Ατομική Διπλωματική Εργασία

EXPLORING THE TOR ECOSYSTEM

Αντριάνα Χρυσοστόμου

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Exploring the TOR ecosystem

Αντριάνα Χρυσοστόμου

Επιβλέπων Καθηγητής
Δρ. Ηλίας Αθανασόπουλος

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2022

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον Δρ. Ηλία Αθανασόπουλο ως Επιβλέπων Καθηγητή για την συνεχή βοήθεια και καθοδήγηση κατά τη διάρκεια της εκπόνησης της παρούσας ατομικής διπλωματικής εργασίας. Θα ήθελα επίσης να τον ευχαριστήσω και για την διαρκή βοήθεια και στήριξη ως Ακαδημαϊκό Σύμβουλο κατά τη διάρκεια φοίτησης μου στο Πανεπιστήμιο Κύπρου.

Ακόμη, θα ήθελα να ευχαριστήσω την οικογένεια μου και ειδικά την μητέρα μου για την ατέρμονη στήριξη και συμπαράστασή τους σε όλες τις δύσκολες στιγμές κατά την διάρκεια της φοίτησής μου.

Τέλος, θα ήθελα να ευχαριστήσω τους φίλους μου που ήταν πάντα στο πλευρό μου και με στήριζαν. Ιδιαίτερες ευχαριστίες στον φίλο μου Ανδρόνικο Χαραλάμπους που ήταν δίπλα μου σε αυτό το ταξίδι από τις πρώτες μέρες στο πανεπιστήμιο, για τις ατελείωτες ώρες μελέτης στα εργαστήρια του πανεπιστημίου.

Περίληψη

Έχει παρατηρηθεί τα τελευταία χρόνια ότι το οικοσύστημα του TOR γίνεται ολοένα και πιο διαδεδομένο και αποκτά νέους χρήστες συνεχώς. Ο κύριος σκοπός της παρούσας ατομικής διπλωματικής εργασίας, είναι να εξερευνηθούν οι λειτουργίες και οι δυνατότητες του TOR ως δίκτυο.

Για την εκπόνηση της παρούσας έρευνας, χρησιμοποιήθηκαν κάποια scripts, τα οποία γράφτηκαν αποκλειστικά για αυτό τον σκοπό. Συγκεκριμένα, τα scripts γράφτηκαν σε Python και έγινε χρήση της βιβλιοθήκης Stem. Έχει επιλεγεί η Python ως η κύρια γλώσσα για τα scripts λόγω του ότι η Python είναι μια αρκετά ισχυρή γλώσσα και είναι ιδανική για γρήγορη ανάπτυξη κώδικα. Η βιβλιοθήκη Stem, είναι μια controller library για το TOR, και έχει παίξει ένα πολύ σημαντικό ρόλο στην υλοποίηση των πειραμάτων που χρειάστηκαν για την παρούσα έρευνα.

Τα προαναφερθέντα scripts, είναι υπεύθυνα για να παίρνουν συγκεκριμένες μετρήσεις σχετικά με την δημιουργία κάποιων κυκλωμάτων (circuits) που δημιουργούνται τυχαία.

Οι προδιαγραφές (π.χ. number of nodes, exit fingerprint) των circuits καθορίζονται ως παράμετροι πριν την εκτέλεση των πειραμάτων. Επίσης, μια από τις προδιαγραφές των circuits είναι το URL στο οποίο θα γίνει request, το οποίο είναι hardcoded – δηλαδή καθορίζεται μέσα από τον κώδικα. Τα ενδιάμεσα nodes του κάθε circuit επιλέγονται τυχαία με βάση του ποια nodes είναι διαθέσιμα την δεδομένη στιγμή.

Τα αποτελέσματα που επιστρέφουν τα scripts διαφέρουν, αναλόγως του ποιο script εκτελείται. Μια πληροφορία που επιστρέφεται είναι η δυνατότητα δημιουργίας ενός circuit με βάση τις παραμέτρους που δίνονται σαν είσοδος. Ακόμα, κάποια scripts επιστρέφουν αποτελέσματα σχετικά με τα ενδιάμεσα nodes, τον χρόνο που χρειάστηκε ένα circuit για να δημιουργηθεί ή ακόμα και τον χρόνο που χρειάστηκε ένα request να εξυπηρετηθεί. Με αυτή την μελέτη, παίρνουμε κάποια βασικά στοιχεία σχετικά με το TOR ως δίκτυο και διάφορα εργαλεία που μπορούν να χρησιμοποιηθούν με σκοπό να καταλάβουμε πως λειτουργεί το TOR.

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1 8

ΕΙΣΑΓΩΓΗ 8

1.1. Τι είναι το TOR *8*

1.2. Onion Routing Protocol *8*

1.3. Σκοπός ΑΔΕ *10*

1.4. Δομή ΑΔΕ *11*

ΚΕΦΑΛΑΙΟ 2 12

ΑΡΧΙΤΕΚΤΟΝΙΚΗ 12

2.1. Περιγραφή πειραμάτων *13*

2.1.1 Πείραμα 1 13

2.1.2 Πείραμα 2 13

2.1.3 Πείραμα 3 13

2.1.4 Πείραμα 4 13

2.1.5 Πείραμα 5 14

2.1.6 Πείραμα 6 14

2.1.7 Βοηθητικό Script 14

2.2. Testbench *15*

2.2.1 Πείραμα 1 15

2.2.2 Πείραμα 2 15

2.2.3 Πείραμα 3 15

2.2.4 Πείραμα 4 15

2.2.5 Πείραμα 5 15

2.2.6 Πείραμα 6 16

ΚΕΦΑΛΑΙΟ 3 16

ΥΛΟΠΟΙΗΣΗ 17

3.1. Python *17*

3.2. Βιβλιοθήκη Stem *18*

3.3. Υλοποίηση Πειραμάτων ... *18*
3.3.1. Πείραμα 1 ... 18
3.3.2. Πείραμα 2 ... 18
3.3.3. Πείραμα 3 ... 19
3.3.4. Πείραμα 4 ... 19
3.3.5. Πείραμα 5 ... 20
3.3.6. Πείραμα 6 ... 20
3.3.7. Βοηθητικό Script ... 21
ΚΕΦΑΛΑΙΟ 4 ... **22**
ΑΞΙΟΛΟΓΗΣΗ ΠΕΙΡΑΜΑΤΩΝ ... 22
4.1 Μέγεθος circuit ... *22*
4.2 Χρόνος δημιουργίας circuit ... *23*
4.3 Χρόνος εξυπηρέτησης request ... *24*
ΚΕΦΑΛΑΙΟ 5 ... **26**
ΣΥΜΠΕΡΑΣΜΑΤΑ ... 26
5.1 Μελλοντική δουλειά ... *26*
5.2 Τελικά σχόλια ... *26*
ΠΑΡΑΡΤΗΜΑ Α ... **28**
ΠΑΡΑΡΤΗΜΑ Β ... **29**
ΠΑΡΑΡΤΗΜΑ Δ ... **31**
ΠΑΡΑΡΤΗΜΑ Ε ... **32**
ΠΑΡΑΡΤΗΜΑ ΣΤ ... **34**

Κεφάλαιο 1

Εισαγωγή

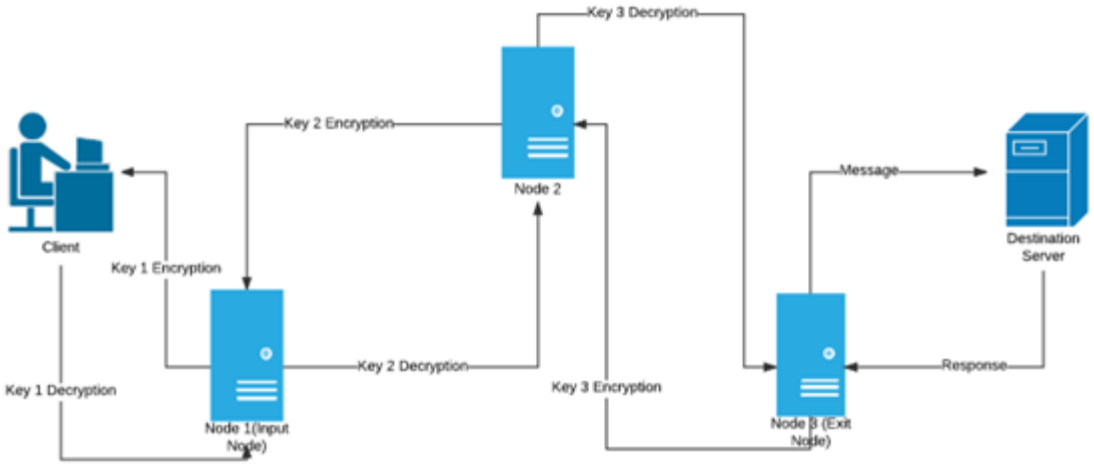
1.1 Τι είναι το TOR 8
1.2 Onion Routing Protocol 9
1.3 Σκοπός ΑΔΕ 10
1.4 Δομή ΑΔΕ 11

1.1. Τι είναι το TOR

Το TOR είναι ένα ανώνυμο δίκτυο επικοινωνίας το οποίο είχε αρχίσει σαν ιδέα την δεκαετία του 90 με σκοπό την ανώνυμη επικοινωνία ακόμα και αν κάποιος παρακολουθούσε το δίκτυο [1]. Σήμερα χρησιμοποιείτε από ένα μεγάλο αριθμό χρηστών που θέλουν να διατηρήσουν την ανωνυμία τους. Το Tor αξιοποιεί μια υλοποίηση του πρωτοκόλλου δρομολόγησης Onion [2] το οποίο θα επεκτείνουμε στο επόμενο κεφάλαιο. Αποτελείται από ένα δίκτυο δρομολογητών οι οποίοι παρέχονται εθελοντικά από τους χρήστες. Με αυτό τον τρόπο μπορεί να αποφευχθεί οποιοδήποτε παραδοσιακός μηχανισμός παρακολούθησης και ανάλυσης δικτύου. Επίσης το Tor χρησιμοποιείται για την δημιουργία κρυφών υπηρεσιών. Οι διακομιστές αυτών των υπηρεσιών μπορούν να προσπελαστούν μόνο μέσω τους προγράμματος περιήγησης του Tor. Μια τέτοια υπηρεσία έχει ως αναγνωριστικό ένα σύνδεσμο onion το οποίο δεν σχετίζεται με κάποια δημόσια διεύθυνση IP. Το Tor μπορεί να ερμηνεύσει αυτούς τους συνδέσμους και μπορεί να προωθήσει πακέτα από/προς αυτήν την υπηρεσία.

1.2. Onion Routing Protocol

Η δρομολόγηση Onion είναι ο τρόπος με τον οποίο το Tor μπορεί να δρομολογήσει πακέτα σε οποιοδήποτε κόμβο και παράλληλα να υπάρχει ανωνυμία μεταξύ αυτής της επικοινωνίας.



Σχήμα 1.1 Onion Routing [3]

Στην πιο πάνω εικόνα βλέπουμε την διαδικασία που εκτελείτε έτσι ώστε να παρέχεται η ανωνυμία μέσα στο δίκτυο. Αρχικά ο client θα επιλέξει από μια λίστα κόμβων που παρέχονται από έναν διακομιστή καταλόγου με τους κόμβους του δικτύου. Οι κόμβοι που έχουν επιλεχθεί θα δημιουργήσουν ένα μονοπάτι "circuit" όπου τα πακέτα θα προωθηθούν κρυπτογραφημένα από αυτούς προς τον παραλήπτη. Για να υπάρξει η ανωνυμία μεταξύ αυτής της επικοινωνίας κανένας κόμβος μέσα στο δίκτυο δεν ξέρει πόσο μεγάλο είναι το μονοπάτι καθώς επίσης κανένας κόμβος δεν ξέρει ποιος ήταν ο αρχικός κόμβος που προωθήθηκαν τα πακέτα. Μόνο ο τελικός κόμβος γνωρίζει ότι είναι ο τελευταίος αφού είναι ο κόμβος που θα αποκρυπτογραφηθούν τα πακέτα και θα προωθηθούν στον τελικό παραλήπτη. Για να επιτευχθεί η πιο πάνω διαδικασία εκτελούνται τα πιο κάτω βήματα (υποθέτουμε ότι το μονοπάτι θα αποτελείται από 3 κόμβους):

1. Ο client ορίζει 3 συμμετρικά κλειδιά τα οποία θα χρησιμοποιηθούν για να γίνει 3ων επιπέδων κρυπτογράφηση του μηνύματος. Έστω message το μήνυμα που θα πρέπει να σταλθεί, τότε θα έχουμε **encrypt(encrypt(encrypt(message, key3), key2), key1)**.
2. Τώρα το μήνυμα θα σταλθεί στον 1ο κόμβο μέσα στο μονοπάτι. Οι μοναδικές πληροφορίες που έχει αυτό ο κόμβος είναι το κλειδί 1 και τον επόμενο κόμβο (κόμβος 2) που θα προωθήσει το μήνυμα. Ο κόμβος 1 αποκρυπτογραφεί το μήνυμα και ακολούθως το προωθεί στον επόμενο κόμβο.

3. Ο κόμβος 2 λαμβάνει το μήνυμα, το αποκρυπτογραφεί και το προωθεί στον επόμενο κόμβο που γνωρίζει (κόμβος 3).

4. Ο κόμβος 3 λαμβάνει το μήνυμα, το αποκρυπτογραφεί και μπορεί να το διαβάσει. Αυτή την στιγμή ο κόμβος γνωρίζει ότι είναι ένας κόμβος εξόδου αλλά δεν μπορεί να ξέρει από πού προήλθε το μήνυμα. Το προωθεί στον παραλήπτη και παίρνει απάντηση πίσω.

Η διαδικασία για να σταλθεί πίσω μια απάντηση από τον παραλήπτη είναι αντίστροφη. Δηλαδή ο κόμβος 3 κρυπτογραφεί με το κλειδί του την απάντηση και την προωθεί στον κόμβο 2 και ούτω καθεξής. Όταν ο client λάβει την κρυπτογραφημένη απάντηση με τα 3 επίπεδα κρυπτογράφησης τότε μπορεί να τα αποκρυπτογραφήσει αφού γνωρίζει τα κλειδιά.

Παρόμοια διαδικασία γίνεται αν το μονοπάτι φτιαχτεί με περισσότερους από 3 κόμβους. Δηλαδή ο κάθε κόμβος θα αποκρυπτογραφεί με το δικό του κλειδί και θα προωθεί το μήνυμα μέχρι να φτάσει στον κόμβο εξόδου όπου το μήνυμα θα προωθηθεί στον παραλήπτη.

1.3. Σκοπός ΑΔΕ

Ο σκοπός της παρούσας ατομικής διπλωματικής εργασίας είναι η εξερεύνηση του TOR ως δίκτυο. Συγκεκριμένα, γράφτηκαν κάποια scripts, τα οποία είναι υπεύθυνα για την συλλογή συγκεκριμένων πληροφοριών σχετικά με την δημιουργία circuits διαφόρων μεγεθών, τα οποία αποτελούνται από τυχαία nodes. Το κοινό χαρακτηριστικό των circuits είναι το Exit Node, το οποίο προκαθορίζεται από τον χρήστη. Ακόμα, μέσα από κάποια από τα πειράματα παρατηρείται ο χρόνος που χρειάζεται ένα request να εξυπηρετηθεί αναλόγως της τοποθεσίας του Exit Node, το οποίο καθορίζεται κάθε φορά που τρέχει το πείραμα. Με τα πειράματα που εκτελέσθηκαν, έχουν εξαχθεί κάποια αποτελέσματα τα οποία αφορούν το μέγεθος του circuit αλλά και τον χρόνο που χρειάζεται να εξυπηρετηθεί ένα request με βάση την τοποθεσία των nodes που απαρτίζουν το κάθε circuit.

1.4. Δομή ΑΔΕ

Η συγκεκριμένη ατομική διπλωματική εργασία αποτελείται από 6 κεφάλαια όπως φαίνεται πιο κάτω :

- Κεφάλαιο 1: Εισαγωγή για το τι είναι το TOR και σύντομη επεξήγηση της ΑΔΕ.
- Κεφάλαιο 2: Περιγραφή πειραμάτων που εκτελέσθηκαν και επεξήγηση του πως έτρεξαν τα πειράματα.
- Κεφάλαιο 3: Σύντομη περιγραφή των εργαλείων που χρησιμοποιήθηκαν για την εκτέλεση των πειραμάτων και επεξήγηση υλοποίησης.
- Κεφάλαιο 4: Πειραματικά αποτελέσματα και σχολιασμός.
- Κεφάλαιο 5: Εξαγωγή συμπερασμάτων με βάση τα αποτελέσματα του κεφαλαίου 4.

Κεφάλαιο 2

Αρχιτεκτονική

2.1 Περιγραφή πειραμάτων 13
2.1.1 Πείραμα 1 13
2.1.2 Πείραμα 2 13
2.1.3 Πείραμα 3 13
2.1.4 Πείραμα 4 13
2.1.5 Πείραμα 5 14
2.1.6 Πείραμα 6 14
2.1.7 Βοηθητικό Script 14
2.2 Testbench – πώς έτρεξαν τα πειράματα 15
2.2.1 Πείραμα 1 15
2.2.2 Πείραμα 2 15
2.2.3 Πείραμα 3 15
2.2.4 Πείραμα 4 15
2.2.5 Πείραμα 5 15
2.2.6 Πείραμα 6 16

Σε αυτό το κεφάλαιο περιγράφεται η αρχιτεκτονική των πειραμάτων που έχουν πραγματοποιηθεί με σκοπό την εκπόνηση της παρούσας ατομικής διπλωματικής εργασίας. Πιο αναλυτικά, πιο κάτω φαίνεται η περιγραφή των πειραμάτων και ο τρόπος με τον οποίο έτρεξαν τα πειράματα (Testbench).

2.1. Περιγραφή πειραμάτων

Για την παρούσα μελέτη, εκτελέσθηκαν επτά διαφορετικά πειράματα, εκ των οποίων μερικά αποτελούν παραλλαγή προηγούμενων πειραμάτων.

2.1.1 Πείραμα 1

Το πρώτο πείραμα (βλ. Παράρτημα Α) έχει σκοπό να ελέγξει κατά πόσο ένας υπολογιστής παίρνει κάθε φορά το ίδιο IP Address ή αν αυτό αλλάζει κάποιες φορές. Ο έλεγχος βασίζεται στο αν η σύνδεση γίνεται μέσω του TOR ή όχι.

2.1.2 Πείραμα 2

Το δεύτερο πείραμα (βλ. Παράρτημα Β) έχει σκοπό να ελέγξει κατά πόσο μπορεί να δημιουργηθεί ένα circuit στο TOR, με βάση κάποια στοιχεία που δίνονται από τον χρήστη. Δηλαδή, μέσα από τον κώδικα δίνονται τα fingerprints των nodes που θέλουμε να απαρτίζουν το circuit και το πρόγραμμα προσπαθεί να το δημιουργήσει. Το πρόγραμμα ενημερώνει τον χρήστη αν μπορεί να δημιουργηθεί το συγκεκριμένο circuit ή όχι.

2.1.3 Πείραμα 3

Το τρίτο πείραμα (βλ. Παράρτημα Γ) έχει σαν σκοπό την δημιουργία τυχαίων circuits στο TOR με βάση τα στοιχεία που καθορίζονται από τον χρήστη. Δηλαδή, το πρόγραμμα παίρνει σαν είσοδο από την γραμμή εντολής τον αριθμό nodes που θέλουμε να απαρτίζουν το circuit, και το fingerprint του exit node. Τότε, επιλέγονται τυχαία κάποια nodes, από τα nodes που είναι διαθέσιμα εκείνη την δεδομένη στιγμή με σκοπό να χρησιμοποιηθούν σαν ενδιάμεσα nodes. Εάν το κύκλωμα καταφέρει να δημιουργηθεί, το πρόγραμμα εκτυπώνει τα fingerprints των ενδιάμεσων nodes του circuit, και τον χρόνο που χρειάστηκε για να δημιουργηθεί το circuit (establishment time), αλλιώς εκτυπώνει μήνυμα αποτυχίας.

2.1.4 Πείραμα 4

Το τέταρτο πείραμα (βλ. Παράρτημα Ε) είναι ένα πείραμα το οποίο φτιάχνει τυχαία circuits στο TOR αναλόγως της εισόδου που δίνεται από τον χρήστη. Το πρόγραμμα παίρνει σαν είσοδο από την γραμμή εντολής τον αριθμό των nodes που θα απαρτίζουν το circuit και το fingerprint του exit node. Τότε, επιλέγονται τυχαία κάποια nodes, από τα

nodes που είναι διαθέσιμα εκείνη την δεδομένη στιγμή με σκοπό να χρησιμοποιηθούν σαν ενδιάμεσα nodes. Σαν έξοδο, το πρόγραμμα δημιουργεί ένα αρχείο Results.txt, το οποίο περιέχει τον αριθμό των nodes που απαρτίζουν το circuit, το fingerprint του exit node, τον χρόνο που χρειάστηκε για να δημιουργηθεί το circuit (establishment time), την πόλη και την χώρα στην οποία βρίσκεται το exit node. Σε περίπτωση που δεν μπορεί να δημιουργηθεί το circuit για οποιοδήποτε λόγο, το πρόγραμμα εκτυπώνει μήνυμα αποτυχίας μέσα στο αρχείο μαζί με τον αριθμό των nodes και το fingerprint του exit node.

2.1.5 Πείραμα 5

Το πέμπτο πείραμα (βλ. Παράρτημα Γ) είναι ένα πείραμα το οποίο προσπαθεί να δημιουργήσει circuits που έχουν διάφορα μεγέθη με τυχαία ενδιάμεσα nodes και προκαθορισμένο μέγεθος και exit node. Ο σκοπός αυτού του πειράματος είναι να βρεθεί το μέγεθος του circuit, με το οποίο δεν αποτυγχάνει ποτέ η δημιουργία circuit.

2.1.6 Πείραμα 6

Το έκτο πείραμα (βλ. Παράρτημα ΣΤ) έχει σαν σκοπό την δημιουργία τυχαίων circuits στο TOR αναλόγως της εισόδου που του δίνεται. Το πρόγραμμα παίρνει σαν είσοδο από την γραμμή εντολής τον αριθμό nodes που θα απαρτίζουν το circuit και το fingerprint του exit node. Επίσης, μέσα στον κώδικα καθορίζεται η ιστοσελίδα στην οποία θα γίνει το request. Σαν έξοδο, το πρόγραμμα δημιουργεί ένα αρχείο Results.txt, το οποίο περιέχει τα fingerprints όλων των nodes του circuit, τον αριθμό των nodes που απαρτίζουν το circuit, το fingerprint του exit node, τον χρόνο που χρειάστηκε για να δημιουργηθεί το circuit (establishment time), τον χρόνο που χρειάστηκε να εξυπηρετηθεί το request (serving time), την πόλη και την χώρα στην οποία βρίσκεται το exit node. Σε περίπτωση που δεν μπορεί να δημιουργηθεί το circuit για οποιοδήποτε λόγο, το πρόγραμμα εκτυπώνει μήνυμα αποτυχίας μέσα στο αρχείο μαζί με τον αριθμό των nodes και το fingerprint του exit node.

2.1.7 Βοηθητικό Script

Για να διευκολυνθεί η διαδικασία με τα πειράματα, γράφτηκε ένα επιπλέον script (βλ. Παράρτημα Δ), το οποίο φτιάχνει μια λίστα με τα active nodes. Δηλαδή, παίρνει σαν είσοδο από την γραμμή εντολής το μέγεθος λίστας που επιθυμούμε. Σαν έξοδο, το

πρόγραμμα δημιουργεί ένα αρχείο ActiveNodes.txt, το οποίο περιέχει την λίστα με τα στοιχεία που χρειαζόμαστε για N nodes (fingerprint, city, country).

2.2. Testbench

Στο συγκεκριμένο υποκεφάλαιο περιγράφεται ο τρόπος με τον οποίο έτρεξαν τα πειράματα.

2.2.1 Πείραμα 1

Για το πρώτο πείραμα, έτρεξαν τα δύο scripts (βλ. Παράρτημα Α) δέκα φορές το καθένα.

2.2.2 Πείραμα 2

Για το δεύτερο πείραμα, το script (βλ. Παράρτημα Β) έτρεξε δέκα φορές. Κάθε φορά που έτρεξε το συγκεκριμένο πείραμα, το exit node ήταν διαφορετικό, αλλά σε όλες τις περιπτώσεις το μέγεθος του circuit ήταν 5.

2.2.3 Πείραμα 3

Για το τρίτο πείραμα, το script (βλ. Παράρτημα Γ) έτρεξε συνολικά τριάντα φορές. Το script εκτελέσθηκε δέκα φορές για circuit μεγέθους 3, δέκα φορές για circuit μεγέθους 6 και δέκα φορές για circuit μεγέθους 9.

2.2.4 Πείραμα 4

Για το τέταρτο πείραμα, το script (βλ. Παράρτημα Ε) έτρεξε συνολικά διακόσιες φορές. Για πέντε διαφορετικά exit nodes, το script έτρεξε σαράντα φορές. Για κάθε exit node, το circuit είχε διαφορετικό μέγεθος (circuits μεγέθους 3, 6, 9 και 12 για το κάθε exit node). Για κάθε μέγεθος το script έτρεξε δέκα φορές.

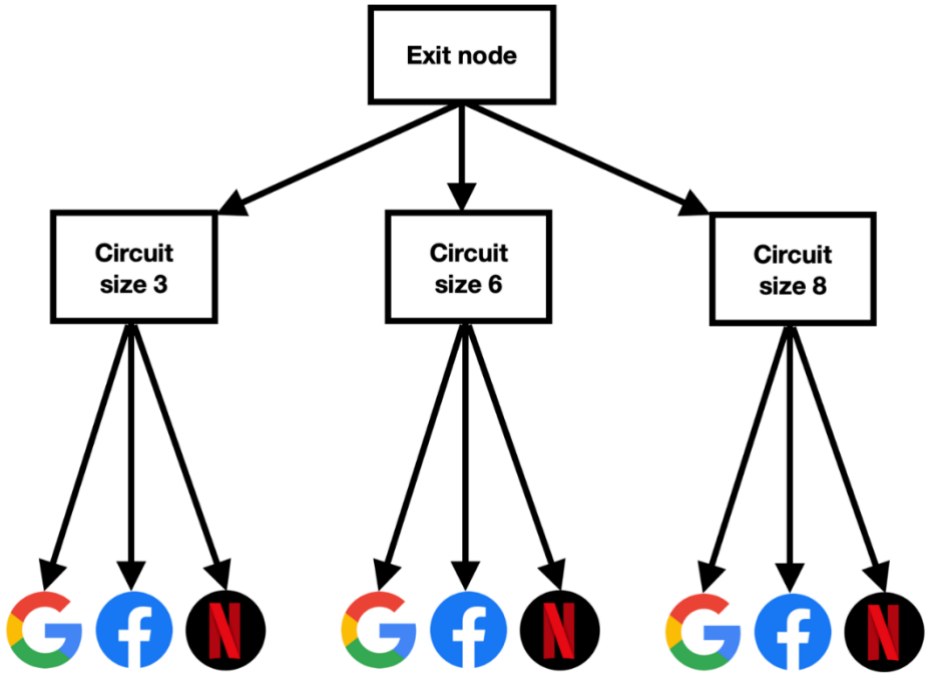
2.2.5 Πείραμα 5

Για το πέμπτο πείραμα, το script (βλ. Παράρτημα Γ) έτρεξε συνολικά διακόσιες φορές. Για δέκα διαφορετικά exit nodes, το script έτρεξε είκοσι φορές. Για κάθε exit node, το script έτρεξε με τέσσερα διαφορετικά μεγέθη (circuits μεγέθους 7, 8, 9 και 10), πέντε

φορές για κάθε μέγεθος.

2.2.6 Πείραμα 6

Για το έκτο πείραμα, το script (βλ. Παράρτημα Z) έτρεξε συνολικά ενενήντα φορές. Για δέκα διαφορετικά exit nodes, το script έτρεξε εννέα φορές. Κάθε φορά που έτρεξε το script για το ίδιο exit node, άλλαζε το μέγεθος του circuit(circuits μεγέθους 3, 6 και 8 για το κάθε exit node), και η ιστοσελίδα στην οποία γινόταν το request (3 διαφορετικές ιστοσελίδες). Πιο κάτω (*Σχήμα 2.2*) φαίνεται ο τρόπος με τον οποίο γίνεται το πείραμα για ένα exit node.



Σχήμα 2.2 Γραφική αναπαράσταση Testbench πειράματος 6

Κεφάλαιο 3

Υλοποίηση

3.1 Python 17
3.2 Βιβλιοθήκη Stem 18
3.3 Υλοποίηση πειραμάτων 18
3.3.1 Πείραμα 1 18
3.3.2 Πείραμα 2 18
3.3.3 Πείραμα 3 19
3.3.4 Πείραμα 4 19
3.3.5 Πείραμα 5 20
3.3.6 Πείραμα 6 20
3.3.7 Βοηθητικό Script 21

Σε αυτό το κεφάλαιο περιγράφεται η υλοποίηση των scripts που γράφτηκαν για την εκπόνηση της παρούσας ατομικής διπλωματική εργασίας. Συγκεκριμένα, φαίνονται αναλυτικά για κάθε πείραμα οι πιο σημαντικές λειτουργίες. Τα scripts είναι γραμμένα στην γλώσσα προγραμματισμού Python σε συνδυασμό με την βιβλιοθήκη Stem.

3.1. Python

Η Python είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου. Η συγκεκριμένη γλώσσα είναι γενικής χρήσης γλώσσα προγραμματισμού. Σχεδιάστηκε από τον Guido van Rossum και η πρώτη έκδοση της κυκλοφόρησε το 1991. Σήμερα είναι μια από τις πιο διαδεδομένες γλώσσες προγραμματισμού χάρη στην ευχρηστία της και στο γεγονός ότι δίνει πολλές δυνατότητες στους χρήστες της με την πληθώρα βιβλιοθηκών που υποστηρίζει. Όλα τα scripts της παρούσας ατομικής διπλωματικής εργασίας είναι γραμμένα σε Python.

3.2. Βιβλιοθήκη Stem

Η βιβλιοθήκη Stem είναι μια controller library της Python που σχεδιάστηκε αποκλειστικά για το TOR. Με την συγκεκριμένη βιβλιοθήκη μπορούμε να πάρουμε πληροφορίες για το TOR μέσω κώδικα. Πιο κάτω φαίνεται ο τρόπος με τον οποίο έγινε χρήση της συγκεκριμένης βιβλιοθήκης έτσι ώστε να δημιουργήσουμε διάφορα circuits ή και να πάρουμε πληροφορίες σχετικά με το ποια nodes είναι ενεργά και διαθέσιμα, ή ακόμα και την τοποθεσία ενός node.

3.3. Υλοποίηση Πειραμάτων

3.3.1. Πείραμα 1

Στο πρώτο πείραμα έχουμε δύο scripts τα οποία επιστρέφουν την IP διεύθυνση με την βοήθεια της βιβλιοθήκης requests. Στο πρώτο script γίνεται το request μέσω μιας εντολής, όπως φαίνεται πιο κάτω.

```
ip = requests.get('https://ident.me').text
```

Σχήμα 3.1 Κώδικας υπεύθυνος για λήψη IP address

Στο δεύτερο script χρησιμοποιούνται εντολές από την βιβλιοθήκη Stem, και έτσι το request γίνεται μέσω του TOR με τις πιο κάτω εντολές:

```
proxies = {
    'http': 'socks5://127.0.0.1:9050',
    'https': 'socks5://127.0.0.1:9050'
}

with Controller.from_port(port = 9051) as c:
    c.authenticate()
    c.signal(Signal.NEWNYM)

ip=requests.get("https://api.ipify.org", proxies=proxies).text
```

Σχήμα 3.2 Κώδικας υπεύθυνος για λήψη IP address

3.3.2. Πείραμα 2

Στο δεύτερο πείραμα ελέγχεται κατά πόσο μπορεί να δημιουργηθεί ένα circuit το οποίο απαρτίζεται από συγκεκριμένα nodes που δίνονται από τον χρήστη. Πιο κάτω φαίνεται το σημείο, όπου γίνεται η προσπάθεια δημιουργίας του circuit και η εκτύπωση του αποτελέσματος. Η δημιουργία και ο έλεγχος αν το circuit έχει δημιουργηθεί, γίνονται με την χρήση εντολών της βιβλιοθήκης Stem.

```
with stem.control.Controller.from_port() as controller:
    controller.authenticate()

    controller.new_circuit(path, await_build=True)

    print(controller.get_info('circuit-status'))
```

Σχήμα 3.3 Κώδικας υπεύθυνος για την δημιουργία circuit

3.3.3. Πείραμα 3

Στο τρίτο πείραμα δημιουργείται ένα circuit το οποίο έχει προκαθορισμένο μέγεθος και exit node. Τα ενδιάμεσα nodes επιλέγονται τυχαία από τα διαθέσιμα ενεργά nodes που υπάρχουν εκείνη την στιγμή. Το μέγεθος του circuit και το exit node δίνονται από τον χρήστη μέσω της γραμμής εντολής. Πιο κάτω φαίνεται ο τρόπος με τον οποίο επιλέγονται τα ενδιάμεσα nodes.

```
for desc in controller.get_network_statuses():
    descriptors.append(desc.fingerprint)

for i in range(0,int(sys.argv[1])-1):
    rindex = random.randint(0,len(descriptors))
    path.append(descriptors[rindex])
```

Σχήμα 3.4 Κώδικας υπεύθυνος για εύρεση ενεργών nodes και τυχαία επιλογή N-1 nodes

3.3.4. Πείραμα 4

Στο τέταρτο πείραμα δημιουργείται ένα circuit το οποίο έχει προκαθορισμένο μέγεθος και exit node. Τα ενδιάμεσα nodes επιλέγονται τυχαία από τα διαθέσιμα ενεργά nodes που υπάρχουν εκείνη την στιγμή. Το μέγεθος του circuit και το exit node δίνονται από τον χρήστη μέσω της γραμμής εντολής. Το πρόγραμμα επιστρέφει σαν αποτέλεσμα ένα αρχείο Results.txt, το οποίο περιέχει το μέγεθος του circuit, το exit node fingerprint, τον χρόνο που χρειάστηκε για να δημιουργηθεί το circuit, και την τοποθεσία του exit node. Πιο κάτω φαίνονται οι εντολές οι οποίες είναι υπεύθυνες για την εύρεση της τοποθεσίας του exit node.

```
reqURL = 'http://api.geoiplookup.net/?query='
ipAddr = controller.get_network_status(sys.argv[2])
ipAddr = ipAddr.address
reqURL += ipAddr
req = requests.get(reqURL)
root = ET.fromstring(req.text)
for child in root[0][0]:
    if child.tag == "city":
        exitLocation = child.text
    if child.tag == "countryname":
        exitLocation += ', '+child.text
```

Σχήμα 3.5 Κώδικας υπεύθυνος για εξεύρεση τοποθεσίας ενός node

3.3.5. Πείραμα 5

Στο πέμπτο πείραμα εκτελείται το ίδιο script πολλές φορές. Κάθε φορά που τρέχει το script, δημιουργείται ένα circuit το οποίο έχει προκαθορισμένο μέγεθος και exit node. Τα ενδιάμεσα nodes επιλέγονται τυχαία από τα διαθέσιμα ενεργά nodes που υπάρχουν εκείνη την στιγμή. Το μέγεθος του circuit και το exit node δίνονται από τον χρήστη μέσω της γραμμής εντολής. Πιο κάτω φαίνεται ο τρόπος με τον οποίο επιλέγονται τα ενδιάμεσα nodes.

```
for desc in controller.get_network_statuses():
    descriptors.append(desc.fingerprint)

for i in range(0,int(sys.argv[1])-1):
    rindex = random.randint(0,len(descriptors))
    path.append(descriptors[rindex])
```

Σχήμα 3.6 Κώδικας υπεύθυνος για εύρεση ενεργών nodes και τυχαία επιλογή N-1 nodes

3.3.6. Πείραμα 6

Στο έκτο πείραμα δημιουργείται ένα circuit το οποίο έχει προκαθορισμένο μέγεθος και exit node και γίνεται request σε μια προκαθορισμένη ιστοσελίδα. Τα ενδιάμεσα nodes επιλέγονται τυχαία από τα διαθέσιμα ενεργά nodes που υπάρχουν εκείνη την στιγμή. Το μέγεθος του circuit και το exit node δίνονται από τον χρήστη μέσω της γραμμής εντολής. Η ιστοσελίδα στην οποία θα γίνει το request καθορίζεται μέσα από τον κώδικα, όπως φαίνεται πιο κάτω (*Σχήμα 3.6*). Το πρόγραμμα επιστρέφει σαν αποτέλεσμα ένα αρχείο Results.txt, το οποίο περιέχει το μέγεθος του circuit, το exit node fingerprint, τον χρόνο που χρειάστηκε για να δημιουργηθεί το circuit, τον χρόνο που χρειάστηκε για να εξυπηρετηθεί το request, και την τοποθεσία του exit node.

```
tempReq = requests.get('https://www.netflix.com/')
```

Σχήμα 3.7 Δημιουργία request προς μια συγκεκριμένη ιστοσελίδα μέσα από τον κώδικα

3.3.7. Βοηθητικό Script

Το συγκεκριμένο script γράφτηκε για να δημιουργεί μια λίστα με τα ενεργά nodes που είναι διαθέσιμα την δεδομένη στιγμή. Ο χρήστης εισάγει στην γραμμή εντολής ένα αριθμό N, ο οποίος αντιστοιχεί στο πλήθος των ενεργών nodes που θέλει να επιστραφούν σαν αποτέλεσμα από το script. Εφόσον τρέξει, το πρόγραμμα επιστρέφει ένα αρχείο ActiveNodes.txt, το οποίο περιέχει N node fingerprints που είναι ενεργά εκείνη την στιγμή. Πιο κάτω φαίνεται ο τρόπος με τον οποίο το script ανιχνεύει τα ενεργά nodes.

```
for desc in controller.get_network_statuses():
    descriptors.append(desc.fingerprint)

for i in range(0,int(sys.argv[1])):
    rindex = random.randint(0,len(descriptors))
    path.append(descriptors[rindex])
    reqURL = 'http://api.geoiplookup.net/?query='
    ipAddr = controller.get_network_status(descriptors[rindex])
    ipAddr = ipAddr.address
    reqURL += ipAddr
    req = requests.get(reqURL)
    root = ET.fromstring(req.text)

    for child in root[0][0]:
        if child.tag == "city":
            path.append(child.tag)
            path.append(child.text)
            results.write(descriptors[rindex]+', '+str(child.tag)+': '+str(child.text)+', ')
        if child.tag == "countryname":
            path.append(child.tag)
            path.append(child.text)
            results.write(str(child.tag)+': '+str(child.text)+'\n')
```

Σχήμα 3.8 Κώδικας υπεύθυνος για εξεύρεση N ενεργών nodes

Κεφάλαιο 4

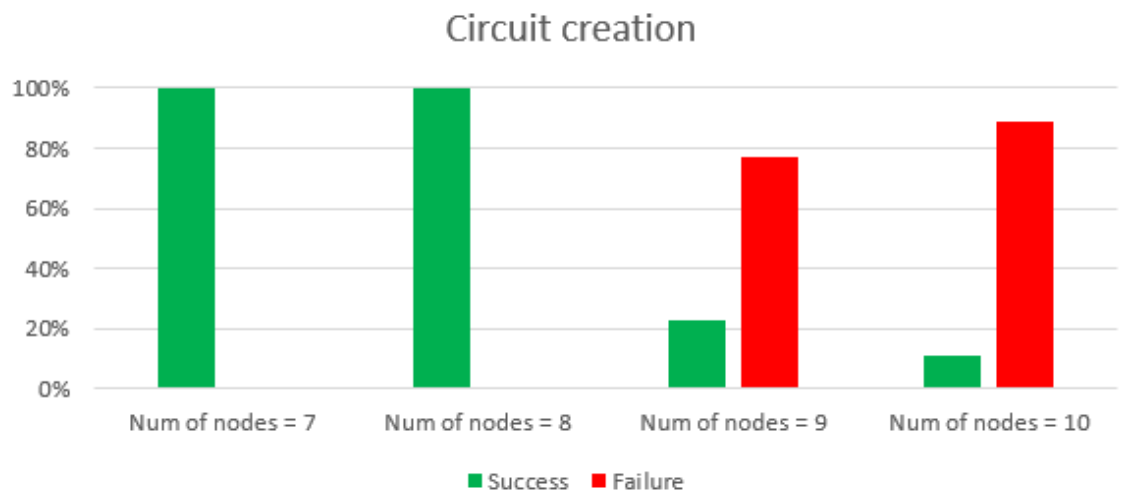
Αξιολόγηση πειραμάτων

4.1 Μέγεθος circuit 22

4.2 Χρόνος δημιουργίας circuit 23

4.3 Χρόνος εξυπηρέτησης request 24

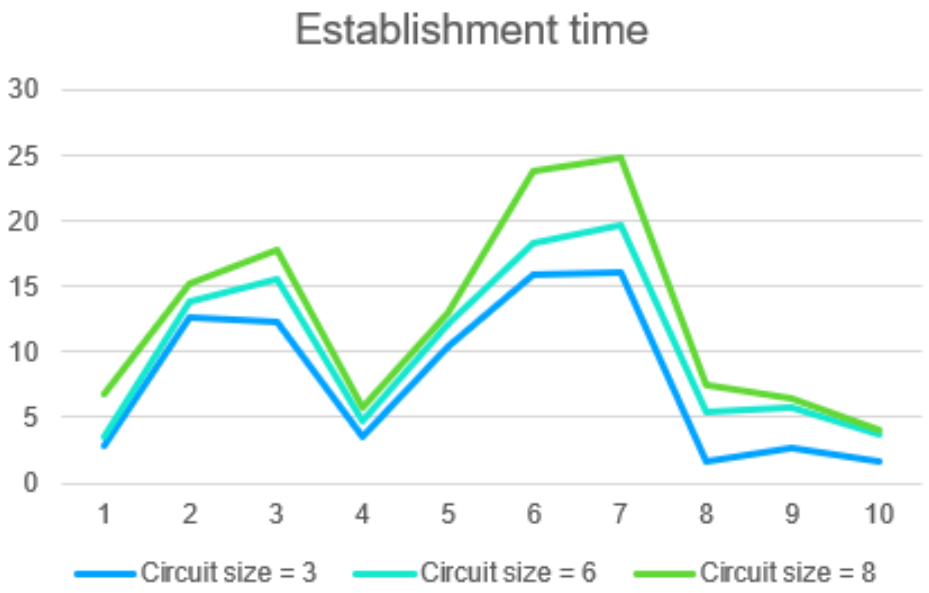
4.1 Μέγεθος circuit



Σχήμα 4.1 Αποτελέσματα πειράματος 5

Στο σχήμα 4.1 φαίνονται τα αποτελέσματα από το πείραμα 5. Σύμφωνα με τα αποτελέσματα του πειράματος, ο μέγιστος αριθμός nodes που μπορεί να έχει ένα circuit έτσι ώστε να δημιουργηθεί σίγουρα είναι 8. Στις πλείστες περιπτώσεις για αριθμό nodes μεγαλύτερο του 8, το circuit δεν μπορεί να δημιουργηθεί λόγω time out error. Συγκεκριμένα, για μέγεθος 9, στο 77% των περιπτώσεων το circuit απέτυχε να δημιουργηθεί και για μέγεθος 10, η αποτυχία ήταν στο 89%.

4.2 Χρόνος δημιουργίας circuit



Σχήμα 4.2 Γραφική αναπαράσταση αποτελεσμάτων πειράματος 6 – Establishment time

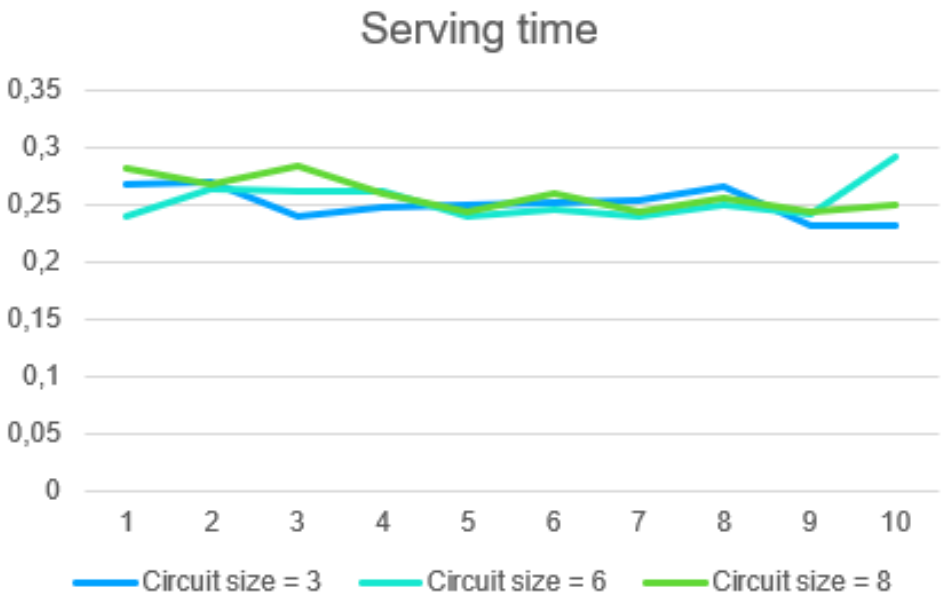
Αρ.	Exit node fingerprint	City, Country
1	06F9E86930823CA86541DDFEB9827A22B9A7A4AD	Singapore, Singapore
2	E00EE4CDB600ACFA7B000CE887B3D8BAF8C3A2DB	Victoria, Canada
3	2A2022D386894D3CB3438752977C27A7AF50BAFA	Central, Hong Kong
4	05299E38F266FF4AFBBAF67B4502B712CE28C339	Moscow, Russia
5	A2166EBF7C49ED87F29823EC3B3FF36D21DBC070	Mumbai, India
6	3CC54860050C2DFB1BB280E4D7F620930D23BFF0	Sydney, Australia
7	D9ABEB7511679FFAA665F65D4A1F94280C859823	New York, United States
8	8EE28EAF0874AF9B03EBBD79B5EB5059E2547355	Amsterdam, Netherlands
9	AA7630738F82C4DF26BB8EFF131564DACDEAC64C	Bergen, Norway
10	1E6ED5FE5F0DAC6B2488CD66C643BFCDF0D4F903	Paris, France

Σχήμα 4.3 Πληροφορίες / Επεξήγηση Σχήματος 4.2

Πιο πάνω φαίνονται τα αποτελέσματα του πειράματος 6. Στο σχήμα 4.2 φαίνονται οι μέσοι χρόνοι (seconds) για 10 διαφορετικά exit nodes και για 3 διαφορετικά μεγέθη circuit. Συγκεκριμένα, τα exit nodes και οι χώρες στις οποίες βρίσκονται, αναγράφονται

στον πίνακα πιο πάνω (*Σχήμα 4.3*). Παρατηρούμε ότι για όλα τα exit nodes ο χρόνος που χρειάζεται το circuit για να δημιουργηθεί είναι ανάλογος του αριθμού nodes που το απαρτίζουν. Ακόμα, παρατηρούμε ότι υπάρχει απόκλιση στους χρόνους μεταξύ των 10 διαφορετικών nodes. Σύμφωνα με τα δεδομένα που έχουμε, μπορούμε να φτάσουμε στο συμπέρασμα ότι ο χρόνος που χρειάζεται ένα circuit να δημιουργηθεί είναι ανάλογος της απόστασης του exit node από το αρχικό node του circuit, δηλαδή του χρήστη.

4.3 Χρόνος εξυπηρέτησης request



Σχήμα 4.4 Γραφική αναπαράσταση αποτελεσμάτων πειράματος 6 – Serving time

Αρ.	Exit node fingerprint	City, Country
1	06F9E86930823CA86541DDFEB9827A22B9A7A4AD	Singapore, Singapore
2	E00EE4CDB600ACFA7B000CE887B3D8BAF8C3A2DB	Victoria, Canada
3	2A2022D386894D3CB3438752977C27A7AF50BAFA	Central, Hong Kong
4	05299E38F266FF4AFBBAF67B4502B712CE28C339	Moscow, Russia
5	A2166EBF7C49ED87F29823EC3B3FF36D21DBC070	Mumbai, India
6	3CC54860050C2DFB1BB280E4D7F620930D23BFF0	Sydney, Australia
7	D9ABEB7511679FFAA665F65D4A1F94280C859823	New York, United States
8	8EE28EAF0874AF9B03EBBD79B5EB5059E2547355	Amsterdam, Netherlands
9	AA7630738F82C4DF26BB8EFF131564DACDEAC64C	Bergen, Norway
10	1E6ED5FE5F0DAC6B2488CD66C643BFCDF0D4F903	Paris, France

Σχήμα 4.5 Πληροφορίες / Επεξήγηση Σχήματος 4.4

Πιο πάνω φαίνονται τα αποτελέσματα του πειράματος 6. Στο σχήμα 4.4 φαίνονται οι μέσοι χρόνοι (seconds) για 10 διαφορετικά exit nodes και για 3 διαφορετικά μεγέθη circuit. Συγκεκριμένα, τα exit nodes και οι χώρες στις οποίες βρίσκονται, αναγράφονται στον πίνακα πιο πάνω (*Σχήμα 4.5*). Παρατηρούμε ότι για όλα τα exit nodes ο χρόνος που χρειάζεται το request για να εξυπηρετηθεί είναι περίπου ο ίδιος. Άρα, μπορούμε να καταλήξουμε στο συμπέρασμα ότι ο χρόνος εξυπηρέτησης ενός request είναι ανεξάρτητος της τοποθεσίας του exit node.

Κεφάλαιο 5

Συμπεράσματα

5.1 Μελλοντική δουλειά 26
5.2 Τελικά σχόλια 26

5.1 Μελλοντική δουλειά

Στο μέλλον θα μπορούσαμε να παίρνουμε την λίστα με τα διαθέσιμα ενεργά nodes και να τα ταξινομούμε με βάση την απόσταση τους από τον χρήστη, έτσι ώστε να ελαχιστοποιηθεί το establishment time.

Ακόμα, θα μπορούσαμε να τρέξουμε πειράματα με μέγεθος circuit 1-8 για exit nodes που βρίσκονται σε διαφορετικές χώρες μεταξύ τους, έτσι ώστε να βρούμε το βέλτιστο μέγεθος circuit για κάθε χώρα.

Επίσης, πιστεύω ότι θα ήταν ενδιαφέρον να τρέξουμε κάποια από τα πειράματα περιορίζοντας τα nodes σε συγκεκριμένη χώρα ή και ήπειρο έτσι ώστε να μελετηθεί σε μεγαλύτερο βάθος ο τρόπος με τον οποίο γίνονται τα circuits στο TOR.

5.2 Τελικά σχόλια

Με την εκπόνηση της παρούσας ατομικής διπλωματικής εργασίας, έχω πάρει μια γενική εικόνα του TOR ως δίκτυο. Τρέχοντας τα προαναφερόμενα πειράματα, έχω καταλήξει σε κάποια βασικά συμπεράσματα σχετικά με την δημιουργία των circuits και την εξυπηρέτηση των requests στο TOR. Αρχικά, έχω προσέξει ότι ο χρόνος που χρειάζεται ένα circuit για να δημιουργηθεί εξαρτάται από τον αριθμό των nodes που το απαρτίζουν αλλά και από την τοποθεσία των exit nodes. Ακόμα, μέσα από τα αποτελέσματα των πειραμάτων κατάλαβα ότι ο χρόνος που χρειάζεται ένα request για να εξυπηρετηθεί είναι ανεξάρτητος του circuit μέσα από το οποίο γίνεται.

Βιβλιογραφία

[1] https://www.torproject.org/about/history/

[2] M. Bernaschi, A. Celestini, S. Guarino and F. Lombardi. Exploring and Analyzing the Tor Hidden Services Graph. ACM Transaction on the Web 11(4):1-26, July 2017

[3] https://www.geeksforgeeks.org/onion-routing/

Παράρτημα Α

Script 1

```
import requests
ip = requests.get('https://ident.me').text
print('IP : ',ip)
```

Script 2

```
import requests
from stem import Signal
from stem.control import Controller
proxies = {
    'http': 'socks5://127.0.0.1:9050',
    'https': 'socks5://127.0.0.1:9050'
}
with Controller.from_port(port = 9051) as c:
    c.authenticate()
    c.signal(Signal.NEWNYM)
ip = requests.get('https://api.ipify.org', proxies=proxies).text
print('IP : ',ip)
```

Παράρτημα Β

```
#Author : Andriana Chrysostomou
#Creates a TOR circuit manually with the relays given in 'path'.

import stem.control

path = ['F6740DEABFD5F62612FA025A5079EA72846B1F67',
'6C333B5BDBA3DFE7A782E40CB970C271ABFED119',
'78E584CEC6BDD9D38D573A7701936F2F4AED93BA',
'9B2BC7EFD661072AFADC533BE8DCF1C19D8C2DCC',
'56781CCC9F6D29FEA148799AB588429C893A473C']


with stem.control.Controller.from_port() as controller:
    controller.authenticate()

    controller.new_circuit(path, await_build=True)

    print(controller.get_info('circuit-status'))
```

Παράρτημα Γ

```
#Author : Andriana Chrysostomou
#Creates a random TOR circuit and prints the path. The circuit
size and the exit node fingerprint are given by the user.

import stem.control
import sys
import random
import time

path = []
descriptors = []

start = time.time()
with stem.control.Controller.from_port() as controller:
    controller.authenticate()

    for desc in controller.get_network_statuses():
        descriptors.append(desc.fingerprint)

    for i in range(0,int(sys.argv[1])-1):
        rindex = random.randint(0,len(descriptors))
        path.append(descriptors[rindex])


    path.append(sys.argv[2])
    print(path)
    try:
        controller.new_circuit(path, await_build=True)
        #print(controller.get_info('circuit-status'))
    except:
        print('You shall not pass!')

end = time.time()
timeNeeded = end-start
print('Time needed : '+str(timeNeeded))
```

Παράρτημα Δ

```
#Author : Andriana Chrysostomou
#Creates a list with exit fingerprints and country and city
where they are located.
#Creates 'ActiveNodes.txt' , which contains the output
(exitFingerprint, city, country)

import stem.control
import sys
import random
import time
import requests
import xml.etree.ElementTree as ET

path = []
descriptors = []
results = open('ActiveNodes.txt',"a")

with stem.control.Controller.from_port() as controller:
    controller.authenticate()

    for desc in controller.get_network_statuses():
        descriptors.append(desc.fingerprint)

    for i in range(0,int(sys.argv[1])):
        rindex = random.randint(0,len(descriptors))
        path.append(descriptors[rindex])
        reqURL = 'http://api.geoiplookup.net/?query='
        ipAddr =
controller.get_network_status(descriptors[rindex])
        ipAddr = ipAddr.address
        reqURL += ipAddr
        req = requests.get(reqURL)
        root = ET.fromstring(req.text)

        for child in root[0][0]:
              if child.tag == "city":
                    path.append(child.tag)
                    path.append(child.text)
                    results.write(descriptors[rindex]+',
'+str(child.tag)+': '+str(child.text)+', ')
              if child.tag == "countryname":
                    path.append(child.tag)
                    path.append(child.text)
                    results.write(str(child.tag)+':
'+str(child.text)+'\n')

results.close()
```

Παράρτημα Ε

```
#Author : Andriana Chrysostomou
#Creates a random circuit and calculates the time needed for the
circuit to be created.
#Also, it finds the country and city where the exit node is
located.
#The circuit size and the exit node fingerprint are given by the
user.
#Creates 'Results.txt' , which contains the output (number of
hops, exit node fingerprint, time needed, exit node location)

import stem.control
import sys
import random
import time
import requests
import xml.etree.ElementTree as ET

path = []
descriptors = []
results = open('Results.txt',"a")
start = time.time()

with stem.control.Controller.from_port() as controller:
    controller.authenticate()

    for desc in controller.get_network_statuses():
        descriptors.append(desc.fingerprint)

    for i in range(0,int(sys.argv[1])-1):
        rindex = random.randint(0,len(descriptors))
        path.append(descriptors[rindex])

    path.append(sys.argv[2])
    print(path)
    try:
        controller.new_circuit(path, await_build=True)
        end = time.time()
        timeNeeded = end-start
        reqURL = 'http://api.geoiplookup.net/?query='
        ipAddr = controller.get_network_status(sys.argv[2])
        ipAddr = ipAddr.address
        reqURL += ipAddr
        req = requests.get(reqURL)
        root = ET.fromstring(req.text)
        for child in root[0][0]:
              if child.tag == "city":
                     exitLocation = child.text
              if child.tag == "countryname":
                     exitLocation += ', '+child.text
```

```
        results.write('Number of hops :
'+str(sys.argv[1])+'\tExit node : '+str(sys.argv[2])+'\tTime
needed : '+' '+str(timeNeeded)+'\tExit node location :
'+str(exitLocation)+'\n')
    except:
        print('You shall not pass!')
        results.write('Number of hops :
'+str(sys.argv[1])+'\tExit node : '+str(sys.argv[2])+'\tStatus :
Failed\n')

results.close()
```

Παράρτημα ΣΤ

```
#Author : Andriana Chrysostomou
#Creates a random circuit and calculates the time needed for the
circuit to be created and the time needed for a request to a
website given in the code to be served.
#Also, it finds the country and city where the exit node is
located.
#The circuit size and the exit node fingerprint are given by the
user.
#Creates 'Results.txt' , which contains the output (number of
hops, exit node fingerprint, establishment time, serving time,
exit node location)

import stem.control
import sys
import random
import time
import requests
import xml.etree.ElementTree as ET

path = []
descriptors = []
results = open('Results.txt',"a")
estStartT = time.time()

with stem.control.Controller.from_port() as controller:
    controller.authenticate()

    for desc in controller.get_network_statuses():
        descriptors.append(desc.fingerprint)

    results.write('Path : ')
    for i in range(0,int(sys.argv[1])-1):
        rindex = random.randint(0,len(descriptors))
        path.append(descriptors[rindex])
        results.write(str(descriptors[rindex])+', ')

    path.append(sys.argv[2])
    print(path)
    results.write(str(sys.argv[2])+'\n')

    try:
        controller.new_circuit(path, await_build=True)
        servStartT = time.time()
        tempReq = requests.get('https://www.netflix.com/')
        servEndT = time.time()
        estEndT = time.time()
        establishmentT = estEndT - estStartT
        servingT = servEndT - servStartT
        reqURL = 'http://api.geoiplookup.net/?query='
        ipAddr = controller.get_network_status(sys.argv[2])
        ipAddr = ipAddr.address
```

```
        reqURL += ipAddr
        req = requests.get(reqURL)
        root = ET.fromstring(req.text)
        for child in root[0][0]:
              if child.tag == "city":
                     exitLocation = child.text
              if child.tag == "countryname":
                     exitLocation += ', '+child.text

        results.write('Number of hops :
'+str(sys.argv[1])+'\tExit node :
'+str(sys.argv[2])+'\tEstablishment time : '+'
'+str(establishmentT)+'\tServing time : '+'
'+str(servingT)+'\tExit node location :
'+str(exitLocation)+'\n')
    except:
        print('You shall not pass!')
        results.write('Number of hops :
'+str(sys.argv[1])+'\tExit node : '+str(sys.argv[2])+'\tStatus :
Failed\n')

results.close()
```