποιεί το security hook λέγεται ***afterParseHook*** και παίρνει δύο παραμέτρους: η μία είναι το κείμενο του script που ελέγχεται εκείνη την στιγμή και η άλλη είναι το DOM στοιχείο που περικλείει το script. Για παράδειγμα έχουμε τον παρακάτω κώδικα:

```
<body onload="alert('hello')"> ... </body>
```

Η συνάρτηση θα πάρει ως πρώτη παράμετρο το 'alert('hello')' και ως δεύτερη το στοιχείο <body>.

Οι Trevor Jim, Nikhil Swamy & Michael Hicks (2007) στηρίχτηκαν στις εξής δύο πολιτκές:

- **WhiteList**

  Επειδή κάθε δημιουργός μιας ιστοσελίδας γνωρίζει επακριβώς ποια scripts θα εκτελεστούν δημιουργεί μια whitelist. Η λίστα αυτή περιέχει όλα τα scripts που είναι γνωστά για την ιστοσελίδα και δεν δημιουργούν κανένα πρόβλημα. Έτσι ο browser όταν στείλει στο security hook ένα script η συνάρτηση afterParserHook θα ελέγξει εάν υπάρχει μέσα στο whitelist. Σε περίπτωση που υπάρχει η συνάρτηση γυρνάει true ειδάλλως επιστρέφει false και το script δεν εκτελείται. Μια πιθανή υλοποίηση μπορεί να είναι η παρακάτω:

```
if (window.JSSecurity) {
  JSSecurity.afterParseHook =
    function(code, elt) {
      if (whitelist[SHA1(code)]) return true;
      else return false;
    };
```

Όπου code είναι το κείμενο του script και όπου elt το DOM element.

- **DOM sandboxing**

Σε αυτήν την πολιτική γίνεται σάρωση όλου του DOM του εγγράφου της ιστοσελίδας. Οι Trevor Jim, Nikhil Swamy & Michael Hicks (2007) υποστηρίζουν ότι συνήθως τα κακόβουλα scripts περιέχονται μέσα σε elements του τύπου και με το ειδικό αναγνωριστικό "noexecute". Έτσι το security hook αυτήν την φορά παίρνει ως παράμετρο μόνο ένα DOM element και το κάνει parse μέχρι να φτάσει στο root στοιχείο. Εάν μέχρι και το root στοιχείο έχει βρεθεί ένα στοιχείο που περιέχει το αναγνωριστικό "noexecute" η συνάρτηση θα επιστρέψει false ειδάλλως θα επιστρέψει true.

Επειδή η παραπάνω στρατηγική είναι πολυ απλή, είναι εύκολο να παρακαμφθεί από τους επιτιθέμενους απλά τοποθετώντας το injected script εκτός των στοιχείων και . Το πρόβλημα μπορεί να διορθωθεί εύκολα κωδικοποιώντας κάθε script που φαίνεται κακόβουλο, σε string της Javascript και στην συνέχεια τοποθετώντας σε HTML μορφή στο έγγραφο.

```
<div class="noexecute" id="n5"></div>
<script>
  document.getElementById("n5").innerHTML =
            "quoted possibly-malicious content"
</script>
```

Στον παραπάνω κώδικα το string που περιέχει τον κακόβουλο κώδικα τοποθετείται με την μέθοδο innerHTML στο στοιχείο . Όταν ο browser κάνει parse την σελίδα θα διαβάσει το string ως ένα στοιχείο της HTML εμποδίζοντας το script να εκτελεστεί.

Οι Collin Jackson, Andrew Bortz, Dan Boneh & John C Mitchell (2006) έστρεψαν την προσοχή τους σε επιθέσεις που γίνονται μέσω της cache μνήμης και των συνδέσμων του browser. Πιο συγκεκριμένα η cache μνήμη χρησιμοποιείται για να ενισχύει την απόδοση του περιηγητή διατηρώντας ορισμένες πολύτιμες πληροφορίες για ένα συγκεκριμένο site χωρίς αυτές να είναι κρυμμένες από άλλα, με αποτέλεσμα να αποτελεί το στόχαστρο πολλών online επιθέσεων. Ένα άλλο χαρακτηριστικό των browsers είναι τα **visited links**. Οι σύνδεσμοι αυτοί χρησιμοποιούνται για να διευκολύνουν την περιήγηση του χρήστη δείχνοντάς τους συνδέσμους με διαφορετικά χρώματα. Αυτό σημαίνει ότι σύνδεσμοι με χρώμα διαφορετικό του μπλέ (είναι το default χρώμα) έχουν ήδη επισκεφτεί από τον χρήστη. Η λογική αυτή παρουσιάζει μεγάλα κενά ασφάλειας καθώς ο επιτιθέμενος μπορεί μέσων αυτών να παρακολουθήσει την συμπεριφορά του χρήστη και να τοποθετήσει κακόβουλα περιεχόμενα στα visited links.

Οι Collin Jackson, Andrew Bortz, Dan Boneh & John C Mitchell (2006) έφτιαξαν μια αρχή (**Same-origin Principle**) σύμφωνα με την οποία μόνο τα site που μπορούν να αποθηκεύουν πληροφορίες στον browser μπορούν να τις διαβάσουν και να τις τροποποιήσουν. Με άλλα λόγια σκοπός του **Same-origin Principle** είναι να περιορίσει την πρόσβαση στα δεδομένα του browser μόνο στα site τα οποία και τα έγραψαν. Έτσι λοιπόν σύμφωνα με αυτήν την αρχή, όσες ιστοσελίδες που επισκέφτεται ο browser (π.χ οι cache υπηρεσίες) δεν υπακούν στις απαιτήσεις (policies) του **Same-origin Principle** δεν θα έχουν το δικαίωμα να έχουν πρόσβαση στο state (δεδομένα π.χ τα cookies) του browser.

Οι Collin Jackson, Andrew Bortz, Dan Boneh & John C Mitchell (2006) δημιούργησαν ένα extension για τον Mozilla με τις εξής δύο προσεγγίσεις:

- **Cached Data Policy**

  Σε αυτήν την πολιτική το extension χρησιμοποιεί δύο κύριους observers για την καταγραφή δεδομένων στην cache μνήμη. Ο ένας observer έχει πρόσβαση στα περιεχόμενα του site που φορτώνονται στον browser και ο άλλος έχει πρόσβαση στο όνομα του host που φορτώνεται. Κατά την διαδικασία αποθήκευσης περιεχομένου στην cache μνήμη, ο observer μπορεί να γνωρίζει ποιά μέρη του περιεχομένου αποθηκεύτηκαν (μέσω κάποιων headers της cache). Έτσι όταν το ίδιο site επιχειρήσει ξανά να

φορτώσει το ίδιο περιεχόμενο, αυτό θα φορτωθεί από την cache μνήμη. Στην περίπτωση όμως που ένα διαφορετικό site επιχειρήσει να φορτώσει το ίδιο περιεχόμενο δεν θα χρησιμοποιηθεί η μνήμη cache. Αυτό συμβαίνει επειδή ο observer (που είναι υπεύθυνος για τα περιεχόμενα των site) δεν παρατήρησε καμία πληροφορία να αποθηκεύεται στην μνήμη cache για αυτό το συγκεκριμένο site όπως και έγινε προηγουμένως. Επομένως, σε αυτήν την περίπτωση που ένα διαφορετικό site επιχειρήσει να φορτώσει ένα περιεχόμενο δημιουργείται μια καινούργια θέση στην cache μνήμη με παρόμοια πληροφορία δηλαδή (περιεχόμενο site, host site).

Στην υλοποίηση των Collin Jackson, Andrew Bortz, Dan Boneh & John C Mitchell το extension απενεργοποιεί τις cache υπηρεσίες του browser και λειτουργεί ως ενδιάμεσος. Όταν τα cookies είναι απενεργοποιημένα τότε απενεργοποιείται και η λειτουργία της cache για το συγκεκριμένο site. Σε περίπτωση που χρησιμοποιούνται cookies για ένα session, η cache ενεργοποιείται. Όταν το session λήξει η cache καθαρίζει ότι πληροφορία είχε μαζέψει και απενεργοποιείται. Εάν τα **third-party** cookies δεν επιτρέπονται, τότε δεν επιτρέπεται ούτε και το caching για αυτά, το ίδιο ισχύει και αντίθετα. Τέλος, όταν ο χρηστής διαγράψει τα cookies για ένα site, διαγράφονται για το συγκεκριμένο site τα πάντα και από την cache.

- **Visited links Policy**

  Το extension χρησιμοποιεί και εδώ observers οι οποίοι έχουν πρόσβαση στο όνομα του referrer host και στο όνομα του host του συνδέσμου που ο χρήστης μόλις επισκέφτηκε. Και οι δύο αυτοί host μπορούν να έχουν πρόσβαση στο state του browser και κανένας άλλος. Έτσι αν ένας σύνδεσμος που βρίσκεται σε μια σελίδα Α δείχνει σε μια σελίδα Β που έχει ήδη επισκεφτεί ο χρήστης, ο σύνδεσμος Α δεν θα πρέπει να έχει δεκτεί επίσκεψη (δηλαδή να έχει χρώμα διαφορετικό του default). Σε διαφορετική περίπτωση θα ισχύουν τα παρακάτω:

  1. Το site της σελίδας Α θα έχει δικαίωμα στο state του browser
  2. Η σελίδα Α θα ανήκει στο ίδιο site με την σελίδα Β

Το extension προσπαθεί μέσα από τις προτιμήσεις του user (που αποθηκεύονται συνήθως σε cookies) να εφαρμόσει την κατάλληλη πολιτική για αυτά. Τέλος, όταν ο χρήστης διαγράψει τα cookies για ένα site αμέσως το extension δηλώνει την είσοδο τους στο history log του browser ως μη χρησιμοποιημένη. Το τελευταίο χρησιμεύει για την διάκριση μεταξύ των απλών συνδέσμων με τους συνδέσμους που έχουν ήδη επισκεφτεί.

4 ΑΝΑΠΤΥΞΗ ΣΥΣΤΗΜΑΤΟΣ ΑΝΙΧΝΕΥΣΗΣ ΑΝΩΜΑΛΩΝ ΣΥΜΠΕΡΙΦΟΡΩΝ

Στο παρόν κεφάλαιο περιγράφεται η υλοποίηση ενός συστήματος, βασισμένο στα μοντέλα που αναφέρθηκαν στο προηγούμενο κεφάλαιο. Πιο συγκεκριμένα βασιστήκαμε στα μοντέλα που ανέδειξαν οι Christopher Kruegel, Giovanni Vigna & Marco Cova με σκόπο να δημιουργήσουμε μια πιλοτική εφαρμογή η οποία θα παρέχει όχι μόνο στους χρήστες αλλά και στις ιστοσελίδες που εισέρχονται, προστασία απέναντι από κακόβουλες επιθέσεις που έχουν σκοπό να παραβιάσουν την ιδιωτικότητά τους.

Στην παρούσα πτυχιακή έγινε προσπάθεια να συνδυάσουμε ορισμένα μοντέλα ανίχνευσης ανώμαλης συμπεριφοράς με κάποιους άλλους μηχανισμούς προστασίας της ιδιωτικότητας σε ένα add-on (extension). Ένα extension ή αλλιώς add-on αποτελεί μια ενίσχυση των δυνατοτήτων ενός browser. Ουσιαστικά πρόκειται για ένα πρόγραμμα ενσωματωμένο σε ένα άλλο πρόγραμμα τα οποία μπορούν και ανταλλάσουν πληροφορίες. Έτσι λοιπόν με την χρήση των add-ons οι χρήστες μπορούν να επιλέγουν τις επιπλέον λειτουργίες που θέλουν να εκτελεί ο browser «διακοσμώντας» τον ανάλογα με τις προτιμήσεις τους. Επομένως, μέσω ενός τέτοιου add-on θέλαμε να ενισχύσουμε την ασφάλεια των χρηστών.

Αρχικά, σκόπος μας ήταν να δημιουργήσουμε ένα σύστημα IDS το οποίο θα λειτουργούσε για μεγάλα κοινωνικά site όπως για παράδειγμα το Facebook. Στην πορεία όμως της έρευνας μας αντιμετωπίσαμε αρκετά εμπόδια, ένα από αυτά ήταν η κρυπτογράφηση. Τα περισότερα site συνήθως χρησιμοποιούν το πρωτοκόλλο HTTPS το οποίο είναι περίπου ίδιο με τον HTTP πρωτόκολλο αλλά με την μόνη διαφορά ότι τα requests στο HTTPS πρωτόκολλο είναι κρυπτογραφημένα. Το γεγονός αυτό καθιστά απρόβλεπτη την λειτουργία ενός συστήματος που έχει στηριχτεί αποκλειστικά στα μοντέλα ανίχνευσης συμπεριφοράς. Αυτό οφείλεται στο γεγονός ότι τα μοντέλα αυτά χρησιμοποιούν τις τιμές των παραμέτρων για να καταγράψουν την φυσιολογική συμπεριφορά τους και κατά την εκτέλεση να ελέγχουν αν αποκλίνουν από την φυσιολογική. Με την κρυπτογράφηση οι τιμές είναι συνήθως τυχαίου μεγέθους και αγνώστου τύπου πράγμα το οποίο ακυρώνει την όλη λογική

των μοντέλων. Τελικά στραφήκαμε σε site που χρησιμοποιούν μόνο το HTTP πρωτόκολλο.

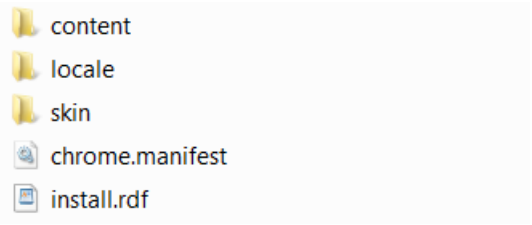
Επομένως παρακάτω, γίνεται λόγος για το σύστημα που δημιουργήθηκε και την χρήση των εργαλείων που χρησιμοποιήθηκαν. Στην συνέχεια περιγράφεται με αναλυτικό τρόπο η λειτουργία του συστήματος δηλαδή το σύνολο των διαδικασιών που λαμβάνουν χώρα κατά την ενεργοποίηση και κατά τον χειρισμό συστήματος. Επίσης δίνεται μια αναλυτική παρουσίαση των διάφορων components που συμβάλουν στην λειτουργία του συστήματος. Τέλος, προκειμένου να εξεταστούν εις βάθος τα μοντέλα, περιγράφεται η εκτελέση ενός πειράματος και εξηγούνται οι παραδοχές που λήφθηκαν υπόψη.

4.1 Τεχνική Υλοποίηση

Για να γίνει πιο κατανοητή η δημιουργία και η λειτουργία του extension θα περιγράφεται κάθε βήμα αναλυτικά.

4.1.1 Δημιουργία Φακέλων του Extension

Όπως αναφέραμε και στο δεύτερο κεφάλαιο, ένα extension για να εγκατασταθεί και να λειτουργήσει θα πρέπει πρώτα να δημιουργηθούν οι κατάλληλοι φάκελοι. Ο φάκελος chrome περιέχει όλους τους απαραίτητους υποφακέλους με τα αντίστοιχα αρχεία τους που θα διαβαστούν από τον browser. Ο root φάκελος θα πρέπει να λέγεται chrome για να αναγνωρίζεται αυτόματα από τον browser όταν αυτός εγκατασταθεί (αν και στις πιο πρόσφατες εκδόσεις του browser χρειάζονται μόνο οι υποφάκελοι).



Εικόνα 4.1.1.1: Φάκελοι που περιγράφουν την λειτουργία του extension

Ο φάκελος **content** περιέχει συνήθως όλα τα xul αρχεία. Τα xul αρχεία όπως αναφέραμε νωρίτερα χρησιμοποιούνται για να περιγράφουν την γραφική διεπιφάνεια του χρήστη πάνω στο extension ή να δημιουργούν άλλα παράθυρα στο browser. Επίσης στον φάκελο content μπορεί να υπάρχουν και JS αρχεία που χρησιμοποιούνται για να προσδίδουν λειτουργικότητα στη γραφική διεπιφάνεια.

Ο φάκελος **locale** περιλαμβάνει τα DTD αρχεία. Τα αρχεία αυτά περιέχουν σταθερές που μπορούν να χρησιμοποιηθούν σε οποιαδήποτε αρχεία που υπάρχουν μέσα στον chrome.

Ο φάκελος **skin** περιλαμβάνει τα όλα CSS αρχεία που χρησιμοποιούνται για την μορφοποίηση του extension. Η μορφοποίηση μπορεί να περιλαμβάνει εικόνες πάνω σε κουμπιά, αλλαγή χρώματος του browser και πολλά άλλα.

Προκειμένου τα xul αρχεία να μπορούν να χρησιμοποιήσουν τα JS, τα DTD και CSS αρχεία και να αποτυπώσουν την λειτουργία τους στην γραφική επιφάνεια του χρήστη θα πρέπει πρώτα να γνωρίζει που να τα βρεί. Αυτόν τον ρόλο έρχεται να διαμορφώσει το **chrome.manifest** αρχείο. Η σύνταξη του αρχείου είναι συγκεκριμένη και περιέχει URLS σε όλα τα αρχεία του chrome, βοηθώντας τον browser να τα βρίσκει εύκολα.

```
content   captainsafetydetection                content/
skin      captainsafetydetection  classic/1.0   skin/
locale    captainsafetydetection  en-US         locale/en-US/

overlay chrome://browser/content/browser.xul  chrome://captainsafetydetection/content/browserOverlay.xul
```

Εικόνα 4.1.1.2: Σύνταξη του chrome.manifest αρχείου

Με τις τρείς πρώτες γραμμές δηλώνουμε τα paths των αρχείων που βρίσκονται μέσα στους φακέλους. Άρα έχουμε ότι τα αρχεία του φακέλου **content** βρίσκονται στο path chrome://something/content/filename. Όπου chrome είναι ο root φάκελος, όπου something είναι ένα οποιοδήποτε όνομα (στην περίπτωσή μας το όνομα του extension) και όπου το filename είναι το όνομα του αρχείου που θέλουμε να χρησιμοποιήσουμε. Το ίδιο ισχύει και για τους άλλους φακέλους. Τέλος, στην τελευταία γραμμή μέσω της εντολής **overlay** δηλώνουμε ότι το αρχείο browser.xul (που είναι το default αρχείο του browser) θα αντικατασταθεί με το browserOverlay.xul αρχείο. Το browserOverlay αρχείο είναι αυτό που περιέχει όλη

την γραφική διεπιφάνεια και του λέμε να αντικαταστήσει το αρχικό αρχείο προκειμένου αυτή να είναι εμφανίσιμη στον χρήστη.

```
<?xml-stylesheet type="text/css" href="chrome://global/skin/" ?>
<?xml-stylesheet type="text/css"
  href="chrome://captainsafetydetection/skin/browserOverlay.css" ?>

<!DOCTYPE overlay SYSTEM
  "chrome://captainsafetydetection/locale/browserOverlay.dtd">

<overlay id="captainsafetydetection-browser-overlay"
  xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">

<script type="application/x-javascript"
    src="chrome://captainsafetydetection/content/browserOverlay.js" />
```

Εικόνα 4.1.1.3: Το browserOverlay xul αρχείο

Η εικόνα 4.1.1.3 αποτελεί ένα παράδειγμα ενσωμάτωσης αρχείων του chrome του extension μέσω URL. Ωστόσο θα μπορούσε κάποιος να μην χρησιμοποιήση τα URLS και να τοποθετήσει απευθείας τον κώδικα των αρχείων. Το τελευταίο, αν και είναι η πιο γρήγορη λύση δεν αποτελεί και την πιο αποδοτική από άποψη εκτέλεσης και χρόνου.

Για να εγκατασταθεί το extension στον browser χρειάζεται το αρχείο **install.rdf.** Το αρχείο αυτό περιέχει σημαντικές πληροφορίες σχετικά με την εφαρμογή και είναι ορατές από τους χρήστες. Οι πληροφορίες αυτές μπορεί να είναι το όνομα της εφαρμογής, η έκδοση της που είναι απαραίτητη για τον έλεγχο της συμβατότητας με τον browser κ.α.

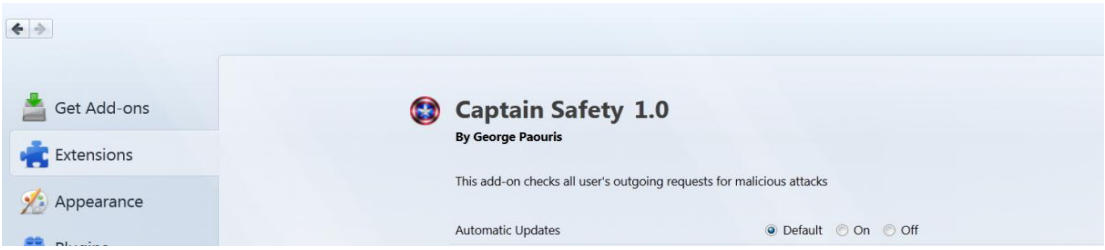
```
<?xml version="1.0"?>
<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:em="http://www.mozilla.org/2004/em-rdf#">

      <Description about="urn:mozilla:install-manifest">
            <em:id>captainsafetydetection@paouris.com</em:id>
            <em:unpack>true</em:unpack>
            <em:name>Captain Safety</em:name>
            <em:version>1.0</em:version>
            <em:type>2</em:type>
            <em:creator>George Paouris</em:creator>
            <em:description>This add-on checks all user's
            outgoing requests for malicious attacks
            data</em:description>
            <em:iconURL>
            chrome://captainsafetydetection/
            skin/captain_america_shield_32x32.png</em:iconURL>
```

```
            <em:targetApplication>
                  <Description>
                        <em:id>{ec8030f7-c20a-464f-9b0e-
                         13a3a9e97384}</em:id>
                        <em:minVersion>18.0</em:minVersion>
                        <em:maxVersion>23.*</em:maxVersion>
                  </Description>
            </em:targetApplication>
      </Description>
</RDF>
```

Ο παραπάνω κώδικας περιγράφει τις πληροφορίες του extension. Για παράδειγμα το `<em:description>` χρησιμοποιείται για να περιγράψει τον σκοπό της εφαρμογής, το `<em:creator>` παρουσιάζει τον δημιουργό της εφαρμογής και το `<em:iconURL>` απλά εμφανίζει το logo της εφαρμογής μέσω ενός URL του chrome. Τέλος, το `<em:minVersion>` και το `<em:maxVersion>` δηλώνουν την ελάχιστη και μέγιστη έκδοση αντίστοιχα του browser στον οποίο μπορεί να λειτουργήσει η εφαρμογή. Η έκδοση συνήθως περιορίζεται από την έκδοση των υπηρεσιών που παρέχει ο browser στους προγραμματιστές.

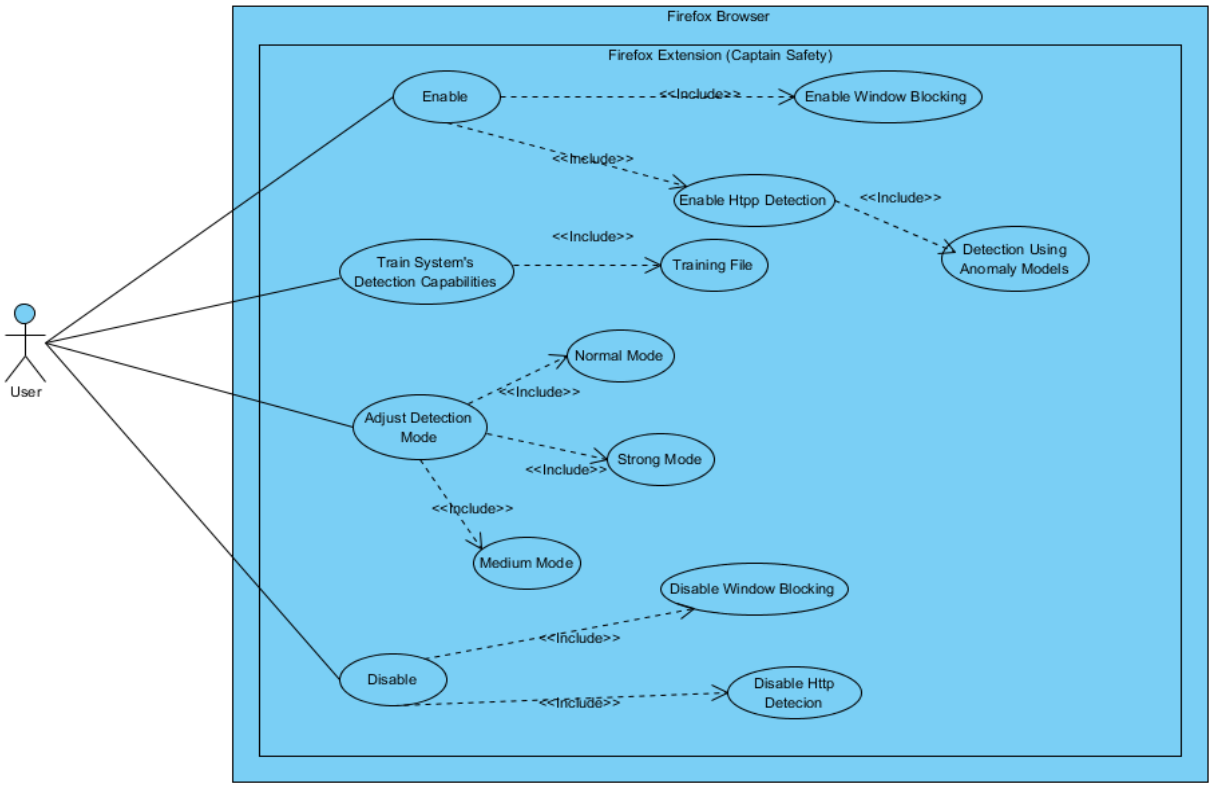
Το αποτέλεσμα του παραπάνω κώδικα όπως φαίνεται στον Firefox browser είναι το εξής:



Εικόνα 4.1.1.4: Παράδειγμα οθόνης χαρακτηριστών του extension

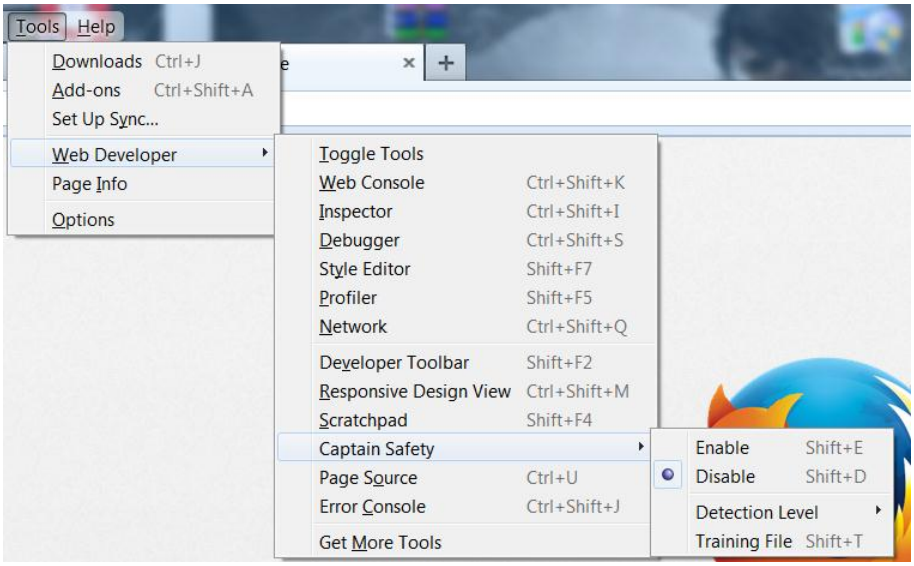
4.1.2 Βασικά στοιχεία (Components) του Extension

Στο παρακάτω σχήμα περιγράφονται τα διάφορα μέρη της εφαρμογής με τα οποία μπορεί ο χρήστης να αλληλεπιδρά.



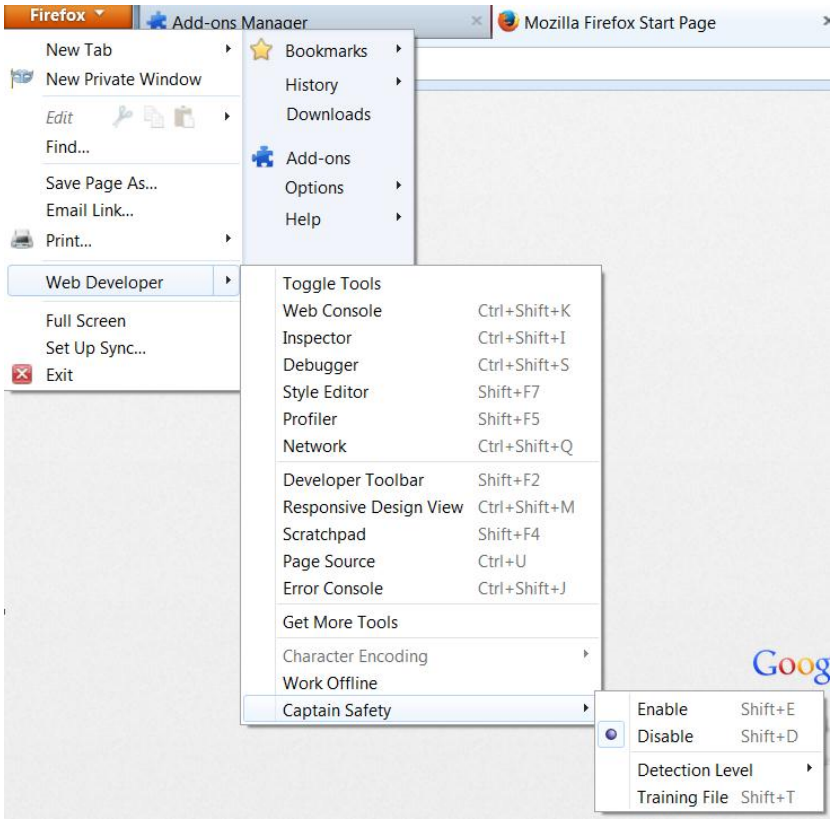
Εικόνα 4.1.2.1: Βασικά μέρη του extension

Όπως παρουσιάζεται και στην εικόνα 4.2.2.1 η εφαρμογή είναι ένα μικρό σύστημα μέσα στον περιηγητή με τον οποίο και αλληλεπιδρά. Όταν το extension εγκατασταθεί εμφανίζεται ένα menu επιλογών της εφαρμογής στο menu του Mozilla. Όταν το menu bar του Firefox είναι ενεργοποιημένο το menu επιλογών του extension παρουσιάζεται ως εξής:



Εικόνα 4.1.2.2: Menu επιλογών του extension όταν το menu του Firefox είναι απενεργοποιημένο

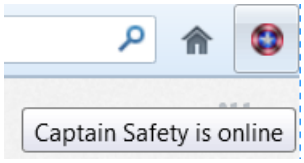
Αντίθετα όταν το menu bar είναι απενεργοποιημένο το menu επιλογών μπορεί να βρεθεί ως εξής:



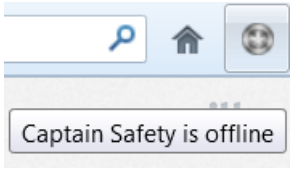
Εικόνα 4.1.2.3: Menu επιλογών του extension όταν το menu του Firefox είναι ενεργοποιημένο

Σύμφωνα με την εικόνα 4.1.2.2 ο χρήστης μπορεί να ενεργοποιήσει και να απενεργοποιήσει την εφαρμογή όποτε αυτός θελήσει. Οι ενέργειες αυτές μπορούν να γίνουν με διάφορους τρόπους:

1. Πατώντας τα κουμπιά Disable/Enable (εικόνα 4.1.2.2).
2. Πατώντας Shift+E για ενεργοποίηση ή Shift+D για απενεργοποίηση.
3. Πατώντας το εικονίδιο που βρίσκεται στο toolbar του Firefox.



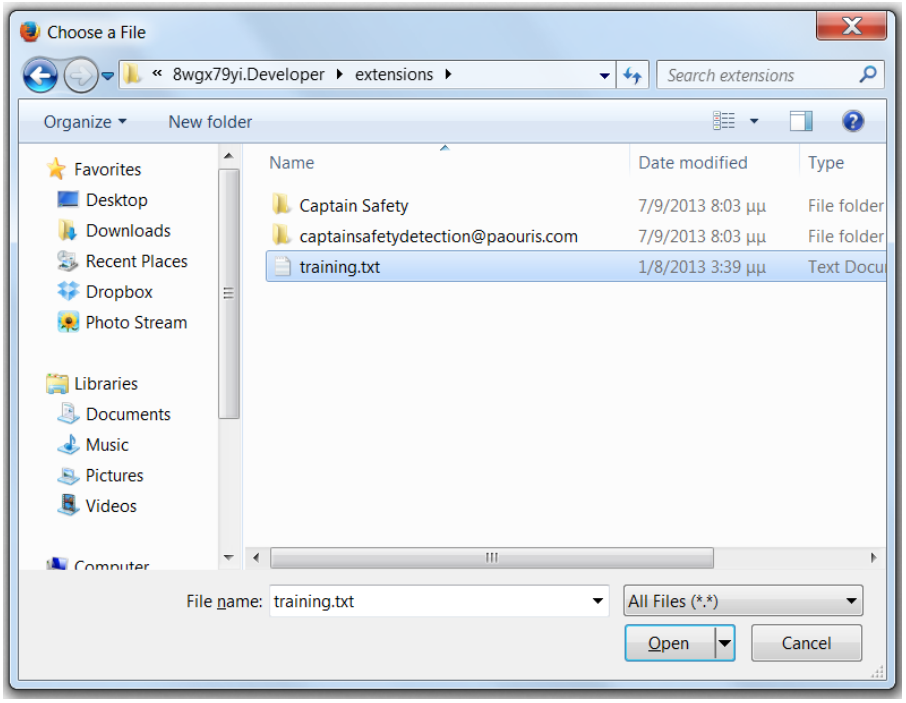
Εικόνα 4.1.2.4: Περίπτωση απενεργοποίησης του extension



Εικόνα 4.1.2.5: Περίπτωση ενεργοποίησης του extension

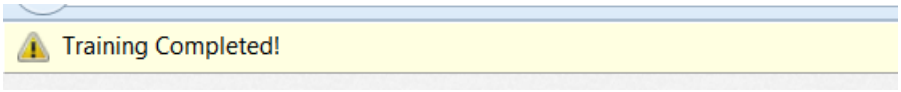
Η ενεργοποίηση/απενεργοποίηση συμβάλλει και στην ενεργοποίηση/απενεργοποίηση άλλων λειτουργιών όπως το **window blocking** και το http detection. Οι λειτουργίες αυτές θα περιγραφούν αναλυτικά στο αμέσως επόμενο τμήμα αυτού του κεφαλαίου.

Ο χρήστης επίσης έχει την δυνατότητα να «προπονεί» την εφαρμογή μέσω ενός **training** αρχείου. Το αρχείο αυτό έχει πολλή μεγάλη σημασία για την λειτουργία της εφαρμογής όσο αφορά το κομμάτι του HTTP detection. Το αρχείο περιέχει όλα τα «καλά» HTTP requests ενός site (από το log του server) από τα οποία θέλουμε να βγάλουμε την φυσιολογική συμπεριφορά και να συγκρίνουμε με τα requests που φεύγουν από τον χρήστη κατά την περιήγησή του στο site. Ο χρήστης μπορεί να φορτώσει το αρχείο πατώντας την επιλογή Training File (ή Shift+T) που φαίνεται στην εικόνα 4.1.2.6.



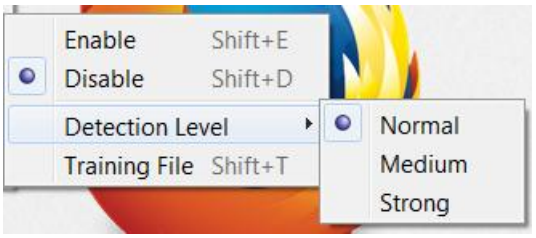
Εικόνα 4.1.2.6: Παράθυρο φόρτωσης του training αρχείου

Αφού το αρχείο φορτωθεί και πάνε όλα καλά εμφανίζεται το εξής μήνυμα:



Εικόνα 4.1.2.7: Αποτέλεσμα ολοκλήρωσης φόρτωσης του αρχείου

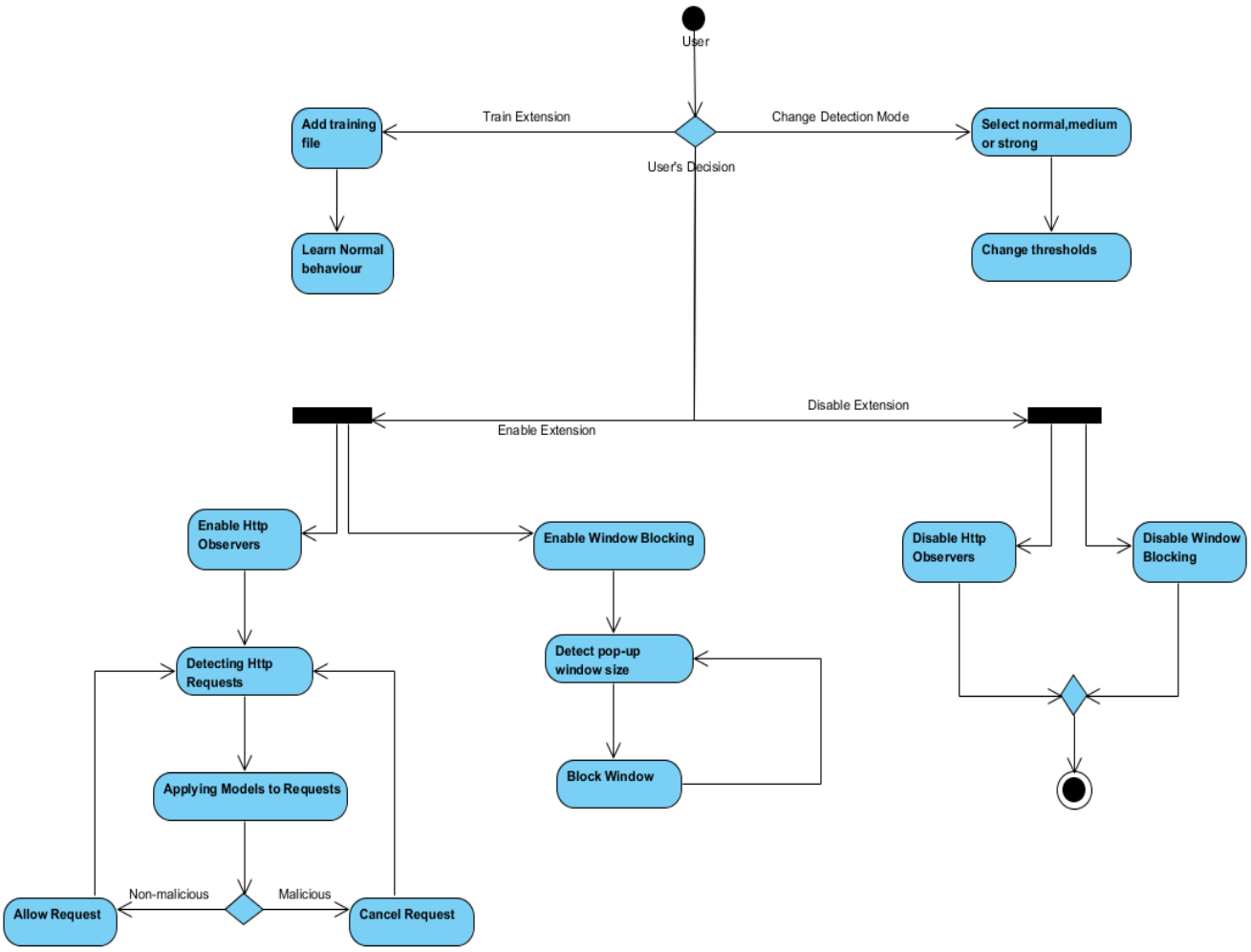
Τέλος, ο χρήστης μπορεί να ορίσει και το επίπεδο ανίχνευσης της εφαρμογής. Τα επίπεδα φαίνοται στην παρακάτω εικόνα.



Εικόνα 4.1.2.8: Menu επιλογής για το επίπεδο της ανίχνευσης

4.1.3 Ροή εκτέλεσης και λειτουργίες του Extension

Στο παρακάτω διάγραμμα παρουσιάζεται η ροή εκτέλεσης της εφαρμογής:



Εικόνα 4.1.3.1: Σχεδιάγραμμα ροής εκτέλεσης του extension

4.1.3.1 Η λειτουργία της «προπόνησης»

Όπως αναφέραμε και νωρίτερα ο χρήστης μπορεί χρησιμοποιεί ένα αρχείο για «προπονεί» την εφαρμογή. Το αρχείο αυτό (βλ. παράρτημα Α) περιέχει όλα τα φυσιολογικά requests ενός site. Όταν λοιπόν ο χρήστης πατήσει για να φορτώσει το αρχείο εκτελείται μια συνάρτηση σε κώδικα Javascript. Η συνάρτηση αυτή κάνει

parse τα HTTP requests του αρχείου για να βγάλει τη φυσιολογική συμπεριφορά του μαζεύοντας μέσους όρους, διακυμάνσεις και οτιδήποτε άλλο χρειάζεται για τα μοντέλα ανίχνευσης ανώμαλης συμπεριφοράς. Τα δεδομένα που μαζεύονται είναι χωρισμένα ανά πρόγραμμα (π.χ login.php) και αποθηκεύονται σε έναν πίνακα (array) για περαιτέρω χρήση. Για περισσότερες πληροφορίες για την συνάρτηση, ο κώδικας είναι ανεβασμένος στο github.com[9].

Ένα επιπλέον χαρακτηριστικό της λειτουργίας του training είναι ότι δεν χρειάζεται να φορτώνει ο χρήστης κάθε φορά το αρχείο. Μετά την πρώτη φορά που φορτώνεται ατο αρχείο η εφαρμογή είναι αρκετά έξυπνη έτσι ώστε να το βρίσκει αυτόματα. Κάθε φορά που ο χρήστης θα κλείνει και θα ανοίγει τον browser, το αρχείο θα φορτώνεται αυτόματα. Αυτός ο μηχανισμός επιτυγχάνεται εύκολα με την χρήση των **Preferences**. Τα **Preferences** του Firefox χρησιμοποιούνται για την τοπική αποθήκευση διάφορων δεδομένων, και παρέχονται από τις υπηρεσίες του **XPCOM**.

Το **XPCOM** είναι μια πλατφόρμα που λειτουργεί σε κατώτερο επίπεδο από τον Firefox και παρέχει objects. Τα διάφορα μέρη του μπορούν να χρησιμοποιηθούν και να υλοποιηθούν από αρκετές γλώσσες όπως την JavaScript**,** Java**,** Python**,** Perl και Ruby. Το **XPCOM** μπορεί επίσης να παρέχει διάφορα components όπως κλάσεις, νήματα, πίνακες και πολλά άλλα.

Ο πιο εύκολος τρόπος να ζητήσουμε την υπηρεσία των **Preferences** από το **XPCOM** φαίνεται στον παρακάτω κώδικα:

```
this.prefs = Components.classes["@mozilla.org/preferences-service;1"]
           .getService(Components.interfaces.nsIPrefService).getBranch("accessibility.");
```

Στο αντικείμενο prefs αποθηκεύουμε το path του αρχείου σε μια συγκεκριμένη τοποθεσία των **Preferences** με τον εξής τρόπο:

```
myExtension.prefs.setCharPref("training",path);
```

και για να αποκτήσουμε αργότερα το path του αρχείου από τα **Preferences** κάνουμε το εξής:

```
var path=myExtension.prefs.getCharPref("training");
```

[9] https://github.com/paourisi/Captain-Safety.

4.1.3.2 Η λειτουργία των Observers

Όταν ο χρήστης ενεργοποιήσει την εφαρμογή ενεργοποιούνται ταυτόχρονα και τα **Observers**. Τα **Observers** είναι αντικείμενα (objects) που μπορούν και παρατηρούν διάφορα γεγονότα που συμβαίνουν κατά την λειτουργία του browser (κάτι σαν τα events της Javascript). Μερικά γεγονότα όπως το κλείσιμο του browser, το άνοιγμα ενός καινούργιου tab είναι μερικά παραδείγματα που μπορούν τα **Observers** να ανιχνεύουν. Όπως τα **Preferences** έτσι και τα **Observers** είναι υπηρεσίες που παρέχονται από την **XPCOM** πλατφόρμα. Η χρήση των **Observers** είναι ιδιαίτερα σημαντική για την λειτουργία της εφαρμογής. Ο ρόλος τους είναι να ανιχνεύουν κάθε HTTP request που φεύγει από τον browser του χρήστη.

Η διαδικασία για την δημιουργία και την καταχώρηση ενός **Observer** που ανιχνεύει HTTP requests φαίνεται παρακάτω:

```
register: function() {
  var observerService = Components.classes["@mozilla.org/observer-service;1"]
                        .getService(Components.interfaces.nsIObserverService);
  observerService.addObserver(this, "http-on-modify-request", false);

},
```

παρόμοια για την διαγραφή ενός **Observer** που ανιχνεύει HTTP requests φαίνεται παρακάτω:

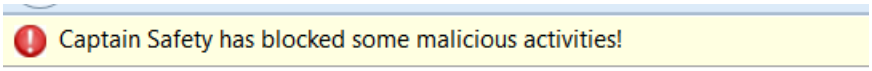
```
unregister: function() {
  var observerService = Components.classes["@mozilla.org/observer-service;1"]
                        .getService(Components.interfaces.nsIObserverService);
  observerService.removeObserver(this, "http-on-modify-request");
}
```

Για τον χειρισμό των γεγονότων που ανιχνεύουν οι **Observers** ορίζεται και μια συνάρτηση η οποία εκτελείται κάθε φορά που πραγματοποιείται ένα request. Μέσα στην συνάρτηση αυτή γίνονται οι εξής έλεγχοι:

- Αν τα request είναι από τον host για τον οποίο η εφαρμογή έχει «προπονηθεί».
- Αν τα request έχουν παραμέτρους ή όχι.

- Αν τα request είναι φυσιολογικά σύμφωνα με τα μοντέλα που έχουν εφαρμοστεί (δηλαδή δεν ξεπερνάνε τα anomaly scores που βρέθηκαν στην φάση της εκπαίδευσης).

Αν κάθε request περάσει επιτυχώς από τους παραπάνω ελέγχους τότε η εφαρμογή επιτρέπει στον χρήστη να περιηγείται στο site. Σε διαφορετική περίπτωση δεν επιτρέπει στον χρήστη να εκτελέσει τις ενέργειες που προκάλεσαν τα requests, παρουσιάζοντας του και σχετικό μήνυμα στο επάνω μέρος του browser.



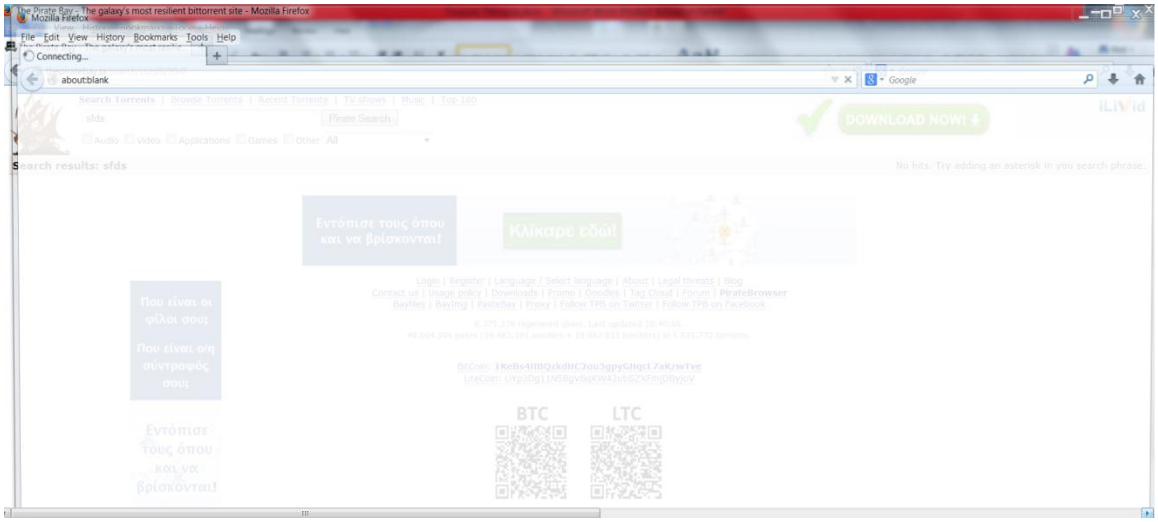
Εικόνα 4.1.3.2.1: Παράδειγμα μηνύματος όταν το extension μπλοκάρει κακόβουλα requests

Για περισσότερες πληροφορίες για τους Observers, ο κώδικας είναι ανεβασμένος στο ανεβασμένος στο github.com.

4.1.3.3 Η λειτουργία του Window Blocking

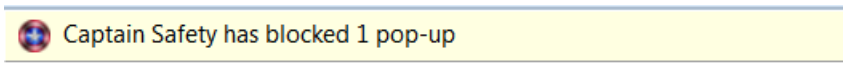
Η δεύτερη λειτουργία που ενεργοποιείται μαζί με τα **Observers** είναι το **window blocking**. Όπως είχαμε αναφέρει και σε προηγούμενα κεφάλαια τα pop-ups παράθυρα μπορεί αποτελούν κακόβουλες δραστηριότητες οι οποίες εκτελούνται χωρίς την θέληση του χρήστη. Η λειτουργία του **window blocking** είναι τελείως ανεξάρτητη από τα μοντέλα, ωστόσο στοχεύουν και αυτά στην προστασία από της ιδιωτικότητας του χρήστη.

Η λογική της λειτουργίας του **window blocking** είναι απλή. Αν κάθε παράθυρο που εμφανίζεται έχει διαφορετικό μέγεθος από το κανονικό παράθυρο του browser τότε θεωρείται pop-up και μπλοκάρεται αμέσως. Ωστόσο υπάρχουν και παράθυρα τα οποία αν και είναι τύπου pop-up ανοίγουν με την θέληση του χρήστη. Για παράδειγμα σε μερικά site, η επιλογή για login ανοίγει ένα καινούργιο παράθυρο στο οποίο ο χρήστης πρέπει να συμπληρώσει τα στοιχεία του. Σε αυτήν την περίπτωση ο χρήστης θα πρέπει να απενεργοποιήσει την εφαρμογή (με μια από τις μεθόδους που αναφέρθηκαν νωρίτερα) για να ανοίξει κανονικά το παράθυρο.



Εικόνα 4.1.3.3.1: Παράδειγμα ενός malicious pop-up φαίνεται

Όταν η εφαρμογή καταφέρει να μπλοκάρει ένα παράθυρο εμφανίζει το εξή μήνυμα στον χρήστη:



Εικόνα 4.1.3.3.2: Παράδειγμα μηνύματος αποτροπής ενός pop-up

Για περισσότερες πληροφορίες για το **window blocking** ο κώδικας, είναι ανεβασμένος στο ανεβασμένος στο [github.com]

4.1.3.4 Η λειτουργία του επιπέδου ανίχνευσης

Είχαμε αναφέρει σε προηγούμενα κεφάλαια ότι το κάθε μοντέλο ανώμαλης συμπεριφοράς έχει ένα anomaly score το οποίο αν ξεπεραστεί θα σημαίνει ότι έχουμε κακόβουλη δραστηριότητα. Στην λειτουργία αυτή το anomaly score αυξάνεται ή μειώνεται επί ένα ποσοστό αλλάζοντας έτσι την ευαισθησία της ανίχνευσης. Περισσότερες λεπτομέρειες για το επίπεδο ανίχνευσης θα αναφερθούν στο αμέσως επόμενο τμήμα του κεφαλαίου.

4.2 Πείραμα

Για την εφαρμογή των μοντέλων ανώμαλης συμπεριφοράς πραγματοποιήθηκε ένα πείραμα στην ιστοσελίδα in.gr. Το in.gr είναι μια απλή ιστοσελίδα κοινωνικού περιεχομένου που χρησιμοποιεί το HTTP πρωτόκολλο. Σκόπος μας ήταν να αποδείξουμε πως το extension θα μπορούσε να λειτουργήσει κανονικά σε οποιαδήποτε ιστοσελίδα προσφέροντας ταυτόχρονα και την απαραίτητη προστασία στον χρήστη και στην ιστοσελίδα.

Πριν προχωρήσουμε στην ανάλυση του πειράματος πρέπει να επισημανθεί ότι το in.gr αν και λειτουργεί με το HTTP πρωτόκολλο, σε ορισμένες παραμέτρους χρησιμοποιεί κρυπτογράφηση. Ιδανικά δεν θα έπρεπε να έχει, ωστόσο αυτό δεν εμπόδισε την σωστή λειτουργία των μοντέλων.

Για την πραγματοποίηση του πειράματος έπρεπε πρώτα να φτιάξουμε ένα αρχείο με όλα τα φυσιολογικά requests. Όπως είχαμε αναφέρει το training αρχείο αποτελεί σημαντικό παράγοντα για την λειτουργία της εφαρμογής. Όσο πιο πολλά requests έχει το αρχείο τόσο πιο καλή επίδοση θα έχει και η εφαρμογή. Για την δημιουργία λοιπόν του training αρχείου κατασκευάσαμε ένα μηχανισμό σε κώδικα Javascript ο οποίος μέσω των **Observers** έπιανε όλα τα request που είχαν παραμέτρους και τα αποθήκευε στο training αρχείο. Τα request αυτά αποθηκεύονταν κατά την περιήγησή μας στο in.gr. Σε αυτό το σημείο είναι σημαντικό να επισημανθούν δύο πράγματα:

1. Μαζεύονται μόνο τα http requests που χρησιμοποιούν την μέθοδο GET. Αυτό συμβαίνει γιατί αυτού του είδους τα requests είναι πιο ευάλωτα σε επιθέσεις από τα requests που χρησιμοποιούν την μέθοδο POST.
2. Τα requests που μαζεύτηκαν για το training αρχείο θεωρήθηκαν ότι είναι φυσιολογικά και δεν περιέχουν κανένα είδος ανωμαλίας. Ο καλύτερος βέβαια τρόπος για να μαζευτούν είναι απευθείας από τον log του server της συγκεκριμένης ιστοσελίδας.

Για την εφαρμογή του πειράματος χρησιμοποιήθηκαν τα μοντέλα που υπέδειξαν οι Christopher Kruegel, Giovanni Vigna & Marco Cova. Πιο συγκεκριμένα χρησιμοποιήθηκαν:

- **Attribute Length**
- **Attribute Presence or Absence**

- **Attribute Order**
- **TypeLearner**

Προκειμένου να παραχθούν οι απαραίτητες πληροφορίες για τα μοντέλα αυτά, η εφαρμογή κάθε φορά που διαβάζει το αρχείο (βλ. Παράρτημα Α) εκτελεί δύο σαρώσεις που αποτελούν και την φάση της εκπαίδευσης.

Η πρώτη σάρωση έχει σκόπο να συγκεντρώσει τις απαραίτητες πληροφορίες για τα μοντέλα. Για παράδειγμα έστω ότι έχουμε τα εξής requests από το παράρτημα Α: 2,3,55 και έστω ότι θέλουμε να βγάλουμε πληροφορίες για την παράμετρο m. Ο μέσος όρος της τιμής του m είναι 18,67 και η διακύμανση της είναι 34,89. Ο μέσος όρος και η διακύμανση υπολογίζονται και για όλες τις άλλες παραμέτρους του συγκεκριμένου προγράμματος (news.in.gr/ajax.aspx). Αυτές η πληροφορίες είναι χρήσιμες για το **attribute length** μοντέλο ενώ για τα υπόλοιπα μοντέλα αποθηκεύονται πληροφορίες όπως ο τύπος της τιμής μιας παραμέτρου, ο αριθμός των παραμέτρων κ.α.

Η δεύτερη σάρωση αποσκοπεί να δημιουργήσει τα φυσιολογικά anomaly scores κάθε παραμέτρου. Σε αυτήν την περίπτωση σαρώνεται από την αρχή του αρχείου ένα προς ένα request. Για παράδειγμα έστω ότι η σάρωση πρόκειται να ελέγξει το δεύτερο από το παράτημα Α request και έστω ότι βρίσκουμε τα φυσιολογικά anomaly scores για το **attribute length** μοντέλο. Επειδή ξέρουμε τον μέσο όρο και την διακύμανση της τιμής της παραμέτρου m (που βρήκαμε στην πρώτη σάρωση) μπορούμε να εφαρμόσουμε τον τύπο (2). Έτσι από τον τύπο (2) έχουμε αποτέλεσμα 1,59 (γίνεται 1 επειδή είναι πιθανότητα) και αν το εφαρμόσουμε στον τύπο (1) (με βάρος 2) παίρνουμε ως αποτέλεσμα 0. Έτσι ξέρουμε ότι το φυσιολογικό anomaly score του m είναι 0. Προφανώς αν στην πορεία βρεθεί για την παράμετρο m, στο συγκεκριμένο πρόγραμμα μεγαλύτερο score, κρατείται αυτό. Παρομοίως η ίδια διαδικασία γίνεται και στις άλλες παραμέτρους για να βρεθούν και εκεί τα φυσιολογικά anomaly scores. Σε αυτό το σημείο είναι σημαντικό να επισημανθούν πάλι δύο πράγματα:

1. Το παράδειγμα που έγινε νωρίτερα με τον μέσο όρο και με τα anomaly score των τριών request δεν είναι και το πιο κατάλληλο λόγου ελάχιστου αρθμού requests.
2. Για κάθε παράμετρο κανονικά πρέπει να προσθέτουμε τα μέγιστα φυσιολογικά anomaly score που προκύπτουν από κάθε μοντέλο όπως γίνεται και στον τύπο

(1). Ωστόσο στο πείραμά μας μόνο το **attribute length** μοντέλο είναι αυτό που επιστρέφει συνεχείς τιμές, οπότε καταλήξαμε να μην προσθέτουμε το anomaly score των άλλων μοντέλων. Ουσιαστικά δεν κρατάμε κανένα anomaly score για τα άλλα τρία μοντέλα επειδή εάν ένα από αυτά επιστρέψει μηδέν σημαίνει ότι σίγουρα θα είναι κακόβουλο και δεν χρειάζεται να το προσθέτουμε στο συνολικό score.

Οι Christopher Kruegel, Giovanni Vigna & Marco Cova κατέληξαν να χρησιμοποιήσουν για το επίπεδο ανίχνευσης τους ως κατώφλι το 10%. Αυτό σημαίνει ότι το anomaly score που υπολογίζεται κατά την φάση εκπαίδευσης θα είναι επί 10% παραπάνω από το κανονικό. Στο πείραμα μας, τα κατώφλια για τα επίπεδα ανίχνευσης φαίνονται στο παρακάτω πινακάκι:

Επίπεδο	***Κατώφλι***
Normal	95%
Medium	60%
Strong	0%

Πίνακας 4.2.1: Κατώφλια για το επίπεδο ανίχνευσης

Η προσέγγιση αυτή οφείλεται στο γεγονός ότι χρησιμοποιούμε μόνο ένα μοντέλο με συνεχείς τιμές που συμβάλει στο συνολικό φυσιολογικό anomaly score. Για παράδειγμα έστω ότι ορίζουμε ως κατώφλι το 10% και έστω ότι το συνολικό φυσιολογικό anomaly score για μια παράμετρο είναι το 8. Από το παρακάτω πινακάκι έχουμε:

	Μοντέλο1	Μοντέλο2	Μοντέλο3
Max anomaly score (during training phase)	**-2-**	**-4-**	**-2-**
Anomaly score (during detection phase)	2	5	2
Anomaly score (during detection phase)	1	5	2

Στην πρώτη περίπτωση παρατηρούμε ότι το συνολικό score είναι 9. Οπότε είτε με ή χωρίς το 10% το request θα απορρίπτοταν. Ωστόσο αν κάποιο μοντέλο γυρίσει score μικρότερο από το μέγιστο (μοντέλο 1 στην τρίτη γραμμή), το 10% θα είχε σωστή εφαρμογή και νόημα. Αυτό συμβαίνει επειδή το μοντέλο 2 αν και επιστρέφει μεγαλύτερο score από το φυσιολογικό (> 4), το συνολικό anomaly score δεν πάυει να είναι μικρότερο από το συνολικό score που ανιχνεύθηκε στην φάση της εκπαίδευσης

επί το 10%. Επομένως στην περίπτωση που υπάρχει μόνο ένα μοντέλο που επιστρέφει συνεχείς πιθανότητες και δεν μπορεί να βασιστεί σε άλλα scores, το 10% δεν αποτελεί το κατάλληλο κατώφλι για το επίπεδο ανίχνευσης.

Επειδή δεν υπήρχε κάποιος μηχανισμός για να πραγματοποιεί real-time επιθέσεις στην ιστοσελίδα του in.gr, αλλάζαμε τα requests που φαίνονταν στο URL address bar του Firefox κατά την περιήγηση μας στην ιστοσελίδα. Για παράδειγμα αν στο address bar εμφανιζόταν ένα request του τύπου http://news.in.gr/greece/article/?aid=12312 το μετατρέπαμε σε http://news.in.gr/greece-/article/?hello=1231259840 το οποίο είναι λάθος εφόσον η εφαρμογή δεν γνωρίζει αυτή την παράμετρο για το συγκεκριμένο πρόγραμμα.

Τέλος, για την εφαρμογή της λειτουργίας του μηχανισμού του **window blocking** περιηγηθήκαμε σε ιστοσελίδες που είναι γνωστές για τα αυτόματα αναδυόμενα παράθυρά τους (π.χ piratebay).

6 Συμπεράσματα και Μελλοντικές Κατευθύνσεις

6.1 Συμπεράσματα

Βρισκόμαστε σε μια εποχή όπου το Διαδίκτυο εξελίσσεται συνεχώς με ραγδαίο ρυθμό προσφέροντας ολοένα και περισσότερες υπηρεσίες. Μέσα από αυτό μπορούμε να ανταλλάσουμε πληροφορίες και να επικοινωνούμε χωρίς κάποιο ουσιαστικό όριο. Ωστόσο η χρήση του Διαδικτύου μπορεί να αποβεί αρκετά αρνητική εάν έχει σκοπό να παραβιάζει την ιδιωτικότητα του χρήστη. Η εργασία στην παρούσα πτυχιακή έρχεται να καλύψει ακριβώς αυτό το πρόβλημα. Πιο συγκεριμένα στην παρούσα πτυχιακή υλοποιήθηκε ένα σύστημα ανίχνευσης ανώμαλης συμπεριφοράς σε HTTP requests. Η εφαρμογή δέχεται ένα αρχείο προπόνησης που περιέχει τα φυσιολογικά requests χωρίς κακόβουλες επιθέσεις για να δημιουργήσει τα φυσιολογικά anomaly scores και για να μπορεί ύστερα να τα συγκρίνει με τα real-time requests για τυχόν αποκλίσεις. Επιπροσθέτως υλοποιήθηκε και ένας μηχανισμός ο οποίος μπλοκάρει τα pop-up παράθυρα του browser αποτρέποντας κακόβουλες δραστηριότητες.

Στα πλαίσια της πτυχιακής εργασίας έγιναν αρκετές δοκιμές ώστε να εξακριβωθεί αν οι παραπάνω μηχανισμοί λειτουργούν επιτυχώς. Ο μηχανισμός του **window blocking** μπόρεσε με επιτυχία να μπλοκάρει κάθε pop-up που εμφανιζόταν στον browser κατά την πλοήγησή μας σε ιστοσελίδες που περιέχουν malicious περιεχόμενο. Ωστόσο όπως είχε αναφερθεί και νωρίτερα υπάρχουν και κάποια pop-ups τα οποία δεν αποτελούν επικίνδυνα για την ιδιωτικότητα του χρήστη και ενεργοποιούνται με την θέλησή του. Σε αυτήν την περίπτωση ο χρήστης θα πρέπει να απενεργοποιήσει το extension για να του εμφανιστεί το παράθυρο.

Τα μοντέλα ανίχνευσης ανώμαλης συμπεριφοράς που εφαρμόστηκαν στο extension μπόρεσαν και αυτά με την σειρά τους να προστατεύσουν με επιτυχία την ιδιωτικότητα του χρήστη. Όπως είχε επισημανθεί και νωρίτερα δεν υπήρχε κάποιος συγκεκριμένος μηχανισμός οποίος να προκαλεί real-time επιθέσεις που να αλλάζει τις παραμέτρους των request. Προκειμένου λοιπόν να μπορούμε να προσομοιώσουμε παρόμοιες επιθέσεις αλλάζαμε τις παραμέτρους των request που φαίνονταν στο address bar του browser με αποτέλεσμα η εφαρμογή να απορρίπτει τα τροποποιημένα request. Εδω αξίζει να σημειωθεί ότι η σωστή λειτουργία ορισμένων μοντέλων της εφαρμογής βασίζεται στην λογική σειρά των παραμέτρων των request (όσο αφορά

την φάση της εκαπίδευσης). Για παράδειγμα οι προγραμματιστές ενός site δεν θα φτιάξουν ποτέ ένα request που θα έχουν δύο ίδιες παραμέτρους.

Ο συνδυασμός των παραπάνω μηχανισμών σε ένα ενιαίο σύστημα αποτελεί μια καινοτόμος λύση η οποία επιτυγχάνει την προστασία όχι μόνο του client αλλά και των servers. Η συγκεκριμένη εφαρμογή μπορεί να εφαρμοστεί σε οποιοδήποτε site είτε είναι κοινωνικού (π.χ ένα forum) ή ψυχαγωγικού περιεχομένου, με τον μοναδικό περιορισμό ότι το site δεν πρέπει να χρησιμοποιεί το πρωτόκολλο HTTPS ή οποιαδήποτε άλλη κρυπτογράφηση.

Η εφαρμογή αυτή ως καινοτόμα ιδέα απαιτεί περαιτέρω ανάλυση και υλοποίηση προκειμένου να υποστηρίζει μεγαλύτερο εύρος λειτουργιών. Για το λόγο αυτόν ο κώδικας της εφαρμογής είναι διαθέσιμος στο github.com προκειμένου να δημιουργηθούν απορίες και συζητήσεις μεταξύ και άλλων προγραμματιστών που ενδιαφέρονται για το θέμα της εφαρμογής. Στόχος είναι να προταθούν και άλλες επεκτάσεις της υλοποίησης του extension.

6.2 Μελλοντικές Κατευθύνσεις

Μια πιθανή μελλοντική τροποποίηση του συστήματος θα μπορούσε να είναι η δυνατότητα να υποστηρίζει φυσιολογικά anomaly scores για παραπάνω από ένα site ταυτοχρόνως. Για παράδειγμα το σύστημα θα μπορούσε να λαμβάνει πολλά αρχεία προπόνησης από διάφορα site και κάθως ο χρήστης θα έμπαινε από το ένα site στο άλλο η εφαρμογή θα καταλάβαινε την αλλαγή του host (π.χ διαβάζοντας το URL στο address bar) και θα φόρτωνε το κατάλληλο αρχείο προπόνησης. Με αυτό τον τρόπο η εφαρμογή γίνεται πιο ευέλικτη υποστηρίζοντας μεγαλύτερο εύρος ιστοσελίδων. Ωστόσο η προσέγγιση αυτή ίσως επηρεάσει την απόδοση του συστήματος επειδή σε real-time θα πρέπει να ψάχνει και να φορτώνει το κατάλληλο αρχείο.

Επίσης η εφαρμογή αυτή θα μπορούσε τροποποιηθεί κατάλληλα ώστε να μπορεί να χρησιμοποιηθεί από μεγάλα site π.χ το Facebook. Σε αυτο το είδος υλοποίησης το Facebook δεν θα επέτρεπε σε κανέναν χρήστη να πλοηγηθεί στο site αν δεν είχαν εγκαταστημένο το extension στο browser τους (υποτίθετε ότι θα υπήρχε τρόπος να αναγνωρίζει το site αν έχει εγκαταστήσει ο χρήστης το extension).

Επίσης οι developers του Facebook θα παρείχαν και ένα αρχείο προπόνησης το οποίο οι χρήστες θα το φόρτωναν στην εφαρμογή. Έτσι αν οι χρήστες έχουν φορτώσει το αρχείο και έχουν ενεργοποιήσει το extension θα μπορούσαν να περιηγηθούν με ασφάλεια στο Facebook. Να επισημανθεί σε αυτή την περίπτωση το extension θα δουλεύει σε HTTPS πρωτόκολλο. Υποτίθετε ότι το extension θα ξέρει να αποκρυπογραφεί τις τιμές των παραμέτρων των requests.

Τέλος, μια μελλοντική τροποποίηση θα μπορούσε να γίνει και στον μηχανισμό του **window blocking**. Ο μηχανισμός αυτός θα μπορούσε να γίνει ακόμα πιο εξειδικευμένος έτσι ώστε να μπορεί να ξεχωρίσει τα κακόβουλα από τα φυσιολογικά pop-ups. Στην περίπτωση αυτή προτείνονται τα εξής:

- Σε μια πρώτη προσέγγιση θα μπορούσε να δημιουργηθεί ένα αρχείο με όλα τα site (καλού περιεχομένου και γνωστά π.χ η google) τα οποία οποία περιέχουν pop-up παράθυρα. Κάθε φορά που ο χρήστης θα έμπαινε σε μια τέτοια ιστοσελίδα, η εφαρμογή αμέσως θα έψαχνε εάν ο host της ιστοσελίδας υπάρχει μέσα στο αρχείο. Σε περίπτωση που υπάρχει ο host τότε κανένα pop-up δεν θα μπλοκάρεται σε αυτή την ιστοσελίδα και αντίθετα. Η προσέγγιση αυτή αν και φαίνεται να αποδίδει καλά αποτελεί μια επίπονη διαδικασία για την δημιουργία του αρχείου.

- Σε μια δεύτερη προσέγγιση η εφαρμογή θα μπορούσε να αποφασίζει αν θα μπλοκάρει η όχι ένα παράθυρο με την βοήθεια του χρήστη. Κάθε φορά που θα εμφανίζεται ένα pop-up, η εφαρμογή θα ρωτάει τον χρήστη εάν επιθυμεί να συνεχίζει να μπλοκάρει τα pop-ups για αυτήν την συγκεκριμένη ιστοσελίδα εμφανίζοντας του ένα παράθυρο επιλογών. Με αυτόν τον τρόπο θα δημιουργείται αυτόματα ένα αρχείο με τις προτιμήσεις του χρήστη, το οποίο θα διαβάζεται από την εφαμρογή κάθε φορά που θα φορτώνει ένας καινούργιος host στον browser.

Παράρτημα Α

Training Αρχείο:

1	http://news.in.gr/greece/article/?aid=1231259840
2	http://news.in.gr/ajax.aspx?m=Polls.PollView&la=1&ct=296&plid=1004&_=1375287779258
3	http://news.in.gr/ajax.aspx?m=News.ArticleHit&aid=1231259840&la=1
4	http://auto.in.gr/ajax.aspx?m=Polls.PollView&la=1&ct=304&plid=931&_=1375360708895
5	http://dictionary.in.gr/WebResource.axd?d=SzKrBQ4iTmnrZ2MKYABiUNrI8OCYWX3WVLpSOIooUUsv0Ig-ZnrXffO9DQs1&t=634937645953150316
6	http://dictionary.in.gr/WebResource.axd?d=CCGRD8o065lm_wQmRN6zefA43hWh5a03Bem80vIFUbBUYSgKxKvk_aZXRy_3qC3UIZD1pg2&t=634937645953150316
7	http://auto.in.gr/news/world/article/?aid=1231259496
8	http://www.in.gr/15781001_photo_19_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODEwMDFfcGhvdG8oMTkpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIxNSZjcj10cnVlJmF0PTQ%3d
9	http://www.in.gr/15780865_photo_15_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODA4NjVfcGhvdG8oMTUpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIxNSZjcj10cnVlJmF0PTQ%3d
10	http://www.in.gr/15781002_photo_21_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODEwMDJfcGhvdG8oMjEpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIxNSZjcj10cnVlJmF0PTQ%3d
11	http://www.in.gr/15780912_photo_17_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODA5MTJfcGhvdG8oMTcpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIxNSZjcj10cnVlJmF0PTQ%3d
12	http://www.in.gr/15781000_photo_25_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODEwMDBfcGhvdG8oMjUpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIxNSZjcj10cnVlJmF0PTQ%3d
13	http://www.in.gr/15781140_photo_26_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODExNDBfcGhvdG8oMjYpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIxNSZjcj10cnVlJmF0PTQ%3d
14	http://www.in.gr/15781004_photo_24_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODEwMDRfcGhvdG8oMjQpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIxNSZjcj10cnVlJmF0PTQ%3d
15	http://www.in.gr/15781003_photo_23_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODEwMDNfcGhvdG8oMjMpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIxNSZjcj10cnVlJmF0PTQ%3d
16	http://www.in.gr/15781220_photo_27_.limghandler.jpg?i=aT1m

	aWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODEyMj BfcGhvdG8oMjcpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIx NSZjcj10cnVlJmF0PTQ%3d
17	http://www.in.gr/15782165_P90129238.limghandler.jpg?i=aT1m aWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODIxNj VfcDkwMTI5MjM4LmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIx NSZjcj10cnVlJmF0PTQ%3d
18	http://www.in.gr/15782167_P90129249.limghandler.jpg?i=aT1m aWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODIxNj dfcDkwMTI5MjQ5LmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIx NSZjcj10cnVlJmF0PTQ%3d
19	http://www.in.gr/15782164_P90129203.limghandler.jpg?i=aT1m aWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODIxNj RfcDkwMTI5MjAzLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIx NSZjcj10cnVlJmF0PTQ%3d
20	http://www.in.gr/15782166_P90129245.limghandler.jpg?i=aT1m aWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODIxNj ZfcDkwMTI5MjQ1LmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIx NSZjcj10cnVlJmF0PTQ%3d
21	http://www.in.gr/15781223_photo_32_.limghandler.jpg?i=aT1m aWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODEyMj NfcGhvdG8oMzIpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIx NSZjcj10cnVlJmF0PTQ%3d
22	http://www.in.gr/15781226_photo_36_.limghandler.jpg?i=aT1m aWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODEyMj ZfcGhvdG8oMzYpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIx NSZjcj10cnVlJmF0PTQ%3d
23	http://www.in.gr/15781222_photo_31_.limghandler.jpg?i=aT1m aWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODEyMj JfcGhvdG8oMzEpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIx NSZjcj10cnVlJmF0PTQ%3d
24	http://www.in.gr/15781218_photo_37_.limghandler.jpg?i=aT1m aWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODEyMT hfcGhvdG8oMzcpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIx NSZjcj10cnVlJmF0PTQ%3d
25	http://www.in.gr/15781221_photo_29_.limghandler.jpg?i=aT1m aWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODEyMj FfcGhvdG8oMjkpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIx NSZjcj10cnVlJmF0PTQ%3d
26	http://www.in.gr/15781225_photo_35_.limghandler.jpg?i=aT1m aWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODEyMj VfcGhvdG8oMzUpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIx NSZjcj10cnVlJmF0PTQ%3d
27	http://www.in.gr/15781224_photo_33_.limghandler.jpg?i=aT1m aWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODEyMj RfcGhvdG8oMzMpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIx NSZjcj10cnVlJmF0PTQ%3d
28	http://www.in.gr/15781626_P90129735.limghandler.jpg?i=aT1m aWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODE2Mj ZfcDkwMTI5NzM1LmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIx NSZjcj10cnVlJmF0PTQ%3d
29	http://www.in.gr/15781625_P90129711.limghandler.jpg?i=aT1m aWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODE2Mj VfcDkwMTI5NzExLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIx NSZjcj10cnVlJmF0PTQ%3d

30	http://www.in.gr/15781627_P90129731.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODE2MjdfcDkwMTI5NzMxLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIxNSZjcj10cnVlJmF0PTQ%3d
31	http://www.in.gr/15782163_photo_44_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODIxNjNfcGhvdG8oNDQpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIxNSZjcj10cnVlJmF0PTQ%3d
32	http://www.in.gr/15782045_photo_43_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODIwNDVfcGhvdG8oNDMpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIxNSZjcj10cnVlJmF0PTQ%3d
33	http://www.in.gr/15782047_photo_39_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODIwNDdfcGhvdG8oMzkpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIxNSZjcj10cnVlJmF0PTQ%3d
34	http://www.in.gr/15782049_photo_41_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODIwNDlfcGhvdG8oNDEpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIxNSZjcj10cnVlJmF0PTQ%3d
35	http://www.in.gr/15780823_photo_14_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODA4MjNfcGhvdG8oMTQpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIxNSZjcj10cnVlJmF0PTQ%3d
36	http://www.in.gr/15780676_photo_12_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODA2NzZfcGhvdG8oMTIpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIxNSZjcj10cnVlJmF0PTQ%3d
37	http://www.in.gr/15780722_photo_13_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODA3MjJfcGhvdG8oMTMpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIxNSZjcj10cnVlJmF0PTQ%3d
38	http://www.in.gr/15780211_photo_11_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3ODAyMTFfcGhvdG8oMTEpLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIxNSZjcj10cnVlJmF0PTQ%3d
39	http://www.in.gr/15779969_photo_8_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3Nzk5NjlfcGhvdG8oOCkuanBnJnc9NzYmaD00OSZzdD10cnVlJmJnPTE2Nzc3MjE1JmNyPXRydWUmYXQ9NA%3d%3d
40	http://www.in.gr/15779970_photo_9_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3Nzk5NzBfcGhvdG8oOSkuanBnJnc9NzYmaD00OSZzdD10cnVlJmJnPTE2Nzc3MjE1JmNyPXRydWUmYXQ9NA%3d%3d
41	http://www.in.gr/15779966_photo_10_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3Nzk5NjZfcGhvdG8oMTApLmpwZyZ3PTc2Jmg9NDkmc3Q9dHJ1ZSZiZz0xNjc3NzIxNSZjcj10cnVlJmF0PTQ%3d
42	http://www.in.gr/15779968_photo_7_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3Nzk5NjhfcGhvdG8oNykuanBnJnc9NzYmaD00OSZzdD10cnVlJmJnPTE2Nzc3MjE1JmNyPXRydWUmYXQ9NA%3d%3d
43	http://www.in.gr/15779967_photo_6_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3Nzk5NjdfcGhvdG8oNikuanBnJnc9NzYmaD00OSZzdD10cnVlJmJnPTE2Nzc3MjE1J

	mNyPXRydWUmYXQ9NA%3d%3d
44	http://www.in.gr/15779644_photo_1_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3Nzk2NDRfcGhvdG8oMSkuanBnJnc9NzYmaD00OSZzdD10cnVlJmJnPTE2Nzc3MjE1JmNyPXRydWUmYXQ9NA%3d%3d
45	http://www.in.gr/15779642_photo_5_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3Nzk2NDJfcGhvdG8oNSkuanBnJnc9NzYmaD00OSZzdD10cnVlJmJnPTE2Nzc3MjE1JmNyPXRydWUmYXQ9NA%3d%3d
46	http://www.in.gr/15779645_photo_2_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3Nzk2NDVfcGhvdG8oMikuanBnJnc9NzYmaD00OSZzdD10cnVlJmJnPTE2Nzc3MjE1JmNyPXRydWUmYXQ9NA%3d%3d
47	http://www.in.gr/15779647_photo_4_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3Nzk2NDdfcGhvdG8oNCkuanBnJnc9NzYmaD00OSZzdD10cnVlJmJnPTE2Nzc3MjE1JmNyPXRydWUmYXQ9NA%3d%3d
48	http://www.in.gr/15779646_photo_3_.limghandler.jpg?i=aT1maWxlcyUyZjElMmZtZWRpYSUyZjIwMTMlMmYwNyUyZjI5JTJmMTU3Nzk2NDZfcGhvdG8oMykuanBnJnc9NzYmaD00OSZzdD10cnVlJmJnPTE2Nzc3MjE1JmNyPXRydWUmYXQ9NA%3d%3d
49	http://auto.in.gr/ajax.aspx?m=Polls.PollView&la=1&ct=304&plid=931&_=1375360713403
50	http://auto.in.gr/ajax.aspx?m=News.ArticleHit&aid=1231259496&la=1
51	http://dictionary.in.gr/WebResource.axd?d=SzKrBQ4iTmnrZ2MKYABiUNrI8OCYWX3WVLpSOIooUUsv0Ig-ZnrXffO9DQs1&t=634937645953150316
52	http://dictionary.in.gr/WebResource.axd?d=CCGRD8o065lm_wQmRN6zefA43hWh5a03Bem80vIFUbBUYSgKxKvk_aZXRy_3qC3UIZD1pg2&t=634937645953150316
53	http://dictionary.in.gr/WebResource.axd?d=SzKrBQ4iTmnrZ2MKYABiUNrI8OCYWX3WVLpSOIooUUsv0Ig-ZnrXffO9DQs1&t=634937645953150316
54	http://dictionary.in.gr/ScriptResource.axd?d=epcmk8QS8lio2hpLnaKwoSNSN72h-LnrF_cdi-L9wak0laAzPReskK9jEinozal-vYQkHjqzK6aF4l2pfrGgnqdqFcmTq_L8aFVJfsdFal24O5R8WkQfjAoBaLM1&t=68c4b60a
55	http://news.in.gr/ajax.aspx?m=News.PostedArticlesListView&la=1&si=1&key=y4ppHifRGnFO%2b5BTeAUJF1jvBgDKZtdL4BEg1fH5aHRf0Lr3YZyKHylSb2kK8guhm7KEOxuY3tX%2b6YIH3AThFrPFmnwwkkx7mLe6mauGJFF5H1tM
56	http://finance.in.gr/WebResource.axd?d=ZS4mncd2-SyqYx0yX9bclklQAiRGCAOljFPDf1IbIm6QD56ofJ_4EnGJE8PJCJUoA4-iW8kbcZQY93f3c8KK2pTLxSaATsNRQAMJ5b3Bkrx9U_lP0&t=634516086340000000
57	http://finance.in.gr/WebResource.axd?d=2Pmg40W6Jce48hC5gwvm9QNyZz-aUamQTGhf4__L9gmF9hE73xxbreCoBQXPzO6T99k--j44q8jjrEsCYl8IW76u0x-3KmVwVpYocy6FBN8LhUAM0&t=634516086340000000
58	http://finance.in.gr/WebResource.axd?d=xsn4krzc6xsXQAtMXnHyduzlubUOMqqPXGG19t6SNOo1lCisA3_V0YNz0RKhmttW9Ah3FJSvOe5XHJTSClkLiybkkdnx4zJAYuoN6WNRhSkPiuP60&t=634516086340000000
59	http://finance.in.gr/WebResource.axd?d=uvz90a6J-

	tvhjiXrjKj_xXTM8cg7CNZ3O5LvuWSkAgf2oZh-kjBoPrDTW_P9dc5zpHYoxTsBW5B54oleOTLGvMtXqF1rECoFfGMbhICMkMB1PuS5ETAwEPjKmjLdGM6rNw9ebQ2&t=634516086340000000
60	http://finance.in.gr/WebResource.axd?d=-Goy0bhjvL_3lEM-znBhJRPQ9YTgG1JRES5sEqdCSF80IyTmv7jil9qAp2a6pHxEiw3Qg-HP5Ifg_TXogujlDMC-vJohvBFq-sfteOo_i-rFXHp50&t=634516086340000000
61	http://finance.in.gr/WebResource.axd?d=kXNkX3ekx3uH0ltHCRcv_sGc2A1qBoZc8V30wmO_hvl0EWn6WsRfNxYA9Cd2xLW4s2vcK50yYL9MWOXXCbgPIOIby-AH5bXGlIQx8gdTRfc6ko7o4y9FtCHRD03wmOiD739sew2&t=634516086340000000
62	http://finance.in.gr/WebResource.axd?d=UD5sjtgWvUfzwvD_jBv5AcsvP_POdmPa-RCPS7wYeolajYJ1hzKn0Te55qk3l0fjd0JY52Vcv-5q1yyZS4r1EhXeAVtbcuQi2B9RZw2&t=634516086340000000
63	http://finance.in.gr/WebResource.axd?d=4gYCdGyRya6dmcevZHUegSxoQRnpqOcVqafTLo9rC8j2dsB7H0HGD0EFjV9NClI0V9T4XbaN9avThs_IRYbL9wPaTV4LISbcnOKC8F9z1ClSbD-tsilsLhzAv9o1&t=634516086340000000
64	http://finance.in.gr/WebResource.axd?d=SzKrBQ4iTmnrZ2MKYABiUNrI8OCYWX3WVLpSOIooUUsv0Ig-ZnrXffO9DQs1&t=634937645904636182
65	http://finance.in.gr/ScriptResource.axd?d=epcmk8QS8lio2hpLnaKwoSNSN72h-LnrF_cdi-L9wak0laAzPReskK9jEinozal-vYQkHjqzK6aF4l2pfrGgnqdqFcmTq_L8aFVJfsdFal24O5R8WkQfjAoBaLM1&t=150492e7
66	http://finance.in.gr/WebResource.axd?d=s80bksIi4PO2KZn7CRYqjgEpCdbfAvjJht8Fv-KA7nZJf0jbFjwJgsGHcr9ibHYuKlBg0Sv5EH-R70Pslc76a-4LslelXSDzy8571rT8U-8PaIKyzV6aZ5xJJBIOPR49Ap5aBQ2&t=634516086340000000
67	http://finance.in.gr/images/In_Finance_Android_Banner_(300x250)_final.swf?go=/android/default.aspx
68	http://finance.in.gr/Telerik.Web.UI.WebResource.axd?compress=1&_TSM_CombinedScripts_=%3b%3bTelerik.Web.UI%2c+Version%3d2011.2.915.40%2c+Culture%3dneutral%2c+PublicKeyToken%3d121fae78165ba3d4%3aen-US%3a9799c67b-558a-475d-bf30-a0ccf0dbb8b7%3a53e1db5a%3a8f70baae%3a1c2121e%3acaa7b3a7%3ad126a8ef%3a92753c09%3a91f742eb
69	http://finance.in.gr/WebResource.axd?d=SzKrBQ4iTmnrZ2MKYABiUNrI8OCYWX3WVLpSOIooUUsv0Ig-ZnrXffO9DQs1&t=634980805110643322
70	http://finance.in.gr/ScriptResource.axd?d=epcmk8QS8lio2hpLnaKwoSNSN72h-LnrF_cdi-L9wak0laAzPReskK9jEinozal-vYQkHjqzK6aF4l2pfrGgnqdqFcmTq_L8aFVJfsdFal24O5R8WkQfjAoBaLM1&t=150492e7
71	http://finance.in.gr/WebResource.axd?d=UFCqp-iisLWUS6gQ1AwBqAc_Td9d3HD8F1ehImeSf5jt3tHJmJINI33ftQaBtsjXlZHxJvnEmaAkqbkE6g8Y7HqEjf4bNfGUjMrtkJ2AI0lXcJBYhVI-diWL26Y9Cd52OBAiuA2&t=634516086340000000
72	http://finance.in.gr/WebResource.axd?d=yzOhl4NmNx-W4AIMs8w8NHtYwld9xzZYEGQkPEAOR1dYml3yn0oD8Rd11_Cji004-_Bfhg5ROunb9gA_RC1tz1-Qizx-kVTxpuR0-gmiGQVudBo7Y0RtgXaFzJo1&t=634516086340000000

Βιβλιογραφία

1. Gustavo Miguel Barroso Assis do Nascimento (2010), *ANOMALY DETECTION OF WEB-BASED ATTACKS*
   http://docs.di.fc.ul.pt/jspui/bitstream/10455/6704/1/Disserta%C3%A7%C3%A3o%20de%20Mestrado%20do%20Gustavo%20Nascimento_Nov-20010.pdf

2. Kenneth L. Ingham, Hajime Inoue (2007), *Comparing Anomaly Detection Techniques for HTTP*
   http://www.i-pi.com/~ingham/pubs/raid2007-revised.pdf

3. Jessica Halida Harjono (2001), *XUL Application Development*
   http://www-nlp.stanford.edu/kirrkirr/theses/Jessica-FYP-Report.pdf

4. Li Dongdong (2009), *How to develop Firefox Extensions*
   2009-12-11_HowToDevelopFirefoxAddOns

5. Jennifer Golbeck (2003), *Introduction to JavaScript*
   http://www.cs.umd.edu/~golbeck/LBSC690o/js1.pdf

6. Thomas Powell and Fritz Schneider (2004), *JavaScript 2.0-The Complete Reference, Second Edition*
   http://freecodingtutorial.files.wordpress.com/2011/10/mcgraw-hill-javascript-the-complete-reference.pdf

7. D. Martakos (1999), *HYPERTEXT TRANSFER PROTOCOL*
   http://alexandra.di.uoa.gr/mmtech/msTech/1_HTML/PDFs/chp2.pdf

8. John Yannakopoulos (2003), *HyperText Transfer Protocol: A Short Course*
   http://condor.depaul.edu/dmumaugh/readings/handouts/SE435/HTTP/http.pdf

9. Trevor Jim, Nikhil Swamy, Michael Hicks (2007), *Defeating Script Injection Attacks with BrowserEnforced Embedded Policies*
   http://drum.lib.umd.edu/bitstream/1903/4008/1/paper.pdf

10. Christopher Kruegel, Giovanni Vigna, Marco Cova (2010), *Detection and Analysis of Drive-by-Download Attacks and Malicious JavaScript Code*
    http://www.cs.ucsb.edu/~vigna/publications/2010_cova_kruegel_vigna_Wepawet.pdf

11. Christopher Kruegel, Giovanni Vigna (2003), *Anomaly Detection of Web-based Attacks*
    http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.4.8030&rep=rep1&type=pdf

12. Collin Jackson, Andrew Bortz, Dan Boneh, John C Mitchell (2006), *Protecting Browser State from Web Privacy Attacks*
    http://crypto.stanford.edu/sameorigin/sameorigin.pdf

13. Wikipedia-Διαδίκτυο: http://el.wikipedia.org/wiki/%CE%94%CE%B9%CE%B1%CE%B4%CE%AF%CE%BA%CF%84%CF%85%CE%BF

14. Wikipedia-ClickJacking: http://en.wikipedia.org/wiki/Clickjacking

15. History of Mozilla: http://www.mozilla.org/en-US/about/history/

16. XUL: http://en.wikipedia.org/wiki/XUL

17. Mozilla's Chrome: https://developer.mozilla.org/en/docs/Chrome_Registration

18. Pop-ups ads: http://en.wikipedia.org/wiki/Pop-up_ad

19. Firefox release history: http://en.wikipedia.org/wiki/Firefox_release_history

20. Mozilla's XPCOM: https://developer.mozilla.org/en-US/docs/XPCOM

21. Mozilla's Observers: https://developer.mozilla.org/en-US/docs/XPCOM_Interface_Reference/nsIObserver

22. XML Namespaces (UNESCO,1999): http://docs.exdat.com/docs/index-275184.html?page=21

23. Javascript: http://en.wikipedia.org/wiki/JavaScript

24. Javascript Tutorial: http://www.w3schools.com/js/

25. Wikibooks-ΜΑΡΙΑΝΝΑ ΣΤΟΥΓΙΑΝΝΙΔΟΥ: http://el.wikibooks.org/wiki/Χρήστης:ΜΑΡΙΑΝΝΑ_ΣΤΟΥΓΙΑΝΝΙΔΟΥ/Βιβλία/Θέματα_Ασφάλειας_Δικτύων_Για_Μηχανικούς_Πληροφορικής

26. HyperText Transfer Protocol: http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol

27. Introduction to XUL: https://developer.mozilla.org/en-US/docs/XUL/Introduction_to_XUL

28. Add-on: http://en.wikipedia.org/wiki/Add-on_%28Mozilla%29