ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ,ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Διανομή κίνησης σε διακομιστές μεσολάβησης SQUID με αυθεντικοποίηση και κρυπτογράφηση

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Παναγιώτης Θεμιστοκλής Μπαριάμης

Επιβλέπων : Ευστάθιος Συκάς , Καθηγητής ΕΜΠ

Ημερομηνία Αξιολόγησης: 11 Ιουνίου 2018

Αθήνα , Ιούνιος 2018

Αθήνα , Ιούνιος 2018

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ,ΗΛΕΚΤΡΟΝΙΚΗΣ ΚΑΙ ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Διανομή κίνησης σε διακομιστές μεσολάβησης SQUID με αυθεντικοποίηση και κρυπτογράφηση

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Παναγιώτης Θεμιστοκλής Μπαριάμης

Επιβλέπων : Ευστάθιος Συκάς, Καθηγητής ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 11 Ιουνίου 2018

……	……	……
Ευστάθιος Συκάς	Γεώργιος Στασσινόπουλος	Ιωάννα Ρουσσάκη
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Επ. καθηγήτρια Ε.Μ.Π.

Αθήνα , Ιούνιος 2018

……

Παναγιώτης Θεμιστοκλής Μπαριάμης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Σκοπός της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη και η λειτουργία μιας συστοιχίας διακομιστών μεσολάβησης που θα τεθεί σε λειτουργία στο Ελληνικό Δίκτυο Σχολείων, προκειμένου να επιτευχθεί η ταυτοποίηση χρηστών που χρησιμοποιούν το internet στα σχολεία και θα αποτρέψει την χρήση του σχολικού δικτύου σε οποιοδήποτε μη πιστοποιημένο ή εξωτερικό χρήστη. Η επιλογή της κατάλληλης διαδρομής γίνεται με βάση την τοπολογία και τις αποφάσεις του διαχειριστή του δικτύου.

Για την ανάπτυξη του μηχανισμού αυτού, χρησιμοποιήθηκε ο διακομιστής μεσολάβησης ανοιχτού λογισμικού squid ο οποίος τρέχει σε λειτουργικό σύστημα FreeBSD. Ο διακομιστής μεσολάβησης αυτός διατίθεται δωρεάν και χρησιμοποιεί αρκετά πρόσθετα πακέτα τα οποία έχουν αναπτυχθεί από την ανοιχτή κοινότητα λογισμικού. Βασικό συστατικό είναι ο διακομιστής πιστοποίησης (ldap server) ο οποίος αναλαμβάνει την πιστοποίηση χρηστών μέσω των διακομιστών μεσολάβησης για τους χρήστες των σχολικών δικτύων.

Επιπλέον, χρησιμοποιήθηκαν τα λογισμικά HAPRoxy και WebPolygraph.

Ο HAPRoxy είναι ένας ανοιχτού λογισμικού load balancer ο οποίος λειτουργεί σε layer 4 & layer 7 και ακολουθεί πιστά το http πρωτόκολλο με αποτέλεσμα να μπορεί να συνεργαστεί καλύτερα με τον squid server.

Τέλος το web polygraph αποτελεί ένα framework για stress test μέσω εξομοίωσης των συνθηκών πιστοποίησης και ανάκτησης ιστοσελίδων σε περιβάλλον υψηλής ζήτησης. Έχει την δυνατότητα δημιουργίας δοκιμαστικών μέσω scripts τα οποία στη συνέχεια αυτοματοποιούν τη διαδικασία παραγωγής υψηλής και ταυτόχρονης κίνησης προς τους διακομιστές. Το web polygraph είναι το μόνο framework που υποστηρίζει client to proxy encryption καθώς και Kerberos authentication.

Λέξεις Κλειδιά : squid proxy server, authentication, load-balancing, ldap, https proxy, BSD squid, Secure TLS Web Proxy, Web Polygraph.

Abstract

The objective of the present thesis is the development and configuration of an array of proxy servers that will serve the Greek School Network, so as to achieve authentication of users that are allowed to use the proxy servers and will deny access to persons not authorized by the Greek School Network to use its Internet Services.

For the development of such a project, squid proxy server was used in conjunction with HAPRoxy server running under FreeBSD 10.4. Squid Proxy Server is an open source proxy server freely distributed and actively developed by a huge community. Basic infrastructure element of the backend authentication is the OPENLDAP server that the Greek School Network already uses for authentication.

HAProxy is an open source layer 4 & 7 load balancer that strictly implements http protocol, so its use with squid proxy is highly integrated

Finally, web polygraph was used for stress testing. Web Polygraph is a framework for stress testing simulating authentication conditions and proxy server stress testing under high load. Web polygraph is the only framework that supports client to proxy encryption as well as Kerberos authentication testing.

Keywords: squid proxy server, authentication, load-balancing, ldap, https proxy, BSD squid, Secure TLS Web Proxy, Web Polygraph.

Ευχαριστίες

Η παρούσα διπλωματική εργασία εκπονήθηκε κατά το ακαδημαϊκό έτος 2017-2018 στον τομέα Επικοινωνιών, Ηλεκτρονικής και Συστημάτων Πληροφορικής της Σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Εθνικού Μετσόβιου Πολυτεχνείου.

Υπεύθυνος κατά την εκπόνηση της διπλωματικής ήταν ο καθηγητής κ. Ευστάθιος Συκάς στον οποίο οφείλω ιδιαίτερες ευχαριστίες για την ανάθεση αυτής και την δυνατότητα που μου δόθηκε να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα. Θα ήθελα επίσης να ευχαριστήσω θερμά τον διδάκτορα ερευνητή ΕΠΙΣΕΥ κ. Δημήτρη Καλογερά για την υποστήριξη και την καθοδήγηση που μου παρείχε κατά την συγγραφή της εργασίας.

Κυρίως όμως θέλω να εκφράσω την ευγνωμοσύνη μου στον πατέρα μου Ανδρέα και τη μητέρα μου Μαρία για την αμέριστη υποστήριξη τους και θυσίες τους όλα αυτά τα χρόνια. Τέλος, θα ήταν παράλειψη να μην ευχαριστήσω την αδερφή μου Γεωργία για την διάθεση, υποστήριξη και την κατανόηση που επέδειξε σε όλη μου την πορεία.

Παναγιώτης Θεμιστοκλής Μπαριάμης

Αθήνα, Ιούνιος 2018

ΠΙΝΑΚΑΣ ΠΕΡΙΟΧΟΜΕΝΩΝ

1 ΕΙΣΑΓΩΓΗ

1.1 ΣΚΟΠΟΣ ΤΗΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Η έκρηξη των κινητών συσκευών, η πολιτική φέρε τη συσκευή σου και η εμφάνιση των υπηρεσιών σύννεφου (cloud services) έχουν οδηγήσει την βιομηχανία δικτύωσης να επανεξετάσει τις αρχιτεκτονικές δικτύου. Οι παραδοσιακές αρχιτεκτονικές δικτύου θεωρούνται ακατάλληλες για να αντιμετωπίσουν τις σημερινές ανάγκες των υπηρεσιών, των διακομιστών και των χρηστών.

Οι διακομιστές μεσολάβησης πλέον δεν είναι μόνο αναγκαίοι για την τοπική αποθήκευση του περιεχομένου αλλά πλέον κάνουν και άλλες εργασίες, από αυθεντικοποίηση των χρηστών μέχρι και/η ssl offloading και αποκρυπτογράφηση της κίνησης που διέρχεται από ένα δίκτυο ώστε να ελεγχθεί το περιεχόμενό της. Σε πολλές περιπτώσεις κρίνεται αναγκαίος ο έλεγχος της κίνησης ειδικά σε σχολικά δίκτυα και εταιρικά περιβάλλοντα.

Η παρούσα διπλωματική εργασία έχει ως σκοπό την ανάπτυξη ενός διακομιστή μεσολάβησης ο οποίος θα συνδυάζει εκτός από την τοπική αποθήκευση του περιεχομένου με την κλασσική μέθοδο του διακομιστή μεσολάβησης, αλλά εκτός από την αυθεντικοποίηση θα επεκτείνει και τις δυνατότητες σε ελέγχους αποκρυπτογράφησης καθώς και ελέγχου του περιεχομένου των χρηστών. Έτσι θα μπορέσει να διασφαλιστεί η ασφάλεια εντός του σχολικού δικτύου καθώς και θα καταγράφεται η κίνηση του κάθε χρήστη σύμφωνα με ένα μοναδικό όνομα χρήστη και συνθηματικό το οποίο έχει λάβει από το sch.gr (Πανελλήνιο Σχολικό Δίκτυο – ΠΣΔ).

Θα δοκιμαστούν SSO μέθοδοι για την χρησιμοποίηση του διακομιστή μεσολάβησης με αυθεντικοποίηση καθώς και άλλες μέθοδοι αυθεντικοποίησης.

1.2 ΔΟΜΗ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Η διπλωματική εργασία αποτελείται από 6 κεφάλαια. Στο πρώτο κεφάλαιο, αναπτύχθηκαν οι ανάγκες που οδήγησαν στην ανάγκη της δημιουργίας της διπλωματικής εργασίας και η οργάνωση του κειμένου.

Στο δεύτερο κεφάλαιο, παρουσιάζεται το θεωρητικό υπόβαθρο για τους μηχανισμούς που προσφέρει το πρωτόκολλο http για αυθεντικοποίηση.

Στο τρίτο κεφάλαιο, αναλύεται ο διακομιστής μεσολάβησης ανοιχτού κώδικα squid ο οποίος χρησιμοποιείται στην παρούσα διπλωματική εργασία. Πιο συγκεκριμένα αναλύεται η παραμετροποίηση του σε διάφορα test cases (http authentication, https authentication, ssl man in the middle, https authentication με ssl offloading από τον load balancer).

Στο τέταρτο κεφάλαιο, η αρχιτεκτονική δικτύου και η δομή της. Στη συνέχεια αναλύεται το πρωτόκολλο http/https και το κομμάτι αυτού που υποστηρίζει το proxying, παρουσιάζονται εν συντομία οι πιο γνωστοί proxy servers και πως αυτοί λειτουργούν, ενώ στο τέλος αναπτύσσεται η δομή του proxy server squid.

Στο πέμπτο κεφάλαιο παρουσιάζονται τα στοιχεία δικτύου καθώς και οι τρόποι με τους οποίες έγιναν οι δοκιμές και τα stress test. Επίσης αναλύονται τα λογισμικά που χρησιμοποιήθηκαν για τα stress test καθώς και για το load balancing. Παρουσιάζονται επίσης τα αποτελέσματα των stress test καθώς και πως αυτά επιβαρύνουν ή όχι την υπάρχουσα υποδομή.

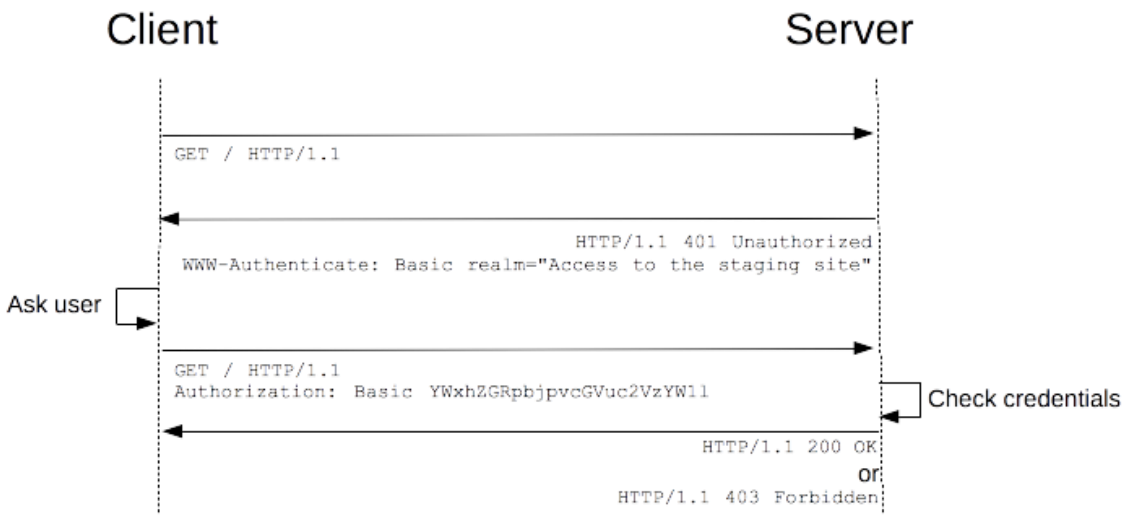
Στο έκτο κεφάλαιο παρουσιάζονται τα συμπεράσματα, ο δυνατός τρόπος λειτουργίας με αυθεντικοποίηση και προτείνονται τρόποι αντιμετώπισης των προβλημάτων τα οποία προέκυψαν κατά την συγγραφή αυτής της διπλωματικής.

Στο Παράρτημα παρατίθενται όλα τα scripts / κώδικας / ρυθμίσεις για όλα τα στοιχεία δικτύου που χρησιμοποιήθηκαν καθώς και για το framework stress test Web Polygraph.

2 AUTHENTICATED PROXYING

2.1 ΓΕΝΙΚΟ HTPP AUTHENTICATION FRAMEWORK

Το RFC 7235 καθορίζει ακριβώς το HTTP authentication framework το οποίο μπορεί να χρησιμοποιηθεί από ένα διακομιστή ώστε να ζητήσει από ένα πελάτη τα διαπιστευτήρια και ο πελάτης να αποστείλει αυτά τα διαπιστευτήρια. Η ροή αυτή της διαδικασίας challenge – response έχει ως εξής: Ο διακομιστής απαντάει σε έναν πελάτη με 401 (Unauthorized) και παρέχει πληροφορίες πως να γίνει η διαπίστευση με μια απάντηση στην οποία οι κεφαλίδες Authenticate περιέχουν τουλάχιστον έναν τρόπο διαπίστευσης. Ο πελάτης ο οποίος θέλει να χρησιμοποιήσει τις πληροφορίες του διακομιστή θα στείλει ένα αίτημα με την κεφαλίδα Authorization στην οποία θα περιλαμβάνονται τα διαπιστευτήρια. Συνήθως, στον πελάτη θα εμφανιστεί ένα παράθυρο στο οποίο ο χρήστης πρέπει να συμπληρώσει το όνομα χρήστη και κωδικό.



Εικόνα 1: Μηχανισμός HTTP Auth

2.1.1 Proxy authentication

Ο ίδιος μηχανισμός που είδαμε παραπάνω μπορεί να χρησιμοποιηθεί και για authentication σε έναν διακομιστή μεσολάβησης. Για να μπορέσει να συνυπάρξει ένας διακομιστής μεσολάβησης και ένας διακομιστής τελικού προορισμού ο οποίος απαιτεί authentication, χρησιμοποιούνται διαφορετικές κεφαλίδες για το διακομιστή μεσολάβησης. Στην περίπτωση των διακομιστών μεσολάβησης το challenge είναι το 407 (Proxy Authentication Required), η κεφαλίδα Proxy-Authenticate της απάντησης περιέχει τουλάχιστον έναν τρόπο authentication τον οποίο υποστηρίζει ο διακομιστής μεσολάβησης, και η Proxy-Authorization κεφαλίδα του αιτήματος χρησιμοποιείται για την αποστολή των διαπιστευτηρίων στον διακομιστή μεσολάβησης.

2.1.2 WWW-Authenticate και Proxy-Authenticate κεφαλίδες

Οι κεφαλίδες απάντησης WWW-Authenticate και Proxy-Authenticate καθορίζουν την μέθοδο αυθεντικοποίησης η οποία μπορεί να χρησιμοποιηθεί για την πρόσβαση σε υπηρεσίες. Πρέπει να ορίζεται ποιο authentication scheme χρησιμοποιείται, ώστε ο πελάτης ο οποίος θέλει να πιστοποιηθεί για την υπηρεσία να γνωρίζει με ποιο τρόπο να αποστείλει τα διαπιστευτήρια. Η σύνταξη αυτών των κεφαλίδων έχει ως εξής :

```
WWW-Authenticate: <type> realm=<realm>
Proxy-Authenticate: <type> realm=<realm>
```

Ο <type> είναι το authentication scheme (πχ Basic). Το realm χρησιμοποιείται για να περιγράψει μια προστατευμένη περιοχή ή για να ορίσει το πεδίο εφαρμογής της πιστοποίησης (πχ πολλοί διακομιστές με μια αυθεντικοποίηση)

2.1.3 Authorization και Proxy-Authorization κεφαλίδες

Αντίστοιχα οι κεφαλίδες Authorization και Proxy-Authorization περιέχουν τα διαπιστευτήρια, ώστε να πιστοποιηθεί ένας χρήστης πελάτης με έναν διακομιστή (μεσολάβησης). Εδώ ο τύπος είναι πάλι απαιτούμενος και ακολουθείται από τα διαπιστευτήρια, τα οποία μπορεί να είναι encoded ή κρυπτογραφημένα ανάλογα με το ποιο authentication scheme έχει χρησιμοποιηθεί.

```
Authorization: <type> <credentials>
Proxy-Authorization: <type> <credentials>
```

2.1.4 Authentication schemes

Το HTTP authentication framework μπορεί να χρησιμοποιήσει πολλά σχήματα αυθεντικοποίησης.

Το πιο σύνηθες σχήμα αυθεντικοποίησης είναι το Basic. Παρακάτω ακολουθούν τα σχήματα αυθεντικοποίησης:

- Basic
- Digest
- NTLM
- Negotiate (για ntlm ή Kerberos)
- Bearer

2.2 PROXYING ΣΤΟ HTTP ΠΡΩΤΟΚΟΛΛΟ

Στο http πρωτόκολλο υπάρχουν 5 είδη μεσολάβησης :

- Διαφανής μεσολάβηση (transparent proxying)
- Μεσολάβηση προώθησης (forwarding proxying)
- Μεσολάβηση προώθησης με κρυπτογράφηση (tls termination proxying)
- Αντίστροφη προώθηση (reverse proxying)
- Αντίστροφη διαμεσολάβηση κρυπτογραφημένων συνδέσεων (reverse tls termination proxying)

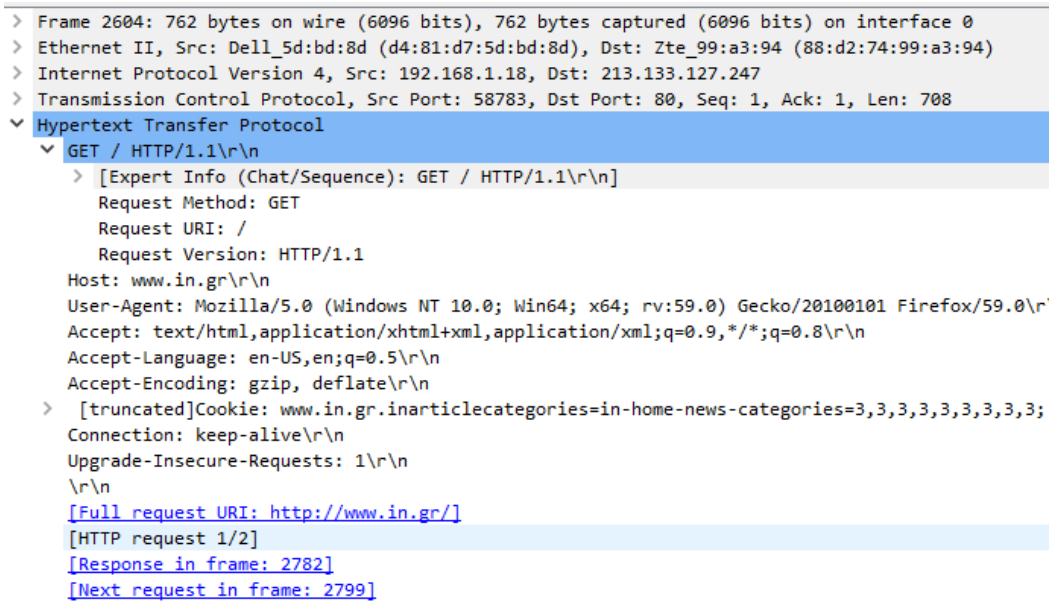
Σε αυτή την διπλωματική εργασία ασχολούμαστε με τους 3 πρώτα είδη μεσολάβησης.

Επειδή η μεσολάβηση προώθησης με κρυπτογράφηση μεταξύ χρήστη πελάτη και διακομιστή μεσολάβησης έχει αρχίσει να υποστηρίζεται από τους web browsers μόνο τα 2 τελευταία χρόνια, η επικρατούσα ονομασία για αυτούς καθώς δεν είναι διαδεδομένοι αυτή τη στιγμή, είναι Secure Web Proxy. Κάθε άλλη ονομασία https proxy ή tls termination proxy (λόγω της χρησιμοποίησης αυτών των όρων αντίστοιχα για διακομιστές μεσολάβησης που υποστηρίζουν το https πρωτόκολλο ή για διακομιστές μεσολάβησης οι οποίοι αναλαμβάνουν τον τερματισμό συνδέσεων tls για την σύνδεση σε backend που δεν υπάρχει κρυπτογράφηση), δεν θα χρησιμοποιηθεί στην παρούσα διπλωματική καθώς περισσότερο συγχέει με τους όρους που έχουν επικρατήσει. Στο εξής κάθε διακομιστής

μεσολάβησης που θα χρησιμοποιεί κρυπτογράφηση μεταξύ χρήστη πελάτη και του ιδίου θα αναφέρεται σαν Secure TLS Web Proxy.

2.2.1 HTTP WITHOUT PROXYING

Σε κανονικές συνθήκες χωρίς proxy μπορούμε να δούμε ότι το http αίτημα στέλνεται απευθείας από τον browser προς τον τελικό προορισμό με τις ακόλουθες κεφαλίδες



Εικόνα 2: Http No Proxy Request

Αυτό που έχει σημασία είναι η κεφαλίδα GET, στην οποία παρατηρούμε ότι το GET ακολουθείται από HTTP /1.1, ενώ το πλήρες αίτημα βρίσκεται εντός του payload με το [Full request URI: http://www.in.gr/]. Αυτή είναι και η κανονική λειτουργία του πρωτοκόλλου και ισχύει και στις περιπτώσεις των σεναρίων χωρίς διακομιστή μεσολάβησης καθώς και στις περιπτώσεις των σεναρίων με διαφανείς διακομιστές μεσολάβησης (transparent proxy server).

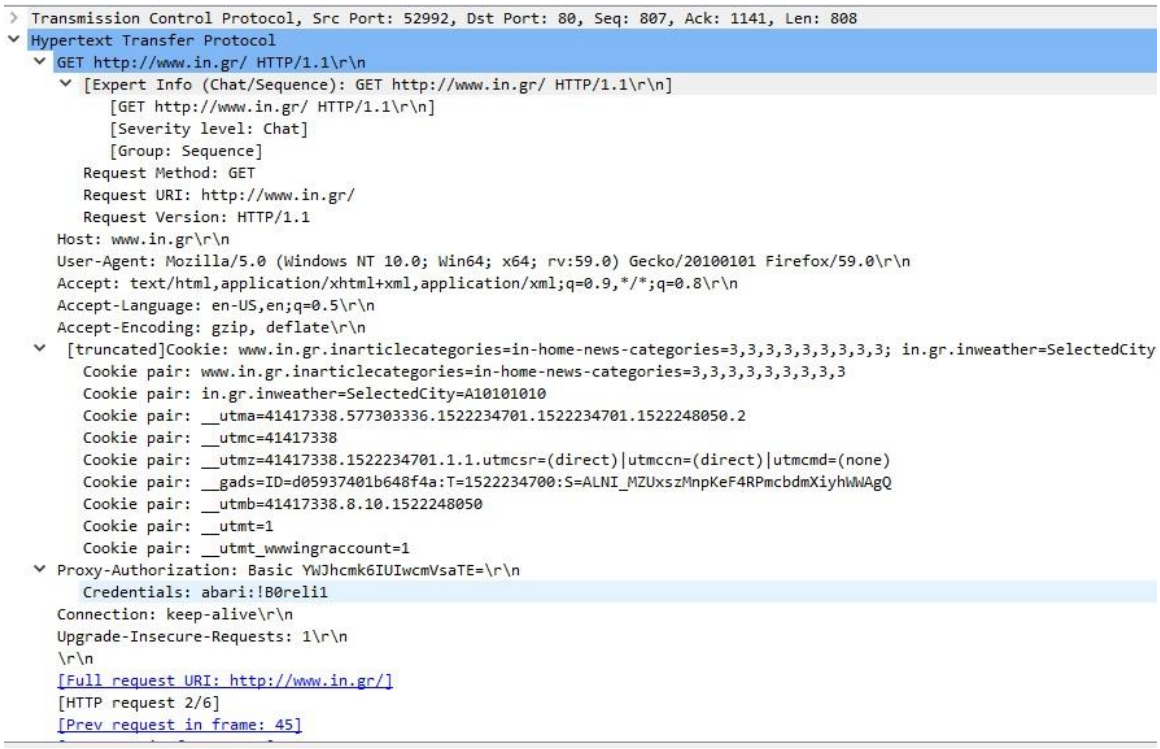
Στην περίπτωση του διαφανούς διακομιστή μεσολάβησης, αυτός αναλαμβάνει απλώς να προωθήσει το αίτημα σε περίπτωση που δεν βρει τοπικό αντίγραφο του περιεχομένου που ζητείται. Στο παρελθόν αυτό ήταν αποδοτικό όταν τα περισσότερα site ήταν στατικά και η επικοινωνία με αυτά ήταν αποκρυπτογραφημένη. Πλέον, τα τελευταία χρόνια οι περισσότεροι ιστότοποι και υπηρεσίες προσφέρουν κρυπτογραφημένη επικοινωνία με αποτέλεσμα ο διακομιστής μεσολάβησης να μην έχει πρόσβαση στο περιεχόμενο η/και τις υπηρεσίες οι οποίες προσφέρονται με αποτέλεσμα να μην χρησιμοποιείται καθόλου η τοπικότητα των δεδομένων. Επιπλέον σε κάθε request ακόμα και αν είναι ακριβώς ίδιο με το προηγούμενο να χρειάζεται να ξαναφέρει όλο το περιεχόμενο επιβαρύνοντας και την διεκπαιρεωτική ικανότητα της γραμμής.

2.2.2 HTTP authenticated forward proxying

Όταν χρησιμοποιούμε το προηγούμενο σενάριο με διαφανή μεσολάβηση, δεν μπορεί να χρησιμοποιηθεί κανένα άλλο εργαλείο όπως αυθεντικοποίηση, η μεσολάβηση στο tls ή deep packet inspection. Για αυτές τις υπηρεσίες, χρειάζεται η διαμεσολάβηση να ορίζεται ρητά στον χρήστη μέσω του browser του. Η πιο απλή περίπτωση είναι να εισάγει ο χρήστης χειροκίνητα τη διεύθυνση του διακομιστή μεσολάβησης και όλες οι συνδέσεις να περνάνε μέσω του διακομιστή μεσολάβησης.

Το πρωτόκολλο http έχει ειδική μεταχείριση για τις περιπτώσεις των μεσολαβητών προώθησης όπως περιγράφεται στο RFC 2068.

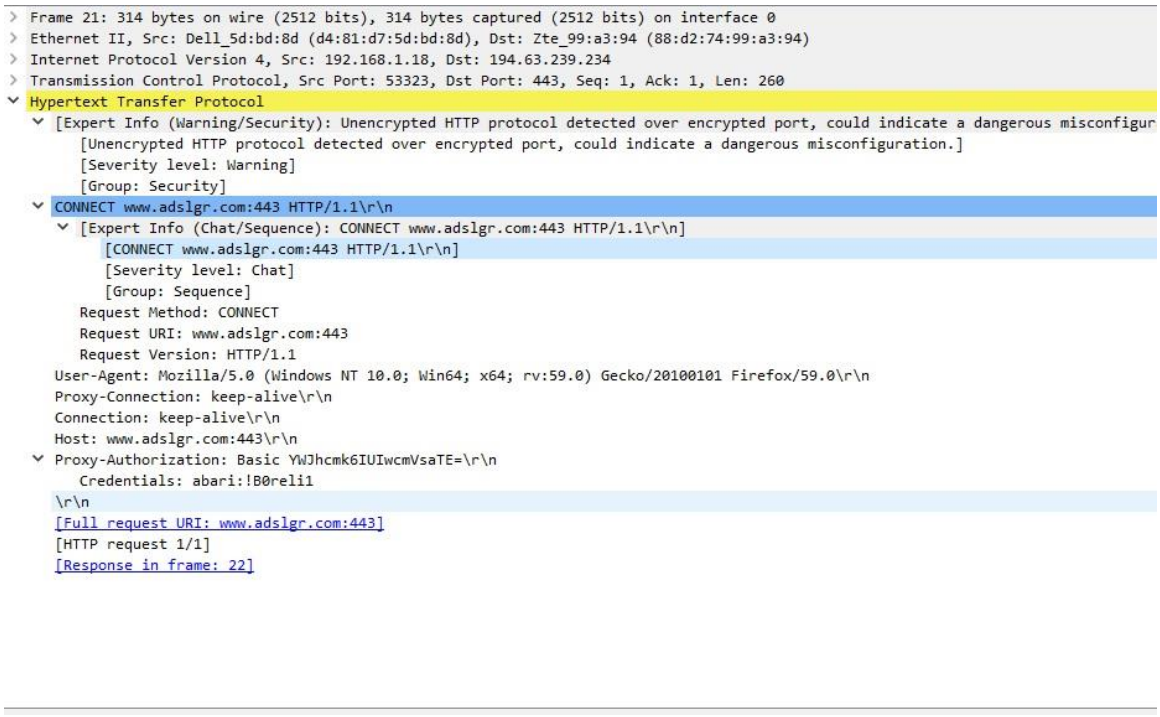
Μπορούμε να δούμε στην παρακάτω καταγραφή ότι όταν έχει δηλωμένο ο browser το διακομιστή μεσολάβησης τότε αλλάζει και η μορφή του request όπως φαίνεται στην παρακάτω εικόνα:



Εικόνα 3: Http With Proxy Request

Σε αυτή την καταγραφή παρατηρούμε πλέον ότι στις κεφαλίδες το GET πεδίο άλλαξε και πλέον ρητά αναφέρεται και η διεύθυνση του διακομιστή, την οποία ζητάμε ώστε να τη γνωρίζει ο διακομιστής μεσολάβησης. Επιπλέον, παρατηρούμε ότι στις κεφαλίδες προστέθηκε και το Proxy-Authorization, το οποίο περνάει με plain text στον διακομιστή μεσολάβησης το όνομα χρήστη και το συνθηματικό. Αυτό όμως είναι ένα πρόβλημα το οποίο παρατηρείται σε πολλά περιβάλλοντα και μπορεί να επιλυθεί με διάφορους τρόπους.

Το προηγούμενο request ήταν για ένα διακομιστή τελικού προορισμού που δεν χρησιμοποιεί κρυπτογράφηση. Στην επόμενη καταγραφή, έχουμε μια αίτηση για ένα διακομιστή τελικού προορισμού ο οποίος χρησιμοποιεί κρυπτογράφηση. Για αυτή την περίπτωση υπάρχει ειδική μέριμνα στο πρωτόκολλο https, το οποίο όταν ζητάμε σύνδεση με ένα διακομιστή που χρησιμοποιεί κρυπτογράφηση, μέσω ενός διακομιστή μεσολάβησης χρησιμοποιεί την εξής διαφοροποίηση όπως στην εξής καταγραφή :



Εικόνα 4: Https Proxy Request

Σε αυτή την καταγραφή παρατηρούμε ότι πλέον έχει αλλάξει εντελώς το αίτημα και αντί για HTTP GET είναι πλέον HTTP CONNECT. Το HTTP CONNECT ενημερώνει τον διακομιστή μεσολάβησης ότι θα γίνει αίτηση σε ένα διακομιστή που χρησιμοποιεί κρυπτογράφηση. Στη συγκεκριμένη περίπτωση πλέον ο διακομιστής μεσολάβησης απλώς αναλαμβάνει τη δρομολόγηση περιεχομένου, χωρίς αυτός να μπορεί να αποθηκεύσει τοπικά το περιεχόμενο (caching), όπως επίσης και να μην μπορεί να μπλοκάρει κίνηση η οποία υπάρχει σε φίλτρα περιεχομένου (στο πεδίο connect γνωρίζουμε μόνο το όνομα τομέα και όχι όλο το url). Παρόλα αυτά, πάλι παρατηρούμε ότι τα διαπιστευτήρια του χρήστη μεταδίδονται σε plain text μεταξύ του πελάτη και του διακομιστή μεσολάβησης, γεγονός που καθιστά και αυτή την περίπτωση επισφαλή.

2.2.3 TLS termination proxying

Οι προηγούμενες περιπτώσεις δεν μπορούσαν να παρεμβληθούν στις συνδέσεις μεταξύ του χρήστη και του τελικού διακομιστή, ο οποίος χρησιμοποιεί κρυπτογράφηση. Για να ξεπεράσουμε αυτό το εμπόδιο πλέον ο διακομιστής μεσολάβησης θα λειτουργεί ακριβώς όπως και μια man-in-the-middle επίθεση. Ο διακομιστής μεσολάβησης θα και είναι αρχή πιστοποίησης, η οποία είναι έμπιστη στον χρήστη και δημιουργεί πιστοποιητικά για το κάθε διακομιστή τελικού προορισμού on-the-fly. Αυτό έχει ως αποτέλεσμα το HTTP CONNECT αίτημα πλέον να τερματίζεται από τον διακομιστή μεσολάβησης, αυτός να έχει πρόσβαση σε όλα τα περιεχόμενα των πακέτων, και από εκεί να ξεκινάει μια καινούρια σύνδεση προς τον τελικό διακομιστή. Τελικά αφού φέρει τα δεδομένα από τον τελικό διακομιστή κρυπτογραφεί τα δεδομένα με το πιστοποιητικό που δημιούργησε πριν και το στέλνει στον τελικό χρήστη. Ο τελικός χρήστης από τη στιγμή που έχει εμπιστευτεί την δικιά μας αρχή πιστοποίησης, δεν αντιλαμβάνεται κάποια διαφοροποίηση και ο browser του εμφανίζει κανονικά ότι η σύνδεση είναι ασφαλής και ότι ο διακομιστής μεσολάβησης είναι στην ουσία ο διακομιστής τελικού προορισμού ο οποίος είχε ζητήσει εξαρχής.

Έτσι μπορούμε να έχουμε πρόσβαση σε όλο το περιεχόμενο το οποίο διακινείται στο δίκτυο μας, και παράλληλα να κάνουμε και caching σε περιεχόμενο στο οποίο δεν είχαμε πρόσβαση πρωτύτερα. Επιπλέον τώρα μπορούμε να εφαρμόσουμε φιλτράρισμα URL σε όλο το URL.

2.3 SECURE TLS WEB PROXYING (HTTPS PROXYING)

Σε όλες τις προηγούμενες εξεταζόμενες περιπτώσεις όλη η επικοινωνία μεταξύ πελάτη (browser ή άλλης υπηρεσίας) γίνεται σε plain text άσχετα με το αν εμείς ζητάμε απλή υπηρεσία ή κρυπτογραφημένη. Η συγκεκριμένη λειτουργία δεν είχε πρόβλημα όταν εφαρμαζόταν transparent proxy. Πλέον όμως όλες οι υπηρεσίες που χρησιμοποιούν πιστοποίηση ή μετακινούν ευαίσθητα δεδομένα χρησιμοποιούν κρυπτογράφηση. Στην περίπτωση μας όμως, που χρησιμοποιούμε διακομιστή μεσολάβησης με πιστοποίηση, τα διαπιστευτήρια του χρήστη διακινούνται μεταξύ του χρήστη πελάτη και του διακομιστή

μεσολάβησης σε plain text. Αυτό δημιούργησε την ανάγκη η επικοινωνία μεταξύ χρήστη και διακομιστή μεσολάβησης να είναι κρυπτογραφημένη. Μέχρι πρότινος αυτό ήταν αδύνατο, καθώς δεν υπήρχε υποστήριξη από τους browsers. Τον τελευταίο χρόνο browser όπως ο Mozzila, Chrome, Chromium άρχισαν να υποστηρίζουν Secure TLS Web Proxying. Επειδή η απευθείας κρυπτογράφηση μεταξύ του πελάτη και του διακομιστή μεσολάβησης έχει δημιουργηθεί τον τελευταίο χρόνο, δεν πρέπει να συγχέεται με την ήδη υπάρχουσα μέθοδο η οποία χρησιμοποιεί τη μέθοδο HTTP CONNECT, ώστε να επικοινωνήσει ο πελάτης με έναν διακομιστή τελικού προορισμού με κρυπτογράφηση μέσω του διακομιστή μεσολάβησης.

Για να λειτουργήσει σωστά αυτό το σενάριο, θα πρέπει ο κάθε διακομιστής μεσολάβησης να διαθέτει και ένα έγκυρο πιστοποιητικό. Για λειτουργία load balancing θα πρέπει οι servers που είναι στο array να έχουν κοινό πιστοποιητικό και το πιστοποιητικό να έχει την ονομασία του load balancer.

3 SQUID PROXY SERVER

3.1 ΤΙ ΕΙΝΑΙ Ο ΔΙΑΚΟΜΙΣΤΗΣ ΜΕΣΟΛΑΒΗΣΗΣ (PROXY SERVER)

Ο Διακομιστής μεσολάβησης (αγγλ. proxy server) είναι διακομιστής που έχει στόχο να βελτιώσει την ταχύτητα πλοήγησης στο Διαδίκτυο και παράλληλα να μειώσει την κίνηση του δικτύου προς το Διαδίκτυο. Τοποθετείται ενδιάμεσα των χρηστών και του Διαδικτύου. Λαμβάνει τα αιτήματα ιστοσελίδων από έναν χρήστη, προσκομίζει τη σελίδα από το Διαδίκτυο, και έπειτα την δίνει στον υπολογιστή που την ζήτησε. Ο διακομιστής μεσολάβησης μπορεί να είναι και μέρος ενός τείχους προστασίας και μπορεί να αποτρέπει τους χάκερς από το να χρησιμοποιήσουν το Διαδίκτυο με σκοπό να αποκτήσουν πρόσβαση σε υπολογιστές ενός ιδιωτικού δικτύου.

Χαρακτηριστικά ενός διακομιστή μεσολάβησης:

- Γρήγορη Μνήμη (Cache)
- Ανώνυμη Πλοήγηση Χρηστών (Anonymous User Navigation)
- Έλεγχος διακινούμενου Περιεχομένου (Content Management)
- Φιλτράρισμα Κίνησης Δικτύου (Filtering)

<u>Δυνατότητες</u>

Ένας διακομιστής μεσολάβησης μπορεί:

- Να προσφέρει πιο γρήγορη πρόσβαση σε ιστοσελίδες ή αρχεία που ήδη έχουν ζητηθεί από άλλους χρήστες.
- Να τροποποιεί το ερώτημα.
- Αντί να φαίνεται ότι τη σελίδα ζήτησε ο χρήστης Χ που δεν έχει πρόσβαση στη σελίδα, να φαίνεται ότι τη ζήτησε ο Ψ ο οποίος έχει δυνατότητα πρόσβασης
- Να τροποποιεί την απάντηση.

- Να αφαιρεί ή να προσθέτει κείμενο.
- Να ελέγχει τα αρχεία ή τις παραπομπές σε αρχεία για ιούς.
- Να μετατρέπει τη μορφή των αρχείων.
- Να περιορίζει την ταχύτητα ή την ποσότητα δεδομένων για κάθε υπολογιστή.
- Να καταγράφει τη χρήση του διαδικτύου για κάθε υπολογιστή.

HTTPS

Το πρωτόκολλο αυτό θεωρητικά δεν επιτρέπει σε διακομιστές μεσολάβησης να δουν το περιεχόμενο του ερωτήματος ή της απάντησης. Όμως, η αλυσίδα TLS/SSL στηρίζεται στην ύπαρξη ενός έμπιστου ριζικού πιστοποιητικού ψηφιακά υπογεγραμμένου από μία αρχή πιστοποίησης (certificate authority). Στο περιβάλλον μιας επιχείρησης ή ενός ιδρύματος μπορεί να έχει καταχωρηθεί στους υπολογιστές ένα πιστοποιητικό της επιχείρησης το οποίο να αναγνωρίζεται ως ριζικό και ο διακομιστής μεσολάβησης να έχει πρόσβαση στο ιδιωτικό του κλειδί. Σε αυτήν την περίπτωση ο διακομιστής μεσολάβησης δρώντας ως επιτιθέμενος μπορεί να αποκτήσει πρόσβαση στα δεδομένα.

Διαφανής διακομιστής μεσολάβησης

Ως Διαφανής διακομιστής μεσολάβησης (αγγλ. transparent proxy) νοείται ο διακομιστής μεσολάβησης που υπάρχει, χωρίς να απαιτείται οποιαδήποτε ειδική ρύθμιση παραμέτρων υπολογιστή-πελάτη. Οι πελάτες δεν χρειάζεται να γνωρίζουν την ύπαρξη του διακομιστή μεσολάβησης. Το RFC 2616 (Hypertext Transfer Protocol—HTTP/1.1) αναφέρει:

"Ο Διαφανής διακομιστής μεσολάβησης πρέπει να είναι ένας διακομιστής μεσολάβησης που δεν τροποποιεί την αίτηση ή απάντηση πέρα από ό,τι απαιτείται για έλεγχο ταυτότητας και αναγνώρισης".

"Ένας μη διαφανής διακομιστής μεσολάβησης είναι ένας διακομιστής μεσολάβησης που τροποποιεί την αίτηση ή την απάντηση, προκειμένου να παρέχει κάποια υπηρεσία προστιθέμενης αξίας στον χρήστη, όπως η μετατροπή τύπων αρχείων ή ανωνυμία."

Ένας διαφανής διακομιστής μεσολάβησης συνήθως βρίσκεται μεταξύ του πελάτη και του Διαδικτύου, με τον διακομιστή μεσολάβησης να εκτελεί κάποιες από τις λειτουργίες ενός δρομολογητή. Όταν δηλαδή το πρόγραμμα πελάτης ζητήσει μια ιστοσελίδα το δίκτυο (τοπικό ή το δίκτυο του ISP) αντί να αποστείλει το αίτημα κατευθείαν μέσω του δρομολογητή, το αποστέλλει σε έναν διακομιστή μεσολάβησης ο οποίος δεν απαιτεί από

το πρόγραμμα πελάτη κάποια ειδική ρύθμιση. Το πρόγραμμα πελάτης δεν μπορεί να γνωρίζει πάντα αυτή τη διαδικασία. Οι δυνατότητες ενός διαφανή διακομιστή μεσολάβησης είναι ακριβώς οι ίδιες όπως και ενός απλού διακομιστή μεσολάβησης.

Στο δίκτυο μιας επιχείρησης, η χρήση ενός μη διαφανούς διακομιστή μεσολάβησης έρχεται αντιμέτωπο με δύο προβλήματα:

- Θα πρέπει να γίνει ειδική ρύθμιση σε όλους τους υπολογιστές με αποτέλεσμα να αυξάνει το κόστος εγκατάστασης και συντήρησης.
- Η ρύθμιση αυτή μπορεί να παρακαμφθεί από τους χρήστες εσκεμμένα ή από λάθος με αποτέλεσμα π.χ. να μην φιλτραριστούν ιοί.

Τα προβλήματα αυτά λύνονται με τη χρησιμοποίηση ενός μη διαφανούς διακομιστή μεσολάβησης.

Αντίστροφος διακομιστής μεσολάβησης

Reverse proxy

Ως Αντίστροφος διακομιστής μεσολάβησης (αγγλ. reverse proxy) νοείται ο διακομιστής μεσολάβησης που υπάρχει πριν από κάποιον εξυπηρετητή ιστοσελίδων. Ο διακομιστής αυτός παραλαμβάνει όλα τα αιτήματα για τον διακομιστή των ιστοσελίδων και απαντά χρησιμοποιώντας προηγούμενα όμοια αιτήματα χωρίς να ξαναζητήσει τις σελίδες από τον εξυπηρετητή ιστοσελίδων. Επίσης μπορεί να χρησιμοποιηθεί για την κρυπτογράφηση των δεδομένων ώστε το πρόγραμμα κρυπτογράφησης να μην επιβαρύνει τον κύριο εξυπηρετητή. Δηλαδή ο αντίστροφος διακομιστής μεσολάβησης να επικοινωνεί με τον κύριο εξυπηρετητή με το απλό πρωτόκολλο http και με τους πελάτες να επικοινωνεί με το πιο περίπλοκο https.

Ανοιχτοί διακομιστές μεσολάβησης

Η πρόσβαση σε διακομιστή μεσολάβησης μπορεί να δοθεί και σε χρήστες εκτός τοπικού δικτύου. Σε μια τέτοια περίπτωση οι χρήστες μπορούν αν εκμεταλλευθούν την δυνατότητα μετεγγραφής των κεφαλίδων των αιτημάτων, ώστε να φαίνεται ότι τα αιτήματα γίνονται από τον υπολογιστή που βρίσκεται εγκατεστημένος ο διακομιστής και επομένως από την περιοχή στην οποία βρίσκεται. Με αυτόν τον τρόπο μπορεί να παρακάμπτεται το τείχος προστασίας του δικτύου ή να εμφανίζεται ο υπολογιστής ότι βρίσκεται σε άλλη τοποθεσία. Η παράκαμψη του τείχους προστασίας όμως, καθώς και η χρήση του διακομιστή μεσολάβησης μπορεί να δημιουργήσει άλλα σοβαρά προβλήματα όπως η έκθεση σε ιούς ή η καταγραφή των δεδομένων που μεταφέρονται.

Διακομιστής μεσολάβησης DNS

Ένας Διακομιστής μεσολάβησης DNS ουσιαστικά μεταφέρει τις κλήσεις για DNS σε διακομιστές Υπηρεσιών DNS, επιστρέφοντας τις απαντήσεις και καταγράφοντας αυτές σε προσωρινή μνήμη. Όλοι οι σύγχρονοι οικιακοί δρομολογητές παρέχουν έναν υποτυπώδη Διακομιστή μεσολάβησης DNS.

3.2 ΤΙ ΕΙΝΑΙ Ο SQUID PROXY SERVER

Ο Squid proxy server είναι ένας caching διακομιστής μεσολάβησης ανοιχτού κώδικα ο οποίος υπάρχει και αναπτύσσεται σταθερά από 5 προγραμματιστές, χρηματοδοτούμενους από δωρεές και μερικές εμπορικές συμφωνίες. Χρησιμοποιείται ευρέως από πανεπιστήμια, επιχειρήσεις, δίκτυα διανομής περιεχομένου μέχρι και μεγάλους ISP.

Το βασικό κομμάτι του squid (core) αποτελείται από έναν caching proxy για τα πρωτόκολλα HTTP,HTTPS και FTP. Λόγω της μεγάλης κοινότητας που τον υποστηρίζει έχουν δημιουργηθεί πολλά πρόσθετα πακέτα εκ των οποίων τα πιο σημαντικά είναι :

- Ταυτοποίηση χρήστη
- Φιλτράρισμα περιεχομένου
- Deep Packet Inspection
- Κρυπτογράφηση χρήστη -> διακομιστή μεσολάβησης
- URL filtering
- Reverse Proxying

Τα κυριότερα αυτά πρόσθετα πακέτα μπορούν να γίνουν compile απευθείας από τον πηγαίο κώδικα του Squid κάνοντας τον πολύ πρακτικό για mass deployments και λειτουργία out of the box.

Ένα σημαντικό πλεονέκτημα του Squid είναι ότι μπορεί να ρυθμιστεί η λειτουργία του σε ιεραρχίες (ICP πρωτόκολλο).

3.3 ΒΑΣΙΚΕΣ ΛΕΙΤΟΥΡΓΙΕΣ ΤΟΥ SQUID

Όλη η λειτουργία του squid ρυθμίζεται από ένα αρχείο το οποίο στην περίπτωση του FreeBSD 10.3 βρίσκεται στη διαδρομή /usr/local/etc/squid/squid.conf.

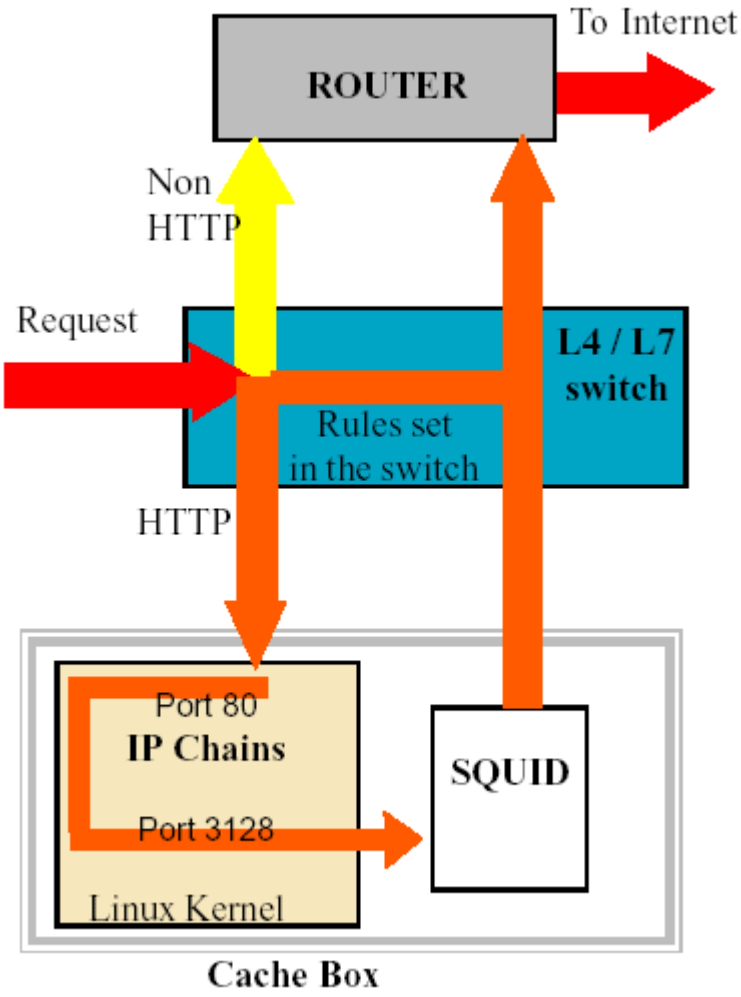
Σε αυτό το αρχείο ρυθμίζεται η λειτουργία (mode στο οποίο θα λειτουργήσει) καθώς επίσης και όλες οι πρόσθετες λειτουργίες και ρυθμίσεις περιβάλλοντος, οι οποίες θα χρησιμοποιηθούν κατά την εκτέλεσή του. Κάθε τέτοια δήλωση σε αυτό το αρχείο ονομάζεται directive και έτσι θα αναφερόμαστε πλέον σε αυτές. Μετά την εγκατάσταση ο squid server τρέχει στην πόρτα 3128 και σε όλες τις λειτουργίες θα κρατήσουμε αυτή την πόρτα αν και είναι πολύ εύκολο να την αλλάξουμε χωρίς κανένα πρόβλημα σε όλη την υπόλοιπη λειτουργία του.

3.3.1 Λειτουργίες (modes) του squid proxy server

3.3.1.1 Transparent mode (διαφανής λειτουργία)

Στη διαφανή λειτουργία ο squid, απλώς παρεμβάλλεται σε κάθε http αίτημα που του έρχεται στην πόρτα 80, αναλαμβάνει αυτός να διακπαιρεώσει το αίτημα και απαντάει σε αυτόν που έκανε το αίτημα με το περιεχόμενο που ζήτησε. Αυτός που έκανε το request σε αυτή την περίπτωση δεν γνωρίζει ότι το περιεχόμενο ήρθε από έναν διακομιστή μεσολάβησης. Αυτό αποτελεί και το λόγο που η διαφανής λειτουργία δεν μπορεί να λειτουργήσει στο πρωτόκολλο https, καθώς ο χρήστης λόγω του tls ξέρει ότι δεν του απαντάει ο διακομιστής προορισμού, αλλά κάποιος άλλος με αποτέλεσμα ο browser να βγάζει μήνυμα λάθους ασφάλειας.

Για να επιτευχθεί αυτή η λειτουργία ο κεντρικός δρομολογητής πρέπει να κάνει ανακατεύθυνση κάθε αίτημα που του έρχεται στην πόρτα 80 προς τη διεύθυνση του που τρέχει ο squid (πορτα 3128).



Εικόνα 5: Squid διαφανής λειτουργία

Για να λειτουργήσει το προηγούμενο σενάριο το directive στο αρχείο ρύθμισης του squid είναι :

http port 3128 intercept

το οποίο ενημερώνει τον squid να ακούει στην πόρτα 3128 μέσω http πρωτοκόλλου και να κάνει παρεμβολή.

Για να μπορέσουμε να κάνουμε προώθηση την πόρτα 80 στην πόρτα 3128 του squid θα πρέπει επιπροσθέτως, να δημιουργήσουμε και τον ακόλουθο κανόνα στο ipfw του FreeBSD με την ακόλουθη εντολή :

-A PREROUTING -i eth0.5 -p tcp -m tcp --dport 80 -j REDIRECT --to-ports 3128

3.3.1.2 Λειτουργία δηλωμένου διακομιστή μεσολάβησης προώθησης (Forward Proxy) μέσω HTTP

Σε αυτή τη λειτουργία ο πελάτης (browser) είναι ρυθμισμένος να χρησιμοποιεί έναν διακομιστή μεσολάβησης (είτε χειροκίνητα είτε με αυτόματο τρόπο, όπως θα δούμε παρακάτω) και επομένως ο browser είναι πλήρως ενημερωμένος.

Τώρα ο χρήστης-πελάτης είναι ενημερωμένος για την ύπαρξη διακομιστή μεσολάβησης και επομένως μπορεί να χρησιμοποιήσει πλήρως τις λειτουργίες του διακομιστή μεσολάβησης. Βασικές λειτουργίες για αυτό το σενάριο χρήσης είναι η αυθεντικοποίηση χρήστη καθώς επίσης και η διαμεσολάβηση για το πρωτόκολλο https (λειτουργίες που όπως είδαμε δεν μπορούν να υπάρξουν σε διακομιστή μεσολάβησης όταν είναι σε λειτουργία διαφανής χρήσης).

Βασικό πλεονέκτημα αυτής της χρήσης είναι ότι εκτός από τη χρήση του caching του διακομιστή μεσολάβησης μπορούμε πλέον να καθορίσουμε και ποιοι μπορούν να το χρησιμοποιήσουν. Στη λειτουργία αυτή ο squid server αυθεντικοποιεί τους χρήστες μέσω του διακομιστή LDAP του Ελληνικού Σχολικού Δικτύου. Αυτό επιτυγχάνεται μέσω του προγράμματος basic_ldap_auth του οποίου ο κώδικας επισυνάπτεται στο παράρτημα 7.2.10.

Ένα επιπλέον προτέρημα σε αυτή τη λειτουργία είναι ότι πλέον ο διακομιστής μεσολάβησης μπορεί να χρησιμοποιηθεί και για φιλτράρισμα URL για το πρωτόκολλο https. Όπως είδαμε σε προηγούμενο κεφάλαιο το πρωτόκολλο https όταν γνωρίζει ότι θα χρησιμοποιήσει διακομιστή μεσολάβησης τότε στέλνει στην κεφαλίδα connect το όνομα τομέα του τελικού προορισμού, οπότε μπορεί πλέον ο διακομιστής μεσολάβησης αφού έχει το όνομα τομέα να εφαρμόσει πάνω του τις πολιτικές οι οποίες έχουν αποφασιστεί για τη συγκεκριμένη κατηγορία χρηστών του Πανελλήνιου Σχολικού Δικτύου. Ένα μειονέκτημα αυτής της μεθόδου είναι ότι το φιλτράρισμα μπορεί να εφαρμοστεί μόνο σε επίπεδο τομέα και όχι σε επίπεδο url.

Για την σωστή λειτουργία αυτού του σεναρίου, οι κεντρικοί δρομολογητές από τους οποίους περνάει η κίνηση για το internet θα πρέπει να κόψουν όλη την κίνηση προς το internet και να επιτρέψουν μόνο την κίνηση προς τις διευθύνσεις των squid cache boxes.

Σε αυτό το σενάριο ο χρήστης-πελάτης πρέπει να γνωρίζει ποιος είναι ο διακομιστής μεσολάβησης. Στο Πανελλήνιο Σχολικό Δίκτυο υπάρχουν 16 διακομιστές μεσολάβησης squid, στους οποίους διανέμεται η κίνηση σύμφωνα με το τελευταίο bit προορισμού.

Από τη στιγμή όμως που ο διακομιστής μεσολάβησης πρέπει να δηλώνεται ρητά τότε αυτή η πολιτική δεν μπορεί να λειτουργήσει στο δικό μας σενάριο. Για να γίνει λειτουργικό αυτό το σενάριο θα εξεταστούν πιο κάτω δύο τρόποι κατανομής φορτίου συνοδευόμενοι με τα πλεονεκτήματα και τα μειονεκτήματα του καθενός.

Για να λειτουργήσει ο squid με αυτό τον τρόπο θα πρέπει στο αρχείο ρυθμίσεων του να δηλωθεί τα ακόλουθα directives :

```
auth_param basic program /usr/local/libexec/squid/basic_ldap_auth -h lp.sch.gr -b /
"dc=sch,dc=gr" -f "uid=%s" -D uid=qmaileye,dc=sch,dc=gr -w qmaileye
auth_param basic children 500
auth_param basic realm Web-Proxy
auth_param basic credentialsttl 60 minute
```

Τα παραπάνω ενημερώνουν τον squid ότι θα χρησιμοποιήσει το πρόγραμμα basic_ldap_auth και θα επικοινωνήσει με τον διακομιστή LDAP (-h) lp.sch.gr. Επιπλέον θα ψάξει στη βάση του διακομιστή LDAP για χρήστες που ανήκουν στον τομέα sch.gr, θα χρησιμοποιήσει το όνομα χρήστη qmaileye.sch.gr με συνθηματικό «qmaileye» για να κάνει bind στον διακομιστή LDAP και θα περάσει σαν παράμετρο uid το όνομα που θα δώσει ο χρήστης όταν του ζητηθεί κατά την πρώτη επικοινωνία με τον διακομιστή μεσολάβησης. Αν το όνομα χρήστη και το συνθηματικό είναι σωστά, τότε επιτρέπει την πρόσβαση στο χρήστη και αποθηκεύει τοπικά το base64 του ονόματος χρήστη και του κωδικού του ώστε να μην κάνει κάθε φορά ερώτηση στον διακομιστή LDAP για τα διαπιστευτήρια του χρήστη.

Στην επόμενη γραμμή ενημερώνουμε τον squid ότι θα μπορεί να χρησιμοποιήσει μέχρι 500 ταυτόχρονες διεργασίες (threads) του προγράμματος basic_ldap_auth ώστε να κάνει ταυτόχρονες αυθεντικοποιήσεις χρηστών και να μην υπάρχει καθυστέρηση αν ζητηθούν πολλές ταυτόχρονες αυθεντικοποιήσεις χρηστών.

Το επόμενο βασικό συστατικό για τη σωστή λειτουργία είναι το realm. Το realm πρέπει ανάλογα με την τεχνική load balancing που θα χρησιμοποιηθεί, να είναι σε όλους τους squid το ίδιο (load balancing με ενδιάμεσο load balancer) ή αν όχι να χρησιμοποιηθεί load balancing με αρχείο αυτορρύθμισης των χρηστών-πελατών. Επιπλέον, θέτουμε και ένα περιθώριο 1 ώρας για την οποία τα διαπιστευτήρια αποθηκεύονται τοπικά στον squid server ώστε να μην προβαίνουμε σε συνεχόμενες ερωτήσεις στον ldap server. Αυτό έχει

σαν αποτέλεσμα και λιγότερο φόρτο εργασίας στον squid server μέσω του προγράμματος basic_ldap_auth και λιγότερο φόρτο στον διακομιστή LDAP.

Το συγκεκριμένο directive κάνει απλώς την ταυτοποίηση και την αυθεντικοποίηση του χρήστη. Κάθε χρήστης που περνάει αυθεντικοποίηση, αποθηκεύεται τοπικά και μπαίνει στο group χρηστών proxy_auth, δηλαδή σε ένα group χρηστών για τους οποίους το πρόγραμμα έχει κάνει επιτυχή αυθεντικοποίηση.

```
acl ldap-auth proxy_auth REQUIRED
……
acl deny all
```

Το άνω directive είναι που αυτό ουσιαστικά απαγορεύει την πρόσβαση σε μη πιστοποιημένους χρήστες του ldap server. Το πρώτο directive πρέπει να τοποθετηθεί μετά από όλα τα directive που αφορούν τις διαχειριστικές συσκευές δικτύου που επικοινωνούν με τον squid server ώστε σε αυτές να μην ζητείται πιστοποίηση. Μετά το πρώτο directive πρέπει να τοποθετηθούν όλα τα directives τα οποία καθορίζουν τους τρόπου πρόσβασης στο internet καθώς και τις επιτρεπόμενες πόρτες και πρωτόκολλα. Τέλος, ενημερώνουμε τον squid ό,τι ότι άλλο δεν δηλώνεται στα directive απαγορεύεται.

3.3.1.3 Λειτουργία δηλωμένου διακομιστή μεσολάβησης (Forward Proxy) μέσω HTTPS (SECURE TLS WEB PROXY)

Στη λειτουργία forward proxy ο squid χρησιμοποιεί κρυπτογράφηση μεταξύ του client και του διακομιστή μεσολάβησης. Αυτό είναι απαραίτητο καθώς το σχήμα αυθεντικοποίησης basic που υποστηρίζει ο διακομιστής LDAP του Πανελλήνιου Σχολικού Δικτύου στέλνει σε κάθε request τα δεδομένα αυθεντικοποίησης (όνομα χρήστη και κωδικό) με κωδικοποίηση base64, η οποία είναι εύκολα αναστρέψιμη και είναι σαν να τα στέλνει σε plain text. Για την προστασία των δεδομένων του πελάτη-χρήστη πρέπει να κρυπτογραφήσουμε την επικοινωνία μεταξύ των. Παρόλο που ο squid και το πρωτόκολλο υποστηρίζουν αυτό το μοντέλο αρκετά χρόνια, οι browsers άρχισαν να το υποστηρίζουν το τελευταίο χρόνο, ενώ ακόμα δεν υποστηρίζεται ακόμα σε όλους τους browsers. Ο πρώτος browser που το υποστήριξε ήταν ο Mozilla Firefox (από την έκδοση 48), και μετά ο Chrome. Η Microsoft έχει δεσμευτεί ότι θα το υποστηρίξει σύντομα μέσω του Internet Explorer, γεγονός που σημαίνει ότι θα υποστηρίζεται σε όλο το λειτουργικό και για όλες τις εφαρμογές που χρησιμοποιούν το WinHttp Api.

Για να πραγματοποιηθεί αυτό το σενάριο ο κάθε squid server θα πρέπει να έχει ένα TLS πιστοποιητικό το οποίο πιστοποιεί την ταυτότητα του. Όμως αυτό δημιουργεί μερικά προβλήματα στο load balancing όπως θα δούμε πιο κάτω καθώς ο κάθε χρήστης-πελάτης μπορεί να ξεκινήσει μια συνεδρία με έναν διακομιστή μεσολάβησης και μπορεί να του απαντάει μόνο αυτός που είναι ρητά δηλωμένος στις ρυθμίσεις του.

Σε αυτό το σενάριο λειτουργίας οποιαδήποτε επικοινωνία χρήστη-πελάτη με τον διακομιστή μεσολάβησης είναι κρυπτογραφημένη. Αυτό ισχύει ακόμα και αν ο χρήστης-πελάτης δεν ζητήσει μια τελική υπηρεσία που είναι κρυπτογραφημένη. Κάθε επικοινωνία μέσω του διακομιστή μεσολάβησης κρυπτογραφείται. Από εκεί και πέρα ο διακομιστής μεσολάβησης δημιουργεί μια σύνδεση με τον τελικό προορισμό είτε είναι κρυπτογραφημένη είτε όχι.

Ανάλογα με την περίπτωση έχουμε τα εξής :

- Ο πελάτης ζητάει μια http υπηρεσία :
    - Σε αυτήν την περίπτωση η επικοινωνία μεταξύ πελάτη proxy είναι κρυπτογραφημένη και στέλνεται ένα αίτημα http προς τον διακομιστή μεσολάβησης όπως είδαμε και ο διακομιστής μεσολάβησης φέρνει τα περιεχόμενα που ζητήθηκαν (τα αποθηκεύει τοπικά) και τα στέλνει μέσω του κρυπτογραφημένου tunnel που δημιουργήθηκε προς τον πελάτη.

- Ο πελάτης ζητάει μια https υπηρεσία :
    - Σε αυτή την περίπτωση δημιουργείται πάλι ένα TLS tunnel μεταξύ πελάτη και διακομιστή μεσολάβησης, ο πελάτης αυτή τη φορά περνάει το http connect αίτημα μέσω του κρυπτογραφημένου tunnel. Ο διακομιστής μεσολάβησης καταλαβαίνει ότι είναι ένα http connect αίτημα και αναλαμβάνει την επικοινωνία και ανταλλαγή αρχικού TLS handshake μεταξύ του πελάτη και του τελικού προορισμού. Όταν ολοκληρωθεί το handshake, ο διακομιστής μεσολάβησης απλώς φέρνει το κρυπτογραφημένο περιεχόμενο από τον τελικό προορισμό, το ξανακρυπτογραφεί σύμφωνα με το TLS tunnel μεταξύ πελάτη – διακομιστή μεσολάβησης και το στέλνει στον πελάτη. Αυτή τη φορά μεταξύ πελάτη και διακομιστή μεσολάβησης έχουμε ένα TLS tunnel μέσα σε ένα άλλο TLS tunnel.

Η λειτουργία αυτή αποτρέπει στον οποιοδήποτε ενδιάμεσο να δει τα διαπιστευτήρια του πελάτη (αφού είδαμε ότι σε basic ldap authentication αυτά μεταδίδονται σε plain text) και επιπλέον δεν μπορεί να δει ούτε τι ιστοτόπους επισκέπτεται ο πελάτης, αφού η κεφαλίδα που στελνόταν στον διακομιστή μεσολάβησης πριν περιείχε και τον τελικό προορισμό είναι πλέον και αυτή κρυπτογραφημένη. Σε αυτή την περίπτωση, το μόνο που μπορεί να δει ένας ενδιάμεσος είναι μόνο το αρχικό TLS handshake με τον διακομιστή μεσολάβησης που περιέχει μόνο το όνομα του διακομιστή μεσολάβησης.

Από τη μεριά του διακομιστή μεσολάβησης προς το internet, ένας ενδιάμεσος μπορεί μόνο να δει το τι ιστοτόπους ζητάει ο διακομιστής μεσολάβησης αλλά χωρίς να γνωρίζει για ποιον πελάτη προορίζονται. Επιπλέον, σε απλές υπηρεσίες http μπορεί να δει το περιεχόμενο των πακέτων που λαμβάνει ο διακομιστής μεσολάβησης όμως αυτό δεν μπορεί να αποφευχθεί καθώς σε οποιαδήποτε τεχνολογία αυτό εξαρτάται καθαρά από τον διακομιστή τελικού προορισμού. Το μόνο που μπορούμε να κάνουμε μέσω του διακομιστή μεσολάβησης είναι να ανακατευθύνουμε την υπηρεσία http σε https σε περίπτωση που ο τελικός διακομιστής το υποστηρίζει, ενώ ο πελάτης μας έχει ζητήσει την απλή υπηρεσία http.

Ο Secure Web Proxy μπορεί να χρησιμοποιηθεί από πιστοποιημένους πλέον χρήστες και όταν αυτοί είναι εκτός του σχολικού δικτύου, αν αυτό κριθεί αναγκαίο από τους διαχειριστές του δικτύου με απόλυτη ασφάλεια, αφού και τα διαπιστευτήρια είναι

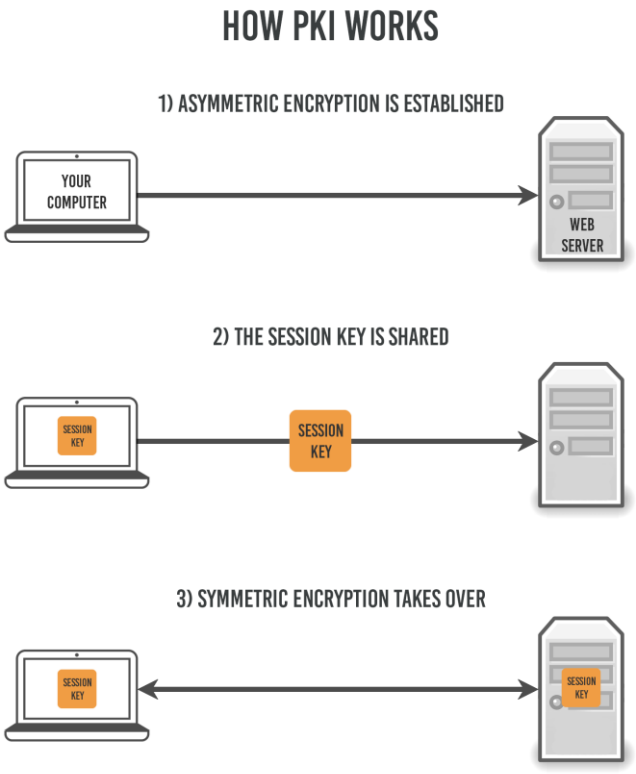
πλέον κρυπτογραφημένα και η σύνδεση με το διακομιστή μεσολάβησης μπορεί να γίνει μόνο από πιστοποιημένους χρήστες.

3.3.1.4 Λειτουργία δηλωμένου διακομιστή μεσολάβησης (Forward Proxy) με deep packet inspection (man in the middle)

Σε αυτή τη λειτουργία, ο διακομιστής μεσολάβησης, όσον αφορά το πρωτόκολλο http, κάνει ακριβώς ότι και πριν. Τώρα όμως στο πρωτόκολλο https αντί να προωθεί τα πακέτα του χρήστη πελάτη στον τελικό προορισμό, αποκρυπτογραφεί τα πακέτα, βλέπει το περιεχόμενο και δρα αυτός σαν να ήταν ο χρήστης-πελάτης και αναλαμβάνει την επικοινωνία με τον τελικό προορισμό. Τέλος, φέρνει τα πακέτα από τον τελικό προορισμό (με δυνατότητα να κάνει caching σε αυτά ώστε να είναι διαθέσιμα και για άλλες αιτήσεις) και τα στέλνει στον χρήστη-πελάτη που τα ζήτησε.

Για να επιτευχθεί αυτή η λειτουργία θα πρέπει ο διακομιστής μεσολάβησης να «υποδύεται» τον τελικό προορισμό ώστε να μπορέσει να παρεμβάλλεται μεταξύ των επικοινωνιών του χρήστη-πελάτη και τελικού προορισμού. Για την κατανόηση αυτού θα αναλύσουμε την λειτουργία του https (http over TLS) πρωτοκόλλου.

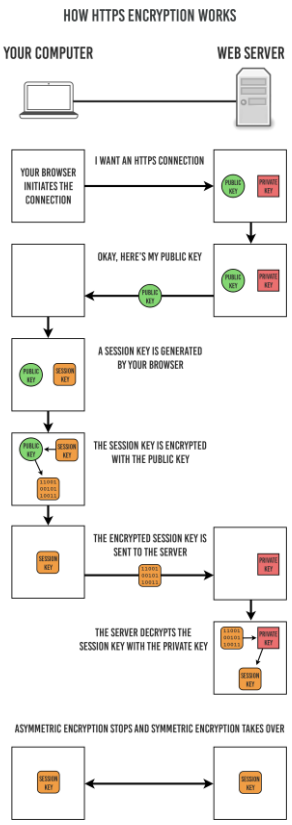
Το TLS (το οποίο χρησιμοποιείται πλέον στο http αντί του SSL) βασίζεται στην κρυπτογραφία ασύμμετρων κλειδιών και συγκεκριμένα στο Public Key Infrastructure. Αυτή η δομή βασίζεται στην χρησιμοποίηση ενός δημοσίου κλειδιού, με το οποίο η μια άκρη κρυπτογραφεί τα περιεχόμενα ενός πακέτου και το πακέτο μπορεί να αποκρυπτογραφηθεί μόνο από ιδιωτικό κλειδί του παραλήπτη. Αφού γίνει η πρώτη επικοινωνία μετά επιλέγεται ένα session key που αφορά μόνο τη συγκεκριμένη συνεδρία και σύμφωνα με αυτό κρυπτογραφούνται τα πακέτα.



Εικόνα 6: Πως λειτουργεί το PKI

Στη δικιά μας περίπτωση του https η ακολουθία έχει ως εξής :

1. Ο χρήστης πελάτης ζητάει τον ιστότοπο https://www.sch.gr
2. Ο διακομιστής ιστού στέλνει το δημόσιο κλειδί του με ένα πιστοποιητικό
3. Ο χρήστης πελάτης ελέγχει αν αυτό το πιστοποιητικό έχει εκδοθεί από μια από τις αρχές έκδοσης πιστοποιητικών, την οποία θεωρεί έμπιστη (σε αυτό το βήμα, αν το πιστοποιητικό δεν έχει εκδοθεί από μια αρχή που θεωρεί ο πελάτης έμπιστη, η σύνδεση διακόπτεται με μήνυμα λάθους πιστοποιητικού)
4. Ο χρήστης πελάτης δημιουργεί ένα τρίτο κλειδί που ονομάζεται session key
5. Το session key κρυπτογραφείται με το δημόσιο κλειδί και στέλνεται στο διακομιστή
6. Ο διακομιστής τώρα αποκρυπτογραφεί το μήνυμα που έστειλε ο πελάτης με το ιδιωτικό του κλειδί και λαμβάνει το session key
7. Το PKI τώρα λαμβάνει τέλος και η ασύμμετρη κρυπτογράφηση αντικαθίσταται με συμμετρική με βάση το session key.



Εικόνα 7: HTTP over TLS encryption

Για να είναι αξιόπιστη μια σύνδεση με έναν https διακομιστή ιστού, πρέπει αυτός ο διακομιστής να έχει προμηθευτεί ένα πιστοποιητικό από μια έγκυρη αρχή πιστοποίησης. Έτσι, αποτρέπεται το γεγονός κάποιος να υποδύεται ότι είναι κάποιος άλλος. Στη περίπτωση μας θα πρέπει ο διακομιστής μεσολάβησης να υποδύεται πάντα ότι είναι κάποιος άλλος και συγκεκριμένα κάθε φορά θα πρέπει να υποδύεται τον διακομιστή του ιστοτόπου τον οποίο ζητάει ο πελάτης. Φυσικά, δεν μπορούμε να αποκτήσουμε από μια αξιόπιστη αρχή ένα πιστοποιητικό που θα μας πιστοποιεί για όλους τους διακομιστές ιστού που μπορεί να ζητήσει ο πελάτης και για αυτό θα δημιουργήσουμε μια δικιά μας αρχή πιστοποίησης.

Στο FreeBSD μέσω του πακέτου openSSL είναι εύκολο να δημιουργήσουμε μια αρχή πιστοποίησης με την εξής εντολή :

```
openssl req -new -newkey rsa:2048 -sha256 -days 365 -nodes -x509 -
extensions v3_ca -keyout proxyCA.pem  -out proxyCA.pem
```

Η εντολή αυτή δημιουργεί ένα ROOT CA πιστοποιητικό (πιστοποιητικό αρχής πιστοποίησης).

Το πιστοποιητικό αυτό θα χρησιμοποιηθεί μετά από τον squid για να δημιουργεί πιστοποιητικά για τον κάθε διακομιστή ιστού που ζητάει ο χρήστης-πελάτης. Το πρόβλημα όμως είναι ότι η δικιά μας ιδιωτική αρχή πιστοποίησης, δεν θεωρείται από κανέναν αξιόπιστη οπότε δεν μπορεί να χρησιμοποιηθεί ακόμα.

Για να χρησιμοποιηθεί θα πρέπει να εξάγουμε ένα πιστοποιητικό ROOT CA με το δημόσιο κλειδί μας και αυτή να εγκατασταθεί στον κάθε χρήστη πελάτη, ώστε να μας θεωρεί αξιόπιστη αρχή. Ο τρόπος δημιουργίας του πιστοποιητικού έχει ως εξής :

```
openssl x509 -in proxyCA.pem -outform DER -out proxyCA.der
```

Σε περιβάλλοντα με Windows, η διαδικασία εισαγωγής του πιστοποιητικού στους πελάτες μπορεί να αυτοματοποιηθεί και απλώς να γίνει εισαγωγή του πιστοποιητικού στον Domain Controller και αυτό να διανεμηθεί στους πελάτες αυτόματα μέσω ενός group policy.

Αφού τώρα οι πελάτες μας θεωρούν αξιόπιστη αρχή δημιουργίας πιστοποιητικών μπορούμε να παρεμβληθούμε στο βήμα 2 της προηγούμενης αλυσίδας γεγονότων.

Για την ακρίβεια κάθε φορά που ένας χρήστης πελάτης μας ζητάει μια υπηρεσία https (πχ https://www.sch.gr) τότε η αλυσίδα γεγονότων έχει ως εξής :

Στη περίπτωση μας του https η ακολουθία έχει ως εξής :

1. Ο χρήστης-πελάτης ζητάει τον ιστότοπο https://www.sch.gr μέσω ενός http connect αιτήματος στον διακομιστή μεσολάβησης.
2. Ο διακομιστής μεσολάβησης «βλέπει» (peak) για ποιον server προορίζεται το πακέτο.
3. Αφού το δει, δημιουργεί ένα πιστοποιητικό για τον διακομιστή www.sch.gr και το στέλνει πίσω στον χρήστη-πελάτη το πιστοποιητικό αυτό με το δημόσιο κλειδί του.
4. Ο χρήστης-πελάτης ελέγχει αυτό το πιστοποιητικό και αν η αρχή πιστοποίησης είναι αξιόπιστη τότε το αποδέχεται.
5. Ο χρήστης-πελάτης δημιουργεί ένα τρίτο κλειδί που ονομάζεται session key.
6. Το session key κρυπτογραφείται με το δημόσιο κλειδί και στέλνεται στο διακομιστή μεσολάβησης.

7. Ο διακομιστής μεσολάβησης τώρα αποκρυπτογραφεί το μήνυμα που έστειλε ο πελάτης με το ιδιωτικό του κλειδί και λαμβάνει το session key.
8. Το PKI λαμβάνει τέλος και η ασύμμετρη κρυπτογράφηση αντικαθίσταται με συμμετρική, σύμφωνα με το session key μεταξύ χρήστη-πελάτη και διακομιστή μεσολάβησης. Πλέον ο διακομιστής μεσολάβησης λαμβάνει όλα τα πακέτα που προορίζονται για τον τελικό προορισμό και τα αποκρυπτογραφεί.
8. Ο διακομιστής μεσολάβησης υποδυόμενος τον χρήστη ζητάει τον ιστότοπο https://www.sch.gr
9. Ο διακομιστής ιστού στέλνει το δημόσιο κλειδί του με ένα πιστοποιητικό.
10. Ο διακομιστής μεσολάβησης ελέγχει αν αυτό το πιστοποιητικό είναι από μια από τις αρχές έκδοσης πιστοποιητικών, την οποία θεωρεί έμπιστη.
11. Ο διακομιστής μεσολάβησης δημιουργεί ένα τρίτο κλειδί που ονομάζεται session key.
12. Το session key κρυπτογραφείται με το δημόσιο κλειδί και στέλνεται στο διακομιστή του sch.gr.
13. Ο διακομιστής τώρα αποκρυπτογραφεί το μήνυμα που έστειλε ο διακομιστής μεσολάβησης με το ιδιωτικό του κλειδί και λαμβάνει το session key.
14. Το PKI τώρα λαμβάνει τέλος και η ασύμμετρη κρυπτογράφηση αντικαθίσταται με συμμετρική σύμφωνα με το session key.

Όπως παρατηρούμε τώρα, ο squid εκτός από caching proxy γίνεται και TLS termination proxy και πελάτης ο ίδιος. Για κάθε https σύνδεση πρέπει να διατηρεί 2 TLS tunnels, ένα μεταξύ αυτού και του πελάτη και ένα μεταξύ αυτού και του τελικού προορισμού. Αυτό φυσικά δημιουργεί ένα μεγαλύτερο φόρτο εργασίας για τον διακομιστή μεσολάβησης, το οποίο όπως θα δούμε παρακάτω μπορεί σε μερικές περιπτώσεις να είναι απαγορευτικό, καθώς για κάθε σύνδεση https λαμβάνουν χώρα 2 κρυπτογραφήσεις/ αποκρυπτογραφήσεις, που είναι πολύ απαιτητικές σε πόρους και επεξεργαστή και μνήμης.

Για να μπει σε αυτή τη λειτουργία ο squid, πρέπει στη ρύθμιση του να εισαχθούν τα εξής directives :

```
http_port 3128 ssl-bump \
  cert=/etc/squid/ssl_cert/proxyCA.pem \
  generate-host-certificates=on dynamic_cert_mem_cache_size=4MB


acl step1 at_step SslBump1


ssl_bump peek step1
ssl_bump bump all


sslproxy_cafile /usr/local/openssl/cabundle.file
```

- Το πρώτο directive μας λέει να χρησιμοποιήσουμε τη λειτουργία ssl-bump (man in the middle) στην πόρτα 3128.
- Το δεύτερο directive μας λέει να επιτρέψουμε το ssl.
- Το τρίτο directive μας λέει πρώτα να εξάγουμε το http connect πεδίο (δηλαδή τον τελικό προορισμό) και να δούμε τον ιστότοπο για τον οποίο πρέπει να δημιουργήσουμε πιστοποιητικό.
- Όταν γίνει το peek αναλαμβάνει η μηχανή δημιουργίας πιστοποιητικών ssl-bump.

Μετά από αυτό το στάδιο για αυτή τη σύνδεση με το τέταρτο directive λέμε ότι ο squid θα παρεμβάλλεται σε όλη την ανταλλαγή πληροφοριών για αυτή τη σύνδεση.

Σε αυτό το σενάριο χρήσης πλέον έχουμε πλήρη έλεγχο του δικτύου και μπορούμε να ελέγχουμε ακόμα και επίσκεψη σε κρυπτογραφημένες υπηρεσίες, ή ακόμα και στην χρησιμοποίηση Secure Web Proxy που έχουν αρχίσει και χρησιμοποιούν σε πολλές περιπτώσεις όσοι θέλουν να αποφύγουν τον έλεγχο πρόσβασης σε συγκεκριμένες κατηγορίες ιστοσελίδων.

Επιπλέον, παρόλη την μεγαλύτερη υπολογιστική ισχύ που μπορεί να χρειαστεί για τα δυο εξτρά στάδια κρυπτογράφησης και αποκρυπτογράφησης, η τεχνική αυτή μας παρέχει μεγαλύτερη δυνατότητα caching περιεχομένου και μας γλιτώνει το αντίστοιχο bandwidth αφού πλέον μπορούμε να αποθηκεύουμε τοπικά πόρους και ιστοσελίδες οι οποίες πριν με την χρήση του https πρωτοκόλλου από τους πελάτες δεν ήταν δυνατή.

3.4 ΣΧΗΜΑΤΑ ΑΥΘΕΝΤΙΚΟΠΟΙΗΣΗΣ ΤΟΥ SQUID PROXY SERVER

Ο Squid, ακολουθεί πιστά το http πρωτόκολλο και οι μηχανισμοί αυθεντικοποίησης που προσφέρονται αυτή τη στιγμή είναι :

- Basic
- Digest
- Microsoft NTLM authentication
- Negotiate/Kerberos authentication
- Bearer

Τα σχήματα αυθεντικοποίησης τα οποία παρέχονται, έχουν να κάνουν καθαρά με το http πρωτόκολλο και αφορούν καθαρά στον μηχανισμό που θα χρησιμοποιηθεί για την αυθεντικοποίηση. Για να προχωρήσει η αυθεντικοποίηση, ο squid για κάθε ένα από αυτά τα σχήματα αυθεντικοποίησης χρησιμοποιεί κάποιο επιπλέον πρόγραμμα το οποίο αποκαλείται authentication helper και ανάλογα με το backend στο οποίο θα κάνει αυθεντικοποίηση χρησιμοποιεί και το αντίστοιχο helper.

Στο Πανελλήνιο Σχολικό Δίκτυο χρησιμοποιείται σαν backend διακομιστή αυθεντικοποίησης ο OPENLDAP. Ο OPENLDAP του Πανελλήνιου Σχολικού Δικτύου αυτή τη στιγμή, μπορεί να παρέχει 2 ειδών σχήματα αυθεντικοποίησης, τα οποία είναι basic και digest. Τη στιγμή συγγραφής της διπλωματικής το setup του OPENLDAP επέτρεπε μόνο basic authentication. Επιπλέον, το digest authentication ανάλογα με τον πελάτη μπορεί να λειτουργήσει και με md5, πράγμα το οποίο το καθιστά ανασφαλές. Το bearer, βασίζεται πάνω στο oauth το οποίο και από τη μεριά του θεωρείται ανασφαλές πρωτόκολλο, οπότε πάλι θα χρειαζόταν κάποια επιπλέον ασφάλεια (TLS encryption) για την εφαρμογή του (άλλωστε και οι δημιουργοί του squid προτείνουν να μην χρησιμοποιείται και για αυτό το λόγο δεν υπάρχει και κανένας authentication helper για να χρησιμοποιηθεί μαζί με το bearer σχήμα αυθεντικοποίησης με τον Squid).

Το πρωτόκολλο NTLM έχει σταματήσει να αναπτύσσεται από την ίδια την Microsoft, καθώς θεωρείται πολύ ευάλωτο σε επιθέσεις και πλέον χρησιμοποιείται μόνο σαν fallback mechanism για χρήστες-πελάτες οι οποίοι δεν έχουν ακόμα υιοθετήσει το Kerberos authentication mechanism.

Από τη μεριά του το πιο ασφαλές και ταυτόχρονα πολύπλοκο πρωτόκολλο είναι το Kerberos για το οποίο θα γίνει ανάλυση πιο κάτω.

3.4.1 Kerberos Authentication

Το Kerberos είναι ένα δικτυακό πρωτόκολλο αυθεντικοποίησης το οποίο αναπτύχθηκε από το MIT το 1980. Το 1993 παρουσιάστηκε η έκδοση 5 η οποία είναι και αυτή η οποία χρησιμοποιείται μέχρι σήμερα. Το Kerberos είναι ένα πρωτόκολλο το οποίο χρησιμοποιεί κρυπτογραφία μυστικού κλειδιού για ασφαλή επικοινωνία και ανταλλαγή διαπιστευτηρίων μέσα από ανασφαλή δίκτυα. Μέχρι στιγμής είναι το πιο ασφαλές πρωτόκολλο δικτυακής επικοινωνίας (για σχεδόν όλες τις υπηρεσίες).

Παρότι το Kerberos είναι open source και δημιουργήθηκε για λειτουργικά *nix, αναπτύχθηκε περισσότερο από την Microsoft και ενσωματώθηκε στα προϊόντα της σε ένα ενιαίο οικοσύστημα εφαρμογών που χρησιμοποιεί με μεγάλη ευκολία αυτό το πρωτόκολλο (Active Directory). Αντίθετα στα λειτουργικά *nix δεν έχει ενσωματωθεί ακόμα και τώρα πλήρως, με αποτέλεσμα να βλέπουμε διάφορα προϊόντα τα οποία παρουσιάζουν πολλές δυσλειτουργίες μεταξύ των.

Στην Ευρώπη, λόγω της απαγόρευσης εξαγωγής κρυπτογραφικών προϊόντων από την Αμερική, νόμιμα επιτρέπεται η χρήση μιας παραλλαγής του Kerberos από τη Σουηδική εταιρεία Heimdal, πακέτο το οποίο είναι προ εγκατεστημένο και στο FreeBSD.

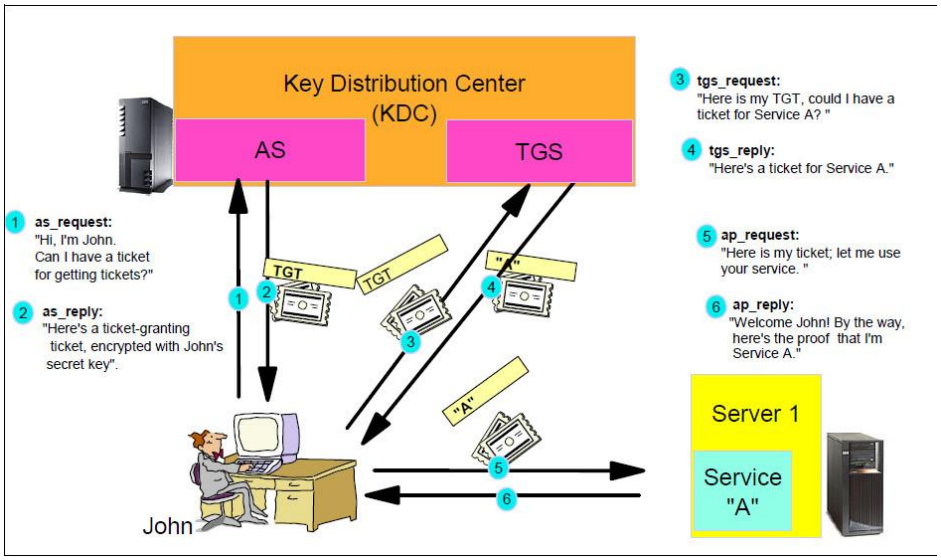
Το Kerberos επειδή χρησιμοποιεί timestamps στα πακέτα του και στους τρόπους πιστοποίησης, είναι πολύ σημαντικό όλο το οικοσύστημα που το χρησιμοποιεί να μην έχει μεγαλύτερη διαφορά ώρας (time skew) πάνω από 2 λεπτά σε σχέση με τον server. Ο server χρησιμοποιεί timestamps σε UTC ώστε να μην έχει πρόβλημα σε διηπειρωτικές συνδέσεις, οπότε ακόμα και μια λάθος ρύθμιση (πολύ κοινό) στην ζώνη ώρας στον πελάτη οδηγεί σε αποτυχία ταυτοποίησης.

Επιπροσθέτως, το Kerberos βασίζεται και είναι tightly integrated με το DNS του περιβάλλοντος που χρησιμοποιείται, και είναι το δεύτερο μεγαλύτερο μειονέκτημά του, καθώς πολλές φορές παρουσιάζει δυσλειτουργίες σε περιβάλλοντα NAT και

περιβάλλοντα με DHCP που δεν ανανεώνουν αυτόματα στον DNS server τις αντιστοιχίες διευθύνσεων με το όνομα του υπολογιστή.

Στο Kerberos υπάρχουν 2 πολύ σημαντικές οντότητες. Είναι αυτή του realm (πιο γνωστό σαν όνομα τομέα) και αφορά το σύνολο υπολογιστών και υπηρεσιών πάνω στο οποίο εφαρμόζεται. Η άλλη σημαντική οντότητα είναι το principal και αφορά κάθε εγγραφή που γίνεται στη βάση δεδομένων του Kerberos. Σαν principal στο Kerberos ορίζεται ο κάθε χρήστης στη μορφή schuser.sch.gr@sch.gr, κάθε υπηρεσία HTTP/squid.sch.gr@sch.gr, αλλά και κάθε υπολογιστής που χρησιμοποιεί το Kerberos στη μορφή host/mypc.sch.gr@sch.gr. Στο παρακάτω παράδειγμα για να μπορέσει ο χρήστης schuser να χρησιμοποιήσει την υπηρεσία HTTP του squid, πρέπει να του επιτρέπεται από τον Kerberos να χρησιμοποιηθεί, καθώς επίσης και ο χρήστης να χρησιμοποιεί τον υπολογιστή mypc, ο οποίος σαν principal (οντότητα) έχει καταχωρηθεί και αναγνωριστεί από τον Kerberos.

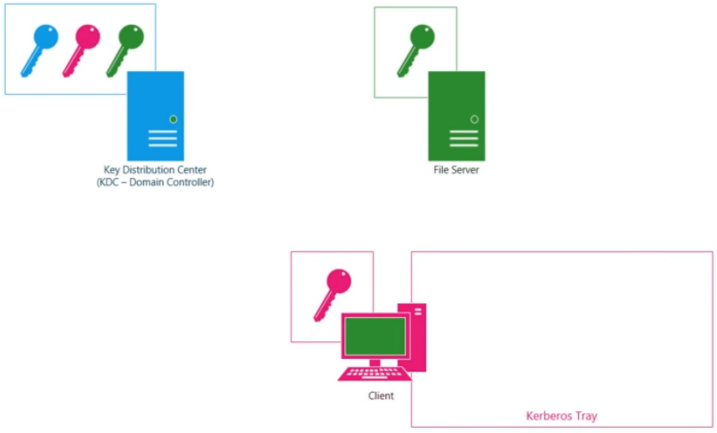
Ο μηχανισμός του Kerberos βασίζεται στο KDC (key distribution center) που με τη σειρά του βασίζεται σε 2 βασικές οντότητες, τον Authentication Server και στον TGT (Ticket Granting Server).



Εικόνα 8: Βασικές οντότητες του Kerberos

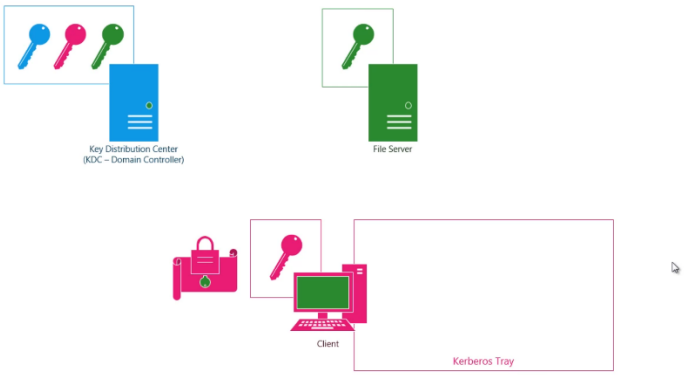
Όπως παρατηρούμε, για να είναι δυνατή η χρήση μιας υπηρεσίας, θα πρέπει πρώτα να έχουμε αποκτήσει ένα TGT (ticket granting tickets) αρχικά από τον KDC. Αυτό είναι και το μειονέκτημα στην περίπτωση μας και θα αναλυθεί παρακάτω.

Το Kerberos για πιστοποίηση σε μια υπηρεσία λειτουργεί ως εξής:



Εικόνα 9: Kerberos Authentication 1ο βήμα

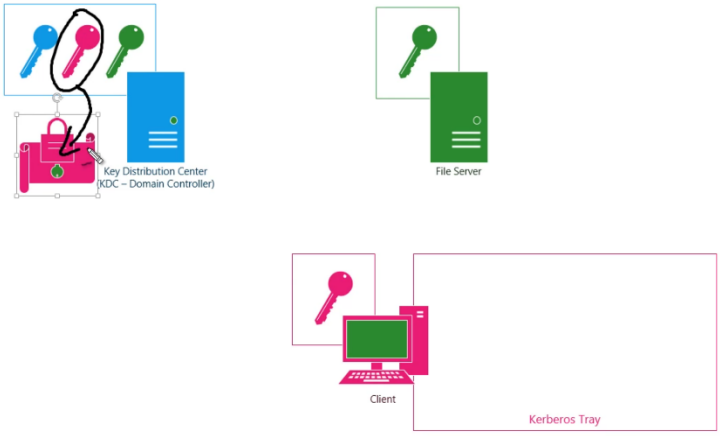
Αρχικά ο KDC περιέχει στην βάση του το ιδιωτικό κλειδί για τον ίδιο τον KDC, το ιδιωτικό κλειδί για τον πελάτη και το ιδιωτικό κλειδί της υπηρεσίας που θέλουμε να χρησιμοποιήσουμε.



.

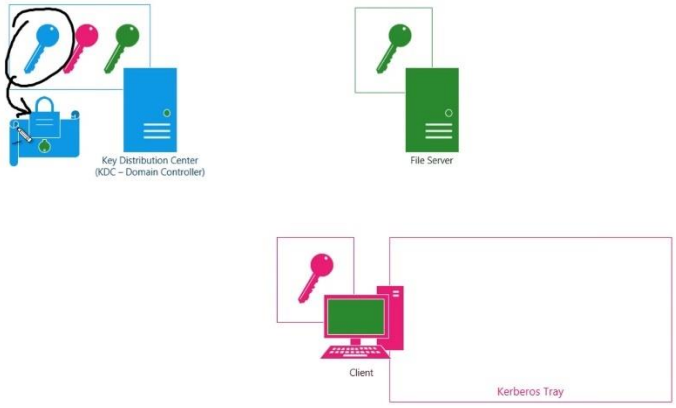
Εικόνα 10: Kerberos Authentication 2ο βήμα

Ο πελάτης (χρήστης) ξεκινάει μια διαδικασία (kinit στο linux με το username και password του, ή αν είναι ρυθμισμένος ο υπολογιστής κατά το αρχικό logon) πιστοποίησης στον KDC. Αρχικά, δημιουργεί ένα πακέτο το οποίο περιέχει μη κρυπτογραφημένο το όνομα χρήστη του και κρυπτογραφεί το υπόλοιπο, το οποίο περιέχει την ημερομηνία και την ώρα όπως και κάποια άλλα στοιχεία με το συνθηματικό που έχει ο χρήστης.



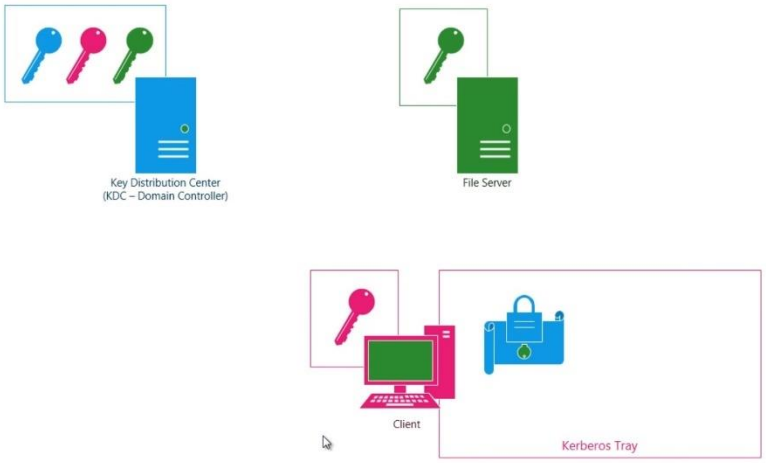
Εικόνα 11: Kerberos Authentication 3ο βήμα

Στη συνέχεια, αφού συμβουλευθεί τα DNS records (υπάρχουν συγκεκριμένα DNS records τα οποία κατευθύνουν προς τον Kerberos στο realm που ανήκει ο πελάτης), στέλνει στον KDC server αυτό το πακέτο. Ο KDC διαβάζει το όνομα χρήστη που είναι μη κρυπτογραφημένο και στη συνέχεια από τη βάση του χρησιμοποιεί το συνθηματικό που γνωρίζει για τον χρήστη και αποκρυπτογραφεί το πακέτο. Αν μπορέσει να αποκρυπτογραφήσει το πακέτο KAI συγκρίνει την ώρα και αν αυτή δεν διαφέρει πάνω από μερικά λεπτά (έως 2), τότε ο KDC αποφασίζει ότι αυτός ο χρήστης είναι όντως αυτός που ισχυρίζεται και αυτό το πακέτο δεν είναι από replay attack (για αυτό πρέπει να είναι συγχρονισμένα τα ρολόγια όλων των υπολογιστών).



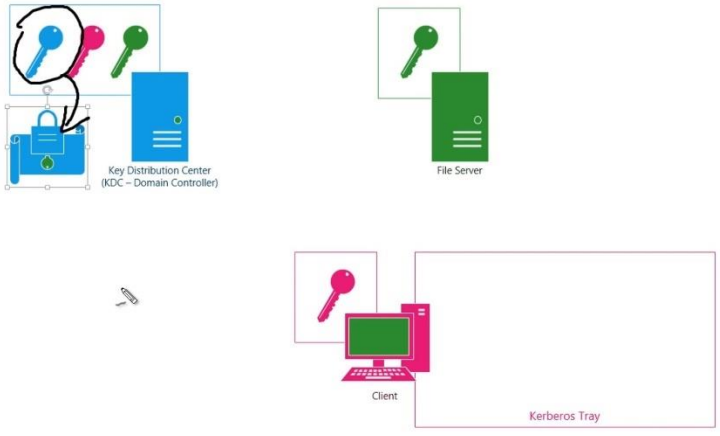
Εικόνα 12: Kerberos Authentication 4ο βήμα

Ο KDC αφού πιστοποίησε τον χρήστη φτιάχνει ένα πακέτο που είναι κρυπτογραφημένο με το δικό του κλειδί και περιέχει κάποιες πληροφορίες για τον χρήστη και το στέλνει πίσω στον χρήστη (ανάλογα με το implementation θα πρέπει στον dns να υπάρχει και το reverse dns της ip που ζήτησε την πιστοποίηση για να το στείλει).



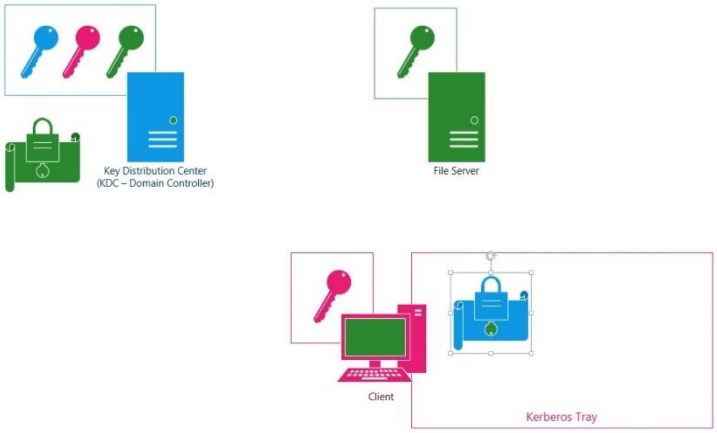
Εικόνα 13: Kerberos Authentication 5ο βήμα

Αφού λάβει το TGT, ο πελάτης το αποθηκεύει σε μια ειδική περιοχή στην μνήμη που ονομάζεται Kerberos tray και δεν γίνεται ποτέ swap για λόγους ασφαλείας. Πλέον, ο πελάτης έχει ένα ticket με το οποίο μπορεί να ζητάει άλλα tickets για άλλες υπηρεσίες, χωρίς ποτέ να έχουν ανταλλάξει οι οντότητες το πραγματικό τους συνθηματικό.



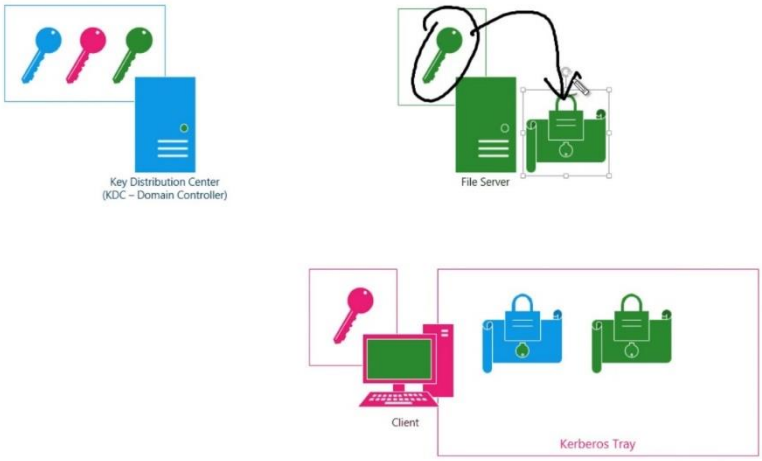
Εικόνα 14: Kerberos Authentication 6ο βήμα

Ο πελάτης επιθυμεί τώρα να χρησιμοποιήσει τον fileserver που ανήκει στο REALM, οπότε στέλνει το TGT που έλαβε από τον KDC μαζί με ένα αίτημα για πρόσβαση στον fileserver.



Εικόνα 15: Kerberos Authentication 7ο βήμα

Ο KDC ξέρει ότι, αφού μπόρεσε και αποκρυπτογράφησε το TGT που του έστειλε ο πελάτης, τότε αυτός είναι πιστοποιημένος από τον ίδιο. Κατόπιν, κοιτάει στη βάση του για την principle fileserver, βρίσκει το συνθηματικό της και κρυπτογραφεί ένα ticket με αυτό το συνθηματικό και το στέλνει στον πελάτη ο οποίος το αποθηκεύει στο Kerberos tray.



Εικόνα 16: Kerberos Authentication 8ο βήμα

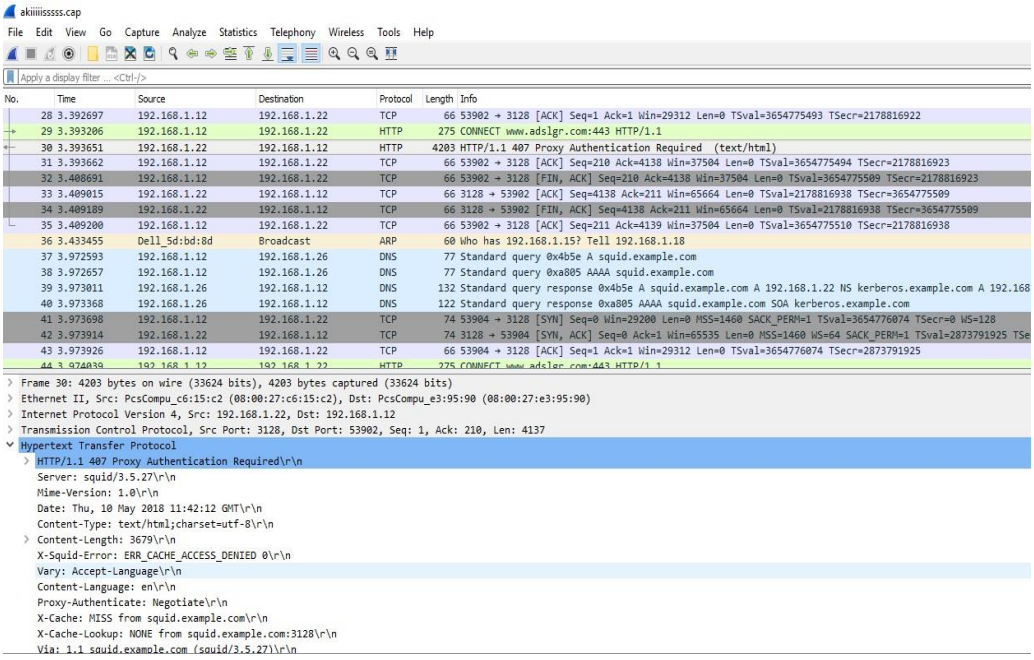
Ο πελάτης τώρα στέλνει το ticket αυτό στον fileserver. Ο fileserver με το συνθηματικό του προσπαθεί να αποκρυπτογραφήσει το πακέτο και αν τα καταφέρει

σημαίνει ότι αυτός ο χρήστης είναι αυτός που ισχυρίζεται και επιπλέον του επιτρέπεται πρόσβαση στις υπηρεσίες μου. Πλέον ο πελάτης επικοινωνεί απευθείας με τον fileserver.

Παρότι το Kerberos είναι το πιο ασφαλές πρωτόκολλο, στην περίπτωση του Πανελληνίου Σχολικού Δικτύου, δεν μπορεί να εφαρμοστεί καθώς οι πελάτες πριν μπορέσουν να αποκτήσουν ένα ticket για την χρησιμοποίηση της υπηρεσίας του squid πρέπει πρώτα να έχουν λάβει ένα TGT από τον KDC. Αυτό σε υπολογιστές που ανήκουν στο Πανελλήνιο Σχολικό Δίκτυο θα πρέπει να γίνει είτε με παρέμβαση του χρήστη μέσω γραμμής εντολών είτε με το όλοι υπολογιστές να γίνουν join στον τομέα (REALM). Θα έπρεπε δηλαδή όλη η υποδομή του Kerberos να υπήρχε ήδη εγκατεστημένη.

Επιπλέον το σενάριο με το Kerberos δεν μπορεί να εφαρμοστεί σε BYOD χωρίς παρέμβαση του χρήστη και σε μερικές περιπτώσεις δεν μπορεί να εφαρμοστεί καθόλου, ειδικά σε κινητές συσκευές που δεν μπορούν να αποκτήσουν έναν TGT από τον KDC.

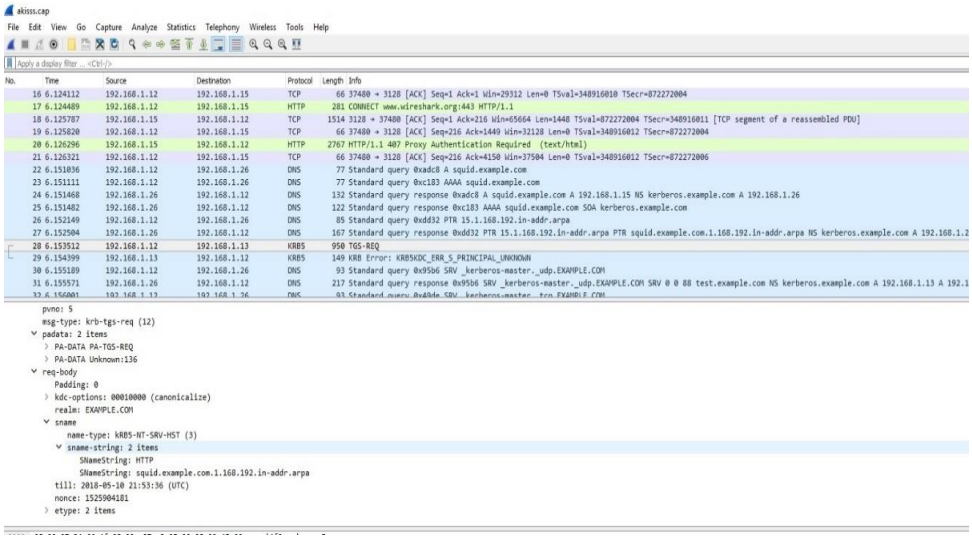
Ακολουθεί ένα τεστ που έγινε κατά το οποίο πρώτα ένας client, που δεν είχε αποκτήσει το αρχικό TGT προσπάθησε να συνδεθεί σε έναν squid server και στο δεύτερο ένας πελάτης που προσπάθησε να συνδεθεί στον squid server, ενώ πρώτα είχε αποκτήσει ένα TGT μέσω της εντολής kinit %username%.



Εικόνα 17: Σύνδεση σε Kerberos Enabled Squid χωρίς TGT

Παρατηρούμε στο προηγούμενο capture ότι παρόλο που ο squid στέλνει στον πελάτη ένα 407 Proxy authentication Required με Proxy-Authenticate Negotiate, ο

πελάτης δεν γνωρίζει που να απευθυνθεί για να αποκτήσει πρόσβαση στην υπηρεσία αυτή, καθώς δεν έχει κανένα TGT για το REALM example.com.



Εικόνα 18: Σύνδεση σε Kerberos Enabled Squid με TGT

Στο παράδειγμα αυτό παρατηρούμε ότι αφού έχει αποκτήσει μέσω του kinit ο πελάτης το TGT, πλέον γνωρίζει που να απευθυνθεί για να πάρει ένα καινούριο ticket για να αποκτήσει πρόσβαση στον squid. Επίσης, βλέπουμε το tight integration του Kerberos με το DNS, καθώς και το ότι ο πελάτης ζητάει ticket όχι για το FQDN, αλλά για το pointer του FQDN του squid.

3.4.2 Digest Authentication

Το digest σχήμα αυθεντικοποίησης, παρότι πολύ ασφαλές σύμφωνα με το τελευταίο RFC στο οποίο έχει ενσωματωθεί το SHA256 hash, κανένας browser μέχρι σήμερα δεν υποστηρίζει ισχυρότερο από MD-5 hash, το οποίο είναι εξαιρετικά ανασφαλές, οπότε και δεν δοκιμάστηκε με τον squid server.

3.4.3 Bearer - OAUTH

Το bearer authentication υποστηρίζεται από τον squid. Οι browser παρόλα αυτά δεν υποστηρίζουν το bearer authentication σε proxy. Για την ακρίβεια από δοκιμές που έγιναν όλοι οι browser δεν απαντούσαν στο 407 status Proxy-Auth* όταν αυτό γινόταν advertised με bearer authentication. Ακόμα και να υποστηριχτεί το συγκεκριμένο σχήμα αυθεντικοποίησης σε διακομιστές μεσολάβησης από τους browsers στο μέλλον, αυτό θα

πρέπει να συνδυαστεί όπως και το basic σχήμα με TLS, καθώς αν χρησιμοποιήσουμε πχ.OAUTH, τα tokens θα διακινούνται σε plain text προς τον proxy. Έτσι θα μπορούσε κάποιος επιτιθέμενος να αποκτήσει το token μας για πρόσβαση όπου αυτό έχει οριστεί ότι θα έχει πρόσβαση.

Ένας επιπλέον αποτρεπτικός παράγοντας είναι ότι πολλά implementation του OAUTH, παραβιάζουν το RFC6750. Συγκεκριμένα χρησιμοποιούν επιπλέον το Form field των κεφαλίδων, κάτι το οποίο δεν είναι σύμφωνο με το RFC. Άλλα implementations χρησιμοποιούν το URL query parameter το οποίο θεωρείται ανασφαλές.

Το OAUTH authentication θα μπορούσε υπό προϋποθέσεις να χρησιμοποιηθεί σε περίπτωση που ο squid server ήταν σε διαφανή χρήση. Θα μπορούσε σε αυτή την περίπτωση στον ίδιο τον squid να τρέχει ένας αντίστροφος διακομιστής μεσολάβησης ο οποίος θα έκανε oauth αυθεντικοποίηση και μετά θα προωθούσε όλη την κίνηση στον squid, σαν να ήταν ο squid ο web server για τον οποίο έκανε αυθεντικοποίηση ο squid. Βέβαια η συγκεκριμένη λειτουργία θα παραβίαζε το http και το oauth πρωτόκολλο και θα αύξανε την υπολογιστική ισχύ που θα χρειαζόταν κάθε μηχάνημα. Σε μια τέτοια δοκιμή που έγινε για μερικούς τομείς μπορούσε να λειτουργήσει και για μερικούς άλλους όχι.

Αυτό είναι επακόλουθο της παραβίασης του oauth πρωτοκόλλου και συγκεκριμένα για το realm. Σε αυτή την περίπτωση όμως ο squid δεν θα μπορούσε να λειτουργήσει σαν accounting server, δηλαδή δεν θα γνωρίζαμε ποιος χρήστης χρησιμοποίησε ποια υπηρεσία.

3.4.4 Basic Authentication

Παρότι το σχήμα αυθεντικοποίηση basic είναι το πιο ανασφαλές από όλα τα σχήματα αυθεντικοποίησης, σε συνδυασμό με το Secure TLS Web Proxy, είναι τόσο ασφαλές όσο το TLS. Είναι σχετικά απλό στη συντήρηση και σε συνδυασμό με το TLS είναι απόλυτα ασφαλές. Η επικοινωνία μεταξύ πελάτη και squid γίνεται μέσω TLS, οπότε τα διαπιστευτήρια είναι ασφαλή και η επικοινωνία μεταξύ squid και LDAP μπορεί επίσης να γίνει μέσω TLS με το LDAPS πρωτόκολλο.

Οι περισσότερες δοκιμές έγιναν με το basic authentication αλλά πάντα μέσω TLS.

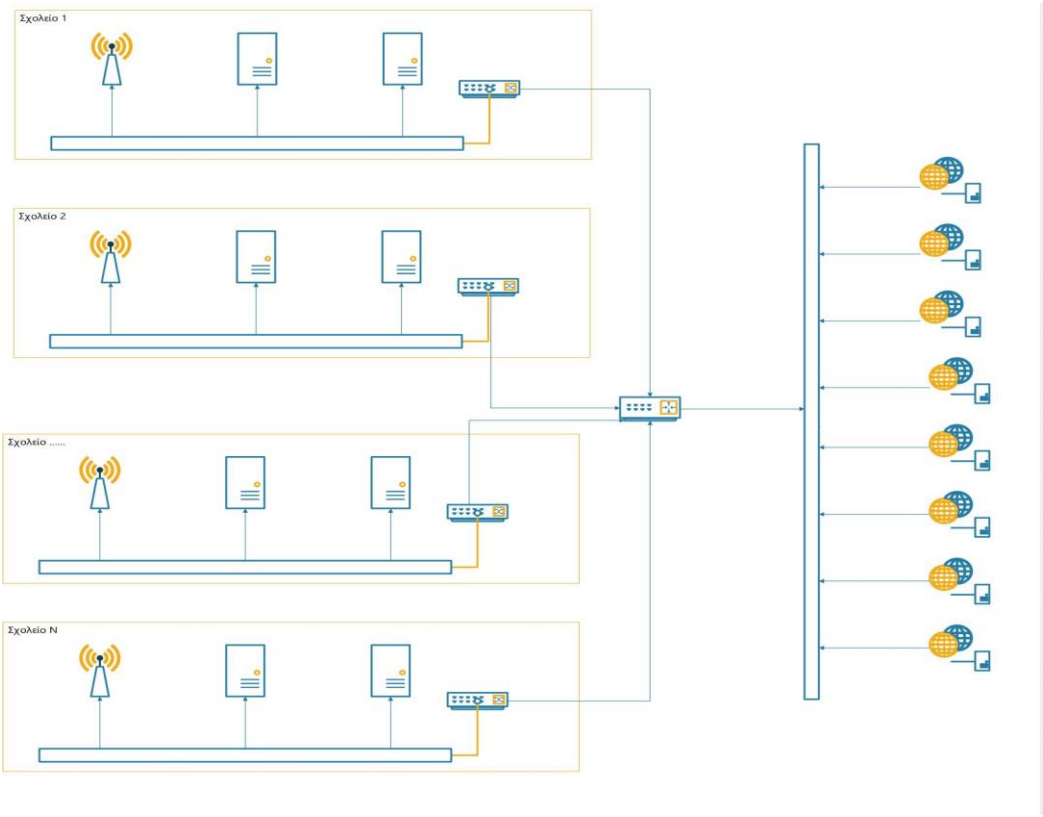
4 ΔΟΜΗ ΔΙΚΤΥΟΥ ΚΑΙ ΑΝΑΛΥΣΗ

4.1 ΥΠΑΡΧΟΝ ΔΙΚΤΥΟ ΜΕ TRANSPARENT PROXY

Το ΠΣΔ πριν την εφαρμογή της παρούσας διπλωματικής λειτουργούσε με διακομιστές μεσολάβησης σε διαφανή χρήση (transparent mode), δηλαδή η συσκευή του χρήστη δεν γνώριζε για την ύπαρξη διακομιστή μεσολάβησης με αποτέλεσμα να μην μπορεί να χρησιμοποιήσει τις εκτεταμένες υπηρεσίες του http/s πρωτοκόλλου οι οποίες χρησιμοποιούνται όταν η συσκευή του χρήστη είναι proxy-aware. Για να μπορέσει το δίκτυο και ειδικά οι διακομιστές μεσολάβησης να υποστηρίξουν τη μεγάλη κίνηση υπάρχουν 16 διακομιστές μεσολάβησης στους οποίους διανέμεται η κίνηση. Η κίνηση αυτή διανέμεται μέσω του WCCP πρωτοκόλλου της CISCO από ένα switch στο οποίο περνάει όλη η κίνηση του σχολικού δικτύου προς το internet για το πρωτόκολλο http. Η διανομή της κίνησης γίνεται σύμφωνα με το τελευταίο bit του προορισμού. Βέβαια αυτό πλέον δεν μας διασφαλίζει την τοπικότητα του περιεχομένου, καθώς πολλές υπηρεσίες έχουν πολλαπλές IP για μια υπηρεσία.

Σε αυτή τη λειτουργία (transparent proxy) δεν μπορεί να δουλέψει ο μηχανισμός πιστοποίησης χρήστη καθώς ο browser του χρήστη δεν είναι ενήμερος για τις αλλαγές που πρέπει να κάνει στις κεφαλίδες όπως είδαμε στο προηγούμενο κεφάλαιο.

Η λειτουργία των διακομιστών μεσολάβησης πριν την εφαρμογή των αλλαγών για forward authenticated proxy συνοψίζονται στην παρακάτω εικόνα:



Εικόνα 19: Current Proxy Network Transparent Proxy

Ανάλογα με τη λειτουργία του διακομιστή μεσολάβησης που θα επιλέξουμε, αυξάνονται και οι λειτουργίες και το πλεονέκτημα χρήσης του. Σε έναν διακομιστή μεσολάβησης που λειτουργεί σε διαφανή λειτουργία δεν είναι δυνατή η αυθεντικοποίηση, η διαμεσολάβηση για το https πρωτόκολλο οπότε και δεν μπορεί να φιλτραριστεί η κίνηση (web/url filtering) στο https πρωτόκολλο και να εφαρμοστούν λίστες αποκλεισμού στους χρήστες. Το ΠΣΔ λειτουργούσε έτσι μέχρι τώρα, δηλαδή όλη η κίνηση για το https πρωτόκολλο δεν γινόταν μέσω μεσολάβησης και άρα δεν μπορούσε να φιλτραριστεί το περιεχόμενό του (έστω και ως προς το όνομα τομέα).

Για να μπορέσουν οι διακομιστές μεσολάβησης να μετατραπούν σε AAA (Authentication, Authorization & Accounting Servers) έπρεπε να γίνουν αρκετές αλλαγές στην υποδομή του δικτύου, στον τρόπο λειτουργίας καθώς και αρκετά τεστ αφού ο φόρτος εργασίας σε αυτούς τους διακομιστές θα αυξανόταν πάρα πολύ ανάλογα και με το σενάριο

χρήσης. Σε όλα τα σενάρια χρήσης πλέον τα https request θα περνάνε μέσω των διακομιστών μεσολάβησης, με αποτέλεσμα να διπλασιαστεί η κίνηση σε αυτούς όσον αφορά τα επιπλέον αιτήματα https. Ένας επιπλέον επιβαρυντικός παράγοντας είναι και η αυθεντικοποίηση μέσω του LDAP directory του ΠΣΔ.

4.2 ΑΛΛΑΓΕΣ ΣΤΟ ΔΙΚΤΥΟ ΓΙΑ ΛΕΙΤΟΥΡΓΙΑ ΜΕ FORWARD AUTHENTICATED PROXY

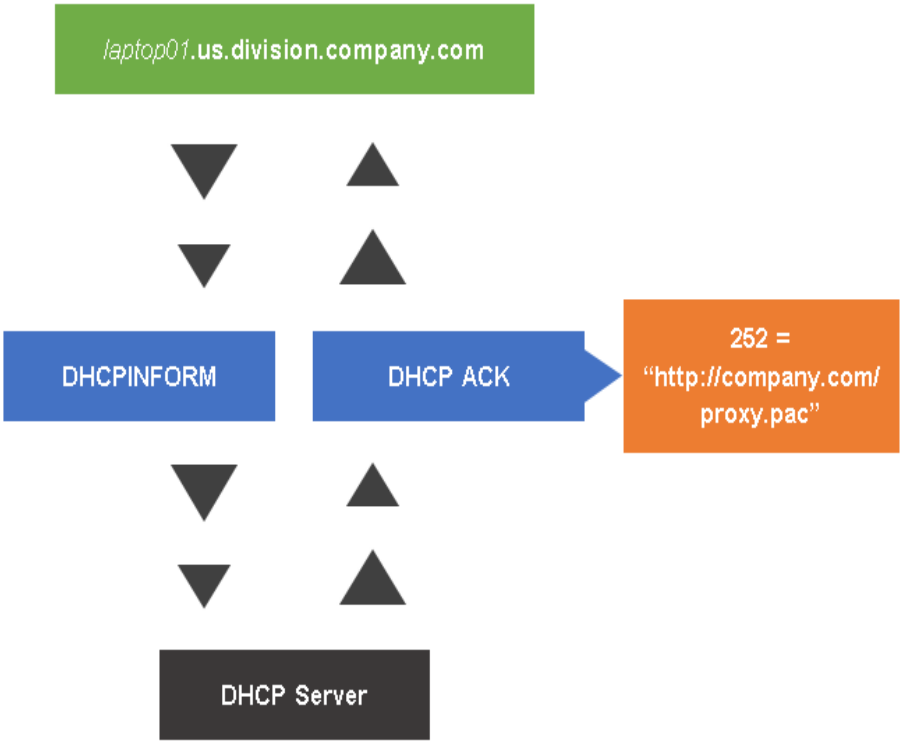
Το υπάρχον δίκτυο ήταν αδύνατο να υποστηρίξει αυτούσιο διακομιστές μεσολάβησης με αυθεντικοποίηση. Για να μπορεί ένας browser να υποστηρίξει αυθεντικοποίηση θα πρέπει πρώτα να γνωρίζει ακριβώς τον διακομιστή μεσολάβησης στον οποίο θα απευθυνθεί. Λόγω του μεγέθους του σχολικού δικτύου ήταν αδύνατο να ενημερώσουμε κάθε χρήστη και κάθε συσκευή για τις αλλαγές που πρέπει να γίνουν στις συσκευές τους ώστε να μπορέσουν να χρησιμοποιήσουν το σχολικό δίκτυο.

Σε αυτό το πρόβλημα, λύση δύναται να μας δώσουν δυο γνώριμα πρωτόκολλα, σε συνδυασμό με ένα πρωτόκολλο των web browser, που μας επιτρέπουν να γίνεται αυτόματα η ρύθμιση του διακομιστή μεσολάβησης χωρίς παρέμβαση του χρήστη.

Οι web browser μπορούν πλέον να ανιχνεύουν αυτόματα την ύπαρξη ενός διακομιστή μεσολάβησης εντός ενός δικτύου μέσω ενός αρχείου αυτόματης ρύθμισης μέσω του πρωτοκόλλου Web Proxy Auto-Discovery Protocol. Το πρωτόκολλο WPAD περιγράφει μόνο τον μηχανισμό για να βρεθεί το αρχείο αυτό ρύθμισης αλλά το πιο γνωστό και διαδεδομένο format αρχείου είναι το proxy auto-config (pac file). Για να γνωρίζει ο web browser που βρίσκεται αυτό το αρχείο ρύθμισης και να ρυθμιστεί μόνος του, το πρωτόκολλο WPAD παρέχει 2 επιλογές. Το WPAD πρωτόκολλο ελέγχει μέσω του πρωτοκόλλου DHCP και DNS αν υπάρχει ρητή εντολή για κάποιο τέτοιο αρχείο αυτορρύθμισης. Η διαδικασία αναλύεται πιο κάτω ξεχωριστά για το κάθε πρωτόκολλο.

4.3 ΑΥΤΟΡΡΥΘΜΙΣΗ BROWSER ΓΙΑ ΔΙΑΚΟΜΙΣΤΗ ΜΕΣΟΛΑΒΗΣΗΣ ΜΕΣΩ DHCP

Το πρωτόκολλο DHCP έχει ένα ειδικό option το 252 (σύμφωνα με τον οργανισμό IANA για private use), το οποίο περιέχει τη διεύθυνση του διακομιστή ιστού που περιέχει το αρχείο wpad.dat. Το συγκεκριμένο option είναι vendor specific και είναι λειτουργικό μόνο εντός της οικογένειας προϊόντων της Microsoft. Για το λόγο αυτό δεν επιλέχθηκε για την παρούσα υλοποίηση και δεν γίνεται περαιτέρω ανάλυση, απλώς παρουσιάζεται συνοπτικά στην παρακάτω εικόνα :



Εικόνα 20: PAC auto-config DHCP

4.4 ΑΥΤΟΡΡΥΘΜΙΣΗ BROWSER ΓΙΑ ΔΙΑΚΟΜΙΣΤΗ ΜΕΣΟΛΑΒΗΣΗΣ ΜΕΣΩ DNS

Σχεδόν όλοι οι καινούριοι browser υποστηρίζουν την ανακάλυψη του αρχείου αυτορρύθμισης από το πρωτόκολλο Web Proxy Auto-Discovery μέσω του DNS.

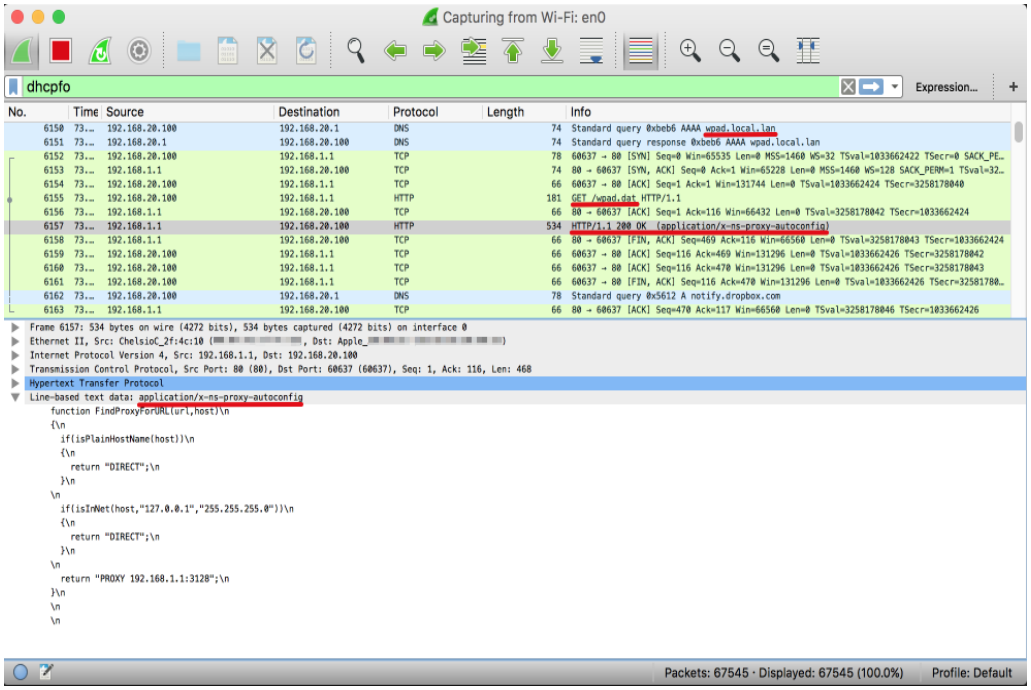
Αρχικά με το που ανοίξει ο browser ΚΑΙ έχει επιλεγμένη την αυτόματη ρύθμιση proxy, τότε αρχίζει και ψάχνει μέσω του DNS πρωτόκολλου ως εξής :

- Αρχικά αφαιρεί το όνομα του σταθμού εργασίας/συσκευής (πχ. 1.sxolio1.sch.gr) από όλο το όνομα τομέα και δοκιμάζει αν υπάρχει διακομιστής με το όνομα wpad.sxolio1.sch.gr και αν υπάρχει, ελέγχει αν έχει έγκυρο pac αρχείο.
- Αν αποτύχει το προηγούμενο βήμα δοκιμάζει με όνομα διακομιστή wpad.sch.gr και με την ίδια λογική επιτυγχάνει ή αποτυγχάνει.
- Σε μερικά implementations η αναζήτηση φτάνει μέχρι το επίπεδο wpad.gr, αλλά αυτά τα ονόματα τομέα από το 2007 και μετά έχουν μπλοκαριστεί για λόγους ασφαλείας.

Όταν ολοκληρωθούν αυτά τα βήματα και έχει βρεθεί ένα έγκυρο PAC αρχείο τότε ο browser ρυθμίζεται αυτόματα, ώστε να χρησιμοποιεί το συγκεκριμένο/συγκεκριμένους διακομιστές μεσολάβησης.

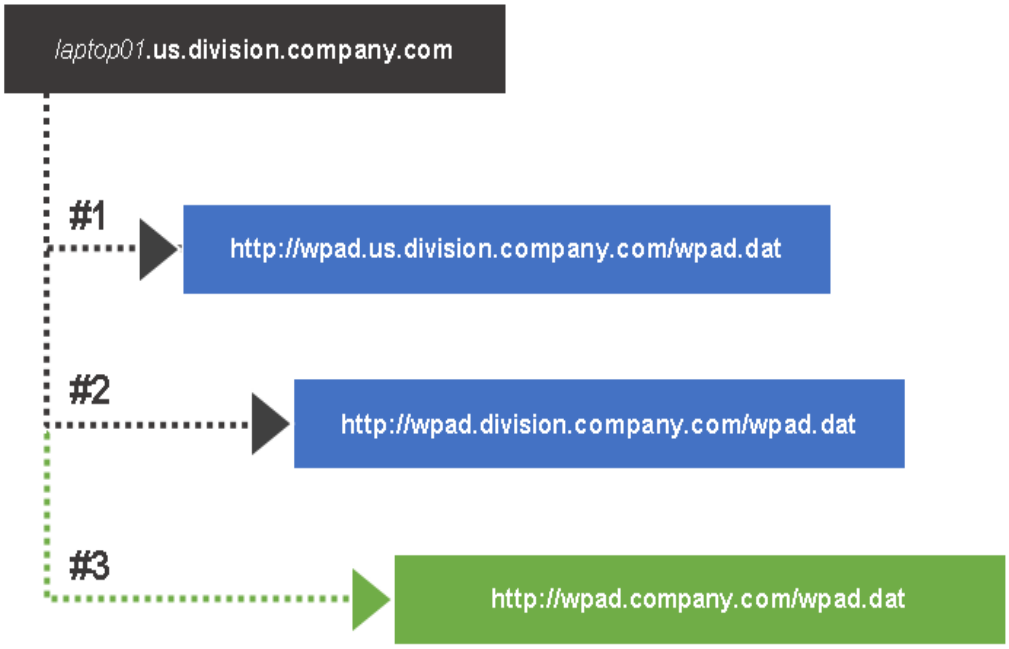
Βασική προϋπόθεση για να δουλέψει το συγκεκριμένο σενάριο είναι η συσκευή η οποία θα εισέλθει εντός του σχολικού δικτύου με όνομα τομέα της μορφής sch.gr. Για να επιτευχθεί αυτό για όλες τις συσκευές θα πρέπει ο δρομολογητής κάθε σχολείου να έχει ενεργοποιημένο το option 15 (DNS domain name) του πρωτοκόλλου DHCP για το όνομα τομέα sch.gr ή για κάποιο υποτομέα x.y.z.sch.gr.

Στην εικόνα παρατηρούμε πως δουλεύει το πρωτόκολλο WPAD σε έναν browser που έχει επιλεγμένη την ρύθμιση για proxy auto config.



Εικόνα 21: Καταγραφή WPAD

1. Ο browser με το που ανοίγει κάνει ένα αίτημα για το διακομιστή wpad.local.lan
2. Ο DNS του απαντάει με τη διεύθυνση του διακομιστή wpad.local.lan
3. Ο browser ζητά το αρχείο wpad.dat (GET /wpad.dat)
4. Ο διακομιστής του επιστρέφει το αρχείο wpad.dat
5. Πλέον ο browser χρησιμοποίησε το αρχείο wpad.dat, το οποίο του λέει ότι ο διακομιστής μεσολάβησης είναι ο 192.168.1.1:3128.



Εικόνα 22: PAC auto-config DNS

4.4.1 Μορφή αρχείου Proxy Auto Config (PAC file)

Το αρχείο PAC είναι ένα αρχείο javascript το οποίο έχει μερικές αυστηρά δηλωμένες συναρτήσεις από τις οποίες μπορεί ο browser να βρει για κάθε δίκτυο, για κάθε host ή για ποια ημέρα της εβδομάδας ποιον διακομιστή μεσολάβησης θα χρησιμοποιήσει ή/και για ποια δίκτυα δεν θα χρησιμοποιήσει διακομιστή μεσολάβησης. Η ρύθμιση αυτή μπορεί να υποστηρίξει ένα απλό δίκτυο με έναν διακομιστή έως και ένα πολύπλοκο δίκτυο με πολλούς διακομιστές μεσολάβησης, διαφορετικά υποδίκτυα και ένα περιορισμένο τρόπο διαμοιρασμού φόρτου εργασίας σε διαφορετικούς διακομιστές μεσολάβησης.

Η γενική πιο απλή μορφή ενός PAC αρχείου έχει την εξής δομή για να είναι λειτουργικό :

```
function FindProxyForURL(url, host){
    // ...
}

ret = FindProxyForURL(url, host);
```

Σε αυτή τη συνάρτηση το όρισμα url αναφέρεται στο URL το οποίο προσπαθεί να συνδεθεί ο χρήστης-πελάτης. Το όρισμα host είναι το όνομα συσκευής (μαζί με τον τομέα) και γίνεται εξαγωγή από το url. Το δεύτερο όρισμα χρησιμοποιείται περισσότερο για ευκολία, καθώς απλώς περιλαμβάνει το βασικό όνομα τομέα το οποίο περιλαμβάνεται μεταξύ των :// και του πρώτου συμβόλου / (πχ το host για το https://www.sch.gr:9090/1/2/3/index.html είναι το www.sch.gr).

Το όρισμα ret (return) επιστρέφει ένα string το οποίο περιγράφει τη ρύθμιση των διακομιστών μεσολάβησης. Αν το string είναι κενό τότε δεν χρησιμοποιείται διακομιστής μεσολάβησης. Το string μπορεί να περιέχει οποιοδήποτε συνδυασμό από τα πιο κάτω ορίσματα (ρυθμίσεις) τα οποία διαχωρίζονται με semicolon (Ελληνικό ερωτηματικό):

- **DIRECT**: Οι συνδέσεις γίνονται απευθείας χωρίς διακομιστή μεσολάβησης.
- **PROXY host:port**: Ο διακομιστής μεσολάβησης “host” θα χρησιμοποιηθεί ο οποίος περιμένει αιτήματα στην πόρτα “port”.
- **SOCKS host:port**: Όπως από πάνω αλλά ο διακομιστής μεσολάβησης χρησιμοποιεί πρωτόκολλο SOCKS.

Οι πιο καινούριες εκδόσεις Firefox και Chrome (οι οποίες υποστηρίζουν Secure Web Proxy) υποστηρίζουν επίσης :

- **HTTP host:proxy** : Το ίδιο με PROXY host:proxy.
- **HTTPS host:proxy** : Με αυτό το όρισμα, ο χρήστης πελάτης χρησιμοποιεί έναν Secure Web Proxy, δηλαδή όλη η επικοινωνία μεταξύ χρήστη πελάτη και διακομιστή μεσολάβησης είναι κρυπτογραφημένη με TLS.

Αν το string που επιστραφεί έχει πολλαπλές τιμές χωρισμένες με semicolon τότε ο χρήστης πελάτης χρησιμοποιεί το αριστερότερο (πρώτο στη σειρά) όρισμα, μέχρι να μην μπορεί να επικοινωνήσει μαζί του. Σε αυτή την περίπτωση χρησιμοποιείται το δεύτερο όρισμα κ.ο.κ. Σε περίπτωση απώλειας επικοινωνίας, ο χρήστης-πελάτης θα χρησιμοποιεί το δεύτερο όρισμα για 30 λεπτά, μετά τα 30 λεπτά θα προσπαθήσει να επικοινωνήσει με το διακομιστή μεσολάβησης που βρίσκεται στο πρώτο όρισμα. Αν αποτύχει πάλι θα ξαναπροσπαθήσει να επικοινωνήσει μαζί του μετά από 1 ώρα και έπειτα θα προσθέτει 30 λεπτά σε κάθε προηγούμενη προσπάθεια και θα προσπαθεί πάλι.

Αν δεν υπάρχει επικοινωνία με όλους τους διακομιστές μεσολάβησης και δεν υπάρχει το όρισμα DIRECT, ο χρήστης-πελάτης θα ρωτήσει τον χειριστή αν όλοι οι διακομιστές μεσολάβησης πρέπει να αγνοηθούν και οι συνδέσεις να γίνονται απευθείας με τον προορισμό. Ανεξαρτήτως εισαγωγής από το χειριστή ο χρήστης-πελάτης θα προσπαθήσει να επικοινωνήσει με τους διακομιστές μεσολάβησης που δηλώνονται στο όρισμα return μετά από 20 λεπτά. Αν δεν υπάρξει επικοινωνία θα προσπαθεί κάθε φορά σε χρονικό διάστημα ίσο με το προηγούμενο και κάθε φορά θα προσθέτει 20 λεπτά.

Το αρχείο PAC υποστηρίζει τις εξής προκαθορισμένες συναρτήσεις :

- Συνθήκες βασιζόμενες στο όρισμα host
  - isPlainHostName()
  - dnsDomainIs()
  - localHostOrDomainIs()
  - isResolvable()
  - isInNet()
- Συναρτήσεις
  - dnsResolve()
  - convert_addr()
  - myIpAddress()
  - dnsDomainLevels()
- Συνθήκες βασιζόμενες στο όρισμα host/url
  - shExpMatch()
- Συνθήκες βασιζόμενες στον χρόνο
  - weekdayRange()
  - dateRange()
  - timeRange()

Στο εξής το όρισμα host σε κάθε συνάρτηση είναι το hostname από το url χωρίς τον αριθμό της πόρτας, εκτός αν ορίζεται διαφορετικά.

4.4.1.1 isPlainHostname()

4.4.1.1.1 Σύνταξη

isPlainHostName(*host*)

4.4.1.1.2 Περιγραφή

Επιστρέφει Boolean και είναι αληθής αν και μόνο αν δεν υπάρχει hostname στο όρισμα hosts (δεν υπάρχουν δηλαδή τελείες).

4.4.1.1.3 Παραδείγματα

```
function FindProxyForURL(url, host) {
  if (!isPlainHostName(host))
    return host;
}

//epistrefei "www.sch.gr"
```

```
function FindProxyForURL(url, host) {
  if (isPlainHostName("www"))
    return "isPlainHostName true";
  return "isPlainHostName false";
}

//epistrefei "isPlainHostName true"
```

4.4.1.2 dnsDomainIs()

4.4.1.2.1 Σύνταξη

dnsDomainIs(*host, domain*)

4.4.1.2.2 Παράμετροι

domain

Το όνομα του τομέα με το οποίο θα κάνουμε τη σύγκριση.

4.4.1.2.3 Περιγραφή

Επιστρέφει Boolean και είναι αληθής αν και μόνο αν τα ορίσματα host και domain ταιριάζουν.

4.4.1.2.4 Παραδείγματα

```
function FindProxyForURL(url, host) {
  if (dnsDomainIs("www.sch.gr", ".sch.gr"))
    return "dnsDomainIs true";
  return "dnsDomainIs false";
}

//epistrefei "dnsDomainIs false"
```

```
function FindProxyForURL(url, host) {
  if (dnsDomainIs("www", ".sch.gr"))
    return "dnsDomainIs true";
  return "dnsDomainIs false";
}

//returns "dnsDomainIs false "
```

4.4.1.3 localHostOrDomainIs()

4.4.1.3.1 Σύνταξη

localHostOrDomainIs(*host, hostdom*)

4.4.1.3.2 Παράμετροι

hostdom

Το FQDN το οποίο θα γίνει η σύγκριση

4.4.1.3.3 Description

Επιστρέφει αληθές αν το hostname ταιριάζει ακριβώς,

4.4.1.3.4 Παραδείγματα

```
function FindProxyForURL(url, host) {
  if (localHostOrDomainIs("www.mozilla.org", "www.mozilla.org"))
    return "localHostOrDomainIs is true (exact match)";
  return "localHostOrDomainIs is false";
}

//returns "localHostOrDomainIs is true (exact match)"
function FindProxyForURL(url, host) {
  if (localHostOrDomainIs("www", "www.mozilla.org"))
    return "localHostOrDomainIs is true (hostname match, domain not specified))";
  return "localHostOrDomainIs is false";
}

//returns "localHostOrDomainIs is true (hostname match, domain not specified))"
function FindProxyForURL(url, host) {
  if (localHostOrDomainIs("www.google.com", "www.mozilla.org"))
    return "localHostOrDomainIs is true";
  return "localHostOrDomainIs is false (domain name mismatch)";
}
```

```
// returns "localHostOrDomainIs is false (domain name mismatch)"
function FindProxyForURL(url, host) {
  if (localHostOrDomainIs("home.mozilla.org", "www.mozilla.org"))
    return "localHostOrDomainIs is true";
  return "localHostOrDomainIs is false (domain name mismatch)";
}

// returns "localHostOrDomainIs is false (hostname mismatch)"
```

4.4.1.4 isResolvable()

```
function FindProxyForURL(url, host) {
  if (isResolvable("www.mozilla.org"))
    return "isResolvable is true";
  return "isResolvable is false";
}

// returns "isResolvable is true"
```

4.4.1.5 isInNet()

```
function FindProxyForURL(url, host) {
// put in the address returned by dnsResolve (see next example)
  if (isInNet(host, "63.245.213.24", "255.255.255.255"))
    return "isInNet is true";
  return "isInNet is false";
}

// returns "isInNet is true"
```

4.4.1.6 dnsResolve()

```
function FindProxyForURL(url, host) {
  return dnsResolve("www.mozilla.org");
}

//returns the string "104.16.41.2"
```

4.4.1.7 convert_addr()

```
function FindProxyForURL(url, host) {
  return convert_addr("104.16.41.2");
}

//returns the decimal number 1745889538
```

4.4.1.8 myIpAddress()

4.4.1.8.1 Example

```
function FindProxyForURL(url, host) {
  return dnsDomainLevels("www");
}

//returns 0
function FindProxyForURL(url, host) {
  return dnsDomainLevels("mozilla.org");
}

//returns 1
function FindProxyForURL(url, host) {
  return dnsDomainLevels("www.mozilla.org");
}

//returns 2
```

4.4.1.9 dnsDomainLevels()

```
function FindProxyForURL(url, host) {
  return dnsDomainLevels("www");
}

//returns 0
function FindProxyForURL(url, host) {
  return dnsDomainLevels("mozilla.org");
}

//returns 1
function FindProxyForURL(url, host) {
  return dnsDomainLevels("www.mozilla.org");
}

//returns 2
```

4.4.1.10 shExpMatch()

```
function FindProxyForURL(url, host) {
  return shExpMatch("http://home.netscape.com/people/ari/index.html", "*/ari/*");
}

//returns true
function FindProxyForURL(url, host) {
  return shExpMatch("http://home.netscape.com/people/montulli/index.html", "*/ari/*");
}

//returns false
```

4.4.1.11 Παράδειγμα : Χρησιμοποίηση proxy εκτός του τοπικού υποδικτύου

Όλοι οι host οι οποίοι δεν είναι fully qualified ή αυτοί που είναι στο τοπικό domain θα συνδεθούν απευθείας. Όλα τα υπόλοιπα θα πάνε μέσω proxy.

```
function FindProxyForURL(url, host) {
  if (isPlainHostName(host) || dnsDomainIs(host, ".sch.gr")) {
    return "DIRECT";
  } else {
    return "HTTPS proxynew.sch.gr:8080";
  }

}
```

```
function FindProxyForURL(url, host) {
  if ((isPlainHostName(host) ||
      dnsDomainIs(host, ".mozilla.org")) &&
      !localHostOrDomainIs(host, "www.mozilla.org") &&
      !localHostOrDoaminIs(host, "merchant.mozilla.org")) {
        return "DIRECT";
  } else {
    return "PROXY w3proxy.mozilla.org:8080; DIRECT";
  }
}
```

4.4.1.12 Χρησιμοποίηση proxy μόνο εάν δεν μπορεί να επιλυθεί το όνομα τομέα

Αυτό το παράδειγμα εφαρμόζεται σε περιπτώσεις που υπάρχει εσωτερικός dns server ο οποίος μπορεί να κάνει resolve μόνο εσωτερικά hostnames. Η ιδέα είναι ότι για τα εσωτερικά sch.gr δεν χρησιμοποιούμε proxy ενώ για όλα τα υπόλοιπα χρησιμοποιούμε.

```
function FindProxyForURL(url, host) {
  if (isResolvable(host))
    return "DIRECT";
  else
    return "HTTPS proxynew.sch.gr:8080";

}
```

Το παραπάνω παράδειγμα απαιτεί την συνδρομή του DNS server κάθε φορά, αυτό μπορεί να μπει σε ένα group πιο έξυπνα ώστε ο DNS να ερωτάται μόνο εάν οι άλλοι κανόνες δεν έχουν φέρει κάποιο αποτέλεσμα:

```
function FindProxyForURL(url, host) {
  if (isPlainHostName(host) ||
      dnsDomainIs(host, ".sch.gr") ||
      isResolvable(host)) {
    return "DIRECT";
  } else {
    return "HTTPS proxynew.sch.gr:8080";
  }

}
```

4.4.1.13 Subnet based decisions

```
function FindProxyForURL(url, host) {
  if (isInNet(host, "198.95.0.0", "255.255.0.0"))
    return "DIRECT";
  else
    return "PROXY proxynew.sch.gr:8080";

}
```

```
function FindProxyForURL(url, host) {
  if (isPlainHostName(host) ||
      dnsDomainIs(host, ".mydomain.com") ||
      isInNet(host, "198.95.0.0", "255.255.0.0")) {
    return "DIRECT";
  } else {
    return "PROXY proxy.mydomain.com:8080";
  }

}
```

4.4.1.14 Load balancing/routing βασιζόμενο στο url

```
function FindProxyForURL(url, host) {
  if (isPlainHostName(host) || dnsDomainIs(host, ".mydomain.com"))
      return "DIRECT";
  else if (shExpMatch(host, "*.com"))
      return "PROXY cache01.att.sch.gr:8080; " +
```

```
            "PROXY cache02.att.sch.gr:8080";
  else if (shExpMatch(host, "*.edu"))
     return "PROXY cache03.att.sch.gr:8080; " +
            "PROXY cache04.att.sch.gr:8080";
  else
     return "PROXY cache05.att.sch.gr:8080; " +
            "PROXY cache06.att.sch.gr:8080";

}
```

4.4.1.15 Ρύθμιση proxy για διαφορετικά πρωτόκολλα

```
function FindProxyForURL(url, host) {
  if (url.substring(0, 5) == "http:") {
    return "PROXY cache01.att.sch.gr:8080";
  }
  else if (url.substring(0, 4) == "ftp:") {
    return "PROXY cache02.att.sch.gr:8080";
  }
  else if (url.substring(0, 7) == "gopher:") {
    return "PROXY cache03.att.sch.gr:8080";
  }
  else if (url.substring(0, 6) == "https:" ||
           url.substring(0, 6) == "snews:") {
    return "PROXY cache04.att.sch.gr:8080";
  } else {
    return "DIRECT";
  }

}
```

Σημείωση : Το ίδιο αποτέλεσμα μπορεί να επιτευχθεί αν χρησιμοποιηθεί η shExpMatch() function

```
if (shExpMatch(url, "http:*")) {
  return "PROXY cache01.att.sch.gr:8080";
}
```

4.5 WEB SERVER ΠΟΥ ΕΞΥΠΗΡΕΤΕΙ ΑΙΤΗΜΑΤΑ ΓΙΑ ΑΡΧΕΙΟ ΑΥΤΟΡΡΥΘΜΙΣΗΣ

Όπως είδαμε παραπάνω το αρχείο αυτορρύθμισης του χρήστη πελάτη είναι ένα javascript αρχείο. Η λειτουργία της αυτορρύθμισης μέσω DNS απλώς μας επιστρέφει το όνομα του διακομιστή στον οποίο βρίσκεται αυτό το αρχείο.

Για το σχολικό δίκτυο αυτό το αρχείο θα πρέπει να βρίσκεται σε έναν web server, ο οποίος θα έχει όνομα wpad.sch.gr. Για την περίπτωση μας μπορεί είτε να δημιουργηθεί ένας τέτοιος web server ή για εξοικονόμηση πόρων μπορεί να εφαρμοστεί σε έναν υπάρχοντα web server με τη μέθοδο alias. Δηλαδή ένας web server που υπάρχει ήδη εξυπηρετεί και αυτά τα αιτήματα.

Το μόνο απαιτούμενο από τον web server είναι η διαδρομή που βρίσκεται το αρχείο αυτορρύθμισης να έχει οριστεί να εξυπηρετεί αρχεία MIME type application/x-ns-proxy-autoconfig.

Ο web server πρέπει να έχει το όνομα wpad.sch.gr και το αρχείο να βρίσκεται στη διαδρομή http://wpad.sch.gr/wpad.dat.

Για τον διακομιστή ιστού αυτό θα πρέπει να οριστούν πολιτικές εντός του δικτύου, οι οποίες θα επιτρέπουν την πρόσβαση στον διακομιστή ιστού αυτό για το πρωτόκολλο http και για την πόρτα 80 χωρίς περιορισμούς και ανοιχτή πρόσβαση από όλα τα σημεία του Ελληνικού Σχολικού Δικτύου.

4.6 ΔΙΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΟΥΣ ΔΙΑΚΟΜΙΣΤΕΣ ΜΕΣΟΛΑΒΗΣΗΣ

Στην παρούσα φάση λειτουργίας το Πανελλήνιο Σχολικό Δίκτυο χρησιμοποίει έναν τρόπο κατανομής της κίνησης στους διακομιστές μεσολάβησης. Ο τρόπος αυτός είναι μέσω του πρωτόκολλου WCCPv2 της CISCO το οποίο κάνει τη δρομολόγηση σύμφωνα με ένα το τελευταίο bit της διεύθυνσης προορισμού.

Αυτή η λειτουργία εν μέρει μας εξασφαλίζει την τοπικότητα του περιεχομένου, καθώς σύμφωνα με τον προορισμό το αίτημα μας θα κατευθυνθεί στον ίδιο διακομιστή μεσολάβησης, οπότε υπάρχουν περισσότερες πιθανότητες να υπάρχει εκεί cached το περιεχόμενο. Εν μέρει, αναφέρθηκε καθώς πολλοί διακομιστές πλέον χρησιμοποιούν και αυτοί load balancing με βάση την IP , οπότε μπορεί ένα αίτημα http για τον ίδιο τελικό προορισμό μπορεί τελικά να πάει σε διαφορετικούς διακομιστές μεσολάβησης με

αποτέλεσμα και αυξανόμενη κίνηση προς το internet (οπότε και μεγαλύτερους χρόνους απόκρισης) και επιπλέον φόρτο εργασίας για τους διακομιστές μεσολάβησης.

Το πρόβλημα με αυτές τις δρομολογήσεις και κατανομή κίνησης είναι ότι μπορούν να λειτουργήσουν μόνο σε περιπτώσεις που ο διακομιστής μεσολάβησης είναι σε διαφανή λειτουργία μεσολάβησης και δεν υπάρχει κρυπτογράφηση μεταξύ χρήστη-πελάτη και διακομιστή μεσολάβησης.

Στην εφαρμογή αυτής της διπλωματικής, οι τρόποι κατανομής της κίνησης δεν μπορούν να λειτουργήσουν καθώς ο χρήστης-πελάτης πρέπει να έχει δηλωμένο τον διακομιστή μεσολάβησης, και ειδικά στην περίπτωση της κρυπτογράφησης μεταξύ χρήστη πελάτη και διακομιστή μεσολάβησης, ο χρήστης-πελάτης περιμένει και μπορεί να επικοινωνήσει μόνο με τον διακομιστή μεσολάβησης ο οποίος είναι ρητά ρυθμισμένος (έστω και για συγκεκριμένους προορισμούς).

Η διανομή κίνησης στη δικιά μας περίπτωση μπορεί να γίνει με τους εξής 2 τρόπους:

- Load Balancing μέσω του PAC αρχείου
- Load Balancing από ένα στοιχείο του δικτύου που υποστηρίζει Layer 7 load balancing.

Φυσικά για κάθε σενάριο λειτουργίας του squid server δεν μπορούν να λειτουργήσουν σωστά και οι 2 τρόποι εκτός από κάποιες εξαιρέσεις, που θα παρουσιαστούν παρακάτω.

4.6.1 Διανομή κίνησης μέσω του PAC αρχείου

Από ότι παρατηρήθηκε πριν το αρχείο pac έχει αρκετές συναρτήσεις και από τη στιγμή που είναι γραμμένο σε javascript είναι δυνατόν να προστεθούν αρκετοί τρόποι χειρισμού του τρόπου και της σειράς προτίμησης των διακομιστών μεσολάβησης. Δυστυχώς όμως, το αρχείο PAC δεν μπορεί να γνωρίζει, όπως ένας αποκλειστικός load balancer, στοιχεία του δικτύου όπως πχ ποιος διακομιστής μεσολάβησης έχει τις λιγότερες συνδέσεις ή το μικρότερο φορτίο με αποτέλεσμα η διανομή της κίνησης να μην είναι δυναμική, αλλά να έχει εφαρμοστεί θεωρητικά σύμφωνα με μετρήσεις. Για παράδειγμα, ένας καλός τρόπος διανομής της κίνησης σε 16 διακομιστές μεσολάβησης είναι ανάλογα με την IP διεύθυνση που έχει πάρει ο χρήστης-πελάτης να εφαρμόσουμε μια συνάρτηση modulo16 και να του αναθέσουμε τον κατάλληλο διακομιστή μεσολάβησης.

Θεωρητικά, μέσω του PAC αρχείου, μπορούμε να διαμοιράσουμε την κίνηση σύμφωνα με το hash του URL. Σύμφωνα με τις οδηγίες των προγραμματιστών των browser, για το PAC αρχείο γίνεται αποτίμηση κάθε φορά που καλείται ένα URL και υποστηρίζει τις βασικές αριθμητικές συναρτήσεις της javascript. Δυστυχώς αυτό δεν συμβαίνει στην πράξη όπως θα δούμε στο πιο κάτω αρχείο. Ένα load balancing μέσω του αρχείου αυτορρύθμισης θα μπορούσε να μας εξασφαλίσει τη μέγιστη τοπικότητα δεδομένων που είναι και το ζητούμενο σε έναν διακομιστή μεσολάβησης. Το πιο κάτω αρχείο δημιουργήθηκε με τις βασικές συναρτήσεις που βασίζονται σε javascript και τις έτοιμες συναρτήσεις που είδαμε πιο πάνω και υποστηρίζουν οι browsers.

```
function FindProxyForURL(url, host) {
  var hash = 0;
  if (host.length == 0) hash=0;
  for (i = 0; i < host.length; i++) {
          char = host.charCodeAt(i);
          hash = ((hash<<5)-hash)+char;
          hash = hash & hash; // Convert to 32bit integer
          hash=hash%2
}
if (hash==0)
return "HTTPS cache01.att.scg.gr:443";
if (hash==1)
           return "HTTPS cache02.att.scg.gr:443";

// DEFAULT RULE: All other traffic, use below proxies, in fail-over order.
   return "HTTPS cache04.att.scg.gr:443";

}
```

Στο παραπάνω αρχείο χρησιμοποιείται μια hash function η οποία μετατρέπει το hostname σε έναν ακέραιο αριθμό. Κατόπιν ελέγχεται αυτός ο αριθμός και με modulo 2 (στο παράδειγμα έχουμε 2 διακομιστές μεσολάβησης) κατευθύνεται η κίνηση ανάλογα με το hash του hostname. Αυτό εξασφαλίζει μεγαλύτερη τοπικότητα δεδομένων καθώς κάθε φορά για το ίδιο hostname (όχι IP όπως στο WCCP) κατευθυνόμαστε στον ίδιο διακομιστή μεσολάβησης.

Δυστυχώς, παρόλο που το παραπάνω αρχείο είναι σύμφωνα με τα standards των browsers για τα PAC αρχεία, το συγκεκριμένο αρχείο επέστρεφε πάντα τον cache04. Για το θέμα έχει ανοιχτεί bug report στον Firefox και στον Chrome.

Ένα μειονέκτημα της παραπάνω λύσης είναι ότι ο browser ζητάει τα διαπιστευτήρια τόσες φορές όσους διακομιστές μεσολάβησης έχουμε στο PAC αρχείο, κάτι το οποίο δεν είναι αντιπαραγωγικό για τον τελικό χρήστη.

Ένας άλλος τρόπος διανομής της κίνησης είναι μέσω της διεύθυνσης IP του χρήστη-πελάτη. Σύμφωνα με αυτή τη διανομή κίνησης ο πελάτης λαμβάνει υπόψιν την τελευταίο octet της IP διεύθυνσης του και με ένα modulo16 αποφασίζει ποιον διακομιστή μεσολάβησης θα χρησιμοποιήσει. Σε αυτή την περίπτωση η κίνηση διανέμεται σύμφωνα με το πόσες συσκευές έχουμε συνδεδεμένες, χωρίς όμως να λαμβάνει υπόψιν την κίνηση κάθε συσκευής. Για παράδειγμα θα μπορούσε 10 συσκευές που συνδέονται σε ένα διακομιστή μεσολάβησης να μην έχουν καθόλου κίνηση ενώ 10 άλλες που συνδέονται σε άλλον να έχουν πολύ μεγαλύτερη κίνηση. Σε αυτή την περίπτωση ο φόρτος εργασίας δεν ισοκατανέμεται σε όλους τους διακομιστές και επιπλέον δεν είμαστε σίγουροι για την τοπικότητα του περιεχομένου.

Το αρχείο το οποίο μπορεί να κάνει αυτή την κατανομή κίνησης είναι το εξής :

```
function FindProxyForURL(aFullURL, aHostname)
  {
    if(isPlainHostName(aHostname))
    {
      return "DIRECT";
    }
    if(isInNet(aHostname, "172.16.0.0", "255.255.0.0"))
    {
      return "DIRECT";
    }
  var myIp = myIpAddress();
  var ipBits = myIp.split(".");
  var mySeg = parseInt(ipBits[3]);
  if((mySeg % 2) == 0)          //EVEN
 {
  return "HTTP cache04.att.sch.gr:3128";
 }
 else  //ODD
  {
   return "HTTP cache06.att.sch.gr:3128";;
  }
}
```

Όπως παρατηρείται σε αυτό το αρχείο, ο χρήστης πελάτης συγκρίνει τη διεύθυνση του με ένα modulo2 (εδώ έχουμε 2 διακομιστές μεσολάβησης) και ανάλογα με το που ανήκει ρυθμίζει και σε ποιον διακομιστή μεσολάβησης θα συνδεθεί.

Το θετικό του συγκεκριμένου αρχείου PAC είναι ότι δεν χρειάζεται κάποιο επιπλέον μηχάνημα να κάνει τη διανομή κίνησης και ότι όλα τα request από την ίδια IP θα πηγαίνουν πάντα στον ίδιο διακομιστή μεσολάβησης. Αυτό, ειδικά για τα session που βασίζονται σε

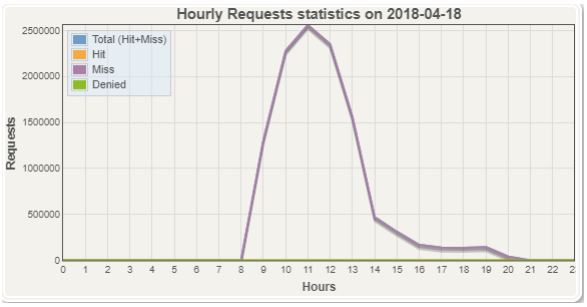
cookies, έχει μεγάλη σημασία καθώς πολλές φορές αν ένα session ξεκινήσει από έναν διακομιστή μεσολάβησης και στο ενδιάμεσο αλλάξει η IP από την οποία προέρχεται στο session, ο τελικός προορισμός μας αποσυνδέει (πχ ebanking, gmail κα).

4.6.2 Διανομή κίνησης μέσω dedicated load balancer και αποφόρτιση του TLS termination

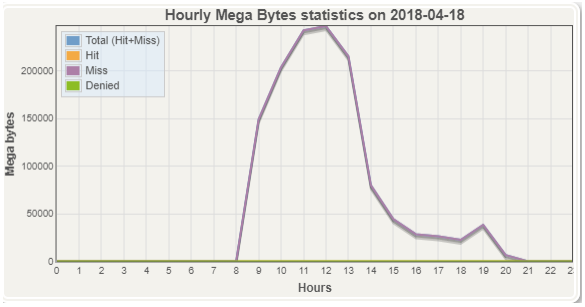
Για να επιτευχθούν σωστά οι λειτουργίες διανομής της κίνησης, θα πρέπει να χρησιμοποιηθεί μια συσκευή δικτύου η οποία να πραγματοποιεί αυτή την εργασία. Η συσκευή αυτή θα πρέπει να είναι ικανή να κάνει layer 7 δρομολόγηση καθώς και τερματισμό του TLS (TLS termination).

Η δρομολόγηση σε layer 7 είναι απαραίτητη καθώς θα πρέπει ο load balancer να μπορεί να εξάγει συμπεράσματα και να δρομολογεί την κίνηση σύμφωνα με τις κεφαλίδες του http αιτήματος. Αντιθέτως με τους περισσότερους load balancers που το κάνουν στο layer 4, θα χρειαστούμε load balancing στο layer 7, καθώς από το layer 4 δεν μπορούμε να εξάγουμε συμπεράσματα για το που κατευθύνεται η κίνηση και άλλες παράμετροι όπως τα sessions που έχουν ξεκινήσει μερικοί χρήστες-πελάτες. Στο Πανελλήνιο Σχολικό Δίκτυο κάθε χρήστης-πελάτης βρίσκεται πίσω από NAT, οπότε η δρομολόγηση κίνησης μπορεί να γίνει μόνο ανά Σχολείο ή συστάδα Σχολείων (ανάλογα πως έχει ρυθμιστεί το δίκτυο κάθε σχολείου). Αυτό καθιστά πολύ δύσκολο το load balancing στο layer 4 και για να γίνει load balancing με αυτά τα δεδομένα στο layer 4, τότε θα πρέπει να βασιστούμε στη δρομολόγηση κάθε ολόκληρης μονάδας. Επειδή η κίνηση είναι εντελώς δυναμική, αυτό μπορεί να αποδώσει μεγάλη κίνηση σε έναν διακομιστή μεσολάβησης και πολύ λιγότερη σε κάποιον άλλο. Επιπλέον, επειδή η διανομή της κίνησης δεν είναι δυναμική, υπάρχει μεγάλη περίπτωση ένας διακομιστής μεσολάβησης να βγει εκτός, λόγω του μεγάλου φορτίου, και κάποιος άλλος να μην έχει καθόλου φορτίο. Με αυτή τη δρομολόγηση το δίκτυο δεν είναι ευέλικτο σε αλλαγές που μπορεί να γίνουν στη δομή του δικτύου με αποτέλεσμα το αυξημένο κόστος συντήρησης του δικτύου.

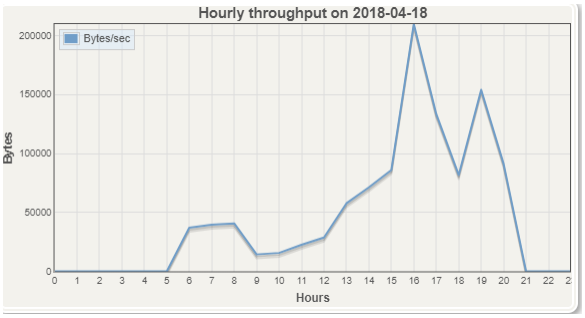
Ένα παράδειγμα το οποίο δεν διανέμει σωστά το φορτίο και δεν χρησιμοποιεί σωστές ρυθμίσεις για την αποφόρτιση διακομιστών παρουσιάζεται στο επόμενο διάγραμμα. Στο διάγραμμα αυτό βλέπουμε τον διακομιστή μεσολάβησης με το μεγαλύτερο φορτίο που λειτουργεί αυτή τη στιγμή στο ΠΣΔ :



Εικόνα 23: Φορτίο αιτημάτων διακομιστή μεσολάβησης



Εικόνα 24: Φορτίο MB/ώρα διακομιστή μεσολάβησης



Εικόνα 25: Throughput/ώρα διακομιστή μεσολάβησης

Από τα παραπάνω γραφήματα έχουμε ένα μεγάλο φορτίο γύρω στις 11 η ώρα το πρωί. Μετά από ανάλυση αυτό το φορτίο προέρχεται από ενημερώσεις των windows, οι οποίες άρχισαν να γίνονται ταυτόχρονα σε όλους τους υπολογιστές του ΠΣΔ. Ο τρόπος που γίνεται τώρα η κατανομή της κίνησης έχει σχέση με το τελευταίο bit του τελικού προορισμού. Επειδή και η Microsoft χρησιμοποιεί μια IP για όλες τις υπηρεσίες ενημέρωσης και κάνει load balancing σε αυτή την IP. Λόγω της δικιάς μας δρομολόγησης όλες οι ενημερώσεις κατευθύνονται στον ίδιο διακομιστή μεσολάβησης, αυξάνοντας δραματικά το φορτίο και συνήθως την ίδια ώρα. Επιπλέον επιβαρυντικό παράγοντα, αποτελεί το ότι ο squid έτσι όπως είναι ρυθμισμένος δεν αποθηκεύει τοπικά τα αρχεία των

ενημερώσεων που κατεβάζει, με αποτέλεσμα κάθε φορά την ίδια ενημέρωση να την ξανακατεβάζει όλη από την αρχή. Επίσης, συμπεραίνεται ότι όταν μειώνονται οι συνδέσεις αυξάνεται πάρα πολύ η διεκπαιρεότητα (throughput).

Μια λύση σε αυτό είναι να αποθηκεύονται τοπικά στον squid όλα τα αρχεία των ενημερώσεων των windows για να μειώσουμε δραματικά τόσο τον αριθμό των συνδέσεων όσο και την καθυστέρηση για να φέρει το αρχείο πολλαπλές φορές ο διακομιστής μεσολάβησης. Από την ανάλυση κίνησης προέκυψε ότι για το ίδιο ακριβώς αρχείο αναβάθμισης των windows, ο διακομιστής μεσολάβησης δεν βρήκε κανένα τοπικά αποθηκευμένο αρχείο σε 49 αιτήματα που του ζητήθηκε. Το αρχείο αυτό έχει μέγεθος 3,3GB και είναι η ενημέρωση των windows με την ονομασία

16299.125.171213-1220.rs3_release_svc_refresh_clientconsumer_ret_x64fre_el-gr_3ee79b0356050e4249e0c3451e61af8ea14879a4.esd .

Από την ανάλυση των αρχείων καταγραφών (log files) του διακομιστή μεσολάβησης προέκυψε ότι ο διακομιστής μεσολάβησης χρειάστηκε να φέρει από τους διακομιστές της Microsoft 150GB μόνο για την συγκεκριμένη ενημέρωση. Τάξη μεγέθους περίπου 46 φορές περισσότερο από το αναμενόμενο. Επιπλέον, ο διακομιστής για αυτή την ενημέρωση χρειάστηκε 515.022 αιτήματα σε μια μέρα εκ των οποίων τα περισσότερα έγιναν σε ώρες αιχμής για το Πανελλήνιο Σχολικό δίκτυο. Η δυσλειτουργία αυτή έχει ως αποτέλεσμα ο διακομιστής μεσολάβησης ενώ θα έπρεπε να λαμβάνει πολύ λιγότερα αιτήματα, να λαμβάνει πολύ περισσότερα καθώς είναι σε υψηλό φόρτο εργασίας και πολλά από αυτά τα αιτήματα να λήγουν ή/και να μην μπορούν να εξυπηρετηθούν. Ως αποτέλεσμα αυξάνονται πάρα πολύ τα αιτήματα στον διακομιστή μεσολάβησης σε μια ουρά την οποία δεν μπορεί να εξυπηρετήσει καθώς συσσωρεύονται και άλλα αιτήματα. Η δυσλειτουργία αυτή δημιουργεί επίσης και περισσότερες συνδέσεις από τον διακομιστή μεσολάβησης προς τους διακομιστές της Microsoft μειώνοντας δραματικά το bandwidth αφού κατεβάζει το ίδιο αρχείο πολλές φορές.

Η λύση σε αυτό το πρόβλημα βρίσκεται σε 2 αλλαγές που πρέπει να γίνουν στο υπάρχον δίκτυο. Η μια είναι στον ίδιο τον squid server και η άλλη είναι στην πολιτική διανομής κίνησης.

4.6.2.1 Αλλαγές στον Squid Server για αποθήκευση περιεχομένου windows update

Ανάλογα με την τελική ρύθμιση του squid server (αν θα χρησιμοποιηθεί TLS encryption μεταξύ χρήστη-πελάτη και διακομιστή μεσολάβησης), πρέπει να γίνουν και οι κατάλληλες αλλαγές στο αρχείο ρύθμισης του squid server. Δυστυχώς, η Microsoft, μέχρι και τη συγγραφή αυτή της διπλωματικής εργασίας, δεν υποστηρίζει ακόμα κρυπτογράφηση προς το διακομιστή μεσολάβησης, οπότε θα πρέπει να ληφθεί μέριμνα για τις ενημερώσεις των windows τόσο στους διακομιστές μεσολάβησης όσο και στο PAC αρχείο αυτορρύθμισης.

Οι αλλαγές στον διακομιστή μεσολάβησης squid μπορούν να γίνουν με την προσθήκη των παρακάτω directives στο αρχείο ρύθμισης :

```
acl windowsupdate dstdomain windowsupdate.microsoft.com
acl windowsupdate dstdomain .update.microsoft.com
acl windowsupdate dstdomain redir.metaservices.microsoft.com
acl windowsupdate dstdomain images.metaservices.microsoft.com
acl windowsupdate dstdomain c.microsoft.com
acl windowsupdate dstdomain .windowsupdate.com
acl windowsupdate dstdomain wustat.windows.com
acl windowsupdate dstdomain crl.microsoft.com
acl windowsupdate dstdomain sls.microsoft.com
acl windowsupdate dstdomain productactivation.one.microsoft.com
acl windowsupdate dstdomain ntservicepack.microsoft.com
acl windowsupdate dstdomain .delivery.mp.microsoft.com
range_offset_limit 5 GB windowsupdate
maximum_object_size 5 GB
quick_abort_min -1

refresh_pattern -i microsoft.com/.*\.(cab|exe|ms[i|u|f]|[ap]sf|wm[v|a]|dat|zip) 4320 80% 43200
reload-into-ims
refresh_pattern -i windowsupdate.com/.*\.(cab|exe|ms[i|u|f]|[ap]sf|wm[v|a]|dat|zip) 4320 80%
43200 reload-into-ims

http_access allow windowsupdate all
```

Τα directives αυτά περιλαμβάνουν όλες τις τοποθεσίες που χρησιμοποιεί η Microsoft για ενημερώσεις για την οικογένεια προϊόντων της. Η Microsoft από τα windows 8 και έπειτα περιλαμβάνει ενημερώσεις οι οποίες φτάνουν μέχρι τα 5GB, οπότε τίθεται το αντικείμενο maximum_object_size στα 5GB. Επίσης, πρέπει να αποθηκεύουν τοπικά τα αντικείμενα που έχουν offset όσο και το μέγεθος του αντικειμένου με την εντολή range_offset_limit. Εκτός αυτού, ενημερώνεται ο squid ότι μόνο για τα ονόματα τομέα των

ενημερώσεων των windows να κρατάει αποθηκευμένα τοπικά τα αντικείμενα τουλάχιστον για 4320 ώρες (180 μέρες).

Τελικά, επιτρέπεται η πρόσβαση σε όλους τους χρήστες χωρίς αυθεντικοποίηση (η Microsoft όπως αναφέραμε δεν υποστηρίζει ακόμα κρυπτογράφηση προς τον διακομιστή μεσολάβησης).

Το συνιστώμενο μέγεθος τοπικής αποθήκευσης για τον διακομιστή μεσολάβησης που θα διαχειρίζεται τις ενημερώσεις εξαρτάται καθαρά από τον αριθμό διαφορετικών οικογενειών προϊόντων που υπάρχουν στο δίκτυο. Ένα μέγεθος της τάξης των 30-60GB κρίνεται αρκετά ικανοποιητικό και θα αποσυμφορήσει αρκετά τον διακομιστή μεσολάβησης.

Για να ενημερωθούν οι χρήστες πελάτες να χρησιμοποιούν για τις ενημερώσεις των windows το συγκεκριμένο διακομιστή μεσολάβησης, πρέπει επίσης να προσθέσουμε στην αρχή του αρχείου αυτορρύθμισης τα εξής :

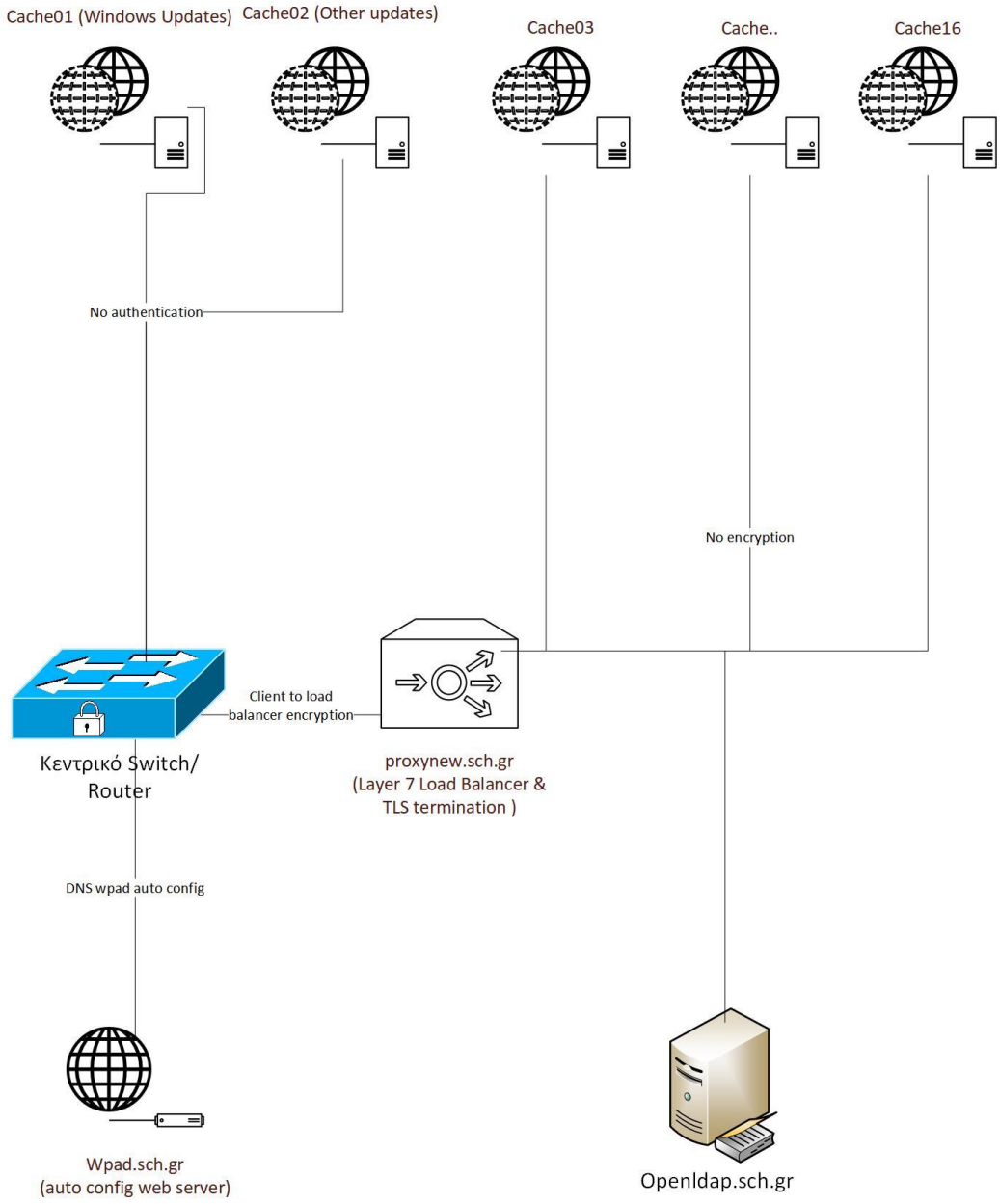
```
if   ((shExpMatch(host, "*.microsoft.com"))||(shExpMatch(host,
"*.windowsupdate.com"))||(shExpMatch(host, "*.windows.com")))
     return "PROXY cachexx.att.sch.gr:3128; " ;
```

Πλέον, όλοι οι χρήστες-πελάτες θα χρησιμοποιούν το συγκεκριμένο διακομιστή μεσολάβησης για να λαμβάνουν τα windows updates χωρίς αυθεντικοποίηση και χωρίς κρυπτογράφηση προς τον διακομιστή μεσολάβησης. Παρόλο που όλοι οι χρήστες-πελάτες θα χρησιμοποιούν αυτό το διακομιστή μεσολάβησης για τα windows update, θα παρατηρηθεί ότι το φορτίο του θα μειωθεί καθώς πλέον θα έχει αποθηκευμένα τοπικά τα αρχεία ενημερώσεων των windows.

Επειδή εκτός από τα windows υπάρχουν και άλλα λειτουργικά ή υπηρεσίες που απαιτούν ενημερώσεις (linux, antivirus κ.α.), προτείνεται η χρήση δυο αποκλειστικών (dedicated) μηχανημάτων για αυτές τις λειτουργίες. Ένα για τις ενημερώσεις των windows και ένα για όλες τις άλλες ενημερώσεις άλλων κατασκευαστών.

4.6.2.2 Ρυθμίσεις διανομής κίνησης στον load balancer

Στο παρακάτω διάγραμμα φαίνεται η προτεινόμενη τοπολογία δικτύου για λειτουργία κρυπτογραφημένης κίνησης μεταξύ χρήστη-πελάτη και load balancer, και μη κρυπτογραφημένης μεταξύ load balancer και διακομιστή μεσολάβησης.



Εικόνα 26: Προτεινόμενη τοπολογία δικτύου

Όπως παρατηρήθηκε πριν, το όνομα χρήστη και το συνθηματικό μεταφέρονται σε κάθε http αίτημα στις κεφαλίδες, οπότε για να ασφαλιστούν τα συνθηματικά επιλέγουμε την σύνδεση με τον διακομιστή μεσολάβησης μέσω TLS. Για να αποφορτιστούν οι διακομιστές μεσολάβησης από την κρυπτογράφηση/αποκρυπτογράφηση τερματίζεται η σύνδεση στο σημείο του load balancer. Αφού η σύνδεση τερματίζεται στον load balancer, ο load balancer πρέπει να έχει εγκατεστημένο το πιστοποιητικό που είδαμε πιο πριν για το όνομα proxynew.sch.gr. Ο load balancer δρομολογεί την κίνηση σύμφωνα με το hash του URL το οποίο ζητάει ο χρήστης-πελάτης, οπότε κάθε φορά που ζητείται κάποιο URL ξέρουμε ότι θα πηγαίνει πάντα στον ίδιο διακομιστή μεσολάβησης.

Ο τρόπος διανομής της κίνησης αυτής αφορά καθαρά το http πρωτόκολλο καθώς στο https δεν είναι δυνατή η πρόσβαση στο περιεχόμενο. Για το πρωτόκολλο https θα πρέπει να διανέμουμε την κίνηση στους διακομιστές μεσολάβησης σύμφωνα με τις λιγότερες συνδέσεις (least connections) ώστε να διανέμεται η κίνηση όσο γίνεται πιο ομοιόμορφα. Κάτι που πρέπει να ληφθεί υπόψιν είναι ότι ο load balancer θα πρέπει να κρατάει τα sessions που έχουν δημιουργηθεί πάντα προς τον ίδιο διακομιστή μεσολάβησης, αλλιώς ο χρήστης πελάτης θα αποσυνδέεται από τις υπηρεσίες που χρησιμοποιεί αν αλλάζει η IP με την οποία συνδέεται στον τελικό προορισμό. Στην ορολογία των διακομιστών μεσολάβησης και load balancers αυτά ονομάζονται sticky sessions. Ο load balancer είναι απαραίτητο να γνωρίζει ποιο session χρήστη-πελάτη πάει σε ποιον διακομιστή μεσολάβησης. Επειδή όλοι οι χρήστες πελάτες βρίσκονται πίσω από NAT, στη δικιά μας περίπτωση δεν μπορούμε να εφαρμόσουμε τους απλούς κανόνες σύμφωνα με την IP προορισμού, αλλά θα χρησιμοποιήσουμε με πιο προχωρημένη λειτουργία των load balancers, όπως θα δούμε πιο κάτω.

5 ΣΤΟΙΧΕΙΑ ΔΙΚΤΥΟΥ, ΡΥΘΜΙΣΕΙΣ ΚΑΙ ΜΕΤΡΗΣΕΙΣ ΑΠΟΔΟΣΗΣ

Για την δοκιμαστική λειτουργία του δικτύου, ανάλυση και μετρήσεις χρησιμοποιήθηκαν τα εξής προγράμματα:

- HAProxy (σαν load balancer με TLS termination)
- Apache Jmeter (πλατφόρμα δημιουργίας κίνησης και ανάλυσης προς τους διακομιστές μεσολάβησης)
- Web Polygraph (πλατφόρμα παραγωγής μεγάλης κίνησης με robot για το stress test στοιχείων δικτύου)
- Squid-Analyzer (πρόγραμμα ανάγνωσης και εξαγωγής αναφορών για τη λειτουργία της cache του squid)
- Nginx Web Server (για το PAC αρχείο αυτορρύθμισης wpad.dat)
- Certbot (εργαλείο αυτόματης δημιουργίας πιστοποιητικού PKI)
- OpenSSL
- Node-Exporter
- Prometheus
- Grafana

5.1 LOAD BALANCER HAPROXY

Ο HAProxy, τα αρχικά του οποίου σημαίνουν High Availability Proxy, είναι ένα πολύ δημοφιλές λογισμικό ανοιχτού κώδικα για TCP/HTTP load balancing και proxying το οποίο μπορεί να τρέξει σε όλες τις πλατφόρμες linux, Solaris και FreeBSD. Η πιο κοινή του χρήση είναι για να βελτιώσει την απόδοση και την αξιοπιστία σε περιβάλλοντα διακομιστών διανέμοντας το φορτίο σε πολλούς διακομιστές (web, application, database). Χρησιμοποιείται σε πολλά δίκτυα υψηλής ζήτησης όπως: GitHub, Imgur, Instagram και Twitter.

Μερικά χαρακτηριστικά και μετρήσεις του HAProxy τα οποία μετρήθηκαν σε ένα σύστημα με έναν Core i7 στα 3.7 GHz, με dual-port 10 Gbps NICs, Linux kernel 3.10, HAProxy 1.6 και OpenSSL 1.0.2.

Ο HAProxy χρησιμοποιούσε ένα process σε έναν αποκλειστικό πυρήνα της CPU, και δύο έξτρα πυρήνες χρησιμοποιήθηκαν για τις διακοπές δικτύου (network interrupts):

- 20 Gbps μέγιστης διακπαιρεότητας σε clear text για αντικείμενα 256 kB ή μεγαλύτερα.
- 4.6 Gbps TLS κίνησης με χρησιμοποίηση του AES256-GCM cipher με μεγάλα αντικείμενα.
- 83000 TCP συνδέσεις ανά δευτερόλεπτο από πελάτη προς τον διακομιστή.
- 82000 HTTP συνδέσεις ανά δευτερόλεπτο από πελάτη προς τον διακομιστή.
- 300000 φιλτραρισμένες TCP συνδέσεις ανά δευτερόλεπτο (anti-DDoS).
- Περίπου 5000 ταυτόχρονες TLS συνδέσεις από άκρο σε άκρο (από τη μεριά του πελάτη μόνο) ανά GB της RAM, συμπεριλαμβανόμενης και της μνήμης που απαιτείται για buffers τους συστήματος
- Περίπου 5000 ταυτόχρονες TLS συνδέσεις από άκρο σε άκρο (και από τις 2 μεριές) ανά GB της RAM, συμπεριλαμβανόμενης και της μνήμης που απαιτείται για buffers τους συστήματος

Στο ίδιο μηχάνημά, ο HAProxy μπορεί να εξυπηρετήσει :

- 5-10 caching διακομιστές μεσολάβησης
- 100 anti-virus διακομιστές μεσολάβησης
- 100-1000 διακομιστές για εφαρμογές

Στην επόμενη ενότητα παρουσιάζεται η ορολογία του HAProxy και πως αυτή χρησιμοποιείται για να δημιουργηθεί και να ρυθμιστεί το περιβάλλον δοκιμών για την παρούσα διπλωματική.

5.1.1 Access Control List (ACL)

Τα ACLs (όπως και στον squid), χρησιμοποιούνται για να ελέγξουν μερικές συνθήκες και σύμφωνα με αυτές να εφαρμόσουν κάποια εντολή (πχ. να διαλέξουν έναν διακομιστή ή να αρνηθούν ένα αίτημα) βασιζόμενα στο αποτέλεσμα του ελέγχου. Η χρησιμοποίηση των ACLs μας επιτρέπει την εύκολη προώθηση της δικτυακής κίνησης βασιζόμενη σε μια πλειάδα παραγόντων όπως ταίριασμα προτύπων, τον αριθμό των συνδέσεων προς τους διακομιστές κ.α.

Παράδειγμα μιας ACL:

```
acl url_blog path_beg /blog
```

Αυτή η ACL ενεργοποιείται αν το path ενός αιτήματος χρήστη-πελάτη ξεκινάει με το /blog. Για παράδειγμα θα ταίριαζε στο *http:/www.sch.gr/blog/xxxxxxxxxxx*.

5.1.2 Backend

Το backend είναι μια συστάδα διακομιστών η οποία λαμβάνει προωθούμενες αιτήσεις από τον HAProxy. Τα Backends ορίζονται στο τμήμα *backend* της ρύθμισης του HAProxy. Στην πιο βασική του ρύθμιση ένα backend μπορεί να έχει:

- Ποιον αλγόριθμο load balance να χρησιμοποιήσει
- Μια λίστα με διακομιστές και πόρτες

Ένα απλό παράδειγμα με δυο backend, ένα *web-backend* και ένα *blog-backend* με δυο διακομιστές web το καθένα, και δουλεύουν στην πόρτα 80:

```
backend web-backend
   balance roundrobin
   server web1 web1.yourdomain.com:80 check
   server web2 web2.yourdomain.com:80 check

backend blog-backend
   balance roundrobin
   mode http
   server blog1 blog1.yourdomain.com:80 check
   server blog1 blog1.yourdomain.com:80 check
```

Το `mode http` διευκρινίζει ότι θα χρησιμοποιηθεί layer 7 proxying (αντίθετα με το mode tcp που χρησιμοποιεί layer 4).

Το `check` στο τέλος της δήλωσης server ορίζει ότι σε αυτούς τους διακομιστές θα πραγματοποιείται ένας έλεγχος λειτουργίας (health check) ώστε να είναι γνωστό ποιοι είναι λειτουργικοί, ενώ αυτοί που δεν λειτουργούν να βγαίνουν αυτόματα από το load balancing.

5.1.3 Frontend

Το frontend ορίζει πως τα αιτήματα θα προωθούνται στα backend. Τα Frontend ορίζονται στο τμήμα *frontend* της ρύθμισης του HAProxy. Οι ορισμοί τους αποτελούνται από τα εξής τμήματα :

- Μια ομάδα από διευθύνσεις IP (πχ 10.1.1.7:80, *:443)
- ACLs
- *use_backend* κανόνες, που μπορούν να ορίσουν ποια backend θα χρησιμοποιηθούν σύμφωνα με τις συνθήκες ACL η/και κάθε άλλου κανόνα που ορίζεται στο *default_backend*.

Ένα frontend μπορεί να ρυθμιστεί για διάφορους τύπους δικτυακής κίνησης.

5.1.4 Είδη Load Balancing με TLS termination

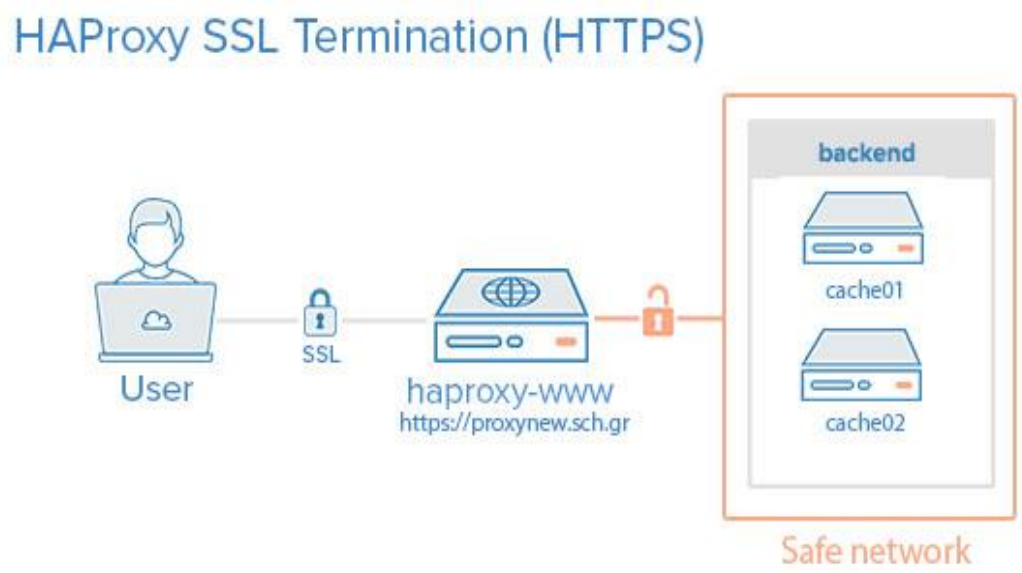
Ο HAProxy υποστηρίζει TLS termination χωρίς load balancing, layer 4 load balancing και layer 7 load balancing. Σε κάθε mode μπορεί να λειτουργεί είτε με TLS termination είτε χωρίς. Βασικό προαπαιτούμενο για το TLS termination όπως διαπιστώθηκε και σε προηγούμενο κεφάλαιο είναι ένα πιστοποιητικό από αξιόπιστη πηγή.

Στο επόμενο σχεδιάγραμμα παρουσιάζεται η λειτουργία TLS termination του HAProxy χωρίς λειτουργία load balancing.

Εικόνα 27: HAProxy TLS Termination

Στο παραπάνω παράδειγμα ο HAProxy λειτουργεί καθαρά σε layer 4 με TLS termination.

Με τον ίδιο ακριβώς τρόπο θα μπορούσε να δουλέψει σε layer 4 load balancing με TLS termination όπως παρακάτω.

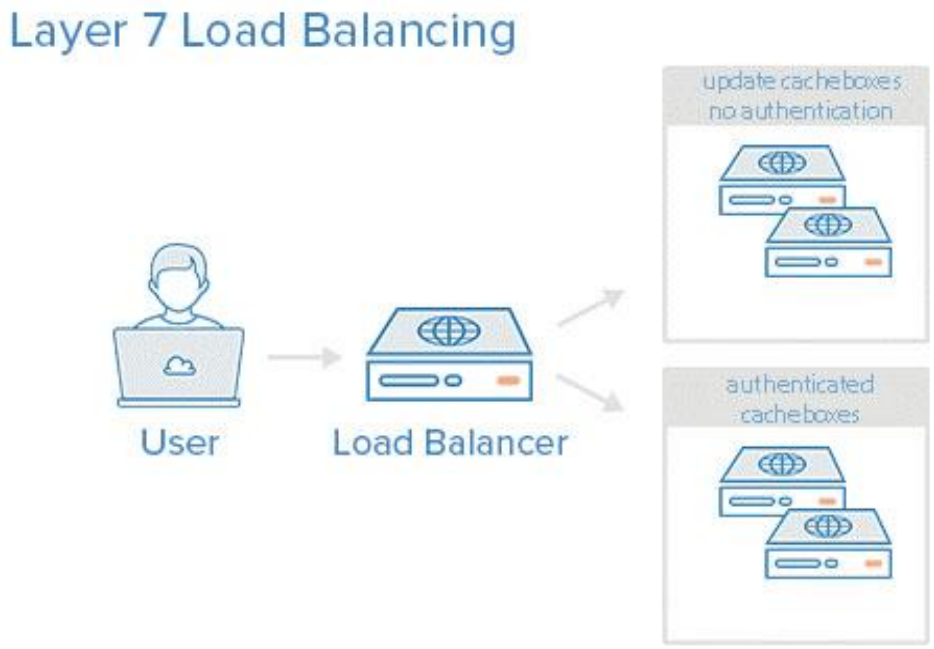


Εικόνα 28: HAProxy Layer 4 Load Balancing

Σε αυτό το παράδειγμα το load balancing μπορεί να λειτουργήσει μόνο σε συμπεράσματα τα οποία εξάγονται από το TCP πρωτόκολλο, και είναι η IP πηγής και προορισμού. Όπως διαπιστώθηκε παραπάνω αυτό δεν μας αποσυμφορεί πραγματικά το φορτίο και μπορεί να αυξηθεί πολύ σε μερικούς διακομιστές μεσολάβησης.

Το σενάριο που προτείνεται στη δικιά μας περίπτωση είναι το layer 7 load balancing βασιζόμενο στο URL hash στο πρωτόκολλο http και στο least connections για το πρωτόκολλο https. Για το πρωτόκολλο https, όμως θα κρατάμε sticky sessions όπως είδαμε πιο πριν.

Στη περίπτωση μας θα χρησιμοποιήσουμε TLS termination για τη σύνδεση με τους διακομιστές μεσολάβησης, οι οποίοι απαιτούν αυθεντικοποίηση και απλή σύνδεση για τους διακομιστές μεσολάβησης, οι οποίοι είναι υπεύθυνοι για τα updates και δεν χρειάζονται αυθεντικοποίηση (μόνο για τα updates).



Εικόνα 29: HAPRoxy Layer 7 Load Balancing

Σε αυτό το παράδειγμα ο χρήστης-πελάτης, μέσω του αρχείου αυτορρύθμισης για τα ονόματα τομέα που χρησιμοποιούνται για τις ενημερώσεις των windows, έχει σαν δηλωμένο διακομιστή μεσολάβησης τον load balancer με την πόρτα 3129. Κάθε αίτημα στην πόρτα 3129 προωθείται στους 2 αποκλειστικούς διακομιστές μεσολάβησης που είναι για τα windows update. Αυτοί οι διακομιστές μεσολάβησης επιτρέπουν τη μη αυθεντικοποιημένη χρήση τους μόνο για τα ονόματα τομέα που ρυθμίσαμε και αφορούν τις ενημερώσεις των windows. Για όλα τα υπόλοιπα μπορούμε είτε να απαιτήσουμε αυθεντικοποίηση, είτε να μην επιτρέπεται καθόλου η χρήση τους για άλλα αιτήματα που δεν αφορούν τις υπηρεσίες ενημέρωσής που έχουμε ρητά επιτρέψει. Προτείνεται η απαγόρευση οποιασδήποτε άλλης κίνησης, εκτός των ενημερώσεων, καθώς εάν χρειαστεί αυθεντικοποίηση στους συγκεκριμένους διακομιστές μεσολάβησης τα διαπιστευτήρια του χρήστη θα μετακινηθούν καθ' όλη τη διαδρομή σαν plain text (καθώς τα windows δεν υποστηρίζουν ακόμα κρυπτογράφηση μεταξύ του χρήστη πελάτη και του διακομιστή μεσολάβησης). Για τους 2 αυτούς διακομιστές μεσολάβησης ο load balancer κατανέμει την κίνηση σύμφωνα με το URL hash. Έτσι επιτυγχάνουμε και ισοκατανομή της κίνησης και κάθε φορά για το ίδιο URL θα πηγαίνουμε στον ίδιο διακομιστή μεσολάβησης. Οπότε αν έχει ζητήσει άλλος χρήστης πριν το ίδιο περιεχόμενο θα εντοπιστεί σίγουρα.

Αντίστοιχα για το backend των διακομιστών μεσολάβησης (οι οποίοι χρησιμοποιούν αυθεντικοποίηση) ο χρήστης πελάτης δημιουργεί μια κρυπτογραφημένη σύνδεση με τον load balancer στην πόρτα 3128. Η πόρτα αυτή στον load balancer αντιστοιχεί με το backend το οποίο απαιτεί αυθεντικοποίηση. Η διανομή κίνησης στο συγκεκριμένο backend γίνεται με 2 τρόπους :

- Αν ο χρήστης πελάτης έχει ζητήσει υπηρεσία τελικού προορισμού http, τότε η διανομή κίνησης γίνεται σύμφωνα με το hash ολόκληρου του URL. Στο πρωτόκολλο http μπορούμε να κάνουμε αποθηκεύσουμε τοπικά το περιεχόμενο, οπότε με το hash του URL ισοκατανέμουμε την κίνηση και για το ίδιο αντικείμενο ξέρουμε ότι θα καταλήξουμε στον ίδιο διακομιστή μεσολάβησης. Έτσι αν έχει ζητηθεί το αντικείμενο από κάποιον άλλο χρήστη, σίγουρα θα προωθηθούμε στον διακομιστή μεσολάβησης που έχει τοπικά αποθηκευμένο αυτό το περιεχόμενο.
- Αν ο χρήστης πελάτης έχει ζητήσει υπηρεσία τελικού προορισμού https, τότε δεν μπορούμε να αποθηκεύσουμε τοπικά το περιεχόμενο και ουσιαστικά απλώς είμαστε ένας ενδιάμεσος κόμβος, ο οποίος αναλαμβάνει την επικοινωνία μεταξύ χρήστη πελάτη και τελικού προορισμού. Αφού δεν μπορούμε να αποθηκεύσουμε το περιεχόμενο, δεν υπάρχει λόγος να κατευθυνόμαστε σε συγκεκριμένο διακομιστή μεσολάβησης, αλλά αυτή τη φορά επιλέγουμε τον διακομιστή μεσολάβησης προς τον οποίο υπάρχουν οι λιγότερες συνδέσεις και άρα έχει και το μικρότερο φορτίο. Επειδή, όπως αναφέρθηκε και πριν υπάρχουν και υπηρεσίες οι οποίες χρειάζονται sticky sessions, ο load balancer σε κάθε αίτημα που προωθεί εισάγει ένα δικό του cookie με το όνομα του διακομιστή μεσολάβησης που θα εξυπηρετήσει το αίτημα. Με τον τρόπο αυτό όταν αιτήματα είναι συνέχεια προηγουμένων (όπως σε μια υπηρεσία https που απαιτεί sticky sessions) δεν χρησιμοποιείται η πολιτική διανομής φορτίου σύμφωνα με τις συνδέσεις, αλλά σύμφωνα με το cookie το αίτημα προωθείται στον διακομιστή μεσολάβησης που αρχικά ξεκίνησε την σύνδεση, οπότε και το sticky session θα διατηρηθεί.

Σύμφωνα με αυτά τα ζητούμενα η ρύθμιση του HAProxy η οποία πετυχαίνει τα προαναφερθέντα περιλαμβάνεται στο Παράρτημα 7.3.

Με την προηγούμενη ρύθμιση δημιουργούνται 2 frontend. Ένα για την πόρτα 3129 (μη κρυπτογραφημένη κίνηση) και ένα για την πόρτα 3128 (κρυπτογραφημένη κίνηση)

Το frontend με την πόρτα 80 διανέμει την κίνηση στο backend που δεν απαιτεί κρυπτογράφηση για τα windows updates.

Το fronted στην πόρτα 3128 διανέμει την κίνηση σε όλους τους άλλους διακομιστές μεσολάβησης που απαιτούν κρυπτογράφηση και είναι για όλο το περιεχόμενο εκτός των updates.

Στο παράρτημα 7.3 περιλαμβάνεται ο κώδικας ο οποίος αυτόματα εγκαθιστά στην διανομή ubuntu server 16.4 τον haproxy server που κάνει αίτηση και παραλαμβάνει τα πιστοποιητικά από την αρχή πιστοποίησης Let's Encrypt και ρυθμίζει τον haproxy να λειτουργήσει ακριβώς όπως διαπιστώθηκε στην προηγούμενη ρύθμιση.

5.2 ΠΛΑΤΦΟΡΜΑ ΔΗΜΙΟΥΡΓΙΑΣ ΦΟΡΤΙΟΥ WEB POLYGRAPH

Το web polygraph είναι ένα δωρεάν εργαλείο ανοιχτού κώδικα για benchmarking για caching proxies, layer 4/7 switches, φίλτρα περιεχομένου, web accelerators, και άλλα ενδιάμεσα στοιχεία δικτύου. Το εργαλείο αυτό έρχεται με μερικά έτοιμα τεστ για Layer 4/7 switches και caching proxies. Αυτά όμως πρέπει κάθε φορά να προσαρμοστούν στις συνθήκες του εκάστοτε περιβάλλοντος.

Στη δοκιμή μας δημιουργήθηκε ένα stress test εξαρχής, λαμβάνοντας υπόψιν τα ήδη υπάρχοντα δεδομένα του Πανελλήνιου Σχολικού Δικτύου. Το τεστ αυτό για να καταφέρει να φτάσει στα όρια μεγάλους διακομιστές, μπορεί να τρέξει σε συγχρονισμό σε πολλά διαφορετικά μηχανήματα, ώστε να παράξει την κίνηση που χρειάζεται για γίνουν τα τεστ. Γι' αυτή τη δυνατότητα θα πρέπει να στηθούν ειδικά routing στο switch που συνδέονται τα προς δοκιμή μηχανήματα και τα μηχανήματα που θα παράγουν το φορτίο. Η συγκεκριμένη λειτουργία δεν μπορεί να υλοποιηθεί στο Πανελλήνιο Σχολικό Δίκτυο, καθώς θα χρειαζόντουσαν περίπου 70.000 ιδιωτικές διευθύνσεις, 3 επιπλέον μηχανήματα και οι ανάλογες δρομολογήσεις μεταξύ των.

Δεδομένου ότι για την παρούσα διπλωματική έχουν διατεθεί 2 διακομιστές για τις δοκιμές και την υλοποίηση των squid, η πιο αξιόπιστη μέτρηση που μπορεί να γίνει και να πλησιάζει την πραγματική εξομοίωση του πραγματικού κόσμου, είναι η παραγωγή και κατανάλωση φορτίου από το ένα μηχάνημα με ενδιάμεσο το άλλο. Στο εξής τα 2 μηχανήματα θα ονομάζονται cache04 και cache06 το οποίο είναι και το πραγματικό FQDN τους.

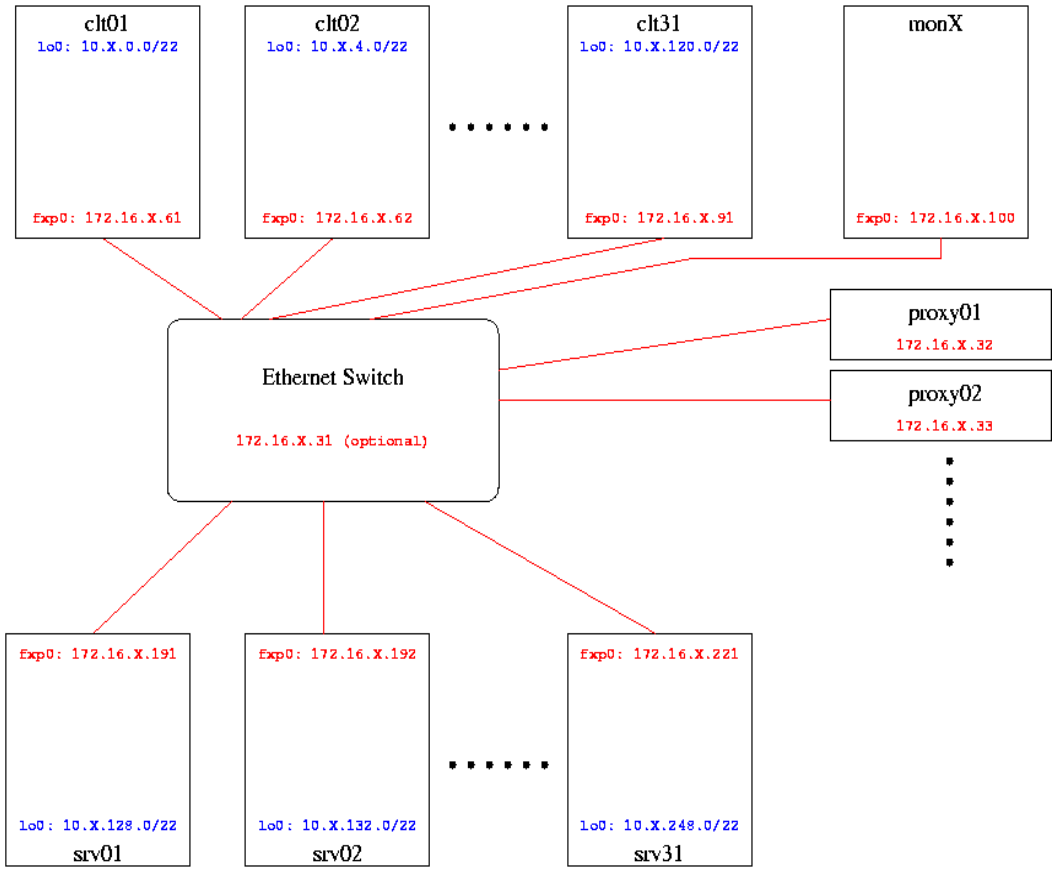
5.2.1 Βασικές έννοιες του Web Polygraph

Το εργαλείο Web Polygraph βασίζεται σε δύο στοιχεία :

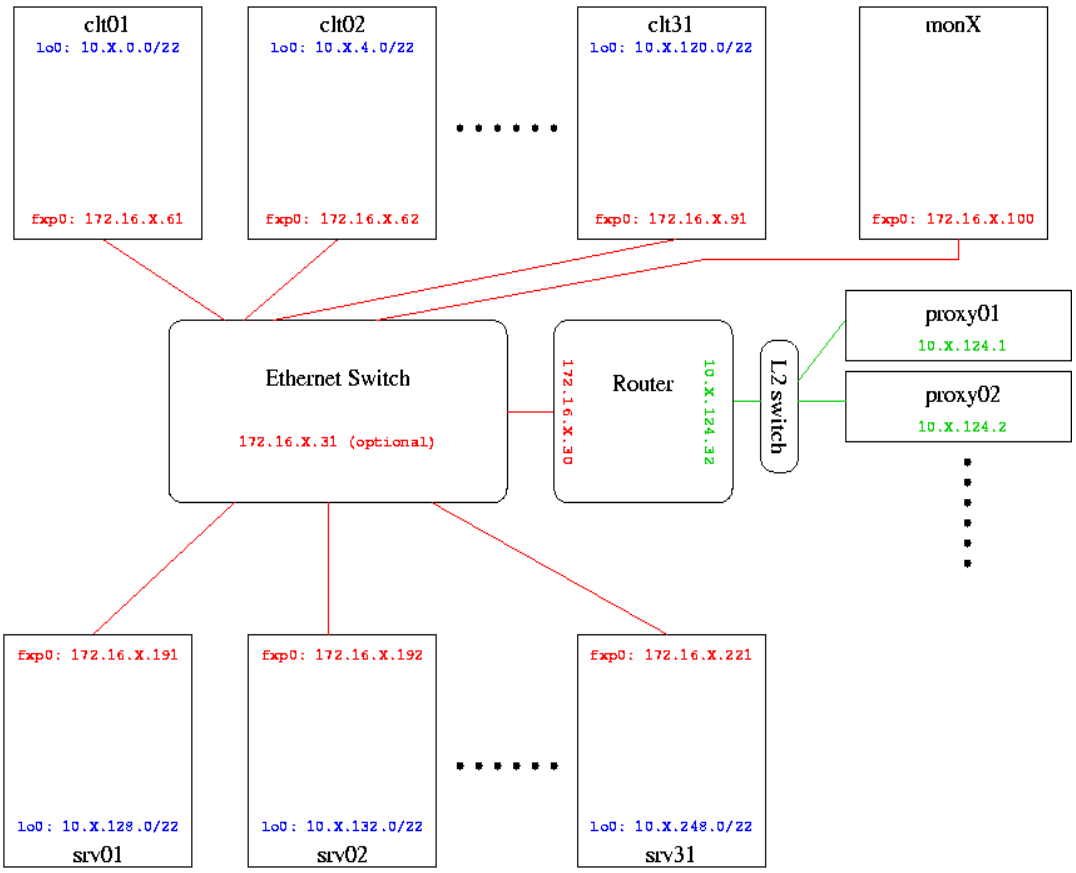
- Ένας διακομιστής, ο οποίος παράγει περιεχόμενο με τα ανάλογα URL.
- Ένας πελάτης, ο οποίος ζητά το περιεχόμενο αυτό από τον server (είτε απευθείας είτε μέσω κάποιου άλλου στοιχείου δικτύου).

Ο πελάτης και ο διακομιστής μπορούν να τρέχουν είτε στο ίδιο μηχάνημα, είτε σε διαφορετικά μηχανήματα. Όπως μπορεί επίσης να έχουμε πολύπλοκες τοπολογίες με πολλούς διακομιστές και πολλούς πελάτες σε διάφορα φυσικά μηχανήματα.

Για λόγους πληρότητας παρουσιάζεται ένα προτεινόμενο addressing scheme με τα αντίστοιχα φυσικά μηχανήματα για την παραγωγή και εξομοίωση φορτίου ενός μεγάλου ISP:



Εικόνα 30: Web Polygraph Addressing Scheme



Εικόνα 31: Web Polygraph Routing Scheme

Ένα πραγματικό stress test, σε ένα δίκτυο που συγκεντρώνονται πολλοί χρήστες πελάτες, θα χρειαζόταν μια τέτοια διαμόρφωση για να γίνει σωστή εξομοίωση φορτίου.

Στις δικές μας δοκιμές (καθώς δεν είναι δυνατόν οι αλλαγές στις δρομολογήσεις), θα χρησιμοποιήσουμε ένα πελάτη και στο ίδιο μηχάνημα θα τρέχει ο διακομιστής. Σε αυτό το μηχάνημα θα δημιουργήσουμε και τα ανάλογα robots (όπως ονομάζονται από το Web Polygraph) τα οποία θα εξομοιώνουν τους πελάτες οι οποίοι θα ζητάνε το περιεχόμενο. Ένα βασικό αρχείο παραγωγής φορτίου στο Web Polygraph πρέπει να έχει το Content (περιεχόμενο που θα δημιουργεί ο Server), τις ρυθμίσεις του server και τις ρυθμίσεις του πελάτη. Ένα πολύ απλό παράδειγμα παραγωγής φορτίου παρατίθεται παρακάτω :

```
#include "/usr/local/share/polygraph/workloads/include/contents.pg"

Content SimpleContent = {
        size = exp(13KB); // response sizes distributed exponentially
        cachable = 60%;   // 40% of content is uncachable
        obj_life_cycle = olcStatic;
};
```

Το συγκεκριμένο αρχείο δημιουργεί ένα περιεχόμενο (Content) το οποίο έχει μέγεθος εκθετικά κατανεμημένο μεταξύ 0-13KB και εκ του οποίου το περιεχόμενο το 60% μπορεί να αποθηκευτεί τοπικά στον διακομιστή μεσολάβησης. Επίσης ορίζουμε ότι το περιεχόμενο είναι εντελώς στατικό και δεν αλλάζει δυναμικά με το χρόνο (στο πραγματικό φορτίο αυτό θα αλλάξει).

```
Server S = {
        kind = "S101";
        contents = [ SimpleContent ];
        direct_access = contents;
        addresses = ['192.168.1.22:9999' ]; // where to create these server agents
};
```

Δημιουργείται επίσης έναν διακομιστής ιστού ο οποίος παράγει περιεχόμενο του προηγούμενου τύπου και έχει απευθείας πρόσβαση σε αυτό και λειτουργεί στην διεύθυνση του cache04.

```
Robot R = {
        kind = "R101";
        pop_model = { pop_distr = popUnif(); };
        recurrence = 55% / SimpleContent.cachable; // adjusted to get 55% DHR
        origins = S.addresses;      // where the origin servers are
        addresses = ['192.168.1.22' ]; // where these robot agents will be created
};
```

Τέλος, δημιουργείται ο πελάτης (Robot) το οποίο θα ζητάει και θα φέρνει το περιεχόμενο από τον προηγούμενο server.

Η αρχή λειτουργίας του Web Polygraph είναι ότι ο διακομιστής δημιουργεί τυχαίο περιεχόμενο, σε τυχαία URL και τα robot έχοντας γνώση των τυχαίων URL που δημιουργήθηκαν ζητούν αυτά τα URL.

5.2.2 Μεθοδολογία stress test

Αφού δεν είναι δυνατόν να χρησιμοποιηθεί το κανονικό έτοιμο φορτίο για authenticated caching proxies του Web Polygraph λόγω του προβλήματος, με το routing και τη διευθυνσιοδότηση, θα δημιουργηθεί ένα φορτίο παρόμοιο με αυτό του Πανελλήνιου Σχολικού Δικτύου από καταγραφές που ελήφθησαν από τους ήδη εν λειτουργία διακομιστών squid.

Για να διαπιστωθεί αν το bottleneck του συστήματος είναι το μηχάνημα παραγωγής κίνησης ή ο διακομιστής μεσολάβησης που θα δοκιμαστεί πρέπει πρώτα να γίνουν μερικά τεστ πάνω στο ίδιο το μηχάνημα δοκιμής. Δεδομένου ότι τα 2 μηχανήματα είναι πανομοιότυπα, υπάρχει η περίπτωση να μην μπορέσουμε να παράξουμε τόσο μεγάλο φορτίο ώστε να κορεστεί κάποιο στοιχείο του διακομιστή μεσολάβησης.

Για να διαπιστωθεί η δυνατότητα παραγωγής κατάλληλου φορτίου, για αρχή θα χρησιμοποιήσουμε το Web Polygraph στο ίδιο μηχάνημα μόνο(cache04), χωρίς κανένα ενδιάμεσο στοιχείο δικτύου και ξεκινώντας από έναν αριθμό request 500/sec. Στη συνέχεια θα αυξάνουμε τον αριθμό των αιτημάτων ανά 50, μέχρι να αρχίσουν τα πρώτα errors και timeouts. Όταν αρχίσουν να αυξάνονται πολύ τα errors και timeouts τότε ξέρουμε ότι έχουμε φτάσει το μηχάνημα αυτό σε σημείο κορεσμού και δεν μπορούμε πλέον να παράξουμε και να καταναλώσουμε μεγαλύτερη κίνηση από αυτό τον Χ αριθμό αιτημάτων/ δευτερόλεπτο.

Όταν βρεθεί το σημείο κορεσμού του μηχανήματος cache04, αλλάζουμε το stress test, ώστε αυτή τη φορά να περιλαμβάνει και έναν μη authenticated και μη caching διακομιστή μεσολάβησης (αφού το πεδίο των δοκιμών μας θα είναι αυτές οι λειτουργίες για να δοκιμάσουμε το σημείο κορεσμού απενεργοποιούμε αυτές τις δυνατότητες οι οποίες μπορεί να μειώσουν την απόδοση του διακομιστή μεσολάβησης). Έχοντας στη διάθεσή μας τα προηγούμενα τεστ, δοκιμάζουμε το τελευταίο φορτίο που είχαμε και δεν είχε παράξει καθόλου errors και timeouts στον cache04. Αν σε αυτή τη δοκιμή δημιουργηθούν timeouts και errors σημαίνει ότι φτάσαμε τον διακομιστή μεσολάβησης σε σημείο κορεσμού πριν φτάσει το μηχάνημα παραγωγής φορτίου και άρα μπορούμε με ασφάλεια να χρησιμοποιήσουμε το συγκεκριμένο test pattern με το συγκεκριμένο μηχάνημα για να εξάγουμε συμπεράσματα.

Αφού διαπιστώσαμε ότι το μηχάνημα παραγωγής φορτίου μπορεί να παράξει αρκετή κίνηση τώρα θα δοκιμάσουμε το σημείο κορεσμού του διακομιστή μεσολάβησης όταν αυτός χρησιμοποιείται μόνο ως forwarding proxy (non caching,non authenticated).

Οι δοκιμές μας θα ξεκινήσουν στα 100 αιτήματα / δευτερόλεπτο και σε κάθε τεστ θα αρχίσουν να αυξάνονται κατά 100 μέχρι να αρχίσουν τα πρώτα errors και timeouts.

Όταν φτάσουμε σε σημείο που αυξάνονται πολύ τα errors και timeouts μπορούμε να πούμε με ασφάλεια ότι ό διακομιστής μεσολάβησης μπορεί να διεκπεραιώσει τόσα αιτήματα / δευτερόλεπτο.

Όταν διαπιστώσουμε το όριο του διακομιστή μεσολάβησης θα βάλουμε τον διακομιστή μεσολάβησης σε λειτουργία authenticated proxy (μέσω LDAP του ΠΣΔ) και θα δοκιμάσουμε πάλι το ίδιο stress test μέχρι να αρχίσουν τα errors και timeouts. Από το συγκεκριμένο τεστ θα εξαγάγουμε την διεκπαιρεωτική ικανότητα του διακομιστή μεσολάβησης όταν αυτός χρησιμοποιηθεί για αυθεντικοποιημένη χρήση. Τα συμπεράσματα που θα βγάλουμε θα μας ενημερώσουν για την μείωση ή όχι της απόδοσης των διακομιστών μεσολάβησης όταν αυτοί τεθούν σε λειτουργία αυθεντικοποιημένης κίνησης εντός του Ελληνικού Σχολικού Δικτύου.

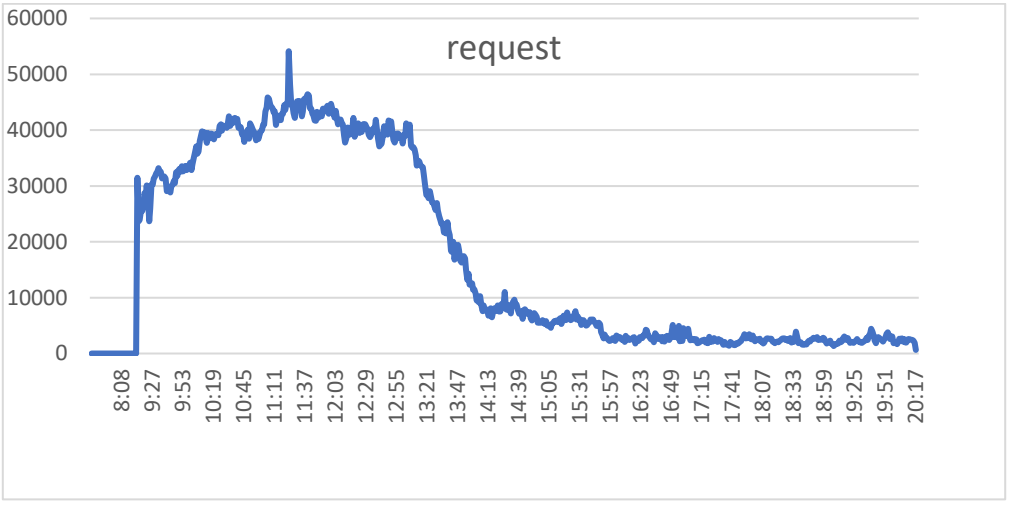
Ένας παράγοντας που δεν μπορούμε να υπολογίσουμε βέβαια είναι πόση θα είναι η μείωση της πραγματικής κίνησης καθώς αυτή τη στιγμή οποιοσδήποτε μπορεί σε ένα σχολείο να συνδεθεί στο Πανελλήνιο Σχολικό Δίκτυο και να χρησιμοποιεί τους διακομιστές μεσολάβησης. Είναι σίγουρό ότι μετά την εφαρμογή της αυθεντικοποιημένης χρήσης των διακομιστών μεσολάβησης ο αριθμός των συνδέσεων και κατ' επέκταση η κίνηση θα μειωθεί αλλά δεν γνωρίζουμε το ποσοστό μείωσης ούτε κατά προσέγγιση. Για ασφάλεια στις μετρήσεις χρησιμοποιούνται όλες οι συνδέσεις που πραγματοποιούνται τώρα στο Πανελλήνιο Σχολικό Δίκτυο.

Για το φορτίο που θέλουμε να δοκιμάσουμε, η δοκιμή στον cache04 μας έβγαλε ότι μπορεί να διαχειριστεί χωρίς errors και timeouts 4500 αιτήσεις/ δευτερόλεπτο για το είδος του φορτίου το οποίο παράγουμε. Η ίδια δοκιμή με ενδιάμεσο τον cache06 (ως διακομιστή μεσολάβησης) το ίδιο ακριβώς φορτίο τον κατέστησε μη αποκρίσιμο και χρειάστηκε επανεκκίνηση για να επανέλθουν οι υπηρεσίες του. Αυτό μας οδηγεί στο ασφαλές συμπέρασμα ότι ο cache04 σαν μηχάνημα δοκιμών μπορεί να παράξει τόσο φορτίο όσο χρειάζεται για να μπορέσουμε να κάνουμε τα stress test και να βγάλουμε σαφή συμπεράσματα.

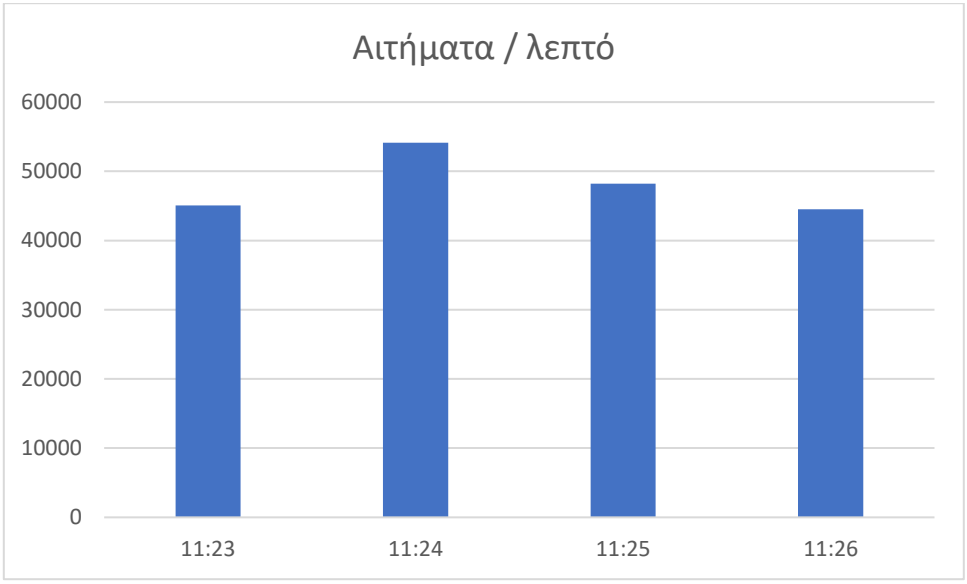
Όπως είδαμε και από την ανάλυση του access.log του πιο φορτωμένου cachebox μέχρι στιγμής του Ελληνικού Σχολικού Δικτύου αυτό είχε σαν μέγιστο αριθμό αιτημάτων το δευτερόλεπτο τις 902.

Το stress test αυτό αφορά καθαρά το πρωτόκολλο http και είναι μέτρο σύγκρισης με την έως τώρα διαφανή λειτουργία των διακομιστών μεσολάβησης.

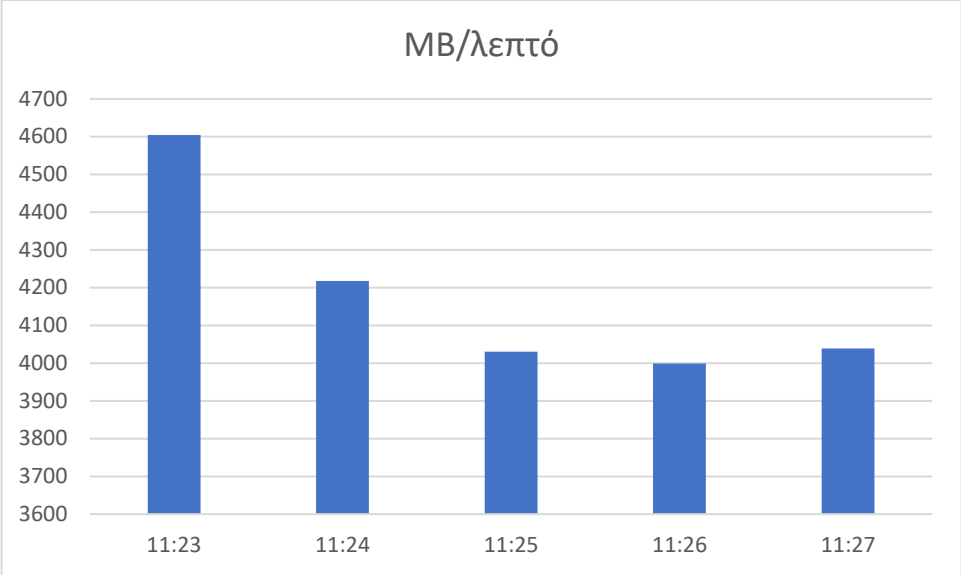
Στα διαγράμματα βλέπουμε την ανάλυση κίνησης από τον διακομιστή μεσολάβησης και καταλήγουμε στα αιτήματα ανά λεπτό. Παρατηρούμε ότι έχουμε περίπου στο μέγιστο φορτίο της ημέρας 54127 αιτήματα/λεπτό ή αλλιώς 902 αιτήματα / δευτερόλεπτο.



Εικόνα 32: Αιτήματα κατά τη διάρκεια της ημέρας παραγωγικού squid στο ΠΣΔ



Εικόνα 33: Αιτήματα/λεπτό παραγωγικού squid στο ΠΣΔ



Εικόνα 34: Throughput/λεπτό παραγωγικού squid στο ΠΣΔ

.

Αναμένουμε λοιπόν τουλάχιστον αν λειτουργήσει ο cache06 (ο forward proxy του test) χωρίς αυθεντικοποίηση να μπορέσει να διαχειριστεί ένα τέτοιο φορτίο.

5.2.3 Εξαγωγή στατιστικών και φορτίου από υπάρχον δίκτυο

Για τις ανάγκες της παρούσας διπλωματικής έπρεπε να εξομοιώσουμε την πραγματική κίνηση που δέχεται ένας forward proxy, έτσι όπως λειτουργεί το Πανελλήνιο Σχολικό Δίκτυο. Για την ανάγκη αυτή επιλέχτηκε ο διακομιστής με το μεγαλύτερο φορτίο (αυτός στον οποίο πέρναγαν τα winows update). Ο squid server στο log file του /var/log/squid/access.log καταγράφει όλη την κίνηση με όλα τα αιτήματα που περνάνε από αυτόν.

Από αυτό το αρχείο μπορούν να εξαχθούν πολλά συμπεράσματα και με την κατάλληλη επεξεργασία να δημιουργηθεί ένα ίδιο φορτίο με αυτό που περνάει από τον cache06. Για τις ανάγκες αυτές χρησιμοποιήθηκαν μερικά έτοιμα scripts του Web Polygraph τα οποία μας ετοιμάζουν το αρχείο παραγωγής φορτίου, καθώς επίσης γράφτηκαν και μερικά ώστε αυτά να μπορέσουν να μετατρέψουν το access.log σε αρχείο κατάλληλο input για το πρόγραμμα παραγωγής φορτίου του Web Polygraph. Τα script αυτά βρίσκονται στο Παράρτημα 7.1.1.

Μέσω αυτών των script καταφέραμε και βγάλαμε ένα αρχείο στο οποίο περιλαμβάνονται σε % ποσοστά τα εξής στοιχεία για κάθε χρήστη πελάτη που θα εξομοιώσουμε καθώς και στοιχεία του διακομιστή.

Για τον χρήστη πελάτη εξομοιώνουμε τα εξής στοιχεία σε διανομές:

- Διανομή χρόνου μεσολάβησης μεταξύ αιτημάτων.
- Διανομή Μέθοδοι αιτημάτων (GET, HEAD, POST).

Για τον διακομιστή :

- Διανομή κωδικών απαντήσεων (206,200,304 κτλ)
- Διανομή μεγέθους απαντήσεων
- Διανομή χρόνου κατά την οποία είναι κατειλημμένος ο διακομιστής (busy_period)
- Διανομή χρόνου σκέψης

Το stress test που θα τρέξουμε, περιλαμβάνει και εξομοιώνει όλα αυτά τα στοιχεία κατά την περίοδο που αυτό λειτουργεί.

Από ότι είδαμε και πριν θα χρειαστούμε 920 αιτήματα /δευτερόλεπτο, το οποίο θα το εξομοιώσουμε με 920 robot του Web Polygraph και τα οποία δεν θα έχουν πάνω από 16 ταυτόχρονες ανοιχτές συνδέσεις το καθένα. Επιπλέον, ο server θεωρεί σαν timeout τα 15 δευτερόλεπτα.

Για την εξομοίωση της αυθεντικοποίησης, έχουν δημιουργηθεί στον πραγματικό LDAP server του Πανελλήνιου Σχολικού Δικτύου 800 λογαριασμοί χρηστών μαθητών και 200 λογαριασμοί χρηστών καθηγητών. Το κάθε robot διαλέγει τυχαία ένα από τα 1000 αυτά διαπιστευτήρια και κατόπιν το χρησιμοποιεί μέχρι να ολοκληρωθεί η συνεδρία που αυτό έχει εκκινήσει. Στην επόμενη συνεδρία επιλέγει πάλι τυχαία ένα από 1000 αυτά διαπιστευτήρια και συνεχίζει.

Ο squid server από τη στιγμή που θα κάνει μια αυθεντικοποίηση, κρατάει τοπικά το base64 του username και password του χρήστη ώστε να μην χρειάζεται να το ξαναζητήσει από τον διακομιστή ldap. Ο χρόνος που κρατάει τοπικά τα διαπιστευτήρια μπορεί να οριστεί με το directive credentials ttl. Για τις δικιές μας δοκιμές θα κρατάμε τοπικά τα διαπιστευτήρια για 1 λεπτό. Αυτό γιατί παρόλο που έχουμε 902 αιτήματα/ δευτερόλεπτο είναι σχεδόν αδύνατο στον ίδιο διακομιστή μεσολάβησης να έρθουν ταυτόχρονα πάνω από 1000 καινούριοι χρήστες. Για λόγους πληρότητας θα δοκιμάσουμε και ένα stress test

όπου τα διαπιστευτήρια δεν αποθηκεύονται καθόλου και αναγκάζεται ο squid server να κάνει ερωτήσεις συνέχεια στον ldap διακομιστή, πράγμα το οποίο δεν έχει νόημα στον πραγματικό κόσμο καθώς το 90% των χρηστών που χρησιμοποίησαν τον διακομιστή μεσολάβησης θα τον χρησιμοποιήσουν και μέσα στο επόμενο 8ωρο (τόσο είναι και το προτεινόμενο tll από τους δημιουργούς του squid server). Επιπλέον στην διαδικασία ταυτοποίησης εισάγουμε και ένα 0,5% το οποίο θα κάνει λάθος στην εισαγωγή του κωδικού του.

Τέλος θα κάνουμε και ένα stress test στο οποίο θα μετρηθεί πόσο CPU intensive είναι το ssl bump του squid server (man in the middle).

5.3 ΓΕΝΙΚΕΣ ΡΥΘΜΙΣΕΙΣ ΓΙΑ ΟΛΑ ΤΑ STRESS TEST

Τα μηχανήματα τα οποία έχουμε στη διάθεσή μας για διακομιστές μεσολάβησης έχουν τα εξής χαρακτηριστικά :

8 cores

4 GB RAM

Ο squid μπορεί να χρησιμοποιήσει SMP αρχιτεκτονική οπότε μπορούμε να μοιράσουμε τις εργασίες σε διαφορετικά cores της CPU. Σύμφωνα με το δικό μας σύστημα μπορούμε να χρησιμοποιήσουμε 5 workers του squid, όπου ο κάθε worker θα χρησιμοποιεί ένα διαφορετικό core της CPU. Χρειαζόμαστε άλλο ένα core για το orchestrator του squid,δηλαδή για το process το οποίο θα διαχειρίζεται τους workers και θα τους αναθέτει εργασίες. Τα υπόλοιπα 2 core θα χρησιμοποιηθούν καθαρά από το λειτουργικό για άλλες εργασίες. (Το λειτουργικό φυσικά μπορεί να χρησιμοποιήσει τα cores που τρέχουν οι squid workers). Η δήλωση των workers γίνεται μέσα στο αρχείο ρύθμισης του squid με το directive workers 5 στην περίπτωσή μας.

Ο squid χρησιμοποιεί το βοηθητικό πρόγραμμα basic_ldap_auth για την αυθεντικοποίηση των χρηστών. Για βοηθητικό πρόγραμμα του squid μπορούμε να τρέξουμε τόσα instances όσα δηλώνουμε και στο αρχείο ρύθμισης. Έτσι μπορούμε να ξεκινήσουμε με ένα αρχικό (startup) αριθμό και πόσα instances θα υπάρχουν idle κάθε φορά για να εξυπηρετήσουν τα αιτήματα. Επιπλέον μπορούμε να βάλουμε και ένα άνω όριο στον αριθμό των instances που μπορεί να γίνουν spawn από τον squid server. Στη δικιά μας περίπτωση με το εξής directive :

auth_param basic children 500 startup=0 idle=1

Σε όλα τα stress test ο squid είναι ρυθμισμένος να κάνει url filtering μέσω του ufdbguard και οι ρυθμίσεις του ufdbguard είναι ακριβώς αυτές οι οποίες χρησιμοποιούνται αυτή τη στιγμή και στο υπάρχον δίκτυο.

Σε όλα τα stress test (εκτός του ssl bump) το φορτίο είναι το ίδιο και είναι αυτό που ορίζεται στο αρχείο workload.pgd στο Παράρτημα 7.1.2.

5.4 ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕΤΡΗΣΕΩΝ

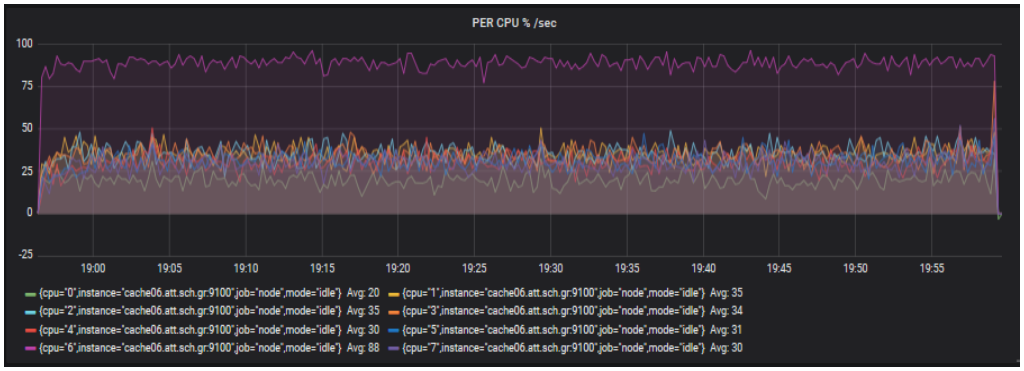
5.4.1 Squid Server σαν intercept proxy

Στην λειτουργία διάφανου διακομιστή μεσολάβησης εφαρμόζοντας την κατανομή φορτίου που πήραμε από τα access.log των εν λειτουργία διακομιστών μεσολάβησης, ο squid κατάφερε να διεκπεραιώσει 920 αιτήματα / δευτερόλεπτο (από διαφορετικούς χρήστες πελάτες) με αμελητέα ποσότητα χαμένων πακέτων (<0.5%).

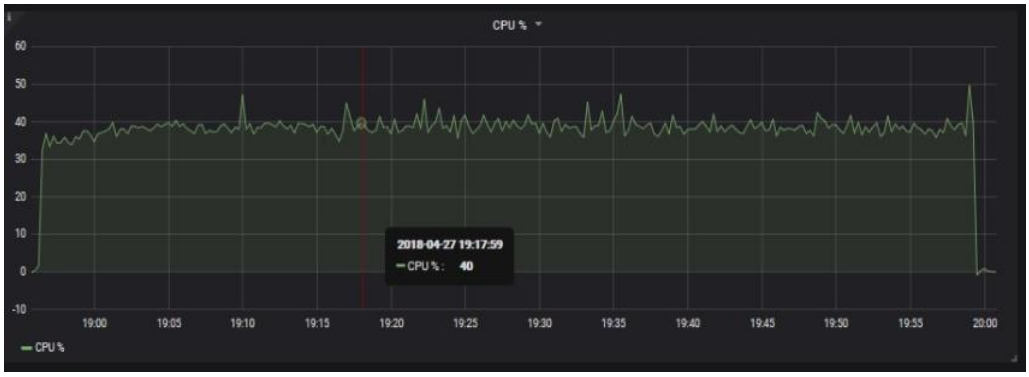
Όσο αυξανόταν το φορτίο και περί τα 950 αιτήματα/ δευτερόλεπτο παρουσιαζόταν εκθετική αύξηση των χαμένων πακέτων. Στα 950 αιτήματα / δευτερόλεπτο το ποσοστό των χαμένων πακέτων εκτοξεύτηκε στο 10%.

Αυτό που παρατηρήθηκε εκτός του ότι ο διακομιστής μεσολάβησης είχε φτάσει σχεδόν στο μέγιστο το bandwidth είναι ότι ένα core του επεξεργαστή ήταν σχεδόν μόνιμα στο 99% και όλος ο χρόνος του επεξεργαστή χρησιμοποιούταν για interrupts. Αυτό έχει σχέση με τον driver ο οποίος χρησιμοποιείται στο FreeBSD για τις συγκεκριμένες κάρτες δικτύου και δεν μπορεί να χρησιμοποιήσει περισσότερα cores του επεξεργαστή για τα interrupts. Μια λύση προτείνεται στο επόμενο κεφάλαιο στα συμπεράσματα.

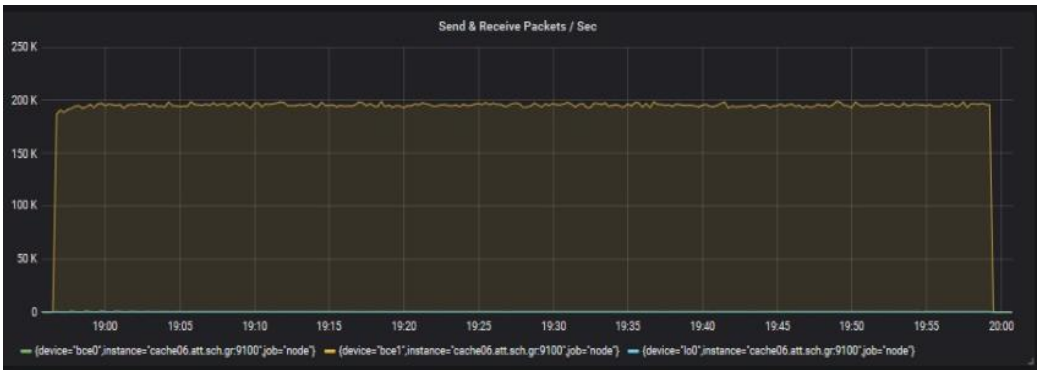
Παρακάτω είναι οι μετρήσεις που πραγματοποιήθηκαν στον squid σε αυτό το mode (το αρχείο ρύθμισης του stress test βρίσκεται στο Παράρτημα 7.1.3).



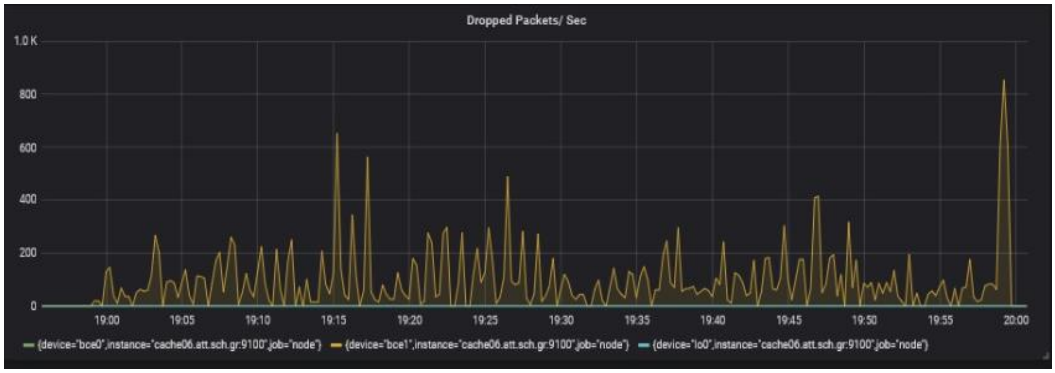
Εικόνα 35: Squid No authentication 920 reqs/sec PER CPU %



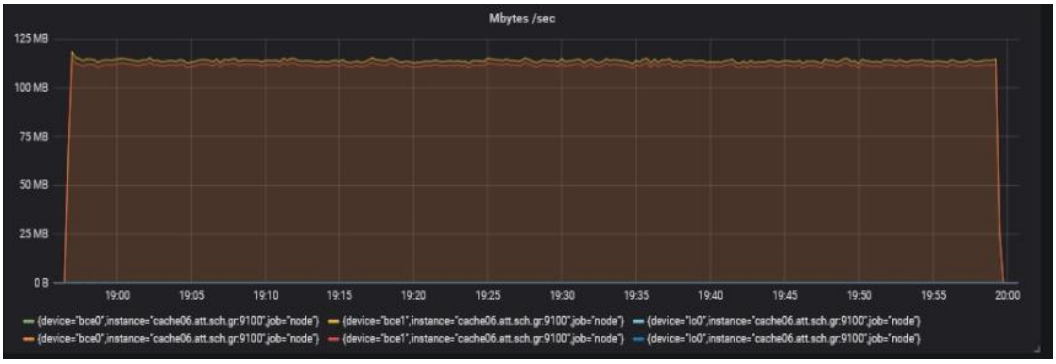
Εικόνα 36: Squid No authentication 920 reqs/sec CPU% Average



Εικόνα 37: Squid No authentication 920 reqs/sec Packets/Sec



Εικόνα 38: Squid No authentication 920 reqs/sec Dropped Packets / sec



Εικόνα 39: Squid No authentication 920 reqs/sec Mbytes/sec

Παρατηρούμε στην Εικόνα 22 ότι η CPU 6 είναι μόνιμα πάνω από το 95% και πλησιάζει το 99%. Αυτό οφείλεται στα interrupts της κάρτας δικτύου. Παρακάτω ακολουθεί και μια μέτρηση στα 970 requests/sec.

Εικόνα 40: Squid No authentication 970 reqs/sec dropped packets/sec

Για το συγκεκριμένο φορτίο παρατηρούμε ότι το ποσοστό χαμένων πακέτων αυξάνεται στο 9,5% περίπου.

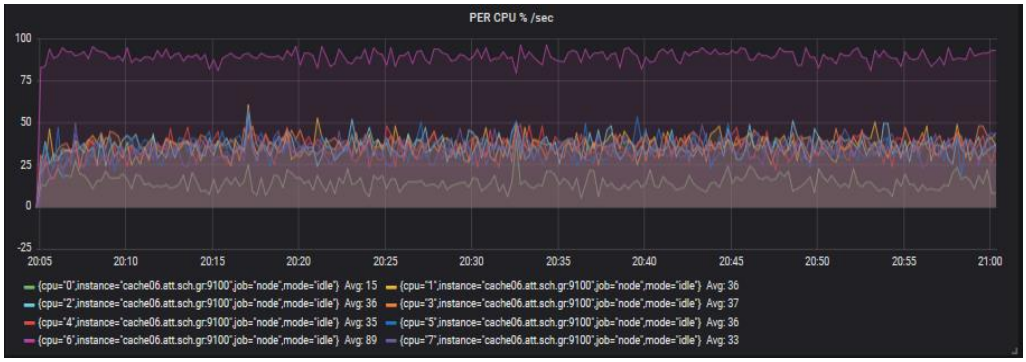
Με αυτά τα δεδομένα αφού έχουμε φτάσει τον squid στο όριο της διεκπεραιωτής του ικανότητας θα κάνουμε το stress test με αυθεντικοποίηση.

5.4.2 Squid Server forward proxy με ldap authentication

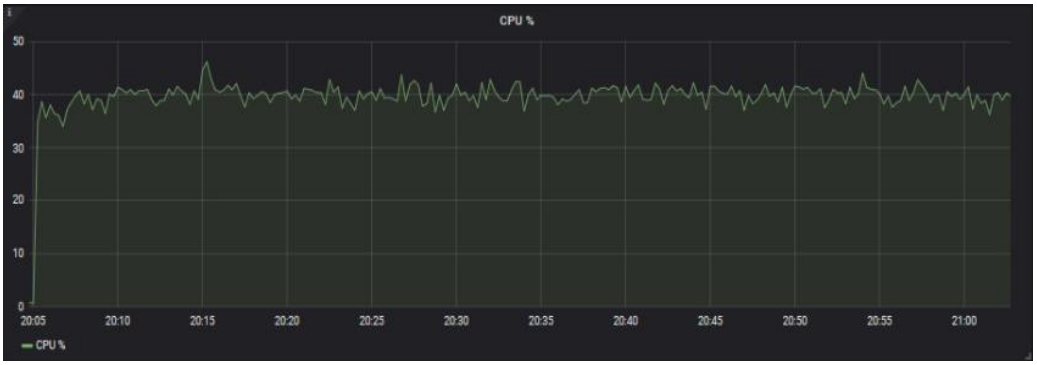
Έχοντας τις προηγούμενες μετρήσεις από το προηγούμενο stress test θα δοκιμάσουμε με τις ίδιες παραμέτρους να παρατηρήσουμε την επιβάρυνση που επιφέρει η αυθεντικοποιήσεις μέσω ldap στο περιβάλλον μας.

Παρακάτω ακολουθούν οι μετρήσεις για το ίδιο φορτίο με 920 request/sec και 1000 διαφορετικά username και passwords. Ο squid έχει ρυθμιστεί ώστε να μην αποθηκεύει καθόλου τα credentials ώστε να δούμε την πραγματική επιβάρυνση της αυθεντικοποιήσεις. Αυτό σημαίνει ότι ο squid θα κάνει 920 αυθεντικοποιήσεις το δευτερόλεπτο σαν κάθε χρήστης πελάτης να είναι μοναδικός.

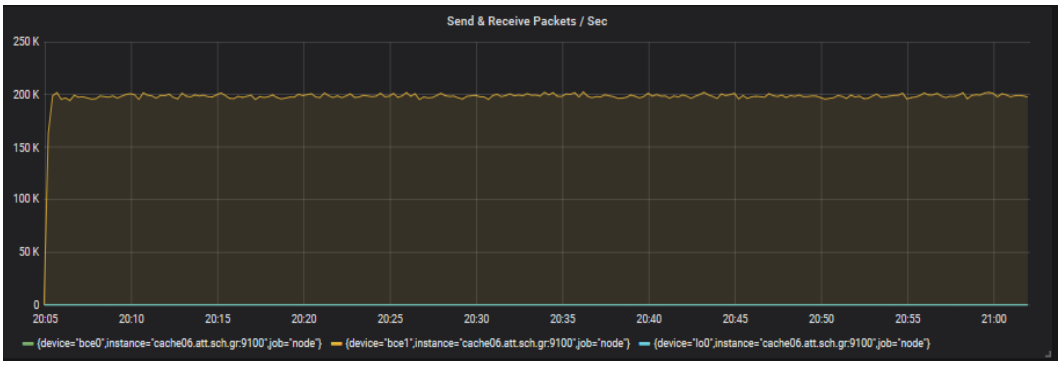
Ακολουθούν τα αποτελέσματα των μετρήσεων.



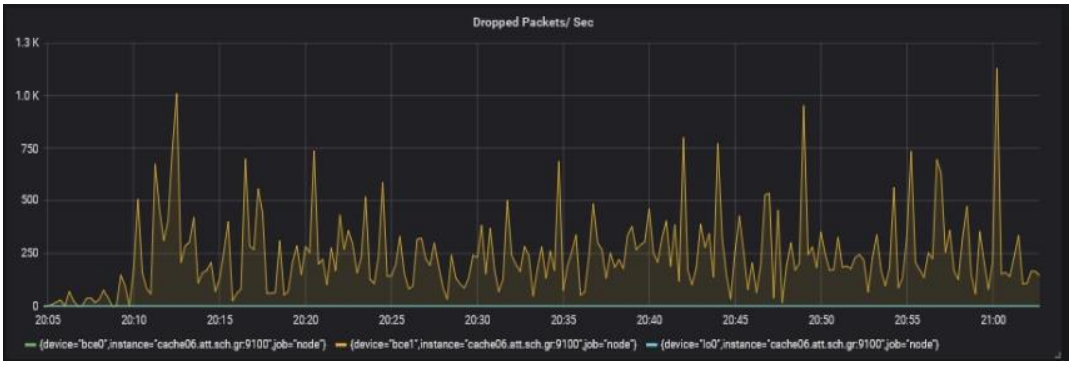
Εικόνα 41: Squid LDAP Authentication 920 reqs/sec PER CPU %



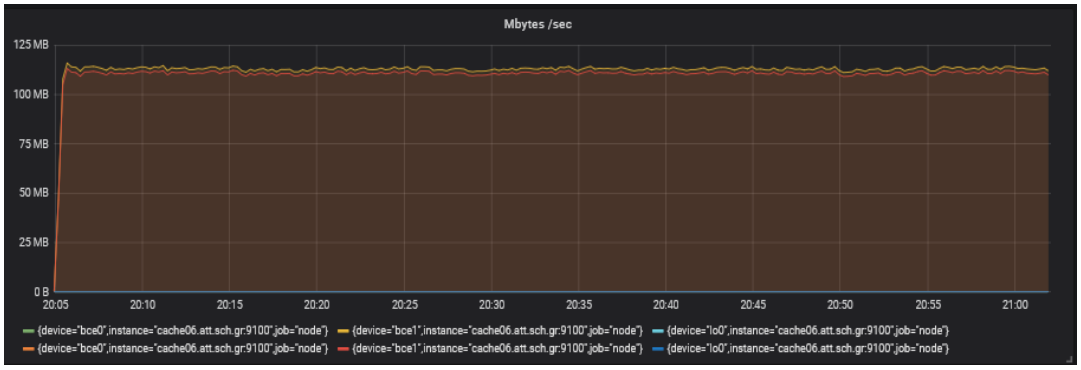
Εικόνα 42: Squid LDAP Authentication 920 reqs/sec CPU%



Εικόνα 43: Squid LDAP Authentication 920 reqs/sec Packets/Sec



Εικόνα 44: Squid LDAP Authentication 920 reqs/sec Dropped Packets/Sec



Εικόνα 45: Squid LDAP Authentication 920 reqs/sec Mbytes/Sec

Παρατηρούμε ότι ανέβηκε μεσοσταθμικά η χρήση των CPU από 30,71% σε 32,57. Αυτό οφείλεται στα threads τα οποία εκτελούνται ώστε να κάνουν την αυθεντικοποίηση. Αυτά τα threads τρέχουν ακριβώς στα ίδια cores που τρέχουν και οι squid workers. Παρατηρούμε λοιπόν ότι αυτά τα cores δεν φτάνουν ποτέ σε κορεσμό (για την ακρίβεια δεν ανεβαίνουν πάνω από 45% καθ' όλη τη διάρκεια του stress test). Είναι ασφαλές λοιπόν να συμπεράνουμε ότι η αυθεντικοποίηση δεν δημιουργεί κάποιο επιπλέον φορτίο το οποίο δεν μπορούν να διαχειριστεί η υποδομή και φυσικά δεν θα αποτελεί το επιπλέον πολύ μικρό φορτίο της αυθεντικοποίησης το bottleneck του συστήματος μας.

Ένα μικρό μεν αλλά αισθητό ποσοστό αύξησης των χαμένων πακέτων οφείλεται στο γεγονός ότι το μηχάνημα δεν μπορεί να υποστηρίξει περισσότερο bandwidth με αποτέλεσμα λόγω των αυξημένων πακέτων προς τον ldap server για authentication να χάνονται μερικά από αυτά, αλλά χωρίς κάποια ιδιαίτερη επιβάρυνση στους χρόνου απάντησης και στους χρόνους απάντησης.

5.4.3 Squid Server με SSLBUMP (Man In The Middle)

Για την εφαρμογή του sslbump γνωρίζουμε εκ των προτέρων ότι το bottleneck του συστήματος θα είναι η επεξεργαστική ισχύς καθώς με το sslbump για κάθε σύνδεση που κάνει ο χρήστης πελάτης έχουμε μια κρυπτογράφηση αποκρυπτογράφηση από τη μεριά του πελάτη και άλλη μια κρυπτογράφηση αποκρυπτογράφηση από τη μεριά του τελικού προορισμού. Η κρυπτογράφηση αποκρυπτογράφηση είναι μια cpu intensive διαδικασία κάτι το οποίο θα παρατηρήσουμε και στα επόμενα stress test.

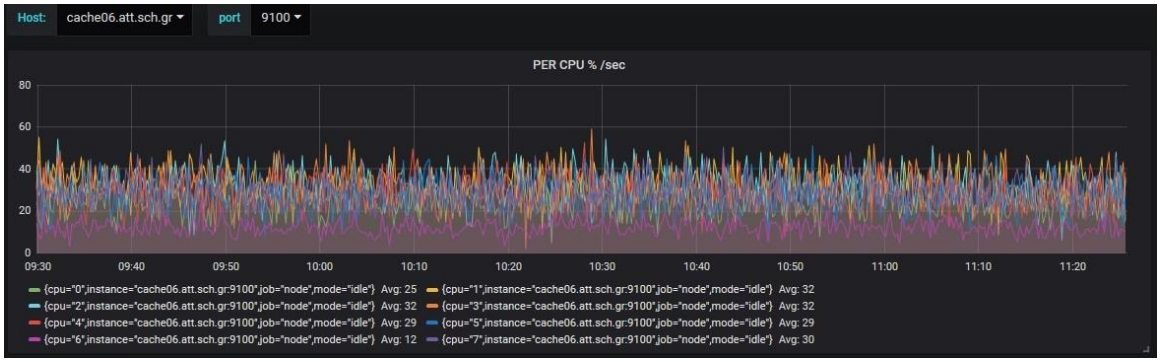
Για αρχή προσπαθήσαμε να περάσουμε το ίδιο φορτίο με τα ίδια αιτήματα ανά δευτερόλεπτο με τα προηγούμενα τεστ και ο διακομιστής μεσολάβησης κατέστη μη αποκρίσιμος. Χρειάστηκε cold reboot για να επανέλθει οπότε γνωρίζουμε ότι το φορτίο που μπορεί να δεχτεί είναι αρκετά μικρότερο από το 50% του προηγουμένου.

Οι δοκιμές τελικά ξεκίνησαν από τα 10 αιτήματα ανά δευτερόλεπτο καθώς ακόμα και στο 50% του φορτίου ο διακομιστής κατέστη πάλι μη αποκρίσιμος. Από τα 10 αιτήματα ανά δευτερόλεπτο κιόλας παρατηρήθηκε μεγαλύτερη χρήση της CPU και συγκεκριμένα από τα cores που χρησιμοποιούν τα threads του squid για την κρυπτογράφηση και αποκρυπτογράφηση.

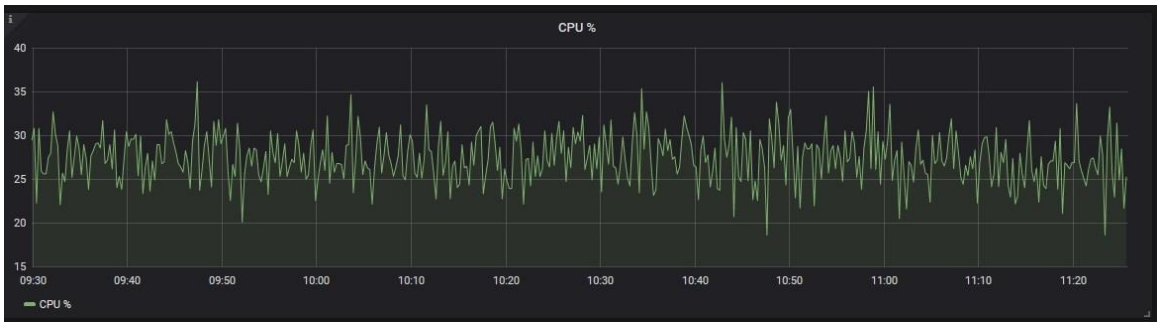
Από τα 10 αιτήματα ανά δευτερόλεπτο η αύξηση της ζήτησης έγινε σχεδόν γραμμικά για να παρατηρήσουμε ότι η επεξεργαστική ισχύς που απαιτείται από το σύστημά μας αυξάνεται σχεδόν εκθετικά.

Οι δοκιμές που έγιναν ξεκίνησαν από τα 10 αιτήματα/δευτερόλεπτο, συνέχισαν στα 50 αιτήματα/δευτερόλεπτο και κατόπιν αυξάνονταν ανά 50 μέχρι να φτάσουμε σε κορεσμό της επεξεργαστικής ισχύς. Ο κορεσμός παρατηρήθηκε σχεδόν στα 160 αιτήματα/δευτερόλεπτο όπου και τα cores που χρησιμοποιούσαν τα threads του squid έφτασαν στο 99%+, και από αυτό το σημείο και μετά άρχισε να γεμίζει το swap partition του squid μέχρι σημείου που όλο το λειτουργικό κατέστη μη αποκρίσιμο.

Στα αποτελέσματα του πρώτου stress test που ακολουθούν, χρησιμοποιήθηκε το ίδιο ακριβώς workload με τα προηγούμενα stress test :



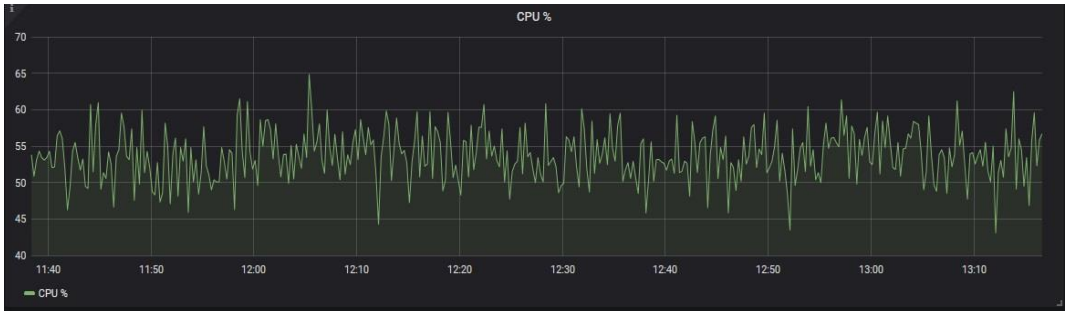
Εικόνα 46: SSL Bump 50 cons/sec per CPU



Εικόνα 47: SSL Bump 50 cons/sec CPU average

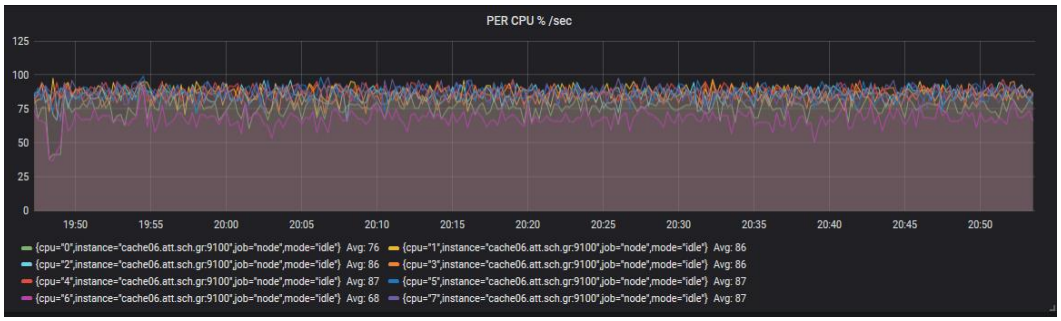
Εικόνα 48: SSL Bump 50 cons/sec packets/sec

Εικόνα 49: SSL Bump 100 cons/sec per CPU

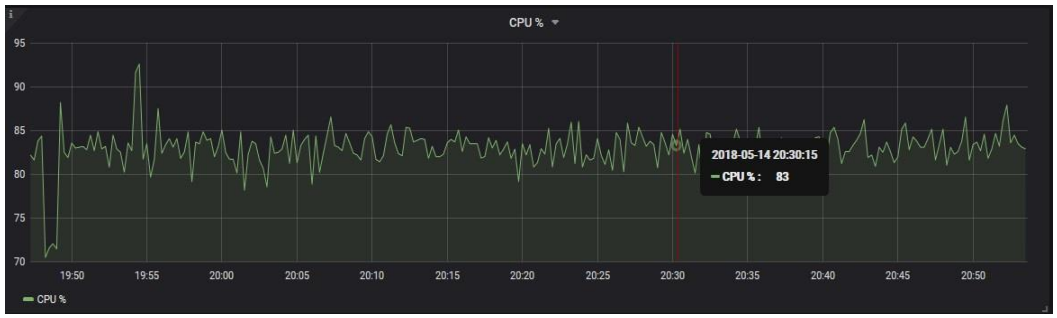


Εικόνα 50: SSL Bump 100 cons/sec CPU average

Εικόνα 51: SSL Bump 100 cons/sec packets/sec



Εικόνα 52: SSL Bump 150 cons/sec per CPU



Εικόνα 53: SSL Bump 150 cons/sec CPU average

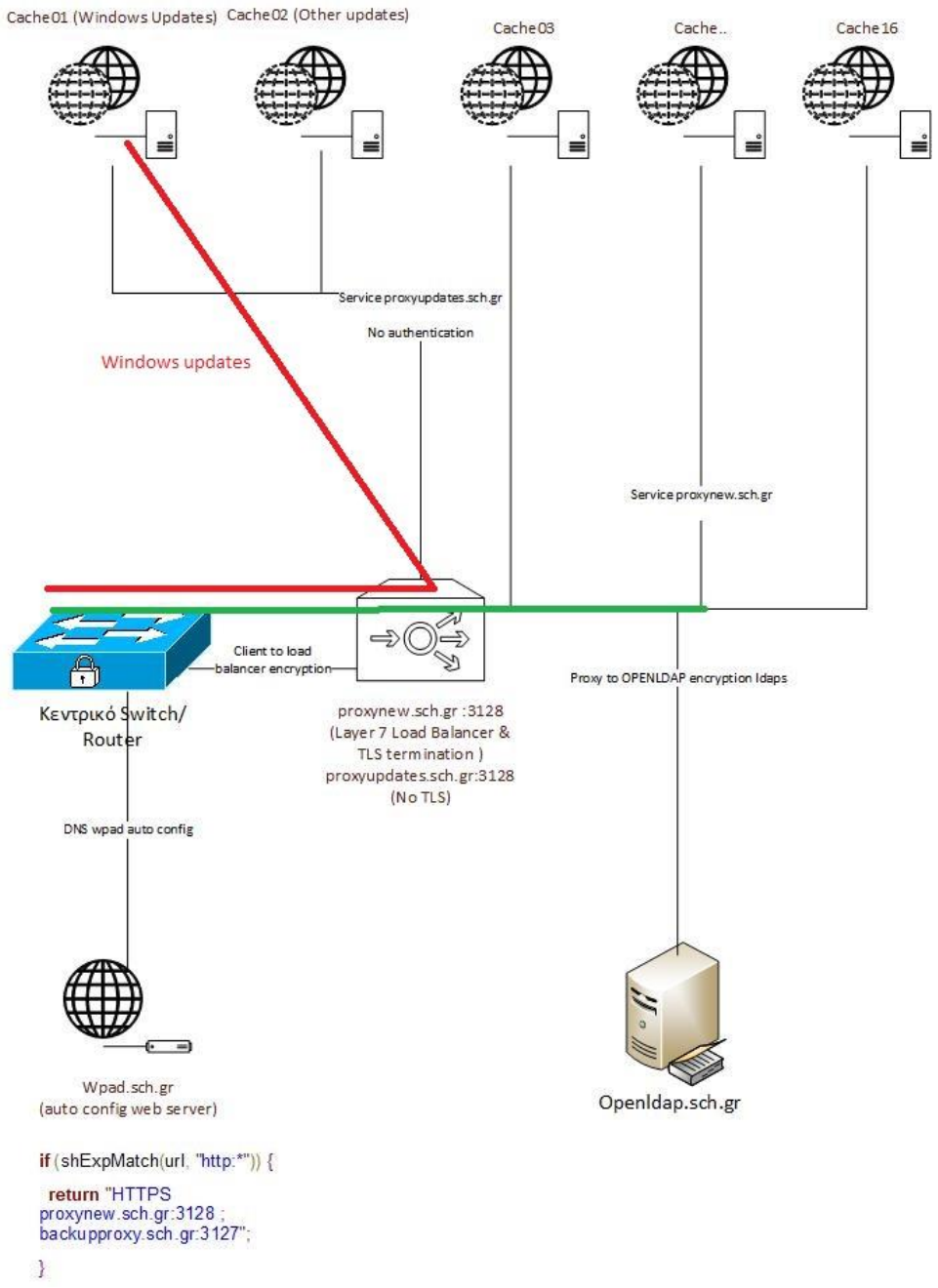
Εικόνα 54: SSL Bump 150 cons/sec packets/sec

Παρατηρούμε από τις προηγούμενες μετρήσεις και συγκεκριμένα για τις cpu 1,2,3,4,5,7 (5 workers του squid και ένας orchestrator) ότι από τη μέση χρήση του 30 % στα 50 req/sec, αυξήθηκε στα 60% στα 100 req/sec ώστε τελικά να φτάσει στο 87% για τα 150 req/sec που από ότι φαίνεται είναι και το ανώτερο όριο ταυτόχρονων συνδέσεων που μπορεί να εξυπηρετήσουν τα συγκεκριμένα μηχανήματα. Στα 160 req/sec η χρήση των CPU έφτασε στο 99%+ με αποτέλεσμα να μην μπορούν να γίνουν καθόλου μετρήσεις και να καθίσταται το μηχάνημα μη αποκρίσιμο.

Παρατηρούμε επίσης ότι το μηχάνημα δεν έφτασε ποτέ σε σημείο κορεσμού του bandwidth (το μηχάνημα υποστηρίζει μέχρι και 150MB/sec), οπότε και γνωρίζουμε ότι το συστατικό κορεσμού είναι η CPU η οποία δεν μπορεί να υποστηρίξει τόσες κρυπτογραφήσεις/αποκρυπτογραφήσεις το δευτερόλεπτο.

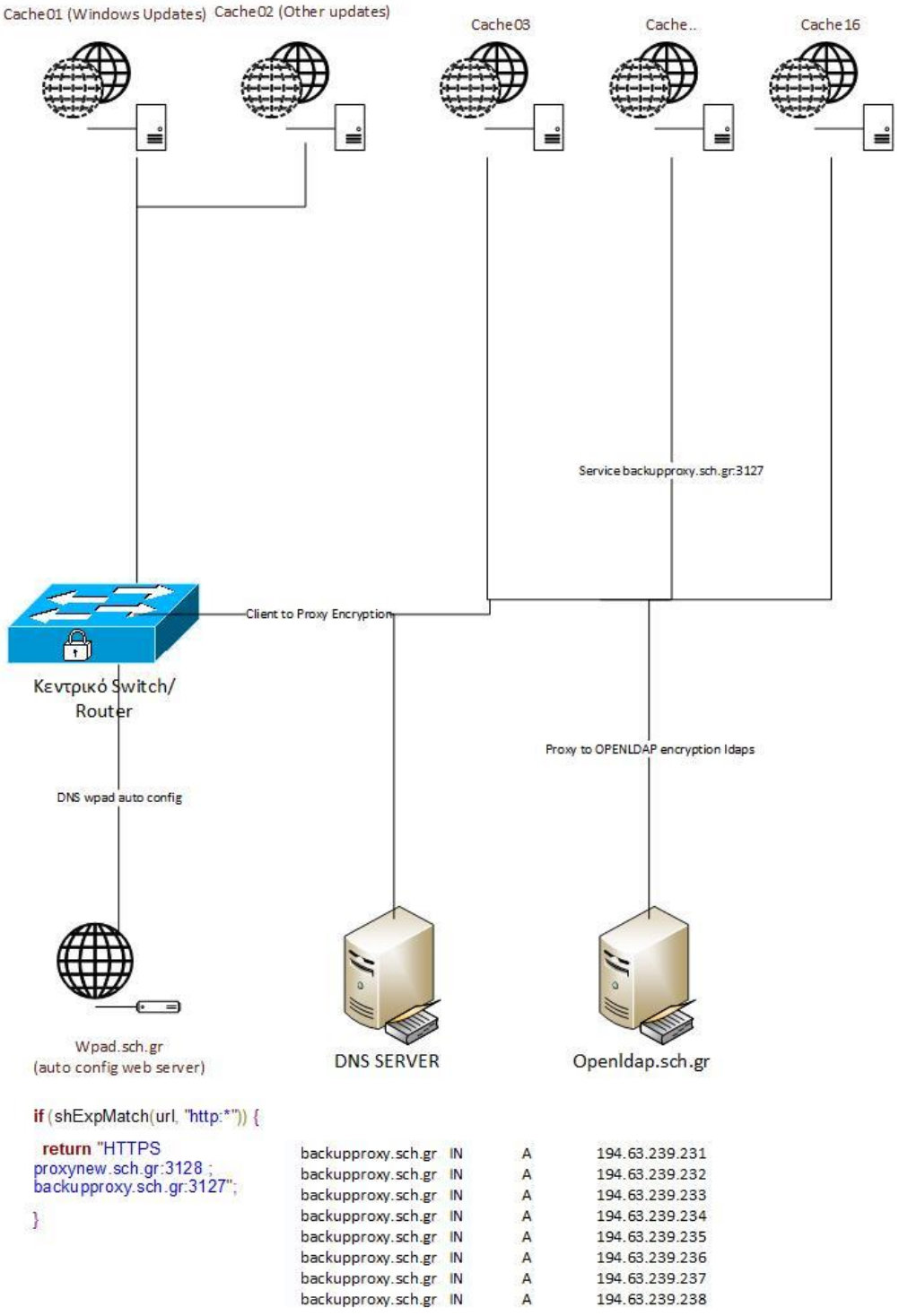
5.5 ΠΡΟΤΕΙΝΟΜΕΝΗ ΛΥΣΗ

Το τελικό σχεδιάγραμμα δικτύου και υπηρεσιών που προτείνεται έχει ως εξής :



Εικόνα 55: Προτεινόμενη λύση

Σε περίπτωση που ο Load balancer βγει εκτός :



Εικόνα 56: Προτεινόμενη λύση (backup)

6 ΣΥΜΠΕΡΑΣΜΑΤΑ

Για να μπορέσει να λειτουργήσει κάποιος SSO (single sign on) μηχανισμός σε συνδυασμό με τον squid θα πρέπει ο χρήστης πελάτης να είναι ήδη ενημερωμένος για αυτό τον μηχανισμό και να έχει ήδη πιστοποιηθεί εκεί ώστε να μπορέσει να αποκτήσει πρόσβαση στον Squid. Ο squid είναι μια εξτρά υπηρεσία σε ένα realm ο οποίος από μόνος του δεν μπορεί να προσφέρει κάποιο μηχανισμό SSO.

Από τις αναλύσεις ο πιο ασφαλής μηχανισμός SSO είναι ο Kerberos. Παρόλα αυτά για να μπορέσει να λειτουργήσει σε συνδυασμό με τον Kerberos ο squid πρώτα απαιτείται ο πελάτης να έχει ήδη αποκτήσει πρόσβαση στον Kerberos Server και να έχει αποκτήσει το Ticket Granting Ticket που αναλύθηκε σε προηγούμενο κεφάλαιο. Μέχρι στιγμής ειδικά στα κινητά δεν υπάρχει κάποιος πελάτης τον οποίο μπορούν να χρησιμοποιήσουν οι χρήστες ώστε να μπορέσουν να κάνουν authenticate αρχικά στον Kerberos Server ώστε να μπορέσουν να αιτηθούν πρόσβαση στην υπηρεσία του Squid.

Ο μηχανισμός bearer OAUTH, παρότι λίγο ασφαλέστερος από τον basic, δεν υποστηρίζεται για τους μηχανισμούς Proxy authenticate από τους browsers προς το παρόν και είναι απίθανο να υποστηριχτεί στο άμεσο μέλλον. Επιπλέον η χρήση του bearer OAUTH δεν πρέπει να χρησιμοποιείται ποτέ χωρίς tls καθώς είναι ευάλωτος σε επιθέσεις. Γενικά ο μηχανισμός αυτός δεν χρησιμοποιείται σε proxy servers καθώς κρίνεται το ίδιο ασφαλής με τον basic (αφού είναι αναγκαίο το tls για την επικοινωνία). Λόγω της αυξημένης πολυπλοκότητας αλλά και της αναγκαιότητας ύπαρξης tls σε όλες τις επικοινωνίες, το bearer OAUTH δεν χρησιμοποιείται ποτέ σε proxy servers, και αυτός είναι μάλλον ο λόγος που οι browsers δεν το υποστηρίζουν.

Ο μηχανισμός digest (παρότι δεν είναι SSO) μπορεί να χρησιμοποιηθεί χωρίς tls με την επικοινωνία προς τον proxy server. Το κυριότερο πρόβλημα του όμως είναι ότι οι περισσότεροι browsers χρησιμοποιούν MD5 για το digest authentication. Το MD5 hash θεωρείται ανασφαλές εδώ και αρκετά χρόνια με αποτέλεσμα το δίκτυο να είναι πλήρως ευάλωτο σε επιθέσεις οι οποίες θα μπορούσαν να ανακτήσουν όλα τα username και password των χρηστών που χρησιμοποιούν τον squid. Ένας επιπλέον αποτρεπτικός παράγοντας είναι ότι για την χρησιμοποίηση του, απαιτείται στο backend (OPENLDAP στην περίπτωσή μας) να αποθηκεύονται τα password των χρηστών σε plaintext.

Επιπροσθέτως η συγκεκριμένη μέθοδος αυξάνει το διαχειριστικό κόστος καθώς οι squid θα πρέπει να συγχρονίζουν τα hash σε τακτά χρονικά διαστήματα με τους ldap servers.

Η μόνη ρεαλιστική επιλογή για BYOD (bring your own device) σενάριο είναι αυτή με basic authentication και TLS μεταξύ πελάτη και proxy. Οι πιο γνωστοί browsers (Firefox & Chrome) υποστηρίζουν αυτού το είδος το σενάριο. Οι περισσότερες καινούριες συσκευές Android επίσης υποστηρίζουν αυτό το σενάριο. Και για όσες κινητές συσκευές δεν υποστηρίζεται υπάρχει η εφαρμογή drony, η οποία χρησιμοποιώντας το wpad αρχείο αυτορρύθμισης, μπορεί να λειτουργήσει με Secure TLS Web Proxy, system wide, για όλες τις εφαρμογές του κινητού (τουλάχιστον για αυτές που χρησιμοποιούν πόρτες οι οποίες θεωρούνται ασφαλής από τον squid, και έχουν ρυθμιστεί ώστε να επιτρέπεται η πρόσβαση σε αυτές). Μοναδικό μειονέκτημα της συγκεκριμένης μεθόδου είναι ότι η Microsoft δεν έχει ακόμα ενσωματώσει την λειτουργία αυτή στους δικούς της browser με αποτέλεσμα να μην μπορούν να λειτουργήσουν σωστά τα windows update.

Για τα updates της Microsoft, των πελατών που χρησιμοποιούν linux καθώς και για updates τρίτων κατασκευαστών έχει ληφθεί ειδική μέριμνα και έχουν διατεθεί 2 από τους 16 squid servers ώστε όλη η κίνηση για αυτά τα updates να περνάει από αυτούς χωρίς authentication. Με τον τρόπο αυτό μπορούμε να έχουμε ταυτόχρονα ελεγχόμενη κίνηση στο δίκτυο μας με authentication και να λειτουργήσουν οι βασικές λειτουργίες των windows.

Για τα updates των windows προτείνεται να δημιουργηθεί νέα υπηρεσία στον load balancer. Η υπηρεσία αυτή μέσω του wpad αρχείου αυτορρύθμισης θα στέλνει όλη αυτή την κίνηση στους squid servers που έχουμε επιλέξει για αυτή τη δουλειά. Επιπροσθέτως σε αυτούς τους squid servers προτείνεται η δημιουργία ενός raid array και η τοπική αποθήκευση περιεχομένου όπως αυτή παρουσιάστηκε νωρίτερα. Έτσι θα μειωθεί αρκετά το bandwidth προς το internet καθαρά για αυτούς τους server και θα αυξηθεί η διακπεραιωτική τους ικανότητα, μειώνοντας και τους χρόνους που λαμβάνουν οι πελάτες τα updates.

Τέλος για την ανάγκη του sslbump (man in the middle), υπάρχουν 2 προϋποθέσεις πριν αυτό εφαρμοστεί. Πρώτη και κυριότερη προϋπόθεση είναι να χρησιμοποιηθούν ssl offload cards οι οποίες είναι συμβατές με το openssl και την έκδοση την οποία θα χρησιμοποιούν οι squid όταν αυτή εφαρμοστεί. Έτσι θα μπορέσουν τα μηχανήματα να αποφορτιστούν από την κρυπτογράφηση/ αποκρυπτογράφηση η οποία είναι απαγορευτική για περισσότερα από 150 αιτήματα/δευτερόλεπτο (150 αιτήματα καθαρά TLS). Επιπλέον για τη λειτουργία αυτή είναι υποχρεωτικό ο κάθε χρήστης πελάτης να έχει

εγκατεστημένο το ριζικό πιστοποιητικό (Root CA) ενός squid server. Το πιστοποιητικό αυτό πρέπει να δημιουργηθεί σε έναν squid server και το ίδιο πιστοποιητικό να χρησιμοποιηθεί και σε όλους τους άλλους. Η διανομή αυτού του πιστοποιητικού σε συστήματα windows μπορεί να γίνει μέσω group policy και στα συστήματα linux απευθείας από κάποιο repository από το οποίο θα κάνουν update. Για τις συσκευές (byod) τελικών χρηστών δεν υπάρχει αυτοματοποιημένος τρόπος εισαγωγής αυτού του πιστοποιητικού και πρέπει να ενημερώνεται ο χρήστης να το κάνει εισαγωγή μόνος του σε περίπτωση που θέλει να χρησιμοποιήσει τις υπηρεσίες. Ένα επιπλέον θέμα που τίθεται είναι σε ποια site μπορεί να γίνεται sslbump. Θα πρέπει site που περιέχουν ευαίσθητα δεδομένα να μην περνάνε από αυτή τη διαδικασία. Για αυτά τα site μπορεί να χρησιμοποιηθεί μηχανισμός bypass του sslbump όπως αναφέρθηκε και σε προηγούμενο κεφάλαιο.

Προτείνεται επίσης να δημιουργηθεί μια νέα υπηρεσία backupproxy.sch.gr. Αυτή η υπηρεσία θα κάνει round robin dns στα μηχανήματα των proxy. Αυτή η υπηρεσία θα χρησιμοποιείται μόνο όταν καταστεί μη αποκρίσιμη η υπηρεσία proxynew.sch.gr. Για να λειτουργήσει αυτή η backup υπηρεσία θα πρέπει να δημιουργηθεί ένα καινούριο πιστοποιητικό το οποίο θα περιέχει και το όνομα της καινούριας υπηρεσίας καθώς και όλων των proxy server που θα αποτελούν μέρος αυτής της υπηρεσίας. Αυτό το πιστοποιητικό θα εγκατασταθεί σε όλους τους proxy servers και αυτοί θα χρησιμοποιούν μια καινούρια πόρτα για αυτή την υπηρεσία, την 3129 με το εξής directive :

```
https_port 443 cert=/usr/local/etc/squid/ssl_cert/pistopoitiko.crt
key=/usr/local/etc/squid/ssl_cert/pistopoiitiko.key
```

Για τη συγκεκριμένη υπηρεσία πρέπει να ληφθεί υπόψιν, ότι πλέον το tls μεταξύ browser και proxy το κάνει τώρα ο ίδιος ο proxy, πράγμα που σημαίνει ότι θα μειωθούν περίπου στο μισό τα αιτήματα/δευτερόλεπτο τα οποία μπορεί να επεξεργαστεί ο κάθε proxy server. Η συγκεκριμένη υπηρεσία παρέχεται μόνο σε περίπτωση που ο load balancer βγει εκτός.

Για να εφαρμοστεί η λύση αυτή πρέπει το ΠΣΔ να δημιουργήσει έναν web server για το wpad.sch.gr , καθώς και να κάνει την εγγραφή στο DNS για το wpad.sch.gr. Επιπλέον πρέπει να φιλοξενηθεί εκεί το αρχείο wpad.dat το οποίο υπάρχει στο παράρτημα 7.5. Προτείνεται μέσω του PAC αρχείου wpad.dat να αρχίσουν να κάνουν το rollout της υπηρεσίας σταδιακά. Αυτό μπορεί να επιτευχθεί ανά δίκτυα (μέσω της συνάρτησης isinnet(host,”10.x.x.x”,”255.255.255.0”). Έτσι μπορούν να αρχίσουν σταδιακά το rollout της καινούριας υπηρεσίας και να εντοπισθούν τυχόν προβλήματα σε ενημερώσεις προϊόντων (εκτός windows και linux) τα οποία δεν είναι ευρέως χρησιμοποιούμενα.

Προτείνεται επίσης η χρησιμοποίηση του load balancer HAProxy όπως αυτός παρουσιάστηκε και με τις ρυθμίσεις που βρίσκονται στο παράρτημα 7.3. Οι δοκιμές που έγιναν στον παραγωγικό load balancer μπόρεσαν αρχικά να λειτουργήσουν μόνο για το πρωτόκολλο http, ενώ παρουσιάστηκαν προβλήματα κατά την λειτουργία στο πρωτόκολλο https, τα οποία πρέπει να επιλυθούν από τους διαχειριστές του δικτύου πριν προχωρήσει η εφαρμογή της λύσης.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Hypertext Transfer Protocol – HTTP/1.1 https://www.ietf.org/rfc/rfc2068.txt

[2] Known HTTP Proxy/Caching Problems https://tools.ietf.org/html/rfc3143

[3] Proxy Server https://en.wikipedia.org/wiki/Proxy_server

[4] Public Key Infrastructure https://en.wikipedia.org/wiki/Public_key_infrastructure

[5] PKI Technical Standards http://www.oasis-pki.org/resources/techstandards/

[6] Understanding HTTP Authentication https://docs.microsoft.com/en-us/dotnet/framework/wcf/feature-details/understanding-http-authentication

[7] HTTP Authentication https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication

[8] TLS Termination Proxy https://en.wikipedia.org/wiki/TLS_termination_proxy

[9] Squid Proxy Server http://www.squid-cache.org/

[10] Basic Authentication https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication

[11] Digest Authentication https://tools.ietf.org/html/rfc2617

[12] The OAuth 2.0 Authorization Framework: Bearer Token Usage self-issued.info/docs/draft-ietf-oauth-v2-bearer.html

[14] Kerberos: The Network Authentication Protocol https://web.mit.edu/kerberos/

[15] An Introduction to HAProxy and Load Balancing Concepts https://www.digitalocean.com/community/tutorials/an-introduction-to-haproxy-and-load-balancing-concepts

[16] Web Polygraph Stress Testing for L4/L7 network appliances http://www.web-polygraph.org/

[17] Kulbir Saini, “Squid Proxy Server 3.1: Beginner's Guide, February 2011

[18] Duane Wessels, “Squid: The Definitive Guide”, 2012

[19] Nick Ramirez, “ Load Balancing with HAProxy: Open-source technology for better scalability, redundancy and availability in your IT infrastructure”, December 2016

[20] Gerardus Blokdyk, “ Proxy Servers, Third Edition”, May 2018

[21] Marshall Kirk McKusick, George V. Neville-Neil, Robert N.M. Watson, " The Design and Implementation of the FreeBSD Operating System (2nd Edition) "

[22] About Implementing WPAD, https://technet.microsoft.com/en-us/library/cc995261.aspx

[23] Web Proxy Autodiscover Protocol, https://tools.ietf.org/html/draft-ietf-wrec-wpad-01

[24] Internet Web Replication and Caching Taxonomy https://www.ietf.org/rfc/rfc3040.txt

[25] Joshua Davies, “Implementing SSL / TLS Using Cryptography and PKI “

[26] Dominique Assing, “ Mobile Access Safety: Beyond BYOD”

7 ΠΑΡΑΡΤΗΜΑ

7.1 WEB POLYGRAPH SCRIPTS & WORKLOADS

7.1.1 Web-Polygraph script για εξαγωγή φορτίου από squid

```
% access-filter --profile server squid-access.log > filtered-access.log
% access-order filtered-access.log | sort ... > ordered-access.log
% access2pgl ordered-access.log > workload.pgd
```

```
# access-filter
# #!/usr/bin/perl -w

use strict;
use integer;

# grok profile
my ($option, $Profile) = @ARGV or
        die("usage: $0 --profile <country|server|content>\n");
die("unsupported option '$option'\n") unless $option eq '--profile';
die("unsupported profile '$Profile'\n") unless
        ($Profile eq 'country' || $Profile eq 'server' || $Profile eq 'content');
shift @ARGV; shift @ARGV;

my %Ips = ();
my %Bads = ();
my %Countries = ();
my %Statuses = ();
my %Protos = ();
my %Methods = ();
my ($cntEntry, $cntGoodEntry, $cntBad,
        $cntIp, $cntGoodIp,
        $cntStatus, $cntGoodStatus,
        $cntUri, $cntGoodUri,
        $cntCountry, $cntGoodCountry,
        $cntMethod, $cntGoodMethod,
        $cntProto, $cntGoodProto) = (0) x 15;

my %GoodCountries = map { ($_ => 1) } qw(US);
my %GoodMethods = map { ($_ => 1) } qw(GET HEAD POST);

my $Registry;

select(STDERR);

while (<>) {
        chomp;
        ++$cntEntry;
        &reportProgress() if $cntEntry % 1000 == 0;

        my @fields = (split);
        my @bad = ();
```

```
        push @bad, 'FC' if @fields < 10;

        # check response status code
        ++$cntStatus;
        my ($sc) = ($fields[3] =~ m|\w+/(\d+)|);
        $sc = '??' unless defined $sc;
        if (defined $Statuses{$sc}) {
                ++$Statuses{$sc};
        } else {
                $Statuses{$sc} = 1;
        }
        my $goodStatus = $Profile eq 'country';
        if ($Profile eq 'server') {
                $goodStatus = $sc ne '??' && ($sc/100 == 2 || $sc/100 == 3);
        }
        elsif ($Profile eq 'content') {
                $goodStatus = $sc eq '200';
        }
        if ($goodStatus) {
                ++$cntGoodStatus;
        } else {
                push @bad, 'SC';
        }

        # check protocol
        ++$cntProto;
        my $uri = $fields[6];
        my ($proto) = ($uri =~ m|(\w+)://|);
        $proto = '??' unless defined $proto;
        if (defined $Protos{$proto}) {
                ++$Protos{$proto};
        } else {
                $Protos{$proto} = 1;
        }
        my $goodProto = $Profile eq 'country' || $proto eq 'http';
        if ($goodProto) {
                ++$cntGoodProto;
        } else {
                push @bad, 'PRT';
        }

        # check URI for query terms
        ++$cntUri;
        if ($Profile ne 'content' || $uri !~ /[\?\&]/) {
                ++$cntGoodUri;
        } else {
                push @bad, 'URI';
        }

        # check request method
        ++$cntMethod;
        my $method = $fields[5];
        $method = '??' unless defined $method;
        if (defined $Methods{$method}) {
                ++$Methods{$method};
        } else {
```

```
		$Methods{$method} = 1;
	}
	my $goodMethod = $Profile eq 'country';
	if ($Profile eq 'server') {
		$goodMethod = exists $GoodMethods{$method};
	}
	elsif ($Profile eq 'content') {
		$goodMethod = $method eq 'GET'
	}
	if ($goodMethod) {
		++$cntGoodMethod;
	} else {
		push @bad, 'MT';
	}

	# check client country code
	++$cntCountry;
	my ($ip, $cc) = ($fields[2] =~ m|([\-\.\d]+)/?(\w+)?|);
	if (!defined $cc && $Profile eq 'country') {
		# init IP registry if needed
		require IP::Country::Fast;
		$Registry = IP::Country::Fast->new() unless $Registry;
		$cc = $Registry ? $Registry->inet_atocc($ip) : '??';
	}
	$cc = '??' unless defined $cc;
	if (defined $Countries{$cc}) {
		++$Countries{$cc};
	} else {
		$Countries{$cc} = 1;
	}
	my $goodCC = $Profile ne 'country';
	if ($Profile eq 'country') {
		$goodCC = !defined(%GoodCountries) || $GoodCountries{$cc};
	}
	if ($goodCC) {
		++$cntGoodCountry;
	} else {
		push @bad, 'CC';
	}

	# maintain an IP:quality map
	if (exists $Ips{$ip}) {
		if ($Ips{$ip}) {
			if (@bad) {
				$Ips{$ip} = 0;
			} else {
				$Ips{$ip} = $fields[0];
			}
		}
		# enable to support good IPs
		# else {
		#	push @bad, 'IP';
		# }
	} else {
		$Ips{$ip} = @bad ? 0 : $fields[0];
		++$cntIp;
```

```
        }

        if (@bad) {
                &recordBads(\@bad);
        } else {
                ++$cntGoodEntry;
        }

        # skip bad entries
        next if @bad;

        # skip bad IPs
        # next unless $Ips{$ip};

        print(STDOUT $_, "\n");
}
&reportProgress();

foreach my $sc (sort { $Statuses{$b} <=> $Statuses{$a} } keys %Statuses) {
        printf("SC: %-3s %6d %6.2f\n", $sc, $Statuses{$sc},
                &percent($Statuses{$sc}, $cntStatus));
}
print("\n");

foreach my $proto (sort { $Protos{$b} <=> $Protos{$a} } keys %Protos) {
        printf("PRT: %-15s %6d %6.2f\n", $proto, $Protos{$proto},
                &percent($Protos{$proto}, $cntProto));
}
print("\n");

printf("URI: %-5s %6d %6.2f\n", 'good', $cntGoodUri,
        &percent($cntGoodUri, $cntUri));
printf("URI: %-5s %6d %6.2f\n", 'bad', $cntUri-$cntGoodUri,
        &percent($cntUri-$cntGoodUri, $cntUri));
print("\n");

foreach my $method (sort { $Methods{$b} <=> $Methods{$a} } keys %Methods) {
        printf("MT: %-10s %6d %6.2f\n", $method, $Methods{$method},
                &percent($Methods{$method}, $cntMethod));
}
print("\n");

foreach my $cc (sort { $Countries{$b} <=> $Countries{$a} } keys %Countries) {
        printf("CC: %2s %6d %6.2f\n", $cc, $Countries{$cc},
                &percent($Countries{$cc}, $cntCountry));
}
print("\n");

printf("entry: %-5s %6d %6.2f\n", 'good', $cntGoodEntry,
        &percent($cntGoodEntry, $cntEntry));
printf("entry: %-5s %6d %6.2f\n", 'bad', $cntEntry-$cntGoodEntry,
        &percent($cntEntry-$cntGoodEntry, $cntEntry));
print("\n");

$cntGoodIp = scalar grep { $_ } values %Ips;
printf("IPs: %-5s %6d %6.2f\n", 'good', $cntGoodIp,
```

```
        &percent($cntGoodIp, $cntIp));
printf("IPs: %-5s %6d %6.2f\n", 'bad', $cntIp-$cntGoodIp,
        &percent($cntIp-$cntGoodIp, $cntIp));
print("\n");

foreach my $bas (sort { $Bads{$b} <=> $Bads{$a} } keys %Bads) {
        printf("Bads: %-3s %6d %6.2f\n", $bas, $Bads{$bas},
                &percent($Bads{$bas}, $cntBad));
}









exit(0);

sub recordBads {
        my $bads = shift;
        foreach my $b (@{$bads}) {
                $Bads{$b} = 0 unless defined $Bads{$b};
                ++$Bads{$b};
                ++$cntBad;
        }
}

sub reportProgress {
        printf("#Klines: %03d IPs: %3d  SC: %6.2f PRT: %6.2f URI: %6.2f MT: %6.2f CC:
%6.2f\n",
                $cntEntry/1000,
                $cntIp,
                &percent($cntGoodStatus, $cntStatus),
                &percent($cntGoodProto, $cntProto),
                &percent($cntGoodUri, $cntUri),
                &percent($cntGoodMethod, $cntMethod),
                &percent($cntGoodCountry, $cntCountry));
}

sub percent {
        my ($part, $whole) = @_;
        $whole = $cntEntry unless defined $whole;
        return -1 unless $whole && defined($part);
        no integer;
        return 100. * $part/$whole;
}
```

```
#access-order
#!/usr/bin/perl -w

while (<>) {
        my @fields = split;
        $fields[0] -= $fields[1]/1000;
        print(map { "$_ " } @fields);
        print("\n");
}
```

```
#!/usr/bin/perl -w
#access2pgl
#
#
#% access2order access.log | sort -t' ' -n +0
#

use strict;

my $SessionIdleTout = 1*60*1000.; # when a busy session ends

my %Ds = (
        InterArrival => &newTimeDistr('my_req_inter_arrival', 'Request interarrival times
during busy periods'),
        SessionBusyDur => &newTimeDistr('my_session_busy_period', 'Duration of a busy
session period'),
        SessionBusyCount => &newNumDistr('my_session_busy_count', 'Number of
requests per busy session period'),
        SessionIdleDur => &newTimeDistr('my_session_idle_period', 'Duration of an idle
session period'),

        Rptm => &newTimeDistr('my_think_time', 'Response times'),
        RequestHeaderSize => &newSizeDistr('my_req_header_size', 'Request header
sizes'),
        RequestBodySize => &newSizeDistr('my_req_content_size', 'Request body sizes'),
        ResponseSize => &newSizeDistr('my_resp_size', 'Response sizes'),
        StatusCodes => &newEventsDistr('my_resp_codes', 'Response status codes'),
#       RequestTypes => &newEventsDistr('my_req_types', 'Request types'),
        RequestMethods => &newEventsDistr('my_req_methods', 'Request methods'),
);

my %Ips = ();
my ($cntEntry, $cntIp) = (0) x 2;
$| = 1;

while (<>) {
        chomp;
        ++$cntEntry;
        &reportProgress() if $cntEntry % 1000 == 0;

        my @fields = (split);
```

```
        my $rptm = $fields[1];
        my $time = $fields[0];
        my $ip = $fields[2];
        my ($result, $scode) = split(m|/|, $fields[3]);

        if (exists $Ips{$ip}) {
                my $last = $Ips{$ip}->{last};
                die("access log not sorted by request time, stopped")
                        if $time < $last;

                &updateDistr($Ds{Rptm}, $rptm) if $scode == 200 || $scode == 304;
                &updateDistr($Ds{RequestHeaderSize}, $fields[10]);
                &updateDistr($Ds{RequestBodySize}, $fields[11]);
                &updateDistr($Ds{ResponseSize}, $fields[4]) if $scode == 200;
                &updateDistr($Ds{StatusCodes}, $scode);
                &updateDistr($Ds{RequestMethods}, $fields[5]);

                my $gap = 1000.*($time - $last);
                if (!defined $SessionIdleTout || $gap < $SessionIdleTout) {
                        &updateDistr($Ds{InterArrival}, $gap);
                } else {
                        &updateDistr($Ds{SessionBusyCount},
                                $Ips{$ip}->{busy_count});
                        &updateDistr($Ds{SessionBusyDur},
                                1000.*($last - $Ips{$ip}->{busy_start}));
                        &updateDistr($Ds{SessionIdleDur}, 1000.*($time - $last));
                        $Ips{$ip}->{busy_start} = $time;
                        $Ips{$ip}->{busy_count} = 0;
                }
                $Ips{$ip}->{last} = $time;
                $Ips{$ip}->{busy_count}++;
        } else {
                ++$cntIp;
                $Ips{$ip} = {
                        last => $time,
                        busy_start => $time,
                        busy_count => 1,
                }
        }
}
&reportProgress();

map { &reportDistr($_) } sort { $a->{id} cmp $b->{id} } values %Ds;

exit(0);

sub newEventsDistr {
        my ($id, $name) = @_;
        return &newDistr($id, $name, [
                &newArea('all', undef(), undef()),
        ]);
}

sub newTimeDistr {
        my ($id, $name) = @_;
        my $distr = &newDistr($id, $name, [
```

```
                &newArea('frequent', 1000, 1),
                &newArea('medium', 10*1000, 10),
                &newArea('occasional', 100*1000, 100),
        ]);
        $distr->{pgl_type} = 'time_distr';
        $distr->{report_factor} = 1000.0; # convert to seconds
        $distr->{report_unit} = 'seconds';
        return $distr;
}

sub newSizeDistr {
        my ($id, $name) = @_;
        my $distr = &newDistr($id, $name, [
                &newArea('tiny',       1024,     1),
                &newArea('small',    10*1024,    10),
                &newArea('medium',  100*1024,   100),
                &newArea('large',  1000*1024,  1000),
                &newArea('huge',  10000*1024, 10000),
        ]);
        $distr->{pgl_type} = 'size_distr';
        $distr->{report_unit} = 'bytes';
        return $distr;
}

sub newNumDistr {
        my $distr = &newSizeDistr(@_);
        $distr->{pgl_type} = 'num_distr';
        $distr->{report_unit} = 'number';
        return $distr;
}

sub newDistr {
        my ($id, $name, $areas) = @_;

        my $d = {
                id => $id,
                name => $name,
                pgl_type => undef(),
                report_factor => undef(),
                report_unit => undef(),

                areas => $areas,

                count => 0,
                sum => 0,
                sqSum => 0,
        };

        # assign minimums
        my $lastMax;
        foreach my $area (@{$d->{areas}}) {
                $area->{min} = $lastMax if defined $lastMax;
                $lastMax = $area->{max};
        }

        return $d;
```

```
}

sub newArea {
	my ($name, $max, $factor) = @_;
	return {
		name => $name,
		min => undef(),
		max => $max,
		factor => $factor,
		values => {},
	};
}

sub updateDistr {
	my ($distr, $value) = @_;
	return unless defined $value && $value ne '-';

	# find matching area
	my $area;
	foreach $a (@{$distr->{areas}}) {
		if (defined $area) {
			$area = $a if defined $a->{min} && $value >= $a->{min};
		} else {
			$area = $a;
		}
	}
	die("no matching area for $value in ". $distr->{name}. " distro, stopped")
		unless $area;

	if (defined $area->{factor}) {
		$distr->{sum} += $value;
		$distr->{sqSum} += $value * $value;

		$value = int($value / $area->{factor});
	}

	$distr->{count}++;

	if (defined $area->{values}->{$value}) {
		$area->{values}->{$value}++;
	} else {
		$area->{values}->{$value} = 1;
	}
}

sub reportDistr {
	my ($distr) = @_;

	printf("# %s\n", $distr->{name});
	printf("#\tcount:   %10d\n", $distr->{count});

	if (defined $distr->{areas}->[0]->{factor}) {
		&reportNumDistr($distr);
	} else {
		&reportEventDistr($distr);
	}
```

```
	printf("\n");
}

sub reportNumDistr {
	my ($distr) = @_;

	if ($distr->{count}) {
		my $mean = $distr->{sum}/$distr->{count};
		my $dev;
		if ($distr->{count} > 1) {
			my $diff = $distr->{sqSum} -
				$distr->{sum}*$distr->{sum}/$distr->{count};
			$dev = sqrt($diff / ($distr->{count}-1));
		}
		my $median = &distrPercentile($distr, 50.0);

		printf("#\tmedian:  %s\n", &distrValue($distr, $median));
		printf("#\tmean:    %s\n", &distrValue($distr, $mean));
		printf("#\tstd_dev: %s\n", &distrValue($distr, $dev)) if defined $dev;
		printf("#\trel_dev: %14.3f%%\n", &percent($dev, $mean)) if $mean > 0;
	}
	printf("#\tunit:    %10s\n", $distr->{report_unit});

	printf("%s %s = {\n", $distr->{pgl_type}, $distr->{id});

	my $sum = 0;
	foreach my $a (@{$distr->{areas}}) {
		&reportNumArea($distr, $a, \$sum);
	}

	printf("}\n");
}

sub reportNumArea {
	my ($distr, $area, $sumPtr) = @_;

	my @keys = sort { $a <=> $b } keys %{$area->{values}};
	my $bin = { min => undef(), max => undef(), count => 0 };
	foreach my $v (@keys) {
		my $c = $area->{values}->{$v};
		my $value = int($v * $area->{factor});
		&nextBin($distr, $bin, ${$sumPtr})
			if ($bin->{count} + $c) >= ($distr->{count}/100.);
		$bin->{count} += $c;
		$bin->{min} = $value unless defined $bin->{min};
		$bin->{max} = $value;
		${$sumPtr} += $c;
	}
	&nextBin($distr, $bin, ${$sumPtr}) if $bin->{count};
}

sub nextBin {
	my ($distr, $bin, $sum) = @_;
	return unless $bin->{count};
```

```
		my ($min, $max) = map { &distrValue($distr, $_) }
			($bin->{min}, $bin->{max});

		printf("\t%s : %s %10.3f # %10.3f\n", $min, $max,
			&percent($bin->{count}, $distr->{count}),
			&percent($sum, $distr->{count}));

		$bin->{count} = 0;
		$bin->{min} = $bin->{max} = undef();
}

sub distrPercentile {
		my ($distr, $level) = @_;

		my $sum = 0;
		my $last;
		foreach my $area (@{$distr->{areas}}) {
			my @keys = sort { $a <=> $b } keys %{$area->{values}};
			foreach my $v (@keys) {
				$sum += $area->{values}->{$v};
				my $value = int($v * $area->{factor});
				return $value if &percent($sum, $distr->{count}) >= $level;
				$last = $value;
			}
		}
		return $last;
}

sub distrValue {
		my ($distr, $v) = @_;
		my $value = $distr->{report_factor} ? $v/$distr->{report_factor} : $v;
		my $f = $distr->{report_factor} ? '%14.3f' : '%10d';
		return sprintf($f, $value);
}

sub reportEventDistr {
		my ($distr) = @_;

		printf("%s = [\n", $distr->{id});
		my $area = $distr->{areas}->[0];

		my @keys = sort { $area->{values}->{$b} <=> $area->{values}->{$a} }
			keys %{$area->{values}};

		my $count = 0;
		foreach my $v (@keys) {
			if (my $c = $area->{values}->{$v}) {
				my $value = sprintf("'%s'", $v);
				if ($count == 0) {
					printf("\t%-10s", $value); # let most frequent entry absorb cal
mistakes
				} else {
					printf(",\n") if $count;
					printf("\t%-10s : %.3f%%", $value, &percent($c, $distr-
>{count}));
				}
```

```
                        ++$count;
                }
        }

        printf("\n];\n", $distr->{id});
}

sub reportProgress {
        printf(STDERR "#%03dK IPs: %3d\n", $cntEntry/1000, $cntIp);
}

sub percent {
        my ($part, $whole) = @_;
        die() unless defined $whole;
        return -1 unless $whole > 0 && defined($part);
        no integer;
        return 100. * $part/$whole;
}
```

7.1.2 Web-Polygraph Workload Emulation

```
// Η μορφή του αρχείου είναι πίνακες του τύπου πχ size_distribution με την μορφή
//απομέγεθος:μέχριμέγεθος ποσοστό% . Η τελευταία στήλη αθροίζει τα ποσοστά αθροιστικά μέχρι
//εκείνο το σημείο
```

```
#       median:      2090
#       mean:       25509
#       std_dev:   511091
#       rel_dev:    2003.519%
#       unit:       bytes
size_distr my_resp_size = {
            0 :      258    0.227 #    0.227
          259 :      265    0.999 #    1.226
          266 :      290    0.937 #    2.163
          291 :      293    0.930 #    3.093
          294 :      309    0.992 #    4.085
          310 :      346    0.991 #    5.076
          347 :      384    0.922 #    5.998
          385 :      422    0.994 #    6.992
          423 :      461    0.991 #    7.983
          462 :      498    0.945 #    8.928
          499 :      517    0.992 #    9.920
          518 :      550    0.989 #   10.909
          551 :      594    0.883 #   11.792
          595 :      628    0.993 #   12.785
          629 :      649    0.978 #   13.763
          650 :      671    0.924 #   14.687
          672 :      681    0.961 #   15.648
          682 :      697    0.599 #   16.247
          698 :      699    0.733 #   16.980
          700 :      701    0.894 #   17.874
          702 :      705    0.783 #   18.657
          706 :      706    1.423 #   20.080
          707 :      707    1.518 #   21.598
          708 :      710    0.985 #   22.583
```

711 : 753 0.986 # 23.570
754 : 799 0.926 # 24.496
800 : 841 0.994 # 25.490
842 : 886 0.947 # 26.437
887 : 923 0.931 # 27.368
924 : 966 0.992 # 28.361
967 : 978 0.749 # 29.110
979 : 998 0.946 # 30.056
999 : 1000 0.946 # 31.001
1001 : 1010 0.885 # 31.886
1011 : 1018 0.826 # 32.712
1019 : 1023 0.474 # 33.186
1020 : 1020 0.643 # 33.830
1030 : 1030 0.989 # 34.819
1040 : 1040 1.698 # 36.516
1050 : 1050 1.796 # 38.312
1060 : 1100 0.907 # 39.219
1110 : 1150 0.980 # 40.198
1160 : 1220 0.987 # 41.186
1230 : 1310 0.973 # 42.159
1320 : 1400 0.941 # 43.100
1410 : 1480 0.925 # 44.025
1490 : 1570 0.913 # 44.938
1580 : 1680 0.930 # 45.868
1690 : 1770 0.995 # 46.863
1780 : 1840 0.888 # 47.752
1850 : 1970 0.949 # 48.701
1980 : 2050 0.683 # 49.384
2060 : 2130 0.958 # 50.342
2140 : 2230 0.955 # 51.296
2240 : 2380 0.982 # 52.278
2390 : 2490 0.753 # 53.032
2500 : 2610 0.949 # 53.981
2620 : 2770 0.936 # 54.917
2780 : 2970 0.939 # 55.856
2980 : 3150 0.968 # 56.825
3160 : 3360 0.962 # 57.787
3370 : 3560 0.994 # 58.780
3570 : 3720 0.980 # 59.761
3730 : 3960 0.947 # 60.708
3970 : 4190 0.997 # 61.705
4200 : 4460 0.982 # 62.686
4470 : 4740 0.990 # 63.676
4750 : 5000 0.996 # 64.672
5010 : 5290 0.998 # 65.670
5300 : 5600 0.998 # 66.668
5610 : 5990 0.987 # 67.654
6000 : 6340 0.985 # 68.640
6350 : 6810 0.997 # 69.637
6820 : 7380 0.993 # 70.630
7390 : 7800 0.987 # 71.616
7810 : 8290 0.989 # 72.605
8300 : 8900 0.991 # 73.597
8910 : 9760 0.976 # 74.573
9770 : 10230 0.654 # 75.227
10200 : 11200 0.998 # 76.224

```
            11300 :      12300      0.953 #     77.177
            12400 :      13400      0.959 #     78.136
            13500 :      14500      0.972 #     79.108
            14600 :      15400      0.989 #     80.097
            15500 :      16000      0.990 #     81.087
            16100 :      17200      0.939 #     82.026
            17300 :      18500      0.983 #     83.008
            18600 :      20000      0.948 #     83.956
            20100 :      21700      0.948 #     84.904
            21800 :      23200      0.987 #     85.891
            23300 :      25500      0.991 #     86.882
            25600 :      28400      0.996 #     87.878
            28500 :      31600      0.996 #     88.874
            31700 :      34500      0.994 #     89.867
            34600 :      38200      0.995 #     90.862
            38300 :      43700      0.983 #     91.845
            43800 :      49800      0.996 #     92.841
            49900 :      58300      1.000 #     93.841
            58400 :      69700      0.994 #     94.836
            69800 :      86100      0.999 #     95.835
            86200 :     102300      0.731 #     96.566
           102000 :     139000      0.991 #     97.557
           140000 :     228000      0.994 #     98.552
           229000 :     565000      0.999 #     99.551
           566000 :    1023000      0.202 #     99.753
          1020000 :  485340000      0.247 #    100.000
}

             63.100 :       67.000      1.000 #     79.969
             67.100 :       71.600      0.989 #     80.958
             71.700 :       76.600      0.984 #     81.942
             76.700 :       82.000      0.993 #     82.935
             82.100 :       88.000      0.995 #     83.930
             88.100 :       94.400      0.991 #     84.921
             94.500 :      101.500      0.997 #     85.919
            101.600 :      109.700      0.991 #     86.909
            109.800 :      118.800      0.996 #     87.905
            118.900 :      129.100      0.997 #     88.902
            129.200 :      142.200      0.992 #     89.894
            142.300 :      158.100      0.997 #     90.891
            158.200 :      177.600      0.999 #     91.890
            177.700 :      202.000      0.997 #     92.887
            202.100 :      236.500      0.998 #     93.885
            236.600 :      286.100      0.997 #     94.882
            286.200 :      360.700      0.999 #     95.881
            360.900 :      498.700      1.000 #     96.881
            498.800 :      768.700      0.999 #     97.880
            768.800 :     1502.600      1.000 #     98.880
           1503.800 :     6411.900      1.000 #     99.880
           6435.800 :    28521.500      0.120 #    100.000
}

# Duration of an idle session period
#          count:      217444
#          median:        187.600
#          mean:          617.637
```

```
#           std_dev:      1559.382
#           rel_dev:      252.476%
#           unit:       seconds
time_distr my_session_idle_period = {
                60.000 :       60.600     0.929 #      0.929
                60.700 :       61.500     0.991 #      1.920
                61.600 :       62.400     0.935 #      2.855
                62.500 :       63.500     0.987 #      3.842
                63.600 :       64.500     0.947 #      4.789
                64.600 :       65.600     0.980 #      5.769
                65.700 :       66.800     0.993 #      6.763
                66.900 :       68.000     0.952 #      7.715
                68.100 :       69.200     0.992 #      8.708
                69.300 :       70.500     0.954 #      9.661
                70.600 :       71.800     0.969 #     10.631
                71.900 :       73.100     0.927 #     11.557
                73.200 :       74.500     0.958 #     12.515
                74.600 :       76.000     0.968 #     13.483
                76.100 :       77.600     0.975 #     14.458
                77.700 :       79.200     0.990 #     15.448
                79.300 :       80.800     0.979 #     16.427
                80.900 :       82.500     0.998 #     17.425
                82.600 :       84.200     0.945 #     18.370
                84.300 :       86.000     0.991 #     19.361
                86.100 :       87.800     0.972 #     20.333
                87.900 :       89.800     0.983 #     21.316
                89.900 :       91.500     0.983 #     22.300
                91.600 :       93.400     0.961 #     23.261
                93.500 :       95.400     0.975 #     24.235
                95.500 :       97.500     0.979 #     25.214
                97.600 :       99.600     0.966 #     26.180
                99.700 :      101.800     0.987 #     27.167
               101.900 :      104.200     0.996 #     28.163
               104.300 :      106.700     0.984 #     29.147
               106.800 :      109.300     0.986 #     30.133
               109.400 :      111.900     0.986 #     31.118
               112.000 :      114.700     0.981 #     32.100
               114.800 :      117.700     0.998 #     33.098
               117.800 :      120.400     0.987 #     34.085
               120.500 :      123.300     0.992 #     35.076
               123.400 :      126.500     0.992 #     36.068
               126.600 :      129.800     0.992 #     37.060
               129.900 :      133.300     0.989 #     38.049
               133.400 :      136.900     0.978 #     39.027
               137.000 :      140.600     0.978 #     40.005
               140.700 :      144.400     0.979 #     40.984
               144.500 :      148.400     1.000 #     41.984
               148.500 :      152.400     0.995 #     42.978
               152.500 :      156.700     0.996 #     43.975
               156.800 :      161.100     0.986 #     44.961
               161.200 :      166.000     0.989 #     45.950
               166.100 :      171.000     0.993 #     46.943
               171.100 :      176.200     0.987 #     47.930
               176.300 :      181.400     0.991 #     48.921
               181.500 :      186.900     0.980 #     49.902
               187.000 :      192.800     0.993 #     50.894
```

```
                192.900 :       198.800      0.995 #     51.889
                198.900 :       205.000      0.983 #     52.872
                205.100 :       211.800      0.995 #     53.867
                211.900 :       219.000      0.990 #     54.857
                219.100 :       226.500      1.000 #     55.857
                226.600 :       234.200      0.993 #     56.850
                234.300 :       241.600      0.984 #     57.834
                241.700 :       249.600      1.000 #     58.834
                249.700 :       258.100      0.993 #     59.827
                258.200 :       267.700      0.992 #     60.818
                267.800 :       277.200      1.000 #     61.818
                277.300 :       287.600      0.995 #     62.813
                287.700 :       298.000      0.999 #     63.813
                298.100 :       306.200      0.991 #     64.803
                306.300 :       316.100      0.999 #     65.802
                316.200 :       328.100      0.994 #     66.796
                328.200 :       341.600      0.997 #     67.793
                341.700 :       356.000      0.998 #     68.791
                356.100 :       370.800      0.996 #     69.787
                370.900 :       387.600      0.998 #     70.785
                387.700 :       404.500      0.998 #     71.783
                404.600 :       421.400      0.999 #     72.782
                421.500 :       440.600      0.997 #     73.780
                440.700 :       462.700      0.998 #     74.778
                462.800 :       484.900      0.999 #     75.777
                485.000 :       510.800      0.997 #     76.774
                510.900 :       538.800      0.999 #     77.773
                538.900 :       569.500      0.998 #     78.771
                569.600 :       600.200      0.988 #     79.759
                600.300 :       620.000      0.999 #     80.758
                620.100 :       659.900      0.997 #     81.756
                660.000 :       707.200      0.998 #     82.754
                707.300 :       755.900      0.999 #     83.753
                756.000 :       814.500      0.999 #     84.752
                814.600 :       879.000      0.999 #     85.751
                879.100 :       938.800      0.999 #     86.751
                938.900 :      1021.100      1.000 #     87.750
               1021.200 :      1124.400      1.000 #     88.750
               1124.500 :      1235.100      0.999 #     89.749
               1235.200 :      1369.400      0.999 #     90.748
               1369.500 :      1533.800      1.000 #     91.748
               1533.900 :      1740.600      0.999 #     92.747
               1740.700 :      1983.100      0.999 #     93.746
               1983.200 :      2341.400      1.000 #     94.746
               2341.500 :      2815.700      1.000 #     95.746
               2816.200 :      3516.700      1.000 #     96.746
               3516.900 :      4588.400      1.000 #     97.746
               4588.900 :      6855.100      1.000 #     98.745
               6858.100 :     14748.000      1.000 #     99.745
              14751.000 :     36041.900      0.255 #    100.000
}

# Response times
#           count:     3468440
#           median:          0.102
#           mean:            1.312
```

```
#           std_dev:        12.913
#           rel_dev:        983.861%
#           unit:       seconds
time_distr my_think_time = {
                    0.001 :         0.003     0.836 #      0.836
                    0.004 :         0.004     2.103 #      2.938
                    0.005 :         0.005     3.734 #      6.672
                    0.006 :         0.006     4.425 #     11.097
                    0.007 :         0.007     2.884 #     13.981
                    0.008 :         0.008     1.665 #     15.645
                    0.009 :         0.009     1.131 #     16.777
                    0.010 :         0.010     0.839 #     17.616
                    0.011 :         0.011     0.643 #     18.260
                    0.012 :         0.013     0.924 #     19.184
                    0.014 :         0.016     0.839 #     20.023
                    0.017 :         0.022     0.962 #     20.985
                    0.023 :         0.029     0.921 #     21.906
                    0.030 :         0.030     1.311 #     23.217
                    0.031 :         0.031     1.886 #     25.103
                    0.032 :         0.032     1.740 #     26.843
                    0.033 :         0.033     1.087 #     27.929
                    0.034 :         0.035     0.980 #     28.910
                    0.036 :         0.039     0.874 #     29.783
                    0.040 :         0.044     0.931 #     30.714
                    0.045 :         0.045     0.486 #     31.200
                    0.046 :         0.046     0.588 #     31.788
                    0.047 :         0.047     0.555 #     32.344
                    0.048 :         0.048     0.616 #     32.960
                    0.049 :         0.049     0.804 #     33.764
                    0.050 :         0.050     0.803 #     34.567
                    0.051 :         0.051     0.696 #     35.262
                    0.052 :         0.052     0.839 #     36.101
                    0.053 :         0.053     0.926 #     37.027
                    0.054 :         0.054     0.755 #     37.782
                    0.055 :         0.055     0.834 #     38.615
                    0.056 :         0.056     0.870 #     39.485
                    0.057 :         0.057     0.763 #     40.248
                    0.058 :         0.058     0.595 #     40.842
                    0.059 :         0.059     0.552 #     41.395
                    0.060 :         0.061     0.990 #     42.385
                    0.062 :         0.063     0.972 #     43.357
                    0.064 :         0.065     0.754 #     44.111
                    0.066 :         0.069     0.904 #     45.015
                    0.070 :         0.074     0.946 #     45.961
                    0.075 :         0.080     0.899 #     46.861
                    0.081 :         0.088     0.946 #     47.807
                    0.089 :         0.093     0.839 #     48.646
                    0.094 :         0.098     0.897 #     49.543
                    0.099 :         0.102     0.748 #     50.291
                    0.103 :         0.104     0.944 #     51.235
                    0.105 :         0.106     0.859 #     52.094
                    0.107 :         0.110     0.829 #     52.923
                    0.111 :         0.116     0.922 #     53.845
                    0.117 :         0.122     0.973 #     54.818
                    0.123 :         0.130     0.951 #     55.769
                    0.131 :         0.136     0.856 #     56.625
```

```
    0.137 :    0.142  0.988 #   57.613
    0.143 :    0.152  0.998 #   58.611
    0.153 :    0.163  0.967 #   59.578
    0.164 :    0.172  0.966 #   60.544
    0.173 :    0.186  0.960 #   61.504
    0.187 :    0.200  0.995 #   62.499
    0.201 :    0.212  0.961 #   63.460
    0.213 :    0.226  0.988 #   64.448
    0.227 :    0.235  0.849 #   65.296
    0.236 :    0.246  0.958 #   66.254
    0.247 :    0.262  0.980 #   67.235
    0.263 :    0.267  0.990 #   68.225
    0.268 :    0.279  0.952 #   69.177
    0.280 :    0.293  0.952 #   70.129
    0.294 :    0.302  0.935 #   71.064
    0.303 :    0.312  0.996 #   72.060
    0.313 :    0.325  0.988 #   73.048
    0.326 :    0.340  0.970 #   74.018
    0.341 :    0.359  0.962 #   74.980
    0.360 :    0.379  0.968 #   75.948
    0.380 :    0.399  0.964 #   76.912
    0.400 :    0.421  0.963 #   77.876
    0.422 :    0.446  0.988 #   78.863
    0.447 :    0.475  0.991 #   79.854
    0.476 :    0.515  0.989 #   80.843
    0.516 :    0.573  0.993 #   81.836
    0.574 :    0.640  0.993 #   82.829
    0.641 :    0.723  0.991 #   83.820
    0.724 :    0.766  0.993 #   84.813
    0.767 :    0.784  0.998 #   85.811
    0.785 :    0.838  0.992 #   86.804
    0.839 :    0.999  0.871 #   87.675
    1.000 :    1.240  0.977 #   88.652
    1.250 :    1.550  0.970 #   89.622
    1.560 :    2.110  0.996 #   90.618
    2.120 :    3.010  0.989 #   91.608
    3.020 :    3.100  0.947 #   92.555
    3.110 :    3.250  0.958 #   93.513
    3.260 :    3.450  0.987 #   94.500
    3.460 :    3.770  0.980 #   95.480
    3.780 :    4.580  0.996 #   96.475
    4.590 :    7.060  0.999 #   97.474
    7.070 :    9.990  0.621 #   98.095
   10.000 :   21.400  0.999 #   99.094
   21.500 : 4052.400  0.906 #  100.000
}
```

7.1.3 Web Polygraph Main Config

```
#include "/usr/local/share/polygraph/workloads/contents.pg"



// Ρύθμιση για test tls man in the middle με authentication
// Για απλό χωρίς ssl σχολιάζω το ssl_wraps σε robots , server
// και Proxy. Το ssl_wrap στον Proxy σημαίνει Secure TLS Proxy
SslWrap sslWrap = {
    ssl_config_file = "openssl.conf";
    root_certificate = "keyall.pem";
    session_resumption = 70%;
    session_cache = 100;
};

Content SimpleContent = {
  size = table("/root/work/size.pgd", "size");
  cachable = 80%;   // 20% of content is uncachable
};

Server S = {
  kind = "PolyMix-4-srv";

  //contents      = [ cntImage: 65%, cntHTML: 15%, cntDownload: 0.5%, cntOther ];
  //direct_access = [ cntHTML, cntDownload, cntOther ];
  contents = [ SimpleContent ];
  direct_access = contents;
  xact_think = norm(2.5sec, 1sec);
  pconn_use_lmt = zipf(16);
  idle_pconn_tout = 15sec;
  abort_prob = 0.1%;
  addresses = ['194.63.239.236:999' ];
  http_versions = [ "1.0" ];  // newer agents use HTTP/1.1 by default
};

PopModel popModel = {
  pop_distr = popUnif();
  hot_set_frac =  1%;  // fraction of WSS (i.e., hot_set_size / WSS)
  hot_set_prob = 10%;  // prob. of req. an object from the hot set

  bhr_discrimination = 90%; // revisit smaller files more often
};
Proxy P = {
  addresses = ['194.63.239.234:443'];
   ssl_wraps = [ sslWrap ];
  };
float HitIfRepeat = 80%;
string[] mycreds = "testproxystudent[0001-1000]:testproxy";
```

```
      Robot R = {
        ssl_wraps = [ sslWrap ];
        kind = "PolyMix-4-rbt";
        recurrence = 55% / SimpleContent.cachable;
        //recurrence      =  55%/HitIfRepeat; // recurrence is not hit ratio
        embed_recur     = 100%;
        pop_model = popModel;

        req_types = [ "Basic", "Ims200": 5%, "Ims304": 10%, "Reload" : 5% ];
        req_methods = [ "GET", "POST" : 1.5%, "HEAD" : 0.1% ];
        abort_prob = 0.1%;
        open_conn_lmt = 4;
        http_proxies = P.addresses;
        credentials = [ "user:password" ];
        req_rate = 1 /sec;
        origins = S.addresses;      // where the origin servers are
        addresses = ['194.63.239.236' ** 920 ]; // where these robot agents will be
created
        http_versions = [ "1.0" ];  // newer agents use HTTP/1.1 by default

        //post_contents = [ cntSimpleRequest ];
      };
      // commit to using these servers and robots
      use(S, R);
```

7.2 SQUID MODULAR CONFIGURATION

7.2.1 General squid.conf

```
workers 5
#debug_options ALL,4
debug_options 55,0 ALL,1

client_db off
#reply_header_access Server deny all
#reply_header_replace Server proxynew.sch.gr/1.1

#Mode (mode1.conf) 1 Transparent , (mode2.conf) 2 Forward , 3 SSL BUMP, 4 TLS Secure
Webproxy
#All work on port 3128
#max_open_disk_fds 0
include /usr/local/etc/squid/mode3.conf
include /usr/local/etc/squid/general.conf
#include /usr/local/etc/squid/webfilter.conf
#Access Lists directives
#http_access allow all
include /usr/local/etc/squid/acls.conf
#Authentication on (authon.conf),Off (authoff.conf)
include /usr/local/etc/squid/authon.conf
#snmp.conf access
include /usr/local/etc/squid/snmp.conf

refresh_pattern ^ftp:           1440    20%     10080
refresh_pattern ^gopher:        1440    0%      1440
refresh_pattern -i (/cgi-bin/|\?) 0  0%      0
refresh_pattern .               0       20%     4320
#maximum_single_addr_tries 2
uri_whitespace encode
cache_mgr cachemaster@sch.gr
cachemgr_passwd disable shutdown objects
request_header_max_size 64 KB
max_filedescriptors 62000
```

7.2.2 Mode1.conf (Transparent)

```
#transparent mode
http_port 3128 intercept
```

7.2.3 Mode2.conf (Forward)

```
#forward mode
http_port 3128
https_port 3129 cert=/usr/local/etc/squid/ssl_cert/cache04.crt
key=/usr/local/etc/squid/ssl_cert/cache04.key
```

7.2.4 Mode3.conf (SSL BUMP)

```
#Deep Inspection MITM
#openssl req -new -newkey rsa:2048 -sha256 -days 365 -nodes -x509 -extensions v3_ca -
keyout proxyCA.pem  -out proxyCA.pem
#(αυτό πρέπει να γίνει import στον χρήστη σαν
#root trusted CA)
#openssl x509 -in proxyCA.pem -outform DER -out proxyCA.der
#Πριν χρησιμοποιηθεί πρέπει να αρχικοποιηθεί το directory
#με τα cached πιστοποιητικά ως εξής
#/usr/local/squid/libexec/ssl_crtd -c -s /var/lib/ssl_db
# chown squid:squid -R /var/lib/ssl_db
http_port 3128 ssl-bump \
  cert=/usr/local/etc/squid/proxyCA.pem \
  generate-host-certificates=on dynamic_cert_mem_cache_size=4MB
sslcrtd_program /usr/local/libexec/squid/ssl_crtd -s /var/ssl_db -M 4MB
sslcrtd_children 100   startup=1 idle=1

#προσοχή στα children είναι 100 children/ worker ,
#με 5 workers είναι 500 children με 4MB το καθένα
# ΠΡΟΣΟΧΗ στη μνήμη αν αυξηθούν πολύ μπορεί να
#crasharei το μηχάνημα με swap error

acl serverIsBank ssl::server_name .bank1.example.com
acl serverIsBank ssl::server_name .bank2.example.net

#καλύτερα όταν μαζευτούν τα site που δεν θα γίνουν bump
#να μπουν σε ένα αρχείο και με include να περιληφθούν
#εδώ στην acl. Επίσης καλύτερα να μπαίνει με dstdomain
#Δίνεται παράδειγμα στο τέλος του αρχείου

acl step1 at_step SslBump1
ssl_bump peek step1
ssl_bump bump !serverIsBank
ssl_bump splice all


#ssl::server_name_regex -i "/etc/squid/doms.nobump"

#/etc/squid/doms.nobump:
#update\.microsoft\.com$
#update\.microsoft\.com\.akadns\.net$
#v10\.vortex\-win\.data\.microsoft.com$
#settings\-win\.data\.microsoft\.com$
```

7.2.5 Mode4.conf (TLS Secure Web Proxy)

```
https_port 443 cert=/usr/local/etc/squid/ssl_cert/cache04.crt
key=/usr/local/etc/squid/ssl_cert/cache04.key
```

7.2.6 General.conf (Γενικές ρυθμίσεις)

```
icp_port            0
dns_v4_first        on
cache_mem           1920 MB
cache_swap_low      80
cache_swap_high     85
maximum_object_size 4096 KB
ipcache_size        94208
ipcache_low         90
ipcache_high        95
#cache_dir           ufs /usr/local/etc/squid/cache 10000 16 256
#coredump_dir       /usr/local/etc/squid/cache
quick_abort_min     0 KB
quick_abort_max     0 KB
positive_dns_ttl    48 hours

# kid1| Logfile: daemon:/var/squid/logs/access.log: queue is too large; some log messages
have been lost.
access_log          daemon:/var/log/squid/access.log squid
#access_log          stdio:/var/squid/logs/access.log
cache_log           /var/log/squid/cache.log
cache_store_log     none
logfile_rotate 0
client_netmask      255.255.255.255
```

7.2.7 Webfilter.conf (ρυθμίσεις για web filter)

```
#X#url_rewrite_program     /usr/local/bin/squidGuard
#X#url_rewrite_children    48 startup=24 idle=12 concurrency=0
#X#url_rewrite_bypass      on

#if ${process_number} = 3
#  # no helpers for coordinator
#else
  url_rewrite_program     /usr/local/ufdbguard/bin/ufdbgclient -C -l /var/log/squid/ufdbguard
  url_rewrite_children  20 startup=6 idle=6 concurrency=2
  url_rewrite_bypass    on
#endif

refresh_pattern     .               0  20%  1440
refresh_pattern     \.(gif|jpg|png)$  0  20%  4320
```

7.2.8 Acls.conf (Access List)

```
#Access Lists
acl manager      proto    cache_object
#acl localhost   src      127.0.0.1/32 ::1
acl noc_lan      src      147.102.220.0/24 2001:648:2000:dc::/64
acl noc_vpn      src      147.102.224.64/27
acl management            src      147.102.220.1 2001:648:2000:dc::1
acl kerberos     src      147.102.222.222
acl cache_srv    src      194.63.239.224/27
acl mon          src      194.63.239.21/32
acl loadbalance src      194.63.239.181
acl load1        src      194.63.239.226
acl SSL_ports    port     443 563
acl Safe_ports   port     80 81 82 21 443 563 70 210 8991
acl Safe_ports   port     280        # http-mgmt
acl Safe_ports   port     488        # gss-http
acl Safe_ports   port     591        # filemaker
acl Safe_ports   port     777        # multiling http
acl Safe_ports   port     6080 6081    # EDUnet helpdesk and webmail
acl Safe_ports   port     7777        # http://aslb5.cc.cec.eu.int:7777/
acl Safe_ports   port     8383        # http://medical-physics.com:8383/
acl Safe_ports   port     9081        # http://portalsrv.eap.gr:9081/wps/portal/!ut/p/.scr/Login/
acl Safe_ports   port     9090        # http://www.cdseda.att.sch.gr:9090/
acl Safe_ports   port     8990        # http://193.92.187.46:8990/F/
acl Safe_ports   port     8000
acl Safe_ports   port     3000        # wikidot.com
acl badsites dstdomain 194.63.239.236              #to deny me
deny_info http://google.com all                 #Deny with redirect to google.com for lan
http_reply_access deny badsites all


acl CONNECT   method CONNECT
acl PURGE     method purge
```

7.2.9 Authoff.conf (χωρίς αυθεντικοποίηση)

```
http_access      allow    loadbalance
http_access      allow    load1
http_access      allow    localhost
http_access      allow    purge localhost
http_access      allow    purge management
http_access      allow    purge noc_lan
http_access      allow    purge noc_vpn
http_access      allow    manager localhost
http_access      allow    manager management
http_access      allow    manager noc_lan
http_access      allow    manager noc_vpn
http_access      allow    manager akis
http_access      allow    all
http_access      deny     manager
http_access      deny     !Safe_ports
http_access      deny     CONNECT !SSL_ports
http_access      deny     localhost
http_access      deny     cache_srv
```

```
http_access     deny    purge all

icp_access      deny    all
miss_access     allow   all
no_cache        deny    all
```

7.2.10 Authbasicldap.conf (basic authentication απευθείας με ldap)

```
auth_param basic children 400 startup=0 idle=1
auth_param basic realm Web-Proxy
auth_param basic credentialsttl 8 hours
auth_param basic program /usr/local/libexec/squid/basic_ldap_auth -h openldap1.att.sch.gr -b
"dc=sch,dc=gr" -f "uid=%s" -D uid=qmaileye,dc=sch,dc=gr -w qmaileye

acl ldap-auth proxy_auth REQUIRED

http_access     allow   loadbalance
http_access     allow   load1
http_access     allow   ldap-auth
http_access     allow   localhost
http_access     allow   purge localhost
http_access     allow   purge management
http_access     allow   purge noc_lan
http_access     allow   purge noc_vpn
http_access     allow   manager localhost
http_access     allow   manager management
http_access     allow   manager noc_lan
http_access     allow   manager noc_vpn
http_access     allow   manager akis
http_access     deny    all
http_access     deny    manager
http_access     deny    !Safe_ports
http_access     deny    CONNECT !SSL_ports
http_access     deny    localhost
http_access     deny    cache_srv
http_access     deny    purge all

icp_access      deny    all
miss_access     allow   all
no_cache        deny    all
```

7.2.11 Snmp.conf

```
if ${process_number} = 1
  snmp_port       163
endif
if ${process_number} = 2
  snmp_port       164
endif

snmp_incoming_address           0.0.0.0
snmp_access     allow     localhost
snmp_access     allow     kerberos
snmp_access     allow     management
snmp_access     allow     noc_lan
snmp_access     allow     noc_vpn
snmp_access     allow     cache_srv
snmp_access     allow     mon
snmp_access     allow     akis
snmp_access     deny      all
```

7.3 HAPROXY INSTALL & CONFIG

```
#!/bin/bash
#dokimastike se ubuntu server 16.04 TLS
ips=$(ip -o addr show up primary scope global |
    while read -r num dev fam addr rest; do echo ${addr%/*}; done)
hostnamectl set-hostname $1
sudo apt-get update
sudo apt-get install haproxy
sudo apt-get install software-properties-common
sudo add-apt-repository ppa:certbot/certbot
sudo apt-get install python-certbot-apache
sudo certbot certonly --manual --preffered-challenges dns
sudo mkdir /etc/haproxy/certs/
sudo cp /etc/letsencrypt/live/$1/*.* /etc/haproxy/
sudo cat /etc/haproxy/fullchain.pem /etc/haproxy/privkey.pem > /etc/haproxy/certs/mykey.pem
sudo rm -rf /etc/haproxy/*.pem
sudo chown haproxy:haproxy /etc/haproxy/certs/
sudo chmod 500 /etc/haproxy/certs/

sudo echo "
frontend www-http

#to fronted auto einai gia mi kryptografimeni kinhsh
#xrisimopoieitai gia ta updates giati polloi
#update clients kai ton windows den ypostirizoun
#kryptografimeni kinisi metaxy client kai proxy
#to fronted auto einai gia tin kryptografimeni kinhsh
"
sudo echo "                bind      "
sudo echo $ips+":80

          reqadd  X-Forwarded-Proto:\ http
          default_backend cacheupdates
```

```
frontend www-https

        bind      "
sudo echo $ips+":443 ssl crt /etc/haproxy/certs/mykey.pem
        reqadd  X-Forwarded-Proto:\ https
        #to cert gia kryptografisi
        #kai na einai apo episimi arxi  (no self-signed)
        #prepei na symfwnei me to onoma tis ypiresias
        default_backend             cacheboxes
backend cacheboxes

#to kanoniko backend me ldap authentication
        cookie SERVERID insert indirect nocache
        #eisago cookie gia sticky sessions
        balance uri whole
        #to geniko loadbalance einai me to hash tou url
        hash-type consistent
        balance leastconn if {ssl_fc}
        #an to protokollo einai https kano load balance me tis ligoteres syndeseis
        server cache03 194.63.239.233:3128     check   cookie cache03
        server cache04 194.63.239.234:3128     check   cookie cache04
        server cache05 194.63.239.235:3128     check   cookie cache05
        server cache06 194.63.239.236:3128     check   cookie cache06
        server cache07 194.63.239.237:3128     check   cookie cache07
        server cache08 194.63.239.238:3128     check   cookie cache08
        server cache09 194.63.239.239:3128     check   cookie cache09
        server cache10 194.63.239.240:3128     check   cookie cache10
        server cache11 194.63.239.241:3128     check   cookie cache11
        server cache12 194.63.239.242:3128     check   cookie cache12
        server cache13 194.63.239.243:3128     check   cookie cache13
        server cache14 194.63.239.244:3128     check   cookie cache14
        server cache15 194.63.239.245:3128     check   cookie cache15
        server cache16 194.63.239.246:3128     check   cookie cache16
backend cacheupdates
#to backend poy einai gia updates
#den xreiazetai authentication alla epitrepei
#mono kinisi prow tous update servers
#pou exoun rithmistei sto squid
        balance uri

        hash-type consistent
        server cache01 194.63.239.231:3128     check
        server cache02 194.63.239.232:3128     check" >> /etc/haproxy/haproxy.cfg
sudo service haproxy restart
```

7.4 KERBEROS SETUP AND CONFIG

7.4.1 Name Server config (Ubuntu Server 12.06TLS, BIND9 name server)

DNS server IP = 192.168.1.26 FQDN = nameserver.sch.gr

KDC server IP = 192.168.1 FQDN = kerberos.sch.gr

Squid server IP = 192.168.1. FQDN = squid.sch.gr

Named.conf

```
include "/etc/bind/named.conf.options";
include "/etc/bind/named.conf.local";
include "/etc/bind/named.conf.default-zones";
```

Named.conf.local

```
zone "sch.gr" {
        type master;
        file "/etc/bind/for.sch.gr";
        allow-transfer { 192.168.1.26; };
        also-notify { 192.168.1.26; };
 };
zone "1.168.192.in-addr.arpa" {
        type master;
        file "/etc/bind/rev.sch.gr";
        allow-transfer { 192.168.1.26; };
        also-notify { 192.168.1.26; };
 };
```

for.sch.gr

```
$TTL 86400
@   IN  SOA     nameserver.sch.gr. sch.gr. (
        201107100  ;Serial
        3600       ;Refresh
        1800       ;Retry
        604800     ;Expire
        86400      ;Minimum TTL
)
@       IN  NS          nameserver.sch.gr.
@       IN  A           192.168.1.26
@       IN  A           192.168.1.13
@       IN  A           192.168.1.22
nameserver  IN      A           192.168.1.26
kerberos    IN      A           192.168.1.13
squid       IN      A           192.168.1.22
_kerberos._udp.sch.gr.        IN  SRV  10  0   88  kerberos.sch.gr.
_kerberos-master._udp.sch.gr. IN  SRV   0  0   88  kerberos.sch.gr.
_kerberos-adm._tcp.sch.gr.    IN  SRV   0  0  749  kerberos.sch.gr.
_kpasswd._udp.sch.gr.         IN  SRV   0  0  464  kerberos.sch.gr.
```

rev.sch.gr

```
$TTL 86400
@   IN  SOA     nameserver.sch.gr. sch.gr. (
      2011071002  ;Serial
      3600        ;Refresh
      1800        ;Retry
      604800      ;Expire
      86400       ;Minimum TTL
)
@       IN  NS          nameserver.sch.gr.
@       IN  PTR         sch.gr.
nameserver   IN  A      192.168.1.26
kerberos     IN  A      192.168.1.13
squid        IN  A      192.168.1.22
26      IN  PTR         nameserver.sch.gr.
13      IN  PTR         kerberos.sch.gr.
22      IN  PTR         squid.sch.gr
```

7.4.2 KDC Server config (FreeBSD 12 , Heimdal Kerberos)

```
echo kdc_enable="YES"       >>  /etc/rc.conf
echo kadmind_enable="YES"   >>  /etc/rc.conf
echo "[libdefaults]
   default_realm = SCH.GR
[realms]
   SCH.GR = {
      kdc = kerberos.sch.gr
      admin_server = kerberos.sch.gr
   }
[domain_realm]
   .sch.gr = SCH.GR"                >>        /etc/krb5.conf
echo "domain sch.gr
nameserver 192.168.1.26"           >>        /etc/resolv.conf
 kstash
# Master key: xxxxxxxxxx
# Verifying password - Master key: xxxxxxxxxx
# kadmin -l
# kadmin> init SCH.GR
#* Accept default options
kadmin -l
kadmin> add testuser
#Max ticket life [unlimited]:
#Max renewable life [unlimited]:
#Attributes []:
#Password: xxxxxxxx
#Verifying password - Password: xxxxxxxx
kadmin> add akis/admin
#Max ticket life [unlimited]:
#Max renewable life [unlimited]:
#Attributes []:
#Password: xxxxxxxx
#Verifying password - Password: xxxxxxxx
echo "akis/admin@SCH.GR all">>        /var/heimdal/kadmind.acl
service kdc start
```

```
service kadmind start
kinit testuser
klist #prepei na vgalei to ticket toy testuser
kdestroy #katastrefoyme to ticket tis dokimis
```

7.4.3 SQUID SERVER KERBEROS CONFIG

```
echo "[libdefaults]
   default_realm = SCH.GR
          default_keytab_name = /etc/PROXY.keytab
[realms]
   SCH.GR = {
      kdc = kerberos.sch.gr
      admin_server = kerberos.sch.gr
   }
[domain_realm]
   .sch.gr = SCH.GR"              >>       /etc/krb5.conf
echo "domain sch.gr
nameserver 192.168.1.26"         >>       /etc/resolv.conf
set KRB5_KTNAME=/etc/PROXY.keytab
chgrp squid /etc/PROXY.keytab
chmod g+r /etc/PROXY.keytab
kinit akis/admin
ktutil get -p akis/admin host/squid.sch.gr
#Max ticket life [unlimited]:
#Max renewable life [unlimited]:
#Principal expiration time [never]:
#Password expiration time [never]:
#Attributes []:
ktutil get -p akis/admin HTTP/squid.sch.gr
#Max ticket life [unlimited]:
#Max renewable life [unlimited]:
#Principal expiration time [never]:
#Password expiration time [never]:
#Attributes []:
#dokimi an mporw na kanw authenticate gia thn HTTP yphresia
/usr/local/libexec/squid/negotiate_kerberos_auth_test squid.sch.gr \
| awk '{sub(/Token:/,"YR"); print $0}END{print "QQ"}' \
| /usr/local/libexec/squid/negotiate_kerberos_auth -r -s HTTP/squid.sch.gr
# to apotelesma prepei na einai san to parakatw
#AF oRQwEqADCgEAoQsGCSqGSIb3EgECAg== host/squid.sch.gr
#BH quit command
#an den vgalei AF kapoio lathos exei ginei
```

Στο squid.conf αλλάζει το authentication :

```
auth_param negotiate program /usr/local/libexec/squid/negotiate_kerberos_auth -d
 -i -s HTTP/squid.sch.gr
auth_param negotiate children 10 startup=1
auth_param negotiate keep_alive on
```

7.4.4 CLIENT configuration (Ubuntu 16.04 Desktop)

```
sudo apt install krb5-user
echo "[libdefaults]
   default_realm = SCH.GR
          default_keytab_name = /etc/PROXY.keytab
[realms]
   SCH.GR = {
      kdc = kerberos.sch.gr
      admin_server = kerberos.sch.gr
   }
[domain_realm]
   .sch.gr = SCH.GR"              >>       /etc/krb5.conf
echo "domain sch.gr
nameserver 192.168.1.26"         >>       /etc/resolv.conf
kinit testuser
klist
```

7.5 PAC ΓΙΑ WPAD (PROXY AUTO CONFIGURE)

```
FunctionFindProxyForURL(url, host)
{
//an afto poy zitaw einai private ip
//tote min xrhsimopoihseis proxy server
if (isInNet(host, "0.0.0.0", "255.0.0.0")||
isInNet(host, "10.0.0.0", "255.0.0.0") ||
isInNet(host, "127.0.0.0", "255.0.0.0") ||
isInNet(host, "169.254.0.0", "255.255.0.0") ||
isInNet(host, "172.16.0.0", "255.240.0.0") ||
isInNet(host, "192.168.0.0", "255.255.255.0")
)
return "DIRECT";
//gia ta domain tou MS update tha xrhsimopoihseis
//tous squid server poy einai gia MS update xwris
//authentication
// proteinetai na ftiaxtei mia kainouria yphresia ston
//load balancer px windowsupdate.sch.gr kai na
// mpei sto arxeio ayth anti gia tous 2 server
if   ((shExpMatch(host, "*.microsoft.com"))||(shExpMatch(host,
"*.windowsupdate.com"))||(shExpMatch(host, "*.windows.com")))
     return "PROXY proxyupdates.sch.gr:3128; cachexx.att.sch.gr:3128; " ;

// an to protokollo einai http tote exrhsimopoihse
// ton secure WEB PROXY proxynew.sch.gr
// pros to paron den kanoume proxy sto https
// gia na kanoume kai sto https aplws
// kanoume comment to epomemo if block
// epeidh h yphresia toy load balancer
// mporei na exei pesei tote vazw kai egw
```

```
// toys ypoloipous
//proteinetai na dhmiourgithei nea yphresia
// backupproxy.sch.gr h opoia tha kanei roundrobin
//dns stous authenticated proxies wste na dianemetai
//oso ginetai to fortio se periptwsh pou pesei o
// load balancer.
if (shExpMatch(url, "http:*")) {
  return "HTTPS proxynew.sch.gr:3128 ; backupproxy.sch.gr:3127";
}


// an exw ftasei mexri edw kai den exw kanei return
// shmainei oti den eimai se kapoia apo tis alles periptwseis
// opote den xerw ti na xrisimopoihsw kai paw directories
return "DIRECT";
}
```

7.6 GRAFANA JSON ΓΙΑ REPORTS

```
{
  "__inputs": [
    {
      "name": "DS_LOCAL",
      "label": "Local",
      "description": "",
      "type": "datasource",
      "pluginId": "prometheus",
      "pluginName": "Prometheus"
    }
  ],
  "__requires": [
    {
      "type": "grafana",
```

```
      "id": "grafana",
      "name": "Grafana",
      "version": "5.0.0"
    },
    {
      "type": "panel",
      "id": "graph",
      "name": "Graph",
      "version": "5.0.0"
    },
    {
      "type": "datasource",
      "id": "prometheus",
      "name": "Prometheus",
      "version": "5.0.0"
    }
  ],
  "annotations": {
    "list": [
      {
        "builtIn": 1,
        "datasource": "-- Grafana --",
        "enable": true,
        "hide": true,
        "iconColor": "rgba(0, 211, 255, 1)",
        "name": "Annotations & Alerts",
        "type": "dashboard"
      }
    ]
  },
  "editable": true,
  "gnetId": null,
  "graphTooltip": 0,
  "id": null,
  "iteration": 1526388557805,
```

```
"links": [],
"panels": [
  {
    "aliasColors": {},
    "bars": false,
    "dashLength": 10,
    "dashes": false,
    "datasource": "${DS_LOCAL}",
    "fill": 1,
    "gridPos": {
      "h": 9,
      "w": 12,
      "x": 0,
      "y": 0
    },
    "id": 12,
    "legend": {
      "avg": false,
      "current": false,
      "max": false,
      "min": false,
      "show": true,
      "total": false,
      "values": false
    },
    "lines": true,
    "linewidth": 1,
    "links": [],
    "nullPointMode": "null",
    "percentage": false,
    "pointradius": 5,
    "points": false,
    "renderer": "flot",
    "seriesOverrides": [],
    "spaceLength": 10,
```

```
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr": "100 - (
(irate(node_cpu_seconds_total{instance='$node:$port',job=\"node\",mode=\"idle\"}
[15s])) * 100)",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "",
          "refId": "A"
        },
        {
          "expr": "",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "",
          "refId": "B"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "PER CPU % /sec",
      "tooltip": {
        "shared": true,
        "sort": 0,
        "value_type": "individual"
      },
      "type": "graph",
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
```

```
      "values": []
    },
    "yaxes": [
      {
        "format": "short",
        "label": null,
        "logBase": 1,
        "max": null,
        "min": null,
        "show": true
      },
      {
        "format": "short",
        "label": null,
        "logBase": 1,
        "max": null,
        "min": null,
        "show": true
      }
    ],
    "yaxis": {
      "align": false,
      "alignLevel": null
    }
  },
  {
    "aliasColors": {},
    "bars": false,
    "dashLength": 10,
    "dashes": false,
    "datasource": "${DS_LOCAL}",
    "fill": 1,
    "gridPos": {
      "h": 9,
      "w": 12,
```

```
        "x": 12,
        "y": 0
      },
      "id": 6,
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
      },
      "lines": true,
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr":
"rate(node_network_receive_drop{instance='$node:$port'}[20s])+rate(node_network_
transmit_drop{instance='$node:$port'}[20s])",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "",
          "refId": "A"
        }
```

```
    ],
    "thresholds": [],
    "timeFrom": null,
    "timeShift": null,
    "title": "Dropped Packets/ Sec",
    "tooltip": {
      "shared": true,
      "sort": 0,
      "value_type": "individual"
    },
    "type": "graph",
    "xaxis": {
      "buckets": null,
      "mode": "time",
      "name": null,
      "show": true,
      "values": []
    },
    "yaxes": [
      {
        "format": "short",
        "label": null,
        "logBase": 1,
        "max": null,
        "min": null,
        "show": true
      },
      {
        "format": "short",
        "label": null,
        "logBase": 1,
        "max": null,
        "min": null,
        "show": true
      }
```

```
    ],
    "yaxis": {
      "align": false,
      "alignLevel": null
    }
  },
  {
    "aliasColors": {},
    "bars": false,
    "dashLength": 10,
    "dashes": false,
    "datasource": "${DS_LOCAL}",
    "description": "Average per 5s for all 8 cores",
    "fill": 1,
    "gridPos": {
      "h": 9,
      "w": 12,
      "x": 0,
      "y": 9
    },
    "id": 2,
    "legend": {
      "avg": false,
      "current": false,
      "max": false,
      "min": false,
      "show": true,
      "total": false,
      "values": false
    },
    "lines": true,
    "linewidth": 1,
    "links": [],
    "nullPointMode": "null",
    "percentage": false,
```

```
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr": "100 - (avg by (instance)
(irate(node_cpu_seconds_total{instance='$node:$port',job=\"node\",mode=\"idle\"}
[5s])) * 100)",
          "format": "time_series",
          "interval": "1s",
          "intervalFactor": 1,
          "legendFormat": "CPU % ",
          "refId": "A"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "CPU %",
      "tooltip": {
        "shared": true,
        "sort": 0,
        "value_type": "individual"
      },
      "type": "graph",
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
        "values": []
```

```
    },
    "yaxes": [
      {
        "format": "short",
        "label": null,
        "logBase": 1,
        "max": null,
        "min": null,
        "show": true
      },
      {
        "format": "short",
        "label": null,
        "logBase": 1,
        "max": null,
        "min": null,
        "show": true
      }
    ],
    "yaxis": {
      "align": false,
      "alignLevel": null
    }
  },
  {
    "aliasColors": {},
    "bars": false,
    "dashLength": 10,
    "dashes": false,
    "datasource": "${DS_LOCAL}",
    "fill": 1,
    "gridPos": {
      "h": 9,
      "w": 12,
      "x": 12,
```

```
        "y": 9
      },
      "id": 8,
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
      },
      "lines": true,
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr":
"rate(node_network_receive_errs{instance='$node:$port'}[20s])+rate(node_network_
transmit_errs{instance='$node:$port'}[20s])",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "",
          "refId": "A"
        }
      ],
```

```
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "Transmit + Receive Errors / Sec",
      "tooltip": {
        "shared": true,
        "sort": 0,
        "value_type": "individual"
      },
      "type": "graph",
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
        "values": []
      },
      "yaxes": [
        {
          "format": "short",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        },
        {
          "format": "short",
          "label": null,
          "logBase": 1,
          "max": null,
          "min": null,
          "show": true
        }
      ],
```

```
    "yaxis": {
      "align": false,
      "alignLevel": null
    }
  },
  {
    "aliasColors": {},
    "bars": false,
    "dashLength": 10,
    "dashes": false,
    "datasource": "${DS_LOCAL}",
    "fill": 1,
    "gridPos": {
      "h": 9,
      "w": 12,
      "x": 0,
      "y": 18
    },
    "id": 4,
    "legend": {
      "avg": false,
      "current": false,
      "max": false,
      "min": false,
      "show": true,
      "total": false,
      "values": false
    },
    "lines": true,
    "linewidth": 1,
    "links": [],
    "nullPointMode": "null",
    "percentage": false,
    "pointradius": 5,
    "points": false,
```

```
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr":
"rate(node_network_receive_packets{instance='$node:$port'}[10s])+rate(node_netwo
rk_transmit_packets{instance='$node:$port'}[10s])",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "",
          "refId": "A"
        }
      ],
      "thresholds": [],
      "timeFrom": null,
      "timeShift": null,
      "title": "Send & Receive Packets / Sec",
      "tooltip": {
        "shared": true,
        "sort": 0,
        "value_type": "individual"
      },
      "type": "graph",
      "xaxis": {
        "buckets": null,
        "mode": "time",
        "name": null,
        "show": true,
        "values": []
      },
      "yaxes": [
        {
```

```
        "format": "short",
        "label": null,
        "logBase": 1,
        "max": null,
        "min": null,
        "show": true
      },
      {
        "format": "short",
        "label": null,
        "logBase": 1,
        "max": null,
        "min": null,
        "show": true
      }
    ],
    "yaxis": {
      "align": false,
      "alignLevel": null
    }
  },
  {
    "aliasColors": {},
    "bars": false,
    "dashLength": 10,
    "dashes": false,
    "datasource": "${DS_LOCAL}",
    "fill": 1,
    "gridPos": {
      "h": 9,
      "w": 12,
      "x": 12,
      "y": 18
    },
    "id": 10,
```

```
      "legend": {
        "avg": false,
        "current": false,
        "max": false,
        "min": false,
        "show": true,
        "total": false,
        "values": false
      },
      "lines": true,
      "linewidth": 1,
      "links": [],
      "nullPointMode": "null",
      "percentage": false,
      "pointradius": 5,
      "points": false,
      "renderer": "flot",
      "seriesOverrides": [],
      "spaceLength": 10,
      "stack": false,
      "steppedLine": false,
      "targets": [
        {
          "expr": "rate(node_network_receive_bytes{instance='$node:$port'}[20s])
",
          "format": "time_series",
          "intervalFactor": 1,
          "legendFormat": "",
          "refId": "A"
        },
        {
          "expr":
"rate(node_network_transmit_bytes{instance='$node:$port'}[20s])",
          "format": "time_series",
          "intervalFactor": 1,
```

```
      "legendFormat": "",
      "refId": "B"
    }
  ],
  "thresholds": [],
  "timeFrom": null,
  "timeShift": null,
  "title": "Mbytes /sec",
  "tooltip": {
    "shared": true,
    "sort": 0,
    "value_type": "individual"
  },
  "type": "graph",
  "xaxis": {
    "buckets": null,
    "mode": "time",
    "name": null,
    "show": true,
    "values": []
  },
  "yaxes": [
    {
      "format": "decbytes",
      "label": "",
      "logBase": 1,
      "max": null,
      "min": null,
      "show": true
    },
    {
      "format": "short",
      "label": null,
      "logBase": 1,
      "max": null,
```

```
          "min": null,
          "show": true
        }
      ],
      "yaxis": {
        "align": false,
        "alignLevel": null
      }
    }
  ],
  "refresh": "10s",
  "schemaVersion": 16,
  "style": "dark",
  "tags": [],
  "templating": {
    "list": [
      {
        "allValue": null,
        "current": {},
        "datasource": "${DS_LOCAL}",
        "hide": 0,
        "includeAll": false,
        "label": "Host:",
        "multi": false,
        "name": "node",
        "options": [],
        "query": "label_values(node_exec_boot_timestamp_seconds{job=\"node\"},
instance)",
        "refresh": 1,
        "regex": "/([^:]+):.*/",
        "sort": 0,
        "tagValuesQuery": "",
        "tags": [],
        "tagsQuery": "",
        "type": "query",
```

```
      "useTags": false
    },
    {
      "allValue": null,
      "current": {},
      "datasource": "${DS_LOCAL}",
      "hide": 0,
      "includeAll": false,
      "label": "port",
      "multi": false,
      "name": "port",
      "options": [],
      "query": "label_values(node_exec_boot_timestamp_seconds, instance)",
      "refresh": 1,
      "regex": "/[^:]+:(.*)/",
      "sort": 0,
      "tagValuesQuery": "",
      "tags": [],
      "tagsQuery": "",
      "type": "query",
      "useTags": false
    }
  ]
},
"time": {
  "from": "now-30m",
  "to": "now"
},
"timepicker": {
  "refresh_intervals": [
    "5s",
    "10s",
    "30s",
    "1m",
    "5m",
```

```
      "15m",
      "30m",
      "1h",
      "2h",
      "1d"
    ],
    "time_options": [
      "5m",
      "15m",
      "1h",
      "6h",
      "12h",
      "24h",
      "2d",
      "7d",
      "30d"
    ]
  },
  "timezone": "",
  "title": "Cache06",
  "uid": "LMA15GGik",
  "version": 9
}
```

7.7 PROMETHEUS CONFIGURATION FILE

```
# my global config
global:
  scrape_interval:     1s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
  evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every 1 minute.
  # scrape_timeout is set to the global default (10s).

# Alertmanager configuration
alerting:
  alertmanagers:
  - static_configs:
    - targets:
      # - alertmanager:9093
```

```
# Load rules once and periodically evaluate them according to the global 'evaluation_interval'.
rule_files:
  # - "first_rules.yml"
  # - "second_rules.yml"

# A scrape configuration containing exactly one endpoint to scrape:
# Here it's Prometheus itself.
scrape_configs:
  # The job name is added as a label `job=<job_name>` to any timeseries scraped from this
config.
  - job_name: 'prometheus'

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ['localhost:9090']
  - job_name: 'node'

    # metrics_path defaults to '/metrics'
    # scheme defaults to 'http'.

    static_configs:
      - targets: ['cache06.att.sch.gr:9100','cache04.att.sch.gr:9100']
```