



ΠΑΝΕΠΙΣΤΗΜΙΟ  
ΠΑΤΡΩΝ  
UNIVERSITY OF PATRAS

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ  
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

---

Σχεδιασμός και Υλοποίηση Μηχανισμού Ανίχνευσης  
Απειλών κατά της Ιδιωτικότητας στο Διαδίκτυο

---

Σταμέλλος Γρηγόριος  
Α.Μ. 5208  
stamellos@ceid.upatras.gr

*Επιβλέπων:*  
Σωτήρης Νικολετσέας  
Αναπληρωτής Καθηγητής  
nikole@cti.gr

Πάτρα, Ιανουάριος 2017



# *Ευχαριστίες*

Αρχικά θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου, κ. Σωτήρη Νικολετσέα που μου ανέθεσε και επέβλεψε την παρούσα διπλωματική εργασία. Ακόμη, θα ήθελα να ευχαριστήσω θερμά την υποψήφια Διδάκτωρ Αντελίνα Μάδια και μεταπτυχιακό φοιτητή Δημήτριο Τσολοβό για τη καθοδήγηση που μου παρείχαν κατά τη διάρκεια εκπόνησης της εργασίας. Τέλος, ένα μεγάλο ευχαριστώ στους γονείς και τις αδερφές μου, για την συμπαράσταση και την υπομονή τους, καθ' όλη τη διάρκεια των σπουδών μου.



## Περίληψη

Το Διαδίκτυο έφερε πραγματική επανάσταση στον τομέα της πληροφορίας. Μέσω αυτού προσφέρεται μία ευρεία κλίμακα υπηρεσιών και τεχνολογιών, καθώς γίνεται πλέον εύκολη η πρόσβαση σε ένα τεράστιο όγκο πληροφοριών. Επακόλουθο αυτού είναι η έκθεση περισσότερων προσωπικών δεδομένων στο διαδίκτυο με αποτέλεσμα η προστασία τους, να αποτελεί ένα από τα κρισιμότερα θέματα τα τελευταία χρόνια. Υπάρχουν εκατοντάδες εταιρείες, επιχειρηματικά μοντέλα και μηχανισμοί παρακολούθησης που χρησιμοποιούνται για τη συλλογή όσο το δυνατόν περισσότερων πληροφοριών για πράγματα τα οποία αναζήτησε ο χρήστης στο διαδίκτυο, ιστοσελίδες που επισκέφθηκε, άτομα με τα οποία ήρθε σε επικοινωνία καθώς και προϊόντα τα οποία αγόρασε. Στην παρούσα διπλωματική εργασία έγινε η ανάπτυξη μίας επέκτασης για το πρόγραμμα περιήγησης Mozilla Firefox, σκοπός της οποίας είναι η ενημέρωση του χρήστη σχετικά με την ασφάλεια των προσωπικών του δεδομένων όταν επισκέπτεται μία ιστοσελίδα στο διαδίκτυο. Εξετάζει αν έχει εγκατασταθεί μία ασφαλής σύνδεση στο διαδίκτυο, ελέγχοντας όλες τις παραμέτρους, μεταξύ χρήστη και ιστότοπου καθώς ανιχνεύει και κάποια είδη cookies.

Αρχικά πραγματοποιήθηκε η μελέτη των κινδύνων που έχουν ανιχνευθεί μέχρι σήμερα. Στη συνέχεια, έγινε η ανάλυση ορισμένων μηχανισμών που υπάρχουν ήδη καθώς και των απειλών που αντιμετωπίζουν. Ακολούθησε η επιλογή των κινδύνων που θα ανιχνεύσει η επέκταση καθώς και η υλοποίηση της. Τέλος, δημιουργήθηκε ένας πλήρως αυτοματοποιημένος μηχανισμός ο οποίος επισκέπτεται καθημερινά μία λίστα από ιστοσελίδες, με σκοπό να γίνει μία γραφική ανάλυση των αποτελεσμάτων η οποία δείχνει τη συμπεριφορά των ιστοσελίδων με το πέρασμα του χρόνου, με βάση τους κινδύνους που ανιχνεύει η επέκταση.

Οι τεχνολογίες που χρησιμοποιήθηκαν είναι οι γλώσσες προγραμματισμού Python για τη δημιουργία ενός αυτοματοποιημένου μηχανισμού επίσκεψης ιστοσελίδων, η Javascript για την υλοποίηση του add-on, η PHP για τη δημιουργία του API μέσω του οποίου γίνεται η μεταφορά των δεδομένων στη βάση, οι γλώσσες σήμανσης υπερκειμένου HTML και CSS οι οποίες δημιουργούν ένα φιλικό περιβάλλον για το χρήστη. Τελειώνοντας, για την υλοποίηση της επέκτασης, χρησιμοποιήθηκε η πλατφόρμα Nodejs και για την αποθήκευση των δεδομένων ο Wamp Server.



# Περιεχόμενα

Περιεχόμενα	vi
<b>1 Εισαγωγή</b>	<b>1</b>
1.1 Privacy Flag . . . . .	2
1.2 Απόρρητο Επικοινωνίας . . . . .	2
1.2.1 Κρυπτογράφηση Δεδομένων ( <i>Data Encryption</i> ) . . . . .	3
1.2.2 HSTS (HTTP Strict Transport Security) . . . . .	3
1.2.3 Μέθοδοι Κρυπτογράφησης ( <i>cipher suite</i> ) . . . . .	5
1.2.4 Αλυσίδα Εμπιστοσύνης . . . . .	5
1.2.5 Certificate pinning . . . . .	6
1.2.6 Public key pinning . . . . .	7
1.2.7 Do Not Track . . . . .	7
1.3 Μηχανισμοί που θέτουν σε κίνδυνο την ιδιωτικότητα . . . . .	8
1.3.1 Πληροφορίες που δίνονται απευθείας από το χρήστη . . . . .	8
1.3.2 HTTP Cookies . . . . .	9
1.3.3 HTTP Cookies από domain τρίτων . . . . .	9
1.3.4 HTTP cookies τα οποία χρησιμοποιούν το Adobe Flash . . . . .	10
1.3.5 Supercookies . . . . .	10
1.3.6 Zombie cookies και Evercookies . . . . .	11
1.3.7 Καταγραφή ιστορικού περιήγησης . . . . .	11
1.3.8 API για την αποθήκευση δεδομένων με τη χρήση SQL ερωτήσεων	12
1.3.9 Ανίχνευση “δακτυλικού αποτυπώματος” συσκευής . . . . .	12
1.3.10 Δυνητικά επικίνδυνα APIs του πλαισίου HTML5 . . . . .	13
1.3.11 Συμμόρφωση με τη πολιτική προστασίας προσωπικών δεδομένων	14
1.4 Δυνητικά ανασφαλείς τεχνολογίες . . . . .	14
1.4.1 Portable Document Format (PDF) . . . . .	15
1.4.2 Πλατφόρμα πολυμέσων Flash . . . . .	15
1.4.3 Τεχνολογία Microsoft Silverlight . . . . .	15
1.4.4 ActiveX . . . . .	16
1.4.5 Γλώσσα προγραμματισμού Java . . . . .	16
<b>2 Σχετικές Εργασίες</b>	<b>17</b>
2.1 Μηχανισμοί Σχετικοί με τη φραγή Υπηρεσιών Διαφήμισης . . . . .	18
2.1.1 Επέκταση Request-Policy . . . . .	19
2.1.2 Επέκταση Adblock Plus . . . . .	19
2.1.3 Επέκταση Privacy Badger . . . . .	20
2.2 Μπλοκάρισμα Εκτέλεσης content script . . . . .	21

2.2.1	Επέκταση NoScript . . . . .	22
2.2.2	Επέκταση Ghostery . . . . .	22
2.2.3	Επέκταση Flashblock . . . . .	23
2.3	Ασφάλεια Επικοινωνίας . . . . .	23
2.3.1	Επέκταση HTTPS Everywhere . . . . .	24
2.3.2	Επέκταση Web of Trust . . . . .	24
<b>3</b>	<b>Μηχανισμός Ανίχνευσης Κινδύνων</b>	<b>27</b>
3.1	Πλατφόρμα Nodejs . . . . .	27
3.1.1	Εγκατάσταση Πλατφόρμας Nodejs για τη δημιουργία του add-on	28
3.2	Έλεγχος HTTPS . . . . .	30
3.3	Έλεγχος HSTS . . . . .	31
3.4	Έλεγχος για τους αλγορίθμους κρυπτογράφησης . . . . .	33
3.5	Έλεγχος για την Αρχή Πιστοποίησης . . . . .	35
3.6	Έλεγχος για το HTTP Public Key Pinning . . . . .	38
3.7	Ανίχνευση HTTP cookies . . . . .	39
3.8	Βάση Δεδομένων . . . . .	41
<b>4</b>	<b>Αξιολόγηση Αποτελεσμάτων</b>	<b>45</b>
4.1	Τρόπος συλλογής δεδομένων . . . . .	45
4.1.1	Συλλογή δεδομένων από τη πλοήγηση του χρήστη στο διαδίκτυο	45
4.1.2	Συλλογή δεδομένων με τη χρήση αυτοματοποιημένου μηχανισμού	46
4.2	Γραφική ανάλυση αποτελεσμάτων . . . . .	47
4.2.1	Γραφικές παραστάσεις με βάση το πλήθος υποβολών . . . . .	47
4.2.2	Γραφικές παραστάσεις με βάση τον κίνδυνο . . . . .	48
4.2.3	Γραφικές παραστάσεις με βάση το domain . . . . .	53
4.3	Γραφική απεικόνιση του μηχανισμού . . . . .	56
<b>5</b>	<b>Επίλογος</b>	<b>59</b>
5.1	Σύνοψη και συμπεράσματα . . . . .	59
5.2	Μελλοντικές επεκτάσεις . . . . .	60
	<b>Βιβλιογραφία</b>	<b>61</b>



# Κεφάλαιο 1

## Εισαγωγή

Το Διαδίκτυο ξεπέρασε τα κλασικά μέσα επικοινωνίας και οδηγεί την ανθρωπότητα σε μία νέα εποχή, στην οποία αυτό θα παίζει πρωταγωνιστικό ρόλο στη δημιουργία, τη διακίνηση και την επεξεργασία πληροφοριών και δεδομένων, επηρεάζοντας ποικιλότροπα τη ζωή του ατόμου. Η εύκολη πρόσβαση σε ένα τεράστιο όγκο δεδομένων, η καλύτερη οργάνωση της δημόσιας διοίκησης, η ευκολία στις συναλλαγές μεταξύ ατόμων και εταιρειών καθώς η ψυχαγωγία και η επικοινωνία είναι μερικές από τις θετικές προεκτάσεις του Διαδικτύου. Επακόλουθο αυτού, είναι η εισροή ατόμων με συμφέροντα τα οποία διαχειρίζονται τα δεδομένα. Τέτοια άτομα είναι από κυβερνητικοί οργανισμοί, τρίτοι που έχουν οικονομικά οφέλη καθώς μέχρι και άτομα με κακόβουλες διαθέσεις[1]. Είναι αποδεκτό το γεγονός ότι παροχείς υπηρεσιών (π.χ. *Youtube*), παροχείς περιεχομένου (π.χ. *Google*, *Facebook*) και άλλες υπηρεσίες τρίτων (π.χ. *DoubleClick*) συλλέγουν τεράστιο όγκο από προσωπικές πληροφορίες όταν ένας χρήστης περιηγείται στο διαδίκτυο. Κατα κύριο λόγο, αυτός ο όγκος πληροφορίας χρησιμοποιείται από διαφημιστικές εταιρείες για εμπορικούς λόγους [2], [3] όπως για παράδειγμα για στοχευμένη διαφήμιση προϊόντος, για σύγκριση τιμών [4], [5] μεταξύ των προϊόντων κλπ. Στις περισσότερες των περιπτώσεων, ο χρήστης δε γνωρίζει το πλήθος των δεδομένων που συλλέγονται για αυτόν καθώς και ποια άτομα τα διαχειρίζονται.

Στο παρόν κεφάλαιο, θα γίνει μία αναφορά του προγράμματος Privacy Flag Project καθώς θα αναλύσουμε τους σημαντικότερους κινδύνους που έχουν ανιχνευθεί οι οποίοι αποτελούν ιδιαίτερη απειλή για τα προσωπικά δεδομένα. Πολλές είναι οι περιπτώσεις όπου οι προσωπικές πληροφορίες δίνονται εθελοντικά από το χρήστη (π.χ. συμπληρώνοντας μια form), αλλά τις περισσότερες φορές η συλλογή γίνεται εμμέσως χωρίς τη γνώση του ατόμου με τρόπους όπως αναλύοντας την *IP* επικεφαλίδα, την *HTTP* αίτηση, τα ερωτήματα που θέτει στις μηχανές αναζήτησης κλπ. Αυτούς τους μηχανισμούς μπορούμε να τους χωρίσουμε σε τρεις βασικές κατηγορίες που είναι, το **Απόρρητο Επικοινωνίας** δηλαδή απειλές που μπορεί να προκύψουν κατά

τη διάρκεια εγκατάστασης μίας ασφαλούς σύνδεσης μεταξύ χρήστη και δικτυακού τόπου, **Μηχανισμοί που θέτουν σε κίνδυνο την ιδιωτικότητα**, σε αυτό περιλαμβάνονται μηχανισμοί όπως τα *Cookies*, *Device fingerprint* κλπ, καθώς και τέλος τις **Δυνητικά Ανασφαλείς Τεχνολογίες**, δηλαδή τεχνολογίες οι οποίες στις περισσότερες των περιπτώσεων λειτουργούν ως εργαλεία για το χρήστη, αλλά πολλές φορές χρησιμοποιούνται προκειμένου να αποκτήσουν πληροφορίες για ένα χρήστη. Τέτοιες τεχνολογίες μπορεί να είναι pdf, flash, java κλπ.

## 1.1 Privacy Flag

Η συλλογή όλο και περισσότερων προσωπικών δεδομένων καθώς και η εμπορευματοποίηση αυτών χωρίς τη γνώση του χρήστη αποτελεί ένα από τα μείζονα προβλήματα των τελευταίων ετών. Η Ευρωπαϊκή Ένωση έχει αναλάβει ηγετικό ρόλο για τη προστασία των προσωπικών δεδομένων, συγκεκριμένα χρηματοδοτεί το ερευνητικό πρόγραμμα Privacy Flag [6]. Σκοπός του συγκεκριμένου project είναι η έρευνα και ο συνδυασμός των δυνατοτήτων της τεχνικής του crowdsourcing, της τεχνολογίας πληροφοριών και επικοινωνίας καθώς και της νομικής εμπειρογνωμοσύνης για τη προστασία της ιδιωτικότητας των χρηστών όταν επισκέπτονται ιστοτόπους, χρησιμοποιούν εφαρμογές των smartphones ή βιώνουν σε ένα περιβάλλον όπως εκείνο των σύγχρονων “έξυπνων” πόλεων. Στόχος είναι η δημιουργία συστημάτων τα οποία θα δίνουν τη δυνατότητα στους χρήστες να παρακολουθούν και να ελέγχουν την ιδιωτική τους ζωή με έναν φιλικό προς αυτούς τρόπο, είτε με τη δημιουργία μίας εφαρμογής για smartphone, είτε με τη δημιουργία ενός add-on για τα προγράμματα περιήγησης, είτε με τη δημιουργία μίας ιστοσελίδας, τα οποία θα είναι συνδεδεμένα με μία κοινή βάση δεδομένων. Η προστασία της ιδιωτικής ζωής είναι θεμελιώδες δικαίωμα κάθε πολίτη και οτιδήποτε μπορεί να τον παρακολουθεί χωρίς τη θέληση του θα πρέπει να μπλοκάρεται. Οι χρήστες θα πρέπει να γνωρίζουν τι γίνεται με το ψηφιακό τους αποτύπωμα στο διαδίκτυο καθώς και από ποιους χρησιμοποιείται.

## 1.2 Απόρρητο Επικοινωνίας

Μερικές δεκαετίες πριν, το ζήτημα της παρακολούθησης του χρήστη στο διαδίκτυο δεν είχε απασχολήσει τόσο τους ερευνητές, τους καταναλωτές και τις διαφημιστικές εταιρείες[7]. Τα χρόνια που μεσολάβησαν, η παρακολούθηση έχει πάρει πραγματικά μεγάλες διαστάσεις και αποτελεί επιτακτική ανάγκη για την εύρεση τρόπων αντιστάθμισης αυτού του “φαινομένου”. Βασική προϋπόθεση για τη προστασία του χρήστη, είναι η δημιουργία μίας ασφαλούς σύνδεσης. Στον παρόν κεφάλαιο θα μελετήσουμε κινδύνους που σχετίζονται με την ασφαλή επικοινωνία μέσα σε ένα δίκτυο

υπολογιστών ή στο διαδίκτυο γενικότερα, καθώς αποτελεί το βασικότερο κομμάτι για τη διασφάλιση των προσωπικών δεδομένων.

### 1.2.1 Κρυπτογράφηση Δεδομένων (*Data Encryption*)

Η κρυπτογράφηση (*data encryption*) των δεδομένων είναι η “κλειδαριά” που κρατάει την ψηφιακή δραστηριότητα του χρήστη ασφαλή. Αποτελεί ίσως το σημαντικότερο μέρος για την προστασία των προσωπικών δεδομένων. Η μέθοδος της κρυπτογράφησης είναι η μετατροπή των ψηφιακών δεδομένων σε μια άλλη μορφή, ονομάζεται *ciphertext*, στην οποία πρόσβαση δεν έχει κανένας άλλο εκτός από τα εξουσιοδοτημένα μέρη (άτομα). Στις μέρες μας, η κρυπτογράφηση είναι μια από τις πιο δημοφιλείς και αποτελεσματικές μεθόδους ασφάλειας των δεδομένων, η οποία χρησιμοποιείται από τους οργανισμούς, αλλά και από δικτυακούς τόπους που διαχειρίζονται ευαίσθητες προσωπικές πληροφορίες. Πρωταρχικός σκοπός της κρυπτογράφησης είναι να προστατεύσει το απόρρητο των ψηφιακών δεδομένων που αποθηκεύονται σε ένα σύστημα υπολογιστών ή μεταδίδονται μέσω του διαδικτύου (*Internet*). Οι σύγχρονοι αλγόριθμοι κρυπτογράφησης είναι ζωτικής σημασίας για τη διασφάλιση των συστημάτων πληροφορικής και των επικοινωνιών, δεδομένου ότι μπορούν να προσφέρουν εμπιστευτικότητα. Η απουσία κρυπτογράφησης μπορεί να οδηγήσει στην παρακολούθηση των εισερχόμενων και εξερχόμενων δεδομένων σε διάφορες εφαρμογές, όπως ιστοσελίδες, ηλεκτρονικά ταχυδρομεία, κοινωνικά δίκτυα κλπ κάτι το οποίο θα καταστήσει τη χρήση αυτών των εφαρμογών επικίνδυνη. Στις ιστοσελίδες αυτή υλοποιείται με τη χρήση του πρωτοκόλλου επικοινωνίας *HTTPS* ή *HTTP Secure*. Το *HTTPS* είναι ένας μηχανισμός για τη κρυπτογράφηση μιας *HTTP* συνεδρίας μέσω του *Transport Layer Security*. Ιδανικά θα έπρεπε όλες οι ιστοσελίδες να χρησιμοποιούν το *HTTPS* πρωτόκολλο, αλλά ειδικότερα οι ιστοσελίδες οι οποίες συλλέγουν ευαίσθητες προσωπικές πληροφορίες, όπως στοιχεία τραπεζικών λογαριασμών, κωδικούς πρόσβασης κλπ. Τα μοντέρνα προγράμματα περιήγησης (*browser*) προσφέρουν μια γραφική αναπαράσταση σχετικά με τη χρήση του *HTTPS* (Σχήμα 1.1), δίπλα από το πεδίο συμπλήρωσης του *URL* που επισκέπτεται ο χρήστης αναγράφεται αν χρησιμοποιείται ή όχι το συγκεκριμένο πρωτόκολλο.

### 1.2.2 HSTS (*HTTP Strict Transport Security*)

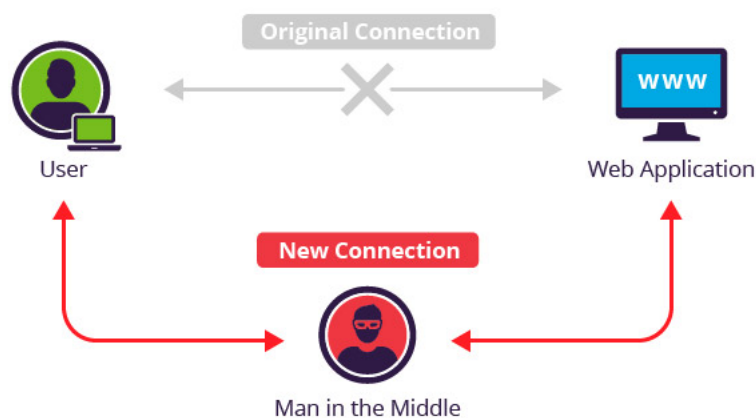
Το *HTTP Strict Transport Security* [8], γνωστό και ως *HSTS*, είναι ένα χαρακτηριστικό ασφαλείας όπου μια ιστοσελίδα “λέει” στο *browser* ότι η επικοινωνία πρέπει να γίνεται μόνο χρησιμοποιώντας το *HTTPS* πρωτόκολλο, σε περίπτωση που χρησιμοποιείται το *HTTP*. Αν πραγματοποιηθεί επικοινωνία μέσω *HTTP*, έχουμε ουσιαστικά τη μη-κρυπτογραφημένη έκδοση, έτσι γίνεται ανακατεύθυνση και η επικοινωνία πραγματοποιείται μέσω *HTTPS*. Με αυτό το τρόπο αποφεύγουμε ένα από



ΣΧΗΜΑ 1.1: Μέθοδος Κρυπτογράφησης.

τους σημαντικότερους κινδύνους, γνωστός και ως *man-in-the-middle attack* [9], όπου η ανακατεύθυνση θα μπορούσε να αξιοποιηθεί με σκοπό να οδηγήσει ένα χρήστη σε μία κακόβουλη ιστοσελίδα, αντί της ασφαλούς έκδοσης της αρχικής.

Αναφέρουμε ένα παράδειγμα, έστω ότι ένας χρήστης βρίσκεται σε ένα αεροδρόμιο και έχει ελεύθερη πρόσβαση στο *WIFI* του αεροδρομίου, ελέγχει το τραπεζικό του υπόλοιπο και πληρώνει κάποιους λογαριασμούς. Το σημείο πρόσβασης που χρησιμοποιεί είναι στην πραγματικότητα το λάπτοπ ενός *hacker*, που έχει διακόψει την αρχική *HTTP* αίτηση και έχει ανακατευθύνει τον χρήστη σε ένα κλώνο του ιστότοπου της τράπεζας αντί του πραγματικού ιστότοπου. Με αυτό το τρόπο τα προσωπικά δεδομένα είναι εκτεθειμένα στον *hacker*. Με τη χρήση της Αυστηρής Ασφάλειας Μεταφοράς (*Strict Transport Security*) επιλύεται το πρόβλημα, με τη προϋπόθεση ότι αν έχουμε πρόσβαση στον ιστότοπο της τράπεζας μια φορά με τη χρήση του πρωτοκόλλου *HTTPS* και η ιστοσελίδα της τράπεζας υποστηρίζει τη τεχνολογία *HSTS*, τότε ο *browser* θα χρησιμοποιεί αυτόματα μόνο *HTTPS* πρωτόκολλο επικοινωνίας, το οποίο αποτρέπει τους *hackers* από μια επίθεση του είδους *man-in-the-middle* (Σχήμα 1.2).



ΣΧΗΜΑ 1.2: Man-in-the-middle επίθεση.

### 1.2.3 Μέθοδοι Κρυπτογράφησης (*cipher suite*)

Οι μέθοδοι κρυπτογράφησης αποτελούν και αυτοί ένα δομικό στοιχείο το οποίο είναι απαραίτητο για τη προστασία των προσωπικών δεδομένων. Συγκεκριμένα το *cipher*, είναι μια έννοια η οποία είναι συνδυασμένη με τον έλεγχο ταυτότητας, την κρυπτογράφηση καθώς και τη χρήση βασικών αλγορίθμων που χρησιμοποιούνται για να καταστήσουν ασφαλή μια σύνδεση στο δίκτυο, χρησιμοποιώντας το πρωτόκολλο δικτύου *TLS/SSL*. Παρά το γεγονός ότι στις μέρες μας υποστηρίζονται υψηλής ποιότητας αλγόριθμοι κρυπτογράφησης, σφάλματα στις ρυθμίσεις του *server* μπορούν να οδηγήσουν στη χρήση καποιού “αδύναμου” *cipher* το οποίο μπορεί να επιτρέψει σε ένα κακόβουλο άτομο (*attacker*) να αποκτήσει πρόσβαση στο υποτιθέμενο ασφαλές κανάλι επικοινωνίας. Υπάρχει μεγάλος αριθμός διαφορετικών τρόπων κρυπτογράφησης που χρησιμοποιούνται, αλλά καποιοί εξ αυτών είναι ξεπερασμένοι με αποτέλεσμα όταν χρησιμοποιούνται να καθιστούν την επικοινωνία ευάλωτη απο μία πιθανή επίθεση. Κάποιες απο τις πιο επικίνδυνες επιθέσεις στις μεθόδους κρυπτογράφησης που έχουν σημειωθεί είναι η επίθεση *Heartbleed* (*Heartbleed attack 2014*) [10] και η *DROWN* (*Aviram, et al. 2016*) [11]. Στη πρώτη, μια σοβαρή ευπάθεια στη βιβλιοθήκη του κρυπτογραφικού λογισμικού *OpenSSL* [12] είχε ως αποτέλεσμα την κλοπή προστατευμένων πληροφοριών. Στη δεύτερη, οι επιτιθέμενοι κατάφεραν να σπάσουν την κρυπτογράφηση με αποτέλεσμα να μπορούν να διαβάσουν ή και να κλέψουν ευαίσθητες επικοινωνίες, συμπεριλαμβανομένων των κωδικών πρόσβασης, αριθμών πιστωτικών καρτών, οικονομικών στοιχείων κλπ.

### 1.2.4 Αλυσίδα Εμπιστοσύνης

Στη κρυπτογραφία, το *X.509* είναι ένα σημαντικό πρότυπο για την υποδομή ενός δημόσιου κλειδιού, το οποίο χρησιμοποιείται για τη διαχείριση των ψηφιακών πιστοποιητικών, του δημόσιου κλειδιού κρυπτογράφησης καθώς αποτελεί και ένα βασικό μέρος του πρωτοκόλλου *Transport Layer Security* το οποίο χρησιμοποιείται για την ασφαλή επικοινωνία στο διαδίκτυο. Το *Certification Chain* ορίζεται ως η επικύρωση της εμπιστοσύνης ενός πιστοποιητικού *X.509* όταν αυτό συγκρίνεται με ένα σημείο πιστοποίησης όπως είναι το *root certificate*. Για να είναι ένα πιστοποιητικό αξιόπιστο, και γενικότερα για να μπορέσει να δημιουργηθεί μια ασφαλής σύνδεση, το εν λόγω πιστοποιητικό πρέπει να έχει εκδοθεί από ένα έμπιστο *Certification Authority (CA)*. Η δομή μιας αλυσίδας είναι μια διαδικασία δόμησης μιας αλυσίδας εμπιστοσύνης, ή διαδρομής πιστοποίησης, από τη τελική πιστοποίηση μέχρι την *root CA*. Αυτή η αλυσίδα θα επικυρώνει τη διαδρομή πιστοποίησης, ελέγχοντας κάθε πιστοποιητικό σε αυτή τη διαδρομή, από την αρχή μέχρι το τέλος. Αν το *cryptoAPI* διαπιστώσει ότι υπάρχει πρόβλημα με ένα από τα πιστοποιητικά, ή αν δε βρεθεί ένα από αυτά, τότε

η διαδρομή πιστοποίησης απορρίπτεται ως μη έμπιστη. Υπήρξαν αρκετές περιπτώσεις επιθέσεων εναντίον χρηστών, με τη χρήση πλαστών ή λάθος πιστοποιητικών κάτι το οποίο είχε καταστροφικές συνέπειες για την ασφάλεια των προσωπικών δεδομένων.

### 1.2.5 Certificate pinning

Η εμπιστευτικότητα, ενώ πλοηγούμαστε στο διαδίκτυο κατορθώνεται χρησιμοποιώντας τα κρυπτογραφικά πρωτόκολλα *SSL/TLS*. Πολλές φορές ακόμη και αν έχει εφαρμοστεί μια ασφαλής μέθοδος κρυπτογράφησης, ένας εισβολέας (*attacker*) μπορεί να εκμεταλλευτεί ένα εγγενές λάθος της δομής της πιστοποίησης (*Certification*) και να γίνει μία *man-in-the-middle* επίθεση που περιγράψαμε παραπάνω. Ενώ γίνεται εγκατάσταση μιας ασφαλούς σύνδεσης μέσω του *TLS* πρωτοκόλλου, το πρόγραμμα περιήγησης του πελάτη επικυρώνει το πιστοποιητικό του διακομιστή. Ακολουθούνται λοιπόν κάποια βήματα επικύρωσης και ένα από αυτά επαληθεύει εάν το πιστοποιητικό του διακομιστή είναι υπογεγραμμένο από το *root CA* ή από κάποιο ενδιάμεσο *CA*, του οποίου το πιστοποιητικό είναι υπογεγραμμένο από το *root CA*. Στη περίπτωση που ένα πιστοποιητικό απορριφθεί, τότε μια προειδοποίηση εμφανίζεται στο χρήστη. Στα σύγχρονα προγράμματα περιήγησης αν υπάρχει τέτοιου είδους πρόβλημα, δεν εγκαθίσταται η σύνδεση έτσι ώστε να αποφευχθούν πιθανοί κίνδυνοι. Τα δημοφιλή προγράμματα περιήγησης έχουν μια λίστα με περίπου 50 *root CA*. Σε αυτά λοιπόν τα *root CA* επιτρέπεται η υπογραφή κάποιας ενδιάμεσης ή *domain* πιστοποίησης για οποιοδήποτε *domain*, κατά τη διάρκεια δημιουργίας της *domain* πιστοποίησης, η αρχή πιστοποίησης (*CA*) συνήθως διασφαλίζεται ότι το *domain* έχει στη κυριότητα του τη πιστοποίηση του ατόμου που έκανε την αίτηση.

Για παράδειγμα όταν ένας εισβολέας καταφέρνει να πάρει ένα υπογεγραμμένο πιστοποιητικό από κάποια ενδιάμεση αρχή, είναι σε θέση να χρησιμοποιήσει αυτό το πιστοποιητικό και να γίνει *man-in-the-middle* σε μία ασφαλή *TLS* σύνδεση. Με αυτό το τρόπο του δίνεται η δυνατότητα να μπορεί να διαβάσει και να διαχειριστεί τη κίνηση για ένα συγκεκριμένο *domain*, ενώ το πρόγραμμα περιήγησης του χρήστη συνεχίζει να εμφανίζει το ενδεικτικά “πράσινο λουκέτο” που σηματοδοτεί την ασφαλή σύνδεση. Για να μπορέσουν λοιπόν να εμποδίσουν τέτοιου είδους επιθέσεις, εισήχθη η τεχνολογία *Certificate Pinning*. Όταν χρησιμοποιείται το *Certificate Pinning*, το πρόγραμμα περιήγησης (ή μια εφαρμογή) γνωρίζοντας ένα συγκεκριμένο πιστοποιητικό για ένα *domain*, απορρίπτει τη δημιουργία μιας *TLS* σύνδεσης στη περίπτωση που το πιστοποιητικό δεν ταιριάζει με το αρχικό. Τα πιστοποιητικά μπορούν να εισαχθούν με δύο τρόπους. Ο ένας τρόπος είναι να εισαχθούν σε μια εφαρμογή κατά τη διάρκεια της δημιουργίας της, έτσι τα πιστοποιητικά θα αποτελούν ήδη μέρος της εφαρμογής (η *Google* το έχει εφαρμόσει στο *Chrome* για υπηρεσίες της).

Ο δεύτερος τρόπος είναι η προσθήκη των πιστοποιητικών όταν γίνει επικοινωνία για πρώτη φορά. Συγκεκριμένα, όταν ο χρήστης επισκέπτεται για πρώτη φορά ένα domain, τότε αυτό αποθηκεύεται στην εφαρμογή, έτσι στις επόμενες συνδέσεις η εφαρμογή θα αρνηθεί τη δημιουργία μιας ασφαλούς σύνδεσης αν υπάρχουν αλλαγές στο πιστοποιητικό για ένα συγκεκριμένο χρονικό διάστημα. Από τους δύο τρόπους που αναφέρθηκαν νωρίτερα, προτιμότερος είναι ο πρώτος.

### 1.2.6 Public key pinning

Η συγκεκριμένη τεχνολογία λειτουργεί παρόμοια με αυτή του *Certificate Pinning*, με τη διαφορά να έγκειται στο γεγονός ότι αντί η εφαρμογή να αποθηκεύει το πιστοποιητικό *X.509*, αποθηκεύει το δημόσιο κλειδί πιστοποιητικού του domain. Αυτή η προσέγγιση είναι πιο σταθερή, δεδομένου ότι ένα domain αλλάζει το πιστοποιητικό του αλλά δεν αλλάζει το αντίστοιχο δημόσιο κλειδί του. Το *Public Key Pinning* συνεργάζεται με την τεχνολογία *HSTS* (*HTTP Strict Transport Security*) που αναφέραμε παραπάνω.

### 1.2.7 Do Not Track

Το *Do Not Track* (*DNT*) είναι ένας μηχανισμός που χρησιμοποιείται για τη προστασία των προσωπικών δεδομένων κατά τη πλοήγηση μας στο διαδίκτυο, ειδικά από τους διαφημιστές που χρησιμοποιούν ολοένα και πιο εξελιγμένες τεχνολογίες παρακολούθησης του χρήστη. Μεγάλες εταιρείες όπως η *Mozilla* και η *Microsoft* έχουν ήδη εισάγει νέες τεχνολογίες *anti-tracking*. Όταν ένας χρήστης, έστω ότι χρησιμοποιεί το *Firefox Mozilla* [13], επιλέγει να ενεργοποιήσει το μηχανισμό *DNT* στο browser του, ο browser στέλνει ένα ειδικό σήμα σε ιστοσελίδες, εταιρείες ανάλυσης, διαφημιστικές εταιρείες και άλλες διαδικτυακές υπηρεσίες οι οποίες τον παρακολουθούν κατά τη πλοήγηση του. Κάθε φορά που ο χρήστης στέλνει ή παραλαμβάνει δεδομένα στο διαδίκτυο, η αίτηση ξεκινά με ένα μικρό κομμάτι πληροφορίας το οποίο ονομάζεται επικεφαλίδα. Οι επικεφαλίδες περιέχουν πληροφορίες οι οποίες σχετίζονται με πληροφορίες όπως το πρόγραμμα περιήγησης του χρήστη, τη γλώσσα του συστήματος του χρήστη και άλλες τεχνικές πληροφορίες. Ο μηχανισμός *DNT* λειτουργεί χρησιμοποιώντας μια απλή, αναγνωρίσιμη από τη μηχανή επικεφαλίδα η οποία μπορεί να πάρει τρεις διαφορετικές τιμές. Στη περίπτωση που λάβει τη τιμή 1 τότε υποδεικνύει ότι ο χρήστης δεν θέλει να παρακολουθείται κατά τη πλοήγηση του, εάν πάρει τη τιμή 0 τότε ο χρήστης επιτρέπει την παρακολούθηση του στο διαδίκτυο και τέλος αν έχει τη τιμή *null* δηλώνει ότι δεν έχει εκφράσει κάποια προτίμηση. Επειδή λοιπόν αυτό το σήμα είναι μια κεφαλίδα, και όχι ένα *cookie* (αναφορά στην δεύτερη ενότητα), ο χρήστης θα είναι σε θέση να διαγράψει τα *cookies* που έχουν αποθηκευτεί στο browser του χωρίς να επηρεάζει τη λειτουργία του *DNT* μηχανισμού. Παρά

το γεγονός ότι ο μηχανισμός *DNT* έχει υιοθετηθεί ευρέως, ορισμένες ιστοσελίδες μπορεί να αγνοούν ή να παρερμηνεύουν το σήμα αυτό.

### 1.3 Μηχανισμοί που θέτουν σε κίνδυνο την ιδιωτικότητα

Σε αυτό το κεφάλαιο θα ασχοληθούμε με τους ιστορικά πρώτους μηχανισμούς παρακολούθησης, οι οποίοι στηρίζονται σε χρονικό διάστημα μιας συνεδρίας (*session*), είναι αρκετά απλές μέθοδοι χωρίς να εγκυμονούν ιδιαίτερους κινδύνους για ένα χρήστη. Θα ασχοληθούμε επίσης με μηχανισμούς που αποθηκεύουν δεδομένα στον υπολογιστή του χρήστη, αυτοί κατά κύριο λόγο χρησιμοποιούνται περισσότερο, έχουν μεθόδους πολύ πιο προηγμένες από τις μεθόδους που βασίζονται σε χρονικό διάστημα μιας συνεδρίας με αποτέλεσμα να παρουσιάζουν μεγαλύτερη απειλή για τα προσωπικά δεδομένα των χρηστών.

#### 1.3.1 Πληροφορίες που δίνονται απευθείας από το χρήστη

Πριν χρησιμοποιηθούν τα *cookies*, το 1994 [14], ο μόνος τρόπος παρακολούθησης ενός χρήστη εκτός από την *IP* διεύθυνσή του ήταν το πέρασμα ενός αναγνωριστικού από τη μία ιστοσελίδα στην άλλη μέσω του *URL* (με τη μέθοδο *GET*) ή ως μιας τιμής από μια *HTML form* (με τη μέθοδο *POST*) [15]. Αυτό το αναγνωριστικό μπορεί να είναι ένα οποιοδήποτε αλφαριθμητικό το οποίο είναι μοναδικό και μπορεί να παρακολουθεί το χρήστη κατά τη διάρκεια μιας συνεδρίας. Με τη συμπλήρωση μιας *HTML form* μπορεί να εισαχθούν πολύ ευαίσθητες πληροφορίες όπως όνομα, διεύθυνση, κωδικοί ακόμη και στοιχεία πιστοτικών καρτών. Ωστόσο ο χρήστης θα πρέπει να ενημερώνεται για τις πληροφορίες που συλλέγονται γι' αυτόν. Συνήθως, αυτή η τεχνική λειτουργεί μέχρι σήμερα, τα στοιχεία που δίνουμε μέσω μιας *HTML form* είναι διαθέσιμα μόνο κατά τη χρονική διάρκεια μιας συνεδρίας. Αυτό έχει ως επακόλουθο ο μηχανισμός να μη εγγυμονεί μεγάλους κινδύνους εκτός και αν η ιστοσελίδα χρησιμοποιεί κάποια γλώσσα προγραμματισμού όπως η *Javascript* η οποία μπορεί να περάσει αυτό το αναγνωριστικό σε τριτούς [16]. Τέλος, άλλος ένας τρόπος αναγνώρισης του χρήστη είναι να του ζητηθεί να εγγραφεί στην ιστοσελίδα. Τότε, οι πόροι που παρέχονται από τη σελίδα (π.χ. *Webmail*) είναι διαθέσιμοι μόνο στο χρήστη ο οποίος συνδέεται στην ιστοσελίδα αυτή. Κάτι τέτοιο κάνει την αναγνώριση του χρήστη ιδιαίτερα εύκολη και απλή. Υπάρχουν, ωστόσο, δύο σημαντικά ζητήματα που σχετίζονται με αυτή τη μέθοδο. Το πρώτο είναι ότι ο χρήστης πρέπει να πιστοποιείται κάθε φορά που επιθυμεί να μπει στην ιστοσελίδα και δεύτερο, η πιστοποίηση είναι διαθέσιμη μόνο κατά τη χρονική περίοδο μιας συνεδρίας αφού του ο χρήστης έχει συνδεθεί. Σε μια τέτοια κατάσταση, ο χρήστης δε μπορεί να



παρακολουθηθεί χωρίς τη γνώση του, καθώς γνωρίζει ότι όταν είναι συνδεδεμένος, οτιδήποτε κάνει μπορεί και να καταγράφεται.

### 1.3.2 HTTP Cookies

Η πιο γνωστή μέθοδος για την παρακολούθηση ενός χρήστη, είναι τα *cookies*. Είναι μικρά κομμάτια δεδομένων, το μέγεθος τους περιορίζεται στα 4KB, που αποθηκεύονται στη μνήμη του προγράμματος περιήγησης [17]. Όταν ένας χρήστης επισκέπτεται για πρώτη φορά μία ιστοσελίδα, ένα αρχείο (*cookie*) με ένα μοναδικό αναγνωριστικό (το οποίο παράγεται τυχαία) αποθηκεύεται στον υπολογιστή του χρήστη. Τότε, η ιστοσελίδα μπορεί να ανακτήσει αυτό το μοναδικό χαρακτηριστικό, κάθε φορά που ο χρήστης επισκέπτεται τη σελίδα, με τη προϋπόθεση ότι δεν τα έχει διαγράψει από το πρόγραμμα περιήγησης του. Αυτή η μέθοδος είναι ιδιαίτερα γρήγορη, δεν προϋποθέτει καμία αλληλεπίδραση με το χρήστη και τέλος δεν γίνεται αντιληπτή από αυτόν καθώς το πρόγραμμα περιήγησης δεν δείχνει καμία ειδοποίηση σχετικά με το αν ορίστηκαν ή διαβάστηκαν *cookies*. Υπάρχουν, λοιπόν, δύο βασικές κατηγορίες από *cookies* και αυτές είναι τα *Session cookies* τα οποία λήγουν όταν ο χρήστης κλείσει το πρόγραμμα περιήγησης του και τα *persistent cookies* τα οποία λήγουν μετά από ένα συγκεκριμένο χρονικό διάστημα [18].

Ο ορισμός των *HTTP Cookies* στον υπολογιστή του χρήστη γίνεται μέσω μιας *HTTP* απάντησης, η οποία έχει ως όρισμα στην επικεφαλίδα της το *Set-Cookie*. Αυτή η διαδικασία πραγματοποιείται αυτόματα όταν επισκεπτόμαστε ένα διαδικτυακό τόπο που χρησιμοποιεί *cookies*. Κάθε φορά που ο χρήστης φορτώνει μια σελίδα, το πρόγραμμα περιήγησης στέλνει το *cookie* πίσω στον *server* και ενημερώνει την ιστοσελίδα για την δραστηριότητα που είχε ο χρήστης. Αυτά τα αρχεία χρησιμοποιούνται για να βοηθήσουν την πλοήγηση του χρήστη στην ιστοσελίδα χρησιμοποιώντας όλα τα χαρακτηριστικά της όπως συνδέσεις, ρυθμίσεις της γλώσσας που χρησιμοποιεί, τα θέματα και λοιπά χαρακτηριστικά της ιστοσελίδας. Δεν συλλέγουν πληροφορίες από τον υπολογιστή του χρήστη και δεν “ψάχνει” τα αρχεία του. Παρά το γεγονός ότι τα *cookies* δεν μπορούν να μεταφέρουν ιούς και να εγκαταστήσουν κακόβουλο λογισμικό στον υπολογιστή του χρήστη, είναι πολύ πιθανό να αποθηκεύουν ευαίσθητες πληροφορίες το οποίο αποτελεί ιδιαίτερη απειλή για την παραβίαση των προσωπικών μας δεδομένων. Τέλος τα *HTTP cookies* μπορούν να χρησιμοποιηθούν ως μηχανισμοί παρακολούθησης από μόνοι τους ή σε συνδυασμό με άλλες τεχνικές.

### 1.3.3 HTTP Cookies από domain τρίτων

Όπως αναφέραμε και στην προηγούμενη υποενότητα, τα *cookies* έχουν ένα *domain* με το οποίο σχετίζονται. Αν οριστούν *cookies* τα οποία προέρχονται από το ίδιο *domain* με το οποίο είμαστε συνδεδεμένοι, τότε αυτά ονομάζονται *first-party cookies*.

Σε περίπτωση που τα *cookies* ορίζονται από διαφορετικό domain από εκείνο που έχουμε επισκεφθεί τότε ονομάζονται *third-party cookies*. Συγκεκριμένα, ενώ τα *first-party cookies* στέλνονται μόνο στο *server* που τα έχει ορίσει, μια ιστοσελίδα μπορεί να περιέχει εικόνες ή άλλα στοιχεία που είναι αποθηκευμένα σε διακομιστές από άλλα domain (π.χ. *ad banners*). Τα *cookies* που αποστέλλονται μέσω αυτών των στοιχείων καλούνται ως *third-party cookies* και χρησιμοποιούνται κατά κύριο λόγο για διαφημιστικούς σκοπούς, παρακολουθώντας το χρήστη κατά τη διάρκεια πλοήγησης του στον ιστό. Τα περισσότερα προγράμματα περιήγησης, ως προεπιλογή, επιτρέπουν την ύπαρξη των *third-party cookies*. Από αυτό καταλαβαίνουμε λοιπόν ότι η συλλογή δεδομένων είναι ακόμη μεγαλύτερη και ο κίνδυνος αυξάνεται δραματικά.

#### 1.3.4 HTTP cookies τα οποία χρησιμοποιούν το Adobe Flash

Είναι μία νέα κατηγορία *cookies* η οποία λειτουργεί παρόμοια με τα απλά, χρησιμοποιώντας το *Adobe Flash Player* για την αποθήκευση πληροφοριών στον υπολογιστή του χρήστη μέσω των *Local Shared Objects (LSO's)* [19]. Είναι σημαντικό να αναφέρουμε ότι παρουσιάζουν τον ίδιο κίνδυνο με τα απλά *cookies*, αλλά δεν είναι τόσο ευκολό στο να τα αποκλειστούν. Στα περισσότερα προγράμματα περιήγησης υπάρχει η δυνατότητα διαγραφής των *cookies* κάτι το οποίο δεν επηρεάζει τα *Flash cookies*. Αυτού του είδους τα *cookies* είναι προσβάσιμα σε όλα τα προγράμματα περιήγησης που είναι εγκατεστημένα στο σύστημα, καθώς τα *Adobe Flash plugins* χρησιμοποιούν το ίδιο αρχείο αποθήκευσης (αποθηκεύονται στη μνήμη του δίσκου ως *.sol* αρχεία). Αποτέλεσμα της προηγούμενης διαδικασίας είναι ότι ο μηχανισμός αυτός δίνει τη δυνατότητα παρακολούθησης των χρηστών σε όλα τα διαφορετικά προγράμματα περιήγησης που χρησιμοποιούνται στον ίδιο υπολογιστή [20]. Εκτός από αυτό, τα *Flash Cookies* δεν λήγουν μετά το πέρασμα κάποιου χρονικού διαστήματος καθώς έχουν και τη δυνατότητα αποθηκεύσεις μεγαλύτερου όγκου πληροφοριών. Συγκεκριμένα μπορούν να αποθηκεύσουν 100KB δεδομένων, κάτι που καθιστά καλύτερη τη παρακολούθηση ενός χρήστη σε σχέση με τα *HTTP cookies* που χρησιμοποιούν 4KB για την αποθήκευση δεδομένων. Διαπιστώνουμε λοιπόν ότι όσο προχωράμε στην ανάλυση των μηχανισμών οι κίνδυνοι αυξάνονται πολύ.

#### 1.3.5 Supercookies

Έχοντας αναλύσει το τρόπο λειτουργίας ενός *HTTP cookie*, είμαστε σε θέση να αναλύσουμε και το τρόπο λειτουργίας ενός *Supercookie*. Συγκεκριμένα είναι ένα είδος *cookie* παρακολούθησης, αλλά μπορεί να θεωρηθεί περισσότερο επικίνδυνο σε σύγκριση με αυτά που έχουμε περιγράψει μέχρι τώρα. Όπως αναφέραμε νωρίτερα, αν ο χρήστης δεν επιθυμεί την παρακολούθηση του κατά τη διάρκεια περιήγησης του στον ιστό, μπορεί απλά να διαγράψει τα *cookies* που είναι αποθηκευμένα στο *browser*

του. Κάτι τέτοιο δεν μπορεί να συμβεί όταν χρησιμοποιούνται *supercookies*, διότι αυτά δεν είναι “ακριβώς” *cookies* και δεν αποθηκεύονται στο πρόγραμμα περιήγησης του χρήστη. Τα *supercookies* έχουν προέλευση από ένα *domain* ανωτάτου επιπέδου (όπως *.com*). Ένας επιτιθέμενος που έχει υπό τον έλεγχο του μια ιστοσελίδα με κακόβουλο περιεχόμενο μπορεί να θέσει ένα *Supercookie* και πιθανότατα μπορεί να διακόψει ή να “μιμηθεί” το νόμιμο αίτημα ενός χρήστη σε μία άλλη ιστοσελίδα η οποία μοιράζεται το ίδιο Top-Level Domain με εκείνο της ιστοσελίδας που περιέχει κακόβουλο λογισμικό.

### 1.3.6 Zombie cookies και Evercookies

Τα *zombie cookies* είναι *HTTP cookies* τα οποία ξαναδημιουργούνται μετά από τη διαγραφή τους, από αντίγραφα που είναι αποθηκευμένα έξω από το πρόγραμμα περιήγησης. Αυτά αποθηκεύονται απευθείας στον υπολογιστή του χρήστη (π.χ. στην τοπική μνήμη του *Flash*, στο *HTML5 storage* και σε άλλα σημεία αποθήκευσης του υπολογιστή). Όταν το *script* ανιχνεύσει την απουσία του *cookie*, τότε το δημιουργεί ξανά από δεδομένα τα οποία είναι αποθηκευμένα στις θέσεις που αναφέραμε παραπάνω. Ένας τέτοιος μηχανισμός όπως είναι λογικό, κάνει ιδιαίτερα δύσκολη τη διαγραφή των *cookies*. Είναι σημαντικό να αναφέρουμε ότι αυτά τα *cookies* μπορεί να εγκατασταθούν σε οποιοδήποτε πρόγραμμα περιήγησης έστω και αν σε αυτό έχει επιλεχθεί να μην γίνονται δεκτά τα *cookies*, δεδομένου ότι η λειτουργία τους δεν είναι ακριβώς η ίδια με εκείνη των παραδοσιακών *cookies*.

Τέλος το *evercookie* είναι μια εφαρμογή βασισμένη στη *Javascript* η οποία δημιουργήθηκε από τον *Samy Kamkar*[21] και παραγεί τα *zombie cookies*, που αναφέραμε νωρίτερα, στο πρόγραμμα περιήγησης του χρήστη. Αυτός είναι και ο πιο ευέλικτος μηχανισμός παρακολούθησης που χρησιμοποιεί πολλαπλά σημεία αποθήκευσης σε συνδυασμό, συμπεριλαμβανομένων των *Flash cookies* [22], *localStorage* [23], *sessionStorage* [24] και των *ETags* [25]. Διαπιστώνουμε λοιπόν, ότι υπάρχουν ιδιαίτερα εξελιγμένοι τρόποι οι οποίοι χρησιμοποιούνται για την παρακολούθηση των χρηστών στο διαδίκτυο κάτι που έχει ως συνέπεια την ύπαρξη υψηλών κινδύνων για την παραβίαση των προσωπικών μας δεδομένων.

### 1.3.7 Καταγραφή ιστορικού περιήγησης

Ο μηχανισμός αυτός παρακολουθεί ποιες σελίδες επισκέφθηκε ο χρήστης και ποιες όχι. Το *History Sniffing*, χρησιμοποιεί το πρόγραμμα περιήγησης του χρήστη για να καθορίσει εάν ο χρήστης έχει επισκεφθεί σε προηγούμενη χρονική στιγμή κάποιες συγκεκριμένες ιστοσελίδες. Στη συνέχεια μεταδίδει τα αποτελέσματα που σύλλεξε σε διάφορα διαφημιστικά δίκτυα και σε τρίτα άτομα (*Bose v. Interclick, Inc. 2011*).

### 1.3.8 API για την αποθήκευση δεδομένων με τη χρήση SQL ερωτήσεων

Εισάγει ένα σύνολο από APIs τα οποία χρησιμοποιούνται για να μπορέσουν να διαχειριστούν τις βάσεις δεδομένων από τη μεριά του χρήστη που χρησιμοποιούν SQL. Ένας διαφημιστής ή ένας τρίτος (γενικότερα οποιοσδήποτε έχει την ικανότητα να πάρει τα περιεχόμενα που διανέμονται σε πολλαπλές τοποθεσίες) μπορεί να χρησιμοποιήσει ένα μοναδικό αναγνωριστικό, που είναι αποθηκευμένο στις βάσεις δεδομένων του χρήστη με σκοπό να τον παρακολουθεί κατά τη διάρκεια πολλών συνεδριών δημιουργώντας έτσι ένα προφίλ του, όπου χρησιμοποιώντας το μπορεί να γίνει ιδιαίτερα στοχευμένη διαφήμιση [26]. Κάτι τέτοιο σε συνδυασμό με μία ιστοσελίδα η οποία γνωρίζει τη πραγματική ταυτότητα του χρήστη (για παράδειγμα μια e-commerce ιστοσελίδα) θα μπορούσε να επιτρέψει σε τέτοιου είδους ομάδες να επεμβαίνουν σε συγκεκριμένα άτομα στοχευμένα, με συνέπεια να παραβιάζεται η ιδιωτικότητα του χρήστη.

### 1.3.9 Ανίχνευση “δακτυλικού αποτυπώματος” συσκευής

Ο συγκεκριμένος μηχανισμός είναι ένας ιδιαίτερα εξελιγμένος τρόπος παρακολούθησης καθώς έχει τη δυνατότητα να εντοπίσει ή και να εντοπίσει εκ νέου έναν χρήστη (ή μια συσκευή), μέσω της διαμόρφωσης των ρυθμίσεων ή άλλων παρατηρήσιμων χαρακτηριστικών. Τα *fingerprints* χρησιμοποιούνται για τον εντοπισμό και την παρακολούθηση [27] συγκεκριμένων χρηστών και συσκευών στο διαδίκτυο, ακόμη και όταν τα *cookies* είναι απενεργοποιημένα. Η *HTML5* έχει εισάγει κάποιες νέες λειτουργίες, όπως είναι το στοιχείο `<canvas>` που χρησιμοποιείται για το σχεδιασμό γραφικών και κινουμένων σχεδίων μέσα στην ιστοσελίδα χρησιμοποιώντας κάποια κομμάτια κώδικα τα οποία είναι γραμμένα σε *JavaScript*. Κατά κύριο λόγο χρησιμοποιείται η μέθοδος *ToDataURL* [28], η οποία επιστρέφει ένα κωδικοποιημένο, *Base64* κρυπτογραφημένο αλφαριθμητικό που έχει ως περιεχόμενο την απαιτούμενη μορφή του στοιχείου *canvas* (π.χ. *png* εικόνα). Καθώς λοιπόν τα προγράμματα περιήγησης παρουσιάζουν διαφορετικά τα γραφικά μιας ιστοσελίδας, το βασισμένο στο *Base64* κρυπτογραφημένο αλφαριθμητικό θα διαφέρει από πρόγραμμα περιήγησης σε πρόγραμμα περιήγησης. Μία τέτοια διαδικασία μπορεί να εφαρμοστεί για να αναγνωρίζει μοναδικά ένα χρήστη. Υπάρχουν λοιπόν μέθοδοι ανίχνευσης μίας συσκευής όπου είναι βασισμένοι στη *JavaScript*, κάνοντας τη έτσι ένα δυνατό εργαλείο για την ανίχνευση αποτυπωμάτων. Επιπλέον γίνεται δυνατή η παρακολούθηση σε όλα τα διαφορετικά προγράμματα περιήγησης χωρίς να χρειάζεται να παραχθούν *cookies*, ή να συνδεθεί ο χρήστης στην ιστοσελίδα. Ως εκτούτου η πλειοψηφία των χρηστών δεν μπορεί να καταλάβει αν παρακολουθείται ή όχι. Είναι αξιοσημείωτο να αναφέρουμε ότι το Τουρκικό Υπουργείο Παιδείας [29] είναι η μοναδική *.gov* ιστοσελίδα, η οποία χρησιμοποιεί το *device fingerprinting*. Ένα ενσωματωμένο στην ιστοσελίδα

*Flash object* εξάγει και στέλνει πίσω στον *server* πληροφορίες σχετικές με το ποντίκι του χρήστη, το πληκτρολόγιο, το μικρόφωνο, τη κάμερα και τις γραμματοσειρές του συστήματος.

### 1.3.10 Δυνητικά επικίνδυνα APIs του πλαισίου HTML5

Η τεχνολογία *HTML5* με τη δημιουργία ορισμένων *APIs* μπορεί να δημιουργήσει πολύ ισχυρές εφαρμογές με εξαιρετικά χαρακτηριστικά. Ορισμένα βέβαια από αυτά τα *APIs* μπορούν να χρησιμοποιηθούν με τρόπο τέτοιο που να παραβιάζουν τα προσωπικά δεδομένα του χρήστη. Θα παρουσιάσουμε λοιπόν ορισμένα από αυτά και τους κινδύνους που εγγυμονούν. Το *Web Audio API (HTML5 Audio 2007)* [30] για παράδειγμα επιτρέπει στους χρήστες να ελέγχουν τον ήχο στο διαδίκτυο, δίνοντας τη δυνατότητα στους προγραμματιστές να διαλέγουν τις πηγές ήχου, να προσθέτουν ειδικά εφέ καθώς και άλλες δυνατότητες. Όταν λοιπόν δίνονται τέτοιες πληροφορίες το *Web Audio API* μπορεί να εξάγει πληροφορίες για το χρήστη, σε κάθε σελίδα που γίνεται χρήση της διεπαφής *AudioNode*. Επίσης, πληροφορίες χρονισμού μπορεί να συλλεχθούν μέσω του *AnalyserNode* ή μέσω της διεπαφής *ScriptProcessorNode*, ως εκ τούτου οι πληροφορίες που συλλέγονται μπορούν να χρησιμοποιηθούν για την δημιουργία ενός αποτυπώματος του χρήστη.

Άλλο πρόγραμμα του οποίου η χρήση μπορεί να εγγυμονεί κινδύνους είναι το *WebRTC (WebRTC 2011)* [31]. Συγκεκριμένα είναι ένα δωρεάν, ανοιχτού κώδικα *project* το οποίο παρέχει στα προγράμματα περιήγησης και στις εφαρμογές κινητών την επικοινωνία σε πραγματικό χρόνο μέσω απλών *APIs*. Τα *WebRTC APIs* σχεδιάστηκαν για να επιτρέπουν σε *Javascript* εφαρμογές να δημιουργούν συνδέσεις σε πραγματικό χρόνο με κανάλια ήχου, βίντεο, ή/και δεδομένων απευθείας μεταξύ των χρηστών μέσω των *browsers* τους, ή σε διακομιστές που υποστηρίζουν τα πρωτόκολλα *WebRTC*.

Το *Geolocation API* [32] χρησιμοποιείται για την ανάκτηση της γεωγραφικής θέσης μιας συσκευής. Σχεδόν σε όλες τις περιπτώσεις, αυτή η πληροφορία αποκαλύπτει την ακριβή θέση του χρήστη, κάτι που παρουσιάζει ιδιαίτερα μεγάλο κίνδυνο για την ιδιωτικότητα του. Μια τέτοιου είδους εφαρμογή θα πρέπει να παρέχει ένα μηχανισμό που θα διασφαλίζει την ιδιωτικότητα του χρήστη καθώς και ότι δε θα δίνονται πληροφορίες που σχετίζονται με αυτόν χωρίς την ρητή άδεια του. Αξίζει να αναφέρουμε βέβαια ότι στις περισσότερες εφαρμογές που χρησιμοποιείται το *API* γίνεται από πριν ερώτηση στον χρήστη και έχει γνώση για τη χρήση του.

Τέλος το *Web storage (Web storage 2011)* και το *DOM storage (Document Object Model storage)* είναι *web* μέθοδοι και πρωτόκολλα τα οποία χρησιμοποιούνται για την αποθήκευση δεδομένων στο πρόγραμμα περιήγησης του χρήστη. Συγκεκριμένα το *Web storage* υποστηρίζει μια επίμονη διαδικασία αποθήκευσης δεδομένων, παρόμοια με

εκείνη των *cookies* με πολύ περισσότερες δυνατότητες και χωρίς καμία πληροφορία να αποθηκεύεται στην επικεφαλίδα της *HTTP* αίτησης. Ένας διαφημιστής για παράδειγμα θα έχει τη δυνατότητα να χρησιμοποιήσει ένα μοναδικό χαρακτηριστικό που θα παρακολουθεί ένα χρήστη σε πολλές συνεδρίες, δημιουργώντας έτσι ένα προφίλ του χρήστη το οποίο θα του δίνει τη δυνατότητα για στοχευμένη διαφήμιση προϊόντος.

#### 1.3.11 Συμμόρφωση με τη πολιτική προστασίας προσωπικών δεδομένων

Ένας αρκετά αποτελεσματικός τρόπος για την προστασία των προσωπικών δεδομένων του χρήστη είναι να ελεγχθεί αν μια ιστοσελίδα συμμορφώνεται με τις γνωστές πολιτικές απορρήτου, όπως το *eTrust* [33] ή *PrivacyTrust* [34]. Γενικά ο ιδιοκτήτης μιας ιστοσελίδας μπορεί να ζητήσει σε μία από τις γνώστες εταιρείες να ελέγξουν εάν η αρχική του σελίδα είναι σύμφωνη με την πολιτική προστασίας των προσωπικών δεδομένων. Αν αυτό συμβεί, τότε ο ιδιοκτήτης της σελίδας μπορεί να τοποθετήσει ένα σήμα (σφραγίδα) στην ιστοσελίδα του. Υπάρχει βέβαια πλήθος περιπτώσεων, που το σήμα αυτό δίνεται σε μία εταιρεία έπειτα από κάποιους πολύ βασικούς ελέγχους που γίνονται ή ζητώντας από τον ιδιοκτήτη της ιστοσελίδας να δηλώσει τη χρήση των προσωπικών δεδομένων που συλλέγει και τη χρήση των *cookies* στο δικτυακό του τόπο καθώς και με ένα οικονομικό αντάλλαγμα προσφέρετε στην ιστοσελίδα αυτό το σήμα εμπιστοσύνης. Η μόνη πλατφόρμα που αποτέλεσαι εξαίρεση ήταν η *P3P* [35]. Η πλατφόρμα *P3P* (*Platform for Privacy Preferences Project*) είναι ένα πρωτόκολλο που επιτρέπει στις ιστοσελίδες να δηλώσουν την προοριζόμενη χρήση των πληροφοριών που συλλέγουν από τους χρήστες των προγραμμάτων περιήγησης. Σχεδιάστηκε για να δώσει στους χρήστες μεγαλύτερο έλεγχο των προσωπικών τους πληροφοριών. Βέβαια είναι σημαντικό να σημειώσουμε ότι η ανάπτυξή σταμάτησε λίγο αργότερα καθώς είχαν υπάρξει πολύ λίγες εφαρμογές του *P3P*. Ο *Microsoft Internet Explorer* και *Edge* ήταν τα μόνα μεγάλα προγράμματα περιήγησης τα οποία υποστήριζαν το *P3P*. Ο πρόεδρος της *TRUSTe* έχει δηλώσει ότι *P3P* δεν έχει εφαρμοστεί ευρέως λόγω της δυσκολίας και της έλλειψης αξίας.

### 1.4 Δυνητικά ανασφαλείς τεχνολογίες

Τον πρώτο καιρό, όταν οι ιστοσελίδες ήταν απλώς στατικά αρχεία *HTML*, οι σελίδες δεν περιείχαν εκτελέσιμο κώδικα. Στην εποχή μας συχνά περιέχουν μικρά προγράμματα, στα οποία περιλαμβάνονται μικρο εφαρμογές *Java*, χρήση του *ActiveX*, *Adobe Flash* κλπ. Το κατέβασμα και η εκτέλεση τέτοιου κώδικα ιστού είναι προφανώς ένας τεράστιος κίνδυνος ασφαλείας. Σε αυτό το υποκεφάλαιο θα

παρουσιάσουμε αναλυτικά τους μηχανισμούς αυτούς καθώς και τους κινδύνους που παρουσιάζουν.

#### 1.4.1 Portable Document Format (PDF)

Το *Portable Document Format* ή συντομογραφικά *PDF* είναι μια μορφή κειμένου που χρησιμοποιείται για να παρουσιάσει έγγραφα με τρόπο που δεν εξαρτάται από το λογισμικό των εφαρμογών, το υλικό και το λειτουργικό σύστημα. Τα περισσότερα σύγχρονα προγράμματα περιήγησης μπορούν να κάνουν μια προεπισκόπηση των αρχείων *PDF* χωρίς να χρειάζεται να ανοίξουν κάποια εξωτερική εφαρμογή. Παρολαυτά ένα έγγραφο *PDF* μπορεί να περιέχει κακόβουλο κώδικα που μπορεί να εκτελέσει άλλες εφαρμογές χωρίς τη γνώση του τελικού χρήστη. Αυτός ο τρόπος έχει χρησιμοποιηθεί αρκετά τα τελευταία χρόνια και συνολικά υπάρχουν περίπου 500 αναγνωρίσιμα τρωτά σημεία, τα οποία είναι εύκολο να καταλάβουμε ότι αποτελούν ιδιαίτερο κίνδυνο για τα προσωπικά δεδομένα του χρήστη.

#### 1.4.2 Πλατφόρμα πολυμέσων Flash

Το *Flash* είναι μια πλατφόρμα πολυμέσων που αναπτύχθηκε από την *Adobe*, η οποία χρησιμοποιείται συνήθως για βίντεο, κινούμενα σχέδια, παιχνίδια καθώς και σε πολλές άλλες εφαρμογές. Πέρα από την δημιουργία των *cookies* που γίνονται μέσω αυτού, στο *Flash* έχουν ανακαλυφθεί και αρκετά τρωτά σημεία ασφαλείας, συγκεκριμένα πάνω από 500 ευπάθειες έχουν ανακαλυφθεί από το 2005. Πολλοί ειδικοί έχουν πάρει θέση για το ζήτημα, συμβουλεύοντας τους χρήστες είτε στο να μην εγκαταστήσουν το *Flash* ή προτείνοντας τους να χρησιμοποιήσουν εργαλεία τα οποία μπλοκάρουν τη λειτουργία του.

#### 1.4.3 Τεχνολογία Microsoft Silverlight

Το *Microsoft Silverlight* είναι μία εφαρμογή για τη σύνταξη και την λειτουργία “πλούσιων” εφαρμογών, είναι μια εφαρμογή παρόμοια με εκείνη του *Adobe Flash*. Ενώ παλαιότερες εκδόσεις του *Silverlight* είχαν επικεντρωθεί σε *streaming media*, οι τρέχουσες εκδόσεις υποστηρίζουν πολυμέσα, γραφικά και κινούμενα σχέδια δίνοντας στους προγραμματιστές τη δυνατότητα ακόμη και για υποστήριξη άλλων γλωσσών προγραμματισμού όπως η *CLI* καθώς και τη χρήση άλλων εργαλείων. Βέβαια, όπως έχει διαπιστωθεί και με το *Flash* της *Adobe*, έτσι κι εδώ έχουν παρουσιαστεί πολλά κενά ασφαλείας, κάτι που μειώνει αρκετά την εμπιστευτικότητα.

#### 1.4.4 ActiveX

Το *ActiveX* είναι ένα ελαττωματικό πλαίσιο λογισμικού το οποίο δημιουργήθηκε από την *Microsoft* το οποίο προσαρμόζει τις παλαιότερες τεχνολογίες *Component Object Model* (COM) και *Object Linking and Embedding* (OLE) για το κατέβασμα περιεχομένου από το διαδίκτυο, και γενικότερα στο παγκόσμιο ιστό. Τα *ActiveX controls* είναι ουσιαστικά μικρά κομμάτια λογισμικού που έχουν πρόσβαση σε όλο τον υπολογιστή του χρήστη στη περίπτωση που αυτός επιθυμεί να τα εγκαταστήσει και να τα τρέξει. Ενώ ο χρήστης χρησιμοποιεί ως πρόγραμμα περιήγησης τον *Internet Explorer*, ορισμένες ιστοσελίδες μπορεί να ζητήσουν από το χρήστη να εγκαταστήσει τα *ActiveX Controls*, συνέπεια αυτού είναι η χρήση του με κακόβουλους σκοπούς.

#### 1.4.5 Γλώσσα προγραμματισμού Java

Η *Java* είναι μια γλώσσα προγραμματισμού και υπολογιστική πλατφόρμα που κυκλοφόρησε πρώτη φορά από την *Sun Microsystems* το 1995. Το *Java Runtime Environment* (JRE) είναι ένα πακέτο λογισμικού που περιέχει ό,τι απαιτείται για “τρέξει” ένα πρόγραμμα σε γλώσσα προγραμματισμού *Java*. Όπως έχουν δείξει έρευνες υπάρχουν πολλά κενά ασφαλείας στο περιβάλλον *Java runtime*. Συγκεκριμένα πάνω από 400 αναγνωρισμένες ευπάθειες έχουν ανακαλυφθεί τα τελευταία πέντε χρόνια. Ορισμένες από αυτές τις ευπάθειες περιλαμβάνουν την εκτέλεση κώδικα, προβλήματα στη μνήμη καθώς και άλλα προβλήματα. Αυτά τα προβλήματα το καθιστούν ως ένα αναξιόπιστο εργαλείο που μπορεί να πλήξει τα προσωπικά μας δεδομένα.



## Κεφάλαιο 2

# Σχετικές Εργασίες

Όπως είδαμε και στο προηγούμενο κεφάλαιο, το διαδίκτυο είναι μια ατέρμονη πηγή προβλημάτων ασφαλείας και παραβίασης της ιδιωτικής μας ζωής. Πλέον η χρήση του διαδικτύου έχει λάβει πολύ σημαντική θέση στην καθημερινότητα του ανθρώπου, καθώς ο χρήστης έχει τη δυνατότητα να κάνει από μία απλή αναζήτηση πληροφοριών, ανταλλαγή ηλεκτρονικών μηνυμάτων μέχρι και *on-line* συναλλαγών. Επακόλουθο αυτού είναι η υπάρξει και η εισροή σε αυτό το χώρο ατόμων με συμφέροντα, με συνέπεια το διαδίκτυο να μην αποτελεί ιδιαίτερα αξιόπιστο μέρος για τα προσωπικά δεδομένα του χρήστη. Αυξανόμενες ποσότητες τόσο από προσωπικές πληροφορίες όσο και από ευαίσθητες πληροφορίες (π.χ. οικογενειακή, οικονομική κατάσταση κλπ) διαρρέουν σε τρίτους καθημερινά με αποτέλεσμα το πρόβλημα να επιδεινώνεται. Έχουν ανακαλυφθεί πολλές στρατηγικές, μέθοδοι και εργαλεία τα οποία μπορούν να ανιχνεύσουν και να αμυνθούν σε κάποιους από τους γνωστούς μηχανισμούς παρακολούθησης.

Με την πάροδο του χρόνου για κάποιους από τους αρχικούς μηχανισμούς παρακολούθησης βρίσκονταν μία λύση (μία επέκταση που μπορούσε να προστατεύσει τα προσωπικά δεδομένα του χρήστη ή άπλα να τον ενημερώσει για τους πιθανούς κινδύνους). Κάθε φορά που ένας τέτοιος μηχανισμός έρχονταν στο προσκήνιο, ένας νέος μηχανισμός παρακολούθησης εμφανίζονταν επίσης. Στο παρόν κεφάλαιο θα συγκρίνουμε και θα αξιολογήσουμε τους μηχανισμούς που υπάρχουν ήδη. Πρέπει να σημειώσουμε ότι θα ασχοληθούμε με μηχανισμούς που χρησιμοποιούνται στο πρόγραμμα περιήγησης του *Firefox*. Μία επέκταση είναι ουσιαστικά ένα πρόσθετο κομμάτι λογισμικού που επεκτείνει την λειτουργικότητα ενός προγράμματος περιήγησης σε έναν υπολογιστή αυξάνοντας κατά πολύ την εμπειρία του χρήστη. Όταν μια ιστοσελίδα δεν συμπεριφέρεται όπως ο χρήστης αναμένει, το πρόγραμμα περιήγησης παρέχει συνήθως στους χρήστες πρόσθετες πληροφορίες. Ένα παράδειγμα αυτού είναι, όταν ένας χρήστης επισκέπτεται μια ιστοσελίδα της οποίας το SSL πιστοποιητικό δεν είναι έγκυρο, τα σύγχρονα προγράμματα περιήγησης είναι σε θέση

να παρουσιάσουν στο χρήστη πρόσθετες πληροφορίες απευθείας για το συγκεκριμένο θέμα.

## 2.1 Μηχανισμοί Σχετικοί με τη φραγή Υπηρεσιών Διαφήμισης

Όταν ένα πρόγραμμα περιήγησης ζητά τη σελίδα από έναν διακομιστή τότε αυτή επιστρέφεται με την απάντηση του διακομιστή. Σε αυτή την απάντηση δίνονται και άλλες οδηγίες για τη δημιουργία νέων αιτήσεων σε άλλους διακομιστές, οι οποίες θα ολοκληρώσουν τη διαδικασία εμφάνισης του διαδικτυακού τόπου. Αυτά τα πρόσθετα αιτήματα συχνά αποκαλούνται και ως cross-site αιτήσεις. Αυτές λοιπόν οι αιτήσεις γίνονται απο domain διαφορετικό από το αρχικό (αυθεντικό). Τέτοια αιτήματα cross-site [36] συχνά καταλήγουν σε διαφημιστικές εταιρείες και εταιρείες άλλων συμφερόντων αποκτώντας πληροφορίες σχετικές με τις συνήθειες πλοήγησης του χρήστη. Ωστόσο είναι σημαντικό να αναφέρουμε ότι ο χρήστης έχει επίγνωση μόνο για την ιστοσελίδα που επισκέφθηκε και δεν μπορεί να καταλάβει την παραπάνω διαδικασία που περιγράψαμε.

Κάθε ιστοσελίδα η οποία δέχεται αιτήματα cross-site είναι σε θέση να συλλέξει και να χρησιμοποιήσει αυτές τις πληροφορίες. Σε ορισμένες περιπτώσεις, αυτή η διαδικασία χρησιμοποιείται για την ανάλυση της κίνησης σε μία ιστοσελίδα. Σε άλλες περιπτώσεις, δεν υπάρχει πρόθεση για παροχή πληροφοριών σε τρίτους άλλα αυτά τα περιεχόμενα χρησιμοποιούνται αποκλειστικά από την ιστοσελίδα για την αύξηση της λειτουργικότητας της (π.χ. ένα Javascript που προσθέτει γραφικά στην ιστοσελίδα). Πρέπει να σημειώσουμε όμως ότι τις περισσότερες φορές αυτού του είδους οι αιτήσεις συλλέγουν πληροφορίες δίνοντας τη δυνατότητα στους διαφημιστές να “συνδέσουν” τις ξεχωριστές συνεδρίες περιήγησης του χρήστη και να δημιουργήσουν ένα προφίλ γι’ αυτόν.

Τα εργαλεία τα οποία βασίζονται στο *Blacklist*, γενικά δεν είναι εύκολα κατανοητά στο τρόπο λειτουργίας τους. Οι χρήστες αυτών των εργαλείων, ίσως δεν μπορούν να καταλάβουν ποίοι κίνδυνοι υπάρχουν για τα προσωπικά τους δεδομένα και το μόνο που επιθυμούν είναι να προφυλαχθούν από πιθανή παραβίαση της ιδιωτικής τους ζωής. Με μία *whitelist* λύση, είναι ακόμη πιο σημαντικό να αναγνωρίσουμε ότι πολλοί χρήστες που επιθυμούν ένα υψηλό επίπεδο προστασίας της ιδιωτικής τους ζωής δεν έχουν πλήρη κατανόηση του τι είναι τα αιτήματα cross-site. Ωστόσο πολλές επεκτάσεις των προγραμμάτων περιήγησης έχουν ξεπεράσει το εμπόδιο κατανόησης του χρήστη και παρέχουν χρήσιμες υπηρεσίες για την διασφάλιση των προσωπικών τους δεδομένων. Σε αυτό το σημείο, πρέπει να δώσουμε μία μικρή εξήγησή των όρων *Blacklist* και *Whitelist* με ένα απλό παράδειγμα. Ας υποθέσουμε ότι θέλουμε

να απορρίπτουμε αυτόματα εισερχόμενες κλήσεις στο τηλέφωνο, θα μπορούσαμε να έχουμε μια *Blacklist* που θα περιείχε τα τηλέφωνα π.χ. διαφημιστικών εταιρειών με αποτέλεσμα να μην μπορούν να μας καλέσουν ή θα μπορούσαμε να έχουμε μία *Whitelist* που θα περιείχε τηλέφωνα φίλων μας, και με αυτό το τρόπο θα ήταν δυνατό μόνο αυτοί να μπορούν να μας καλέσουν.

### 2.1.1 Επέκταση Request-Policy

Το Request-Policy [37] είναι μία επέκταση του *Mozilla* που παρέχει μία λίστα επιτρεπόμενων (*whitelist*) *cross-site* αιτημάτων. Όλα τα αιτήματα τα οποία δεν προορίζονται από το χρήστη αποκλείονται από προεπιλογή. Πολλές *cross-site* αιτήσεις είναι πλήρως εκούσιες από το χρήστη. Για παράδειγμα, οι χρήστες συχνά ακολουθούν συνδέσεις από τη μία ιστοσελίδα στην άλλη, έτσι τέτοιου είδους αιτήσεις δεν θα πρέπει να μπλοκάρονται. Αυτή η επέκταση, προκειμένου να έχει όσο το δυνατόν μεγαλύτερη ακρίβεια, σχεδιάστηκε να μπλοκάρει όσο το δυνατόν περισσότερες αιτήσεις διαφορετικού τύπου περιεχομένου γίνεται. Με αυτό το τρόπο ελαχιστοποιεί την πιθανότητα παράλειψης ενός αιτήματος τύπου *cross-site* το οποίο θα μπορούσε να οδηγήσει σε ένα κενό ασφαλείας το οποίο μπορεί να πλήξει τα προσωπικά μας δεδομένα. Ωστόσο υπάρχουν αιτήματα που είναι πολύ δύσκολο να τα διαχειριστεί το *Request-Policy*, συγκεκριμένα όταν ένας χρήστης κάνει κλικ σε ένα σύνδεσμο *cross-site* τότε η επέκταση πρέπει να επιτρέψει αυτό το αίτημα. Υπάρχουν βέβαια και κάποιοι σύνδεσμοι που όταν ενεργοποιηθούν (π.χ. *Javascript*) ανακατευθύνει το πρόγραμμα περιήγησης σε μία διαφορετική ιστοσελίδα, μία τέτοια ενέργεια όπως είναι λογικό δε μπορεί να ανιχνευθεί ως ενέργεια του χρήστη με αποτέλεσμα να μπλοκάρεται. Ακόμη πιο δύσκολη είναι η διαχείριση *cross-site* αιτημάτων που εκτελούνται από *plugins* του προγράμματος περιήγησης (π.χ. *Flash* και *Java*) καθώς και από άλλες επεκτάσεις του προγράμματος περιήγησης τα οποία μπορεί να παρακάμπτουν την *whitelist* της επέκτασης *Request-Policy*. Για να συνοψίσουμε, η συγκεκριμένη επέκταση είναι ένας πολύ καλός μηχανισμός για τη προστασία των προσωπικών μας δεδομένων καθώς μπλοκάρει όλα τα *cross-site* αιτήματα, δηλαδή αιτήματα σε τρίτα άτομα, αλλά ο χρήστης μπορεί να διατηρήσει την *whitelist* του.

### 2.1.2 Επέκταση Adblock Plus

Ακόμη μία επέκταση η οποία ανήκει στην ίδια κατηγορία και είναι ιδιαίτερα ισχυρή είναι το Adblock Plus [38]. Είναι μία ανοιχτού κώδικα επέκταση η οποία υποστηρίζεται από το *Mozilla*, δημιουργήθηκε το 2016 και έχει πάνω από ένα εκατομμύριο χρήστες. Συγκεκριμένα μπλοκάρει όλες τις ανεπιθύμητες διαφημίσεις όπως για παράδειγμα *video* διαφημίσεων στο *Youtube*, διαφημίσεις στο *Facebook*, διαφημιστικά *banner* καθώς και *pop-ups*. Το *Adblock Plus* είναι ιδιαίτερα ισχύρο εργαλείο καθώς

μπορεί να προσδιορίσει τις αποδεκτές διαφημίσεις. Επιπλέον, μπορεί να ρυθμιστεί έτσι ώστε να μπλοκάρει *domain* τα οποία περιέχουν κακόβουλο λογισμικό, μη επιτρέποντας την εξάπλωση τους, καθώς μπορεί να απενεργοποιήσει και όλους τους μηχανισμούς παρακολούθησης. Τέλος, έχει την ικανότητα να απενεργοποιεί όλα τα *button* των μέσων κοινωνικής δικτύωσης. Το τελευταίο είναι και ιδιαίτερα σημαντικό, καθώς το Facebook [39] έχει παραδεχτεί ότι ενώ επισκεπτόμαστε μία ιστοσελίδα η οποία περιέχει το κουμπί Like (Σχήμα 2.1), πληροφορίες οι οποίες σχετίζονται με το αναγνωριστικό του Facebook μας στέλνονται στο διακομιστή για να μας δείξει ποιός από τους φίλους μας έχει κάνει Like στην ίδια σελίδα. Την ίδια στιγμή, κάποιες πρόσθετες πληροφορίες οι οποίες σχετίζονται με το πρόγραμμα περιήγησης αποστέλλονται στο Facebook μαζί με τη διεύθυνση της ιστοσελίδας στην οποία βρισκόμασταν. Η πληροφορία που συλλέγονται από την Atlas [40], από διάφορες πηγές (π.χ. Facebook) χρησιμοποιούνται για την πώληση διαφημίσεων από το Facebook για προϊόντα που ανήκουν σε άλλες οντότητες. Το Adblock Plus λοιπόν έχει την δυνατότητα να απενεργοποιήσει μία τέτοια διαδικασία προστατεύοντας τα προσωπικά δεδομένα των χρηστών. Τέλος είναι σημαντικό να αναφέρουμε ότι οι διαφημιστές πλέον ενσωματώνουν το διαφημιστικό τους περιεχόμενο με το *video* (π.χ. Youtube) με αποτέλεσμα να μην είναι δυνατός ο αποκλεισμός της διαφήμισης.



ΣΧΗΜΑ 2.1: Like Button.

### 2.1.3 Επέκταση Privacy Badger

Τέλος το Privacy Badger [41] είναι και αυτό μία επέκταση του προγράμματος περιήγησης που μπλοκάρει *cross-site* αιτήσεις, δηλαδή σταματά τους διαφημιστές και άλλους τρίτους που κρυφά παρακολουθούν την πλοήγησή του χρήστη στο διαδίκτυο καθώς συλλέγουν πληροφορίες γι' αυτόν όπως για το ποιες σελίδες επισκέφθηκε. Εάν ένας διαφημιστής διαπιστωθεί ότι παρακολουθεί τον χρήστη σε πολλαπλές ιστοσελίδες χωρίς την άδειά του, το Privacy Badger μπλοκάρει αυτόματα τον διαφημιστή μη επιτρέποντας του τη φόρτωση οποιουδήποτε επιπλέον περιεχομένου στο πρόγραμμα περιήγησης του. Είναι σημαντικό να αναφέρουμε ότι ένα μέρος του κώδικα αυτής της επέκτασης βασίστηκε στο Adblock Plus. Η επέκταση Privacy Badger σχεδιάστηκε για να αναλύει αυτόματα και να μπλοκάρει οποιοδήποτε στοιχείο ακολουθεί τον χρήστη χωρίς την συγκατάθεση του ίδιου. Συγκεκριμένα, όταν προβάλλεται μία

ιστοσελίδα, αυτή αποτελείται συχνά από περιεχόμενα που προέρχονται από πολλές διαφορετικές πηγές. Για παράδειγμα, μία ιστοσελίδα ειδήσεων μπορεί να φορτώσει το πραγματικό ειδησεογραφικό κείμενο από την εταιρεία ειδήσεων, τις διαφημίσεις από την εταιρεία διαφημίσεων και το τμήμα των σχολίων από μία διαφορετική εταιρεία που παρέχει την εν λόγω υπηρεσία. Η παρούσα επέκταση λοιπόν παρακολουθεί όλη αυτή τη διαδικασία, έτσι αν κατά τη διάρκεια περιήγησης του χρήστη στο διαδίκτυο διαπιστωθεί ότι η ίδια πηγή τον παρακολουθεί στις διάφορες ιστοσελίδες που επισκέπτεται, τότε αυτός ο μηχανισμός “λέει” στο πρόγραμμα περιήγησης να μην φορτώνει πλέον οποιοδήποτε περιεχόμενο προέρχεται από τη συγκεκριμένη πηγή. Σε πιο τεχνικό επίπεδο, το *Privacy Badger* “σημειώνει” τα *domain* τρίτων τα οποία ενσωματώνουν εικόνες, *scripts* και διαφημίσεις στις ιστοσελίδες που επισκέπτεται ο χρήστης. Εάν διαπιστωθεί ότι κάποιος τρίτος παρακολουθεί ένα χρήστη χωρίς την αδεία του, με τη χρήση ενός μοναδικού αναγνωριστικού όπως *cookie* (ή *supercookies* και *canvas fingerprinting*), συλλέγοντας πληροφορίες γι αυτόν, τότε η επέκταση μπλοκάρει αυτόματα το περιεχόμενο που φορτώνεται από τρίτους. Πρέπει να σημειώσουμε ότι αρκετές φορές κάποια *domain* τρίτων φορτώνουν περιεχόμενο σε μία ιστοσελίδα δίνοντας της μεγαλύτερη λειτουργικότητα όπως φόρτωση χαρτών, εικόνων ή *stylesheets*. Στη τελευταία περίπτωση, ο μηχανισμός θα πρέπει να επιτρέψει την *cross-site* αίτηση στο *domain* τρίτου. Τέλος να σημειώσουμε ότι το *Privacy Badger* δεν μπλοκάρει όλες τις διαφημίσεις αλλά αποτρέπει εκείνες οι οποίες προβάλλονται χωρίς τη συναίνεση του χρήστη, καθώς δεν είναι μία παραδοσιακή *ad block* επέκταση.

## 2.2 Μπλοκάρισμα Εκτέλεσης content script

Υπάρχουν αρκετοί μηχανισμοί που χρησιμοποιούνται ευρέως στο διαδίκτυο οι οποίοι είναι δυνητικά ανασφαλείς. Για τους περισσότερους από αυτούς κάναμε αναφορά στο πρώτο κεφάλαιο, αναλύοντας κάποιους από τους κινδύνους που μπορεί να παρουσιάσουν. Είχαμε αναφέρει ότι τα *LSOs* (*Local Shared Objects*) για παράδειγμα χρησιμοποιούνται από το *Adobe Flash* για να αποθηκεύουν δεδομένα στον υπολογιστή του χρήστη, συγκεκριμένα μπορούν να αποθηκεύσουν πάνω από 100KB δεδομένων, κάνοντας την παρακολούθηση ιδιαίτερα επικίνδυνη, καθώς γίνεται συλλογή μεγάλου όγκου πληροφοριών. Επίσης αυτά είναι προσβάσιμα από όλα τα προγράμματα περιήγησης που είναι εγκατεστημένα στον υπολογιστή του χρήστη διότι όλα τα στιγμιότυπα των *Adobe Flash plugins* μοιράζονται τον ίδιο κατάλογο αποθήκευσης. Αποτέλεσμα αυτού είναι ότι ένας χρήστης μπορεί να παρακολουθείται ενώ χρησιμοποιεί διαφορετικά προγράμματα περιήγησης. Όλο και δυσκολότεροι μηχανισμοί παρακολούθησης χρησιμοποιούνται με αποτέλεσμα να υπάρχει ανάγκη για την αντιστάθμιση αυτού του φαινομένου. Στο συγκεκριμένο υποκεφάλαιο θα παρουσιαστούν τρεις επεκτάσεις που υποστηρίζουν τα προγράμματα περιήγησης, οι

οποίες είναι σε θέση να εμποδίσουν την εκτέλεση της *Javascript*, του *Flash* και άλλων περιεχομένων. Είναι ιδιαίτερα σημαντικοί μηχανισμοί, καθώς η χρήση τους παρέχει μεγαλύτερη προστασία στα προσωπικά δεδομένα του χρήστη.

### 2.2.1 Επέκταση NoScript

Το *NoScript* [42] (ή *NoScript Security Suite*) είναι μία δωρεάν και ανοιχτού κώδικα επέκταση για το πρόγραμμα περιήγησης *Mozilla*, δημιουργήθηκε και συντηρείται ενεργά από τον *Giorgio Maone*, μέλος της ομάδας ασφαλείας της *Mozilla*. Η συγκεκριμένη επέκταση μπλοκάρει περιεχόμενα *Javascript*, *Java*, *Flash*, *Silverlight* καθώς και άλλα “ενεργά” περιεχόμενα που ενσωματώνονται σε μία ιστοσελίδα. Αυτή η διαδικασία στηρίζεται στο γεγονός ότι οι κακόβουλες ιστοσελίδες μπορεί να χρησιμοποιούν αυτές τις τεχνολογίες με πολλούς διαφορετικούς επιβλαβείς τρόπους για τον χρήστη. Ο χρήστης έχει τη δυνατότητα να ενεργοποιήσει, σε προσωρινή ή μόνιμη βάση, το περιεχόμενο σε ιστοσελίδες που θεωρεί αξιόπιστες (*whitelist*), δίνοντας τη ρητή του άδεια του. Στη περίπτωση που επιλεχθεί η προσωρινή επίτρεψη των περιεχομένων, μετά τη λήξη της συνεδρίας στη περίπτωση που συναντήσουμε το ίδιο περιεχόμενο, θα το μπλοκάρει η επέκταση.

Πολλές επιθέσεις στον *web browser* απαιτούν την ύπαρξη *script*. Υπάρχει ρύθμιση στο πρόγραμμα περιήγησης η οποία απενεργοποιεί τα *script* μειώνοντας έτσι τις πιθανότητες εκμετάλλευσης των προσωπικών μας δεδομένων. Το μπλοκάρισμα *plug-in* περιεχομένων, βοηθά στην άμβλυνση τυχόν τρωτών σημείων στα *plug-in* που χρησιμοποιούν τεχνολογίες όπως η *Java*, *Flash*, *Acrobat* κλπ (έχουμε κάνει ιδιαίτερη αναφορά στο πρώτο κεφάλαιο). Το *NoScript* μπλοκάρει αυτά τα περιεχόμενα, τοποθετώντας στα στοιχεία που έχουν μπλοκαριστεί ένα εικονίδιο. Όταν ο χρήστης κλικάρει πάνω σε αυτά τα εικονίδια, τότε ενεργοποιούνται τα συγκεκριμένα περιεχόμενα. Τέλος το *NoScript*, παρέχει πρόσθετες άμυνες ενάντια σε πιθανές επιθέσεις όπως το *Clickjacking*, *man-in-the-middle* [9] επιθέσεις χρησιμοποιώντας αντίμετρα τα οποία δεν επηρεάζουν το μπλοκάρισμα των *script* που αναφέραμε νωρίτερα.

### 2.2.2 Επέκταση Ghostery

Το *Ghostery* [43] είναι ένα δωρεάν λογισμικό προστασίας των προσωπικών δεδομένων που υποστηρίζεται από τα προγράμματα περιήγησης, η οποία ανήκει στην *Ghostery Inc*. Μπορεί να παρακολουθεί όλους τους διαφορετικούς διακομιστές που καλούνται από μία συγκεκριμένη ιστοσελίδα και τις “συγκρίνει” με αυτούς που έχει μέσω μίας βιβλιοθήκης δεδομένων (*trackers*). Αν διαπιστωθεί μία τέτοια αντιστοιχία, τότε η επέκταση δείχνει τον *tracker* μέσω μίας λίστας σε ένα *pop-up*. Η συγκεκριμένη επέκταση, αν ρυθμιστεί να εμποδίζει την επικοινωνία με μία ή περισσότερες από αυτές τις εταιρείες, τότε την διακόπτει απευθείας. Επίσης, επιτρέπει στους χρήστες

να εντοπίζουν και να ελέγχουν εύκολα κώδικα *Javascript* (γνωστά και ως “*Tags*” και “*Trackers*”) τα οποία είναι ενσωματωμένα σε μία ιστοσελίδα, αόρατα από τον χρήστη, μη επιτρέποντας σε τρίτους τη συλλογή των συνηθειών περιήγησης του χρήστη με τη χρήση *HTTP cookies* ή άλλων μηχανισμών όπως είναι το *canvas fingerprinting*. Τέλος δημιουργεί “*whitelist*” από ιστοσελίδες που χρησιμοποιούν *third-party script* τα οποία έχουν ως μόνο στόχο να αυξήσουν τη λειτουργικότητα της ιστοσελίδας, μη μπλοκάροντας τα περιεχόμενα αυτά. Ένα ιδιαίτερο χαρακτηριστικό αυτής της επέκτασης είναι ότι όταν ένας *tracker* μπλοκάρετε τότε οποιοδήποτε *cookie* που τοποθετείται από αυτόν δεν είναι προσβάσιμο από κανέναν άλλο εκτός από το χρήστη. Με αυτό το τρόπο δεν μπορεί να διαβαστεί και να κληθεί, καθιστώντας το έτσι ανενεργό.

### 2.2.3 Επέκταση Flashblock

Το *Flashblock* [44] είναι μία επέκταση του *Firefox* για τον *Mozilla*, η οποία φιλτράρει τα περιεχόμενα που είναι ενσωματωμένα σε μία ιστοσελίδα. Η λειτουργικότητα του είναι παρόμοια με εκείνη του *NoScript*. Συγκεκριμένα το *Flashblock* αποτρέπει την εμφάνιση στοιχείων στην ιστοσελίδα που επισκέπτεται ο χρήστης, όπως διαφημιστικά *banner* και *Flash script* τα οποία παραβιάζουν τα προσωπικά δεδομένα του χρήστη. Επιπλέον παρέχει ένα μηχανισμό που επιτρέπει στους χρήστες να εμφανίσουν πιθανά περιεχόμενα που έχει αποκλείσει ή με το πάτημα πάνω σε αυτά τα στοιχεία ή με τη χρήση μιας *whitelist*. Σε σύγκριση με το *NoScript* είναι λίγο πιο αδύναμος σαν μηχανισμός καθώς δεν μπορεί να αντιμετωπίσει άλλους μηχανισμούς παρακολούθησης που μπορεί να εμφανιστούν σε μία ιστοσελίδα.

## 2.3 Ασφάλεια Επικοινωνίας

Κατά τις πρώτες δεκαετίες η χρήση του διαδικτύου χρησιμοποιούνταν κυρίως από πανεπιστημιακούς ερευνητές για την αποστολή μηνυμάτων μέσω του ηλεκτρονικού ταχυδρομείου και από τις ένοπλες δυνάμεις. Υπό αυτές τις συνθήκες, δεν δίνονταν ιδιαίτερη σημασία στην ασφάλεια. Στις μέρες μας, που εκατομμύρια άνθρωποι χρησιμοποιούν τα δίκτυα για τραπεζικές συναλλαγές, αγορές και άλλες δραστηριότητες, ανακαλύπτονται διαρκώς αδυναμίες, τοποθετώντας την ασφάλεια των δικτύων ως ένα από τα μεγαλύτερα προβλήματα στις μέρες μας. Η ασφάλεια των δικτύων προσπαθεί να εξασφαλίσει ότι αδιάκριτα (ή κακόβουλα άτομα) δε θα μπορέσουν να διαβάσουν, ή ακόμη χειρότερα να τροποποιήσουν, χωρίς να γίνουν αντιληπτοί τα μηνύματα που προορίζονται για άλλους παραλήπτες (*man-in-the middle επιθέσεις*). Ασχολείται με όσους προσπαθούν να προσπελάσουν απομακρυσμένες υπηρεσίες τις οποίες δεν είναι εξουσιοδοτημένοι να χρησιμοποιούν, με θέματα πιστοποίησης, δηλαδή αν το άτομο με το οποίο έχουμε έρθει σε επικοινωνία είναι

όντως αυτό και όχι κάποιος τρίτος, καθώς τέλος και με την ηλεκτρονική υπογραφή, είναι κάτι σαν “πιστοποιητικό” το οποίο κατοχυρώνει μία συμφωνία μεταξύ δύο ή περισσότερων ατόμων στο διαδίκτυο.

Τα περισσότερα προβλήματα ασφαλείας προκαλούνται σκόπιμα από κακόβουλα άτομα τα οποία προσπαθούν να αποκομίσουν κάποιο κέρδος, να προσελκύσουν την προσοχή, ή να βλάψουν κάποιον. Αρκετές είναι και οι φορές όπου τέτοια άτομα χρηματοδοτούνται από τρίτους για να αποκτήσουν πρόσβαση σε ευαίσθητα δεδομένα (αναλύσεις εταιρειών, στρατηγικά πλάνα εταιρειών, κυβερνητικές πληροφορίες κλπ). Στο παρόν κεφάλαιο λοιπόν θα παρουσιάσουμε και θα συγκρίνουμε κάποιες επεκτάσεις των προγραμμάτων περιήγησης τα οποία αποτρέπουν τέτοιους κινδύνους, που παραβιάζουν ευαίσθητα προσωπικά δεδομένα του χρήστη.

### 2.3.1 Επέκταση HTTPS Everywhere

Το HTTPS Everywhere [45] έχει παραχθεί από μια συνεργασία του Tor Project [46] και του Electronic Frontier Foundation. Πολλές είναι οι ιστοσελίδες οι οποίες διαχειρίζονται πολύ ευαίσθητες προσωπικές πληροφορίες, χωρίς να υποστηρίζουν κάποιου είδους κρυπτογράφηση. Το *HTTPS Everywhere* δίνει λύση σε αυτό το πρόβλημα, “κάνοντας” αυτόματα τις ιστοσελίδες να χρησιμοποιούν περισσότερο ασφαλή σύνδεση μέσω του *HTTPS* πρωτοκόλλου αντί για *HTTP*, όπου είναι δυνατόν. Επίσης δίνει πληροφορίες για το ποιες ιστοσελίδες χρησιμοποιούν το *HTTPS* πρωτόκολλο. Το *SSL Observatory* είναι ένα χαρακτηριστικό του *HTTPS Everywhere* το οποίο εισήχθη στην έκδοση 2.0.1, η οποία αναλύει τα *public key certificates* για να καθοριστεί εάν οι αρχές πιστοποιητικού έχουν παραβιαστεί και κατ’ επέκταση αν χρήστης είναι ευάλωτος σε μία *man-in-the-middle* επίθεση. Δυστυχώς, πολλές ιστοσελίδες περιλαμβάνουν πολλά περιεχόμενα από *third party domain* τα οποία δεν είναι διαθέσιμα μέσω *HTTPS*. Κάτι τέτοιο έχει ως αποτέλεσμα ο χρήστης να είναι ευάλωτος σε τρίτους οι οποίοι μπορεί να κάνουν ανάλυση της κυκλοφορίας στη συγκεκριμένη ιστοσελίδα μέχρι και να αποκτήσουν πρόσβαση σε προσωπικά του δεδομένα. Είναι σημαντικό να επισημανθεί ότι ένα μέρος κώδικα του *HTTPS Everywhere* έχει βασιστεί στην επέκταση *NoScript*, με τη διαφορά ότι η συγκεκριμένη επέκταση είναι πιο εύκολη στη χρήση της.

### 2.3.2 Επέκταση Web of Trust

Το *Web of Trust (WOT)* [47] είναι μία παγκοσμίως γνωστή ιστοσελίδα η οποία βοηθά το χρήστη να παίρνει συνειδητές αποφάσεις για το αν θα εμπιστευθεί μία ιστοσελίδα ή όχι όταν ψάχνει, αγοράζει ή απλά περιηγείται στο διαδίκτυο. Η υπηρεσία *WOT* ιδρύθηκε το 2006 από τον Sami Tolvanen και τον Timo Ala-Kleemola,



που έγραψαν αυτό το λογισμικό ως μεταπτυχιακοί φοιτητές στο Τεχνολογικό Πανεπιστήμιο Tampere της Φιλανδίας. Με αυτό το τρόπο αυξάνεται η ασφάλεια των προσωπικών μας δεδομένων στο διαδίκτυο. Το WOT είναι βασισμένο στην προσέγγιση *Crowdsourcing*, δηλαδή εκατομμύρια χρήστες “βαθμολογούν” τις ιστοσελίδες που επισκέπτονται μοιράζοντας τις προσωπικές τους εμπειρίες. Αυτό βοηθά το χρήστη να αποφεύγει τις διαδικτυακές απειλές που μόνο στην καθημερινότητα του μπορεί να ανιχνεύσει, όπως διάφορες απάτες, αναξιόπιστες συνδέσεις, μη ασφαλή ηλεκτρονικά καταστήματα κλπ. Αυτό το εργαλείο είναι διαθέσιμο σε δύο μορφές, οι χρήστες μπορούν είτε να αναζητήσουν τη φήμη ενός δικτυακού τόπου στην ιστοσελίδα WOT είτε να κατεβάσουν την επέκταση του προγράμματος περιήγησης η οποία ελέγχει αυτόματα τη βαθμολογία για κάθε ιστοσελίδα που επισκέπτεται ο χρήστης. Συγκεκριμένα η επέκταση δείχνει τη φήμη της ιστοσελίδας με μία ένδειξη δίπλα από τα αποτελέσματα αναζήτησης σε μηχανές αναζήτησης όπως το Google, το Yahoo!, το Bing ή οποιαδήποτε άλλο χρησιμοποιείται. Υπάρχουν τρεις δυνατές ενδείξεις, με πράσινο χρώμα εμφανίζονται οι αξιόπιστες ιστοσελίδες, με κόκκινο χρώμα οι μη αξιόπιστες και με πορτοκαλί οι σελίδες στις οποίες πρέπει να είμαστε προσεκτικοί. Αυτά τα εικονίδια είναι επίσης ορατά δίπλα από τις συνδέσεις σε ιστοσελίδες κοινωνικής δικτύωσης όπως το Facebook και το Twitter, υπηρεσίες ηλεκτρονικού ταχυδρομείου όπως το Gmail και το Yahoo Mail καθώς και δημοφιλείς ιστοσελίδες όπως το Wikipedia. Κάνοντας κλικ στο εικονίδιο που εμφανίζεται ο χρήστης μπορεί να βρει λεπτομέρειες σχετικές με την φήμη της ιστοσελίδας, πληροφορίες σχετικές με την ασφάλεια μίας ιστοσελίδας καθώς και σχόλια των χρηστών.



## Κεφάλαιο 3

# Μηχανισμός Ανίχνευσης Κινδύνων

Σε αυτό το κεφάλαιο γίνεται αναφορά για προβλήματα ασφαλείας και απώλεια προσωπικών δεδομένων από διάφορους ιστότοπους. Η ολοένα αυξανόμενη χρήση του διαδικτύου και αντίστοιχα η αύξηση της παραβίασης των προσωπικών μας δεδομένων, δημιουργεί την ανάγκη για την δημιουργία αντιμέτρων. Γενικότερα η ασφάλεια στο διαδίκτυο μπορεί να υποδιαιρεθεί σε τρία βασικά μέρη. Πρώτον, πως ονομάζονται με ασφάλεια τα αντικείμενα και οι πόροι στο διαδίκτυο, δεύτερον πως εγκαθιδρύονται ασφαλείς συνδέσεις με πιστοποίηση ταυτότητας και τέλος τι συμβαίνει όταν μία ιστοσελίδα στέλνει στο χρήστη κάποιο τμήμα εκτελέσιμου κώδικα. Στο συγκεκριμένο κεφάλαιο θα παρουσιάσουμε έναν μηχανισμό (*add-on*) (για το πρόγραμμα περιήγησης του Mozilla Firefox) ανίχνευσης ορισμένων κινδύνων που αναφέραμε στο πρώτο κεφάλαιο, τον λόγο επιλογής αντιμετώπισης των συγκεκριμένων κινδύνων, τον τρόπο υλοποίησης τους καθώς και τα εργαλεία που χρησιμοποιήσαμε για την υλοποίησή τους.

### 3.1 Πλατφόρμα Nodejs

Για την υλοποίησή του μηχανισμού χρησιμοποιήσαμε το *Nodejs* το οποίο είναι μία πλατφόρμα βασισμένη στο *Chrome's JavaScript runtime*, η οποία δίνει τη δυνατότητα για την εύκολη και γρήγορη δημιουργία διαδικτυακών εφαρμογών καθώς δίνεται και η δυνατότητα επεκτασιμότητας. Το *Nodejs* είναι ένα ανοιχτού κώδικα, *cross-platform* περιβάλλον για την ανάπτυξη *server-side* και *networking* εφαρμογών. Οι εφαρμογές αυτές γράφονται σε *Javascript* και μπορούν να εκτελεστούν στα λειτουργικά συστήματα, *Microsoft Windows* και *Linux*. Ένα από τα πλεονεκτήματά του είναι ότι παρέχει μία πλούσια βιβλιοθήκη διαφόρων *Javascript modules* η οποία δίνει τη δυνατότητα απλοποίησης κατά τη διαδικασία ανάπτυξης μιας διαδικτυακής εφαρμογής. Επίσης υπάρχουν άλλοι δύο βασικοί λόγοι που κάνουν το εργαλείο αυτό ιδιαίτερα ισχυρό. Πρώτον, όλα τα *APIs* της βιβλιοθήκης του *Node.js* είναι ασύγχρονα,

και έτσι δε μπλοκάρονται. Συγκεκριμένα ένας *server* βασισμένος στο *Nodejs* δε περιμένει την επιστροφή δεδομένων από το *API*, αλλά προχωράει στην εκτέλεση του επόμενου *API* και με ένα μηχανισμό που ονομάζεται *event-handler* ενημερώνεται για τα δεδομένα που πήρε από τη κλήση του προηγούμενου *API*. Και δεύτερον, το *Nodejs* είναι πολύ γρήγορο στην εκτέλεση του κώδικα καθώς είναι βασισμένο πάνω στο *Google Chrome's V8 JavaScript Engine* που αναφέραμε και προηγούμενα. Λόγω των πλεονεκτημάτων που μας παρέχουν αυτές οι νέες δυνατότητες, έγινε η επιλογή της συγκεκριμένης πλατφόρμας για την υλοποίηση του μηχανισμού μας.

### 3.1.1 Εγκατάσταση Πλατφόρμας *Nodejs* για τη δημιουργία του *add-on*

Το *add-on SDK* (*Software development kit*) περιλαμβάνει ένα εργαλείο γραμμής εντολών το οποίο χρησιμοποιείται για την αρχικοποίηση, την εκτέλεση, τον έλεγχο και την δημιουργία του πακέτου του *add-on*. Το τρέχον εργαλείο ονομάζεται *JPM* και βασίζεται στο *Nodejs* που αναφέραμε νωρίτερα. Πρέπει να επισημάνουμε ακόμη ότι το συγκεκριμένο εργαλείο, για την δημιουργία ενός *add-on*, υποστηρίζεται από την έκδοση 38 του *Firefox* και μετά.

Για την δημιουργία του *add-on*, από τη γραμμή εντολών πληκτρολογούμε την εντολή :

```
1 jpm init
```

και με αυτό το τρόπο γίνεται η αρχικοποίηση του *add-on* δημιουργώντας το σχετικό (Σχήμα 3.1) για το μηχανισμό.



ΣΧΗΜΑ 3.1: Αρχικοποίηση *Add-on*.

Ζητούνται κάποιες πληροφορίες οι οποίες σχετίζονται με το *add-on*, όπως είναι ο τίτλος, το όνομα, η έκδοση, η περιγραφή και άλλες πληροφορίες οι οποίες αποθηκεύονται στο αρχείο *package.json* παρέχοντας έτσι όλες τις πληροφορίες του μηχανισμού (Σχήμα 3.2).

```
C:\Users\user\Desktop\privacy-thesis>jpm init
title: (My Jetpack Addon) Privacy Thesis Upatras
name: (privacy-thesis)
version: (0.0.1)
description: (A basic add-on) Privacy
entry point: (index.js) privacy_addon.js
author: Stamellos Grigorios
engines (comma separated): (firefox,fennec)
license: (MIT)
JPM [info] About to write to C:\Users\user\Desktop\privacy-thesis\package.json:

{
  "title": "Privacy Thesis Upatras",
  "name": "privacy-thesis",
  "version": "0.0.1",
  "description": "Privacy",
  "main": "privacy_addon.js",
  "author": "Stamellos Grigorios",
  "engines": {
    "firefox": ">=38.0a1",
    "fennec": ">=38.0a1"
  },
  "license": "MIT",
  "keywords": [
    "jetpack"
  ]
}

Is this ok? (yes) _
```

ΣΧΗΜΑ 3.2: Δημιουργία του package.json.

Έπειτα μπορεί να αρχίσει η υλοποίηση του *add-on* γράφοντας τον εκτελέσιμο κώδικα στο αρχείο *privacy-addon.js*. Για να μεταγλωττίσουμε το εκτελέσιμο αρχείο χρησιμοποιούμε την εντολή:

```
1 jpm run
```

η οποία δημιουργεί ένα νέο στιγμιότυπο του *Firefox* το οποίο έχει εγκατεστημένο το *add-on*. Μόλις κλείσουμε αυτό το στιγμιότυπο τότε το *add-on* μας δεν είναι πλέον εγκατεστημένο στον *Firefox*. Τέλος, αφού έχουμε υλοποιήσει το μηχανισμό, χρησιμοποιούμε την εντολή:

```
1 jpm xpi
```

η οποία δημιουργεί ένα πακέτο του *add-on* ως *XPI* αρχείο, το οποίο είναι και η εγκαταστήσιμη μορφή αρχείου για τα *add-on* του *Mozilla*. Αυτές είναι η τρεις βασικές εντολές για τη δημιουργία ενός *add-on*, σε συνδυασμό με την εγκατάσταση της πλατφόρμας *Nodejs* που αναφέραμε νωρίτερα.

## 3.2 Έλεγχος HTTPS

Ένα από τα θεμελιώδη ζητήματα τα οποία θα εξετάσουμε είναι και η ύπαρξη του *HTTPS* (*Hypertext Transfer Protocol Secure*) με σκοπό να επιτευχθεί μία ασφαλής σύνδεση στο διαδίκτυο. Συγκεκριμένα το *HTTPS* είναι το πρωτόκολλο επικοινωνίας ενός χρήστη με έναν *server*, για την προβολή ιστοσελίδων. Η λειτουργία του είναι ίδια με εκείνη του *HTTP* με τη βασική διαφορά ότι η μεταφορά των δεδομένων γίνεται με κάποια ασφάλεια. Ένας σύνδεσμος (*URL*) που αρχίζει με το πρόθεμα *HTTPS* μας δείχνει ότι θα χρησιμοποιηθεί κανονικά το πρωτόκολλο επικοινωνίας *HTTP* αλλά η σύνδεση θα γίνει σε διαφορετική πόρτα, συγκεκριμένα την 443 αντί της 80 που χρησιμοποιεί το *http*, και τα δεδομένα θα ανταλλάσσονται κρυπτογραφημένα. Μία τέτοιου είδους κρυπτογράφηση είναι η *SSL* και επιτυγχάνεται με την εγκατάσταση *SSL* (*Secure Sockets Layer*) πιστοποιητικών που εγκαθιστά κάποιος στην ιστοσελίδα του. Με την παραπάνω διαδικασία τα δεδομένα μεταφέρονται από τον *server* με ασφάλεια και προβάλλεται η ιστοσελίδα στο πρόγραμμα περιήγησης μας.

Η διαδικασία που ακολουθήσαμε για να ελέγξουμε αν γίνεται με ασφάλεια η μεταφορά των δεδομένων είναι κάνοντας έλεγχο στο σύνδεσμο *URL* για την ύπαρξη του *HTTPS*. Κάθε φορά που μία νέα σελίδα φορτώνεται στο πρόγραμμα περιήγησης μας, το *add-on* παίρνει το *URL* και από αυτό μπορεί να εξάγει το πρωτόκολλο επικοινωνίας που χρησιμοποιεί η ιστοσελίδα. Συγκεκριμένα :

```
1 tabs.on('load', function(tab) {  
2     var URL = tab.url;  
3     var url = require("sdk/url").URL(URL);  
4     var protocol = url.protocol;  
5 }
```

Με αυτό το τρόπο έχουμε στην μεταβλητή *protocol* το πρωτόκολλο επικοινωνίας που χρησιμοποιείται (*HTTP* ή *HTTPS*). Δίνοντας σαν όρισμα σε μία συνάρτηση τη μεταβλητή αυτή με τον έλεγχο που φαίνεται στο παρακάτω τμήμα κώδικα, μπορούμε να αποφανθούμε αν χρησιμοποιείται το *HTTPS* πρωτόκολλο ή όχι.

```
1 function isSecureHttps(protocol){  
2     var x=0;  
3     if(protocol === "https:"){  
4         x=1;  
5     }  
6     return x;  
7 }
```

Όταν επιστρέφεται η τιμή 1 τότε διαπιστώνουμε ότι γίνεται χρήση του *HTTPS* πρωτοκόλλου, αλλιώς με τη τιμή 0 έχουμε τη χρήση του *HTTP*. Καθώς το *add-on* που

έχουμε υλοποιήσει σχετίζεται με την ασφάλεια των προσωπικών μας δεδομένων, είναι απαραίτητο να ελέγξουμε αν χρησιμοποιείται το *HTTPS* πρωτόκολλο ειδικότερα στις μέρες μας που χρειάζεται αυξημένη ασφάλεια, διότι διακινούνται ευαίσθητες προσωπικές πληροφορίες όπως αριθμοί πιστωτικών καρτών, κωδικοί πρόσβασης κλπ. Η κρυπτογράφηση που χρησιμοποιείται διασφαλίζει ότι τα κρυπτογραφημένα δεδομένα δεν θα μπορούν να υποκλαπούν από άλλους κακόβουλους χρήστες ή από επιθέσεις τύπου *man-in-the-middle*.

### 3.3 Έλεγχος HSTS

Το *HTTP Strict Transport Security (HSTS)* είναι ένα απλό χαρακτηριστικό ασφαλείας το οποίο αναγκάζει τα προγράμματα περιήγησης να επικοινωνούν με τους servers μόνο μέσω *HTTPS* συνδέσεων. Η μεταφορά ιστοσελίδων και δεδομένων μέσω συνδέσεων *HTTP* εκθέτει τους χρήστες σε μεγάλο κίνδυνο ασφαλείας ιδιαίτερα όταν αποστέλλονται ευαίσθητα προσωπικά δεδομένα.

Επειδή οι *servers* συνήθως δεν συνδέονται άμεσα μεταξύ τους, πρέπει να περάσουν τα αιτήματα τους και τις απαντήσεις αυτών μέσω μιας σειράς από network routers, αυτά τα routers, βρίσκονται στο ενδιάμεσο μεταξύ των servers, έχουν πλήρη πρόσβαση σε αιτήσεις που αποστέλλονται μέσω συνδέσεων *HTTP*. Από τη στιγμή που τα δεδομένα μεταφέρονται σε μη κρυπτογραφημένη μορφή, οι routers μπορεί να λειτουργήσουν ως *man-in-the-middle* με αποτέλεσμα να μπορούν να διαβάσουν ακόμη και να διαχειριστούν τα δεδομένα αυτά κατά τη μεταφορά. Το *HSTS* ουσιαστικά αναγκάζει όλες τις απαντήσεις να περάσουν μέσω του *HTTPS* πρωτοκόλλου αντί του *HTTP*. Με αυτό το τρόπο εξασφαλίζεται ότι το κανάλι κρυπτογραφείται πριν αποσταλεί οποιοδήποτε δεδομένο. Η ενεργοποίηση του *HSTS* στον *server* περιλαμβάνει την προσθήκη της παρακάτω *HSTS* επικεφαλίδας.

```
1 Strict-Transport-Security: max-age=expireTime [; includeSubdomains]
```

όπου το *max-age* μας δείχνει τον ελάχιστο χρόνο σε δευτερόλεπτα, που το πρόγραμμα περιήγησης θα πρέπει να συνδέεται με τον *server* μέσω *HTTPS* σύνδεσης. Η παράμετρος *includeSubdomains* μας υποδεικνύει ότι η ίδια διαδικασία θα πρέπει να ακολουθείται και για τα υπάρχοντα ή για τη προσθήκη μελλοντικών *subdomains*. Για παράδειγμα,

```
1 Strict-Transport-Security: max-age=31536000; includeSubDomains
```

μας δείχνει ότι η επικοινωνία θα γίνεται μέσω του *HTTPS* πρωτοκόλλου για τον επόμενο ένα χρόνο, συμπεριλαμβάνοντας και τα *subdomains*.

Το *add-on* μπορεί να ανιχνεύσει αν χρησιμοποιείται ο μηχανισμός *HSTS*. Αρχικά, κάθε φορά που φορτώνει μία σελίδα στο πρόγραμμα περιήγησης, απομονώνουμε από το *URL* το *hostname* όπως φαίνεται και παρακάτω,

```
1 tabs.on('load', function(tab) {  
2     var URL = tab.url;  
3     var url = require("sdk/url").URL(URL);  
4     var hostname = url.hostname;  
5 }
```

στη συνέχεια για να μπορέσουμε να αποκτήσουμε πρόσβαση στη βιβλιοθήκη που μας δίνει την επιθυμητή συνάρτηση χρησιμοποιούμε την εντολή:

```
1 var gSSService = Cc["@mozilla.org/ssservice;1"]  
2     .getService(Ci.nsISiteSecurityService);
```

Τέλος, καλούμε τη συνάρτηση *isSecureURI* η οποία έχει τρία ορίσματα. Το πρώτο δείχνει για το είδος του μηχανισμού που θέλουμε να αποκτήσουμε πληροφορία, στη συγκεκριμένη περίπτωση είναι για το μηχανισμό *HSTS*, το δεύτερο όρισμα είναι το *URI* το οποίο επισκεπτόμαστε και το τελευταίο όρισμα είναι κάποια *FLAGS* στα οποία δηλώνεται ότι αν έχουμε ενεργοποιήσει την ιδιωτική περιήγηση στον *browser* μας, δε μπορεί να αποθηκευτεί κανένα δεδομένο. Ο κώδικας που χρησιμοποιήσαμε είναι ο παρακάτω:

```
1 function isSecureUri(hostname){  
2     var uri = Services.io.newURI("https://" + hostname, null, null);  
3     var resultURI = gSSService.isSecureURI(STS_TYPE, uri, FLAGS);  
4     var x=0;  
5     if(resultURI=== true){  
6         x=1;  
7     }  
8     return x;  
9 }
```

η συνάρτηση αυτή μας επιστρέφει μία boolean τιμή, και συγκεκριμένα όταν επιστρέφεται η τιμή 1 τότε σηματοδοτείται τη χρήση του μηχανισμού *HSTS* από την ιστοσελίδα, για τιμή 0 δεν έχουμε τη χρήση του μηχανισμού. Είναι ένας ιδιαίτερα σημαντικός μηχανισμός, και ουσιαστικά μία επέκταση του *HTTPS* καθώς αυτόματα μετατρέπει μη ασφαλείς συνδέσεις οι οποίες αναφέρονται σε μία διαδικτυακή εφαρμογή, σε ασφαλείς συνδέσεις. Για παράδειγμα αν έχουμε το *URL*:

`http://example.com/some/page/`

τότε αυτό μετατρέπεται αυτόματα σε:

`https://example.com/some/page/`



Διαπιστώνουμε λοιπόν ότι η πολιτική του *HSTS* μπορεί να προστατέψει το χρήστη από κάποιες διαδικτυακές υποκλοπές και επιθέσεις στο διαδίκτυο. Οι *man-in-the-middle* επιθέσεις έχουν σημαντικά μειωμένη πιθανότητα να υποκλέψουν αιτήματα και απαντήσεις μεταξύ ενός χρήστη και του *server*, όταν το πρόγραμμα περιήγησης του χρήστη χρησιμοποιεί την τεχνολογία *HSTS*.

### 3.4 Έλεγχος για τους αλγορίθμους κρυπτογράφησης

Έχουμε ασχοληθεί με δύο βασικές λειτουργίες οι οποίες μας εξασφαλίζουν την προστασία των προσωπικών μας δεδομένων όπως η χρήση του πρωτοκόλλου επικοινωνίας *HTTP Secure* και του *HSTS*, συνεχίζοντας εξετάζουμε το είδος των αλγορίθμων κρυπτογράφησης που χρησιμοποιούνται και διάφορα τεχνικά χαρακτηριστικά τους. Όπως έχουμε αναφέρει, είναι πολλοί οι αλγόριθμοι κρυπτογράφησης που μπορούν να χρησιμοποιηθούν για την κρυπτογράφηση των δεδομένων. Μερικοί από αυτούς παρέχουν υψηλά επίπεδα ασφαλείας, αλλά απαιτούν ένα μεγάλο αριθμό υπολογισμών για την κρυπτογράφηση και αποκρυπτογράφηση των δεδομένων. Άλλοι αλγόριθμοι είναι λιγότερο ασφαλείς, αλλά έχουν γρηγορότερη απόκριση κατά τη διαδικασία της κρυπτογράφησης και αποκρυπτογράφησης. Το μήκος του κλειδιού που χρησιμοποιείται για την κρυπτογράφηση επηρεάζει το επίπεδο ασφαλείας και συγκεκριμένα όσο μεγαλύτερο είναι το κλειδί ασφαλείας τόσο πιο ασφαλή είναι τα δεδομένα. Όπως αναφέραμε και νωρίτερα, ουσιαστικά όταν χρησιμοποιείται το πρωτόκολλο επικοινωνίας *HTTPS*, υπάρχει κρυπτογράφηση *TLS/SSL*. Ανάλογα με την ανάγκη του χρήστη αλλάζει και το επίπεδο ασφαλείας και για να είναι δυνατή η επικοινωνία με άλλους που μπορεί να έχουν διαφορετικές ανάγκες, το *SSL* ορίζει ένα *cipher suite*, δηλαδή ένα σύνολο από αλγορίθμους κρυπτογράφησης. Όταν εγκαθίσταται μία *SSL* σύνδεση κατά τη διάρκεια της *SSL handshake*, ο χρήστης ανταλλάσσει με τον *server* πληροφορίες σχετικά με το ποια *cipher suite* έχουν και οι δύο. Έπειτα επικοινωνούν χρησιμοποιώντας την από κοινού *cipher suite* η οποία παρέχει τη μέγιστη δυνατή ασφάλεια. Με τον μηχανισμό μας μπορούμε να δούμε αρχικά αν χρησιμοποιείται κάποιο *cipher suite* και αν ναι, μπορούμε να εξετάσουμε κάποια επιπλέον τεχνικά χαρακτηριστικά τα οποία θα περιγράψουμε παρακάτω.

Το *add-on*, κάθε φορά που φορτώνεται μία νέα σελίδα στο πρόγραμμα περιήγησης μας, παίρνει το *url* και κάνει μία σύγχρονη *HTTP* αίτηση μέσω της οποίας μπορούμε να πάρουμε της πληροφορίες που επιθυμούμε. Συγκεκριμένα παίρνουμε το *url* όπως φαίνεται παρακάτω:

```
1 tabs.on('load', function(tab) {  
2   var URL = tab.url;  
3   var url = require("sdk/url").URL(URL);  
4 }
```

κάθε φορά που κάνουμε αυτή την *HTTP* αίτηση, από το κανάλι επικοινωνίας που δημιουργείται μπορούμε να λάβουμε κάποιες πληροφορίες οι οποίες σχετίζονται με την ασφάλεια. Κάθε φορά που έχουμε μία νέα σύνδεση δημιουργείτε ένα στιγμιότυπο του καναλιού οπου με τη χρήση της κατάλληλης βιβλιοθήκης μπορούμε να πάρουμε τις πληροφορίες που επιθυμούμε. Συγκεκριμένα:

```
1 let channel = oReq.channel;
2 var secInfo = channel.securityInfo;
3
4 if (secInfo instanceof Ci.nsISSLSecurityProvider) {
5     var cipher = secInfo.QueryInterface(Ci.nsISSLSecurityProvider)
6         .SSLStatus.QueryInterface(Ci.nsISSLSecurityStatus);
7
8     var protocolVersion = cipher.protocolVersion;
9
10    dump("\tCipher Name = " + cipher.cipherName + "\n");
11    dump("\tKey Length = " + cipher.keyLength + " bits \n");
12    dump("\tSecret Key Length = " + cipher.secretKeyLength + " bits \n");
13    dump("\tUntrusted = " + cipher.isUntrusted + "\n");
14    dump("\tProtocol Version = " + protocolVersion + "\n");
15    dump("\tis Not Valid At This Time= " + cipher.isNotValidAtThisTime + "\n");
16    ;
17    switch (protocolVersion) {
18        case Ci.nsISSLSecurityStatus.SSL_VERSION_3:
19            dump("\tProtocol version : SSLv3\n");
20            break;
21        case Ci.nsISSLSecurityStatus.TLS_VERSION_1:
22            dump("\tProtocol version : TLSv1\n");
23            break;
24        case Ci.nsISSLSecurityStatus.TLS_VERSION_1_1:
25            dump("\tProtocol version : TLSv1.1\n");
26            break;
27        case Ci.nsISSLSecurityStatus.TLS_VERSION_1_2:
28            dump("\tProtocol version : TLSv1.2\n");
29            break;
30    }
31 }
```

Μόλις λοιπόν φορτωθεί μία νέα σελίδα, εξετάζουμε πρώτον αν χρησιμοποιείται κάποιο *cipher-suite* επιστρέφοντας τη τιμή 1 αλλιώς παίρνουμε την τιμή 0 και δεύτερον αν χρησιμοποιείται τότε μπορούμε να συλλέξουμε κάποιες πληροφορίες οι οποίες σχετίζονται με τους αλγορίθμους κρυπτογράφησης (Σχήμα 3.3).

Αναλυτικότερα, πέρα από το *cipher name*, το οποίο θα ερμηνεύσουμε παρακάτω, μας επιστρέφεται το μέγεθος του δημοσίου κλειδιού σε *bit*, το μήκος του μυστικού

```
Cipher Name = TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256  
Key Length = 128 bits  
Secret Key Length = 128 bits  
Untrusted = false  
Protocol Version = 3  
is Not Valid At This Time= false
```

ΣΧΗΜΑ 3.3: Παράδειγμα cipher-suite.

κλειδιού επίσης σε bit καθώς και το είδος του πρωτοκόλλου που χρησιμοποιείται. Για παράδειγμα το *cipher name* της Google είναι:

TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256

όπου το *TLS* μας δείχνει το πρωτόκολλο που χρησιμοποιείται, το *ECDHE* σηματοδοτεί τον αλγόριθμο ανταλλαγής κλειδιού, το *RSA* είναι ο αλγόριθμος ελέγχου ταυτότητας, *AES-128-GCM* είναι ο κύριος αλγόριθμος κρυπτογράφησης και συγκεκριμένα σημαίνει ότι αυτό το *cipher-suite* χρησιμοποιεί 128 bit για το τρόπο λειτουργίας αλγορίθμου συμμετρικού κλειδιού και τέλος το *SHA256* δείχνει τον αλγόριθμο *MAC* (*Message Authentication Code*).

Όπως αναφέραμε και νωρίτερα, όταν εγκαθιδρύεται μία *TLS* σύνδεση γίνεται ένα *TLS handshake* και ο χρήστης στέλνει μία λίστα από *cipher suites* τα οποία υποστηρίζει κατά σειρά προτεραιότητας, έπειτα ο *server* απαντάει με το *cipher suite* που έχει επιλέξει από τη λίστα του χρήστη. Για να μπορέσει να γίνει έλεγχος σχετικά με το ποιους αλγορίθμους κρυπτογράφησης υποστηρίζει ο *server*, μπορεί να χρησιμοποιηθεί ένας *SSL/TLS scanner*. Διαπιστώνουμε λοιπόν ότι με αυτή τη διαδικασία μπορούμε να ελέγξουμε αν έχει εγκατασταθεί μία ασφαλής σύνδεση στο διαδίκτυο αποφεύγοντας αρκετούς κινδύνους όπως μία *man-in-the-middle* επίθεση, το οποίο αποτελεί το βασικό έλεγχο για την ασφαλή περιήγηση μας στο διαδίκτυο, πόσο μάλλον όταν σε αυτό εκθέτουμε σημαντικές προσωπικές πληροφορίες.

### 3.5 Έλεγχος για την Αρχή Πιστοποίησης

Για να μπορέσει να ολοκληρωθεί με ασφάλεια μία επικοινωνία στο διαδίκτυο εισάγονται οι Ψηφιακές υπογραφές, μέσω ενός κέντρου πιστοποίησης δημοσίων κλειδιών που ανήκουν σε άτομα, εταιρείες, και άλλους οργανισμούς. Ένας οργανισμός που πιστοποιεί δημόσια κλειδιά ονομάζεται Αρχή Πιστοποίησης ή *Certification Authority (CA)*. Η αυθεντικότητα πολλών εγγράφων καθορίζεται από την παρουσία ή την απουσία μιας εξουσιοδοτημένης χειρόγραφης υπογραφής. Θα πρέπει λοιπόν να εφαρμοστεί κάτι ανάλογο και στο διαδίκτυο έτσι ώστε πρώτον, ο παραλήπτης να μπορεί να επαληθεύσει την υποτιθέμενη ταυτότητα του αποστολέα, δεύτερον ο αποστολέας να μην μπορεί αργότερα να απαρνηθεί τα περιεχόμενα του μηνύματος

και τέλος ο παραλήπτης να μην μπορεί να ανακατασκευάσει το μήνυμα. Μία τέτοια διαδικασία είναι ιδιαίτερα χρήσιμη για τα οικονομικά συστήματα.

Για παράδειγμα, όταν ένας χρήστης κάνει μία μεγάλη παραγγελία από προϊόντα μέσω διαδικτύου, τότε αυτός που παράγει τα προϊόντα θα πρέπει να είναι σίγουρος για το άτομο που του έδωσε την παραγγελία, θα πρέπει επίσης να μπορεί να προστατευτεί από πιθανή απάτη, δηλαδή μπορεί αυτός που έκανε την παραγγελία να ισχυριστεί ότι δεν την έκανε ποτέ ζημιώνοντας το πρώτο και τέλος θα πρέπει να προστατευτεί και ο πελάτης στη περίπτωση όπου αυτός που παράγει τα προϊόντα προσπαθήσει να κατασκευάσει ένα υπογεγραμμένο μήνυμα αλλάζοντας το περιεχόμενο της αρχικής παραγγελίας.

Η κρυπτογραφία δημοσίου κλειδιού κάνει δυνατή την ασφαλή επικοινωνία ανάμεσα σε άτομα τα οποία δε μοιράζονται κοινό κλειδί. Κάνει επίσης δυνατή την υπογραφή μηνυμάτων χωρίς την παρουσία ενός έμπιστου τρίτου “ατόμου”. Ο μηχανισμός μας εξετάζει αρχικά αν υποστηρίζεται το CA, και αν ναι μας επιστρέφει πληροφορίες για τον οργανισμό, τον αλγόριθμο υπογραφής, τον εκδότη και κάποια άλλα χαρακτηριστικά που θα παρουσιάσουμε παρακάτω. Τέλος μπορούμε να δούμε την χρονική διάρκεια που ισχύει το πιστοποιητικό. Για να μπορέσουμε να αποκτήσουμε αυτές τις πληροφορίες ακολουθούμε παρόμοια διαδικασία με εκείνη του *cipher-suite*. Κάθε φορά που φορτώνεται μία νέα σελίδα (*URL*) τότε κάνουμε μία σύγχρονη *HTTP* αίτηση σε αυτή. Με αυτό το τρόπο μπορούμε να δημιουργήσουμε ένα στιγμιότυπο του καναλιού επικοινωνίας που με την χρήση της βιβλιοθήκης *SSL Status* μπορούμε να πάρουμε τις απαραίτητες πληροφορίες, όπως περιγράψαμε και για το *Cipher suite*. Συγκεκριμένα:

```
1 if (secInfo instanceof Ci.nsISSLStatusProvider) {  
2     var cert = secInfo.QueryInterface(Ci.nsISSLStatusProvider)  
3         .SSLStatus.QueryInterface(Ci.nsISSLStatus).serverCert;  
4  
5 }
```

LISTING 3.1: Αλληλεπίδραση με τη βιβλιοθήκη nsIX509Cert

με τη μόνη διαφορά, ότι μέσω αυτής της βιβλιοθήκης θα αποκτήσουμε πρόσβαση στην βιβλιοθήκη *nsIX509Cert* η οποία θα μας δώσει τις τιμές που επιθυμούμε. Ο κώδικας της βιβλιοθήκης είναι:

```
1 #include "nsISupports.idl"  
2  
3 interface nsIX509Cert;  
4  
5 [scriptable, uuid(fa9ba95b-ca3b-498a-b889-7c79cf28fee8)]  
6 interface nsISSLStatus : nsISupports {  
7     readonly attribute nsIX509Cert serverCert;
```

Αναλυτικά ο κώδικας είναι:

```

1 if (secInfo instanceof Ci.nsISSSLStatusProvider) {
2     var cert = secInfo.QueryInterface(Ci.nsISSSLStatusProvider)
3         .SSLStatus.QueryInterface(Ci.nsISSSLStatus).serverCert;
4
5     dump("\tCommon name (CN) = " + cert.commonName + "\n");
6     dump("\tIssuer = " + cert.issuerOrganization + "\n");
7     dump("\tOrganisation = " + cert.organization + "\n");
8     dump("\tSHA1 fingerprint = " + cert.sha1Fingerprint + "\n");
9     dump("\tSHA256 fingerprint = " + cert.sha256Fingerprint + "\n");
10    dump("\tToken Name = " + cert.tokenName + "\n");
11 }

```

LISTING 3.2: Βασικός κώδικας για την αναγνώριση του CA

όπου το *Common Name* είναι η ονομασία του θέματος, το *issuer Organization* είναι , το *SHA1 fingerprint* είναι ο ασφαλής αλγόριθμος κατακερματισμού και τέλος το *token Name* είναι ένα από τον άνθρωπο αναγνώσιμο όνομα που προσδιορίζει το υλικό ή το λογισμικό *token* στο οποίο είναι αποθηκευμένο το πιστοποιητικό. Για παράδειγμα το CA του Youtube παρουσιάζεται παρακάτω (Σχήμα 3.4) και μας δείχνει τις πρόσθετες πληροφορίες.

```

Common name (CN) = *.google.com
Issuer = Google Inc
Organisation = Google Inc
SHA1 fingerprint = 8F:51:12:06:A0:CC:4E:CD:E8:A3:8B:38:F8:87:59:E5:AF:95:CA:CD
SHA256 fingerprint = B9:10:21:03:FA:FE:01:4A:46:E3:E5:A8:8D:F3:06:A9:03:E3:58:C8
:21:87:61:C8:45:0C:90:4D:99:55:83:E9
Token Name = Software Security Device

```

ΣΧΗΜΑ 3.4: Πληροφορίες σχετικά με το CA του Youtube.

Για να μπορέσουμε να πάρουμε και το τελευταίο χαρακτηριστικό που αναφέραμε, δηλαδή τη χρονική διάρκεια που ισχύει ένα πιστοποιητικό, πρέπει να αποκτήσουμε πρόσβαση σε μία άλλη βιβλιοθήκη που ονομάζεται *nsIX509CertValidity*. Για να αποκτήσουμε πρόσβαση σε αυτή, εκτελέσαμε το παρακάτω τμήμα κώδικα:

```

1 var validity = cert.validity.QueryInterface(Ci.nsIX509CertValidity);

```

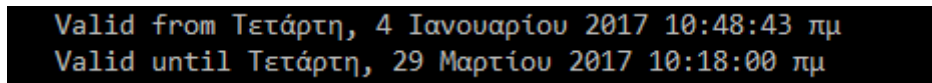
και με τις δύο παρακάτω εντολές:

```

1 dump("\tValid from " + validity.notBeforeGMT + "\n");
2 dump("\tValid until " + validity.notAfterGMT + "\n");

```

επιστρέφεται το χρονικό διάστημα κατά το οποίο βρίσκεται σε ισχύ η πιστοποίηση, συγκεκριμένα το *notBefore* είναι η πρώτη χρονική στιγμή κατά την οποία βρίσκεται σε ισχύ το πιστοποιητικό και το *notAfter* είναι η τελευταία χρονική στιγμή όπου είναι έγκυρο το πιστοποιητικό (Σχήμα 3.5).



```
Valid from Τετάρτη, 4 Ιανουαρίου 2017 10:48:43 πμ
Valid until Τετάρτη, 29 Μαρτίου 2017 10:18:00 πμ
```

ΣΧΗΜΑ 3.5: Χρονικό διάστημα που είναι έγκυρο το πιστοποιητικό.

Τα αξιόπιστα πιστοποιητικά μπορούν να χρησιμοποιηθούν για τη δημιουργία ασφαλών συνδέσεων σε έναν *server* μέσω του διαδικτύου. Το πιστοποιητικό είναι απαραίτητο προκειμένου να μπορέσουμε να παρακάμψουμε ένα κακόβουλο “άτομο” που τυχαίνει να είναι στη διαδρομή του *server* στόχου, το οποίο ενεργεί σαν να ήταν αυτός ο στόχος. Ένα τέτοιο σενάριο ονομάζεται συχνά και ως *man-in-the-middle* επίθεση. Ο χρήστης χρησιμοποιεί το *CA* πιστοποιητικό για να επικυρώσει την υπογραφή *CA* στο πιστοποιητικό του *server*. Αυτό είναι μια διαδικασία που ακολουθείται έτσι ώστε να γίνει έναρξη μίας ασφαλούς σύνδεσης στο διαδίκτυο.

### 3.6 Έλεγχος για το HTTP Public Key Pinning

Τα ασφαλή κανάλια επικοινωνίας αποτελούν κομβικό σημείο για τους χρήστες στο διαδίκτυο πόσο μάλλον για εκείνους που έχουν χρηματικές συναλλαγές και κλείνουν συμφωνίες μέσω αυτού. Οι χρήστες επιθυμούν μία *end-to-end* ασφάλεια κατά την αποστολή και λήψη δεδομένων, ιδιαίτερα όταν εκτίθενται ευαίσθητα προσωπικά δεδομένα. Σε αυτή την ενότητα θα παρουσιάσουμε και τον τελευταίο μηχανισμό ο οποίος χρησιμοποιείται έτσι ώστε να επιτευχθεί μία ασφαλής σύνδεση. Το *HTTP Public Key Pinning (HPKP)*, είναι ένας μηχανισμός ασφαλείας που επιτρέπει στις *HTTPS* ιστοσελίδες να αντισταθούν στη πλαστοπροσωπία από εισβολείς οι οποίοι χρησιμοποιούν ένα λανθασμένο ή πλαστό πιστοποιητικό. Για να διασφαλιστεί λοιπόν η αυθεντικότητα του δημοσίου κλειδιού ενός *server* που χρησιμοποιείται κατά τη δημιουργία *TLS* συνδέσεων, αυτό το κλειδί είναι ενσωματωμένο στο *X.509* πιστοποιητικό το οποίο συνήθως είναι υπογεγραμμένο από μία Αρχή Πιστοποίησης (*CA*). Τα προγράμματα περιήγησης εμπιστεύονται πολλά από αυτά τα *CA*, τα οποία μπορούν να δημιουργούν πιστοποιητικά για αυθαίρετα *domain*. Αν κάποιος επιτιθέμενος μπορέσει να υπονομεύσει ένα *CA*, τότε είναι πολύ πιθανό να υπάρχουν *man-in-the-middle* επιθέσεις σε διάφορες *TLS* συνδέσεις. Με τον μηχανισμό *HPKP* λοιπόν, μπορεί να παρακαμφθεί αυτή η απειλή για το πρωτόκολλο *HTTPS* λέγοντας στο χρήστη ποιο δημόσιο κλειδί ανήκει στον *web server*.

Ο μηχανισμός που έχουμε υλοποιήσει είναι σε θέση να ανιχνεύσει αν χρησιμοποιείται αυτός ο μηχανισμός ή όχι επιστρέφοντας μία *boolean* τιμή. Η διαδικασία που ακολουθούμε είναι η ίδια με εκείνη για την ανίχνευση του μηχανισμού *HSTS*. Συγκεκριμένα, όταν φορτώνεται μία νέα σελίδα τότε απομονώνουμε το *hostname* από το *URL* και το δίνουμε ως όρισμα στη παρακάτω συνάρτηση :



```

1 function keyPins(hostname){
2   var uri = Services.io.newURI("https://" + hostname, null, null);
3   var resultURI = gSSService.isSecureURI(STS_TYPE, uri, FLAGS);
4   var x=0;
5   if(resultURI=== true){
6     x=1;
7   }
8   return x;
9 }

```

με τη διαφορά να εντοπίζεται στο όρισμα *STS-TYPE* της συνάρτησης *isSecureURI*, όπου στη προκειμένη περίπτωση αντί για:

```

1 var STS_TYPE = Ci.nsISiteSecurityService.HEADER_HSTS;

```

έχουμε την εντολή:

```

1 var STS_TYPE = Ci.nsISiteSecurityService.HEADER_HPKP;

```

Όπως αναφέραμε και νωρίτερα το αποτέλεσμα που επιστρέφεται είναι μία *boolean* τιμή και συγκεκριμένα, όταν επιστρέφεται η τιμή 1 σηματοδοτεί τη χρήση του μηχανισμού HPKP ενώ το 0 όχι. Με την ανίχνευση και αυτού του μηχανισμού, έχουμε συμπεριλάβει τις περισσότερες δυνατές παραμέτρους με τις οποίες επιτυγχάνεται μία ασφαλής σύνδεση στο διαδίκτυο. Μία ασφαλής σύνδεση στο διαδίκτυο αποτελεί το βασικό σκαλοπάτι για την προστασία των προσωπικών μας δεδομένων, καθώς δεν υπάρχει νόημα να εξεταστεί κάποιος άλλος κίνδυνος από τη στιγμή που δεν γίνεται με ασφάλεια η μεταφορά των δεδομένων και μη γνωρίζοντας το άτομο με το οποίο έρχεσαι σε επικοινωνία.

### 3.7 Ανίχνευση HTTP cookies

Τα *HTTP cookie*, είναι μικρά “κομμάτια” δεδομένων που στέλνει ο *server* πίσω στο πρόγραμμα περιήγησης του χρήστη, τα οποία πιθανόν να αποθηκεύονται εκεί (μόνο τα *Session cookies* δεν αποθηκεύονται) και στέλνονται μαζί πίσω όταν κληθούν από τον ίδιο *server*. Με αυτόν το τρόπο συλλέγονται βασικές πληροφορίες οι οποίες σχετίζονται με τη δραστηριότητα του χρήστη, από το τι ώρα και τι αναζήτηση σε ένα διαδικτυακό τόπο μέχρι κάποια από τα χαρακτηριστικά του μηχανήματος που χρησιμοποιεί.

Το *add-on* μπορεί να ανιχνεύσει τα *cookies* τα οποία προέρχονται από ένα συγκεκριμένο *host*. Κάθε φορά λοιπόν που φορτώνεται μία νέα σελίδα παίρνουμε το *host* από το *URL* όπως φαίνεται παρακάτω:

```
1 tabs.on('load', function(tab) {
2   var URL = tab.url;
3   var url = require("sdk/url").URL(URL);
4   var host = url.host;
5 }
```

στη συνέχεια με την εντολή:

```
1 var cookieManager = Cc["@mozilla.org/cookieManager;1"]
2   .getService(Ci.nsICookieManager2);
```

αποκτούμε πρόσβαση στην επιθυμητή βιβλιοθήκη και δίνοντας ως όρισμα το host στη συνάρτηση `countCookiesFromHost` όπως φαίνεται παρακάτω,

```
1 function cookiesNum(host){
2   console.log(host);
3   var resultCookies= cookieManager.countCookiesFromHost(host);
4   return resultCookies;
5 }
```

επιστρέφεται ο αριθμός των *cookies* τα οποία υπάρχουν στο *base domain* του *host* που πήραμε από το *URL*. Για παράδειγμα το *host* “*tech.in.gr*”, έχει ως *base domain* το “*in.gr*”. Έτσι οποιοδήποτε *host* ή *domain cookie* για το “*in.gr*” και τα *subdomains* του θα μετρηθεί. Για να μπορέσουμε να πάρουμε περισσότερες πληροφορίες και όχι μόνο τον αριθμό των *cookies* που προέρχονται από ένα συγκεκριμένο *host*, χρησιμοποιούμε τη συνάρτηση `getCookiesFromHost`, η οποία βρίσκεται στην ίδια βιβλιοθήκη. Αυτή η συνάρτηση μας επιστρέφει έναν *enumerator*, δηλαδή ένα αντικείμενο το οποίο έχει όλες τις πληροφορίες για όλα τα *cookies*.

```
1 function getCookies(host){
2   var cookies = cookieManager.getCookiesFromHost(host);
3   var count = 0;
4
5   while (cookies.hasMoreElements()){
6     var cookie = cookies.getNext().QueryInterface(Ci.nsICookie2);
7
8     dump("\tCookie host: " + cookie.host + " Is Domain cookie: " +cookie.isDomain
9       + "; Cookie Name : " + cookie.name + " = Cookie value: " + cookie.value + "
10       ; Is Session Cookie : " + cookie.isSession + "; Expiry time : " + cookie.
11       expiry + "; It is an Http only cookie : " + cookie.isHttpOnly + "\n");
12   }
13   return count;
14 }
```

Ο *enumerator* που είναι ένα αντικείμενο με *cookies*, επιστρέφει το κάθε ένα ξεχωριστά χρησιμοποιώντας τη συνάρτηση `cookies.hasMoreElements`. Για κάθε ένα από αυτά που



βρίσκει επικοινωνεί με τη βιβλιοθήκη *nsICookie2* από την οποία απόκτη πληροφορίες, όπως το όνομα του *cookie*, την τιμή του, αν είναι *HTTPOnly*, αν είναι *sessionCookie*, *domainCookie* και τέλος το πότε λήγει αυτό το *cookie* (Σχήμα 3.6).

```
Cookie host: .youtube.com Is Domain cookie: true; Cookie Name :VISITOR_INFO_LIV
E = Cookie value:vb-pcGjHdMw; Is Session Cookie :false; Expiry time :1505344176; It is
an Http only cookie :true
Cookie host: .youtube.com Is Domain cookie: true; Cookie Name :YSC = Cookie valu
e:ZPeRHkuT0xM; Is Session Cookie :true; Expiry time :9223372036854776000; It is an Htt
p only cookie :true
Cookie host: .youtube.com Is Domain cookie: true; Cookie Name :CONSENT = Cookie
value:WP.25bf73; Is Session Cookie :false; Expiry time :2145916800; It is an Http only
cookie :false
Cookie host: .youtube.com Is Domain cookie: true; Cookie Name :PREF = Cookie val
ue:f1=500000000&f5=30; Is Session Cookie :false; Expiry time :1547379957; It is an Http
only cookie :false
```

ΣΧΗΜΑ 3.6: Cookies host για το Youtube.

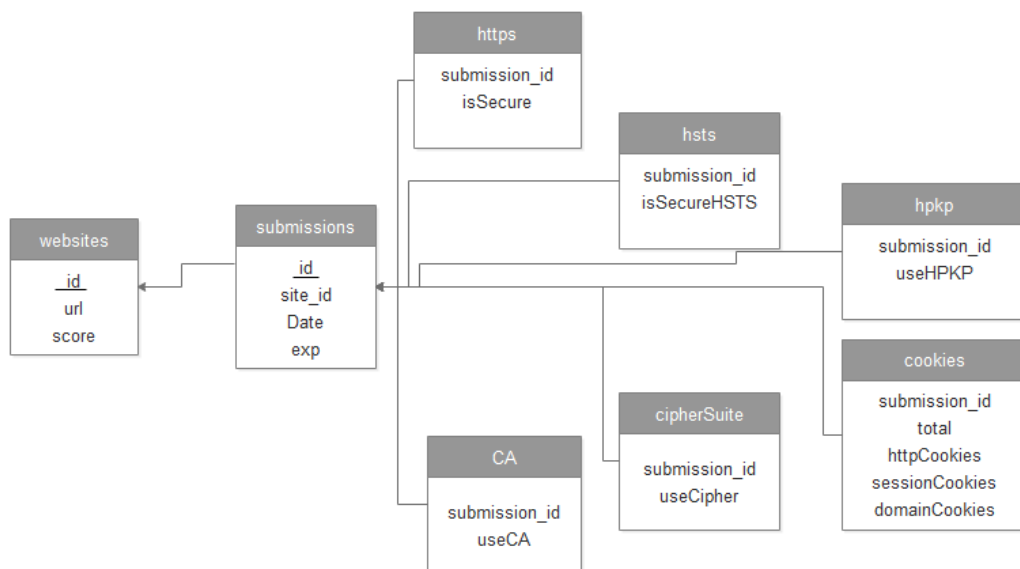
### 3.8 Βάση Δεδομένων

Έχοντας υλοποιήσει τον μηχανισμό, το επόμενο βήμα είναι η αποθήκευση των δεδομένων στη βάση. Αρχικά έγινε η εγκατάσταση του *Wamp Server*, το οποίο είναι ένα περιβάλλον που δίνει τη δυνατότητα δημιουργίας διαδικτυακών εφαρμογών με τη χρήση του *Apache2*, της *PHP* και βάσης δεδομένων η οποία είναι βασισμένη στην *MySQL*. Συγκεκριμένα το *PhpMyAdmin* επιτρέπει την εύκολη δημιουργία και διαχείριση της βάσης δεδομένων.

Ο σχεδιασμός της βάσης έγινε με τέτοιο τρόπο έτσι ώστε να αποθηκεύονται τιμές, οι οποίες θα χρησιμοποιηθούν για τους υπολογισμούς καθώς και για την εξαγωγή των συμπερασμάτων μας. Το σχήμα της βάσης φαίνεται παρακάτω (Σχήμα 3.7).

Η αποθήκευση των δεδομένων δεν ήταν δυνατόν να επιτευχθεί απευθείας από το add-on στη βάση, έτσι ήταν απαραίτητη η δημιουργία ενός API, δηλαδή ενός μηχανισμού ο οποίος θα δέχεται τα δεδομένα από το add-on θα συνδέεται με τη βάση και θα τα αποθηκεύει, καθώς θα επιστρέφει και τα δεδομένα από τη βάση σε json μορφή η οποία είναι εύκολα διαχειρίσιμη από το add-on. Συγκεκριμένα, κάθε φορά που φορτώνεται μία σελίδα στο πρόγραμμα περιήγησης, το add-on ανακτά όλες τις πληροφορίες που αναφέραμε και μέσω μίας HTTP αίτησης, καλεί αρχικά το API και του στέλνει τα δεδομένα όπως φαίνεται παρακάτω:

```
1 var Request = require("sdk/request").Request;
2 var quijote = Request({
3   url: "http://localhost/my_addon/post1.php",
4   content:{url:URL,isSecure:result_Https,isSecureHSTS:result_HSTS,useHPKP:
      result_keyPins,total:result_cookiesNum,httpCookies:result_getCookies,
      sessionCookies:result_getCookies3,thirdPartyCookies:result_getCookies2,
      useCA:result_CA,useCipher:result_CipherSSL},
```



ΣΧΗΜΑ 3.7: Σχήμα Βάσης δεδομένων.

```

5  onComplete: function (response) {
6      var data = response.json;
7      }
8  });
9  quijote.post();
10 });

```

όπου στο πεδίο url γίνεται η ανάθεση της διεύθυνσης του API που καλείται και στο πεδίο content γίνεται η ανάθεση των τιμών που ανιχνεύσαμε με το add-on μας στις μεταβλητές. Τέλος κάθε φορά που έχουμε νέα δεδομένα με την εντολή .post() γίνεται η αποστολή των δεδομένων στο API μέσω της μεθόδου POST. Συγκεκριμένα στο API, γίνεται έλεγχος αν έχει κληθεί η μέθοδος POST όπως φαίνεται παρακάτω:

```

1  if($_SERVER['REQUEST_METHOD'] == "POST")

```

αν η συνθήκη αυτή γίνει TRUE, τότε οι τιμές από το add-on μας αποθηκεύονται στις μεταβλητές του API όπως φαίνεται παρακάτω.

```

1  $url = isset($_POST['url'])?mysqli_real_escape_string($conn,$_POST['url']) :
    '';
2  $isSecureHttps = isset($_POST['isSecure'])?mysqli_real_escape_string($conn,
    $_POST['isSecure']) : '';
3  $isSecureHSTS = isset($_POST['isSecureHSTS']) ? mysqli_real_escape_string(
    $conn,$_POST['isSecureHSTS']) : '';

```

Τέλος, μόλις το API συνδεθεί με τη βάση δεδομένων και με την εκτέλεση της Mysql εντολής INSERT γίνεται η αποθήκευση των δεδομένων στη βάση.

Το add-on, αποτελείται λοιπόν από τέσσερα βασικά κομμάτια που είναι, πρώτον η εγκατάσταση του Nodejs για την υλοποίηση του μηχανισμού, δεύτερον η δημιουργία του add-on, τρίτον η δημιουργία του API για την αποθήκευση των δεδομένων στη βάση και τέταρτον, η δημιουργία της βάσης.



## Κεφάλαιο 4

# Αξιολόγηση Αποτελεσμάτων

Στη παρούσα διπλωματική εργασία πέρα από την μελέτη, ανάλυση και ανίχνευση ορισμένων από τους σημαντικότερους κινδύνους, πραγματοποιήθηκαν κάποια πειράματα για την εξαγωγή συμπερασμάτων σχετικά με την ασφάλεια των δεδομένων στο διαδίκτυο. Σε αυτό το κεφάλαιο θα παρουσιάσουμε το τρόπο συλλογής των δεδομένων, καθώς θα γίνει και επεξήγηση της επιλογής αυτών των τρόπων. Επίσης προβάλετε η γραφική απεικόνιση των αποτελεσμάτων καθώς και η ανάλυση αυτών. Τέλος θα παρουσιάσουμε μία γραφική απεικόνιση του μηχανισμού που υλοποιήσαμε.

### 4.1 Τρόπος συλλογής δεδομένων

Σε αυτό το υποκεφάλαιο θα παρουσιάσουμε τους δύο διαφορετικούς τρόπους που χρησιμοποιήσαμε για τη συλλογή δεδομένων. Η χρήση της βάσης δεδομένων δεν ήταν αναγκαία για την ενημέρωση του χρήστη σχετικά με την ασφάλεια του όταν επισκέπτεται ένα διαδικτυακό τόπο, αλλά ήταν αναγκαία για την αποθήκευση των δεδομένων. Τα δεδομένα αυτά χρησιμοποιήθηκαν για την ανάλυση και εξαγωγή των συμπερασμάτων. Κάθε φορά που φορτώνεται μία ιστοσελίδα στο πρόγραμμα περιήγησης τότε τα δεδομένα που εξάγονται στο χρήστη, αποθηκεύονται και στη βάση δεδομένων.

#### 4.1.1 Συλλογή δεδομένων από τη πλοήγηση του χρήστη στο διαδίκτυο

Ο πρώτος τρόπος συλλογής δεδομένων επιτεύχθηκε μέσω της τυχαίας πλοήγησης του χρήστη στο διαδίκτυο. Πέρα από την κανονική του δραστηριότητα, κάθε μέρα επισκέπτονταν και μία λίστα από συγκεκριμένες ιστοσελίδες. Σε αυτές δεν ακολουθούσε κάποιο συγκεκριμένο πρότυπο, αλλά η επιλογή της πλοήγησης του στο διαδικτυακό τόπο σχετίζονταν αποκλειστικά από τις προτιμήσεις του χρήστη. Η επιλογή του συγκεκριμένου τρόπου συλλογής δεδομένων στοχεύει στην εξαγωγή

συμπερασμάτων για τη συμπεριφορά των ιστοσελίδων κατά τη διάρκεια μίας πραγματικής πλοήγησης.

#### 4.1.2 Συλλογή δεδομένων με τη χρήση αυτοματοποιημένου μηχανισμού

Ο δεύτερος τρόπος συλλογής δεδομένων πραγματοποιήθηκε με τη χρήση ενός αυτοματοποιημένου μηχανισμού. Συγκεκριμένα υλοποιήθηκε ένας μηχανισμός σε γλώσσα προγραμματισμού Python, ο οποίος επισκέπτονταν καθημερινά και σε συγκεκριμένη ώρα μία λίστα από ιστοσελίδες. Οι βασικές εντολές υλοποίησης του μηχανισμού είναι οι παρακάτω:

```
1 import webbrowser
2 import os
3 import time
4 webbrowser.register('firefox', None, webbrowser.GenericBrowser('C:\\Program
   Files (x86)\\Mozilla Firefox\\firefox.exe'))
5 a=webbrowser.get('firefox')
6 my_websites = ['url','url','url']
7 for url in my_websites:
8     a.open(url)
```

Αναλυτικότερα, καλείται μία βιβλιοθήκη της Python (webbrowser), με την οποία μπορεί ο προγραμματιστής να αλληλεπιδρά με ένα πρόγραμμα περιήγησης. Για να μπορέσει να επιτευχθεί αλληλεπίδραση συγκεκριμένα με το Mozilla Firefox χρησιμοποιήθηκαν οι παρακάτω εντολές:

```
1 webbrowser.register('firefox', None, webbrowser.GenericBrowser('C:\\Program
   Files (x86)\\Mozilla Firefox\\firefox.exe'))
2 a=webbrowser.get('firefox')
```

Κάτι τέτοιο ήταν αναγκαίο καθώς ο μηχανισμός έχει υλοποιηθεί για το Mozilla Firefox. Έπειτα για κάθε url που υπάρχει στην λίστα, ανοίγει ένα νέο παράθυρο στο πρόγραμμα περιήγησης μέσω της εντολής a.open(url). Επειδή ο μηχανισμός είναι πλήρως αυτοματοποιημένος, ορίστηκε μετά από δέκα λεπτά από τη στιγμή που ανοίχθηκε και το τελευταίο url να κλείνει αυτόματα. Συγκεκριμένα με την εντολή:

```
1 time.sleep(360)
```

επιτυγχάνεται αυτή η χρονοκαθυστέρηση των δέκα λεπτών και τελικά το πρόγραμμα περιήγησης τερματίζει με την εντολή:

```
1 os.system("TASKKILL /F /IM firefox.exe")
```

Με την προηγούμενη διαδικασία, περιγράψαμε τον τρόπο με τον οποίο ανοίγουν οι σελίδες στον Mozilla Firefox. Για να μπορέσουμε να τρέχουμε αυτό το script έτσι

ώστε να ανοίγει τις σελίδες, δημιουργήσαμε ένα cron job που είναι μία προγραμματισμένη διεργασία την οποία ορίσαμε να τρέχει ανά τρεις ώρες. Συγκεκριμένα ο κώδικας που χρησιμοποιήθηκε είναι:

```
1 schtasks /create /sc minute /mo 180 /tn "hello" /tr C:\Users\user\Desktop\
  auto_open.py
```

Το όρισμα mo 180 σηματοδοτεί ότι κάθε 180 λεπτά θα τρέχει η διεργασία, το tn "name" σηματοδοτεί το όνομα της διεργασίας και τέλος ορίζεται το path που βρίσκεται το εκτελέσιμο αρχείο.

Ο λόγος που χρησιμοποιήθηκε ο συγκεκριμένος τρόπος επιλογής δεδομένων είναι για την ανάλυση της συμπεριφοράς των διαδικτυακών τόπων με την επίσκεψη συγκεκριμένων σελίδων του domain και σε συγκεκριμένη ώρα.

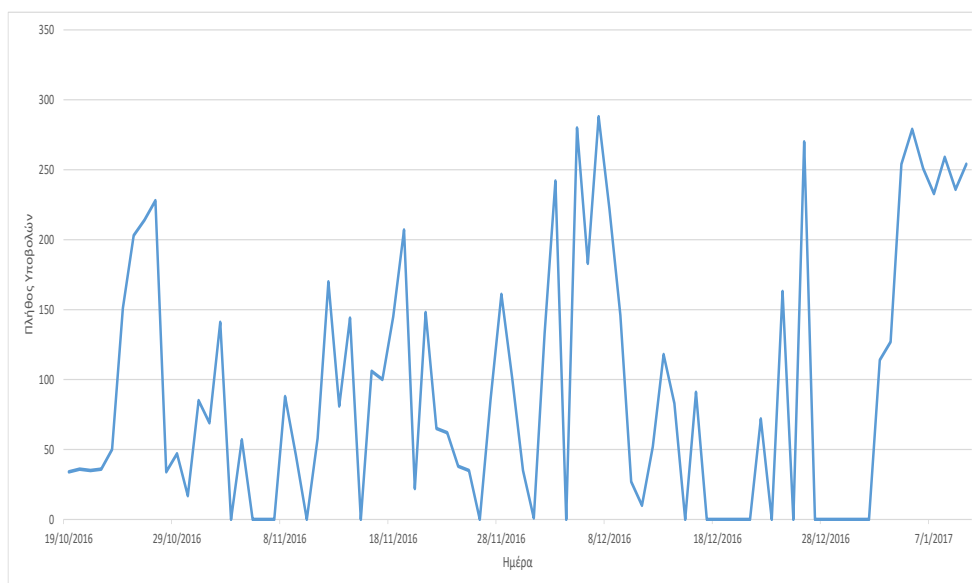
## 4.2 Γραφική ανάλυση αποτελεσμάτων

Σε αυτό το υποκεφάλαιο θα παρουσιάσουμε τις γραφικές παραστάσεις που εξήχθησαν από τα δεδομένα που συλλέξαμε, καθώς θα πραγματοποιηθεί και η ανάλυση αυτών των αποτελεσμάτων. Συγκεκριμένα θα παρουσιάσουμε τρεις διαφορετικές κατηγορίες γραφικών παραστάσεων. Στη πρώτη κατηγορία ανήκουν γραφικές παραστάσεις που σχετίζονται με το πλήθος υποβολών urls, στη δεύτερη κατηγορία γίνεται γραφική ανάλυση κάθε κινδύνου ξεχωριστά και τέλος τρίτη παρουσιάζονται γραφικές αναπαραστάσεις για κάποια συγκεκριμένα domain.

### 4.2.1 Γραφικές παραστάσεις με βάση το πλήθος υποβολών

Στο Σχήμα 4.1, παρουσιάζεται η γραφική παράσταση η οποία απεικονίζει τον αριθμό των υποβολών ανά μέρα. Στη παρούσα γραφική αναπαράσταση μπορούμε να δούμε τον ρυθμό συλλογής δεδομένων ανά ημέρα, παρατηρώντας ότι υπάρχουν ορισμένες απότομες πτωτικές τάσεις στη γραφική παράσταση οι οποίες δεν οφείλονται στη μη λειτουργία του μηχανισμού, αλλά στην μη σύνδεση του μηχανισμού με τη βάση δεδομένων λόγω προβλήματος του τοπικού διακομιστή.

Στην επόμενη γραφική παράσταση παρουσιάζεται το αθροιστικό πλήθος υποβολών ανά ημέρα (Σχήμα 4.2). Σκοπός της συγκεκριμένης γραφικής παράστασης ήταν η ανάδειξη της συνεχούς υποβολής url καθώς και το συνολικό πλήθος δεδομένων που συλλέχθηκε. Στη περίπτωση που η συλλογή των δεδομένων δεν πραγματοποιούνταν καθημερινά, τότε στη γραφική παράσταση θα απεικονίζονταν μία σταθερή γραμμή παράλληλη στον άξονα y.



ΣΧΗΜΑ 4.1: Πλήθος υποβολών ανά ημέρα.

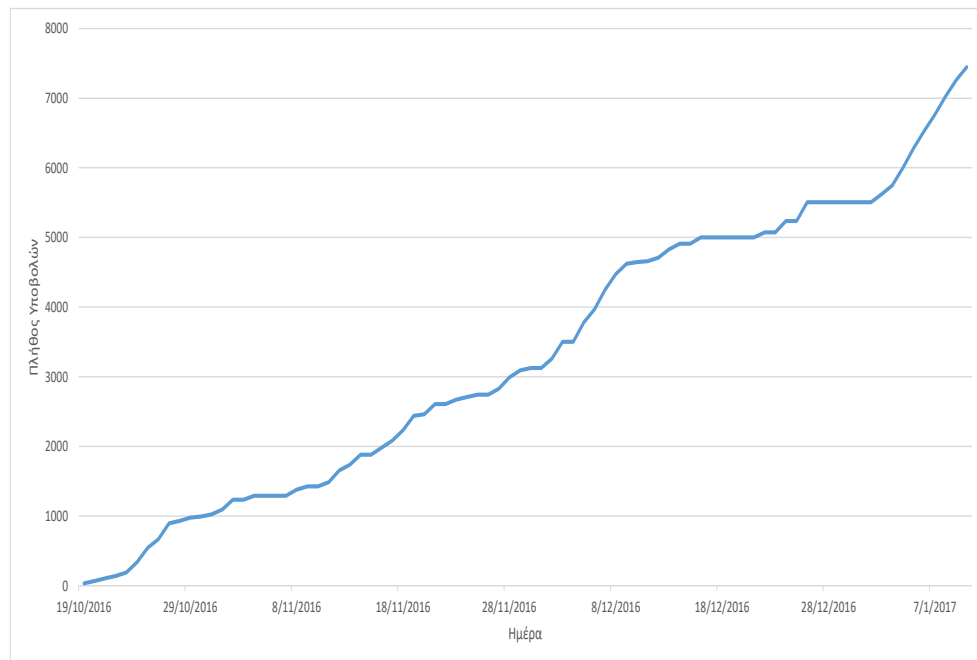
#### 4.2.2 Γραφικές παραστάσεις με βάση τον κίνδυνο

Στη παρούσα κατηγορία γραφικών παραστάσεων, θα πραγματοποιηθεί η γραφική απεικόνιση των αποτελεσμάτων για το κάθε κίνδυνο ξεχωριστά. Με αυτόν το τρόπο στοχεύουμε στην καλύτερη κατανόηση λειτουργίας του κάθε κινδύνου, για ένα πλήθος από url.

Στη γραφική παράσταση 4.3 αναδεικνύεται το ποσοστό των url τα οποία χρησιμοποιούν το πρωτόκολλο επικοινωνίας HTTPS και ποια όχι. Πρέπει να σημειωθεί ότι αυτή η γραφική παράσταση αποδεικνύει ότι η πλειοψηφία των ιστοσελίδων δε χρησιμοποιεί κρυπτογράφηση των δεδομένων. Αυτό είναι ιδιαίτερα ανησυχητικό κυρίως για ιστοσελίδες οι οποίες διαχειρίζονται ευαίσθητες προσωπικές πληροφορίες.

Στη συνέχεια παρουσιάζονται τα αποτελέσματα τα οποία σχετίζονται με την χρήση του μηχανισμού HSTS (Σχήμα 4.4). Παρατηρούμε ότι μόλις το 35 τοις εκατό των ιστοσελίδων που επισκεφθήκαμε χρησιμοποιεί το συγκεκριμένο μηχανισμό. Ο μηχανισμός αυτός είναι μία πρόσθετη λειτουργία του πρωτοκόλλου επικοινωνίας HTTPS. Είναι σημαντικό να παρατηρήσουμε ότι το ποσοστό των url που τον χρησιμοποιεί δε θα μπορούσε να ξεπεράσει το 41 τοις εκατό, καθώς από την προηγούμενη



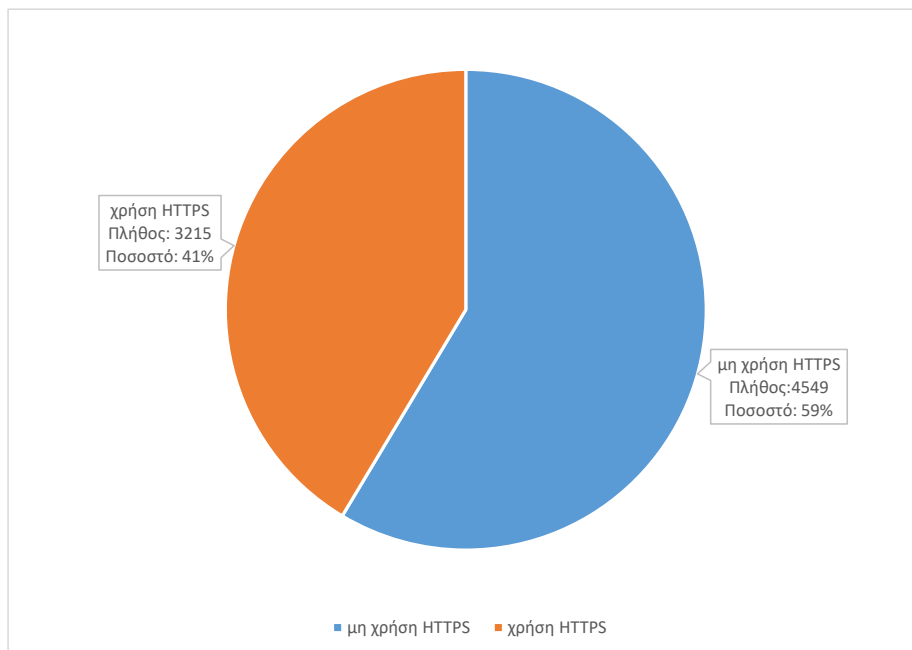


ΣΧΗΜΑ 4.2: Αθροιστικό πλήθος υποβολών ανά ημέρα.

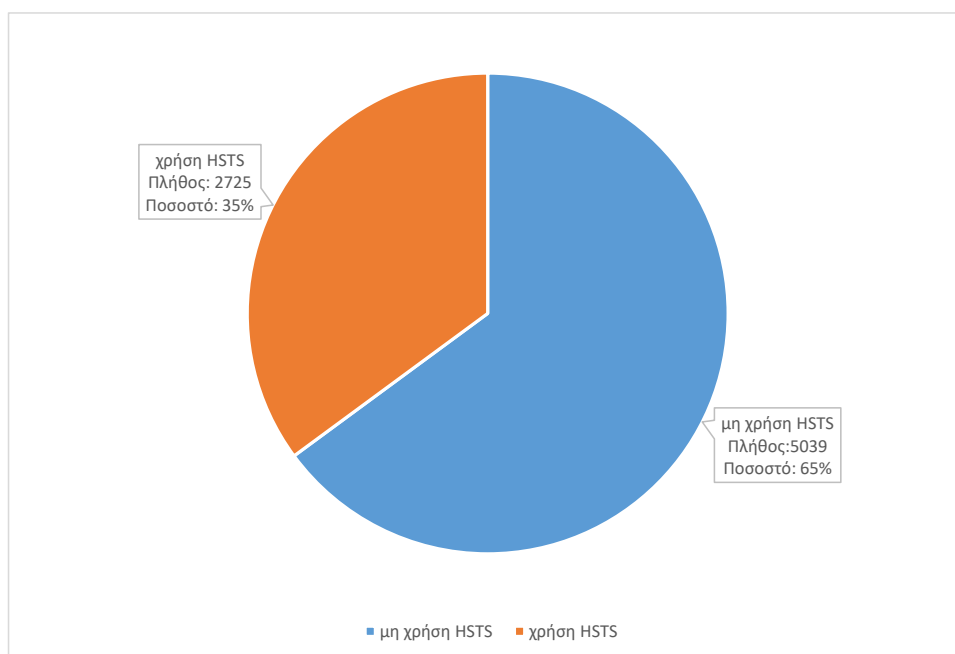
γραφική παράσταση διαπιστώσαμε ότι το 41 τοις χρησιμοποιεί το πρωτόκολλο επικοινωνίας HTTPS το οποίο είναι απαραίτητο για τη χρήση του μηχανισμού HSTS.

Στη γραφική παράσταση του Σχήματος 4.5 μελετάμε το ποσοστό χρήσης του Cipher suite. Διαπιστώσαμε, ότι μόλις το 35 τοις εκατό των url που επισκεφθήκαμε χρησιμοποιούν cipher suite, το οποίο είναι απαραίτητο για την κρυπτογράφηση των δεδομένων. Και πάλι αυτό το ποσοστό αναμένονταν να είναι μικρότερο του 40 τοις εκατό, εξαιτίας του γεγονότος ότι ένας διαδικτυακός τόπος που δε χρησιμοποιεί το πρωτόκολλο HTTPS δε μπορεί να χρησιμοποιεί Cipher suite.

Συνεχίζοντας, εξετάζουμε το ποσοστό χρήσης του Certification Authority (Σχήμα 4.6). Παρατηρούμε ότι το ποσοστό χρήσης του Certification Authority είναι επίσης 35 τοις εκατό. Πρέπει να σημειωθεί ότι το ποσοστό αναμένονταν να είναι ακριβώς το ίδιο με εκείνο της χρήσης Cipher suite, καθώς είναι απαραίτητη η ύπαρξη κρυπτογράφησης για τη δημιουργία μίας ψηφιακής υπογραφής. Επιπλέον, η απόκτηση πρόσβασης στην βιβλιοθήκη για το CA πραγματοποιείται αφότου έχει αποκτηθεί πρόσβαση για το Cipher suite. Εκτός από το ποσοστό που είναι το ίδιο, αξίζει να παρατηρηθεί και ποσοτικά το πλήθος των url που χρησιμοποιούν αυτούς τους μηχανισμούς. Συγκεκριμένα, το πλήθος των url που χρησιμοποιεί το Cipher suite είναι

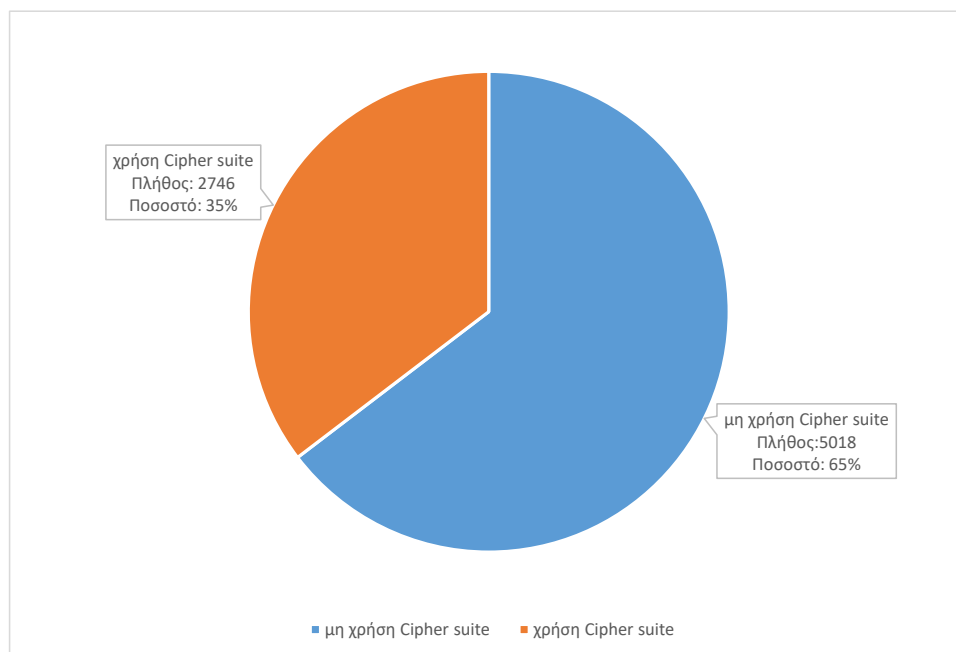


ΣΧΗΜΑ 4.3: Χρήση HTTPS πρωτοκόλλου.

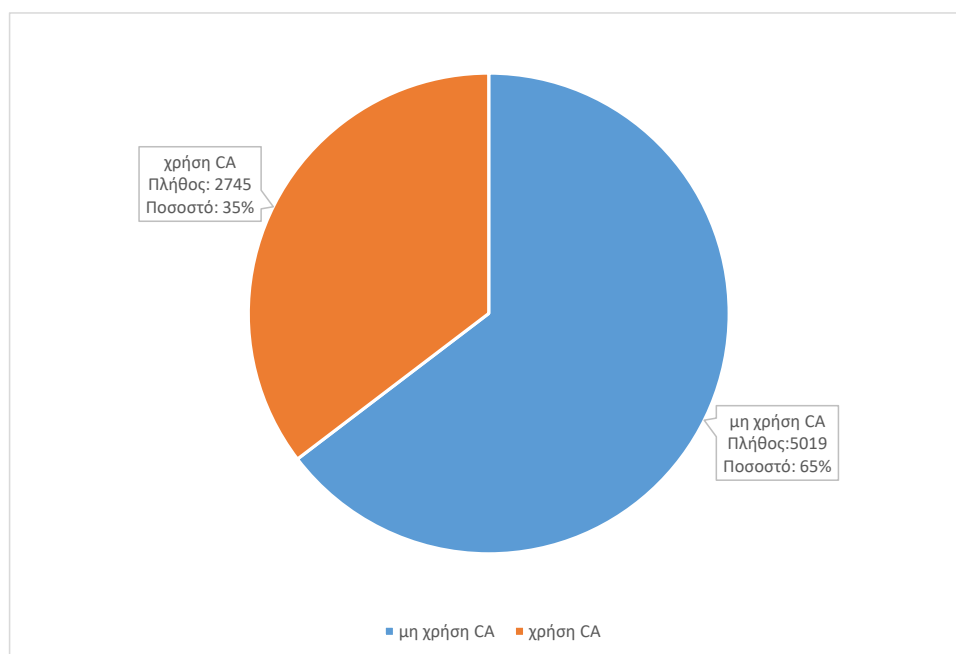


ΣΧΗΜΑ 4.4: Χρήση του μηχανισμού HSTS.

2746 ενώ το πλήθος που χρησιμοποιεί CA είναι 2745. Διαπιστώνουμε ότι η διαφορά είναι μόλις μία μονάδα, η οποία μπορεί να οφείλεται στη λήξη της προθεσμίας εγκυρότητας του CA πιστοποιητικού, καθώς υπάρχει συγκεκριμένο χρονικό διάστημα το οποίο ισχύει (πιθανώς να είναι το myceid καθώς είχε λήξει το πιστοποιητικό)

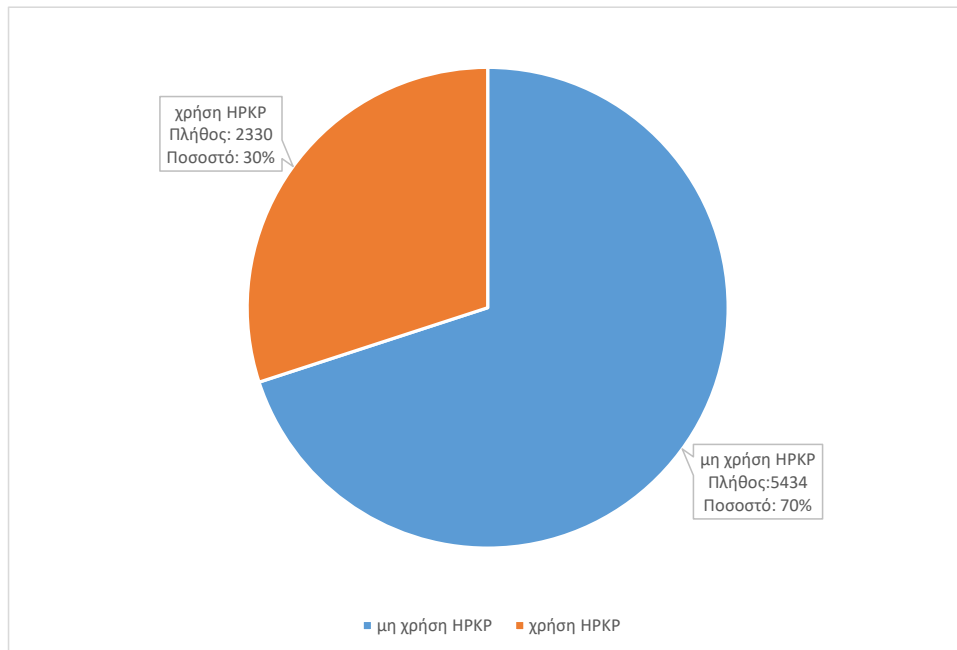


ΣΧΗΜΑ 4.5: Χρήση Cipher suite.



ΣΧΗΜΑ 4.6: Χρήση του Certification Authority.

Τελειώνοντας με την ανάλυση των κινδύνων που αφορούν τη δημιουργία μίας ασφαλούς επικοινωνίας στο διαδίκτυο, μελετάμε το ποσοστό χρήσης του μηχανισμού HPKP (Σχήμα 4.7). Διαπιστώνουμε ότι το ποσοστό χρήσης του συγκεκριμένου

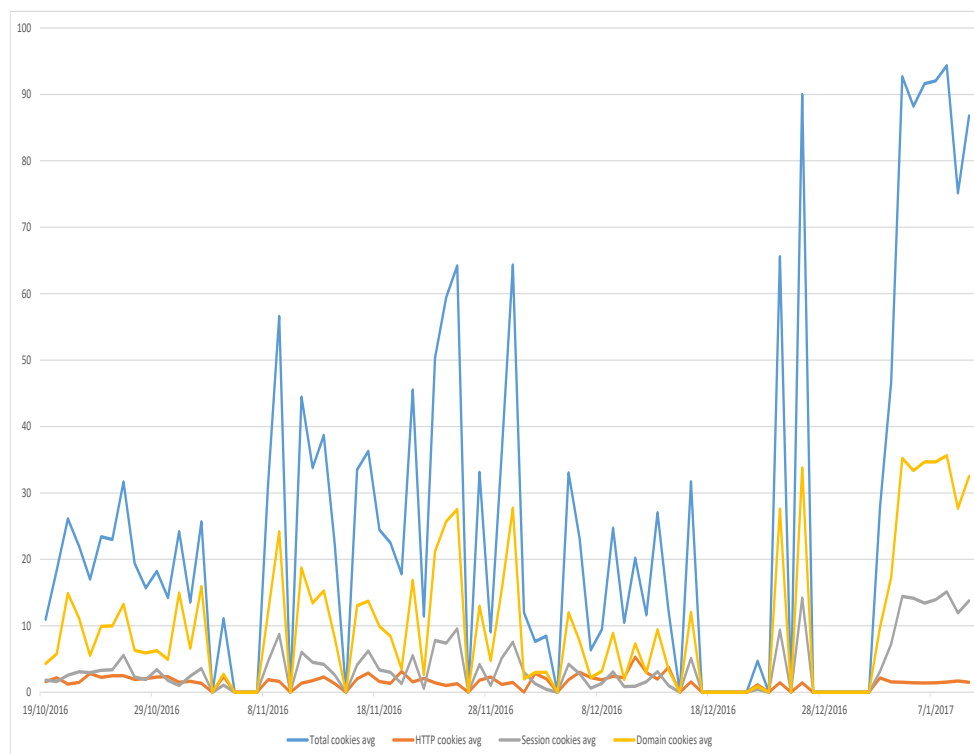


ΣΧΗΜΑ 4.7: Χρήση του μηχανισμού HPKP.

μηχανισμού είναι ακόμη μικρότερο. Όσο προχωράμε την ανάλυση για τη δημιουργία μίας ασφαλούς σύνδεσης, παρατηρούμε ότι όλο και λιγότερα url χρησιμοποιούν όλους τους μηχανισμούς που είναι απαραίτητοι για τη δημιουργία μίας ασφαλούς σύνδεσης. Και πάλι αναμένονταν το ποσοστό χρήσης του μηχανισμού να ήταν μικρότερο του 35 τοις εκατό, καθώς προϋπόθεση για την ύπαρξη του μηχανισμού είναι και η ύπαρξη κρυπτογράφησης.

Στη συνέχεια, παραθέτουμε την γραφική παράσταση (Σχήμα 4.8) η οποία απεικονίζει τον μέσο όρο των cookies ανά ημέρα για όλα τα url που επισκεφθήκαμε. Από αυτή τη γραφική παράσταση, γίνεται αντιληπτή η διαφορά στις τιμές μεταξύ των διαφορετικών cookies. Συγκεκριμένα παρατηρείται ότι ο αριθμός των HTTP cookies είναι καθ' όλη τη εξέλιξη των πειραμάτων περισσότερα και από τα Session cookies και από τα Domain cookies. Πρέπει να επισημάνουμε ότι ο αριθμός των total cookies αναπαριστά των αριθμό των cookies για ένα domain αλλά και για τα subdomains του, γι' αυτό και παρατηρείται τόσο μεγάλο πλήθος σε σχέση με τα υπόλοιπα είδη.

Στο Σχήμα 4.9, παρουσιάζεται ο μέσος όρος των cookies ανά δέκα μέρες. Σε αυτή τη γραφική παράσταση γίνονται εύκολα αντιληπτοί οι δύο διαφορετικοί τρόποι συλλογής των δεδομένων. Στις πέντε πρώτες μπάρες παρατηρούμε ίδια συμπεριφορά στον αριθμό των cookies, με τις τιμές να μην αποκτούν ιδιαίτερα μεγάλες αποκλίσεις. Τα αποτελέσματα των τεσσάρων τελευταίων μπαρών, έχουν εξαχθεί από τη συλλογή δεδομένων με τη χρήση του αυτοματοποιημένου μηχανισμού. Εξαιτίας της επίσκεψης των url οχτώ φορές κάθε μέρα, διαπιστώνεται αυτή η μεγάλη αύξηση του



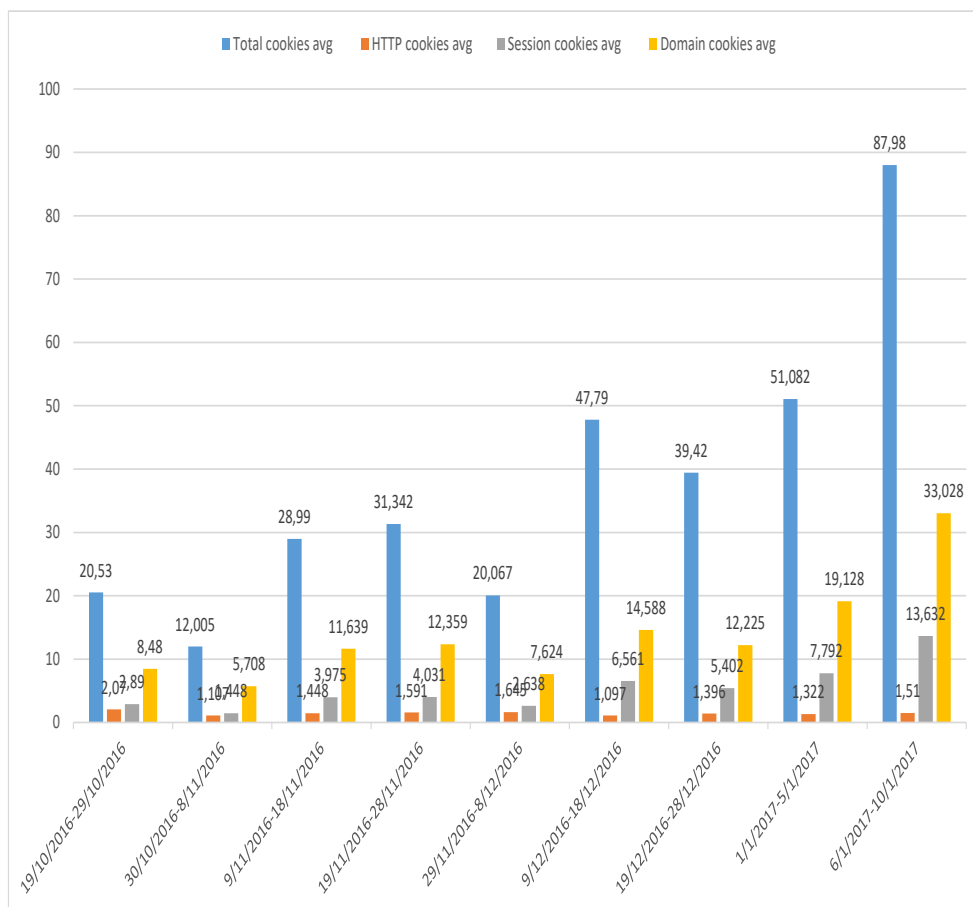
ΣΧΗΜΑ 4.8: Μέσος όρος cookies ανά ημέρα .

αριθμού των cookies σε σχέση με τις προηγούμενες ημερομηνίες.

#### 4.2.3 Γραφικές παραστάσεις με βάση το domain

Σε αυτό το υποκεφάλαιο θα παρουσιάσουμε τις γραφικές παραστάσεις που απεικονίζουν το πλήθος των cookies με τη πάροδο του χρόνου για τρία διαφορετικά domain. Επιπλέον γίνεται η ανάλυση αυτών, καθώς παρουσιάζονται και τα συμπεράσματα τα οποία προήλθαν από αυτές τις γραφικές παραστάσεις.

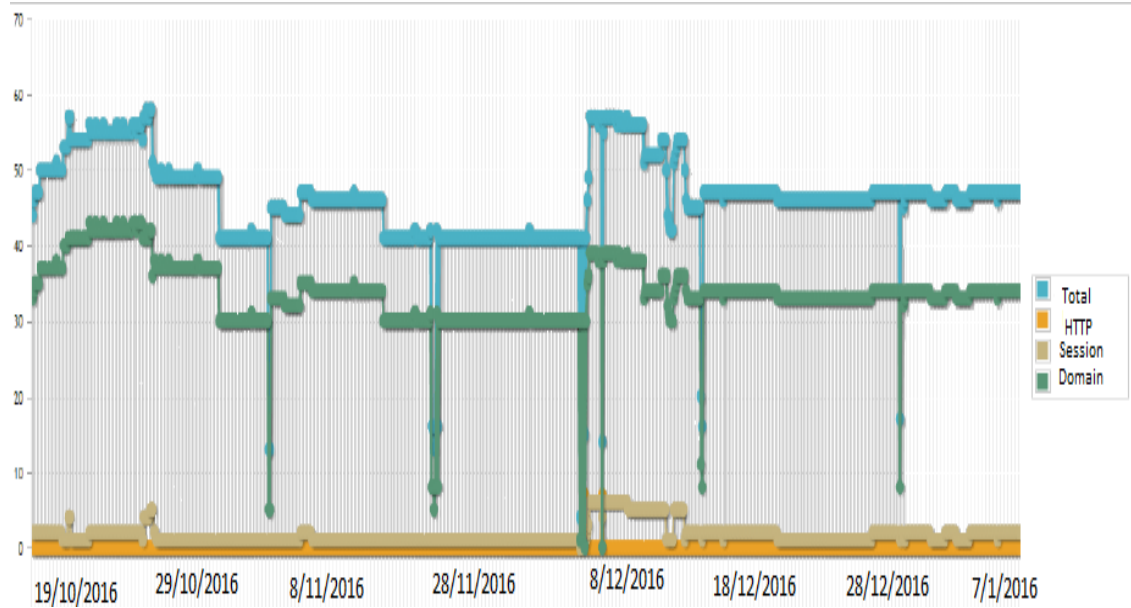
Η γραφική παράσταση που απεικονίζεται στο Σχήμα 4.10, παρουσιάζει τον αριθμό των cookies για το domain του sport24.gr. Γίνεται αντιληπτό ότι από την αρχή μέχρι και τις 08/12/2016, η συλλογή των δεδομένων πραγματοποιούνταν με την τυχαία πλοήγηση του χρήστη στο διαδικτυακό τόπο. Αυτό το συμπέρασμα προκύπτει από τις αυξομειώσεις που παρατηρούμε στον αριθμό των cookies. Ανάλογα με τις επιλογές του χρήστη μπορεί να προστίθενται νέα cookies τα οποία αποθηκεύουν τις προτιμήσεις του. Για παράδειγμα, ο χρήστης μπορεί να ενημερώνεται μόνο για την ομάδα που υποστηρίζει στο διαδικτυακό τόπο, έτσι μπορεί να οριστεί ένα cookie το οποίο να κρατά στοιχεία με βάση αυτό το χαρακτηριστικό. Αντίθετα



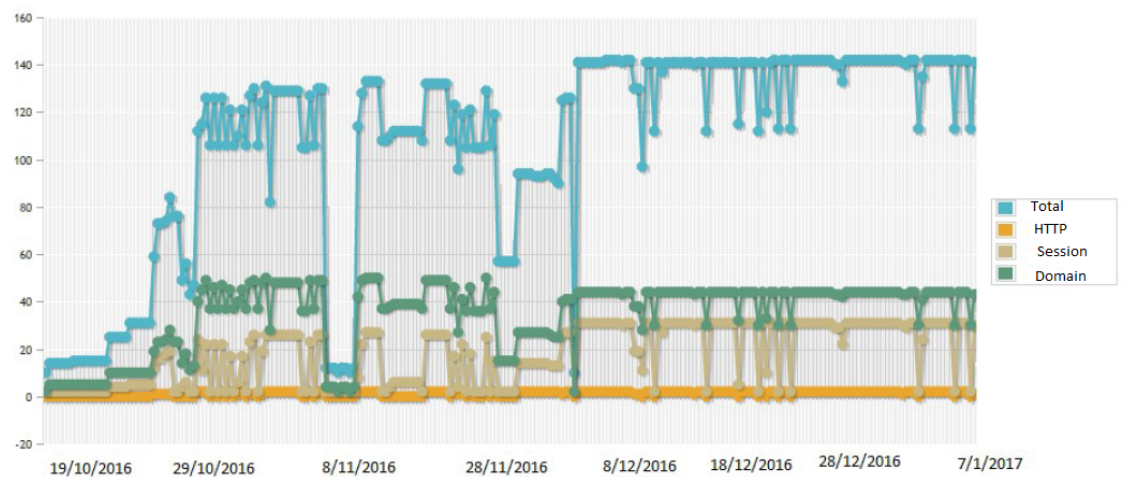
ΣΧΗΜΑ 4.9: Μέσος όρος cookies ανά 10 μέρες.

από τις 18/12/2016 και μετά, η συλλογή των δεδομένων πραγματοποιήθηκε με τον αυτοματοποιημένο μηχανισμό. Με τη χρήση αυτού του μηχανισμού είχαμε την συγκεκριμένη επίσκεψη σελίδων του διαδικτυακού τόπου με αποτέλεσμα ο αριθμός των cookies να μην παρουσιάζει μεταβολές. Αυτό συμβαίνει, διότι η τιμή των cookies που έχουν οριστεί απλά ενημερώνουν τις νέες τιμές χωρίς να προσθέτουν νέα cookies.

Παρόμοια συμπεριφορά παρατηρείται και για το διαδικτυακό τόπο in.gr (Σχήμα 4.11). Από τις 19/10/2016 μέχρι και τις 8/12/2016 παρουσιάζεται μία αυξομείωση του αριθμού των cookies, η οποία οφείλεται και πάλι στις επιλογές του χρήστη. Αντίθετα από τις 8/12/2016 και μετά έχουμε μία σταθερή τιμή στον αριθμό των cookies, για τον ίδιο λόγο που περιγράψαμε και νωρίτερα. Είναι σημαντικό να αναφέρουμε ότι οι απότομες πτώσεις που παρατηρούνται στον αριθμό των cookies, οφείλονται στη δημιουργία στιγμιότυπων του Mozilla Firefox, τα οποία χρησιμοποιούνταν για βελτιώσεις του μηχανισμού. Συγκεκριμένα, κάθε φορά που γίνονταν εκκίνηση ενός στιγμιότυπου του Firefox, κανένα cookie δεν ήταν αποθηκευμένο στο browser καθώς δημιουργούνταν ένα νέο προφίλ, με αποτέλεσμα να πραγματοποιείται αυτή η

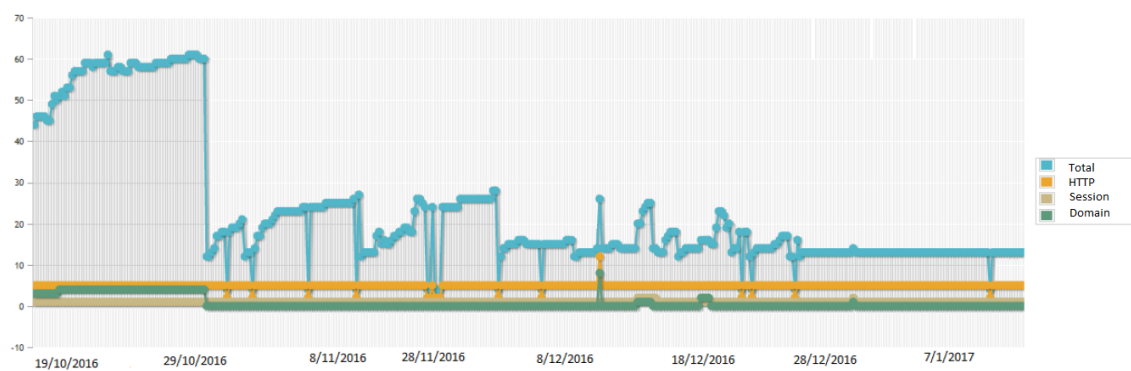


ΣΧΗΜΑ 4.10: Πλήθος cookies για το sport24.gr.



ΣΧΗΜΑ 4.11: Πλήθος cookies για το in.gr.

απότομη πτώση του αριθμού των cookies.

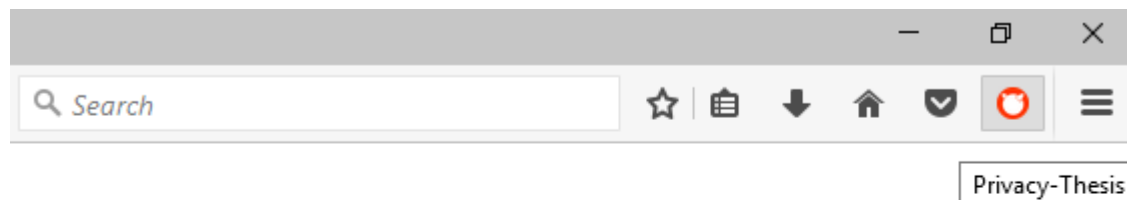


ΣΧΗΜΑ 4.12: Πλήθος cookies για το youtube.com.

Τέλος, η ίδια η γραφική αναπαράσταση για το youtube.com παρουσιάζει τα ίδια αποτελέσματα. Συγκεκριμένα, από την αρχή των μετρήσεων μέχρι και τις 8/12/2016 ο αριθμός των cookies μεταβάλετε. Αυτό οφείλεται στις προτιμήσεις του χρήστη και συγκεκριμένα, ανάλογα με το είδος των βίντεο που παρακολουθεί, προτείνονται παρόμοια βίντεο στο χρήστη. Αυτή η διαδικασία είναι αποτέλεσμα των cookies. Τέλος να σημειώσουμε ότι από τις 8/12/2016 παρατείται περισσότερο σταθερή τιμή στον αριθμό, ειδικά από τις 28/12/2016. Αυτό συμβαίνει διότι στο πρώτο μισό του χρονικού διαστήματος που η συλλογή πραγματοποιούνταν με τον μηχανισμό, πέρα από τη συλλογή με αυτό το τρόπο υπήρχε και η παρέμβαση του χρήστη στο συγκεκριμένο ιστότοπο, με αποτέλεσμα να παρατηρούνται αυτές οι μικρές μεταβολές των cookies. (Σχήμα 4.12)

### 4.3 Γραφική απεικόνιση του μηχανισμού

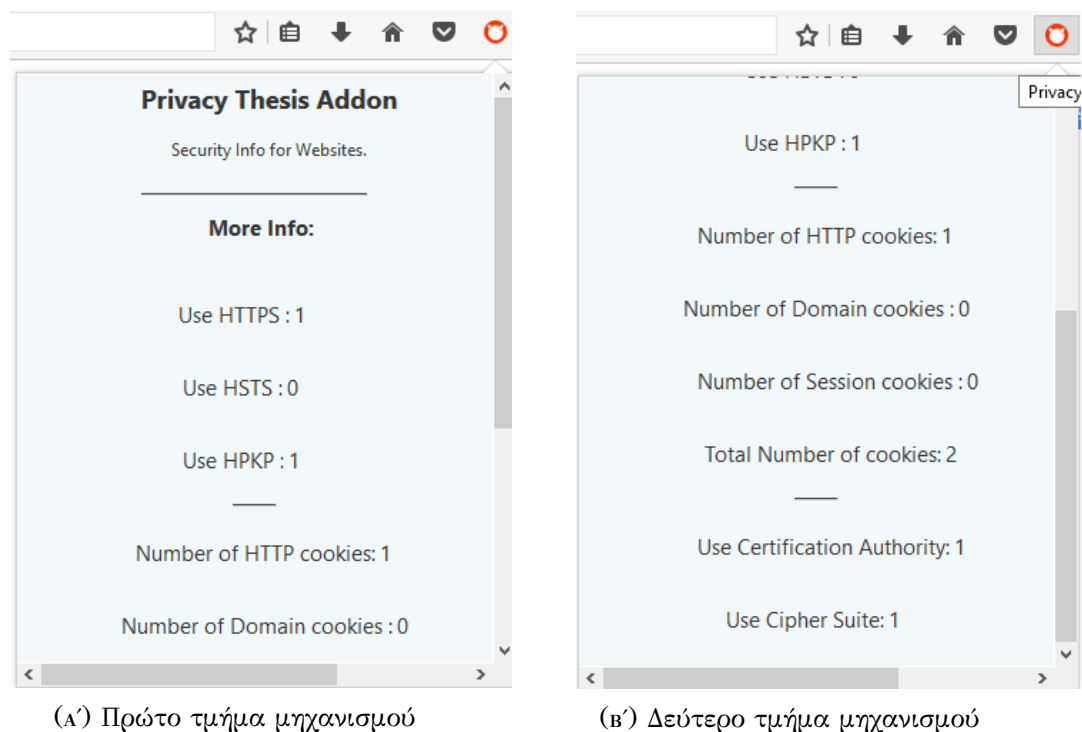
Τέλος, στο παρόν υποκεφάλαιο θα γίνει μία σύντομη περιγραφή της λειτουργίας του add-on που υλοποιήσαμε. Συγκεκριμένα όταν ο χρήστης εγκαταστήσει τον μηχανισμό στο πρόγραμμα περιήγησης Mozilla, τότε στη γραμμή εργαλείων του προγράμματος περιήγησης θα εμφανίζεται το εικονίδιο του add-on όπως φαίνεται και στο Σχήμα 4.13.



ΣΧΗΜΑ 4.13: Εικονίδιο στη γραμμή εργαλείων.

Κάθε φορά που ο χρήστης πατάει το εικονίδιο τότε ένα μικρό παράθυρό ανοίγει στην πάνω δεξιά θέση του προγράμματος ενημερώνοντας τον χρήστη σχετικά με την ασφάλεια για τον συγκεκριμένο ιστότοπο που έχει επισκεφθεί. Στο Σχήμα 4.14 παρουσιάζονται οι πληροφορίες οι οποίες σχετίζονται με τον ιστότοπο google.gr. Αναλυτικά κάθε φορά που θέλουμε να ελέγξουμε την ασφάλεια της σελίδας που επισκεφθήκαμε, ανοίγοντας το add-on καλούνται οι συναρτήσεις του κωδικά και επιστρέφονται οι τιμές. Για τους μηχανισμούς οι οποίοι επιστρέφουν μία boolean μεταβλητή, η τιμή 0 αντιστοιχεί στη τιμή false και κατ' επέκταση στη μη χρήση του μηχανισμού και αντίστοιχα η τιμή 1 αντιστοιχεί στη τιμή true και κατ' επέκταση στη χρήση του μηχανισμού. Κάνοντας scroll down στο παράθυρο που εμφανίζεται μπορούμε να δούμε τον αριθμό των διαφορετικών τύπων cookies τα οποία ανιχνεύονται για το συγκεκριμένο ιστότοπο.

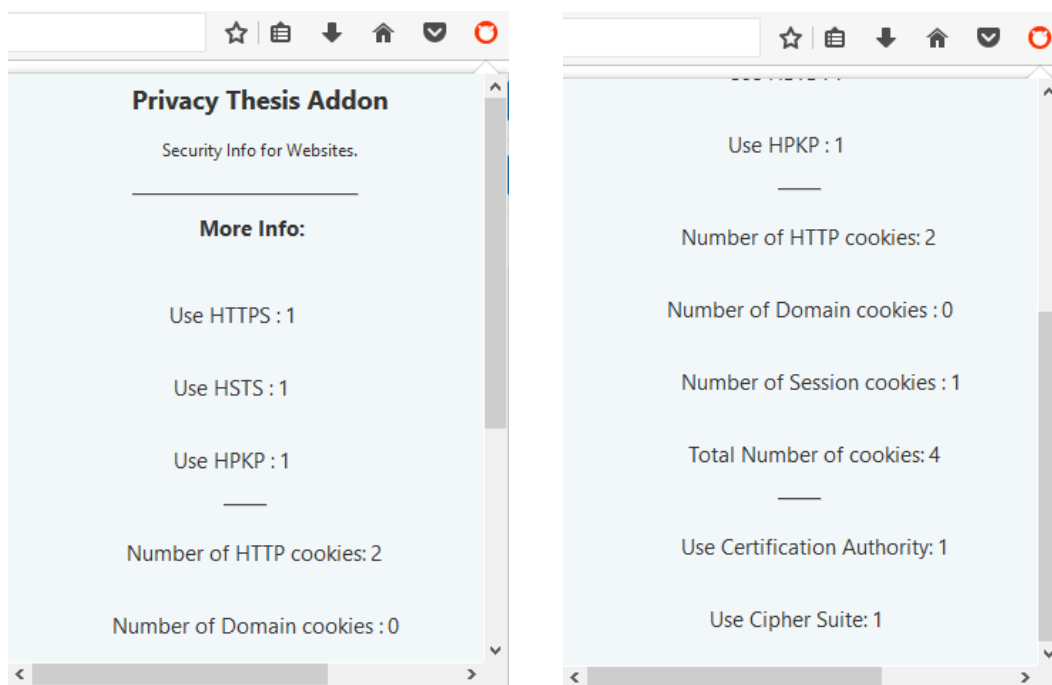




ΣΧΗΜΑ 4.14: Πληροφορίες για τους κινδύνους του διαδικτυακού τόπου google.gr

Στο Σχήμα 4.15 παρουσιάζουμε και τα αποτελέσματα του μηχανισμού όταν επισκεπτόμαστε το youtube.com . Είναι σημαντικό να αναφέρουμε ότι παράλληλα με την διαδικασία που αναφέραμε, δηλαδή την εκτύπωση των αποτελεσμάτων γίνεται και η αποθήκευση αυτών στη βάση δεδομένων.

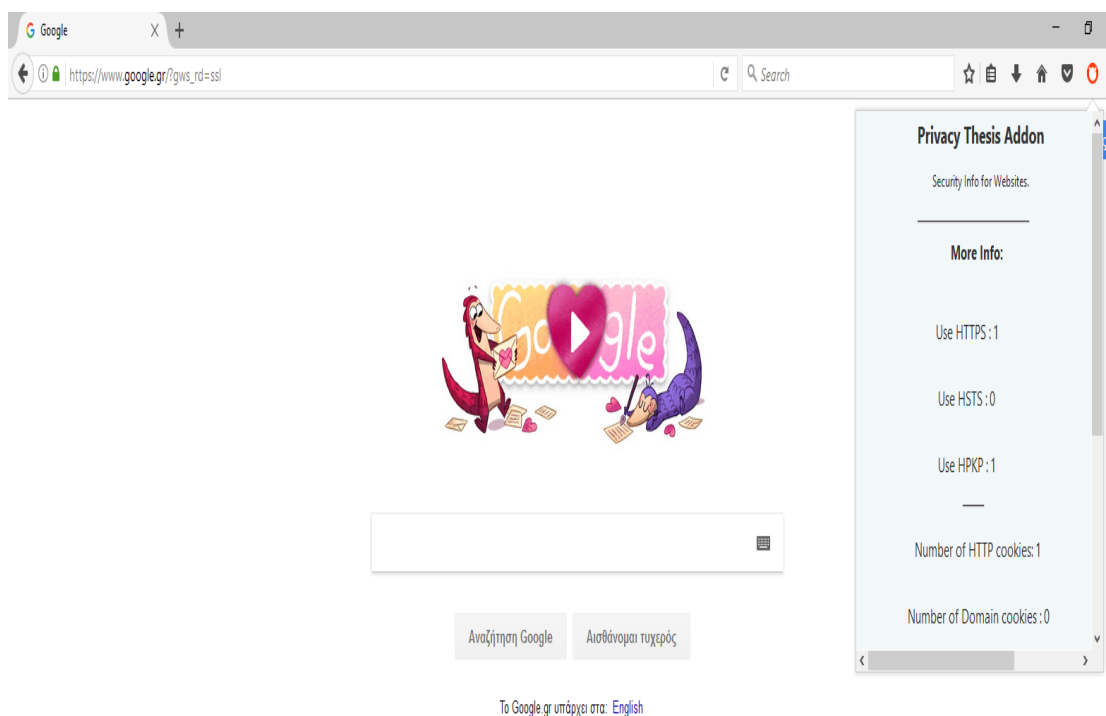
Τέλος, στο Σχήμα 4.16 παρουσιάζεται η γενικό στιγμιότυπο εμφάνισης του μηχανισμού στο πρόγραμμα περιήγησης.



(α') Πρώτο τμήμα μηχανισμού

(β') Δεύτερο τμήμα μηχανισμού

ΣΧΗΜΑ 4.15: Πληροφορίες για τους κινδύνους του διαδικτυακού τόπου youtube.com



ΣΧΗΜΑ 4.16: Εμφάνιση γενικού στιγμιότυπου.

## Κεφάλαιο 5

# Επίλογος

Σε αυτό το κεφάλαιο συνοψίζονται τα συμπεράσματα και τα αποτελέσματα τα οποία εξήχθησαν μετά το πέρας της διπλωματικής εργασίας. Επιπλέον αναφέρονται κάποιες επισημάνσεις και προτείνονται μερικές ιδέες για μελλοντικές επεκτάσεις της εφαρμογής που θα μπορούσαν να υλοποιηθούν.

### 5.1 Σύνοψη και συμπεράσματα

Στα πλαίσια της παρούσας εργασίας παρουσιάστηκε ένας μηχανισμός ανίχνευσης κινδύνων στο διαδίκτυο, αναπτύσσοντας ένα add-on το οποίο υποστηρίζεται από το πρόγραμμα περιήγησης του Mozilla Firefox. Όλα τα δεδομένα που ανιχνεύονταν, αποθηκεύονταν στη βάση δεδομένων τα οποία αξιοποιήθηκαν για την εξαγωγή των συμπερασμάτων.

Μελετώντας τους περισσότερο σημαντικούς κινδύνους που αναπτύσσονται γύρω από την ασφάλεια των προσωπικών δεδομένων, υπήρξαν πολλά συμπεράσματα τα οποία προέκυψαν. Μεγάλος όγκος προσωπικών πληροφοριών εκτίθεται στο διαδίκτυο καθημερινά, αυξάνοντας ιδιαίτερα τον κίνδυνο παραβίασης της ιδιωτικότητας του χρήστη. Πολλές φορές ο χρήστης δεν γνωρίζει σε ποιόν καταλήγουν τα δεδομένα αυτά αλλά και ποίος τα διαχειρίζεται. Υπάρχουν αρκετές μέθοδοι που χρησιμοποιούνται για την παραβίαση των προσωπικών δεδομένων, από κινδύνους που εγχυμονούν κατά τη διάρκεια δημιουργίας μίας επικοινωνίας στο διαδίκτυο (π.χ. man-in-the-middle επίθεση), τη παρακολούθηση της δραστηριότητας του χρήστη στο διαδίκτυο (χρήση cookies, device fingerprint) μέχρι και εργαλεία τα οποία είναι δυνητικά ανασφαλής, με τη χρήση των οποίων μπορεί να εγκατασταθεί κακόβουλο λογισμικό στον ηλεκτρονικό υπολογιστή του χρήστη (π.χ. pdf, java κλπ). Συμπεραίνουμε ότι είναι αναγκαία η δημιουργία μηχανισμών, οι οποίοι θα ενημερώνουν το χρήστη σχετικά με την ασφάλεια των προσωπικών του δεδομένων ή ακόμη καλύτερα η δημιουργία μηχανισμών που θα αντιμετωπίζουν ορισμένους από αυτούς τους κινδύνους.

Χαρακτηριστικό συμπέρασμα της εργασίας μας, είναι ότι μόλις το 41 τοις εκατό του δειγματοχώρου χρησιμοποιεί τον πιο απλό αλλά ιδιαίτερα σημαντικό τρόπο κρυπτογράφησης, που είναι η χρήση του πρωτοκόλλου επικοινωνίας HTTPS. Επίσης διαπιστώνουμε ότι το ποσοστό των ιστότοπων οι οποίοι χρησιμοποιούν όλους τους μηχανισμούς που είναι απαραίτητοι για τη διασφάλιση των δεδομένων είναι εξαιρετικά μικρό. Συνοψίζοντας, καταλήγουμε στο συμπέρασμα ότι είναι ιδιαίτερα σημαντική η ενημέρωση του χρήστη για την προστασία των προσωπικών του δεδομένων, καθώς οι κίνδυνοι έχουν αυξηθεί δραματικά, με τη δημιουργία περισσότερων και πιο προηγμένων μεθόδων παρακολούθησης, γι' αυτό και η προστασία των προσωπικών δεδομένων αποτελεί ένα από τα σημαντικότερα προβλήματα στις μέρες μας.

## 5.2 Μελλοντικές επεκτάσεις

Μία από τις επεκτάσεις που θα μπορούσαν να πραγματοποιηθούν στον μηχανισμό που υλοποιήσαμε, είναι η ανάλυση περισσότερων κινδύνων. Ο μηχανισμός παρέχει πληροφορίες σχετικά με τη δημιουργία ασφαλούς σύνδεσης στο διαδίκτυο, καθώς και ανιχνεύει τα HTTP, Session και Domain cookies. Επομένως θα μπορούσε να ανιχνευθούν περισσότερα είδη cookies όπως τα third Party cookies, τα Flash cookies, τα Supercookies και τα Evercookies. Με αυτό το τρόπο θα ολοκληρώνονταν η κατηγορία των cookies με τη δυνατότητα εξαγωγής καλύτερων συμπερασμάτων σχετικά με τη λειτουργία τους.

Ακόμη μία επέκταση που θα μπορούσε να πραγματοποιηθεί, είναι η δημιουργία ενός webcrawler, η χρήση του οποίου θα έδινε τη δυνατότητα για ανάλυση μεγαλύτερου όγκου δεδομένων. Επιπλέον με την επίσκεψη των ιστότοπων μέσω του webcrawler θα μπορούσε να πραγματοποιηθεί καλύτερη ανάλυση για την συμπεριφορά των cookies και μεταξύ των ιστοσελίδων. Τέλος, θα μπορούσε να σχεδιαστεί μία μετρική συνάρτηση, βάση της οποίας θα εξάγονταν μία τιμή η οποία θα μας ενημερώνει για την ασφάλεια των δεδομένων μας σε ένα διαδικτυακό τόπο. Με αυτό το τρόπο θα μπορούσε να πραγματοποιηθεί μία γραφική απεικόνιση του αποτελέσματος, η οποία είναι θα είναι ευκολότερα κατανοήσιμη από το χρήστη.

# Βιβλιογραφία

- [1] J. R. Mayer and J. C. Mitchell. Third-party web tracking: Policy and technology. *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, 2012.
- [2] Get the facts - don't get scroogled! -scroogled. <http://www.scroogled.com/mail/GetTheFacts/>, 2014. Available.
- [3] Adstack | email optimization platform. <http://adstack.com>, 2014. Available.
- [4] V. Erramilli J. Mikians, L. Gyarmati and N. Laourtis. Detecting price and search discrimination on the internet. *Proceedings of the 11th ACM Workshop on Hot Topics in Networks (Hotnets'12)*, pages 79–84, October 2012.
- [5] Credit card issuers watch online how you shop, customize offers-creditcards.com. <http://www.creditcards.com/credit-cards-news/credit-card-issuers-watching-online-shopping-offers-1273.php/>, 2011. Available.
- [6] Enabling crowd-sourcing based privacy protection for smartphone applications, websites and internet of things deployments. <http://www.privacyflag.eu/>, 2015. Available.
- [7] J. M. Urban C. J. Hoofnagle and S. Li. Privacy and modern advertising. October 2012.
- [8] A. Barth C. Jackson and J. Hodges. Http strict transport security (hsts). 2012.
- [9] Yvo Desmedt. Man-in-the-middle attack. page 759, New York, NY, USA, 2011. Springer US.
- [10] The heartbleed bug. <http://heartbleed.com/>, 2014. Available.
- [11] The drown attack. <https://drownattack.com/>, 2016. Available.
- [12] Openssl. <https://www.openssl.org/>, 2016. Available.
- [13] J. Mayer. The new firefox cookie policy. <https://www.webpolicy.org/2013/02/22/the-new-firefox-cookie-policy/>, February 2013. Available.

- [14] Giving the web a memory cost its users privacy - nytimes.com. <https://www.nytimes.com/2001/09/04/technology/04C00K.html/>, 2011. Available.
- [15] K. McKinley. Cleaning up after cookies. [https://www.isecpartners.com/media/11976/isec\\_cleaning\\_up\\_after\\_cookies.pdf/](https://www.isecpartners.com/media/11976/isec_cleaning_up_after_cookies.pdf/), December 2008. Accessible.
- [16] Http access control (cors)-http- mozilla developer. [https://developer.mozilla.org/en-US/docs/Web/HTTP/Access\\_control\\_CORS/](https://developer.mozilla.org/en-US/docs/Web/HTTP/Access_control_CORS/), 2014. Available.
- [17] Rfc 6265- http state management mechanism - ietf tools. <https://tools.ietf.org/html/rfc6265/>, 2011. Available.
- [18] M. Nasir. Tracking and identifying individual users in a web surfing session. *Computer and Network Security*, January 2014. Accessible.
- [19] A. Soltani N. Good M. Ayenson, D. J. Wambach and C. J. Hoofnagle. Flash cookies and privacy ii: Now with html5 and etag respawning. *Social Science Research Network*, 2011.
- [20] S. Mittal. User privacy and the evolution of third-party tracking mechanisms on the world wide web. <https://purl.stanford.edu/hw48fn9717/>, 2010. Accessible.
- [21] Evercookie - virtually irrevocable persistent cookies. <https://samy.pl/evercookie/>, 2010. Available.
- [22] Local shared object - wikipedia, the free encyclopedia. [https://en.wikipedia.org/wiki/Local\\_shared\\_object/](https://en.wikipedia.org/wiki/Local_shared_object/), 2014. Available.
- [23] Web storage: dom-localstorage - world wide web consortium. <https://dev.w3.org/html5/webstorage/#domlocalstorage/>, 2014. Available.
- [24] Web storage: the-sessionstorage-attribute - world wide web consortium. <https://dev.w3.org/html5/webstorage/#the-sessionstorage-attribute/>, 2014. Available.
- [25] J. Mogul H. Frystyk L. Masinter P. Leach R. Fielding, J. Gettys and Berners-Lee. Hypertext transfer protocol- http/1.1, 1999.
- [26] You deleted your cookies? think again | wired. <https://www.wired.com/2009/08/you-deleted-your-cookies-think-again/>, 2009.
- [27] G. G. Gulyas K. Boda, A. M. Foldes and S. Imre. User tracking on the web via cross-browser fingerprinting. *Information Security Technology for Applications*, pages 31–46, 2012.

- [28] K. Mowery and H. Shacham. Pixel perfect: Fingerprinting canvas in html5. *Web 2.0 Workshop on Security and Privacy (W2SP)*, 2012.
- [29] Ministry of national education- tc milli egitim bakanligi. <https://meb.gov.tr/>, 2014.
- [30] Web audio api. <https://www.w3.org/TR/webaudio/>, December 2015.
- [31] Webrtc project. <https://webrtc.org/web-apis/firefox/>, 2016.
- [32] W3c geolocation api. [https://en.wikipedia.org/wiki/W3C\\_Geolocation\\_API](https://en.wikipedia.org/wiki/W3C_Geolocation_API), 2015.
- [33] e-trust. <http://www.etrust.org/>, 2015.
- [34] Privacy Trust, howpublished = "https://www.privacytrust.com/certification/privacy/index.html", year = 2015,.
- [35] Platform for privacy preferences project. <https://www.w3.org/P3P/>, 2015.
- [36] Ccillin Jackson Adam Barth and John C. Mitchell. Robust defences for cross-site request forgery. pages 75–88, 2008.
- [37] Requestpolicy - firefox addon for privacy and security. <https://requestpolicycontinued.github.io/>, 2014. Available.
- [38] Adblock plus - surf the web without annoying ads! <https://adblockplus.org/>, 2014. Available.
- [39] What information does facebook get when i visit a site with the like button or another social plugin? <https://www.facebook.com/help/186325668085084/>, 2014. Available.
- [40] Atlas solutions. <https://atlassolutions.com/>, 2014. Available.
- [41] Privacy badger. <https://www.eff.org/privacybadger/>, 2014. Available.
- [42] Noscript security suite :: Add-on for firefox - mozilla add-ons. <https://addons.mozilla.org/en/firefox/addon/noscript/>, 2014. Available.
- [43] Ghostery. <https://www.ghostery.com/>, December 2015. Available.
- [44] Flashblock :: Add-ons for firefox-mozilla add-ons. <https://addons.mozilla.org/en/firefox/addon/flashblock/>, 2013. Available.
- [45] Https everywhere. <https://addons.mozilla.org/en/firefox/addon/https-everywhere/>, 2013. Available.

- [46] N. Schmucker. Web tracking. page 9, 2011.
- [47] Web of trust. <https://addons.mozilla.org/el/firefox/addon/wot-safe-browsing-tool/>, 2011.