Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Επικοινωνιων, Ηλεκτρονικης και Συστηματων Πληροφορικης
NETMODE (Network Management & Optimal Design Laboratory)

Ανίχνευση Κίνησης DGA-Based Botnet με Μεθόδους Federated Learning

Μελέτη και υλοποίηση

Διπλωματικη Εργασια

του

ΓΕΩΡΓΙΟΥ Χ. ΣΟΥΛΙΩΤΗ

Επιβλέπων: Συμεών Παπαβασιλείου
Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2020

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Επικοινωνιων, Ηλεκτρονικης και Συστηματων Πληροφορικης
NETMODE (Network Management & Optimal Design Laboratory)

Ανίχνευση Κίνησης DGA-Based Botnet με Μεθόδους Federated Learning

Μελέτη και υλοποίηση

Διπλωματικη Εργασια

του

ΓΕΩΡΓΙΟΥ Χ. ΣΟΥΛΙΩΤΗ

Επιβλέπων: Συμεών Παπαβασιλείου
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 5η Νοεμβρίου 2020.

(Υπογραφή)	*(Υπογραφή)*	*(Υπογραφή)*
.....	.....	.....
Συμεών Παπαβασιλείου	Ευστάθιος Συκάς	Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.	Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2020

Εθνικο Μετσοβιο Πολυτεχνειο
Σχολη Ηλεκτρολογων Μηχανικων και Μηχανικων Υπολογιστων
Τομεας Επικοινωνιων, Ηλεκτρονικης και Συστηματων Πληροφορικης
NETMODE (Network Management & Optimal Design Laboratory)

Δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάσει επιστημονικής παράφρασης.

(Υπογραφή)

.....
Γεώργιος Σουλιώτης

5 Νοεμβρίου 2020

Περίληψη

Σήμερα, ένα μεγάλο ποσοστό των botnets χρησιμοποιούν Domain Generation Algorithms (DGAs), για να αποκρύπτουν την ταυτότητά τους μέσω της περιοδικής εναλλαγής του domain name που εκχωρείται στον C&C server. Σημαντικό ρόλο στην υλοποίηση των DGA παίζει το πρωτόκολλο DNS, του οποίου η κίνηση δεν αποκόπτεται από τα firewalls. Με την τεχνική αυτή, η κακόβουλη κίνηση μπορεί να παρακάμπτει τα στατικά συστήματα ασφαλείας, ενώ ο εντοπισμός του C&C και η αποκοπή του από τα bots του καθίσταται εξαιρετικά απαιτητική διαδικασία. Έχει παρατηρηθεί ότι η μορφή των domain names που παράγονται από DGA διαφέρει σημαντικά από αυτή των νόμιμων ονομάτων. Εκμεταλλευόμενοι την πληροφορία αυτή πολλοί ερευνητές έχουν στραφεί στην ανάπτυξη ταξινομητών με μεθόδους Deep Learning, με απώτερο σκοπό την ανίχνευση αλγοριθμικά παραγόμενων domain names. Συνήθως για την εκπαίδευση των ανιχνευτών χρησιμοποιούνται δεδομένα που προκύπτουν από την ανάλυση εγγραφών Passive DNS.

Ωστόσο, η καταπολέμηση των botnets που βασίζονται σε DGA, είναι μια πρόκληση την οποία για να ξεπεράσουν οι οργανισμοί ασφαλείας συχνά χρειάζεται να συνεργαστούν και πιθανώς να μοιραστούν τα δεδομένα που διαθέτουν, για την εκπαίδευση αρτιότερων και πιο ενημερωμένων μοντέλων. Εντούτοις, κάτω από τις παραδοσιακές συνθήκες εκπαίδευσης με Distributed Deep Learning, όπου τα δεδομένα εκπαίδευσης εκτίθενται σε έναν κεντρικό server, ο εμπορικός ανταγωνισμός και τα αυστηρά πρωτόκολλα ιδιωτικότητας αποτελούν τροχοπέδη στη συνεργασία αυτή. Προκειμένου να εξαλείψουμε τις ανησυχίες σχετικά με το απόρρητο και την ασφάλεια των δεδομένων, προτείνουμε σε αυτή τη διπλωματική ένα περιβάλλον συνεργατικής εκπαίδευσης που βασίζεται στη σύγχρονη αρχιτεκτονική του Federated Learning. Κατά την εκπαίδευση με Federated Learning, τα μόνα δεδομένα που ανταλλάσσονται, είναι τα τοπικά μοντέλα που εκπαιδεύονται σε κάθε client και αποστέλλονται σε έναν κεντρικό server για να συμψηφιστούν σε ένα νέο γενικό μοντέλο. Τα δεδομένα εκπαίδευσης παραμένουν προστατευμένα στις τοπικές συσκευές καθ' όλη τη διάρκεια της εκπαίδευσης. Επομένως, ένα σύστημα βασισμένο σε αυτή τη φιλοσοφία φαίνεται να ανταποκρίνεται πλήρως στις ανάγκες που περιγράφηκαν προηγουμένως. Για τον λόγο αυτό, αναπτύσσουμε στην παρούσα διπλωματική μια πειραματική διάταξη Federated Learning για την εκπαίδευση τριών Deep Learning μοντέλων (CNN, LSTM, Bidirectional LSTM). Σκοπός μας είναι να εξάγουμε τα απαραίτητα συμπεράσματα σχετικά με τις προοπτικές και τους περιορισμούς της τεχνικής αυτής, σε ό,τι αφορά την ανίχνευση DGA domain names.

Λέξεις Κλειδιά

Botnet, Domain Generation Algorithm (DGA), Deep Learning, Federated Learning,

Federated Averaging, Passive DNS, Recursive DNS Server

Abstract

Nowadays, a great number of botnets leverage on Domain Generation Algorithms (DGAs) to hide their identity through the periodic switching of the domain name assigned to the C&C server. An important role in the implementation of DGAs is played by the DNS protocol, which is not filtered by firewalls. With this technique, malicious traffic can bypass static security systems, while locating the C&C server and preventing communications with its bots becomes an extremely demanding procedure. At the same time, it has been observed that the format of domain names produced by DGAs significantly differs from that of legal names. Thus, many researchers have developed classifiers based on Deep Learning methods, with the ultimate goal of detecting algorithmically generated domain names. Data derived from Passive DNS record analysis is usually used to train such classifiers.

Nevertheless, confronting DGA-based botnets is a challenge that security agencies must cooperate to overcome and possibly share their data to train better and more up-to-date models. However, under the traditional training conditions with Distributed Deep Learning, where training data is exposed to a central server, commercial competition and strict privacy protocols are a barrier to this cooperation. The issue of data privacy and security was the source of inspiration for this thesis. To eliminate the concern, as well as to provide additional motivations for the interested organizations to participate in a collaborative training environment of their classifiers, we take advantage of the modern architecture of Federated Learning. During the federated learning training process, the only data exchanged is the local models that are trained in each client and delivered to a central server to be aggregated to a new global model. Training data remain protected on local devices throughout the training process. Therefore, a system based on this philosophy seems to fully meet the requirements described previously. For this reason, we are developing in this thesis an experimental Federated Learning environment for the needs of training three Deep Learning models (CNN, LSTM, Bidirectional LSTM), in order to reach the necessary conclusions regarding with the pros and cons of this technique, in terms of detecting DGA domain names.

Keywords

Botnet, Domain Generation Algorithm (DGA), Deep Learning, Federated Learning, Federated Averaging, Passive DNS, Recursive DNS Server

στους γονείς μου

Ευχαριστίες

Θα ήθελα καταρχήν να ευχαριστήσω τον Καθηγητή Ε.Μ.Π. κ. Συμεών Παπαβασιλείου για την επίβλεψη αυτής της διπλωματικής εργασίας. Ευχαριστώ τον Ομότιμο Καθηγητή Ε.Μ.Π. κ. Βασίλειο Μάγκλαρη που μου πρότεινε την εργασία αυτή και μου έδωσε την ευκαιρία να την εκπονήσω στο εργαστήριο NETMODE (Network Management & Optimal Design Laboratory) της σχολής. Επίσης ευχαριστώ ιδιαίτερα τον Υποψήφιο Διδάκτωρα Νίκο Κωστόπουλο για την καθοδήγησή του και την εξαιρετική συνεργασία που είχαμε. Τέλος θα ήθελα να ευχαριστήσω τους γονείς μου για την καθοδήγηση και την ηθική συμπαράσταση που μου προσέφεραν όλα αυτά τα χρόνια.

Αθήνα, Νοέμβριος 2020

Γεώργιος Σουλιώτης

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

Η τεχνολογία των botnet αποτελεί εδώ και αρκετά χρόνια τον κύριο άξονα για την ενορχήστρωση και υποστήριξη μιας μεγάλης ποικιλίας κυβερνοεπιθέσεων, όπως DDoS attacks, phishing κλπ. Οι κακόβουλοι διαχειριστές δικτύων botnet (botmasters), χρησιμοποιούν κατά κόρον Domain Generation Algorithms (DGAs), προκειμένου να καταστεί εφικτή η επικοινωνία του C&C server με τα bots του, με τέτοιον τρόπο, ώστε να παρακάμπτονται οι γνωστοί μηχανισμοί ανίχνευσης κακόβουλου λογισμικού. Τυπικά, ένας αλγόριθμος DGA, χρησιμοποιώντας ένα seed, γνωστό μόνο στις συσκευές που συμμετέχουν στο botnet, παράγει περιοδικά μέσα στη μέρα, ένα ψευδοτυχαίο σύνολο από domain names, τα οποία λειτουργούν ως υποψήφια για εκχώρηση στον C&C server. Κατά συνέπεια, οι παραδοσιακές στατικές τεχνικές αντιμετώπισης των botnet καθίστανται αναποτελεσματικές, αφού με τον ρυθμό που ο C&C αλλάζει ονόματα, αδυνατούν να τον εντοπίσουν και να τον αποκόψουν εγκαίρως. Παράδειγμα τέτοιων στατικών τεχνικών αποτελεί το blacklisting του στατικού domain name ενός C&C server, μόλις αυτός εντοπιστεί, καθώς και το reverse engineering του malware μιας μολυσμένης συσκευής, για την κατά προσέγγιση ανακατασκευή του DGA που χρησιμοποιείται και τη μελέτη του τρόπου με τον οποίο παράγονται τα κακόβουλα ονόματα. Ιδανικά όμως, θα πρέπει ο εντοπισμός κακόβουλων ονομάτων να γίνεται σε πραγματικό χρόνο, με τις προβλέψεις να εκτελούνται σε επίπεδο μεμονωμένων ονομάτων, για την αποφυγή της εδραίωσης επικοινωνίας των bots με τον C&C server.

Ο τομέας της μηχανικής μάθησης (Machine Learning) έχει προσελκύσει σε μεγάλο βαθμό το ενδιαφέρον των ερευνητών ασφαλείας στον κυβερνοχώρο, για την αντιμετώπιση του προβλήματος αυτού. Σε προηγούμενες προσεγγίσεις που βασίζονται σε μεθόδους Machine Learning, ένας ανιχνευτής DGA ονομάτων απαιτεί την εξαγωγή στατιστικών χαρακτηριστικών, καθορισμένων από τον άνθρωπο, για να είναι αποτελεσματικός. Αντίθετα, ένας ανιχνευτής βασιζόμενος σε τεχνικές Supervised Deep Learning (υποσύνολο του Machine Learning) εξάγει αυτόματα τα απαραίτητα χαρακτηριστικά, κατά τη διαδικασία της εκπαίδευσης, ενώ βασίζεται αποκλειστικά στη συμβολοσειρά του domain name καθ' αυτού, για να πραγματοποιήσει τις απαραίτητες προβλέψεις, κατηγοριοποιώντας τα ονόματα ανάμεσα σε DGA και έγκυρα (legit). Η λειτουργία αυτή είναι πολύ χρήσιμη, καθώς οι κακόβουλοι δεν γνωρίζουν πλέον τα χαρακτηριστικά που χρησιμοποιούνται για την εκπαίδευση των ανιχνευτών-μοντέλων. Ως αποτέλεσμα δεν μπορούν να τροποποιούν τους DGA για την παραγωγή ονομάτων που δεν εντοπίζονται με βάση τα χαρακτηριστικά αυτά. Τα μοντέλα που εκπαιδεύονται

με τεχνικές Deep Learning παρουσιάζουν υψηλή ακρίβεια και γενικεύουν αποδοτικά, όπως προκύπτει από σχετικές έρευνες που θα παρουσιαστούν εκτενώς σε επόμενο κεφάλαιο.

Πράγματι, όπως θα εξεταστεί αναλυτικά κατά τη διάρκεια αυτής της διπλωματικής, ο εντοπισμός κίνησης botnet με τη συνδρομή του Deep Learning είναι μια εξαιρετικά υποσχόμενη μέθοδος. Στην προσέγγιση αυτή όμως υπεισέρχονται και κάποιοι περιορισμοί τους οποίους πρέπει να ξεπεράσουμε, ώστε να βελτιώσουμε την απόδοση του προτεινόμενου μοντέλου. Ως γνωστό, η ικανότητα ενός μοντέλου Deep Learning να κατηγοριοποιεί σωστά τα πρότυπα που λαμβάνει ως είσοδο, εξαρτάται σε μεγάλο βαθμό από το dataset που χρησιμοποιείται για την εκπαίδευσή του. Στην πλειοψηφία τους, οι έρευνες που έχουν δημοσιευτεί, χρησιμοποιούν ονόματα DGA που είναι διαθέσιμα στο κοινό και έχουν προκύψει από δημοσιευμένα σχετικά projects, καθώς και από επιτυχημένες προσπάθειες reverse engineering για διάφορες οικογένειες DGA. Το πρόβλημα με τα εν λόγω datasets είναι κατά πρώτον ο περιορισμένος αριθμός ονομάτων από τα οποία αποτελούνται και κατά δεύτερον και κυριότερο, το γεγονός πως ένα μεγάλο ποσοστό από αυτά είναι απαρχαιωμένα. Ομολογουμένως, στερούνται ονομάτων, προερχόμενων από πιο πρόσφατους DGAs, με αποτέλεσμα να μειώνεται η διακριτική ικανότητα των εκπαιδευόμενων μοντέλων, σε ότι αφορά νεοεμφανιζόμενους DGAs. Από την άλλη πλευρά, οι οργανισμοί που ειδικεύονται στον τομέα της ασφάλειας στον κυβερνοχώρο και οι πάροχοι ISP εκπαιδεύουν τα δικά τους μοντέλα, χρησιμοποιώντας για την διαδικασία της εκπαίδευσης δεδομένα που δεν είναι σε καμία περίπτωση διατεθειμένοι να μοιραστούν, λόγω ανταγωνισμού και οικονομικών συμφερόντων. Το πρόβλημα γίνεται πιο υπαρκτό αν συνυπολογίσουμε, ότι μία από τις μεγαλύτερες πηγές τόσο DGA ονομάτων, όσο και legit, ένας recursive DNS, υπακούει σε αυστηρά πρωτόκολλα ιδιωτικότητας, καθιστώντας αδύνατη την πρόσβαση σε δεδομένα ζωτικής σημασίας (π.χ. logs με DNS queries) για την βέλτιστη και πιο ενημερωμένη εκπαίδευση ενός μοντέλου Deep Learning. Καταλήγουμε λοιπόν σε ένα σενάριο, στο οποίο ενώ όλοι οι ενδιαφερόμενοι επιθυμούν να ενισχύσουν τα μοντέλα τους, χρησιμοποιώντας τη μεγαλύτερη δυνατή ποικιλία και τα πιο πρόσφατα διαθέσιμα δεδομένα για την εκπαίδευση, κανείς δεν θέλει να συμβάλλει προς αυτή την κατεύθυνση με το να εκθέσει τα δεδομένα του στο κοινό.

Η παραπάνω κατάσταση μοιάζει να οδηγεί σε αδιέξοδο. Και όμως, φαίνεται να υπάρχει τρόπος, ώστε να εκπαιδεύεται ένα μοντέλο Deep Learning, συνδυάζοντας όλα τα διαθέσιμα δεδομένα κατά την εκπαίδευση, χωρίς αυτά να αποκαλύπτονται σε κανέναν και ικανοποιώντας παράλληλα τις απαιτήσεις για ιδιωτικότητα. Ο λόγος γίνεται για τον πολύ καινούριο κλάδο του Federated Learning. Στο παραδοσιακό Distributed Deep Learning, οι ενδιαφερόμενοι αποστέλλουν τα δεδομένα τους σε έναν κεντρικό server, ο οποίος θα εκτελέσει την εκπαίδευση του μοντέλου πάνω σε αυτά και με το πέρας της διαδικασίας θα επιστρέψει στους συμμετέχοντες το μοντέλο που προέκυψε. Είναι σαφές ότι η διαδικασία που περιγράφεται δεν προστατεύει το απόρρητο των δεδομένων, καθώς ο κάθε ενδιαφερόμενος πρέπει να εκθέσει τα δεδομένα του στο διαδίκτυο, προκειμένου να τα στείλει σε έναν κεντρικό server, ο οποίος όμως δεν είναι βέβαιο ότι μπορεί να εξασφαλίσει το απόρρητο. Εν αντιθέσει, το Federated Learning στη βασικότερη αρχιτεκτονική του υλοποιείται από έναν κεντρικό aggregating server, ο οποίος, αρχικοποιεί ένα γενικό μοντέλο και το στέλνει σε όλους τους συμμετέχοντες (clients). Αυτοί με τη σειρά τους εκπαιδεύουν το γενικό μοντέλο που έλαβαν

αποκλειστικά πάνω στα δικά τους δεδομένα. Στο πέρας κάθε γύρου εκπαίδευσης προκύπτει για κάθε client ένα τοπικό μοντέλο, το οποίο θα σταλεί πίσω στον κεντρικό server. Μόλις αυτός συλλέξει όλα τα τοπικά μοντέλα, τα συμψηφίζει (παραδοσιακά με Federated Averaging), και παράγει το νέο γενικό μοντέλο, ξεκινώντας τον νέο γύρο εκπαίδευσης. Ο κύκλος αυτός επαναλαμβάνεται μέχρι να συμπληρωθεί κάποιος προκαθορισμένος αριθμός εποχών, ή να ικανοποιηθεί κάποια συνθήκη σχετικά με την επίδοση του εκπαιδευόμενου μοντέλου. Το κύριο χαρακτηριστικό αυτής της αρχιτεκτονικής είναι, ότι τα μόνα δεδομένα που ανταλλάσσονται είναι οι παράμετροι των μοντέλων καθαυτών, ενώ τα δεδομένα των clients, παραμένουν προστατευμένα στις συσκευές όπου βρίσκονταν αρχικά, χωρίς να υπάρχει καμία ανταλλαγή αυτών. Ο κάθε client, συμβάλλει στη διαδικασία, έχοντας όμως πλήρη άγνοια για τον τύπο και τον όγκο των δεδομένων των υπολοίπων. Το σχήμα αυτό σέβεται πλήρως το απόρρητο των δεδομένων, που στην περίπτωσή μας είναι ένα σύνολο από domain names, (είτε legit, είτε DGA), ενώ ταυτόχρονα παρέχει ένα συνεργατικό περιβάλλον εκπαίδευσης του μοντέλου, με σκοπό το καλύτερο δυνατό αποτέλεσμα, προς κοινό όφελος.

1.1 Αντικείμενο της Διπλωματικής

Η ιδέα μας είναι να αναπτύξουμε έναν δυαδικό ανιχνευτή για έγκυρα και κακόβουλα ονόματα, που θα μπορεί να εκτελεί προβλέψεις σε πραγματικό χρόνο. Στην ιδανικότερη μορφή του θα λειτουργεί σε έναν recursive DNS server, ώστε να μπλοκάρει άμεσα οποιαδήποτε κίνηση botnet εντοπίζεται, στο πιο αρχικό στάδιο, που είναι η εδραίωση επικοινωνίας του C&C server με τα bots του. Εξετάζουμε ένα σενάριο όπου το μοντέλο του ανιχνευτή θα εκπαιδεύεται ανά διαστήματα, με Federated Learning, όπου οι clients θα είναι οι ίδιοι οι recursive DNS servers. Τα domain names, που θα χρησιμοποιούνται για την εκπαίδευση, προέρχονται τόσο από NXDomain αποκρίσεις (πιθανά κακόβουλα ονόματα), όσο και από γνήσια resolved ονόματα. Η επιλογή σχετικά με τις NXDomain αποκρίσεις, βασίζεται στο γεγονός ότι μόνο ένα πολύ μικρό μέρος των αλγοριθμικά παραγόμενων ονομάτων, προορίζεται για καταχώρηση στον C&C server, οπότε η πλειοψηφία τους περιορίζεται σε αρνητικές DNS αποκρίσεις. Μάλιστα, εκμεταλλευόμενοι την προηγούμενη πληροφορία, παρατηρούμε ότι ο ανιχνευτής μας μπορεί να χρησιμοποιηθεί, εκτός από το μπλοκάρισμα της κίνησης botnet και για τον εντοπισμό του C&C server, ελέγχοντας τα ονόματα που ενώ κατηγοριοποιήθηκαν ως κακόβουλα, έγιναν τελικά resolved. Επιτυγχάνεται έτσι η εκπαίδευση του ανιχνευτή, ανά τακτά χρονικά διαστήματα, πάνω σε μια μεγάλη ποικιλία domain names που προέρχονται από πραγματικό traffic, ώστε να παραμένει ενημερωμένος σε ό,τι αφορά νέες οικογένειες DGA.

Συνοπτικά, το αντικείμενο της διπλωματικής αυτής είναι:

1. Η μελέτη τριών βασικών μοντέλων Deep Learning (CNN, LSTM, Bidirectional LSTM), που έχουν χρησιμοποιηθεί με μεγάλη επιτυχία σε προηγούμενες έρευνες, για τον εντοπισμό ονομάτων botnet και η προσπάθεια για περαιτέρω βελτίωση αυτών.

2. Η υλοποίηση ενός συστήματος, με τη βοήθεια του PySyft framework που να εφαρμόζει τις βασικές ιδέες και τη φιλοσοφία του Federated Learning, με σκοπό να εξετάσουμε

κατά πόσο μπορεί να εφαρμοστεί η συγκεκριμένη αρχιτεκτονική για την εξέλιξη των παραδοσιακών συστημάτων εντοπισμού ονομάτων botnet και αν ναι, τι προοπτικές-περιορισμοί υπάρχουν.

Απ' όσο γνωρίζουμε, έως και την περίοδο που ξεκίνησε η συγγραφή της συγκεκριμένης διπλωματικής, αυτή είναι η πρώτη προσπάθεια για μια προσέγγιση του ζητήματος της ανίχνευσης domain name, παραγόμενων από DGA, επιστρατεύοντας τεχνικές Federated Learning.

1.2 Οργάνωση του Τόμου

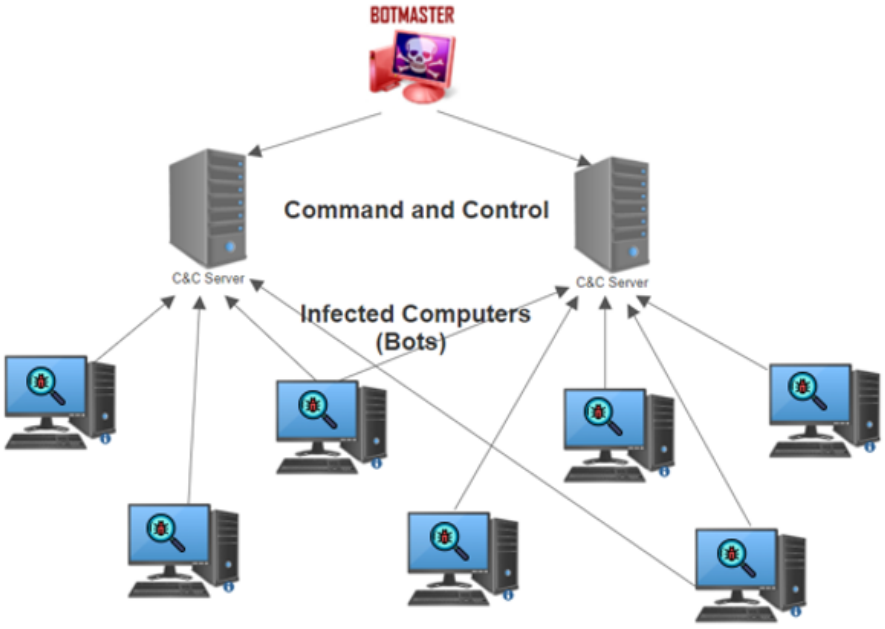
Το υπόλοιπο της διπλωματικής αυτής είναι οργανωμένο σε έξι κεφάλαια:

- Το κεφάλαιο 2 αποτελεί το απαραίτητο θεωρητικό υπόβαθρο αυτής της διπλωματικής. Σε αυτό παρουσιάζονται και εξηγούνται όλες οι σημαντικές έννοιες που αφορούν το αντικείμενο της μελέτης μας, όπως ο τρόπος λειτουργίας ενός DGA-based botnet και η αρχιτεκτονική του Federated Learning.
- Στο κεφάλαιο 3 παρουσιάζεται εκτενώς η σχετική έρευνα που έχει προηγηθεί της εργασίας αυτής και που έχει αποτελέσει θεμέλιο για τη συγγραφή της και για περεταίρω προβληματισμό.
- Στο κεφάλαιο 4 εξηγείται η μεθοδολογία που ακολουθείται για την ανίχνευση DGA domain names. Πιο συγκεκριμένα, παρουσιάζονται τα μοντέλα Deep Learning που αναπτύσσουμε για το σκοπό αυτό και εξηγείται τόσο η αρχιτεκτονική τους, όσο και οι βασικές αρχές λειτουργίας τους. Περιγράφεται επίσης το πειραματικό σύστημα που έχουμε αναπτύξει, προκειμένου να υλοποιήσουμε την αρχιτεκτονική και τις βασικές ιδέες του Federated Learning, για τον εντοπισμό DGA domain names.
- Στο κεφάλαιο 5 περιγράφουμε τα dataset που χρησιμοποιούμε για την εκπαίδευση και την αξιολόγηση των μοντέλων μας, ενώ εξηγείται και ο τρόπος δημιουργίας τους.
- Το κεφάλαιο 6 συνοψίζει το σύνολο των πειραμάτων που εκτελέστηκαν για την ανάλυση της συμπεριφοράς του συστήματός μας, καθώς και τα αποτελέσματα και τις μετρήσεις που προέκυψαν από αυτά. Παράλληλα συγκρίνονται οι επιδόσεις του κλασσικού (Non-Federated) Deep Learning, με τις αντίστοιχες του Federated Learning.
- Το κεφάλαιο 7 αποτελεί τον επίλογο της διπλωματικής με την παράθεση των συμπερασμάτων που προκύπτουν. Σε αυτό αναφέρονται επίσης μελοντικές κατευθύνσεις και ιδέες για έρευνα, ως επέκταση της παρούσας εργασίας.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

Ο Παγκόσμιος Ιστός αποτελείται από εκατομμύρια συσκευές, οι οποίες φιλοξενούν, καθώς και ανταλλάζουν τεράστιο όγκο πληροφοριών. Ωστόσο, ένα μέρος της κίνησης αυτής οφείλεται σε κακόβουλες δραστηριότητες. Χαρακτηριστικό παράδειγμα τέτοιου είδους κίνησης είναι τα μηνύματα botnet.



Εικόνα 2.1: *Botnet Structure*

Ως botnet ορίζεται ένα δίκτυο από μολυσμένες συσκευές (bots) με πρόσβαση στο Internet, στις οποίες έχει εγκατασταθεί κακόβουλο λογισμικό (malware), εν αγνοία των διαχειριστών τους. Σκοπός του malware είναι ο έλεγχος των bots από τον κακόβουλο διαχειριστή του botnet (botmaster) και ο συντονισμός τους μέσω κεντρικών Command and Control (C&C) servers για την εκμετάλλευση της συνδυασμένης υπολογιστικής τους ισχύος και την διεκπαιρέωση μιας σειράς κυβερνοεπιθέσεων [1]. Ένα σύγχρονο botnet μπορεί να αποτελείται από εκατοντάδες χιλιάδες ή και εκατομμύρια bots και για τον συντονισμό τους να χρησιμοποιούνται δεκάδες C&C servers.

Χαρακτηριστικά παραδείγματα κακόβουλης δράσης botnet είναι:

- Υποκλοπή ευαίσθητων προσωπικών δεδομένων, όπως passwords.
- Αποστολή ανεπιθύμητης αλληλογραφίας (spam).
- Επιθέσεις Distributed Denial of Service (DDoS).
- Κρυπτογράφηση των αρχείων ενός υπολογιστή (ransomware) με σκοπό την απαίτηση αντιτίμου σε κάποιο κρυπτονόμισμα για την αποστολή του κλειδιού αποκρυπτογράφησης.
- Χρήση επεξεργαστικής ισχύος για cryptocurrency mining, προς όφελος του επιτιθέμενου.

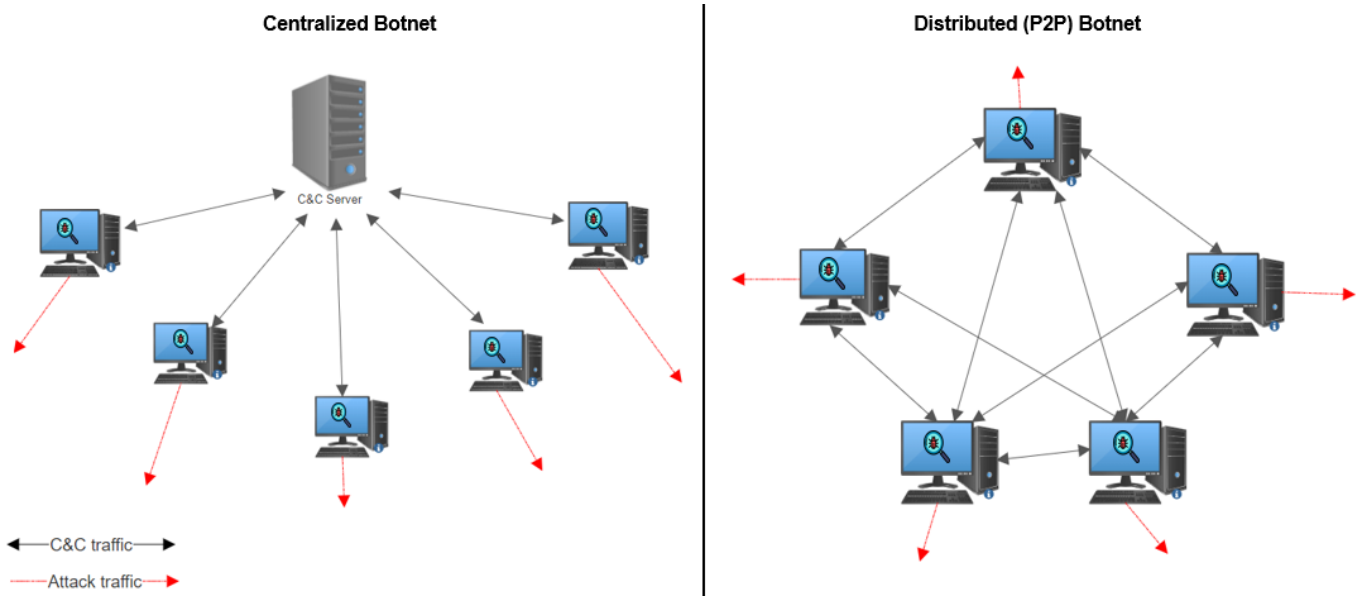
2.1 Πρώιμες Υλοποιήσεις Botnet

Για να καταστεί λειτουργικό ένα botnet, είναι απαραίτητη η επικοινωνία των μολυσμένων συσκευών (bots) με κάποιον κεντρικό Command and Control (C&C) server, ο οποίος λειτουργεί ως συντονιστής, στέλνοντας οδηγίες-εντολές για την ενορχήστρωση και διενέργηση των επιθέσεων. Μάλιστα, ανάλογα με τις ανάγκες του botnet μπορεί να λειτουργούν παραπάνω από ένας C&C servers για λογαριασμό του botmaster. Για τον σκοπό αυτό, οι παλαιότερες οικογένειες malware, εκχωρούσαν στον C&C server ένα static domain name, το οποίο γνωστοποιούταν με hardcoding σε επίπεδο byte στη μολυσμένη συσκευή κατά την εγκατάσταση του malware. Εντούτοις, οι ερευνητές κυβερνοασφάλειας, αντιμετώπιζαν την απειλή αυτή, απλά προσθέτοντας το εκάστοτε domain name σε μία blacklist, μόλις ανακάλυπταν μια καινούρια οικογένεια malware, αποκόβοντας έτσι τον C&C από τα bots του.

Άλλοι μηχανισμοί επικοινωνίας botnet βασίστηκαν σε πρωτόκολλα, όπως το Internet Relay Chat (IRC) και το Hypertext Transfer Protocol (HTTP) [2]. Το IRC είναι ένα πρωτόκολλο για ανταλλαγή μηνυμάτων κειμένου σε πραγματικό χρόνο μεταξύ υπολογιστών που είναι συνδεδεμένοι στο Διαδίκτυο και δημιουργήθηκε το 1988. Χρησιμοποιείται κυρίως για ομαδική συζήτηση σε αίθουσες συνομιλίας που ονομάζονται κανάλια, ενώ μπορεί να υποστηρίξει ιδιωτικά μηνύματα μεταξύ δύο χρηστών, μεταφορά δεδομένων, και διάφορες εντολές server-side και client-side. Το IRC ήταν μια δημοφιλής μέθοδος που χρησιμοποιήθηκε πρωταρχικά από τους botmasters για την αποστολή εντολών σε μεμονωμένους υπολογιστές στο botnet τους. Η ανταλλαγή μηνυμάτων γινόταν είτε σε ένα συγκεκριμένο κανάλι, σε ένα δημόσιο δίκτυο IRC, ή σε έναν ξεχωριστό διακομιστή IRC, που υποστηρίζει τα απαραίτητα κανάλια και λειτουργεί ως C&C server. Τα bots συνδέονταν στο αντίστοιχο κανάλι και παρέμεναν συνδεδεμένα, περιμένοντας νέες εντολές από τον botmaster. Από την άλλη πλευρά, σε μετέπειτα υλοποιήσεις που βασίστηκαν στο πρωτόκολλο HTTP, αντί τα bots να παραμένουν σε σύνδεση, επισκέπτονταν περιοδικά συγκεκριμένους Web servers, για να λαμβάνουν ενημερώσεις και νέες εντολές. Οι botmasters χρησιμοποιούν το πρωτόκολλο HTTP για να αποκρύψουν τις δραστηριότητές τους μεταξύ της φυσιολογικής κίνησης του διαδικτύου και να αποφεύγουν εύκολα τις τρέχουσες μεθόδους ανίχνευσης, όπως είναι τα τείχη προστασίας (firewalls). Λόγω του ευρέος φάσματος των υπηρεσιών HTTP που χρησιμοποιούνται, σε αντίθεση με το IRC, είναι πιο δύσκολο να εντοπιστεί και να αποκλειστεί η κίνηση botnet αυτής

της μορφής. Επιπλέον, αυτή η υπηρεσία χρησιμοποιείται συνήθως από νόμιμες εφαρμογές και υπηρεσίες στο Διαδίκτυο, όπως η περίοδος σύνδεσης του Gmail (η οποία περιοδικά ελέγχει για νέα μηνύματα ηλεκτρονικού ταχυδρομείου). Οπότε οι κακόβουλοι μπορούν να δημιουργήσουν το ίδιο περιοδικό μοτίβο αποστολής μηνυμάτων από τα bots, ώστε η κίνηση να παραπέμπει σε γνήσια. Για μεγάλο χρονικό διάστημα η μέθοδος αυτή αποτέλεσε μια σημαντική πρόκληση για τον τομέα της ασφάλειας. Παρόλα αυτά, μετά από στοχευμένες μελέτες [3], [4], αναπτύχθηκαν μηχανισμοί για τον αποτελεσματικό εντοπισμό και αποκλεισμό της κακόβουλης κίνησης botnet που χρησιμοποιούσε τα πρωτόκολλα που περιγράφηκαν.

Το γεγονός αυτό ανάγκασε τους κακόβουλους να προβούν στη θέσπιση νέων τεχνικών και πρωτοκόλλων, για την επικοινωνία μεταξύ του C&C και των bots του. Πολλές προσπάθειες επικεντρώθηκαν στην υλοποίηση αποκεντρωμένων botnet, με βάση την αρχιτεκτονική Peer to Peer (P2P), ώστε να μην υπάρχει η εξάρτηση από έναν κεντρικό server και οι εντολές να αποστέλονται από τον botmaster στα bots του με κατανεμημένο τρόπο. Τέτοιες υλοποιήσεις, όπως το Waledac [5], το Nugache και το Storm [6], καθιστούν την εξάρθρωση ενός botnet εξαιρετικά δύσκολη, αφού πλέον δεν αρκεί ο εντοπισμός και αποκλεισμός του κεντρικού C&C server, αλλά πολλών bot (peers) ταυτόχρονα. Ευτυχώς όμως, υλοποιήσεις botnet σαν και αυτές είναι αρκετά πολύπλοκες, οπότε δεν συναντώνται συχνά. Δεν αποκλείεται όμως, μελλοντικά, να αποτελέσουν πιο εκτεταμένη απειλή. Τελικά, η πιο μοντέρνα και απλή τεχνική που έχουν υιοθετήσει οι κάτοχοι botnet, είναι οι λεγόμενοι Domain Generation Algorithms (DGAs).



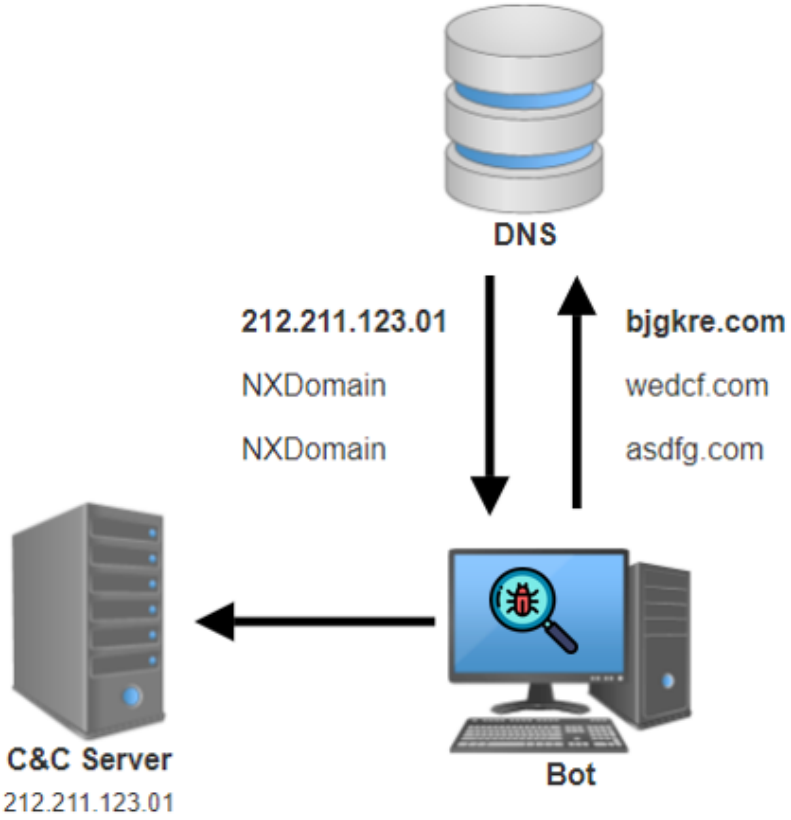
Εικόνα 2.2: *Centralized vs P2P Botnets*

2.2 Η Συμβολή των DGA στην Υλοποίηση Μη-Ανιχνεύσιμων Botnet

Ως αλγόριθμος, ένας DGA χρησιμοποιείται, σύμφωνα με τους Plohmann et al. [7], για την περιοδική παραγωγή ενός μεγάλου συνόλου ονομάτων (domain names), σε ένα δυναμικό περιβάλλον, που σε αντίθεση με τις παλιές στατικές προσεγγίσεις, δίνουν τη δυνατότητα στους

botmasters να εναλλάσσουν συχνά την ταυτότητα του C&C server. Η περιοδική αλλαγή του domain name, με σκοπό την αποφυγή της ανίχνευσης, αναφέρεται συχνά στη βιβλιογραφία με τον όρο domain-fluxing. Πιο συγκεκριμένα, όλες οι συσκευές που συμμετέχουν στο botnet, εκτελούν ταυτόχρονα μετά απο προκαθορισμένο χρονικό διάστημα τον σχετικό αλγόριθμο. Αρχικά, λαμβάνουν ως είσοδο ένα seed, γνωστό μόνο στους συμμετέχοντες στο botnet, που συνήθως είναι η ισχύουσα ημερομηνία, ένας ακέραιος αριθμός, ένα hash code, ή ακόμα και δύο συλλογές αγγλικών λέξεων, αναλόγως την οικογένεια στην οποία ανήκει ο DGA. Η επιλογή του seed είναι ζωτικής σημασίας για τον τρόπο με τον οποίο παράγονται τα ονόματα σε κάθε κύκλο εκτέλεσης ενός DGA. Υπάρχουν τέσσερα βασικά σχήματα παραγωγής:

1. Υπολογισμός μιας ακολουθίας αλφαριθμητικών (π.χ. DirCrypt).
2. Υπολογισμός της δεκαεξαδικής αναπαράστασης ενός hash code, συνήθως με αλγορίθμους κρυπτογράφησης MD5 και SHA256 (π.χ. MoneroDownloader, Bamital).
3. Μετάθεση ή και μετάλλαξη των αλφαριθμητικών ενός αρχικού domain name (π.χ. Volatile Cedar).
4. Συνένωση δύο ή παραπάνω αγγλικών λέξεων, προερχόμενες από διαφορετικές λίστες λέξεων (π.χ. Matsnu, Suppobox).



Εικόνα 2.3: *Domain-Fluxing*

Επίσης, σύμφωνα με τους Barabosch et al. [8], οι DGAs διακρίνονται και σε επιπλέον κατηγορίες, με βάση δύο πολύ σημαντικές ιδιότητες των seeds τους, την εξάρτηση από τον χρόνο

(Time-Dependent/Time-Indepedent) και την αιτιότητα (Deterministic/Non-Deterministic), όπως εξηγείται στο κεφάλαιο 3. Πάντως, χωρίς βλάβη της γενικότητας, από τα αποτελέσματα της μελέτης των Plohmann et al. (2016) [7], προκύπτει ότι σπάνια παρατηρούνται Non-Deterministic DGAs (μόνο ένας από τους 43 που εξετάστηκαν), ενώ οι περισσότεροι χρησιμοποιούν αλφαριθμητικά seeds για τον υπολογισμό των ονομάτων (36 από τους 43).

Με έναν από τους παραπάνω τρόπους παράγεται το επιθυμητό σύνολο από domain names, υποψηφίων για εκχώρηση στον C&C server. Στη συνέχεια, ένα υποσύνολο των παραγόμενων domain names εκχωρούνται στους C&C servers (συνήθως με αυτόματο και ανώνυμο registration) και τα bots ξεκινούν να εκτελούν DNS queries για κάθε ένα από τα domain names του παραγόμενου συνόλου, μέχρις ότου κάποιο από αυτά να επιστρέψει θετικό response (Εικόνα 2.3). Φυσικά το όνομα για το οποίο συμβαίνει αυτό θα είναι αυτό που εκχωρήθηκε στον C&C server, οπότε η επικοινωνία αποκαθίσταται. Σε επόμενο στάδιο, τα bots λαμβάνουν εντολές από τον υπεύθυνο C&C και ανταλλάζουν δεδομένα με αυτόν.

2.3 Η Εξάρτηση από το Πρωτόκολλο DNS

Γενικά, παρατηρούμε ότι οι υλοποιήσεις botnet που βασίζονται στη φιλοσοφία των DGA, χρησιμοποιούν κατά κύριο λόγο το πρωτόκολλο DNS (Domain Name System) για την εδραίωση ενός επικοινωνιακού διαύλου μεταξύ C&C server και των bots. Ο ρόλος του DNS είναι καθοριστικός για τη λειτουργικότητα του Internet, αφού λειτουργεί ως μια κατανεμημένη υπηρεσία καταλόγου, με σκοπό τη μετάφραση των domain names στις αντίστοιχες IP διευθύνσεις τους. Λόγω της καθολικής αποδοχής και χρήσης της υπηρεσίας (πασίγνωστη θύρα 53), η κίνηση DNS θεωρείται γενικά νόμιμη και δεν φιλτράρεται στα firewalls. Έτσι οι κακόβουλοι, εκμεταλλευόμενοι την ιδιότητα των μηνυμάτων DNS, να παρακάμπτουν τους μηχανισμούς ασφαλείας, χρησιμοποιούν τη σχετική υπηρεσία για την εκτέλεση του domain-fluxing στις DGA-based υλοποιήσεις botnet. Πράγματι, όπως περιγράφηκε προηγουμένως, αφού παραχθεί το σύνολο των DGA ονομάτων σε κάθε κύκλο εκτέλεσης, ακολουθεί σειριακή αποστολή DNS ερωτημάτων από τα bots, μέχρι κάποιο domain name να γίνει resolved και να εντοπιστεί ο C&C server. Μάλιστα, πολλοί botmasters χρησιμοποιούν περαιτέρω το πρωτόκολλο DNS, ακόμα και μετά την αποκατάσταση της επικοινωνίας, ώστε να πραγματοποιείται ανεμπόδιστα η λήψη εντολών και η ανταλλαγή δεδομένων στο botnet. Η τεχνική αυτή ονομάζεται DNS tunneling και αφορά στη δυνατότητα κωδικοποίησης δεδομένων άλλων προγραμμάτων ή πρωτοκόλλων σε ερωτήματα και αποκρίσεις DNS [9].

2.3.1 DNS-tunneling

Σε τέτοιου είδους αρχιτεκτονικές, ως C&C server χρησιμοποιείται ένας παραβιασμένος DNS server, που ελέγχεται από τον botmaster. Από την πλευρά των bots, σε αυτά αρκεί να έχει εγκατασταθεί το κατάλληλο κακόβουλο λογισμικό, ώστε να παράγουν κωδικοποιημένα ερωτήματα DNS, τα οποία προωθούνται στον recursive DNS του τοπικού δικτύου (συνήθως το router), ο οποίος εν αγνοία του λειτουργεί ως proxy για λογαριασμό του malware. Αυτός με τη σειρά του στέλνει το ερώτημα στον C&C που έχει οριστεί ως default DNS resolver, όπου και αποκωδικοποιείται. Στο [10] περιγράφεται λεπτομερώς η δομή και η λειτουργία του

botnet Feederbot, το οποίο χρησιμοποιεί DNS tunneling. Αναλυτικά η συνήθης διαδικασία που ακολουθείται περιγράφεται στα εξής βήματα:

1. Η μολυσμένη συσκευή στέλνει ένα ερώτημα στον C&C, το οποίο περιέχει ένα ενσωματωμένο (ίσως κρυπτογραφημένο) μήνυμα στο αντίστοιχο payload. Για τον λόγο αυτό προτιμώνται DNS ερωτήματα για TXT εγγραφές, αφού ο συγκεκριμένος τύπος παρέχει το μεγαλύτερο διαθέσιμο payload για ενσωμάττωση του κακόβουλου μηνύματος (π.χ αίτημα για λήψη καινούριας εντολής). Παράλληλα, στο domain name για το οποίο εκτελείται το ερώτημα, κωδικοποιούνται ορισμένοι παράμετροι, όπως το κλειδί αποκρυπτογράφησης του μηνύματος.

2. Ο τοπικός recursive DNS προωθεί το ερώτημα στον default DNS resolver, ο οποίος λειτουργεί ως C&C server.

3. Ακολουθεί η αποκωδικοποίηση και μετά η αποστολή στη μολυσμένη συσκευή ενός DNS response, ίδιας μορφής με το query, που περιέχει μία καινούρια εντολή προς εκτέλεση.

Οι botmasters πρέπει να είναι προσεκτικοί σχετικά με τα domain names για τα οποία εκτελούνται τα ερωτήματα, ώστε κάθε φορά να είναι διαφορετικά μεταξύ τους. Αλλιώς ο recursive DNS δεν θα προωθήσει το μήνυμα στον C&C, αφού θα έχει καταχωρήσει στην cache του την πιο πρόσφατη απάντηση. Πάντως στη διπλωματική αυτή η μελέτη περιορίζεται στην ανίχνευση και τον αποκλεισμό κίνησης DGA-based botnet, που προκύπτει κατά το domain-fluxing, ώστε ιδανικά, να περιορίσουμε το botnet στα πρώτα στάδια ζωής του.

2.4 Συστήματα Εντοπισμού DGA-based botnet

Στις παλιές στατικές μεθόδους των επιτιθέμενων, οι ερευνητές είχαν τον χρόνο με το μέρος τους, αφού μπορούσαν να παρατηρήσουν τυχόν ανωμαλίες, ή ύποπτη κίνηση στο διαδίκτυο που σχετιζόταν με το domain name του C&C server και να τον περιορίσουν, εξουδετερώνοντας το botnet. Αντίθετα, η φιλοσοφία των DGA, βασίζεται στα μικρής διάρκειας ζωής domain names, που εκχωρούνται ανά λίγες ώρες στους C&C servers. Κατά συνέπεια, ακόμα και αν εντοπιζόταν τελικά ένα domain name που δημιουργήθηκε κατά τα προηγούμενα, η πληροφορία αυτή θα ήταν άχρηστη, καθώς το σχετικό όνομα θα είχε ήδη αντικατασταθεί με ένα καινούριο. Κατ' αυτόν τον τρόπο το malware, μπορούσε να παρακάμπτει τις παραδοσιακές τεχνικές εντοπισμού, όπως firewalls και reputation systems. Δημιουργείται έτσι μία ασύμμετρη κατάσταση μεταξύ κακόβουλων και ερευνητών ασφαλείας, στην οποία οι μεν χρειάζεται απλά να εντοπίσουν και να αποκτήσουν πρόσβαση μόνο στο domain name που καταχωρήθηκε στον C&C, ενώ οι δε πρέπει να ανιχνέυσουν ένα σύνολο από παραγόμενα ονόματα, ώστε να διασφαλίσουν τον αποκλεισμό της κακόβουλης κίνησης και κατ' επέκταση τον εντοπισμό του κεντρικού server.

Η χρήση των DGAs από τους κακόβουλους, δημιούργησε αρχικά μεγάλη σύγχηση στον τομέα της κυβερνοασφάλειας, αφού πλέον η πολυπλοκότητα του προβλήματος εντοπισμού botnet, καθώς και η απαίτηση υπολογιστικών πόρων, αυξήθηκε σημαντικά. Πάραυτα, μετά

από εκτεταμμένη έρευνα, έχει αναπτυχθεί μια σειρά από στρατηγικές για τον σκοπό αυτό, οι οποίες εξετάζονται ενδελεχώς στο κεφάλαιο 3. Πολλές από αυτές εστιάζουν στην ανάλυση της δομής και της λειτουργίας των DGAs με μεθόδους reverse-engineering, για να εντοπίσουν ύποπτες κινήσεις στο traffic και να περιορίσουν τυχόν κακόβουλους. Ωστόσο, η τεχνική αυτή έχει αρκετά περιορισμένη απόδοση, ενώ παράλληλα δεν έχει προοπτικές κλιμακωσιμότητας scalability. Επιπροσθέτως, οι υψηλές χρονικές απαιτήσεις, την καθιστούν μη αποδοτική για εφαρμογή σε συστήματα ανίχνευσης πραγματικού χρόνου. Ένα επίσης σοβαρό μειονέκτημα εις βάρος του reverse-engineering είναι πως για να διενεργηθεί η σχετική ανάλυση, θεωρείται δεδομένη η ύπαρξη δειγμάτων domain names που έχουν παραχθεί από κάποιον γνωστό DGA, η οποία δεν είναι πάντα εφικτή, ειδικά για νεοεμφανιζόμενους DGAs. Μια πιο πρακτική προσέγγιση είναι η ανάλυση της κίνησης DNS για να διαπιστωθεί εάν τα domain names έχουν δημιουργηθεί αλγοριθμικά. Γενικά, η μέθοδος των DGA διατηρεί ένα ευάλωτο σημείο εκ κατασκευής. Κατά τη διάρκεια της αναζήτησης του ονόματος που εκχωρήθηκε στον C&C server, τα bots καταχράζονται το πρωτόκολλο DNS, στέλνοντας ερωτήματα για το κάθε domain name, μέχρι να λάβουν θετική απόκριση. Συνεπώς, για κάθε τέτοιο ερώτημα που δεν επιλύεται, δηλαδή για κάθε domain name που παράχθηκε από τον DGA, αλλά δεν εκχωρήθηκε στον C&C σερερ, παράγεται ένα NXDomain (Non Existent Domain) response. Προκαλείται έτσι μια παρατηρίσιμη ανωμαλία στο δίκτυο, που αφορά αυξημένο αριθμό NXDomain αποκρίσεων. Φυσικά μια NXDomain απόκριση μπορεί να προκύψει και για άλλους λόγους, όπως για παράδειγμα τυπογραφικά λάθη των χρηστών. Ωστόσο, τέτοιοι λόγοι δεν δικαιολογούν τις συνήθεις απότομες μεταβολές στον όγκο των NXDomain responses, που προκαλούνται κατά τον κύκλο εκτέλεσης ενός DGA. Επομένως, τα εν λόγω ίχνη που αφήνει ένα DGA-based botnet στο διαδίκτυο, χρησιμοποιούνται σε μελέτες για την ανάλυσή τους και τον διαχωρισμό τους σε κλάσεις, με βάση τα πιθανά κοινά χαρακτηριστικά τους (π.χ. λεξιλογικά), με σκοπό την ανίχνευση μέσα από αυτή τη διαδικασία νέων οικογενειών DGA. Μετά τον εντοπισμό μιας νέας οικογένειας DGA, μελετώνται όλα τα domain names που δημιουργήθηκαν από τον σχετικό DGA. Μόλις βρεθεί το domain name του C&C server, ο διαχειριστής δικτύου μπορεί να τον αποσυνδέσει από τα bots του, αχρηστεύοντας έτσι το botnet.

Μία από τις πιο σύγχρονες και αποτελεσματικές μεθοδολογίες για τον εντοπισμό αλγοριθμικά παραγόμενων domain names, η οποία εξετάζεται και στην παρούσα διπλωματική, επικεντρώνεται στο Deep Learning και βασίζεται στην υπόθεση ότι οι botmasters, πρέπει να λαμβάνουν υπόψη τους κάποιους περιορισμούς κατά τη δημιουργία των domain names. Αφενός πρέπει να αποφεύγεται η χρήση πραγματικών λέξεων στα ονόματα των server τους (εξαίρεση αποτελούν οι wordlist-based DGAs), διότι είναι πιθανό να έχουν ήδη καταχωρηθεί ως πραγματικά domain names. Αφετέρου, τα παραγόμενα ονόματα πρέπει να είναι δύσκολα ανιχνεύσιμα από τον καθένα και ειδικότερα από τους επαγγελματίες της κυβερνοασφάλειας. Ως εκ τούτου, οι botmasters καταφεύγουν στη δημιουργία ψευδοτυχαίων domain names, που διαφέρουν σημαντικά από τα αντίστοιχα που δημιουργούνται από τους διαχειριστές δικτύων και με βάση αυτή τη διαφορά γίνεται η κατηγοριοποίηση ονομάτων σε γνήσια και αλγοριθμικά παραγόμενα. Οι πιο πρόσφατες μελέτες που χρησιμοποιούν Deep Learning προς αυτήν την κατεύθυνση, χρησιμοποιούν datasets που αποτελούνται τόσο από

DGA ονόματα, όσο και από γνήσια, για την εκπαίδευση ενός μοντέλου που θα εντοπίζει την κακόβουλη κίνηση και θα κάνει δυαδικές προβλέψεις σε επίπεδο μεμονωμένων domain names, ανάμεσα σε legit και malicious. Κατά κοινή ομολογία, η απόδοση ενός μοντέλου Deep Learning εξαρτάται σε μεγάλο βαθμό από την ποιότητα και τη δομή του dataset που επιλέγεται. Για την κατασκευή καταλλήλων dataset εκπαίδευσης, πολλές τεχνικές επικεντρώνονται στην ανάλυση εγγραφών passive DNS και στην εξαγωγή χαρακτηριστικών από αυτές.

2.4.1 Τεχνικές Passive DNS

Η ιδέα του passive DNS εφευρέθηκε από τον Florian Weimer [11] το 2004, για την καταπολέμηση κακόβουλου λογισμικού. Βασικά, όταν ένας recursive DNS λαμβάνει ένα ερώτημα σχετικά με κάποιο domain name εξετάζει πρώτα την προσωρινή μνήμη cache και τα έγκυρα δεδομένα του για μια απάντηση, και εάν αυτή δεν υπάρχει, ξεκινάει να ρωτά τους root name servers έως ότου εντοπίσει τον authoritative DNS server. Αυτός αποκρίνεται με τη σχετική απάντηση και ο recursive DNS μεσολαβεί για να τη στείλει, αφού πρώτα ενημερώσει την cache του. Η απόκριση αυτή καταγράφεται και αποθηκεύεται αμέσως από τον passive DNS, ο οποίος λειτουργεί πάνω από τον recursive. Αυτό σημαίνει ότι τα παθητικά δεδομένα DNS συνίστανται κυρίως από έγκυρες αποκρίσεις authoritative DNS server (μαζί με αρνητικές αποκρίσεις φυσικά, όπως NXDomain). Αυτά τα δεδομένα σφραγίζονται χρονικά, συμπιέζονται, και στη συνέχεια αναπαράγονται σε μια κεντρική βάση δεδομένων για αρχειοθέτηση και ανάλυση. Έτσι το ιστορικό των συλλεγόμενων DNS δεδομένων, δίνει τη δυνατότητα στους ερευνητές ασφαλείας, να κρατάνε αρχείο με τα κακόβουλα DGA ονόματα και να τα μελετούν, ακόμα και αν οι σχετικές εγγραφές έχουν αντικατασταθεί με καινούριες στην προσωρινή μνήμη των recursive DNS.

Με αυτόν τον τρόπο, οι οργανισμοί που ασχολούνται με την ασφάλεια του Internet, βρίσκουν ονόματα DGA, που μπορούν να χρησιμοποιήσουν στα dataset που κατασκευάζουν για την εκπαίδευση των Deep Learning μοντέλων τους. Στη συνέχεια εγκαθιστούν τα μοντέλα τους ως ανιχνευτές σε έναν DNS server για να εντοπίσουν σε πραγματικό χρόνο την κακόβουλη κίνηση που προκαλείται από το domain-fluxing. Συνίσταται η εγκατάσταση του ανιχνευτή να γίνεται σε έναν recursive DNS, ώστε η κακόβουλη κίνηση να αποκόπτεται πριν προωθηθεί στην ανώτερη ιεραρχία DNS. Τελικά, με βάση τις προβλέψεις του ανιχνευτή, σχετικά με τα κακόβουλα ονόματα, κάθε εταιρία που δραστηριοποιείται στον τομέα της ασφάλειας εμπλουτίζει το DGA αρχείο της και χρησιμοποιεί τις νέες εγγραφές για την ενημέρωση πιθανών blacklist και των dataset που χρησιμοποιεί για την εκπαίδευση των μοντέλων της. Δυστυχώς όμως, ο εντοπισμός ενός C&C server και η συνολική εξάρθρωση ενός DGA-based botnet, συχνά απαιτεί πολύ περισσότερους πόρους και πληροφορίες από αυτές που μπορεί να διαθέτει ένας μεμονωμένος οργανισμός. Χαρακτηριστικό παράδειγμα αποτελεί το διάσημο Conficker botnet, το οποίο δημιουργούσε 50.000 ονόματα καθημερινά και χρησιμοποιούσε 113 TLDs. Για τον περιορισμό της απειλής του, χρειάστηκε η συνεργασία 30 διαφορετικών οργανισμών , συμπεριλαμβανομένου του ICANN [12]. Παραταύτα, οι εταιρίες και οι οργανισμοί ασφαλείας δεν είναι πάντα διατεθειμένοι να συνεργαστούν μεταξύ τους, ανταλλάσσοντας τα δεδομένα τους για την πιο αποτελεσματική εκπαίδευση μοντέλων-ανιχνευτών, είτε εξαιτίας αυστηρών

πρωτοκόλλων ιδιωτικότητας, είτε λόγω εμπορικού ανταγωνισμού και συμφερόντων. Η στάση αυτή επιφέρει αρνητικές επιπτώσεις στις προοπτικές ανάπτυξης βελτιωμένων μοντέλων ανίχνευσης αλγοριθμικά παραγόμενων domain names. Για να προσπεραστεί το εμπόδιο αυτό, προτείνεται στην παρούσα διπλωματική η επιστράτευση της αρχιτεκτονικής του Federated Learning.

2.5 Federated Learning: Συνεργατική Καταπολέμηση των DGA-Based Botnet

Οι αλγόριθμοι που βασίζονται σε μεθόδους βαθιάς μηχανικής μάθησης (Deep Learning) χρειάζονται μεγάλο όγκο δεδομένων για να επιτύχουν την επιθυμητή απόδοση. Συνήθως, τα απαιτούμενα σύνολα δεδομένων (dataset) είναι μεγάλης κλίμακας και είναι κατανεμημένα σε πολλές πηγές, γεγονός που καθιστά την πρόσβαση σε αυτά από έναν κεντρικό server επίπονη και κοστοβόρα. Στην εποχή λοιπόν των Big Data, υπάρχει η πρόκληση για υλοποίηση ενός κατανεμημένου συστήματος μηχανικής μάθησης. Επίσης, η ευαισθητοποίηση του κοινού σε θέματα ιδιωτικότητας των προσωπικών τους δεδομένων, οδήγησε τους νομοθέτες στη θέσπιση νέων κανονισμών, όπως ο GDPR για την προστασία του απορρήτου. Πρόσφατα (2018), οι επιστήμονες της Google πρότειναν ένα νέο σχήμα εκπαίδευσης που φαίνεται να ανταποκρίνεται στο πρόβλημα αυτό. Στη μελέτη τους [13] εφαρμόζουν τεχνικές Federated Learning για την βελτίωση της εμπειρίας που παρέχει το Google Gboard, που σχετίζεται με την πρόβλεψη συμπλήρωσης λέξεων κατά την πληκτρολόγηση. Κάθε συσκευή εκπαιδεύει τοπικά ένα μοντέλο μηχανικής μάθησης και στέλνει τις ενημερώσεις σε έναν κεντρικό server, ο οποίος είναι υπεύθυνος για τον συμψηφισμό των τοπικών ενημερώσεων, δημιουργόντας ένα βελτιωμένο γενικό μοντέλο και στέλνοντάς το πίσω σε κάθε συσκευή.

Το Federated Learning (FL) είναι μια καινούρια αρχιτεκτονική μηχανικής μάθησης, όπου πολλοί πελάτες clients (π.χ. κινητές συσκευές, ή ολόκληροι οργανισμοί και εταιρίες στην περίπτωσή μας) εκπαιδεύουν συνεργατικά ένα μοντέλο υπό την ενορχήστρωση ενός κεντρικού server, διατηρώντας τα δεδομένα εκπαίδευσης αποκεντρωμένα και προστατευμένα. Σκοπός είναι η διασφάλιση της ιδιωτικότητας των δεδομένων που χρησιμοποιούνται για την εκπαίδευση, καθώς και η ελαχιστοποίηση του κόστους επικοινωνίας που προκύπτει από τη μεταφορά τεραστίου όγκου δεδομένων στις παραδοσιακές μεθόδους μηχανικής μάθησης. Ο συγκεκριμένος όρος χρησιμοποιήθηκε για πρώτη φορά το 2016 από τους McMahan et al. [14]. Οι Kairouz et al. [15] έδωσαν τον εξής ορισμό για το Federated Learning:

*Το **Federated Learning** είναι ένα περιβάλλον μηχανικής μάθησης, όπου συνεργάζονται πολλοί πελάτες (clients) για την επίλυση ενός προβλήματος μηχανικής μάθησης, υπό τον συντονισμό ενός κεντρικού server ή υπηρεσίας. Τα πρωτογενή δεδομένα κάθε πελάτη αποθηκεύονται τοπικά και δεν ανταλλάσσονται ή μεταφέρονται. Αυτί αυτού, οι εστιασμένες τοπικές ενημερώσεις των μοντέλων που προορίζονται για άμεσο συμψηφισμό σε ένα ενημερωμένο γενικό μοντέλο, χρησιμοποιούνται για την επίτευξη του μαθησιακού στόχου.*

2.5.1 Ο Κύκλος Ζωής του Federated Learning

Τυπικά, τα βήματα της εκπαίδευσης σε ένα απλοποιημένο περιβάλλον Federated Learning είναι τα παρακάτω:

- **Βήμα 1:** Ο κεντρικός server αρχικοποποιεί ένα γενικό μοντέλο.
- **Βήμα 2:** Ο κεντρικός server στέλνει το γενικό μοντέλο σε κάθε συσκευή που συμμετέχει στη διαδικασία.
- **Βήμα 3:** Κάθε συσκευή εκπαιδεύει τοπικά το μοντέλο που έλαβε, χρησιμοποιώντας μόνο τα δικά της δεδομένα.
- **Βήμα 4:** Με το πέρας της εκπαίδευσης, κάθε συσκευή στέλνει πίσω στον κεντρικό server το τοπικό μοντέλο που προέκυψε.
- **Βήμα 5:** Μόλις ο κεντρικός server συλλέξει όλα τα τοπικά μοντέλα, ξεκινάει η διαδικασία συμψηφισμού τους (παραδοσιακά με Federated Averaging) και προκύπτει ένα νέο, ενημερωμένο, γενικό μοντέλο.
- **Βήμα 6:** Ο κεντρικός server στέλνει εκ νέου το συμψηφισμένο μοντέλο σε κάθε συσκευή, ώστε να εκπαιδευτεί εκεί τοπικά.

Μετά τον πρώτο γύρο της εκπαίδευσης σε περιβάλλον Federated Learning, επαναλαμβάνονται τα βήματα από 2 έως 6, μέχρις ότου να συμπληρωθεί ένας προκαθορισμένος αριθμός γύρων, ή να επιτευχθεί κάποιο κριτήριο απόδοσης (π.χ. ακρίβεια).

Όπως αναφέρθηκε, τα παραπάνω βήματα περιγράφουν τον κύκλο ζωής ενός απλοποιημένου σεναρίου Federated Learning. Σε μία πραγματική εφαρμογή, αναλόγως και τη φύση αυτής, είναι πιθανό ο αριθμός των clients να γίνει τόσο μεγάλος (εκατομύρια συσκευές), σε σημείο που να τίθενται ζητήματα επικοινωνίας και κλιμακωσιμότητας (scalability). Για τον λόγο αυτό, όπως περιγράφεται και στη μελέτη των Kairouz et al. [15], κατά το βήμα 2 επιλέγεται ένα υποσύνολο των clients που θα συμμετέχουν στην εκπαίδευση ανά γύρο. Η επιλογή αυτή μπορεί να γίνεται είτε τυχαία, είτε με βάση κάποια κριτήρια, όπως είναι η υπολογιστική ισχύς και η ταχύτητα της κάθε συσκευής, ή η φήμη της σε σχέση με τη συνεισφορά του τοπικού μοντέλου της στο γενικό, σε κάθε γύρο. Επίσης, σε ένα πραγματικό δίκτυο, είναι πιθανό, λόγω σφαλμάτων, ή προσωρινού αυξημένου φόρτου σε έναν client, να προκύψουν σημαντικές καθυστερήσεις κατά την εκπαίδευση τοπικά. Οι clients αυτοί χαρακτηρίζονται από τον κεντρικό server ως strugglers και μετά από τη λήξη ενός χρονικού περιθωρίου, εκτελείται ο συμψηφισμός, χωρίς τα τοπικά μοντέλα τους. Η παρατήρηση αυτή αφορά το βήμα 5. Προκειμένου να μειωθεί περαιτέρω το κόστος επικοινωνίας κατά την αποστολή των τοπικών μοντέλων, οι Konečný et al. (2016) [16] παρουσίασαν δύο διαφορετικές στρατηγικές:

- **Structured updates**: η ενημέρωση γίνεται μόνο για έναν περιορισμένο αριθμό παραμέτρων που αντιπροσωπεύουν τα βάρη του μοντέλου, τα οποία συμβάλλουν περισσότερο στη διακριτική του ικανότητα.

- **Sketched updates**: η ενημέρωση γίνεται πάνω σε όλες τις παραμέτρους του μοντέλου, οι οποίες στη συνέχεια συμπιέζονται σε ένα αρχείο και αποστέλλονται στον κεντρικό server.

Ωστόσο, οι Sattler et al. (2020) [17] υποστηρίζουν στην εργασία τους, ότι οι προσεγγίσεις αυτές, ενώ πράγματι μειώνουν τις καθυστερήσεις και το κόστος της επικοινωνίας στο Federated Learning, αυτό γίνεται εις βάρος της ταχύτητας σύγκλισης του μοντέλου. Επιπλέον ισχυρίζονται ότι για να αποδόσουν οι συγκεκριμένες τεχνικές απαιτείται η συμμετοχή μεγάλου αριθμού clients στη διαδικασία της εκπαίδευσης.

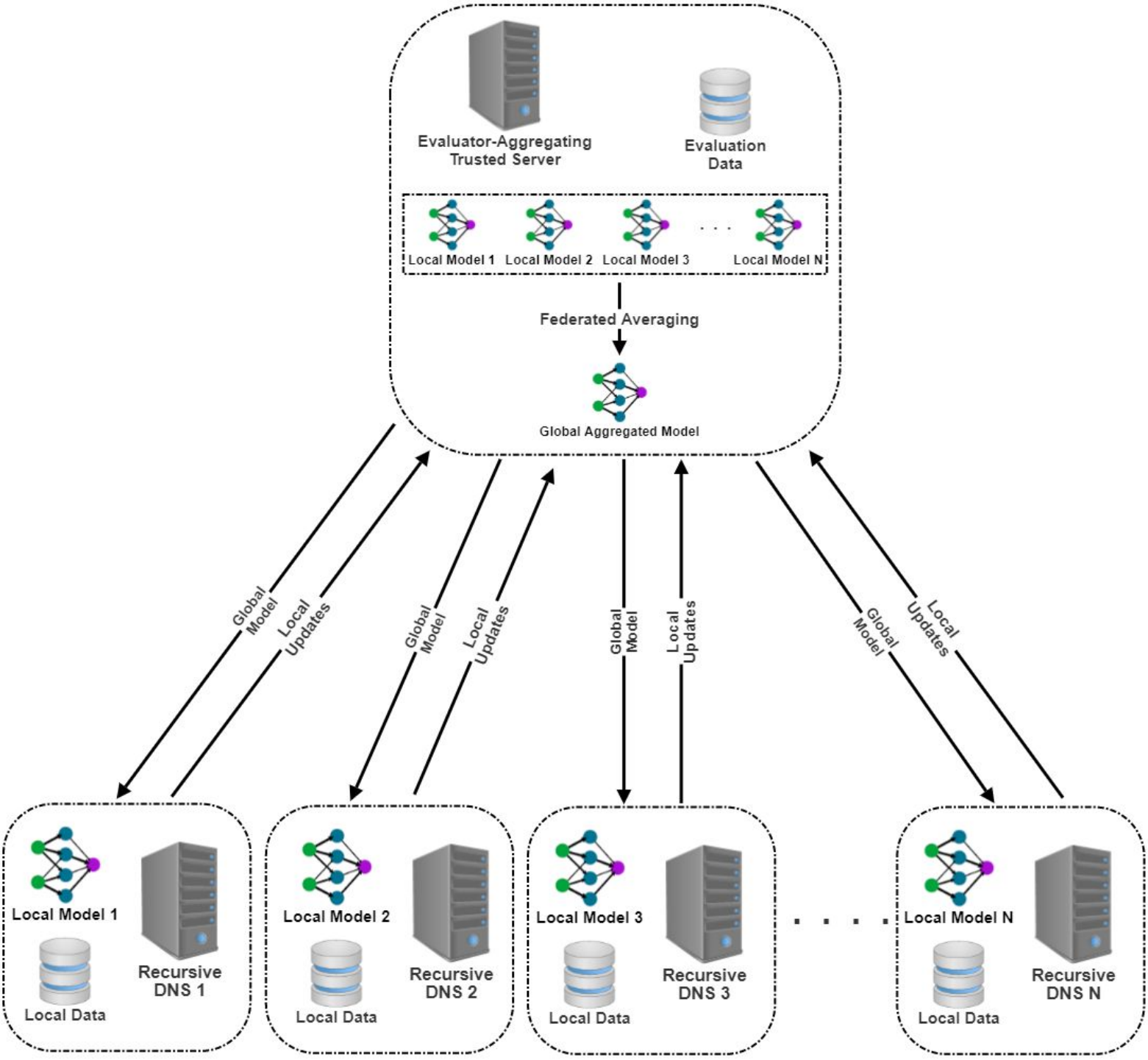
2.5.2 Federated Averaging

Ένα απο τα πιο σημαντικά σημεία στη διαδικασία του Federated Learning είναι ο τρόπος με τον οποίο συμψηφίζονται τα τοπικά μοντέλα σε ένα νέο γενικό. Ο αλγόριθμος που χρησιμοποιείται παραδοσιακά για τον σκοπό αυτό είναι το Federated Averaging. Κατά βάση, πρόκειται για μια πολύ απλή διαδικασία, στην οποία το γενικό μοντέλο προκύπτει υπολογίζοντας τον μέσο όρο των αντιστοίχων βαρών όλων των τοπικών μοντέλων. Εξού και η ονομασία που δόθηκε στον αλγόριθμο. Γενικά η επιλογή του κατάλληλου αλγορίθμου για την εξαγωγή του γενικού μοντέλου παίζει καθοριστικό ρόλο, σε κάθε πτυχή του Federated Learning, επηρεάζοντας είτε θετικά, είτε αρνητικά την ταχύτητα σύγκλισης του μοντέλου, το κόστος επικοινωνίας και τελικά την ακρίβεια του τελικού μοντέλου. Οι McMahan et al. [14] δίνουν την παρακάτω περιγραφή στην εργασία τους:

Αλγοριθμος 2.1: ***FederatedAveraging.*** *The K clients are indexed by k, B is the local minibatch size, E is the number of local epochs and η is the learning rate.*

Procedure ServerExecutes:
  initialize w_0
  for each round t = 1,2,... **do**
    $m \leftarrow max(C \times K, 1)$
    $S_t \leftarrow$ (*random set of m clients*)
    for each client k $\in S_t$ **in parallel do**
      $w_{t+1}^k \leftarrow ClientUpdate(k, w_t)$
    end for
    $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
  end for
end Procedure

Procedure ClientUpdate(k,w): // run on client k
  $B \leftarrow$ (*split* P_k *into batches of* B)
  for each local epoch i from 1 to E **do**
    for batch b $\in B$ **do**
      $w \leftarrow w - \eta \nabla \ell(w; b)$
    end for
  end for
end Procedure



Εικόνα 2.4: *Federated Learning for DGA-based Botnet Detection*

Από τον αλγόριθμο που περιγράφεται παρατηρούμε ότι η διαδικασία της εκπαίδευσης ξεκινάει με τον aggregating server να αρχικοποιεί το γενικό μοντέλο w_0. Όπου w_t είναι ο πίνακας των βαρών, δηλαδή των παραμέτρων που αντιπροσωπεύουν το μοντέλο σε κάθε γύρο εκπαίδευσης t. Ακολουθεί η αποστολή του γενικού μοντέλου στους clients. Στη συνέχεια, επιλέγεται ένα υποσύνολο S_t από clients που θα συμμετέχουν στον εκάστοτε γύρο εκπαίδευσης. Η λειτουργία αυτή αφορά περιπτώσεις όπου ο αριθμός των clients είναι μεγάλος. Όταν ο αριθμός τους είναι μικρός, το βήμα αυτό μπορεί να παραλείπεται και να συμμετέχουν όλοι οι clients σε κάθε γύρο. Μόλις οι clients ολοκληρώσουν την εκπαίδευση των τοπικών μοντέλων τους, τα επιστρέφουν στον aggregating server ώστε να συμψηφιστούν. Το νέο γενικό μοντέλο w_{t+1} προκύπτει από τον σταθμισμένο μέσο όρο των παραμέτρων των αντίστοιχων τοπικών μοντέλων. Ως συντελεστής βαρύτητας για κάθε τοπικό μοντέλο χρησιμοποιείται ο λόγος των εγγραφών n_k του client k προς τον συνολικό αριθμό εγγραφών n όλων των clients.

2.5.3 Federated Learning versus Distributed Deep Learning

Μιας και στη διπλωματική αυτή μας ενδιαφέρουν τα πλεονεκτήματα και τα μειονεκτήματα του Federated Learning σε σχέση με τις παραδοσιακές τεχνικές Distributed Deep Learning, για τη συνεργατική αντιμετώπιση των DGA-based botnets, προβαίνουμε στη σχετική σύγκριση. Συγκριτικά λοιπόν με το συμβατικό Distributed Deep Learning, κατά την εκπαίδευση ενός μοντέλου σε ένα περιβάλλον Federated Learning αποφεύγεται η συγκεντρωτική αποστολή των δεδομένων σε έναν κεντρικό server, με αποτέλεσμα να μην επιβαρύνεται το δίκτυο με μεγάλο φόρτο ανταλλαγής δεδομένων. Έτσι η εκπαίδευση δεν περιορίζεται και δεν καθυστερεί εξαιτίας πιθανών bottlenecks του δικτύου. Αντιθέτως, τα μόνα δεδομένα που ανταλλάσσονται σε κάθε γύρο εκπαίδευσης είναι οι παράμετροι των τοπικών μοντέλων, με σκοπό να συμψηφιστούν σε ένα νέο βελτιωμένο μοντέλο, στον κεντρικό server. Η φιλοσοφία πάνω στην οποία λειτουργεί το Federated Learning προτρέπει όλο και περισσότερους οργανισμούς να συμμετέχουν σε συνεργατικές διαδικασίες ενίσχυσης των μοντέλων τους, χωρίς να ανησυχούν για την πιθανή έκθεση των δεδομένων τους, αφού αυτά παραμένουν αποθηκευμένα στις τοπικές συσκευές καθόλη τη διάρκεια της εκπαίδευσης. Συνεπώς, λόγω της μεγαλύτερης προθυμίας των ενδιαφερόμενων οργανισμών να συμβάλλουν, η εκπαίδευση εκτελείται πάνω σε μια μεγαλύτερη ποικιλία δεδομένων. Μάλιστα, λόγω της συνεργατικής φύσεως του Federated Learning, παρατηρείται η αποδοτικότερη χρήση των υπολογιστικών πόρων του δικτύου, αφού πολλές συσκευές ταυτόχρονα συμβάλλουν στην εκπαίδευση με την υπολογιστική τους ισχύ.

Από την άλλη πλευρά, υπάρχουν κάποιοι περιοριστικοί παράγοντες που επιβαρύνουν την απόδοση του Federated Learning. Γενικά, στο Distributed Deep Learning, ο κεντρικός server για να επιταχύνει τη διαδικασία, διαμερίζει το dataset σε ομογενείς ομάδες δεδομένων (Identical Distributed Data - IDD) με παρόμοιο όγκο η καθεμιά, ώστε να εκπαιδευτούν κατανεμημένα σε ένα cluster. Αντιθέτως, στο Federated Learning, τα δεδομένα είναι εξαρχής διαχωρισμένα στις τοπικές συσκευές (στην περίπτωσή μας στους recursive DNS servers). Επομένως, το περιβάλλον εκπαίδευσης χαρακτηρίζεται τις περισσότερες φορές από ετερογενείς ομάδες δεδομένων (Non-Identical Distributed Data - Non-IDD), που μπορεί να διαφέρουν σημαντικά και σε όγκο. Για παράδειγμα, έστω ότι σε ένα περιβάλλον Distributed Deep

Learning το γενικό dataset αποτελούνταν από 6 κατηγορίες domain names (1 κατηγορία για τα γνήσια ονόματα και 5 οικογένειες DGA ονομάτων), με 30.000 εγγραφές για την κάθε κατηγορία. Τότε ο διαχωρισμός των δεδομένων θα γινόταν ομοιόμορφα, δηλαδή κάθε ομάδα θα αποτελούνταν από εγγραφές και των 6 κατηγοριών και μάλιστα ισόποσα μοιρασμένες. Αντιθέτως, σε ένα περιβάλλον Federated Learning, το πιο πιθανό είναι, αναλόγως με την έρευνα που είχε διενεργήσει κάθε οργανισμός σε εγγραφές Passive DNS, κάθε recursive DNS να γνωρίζει μόνο ένα μέρος των DGA οικογενειών. Δηλαδή ένας recursive DNS θα μπορούσε να κατέχει 10.000 εγγραφές αποτελούμενες μόνο από 2 οικογένειες DGA ονομάτων, ενώ ένας άλλος από 80.000 εγγραφές αποτελούμενες από 4 οικογένειες DGA ονομάτων. Σύμφωνα, με μια σειρά από μελέτες [18], [19] που έχουν επικεντρωθεί στη συγκεκριμένη ιδιότητα του Federated Learning, έχουν καταλήξει στο συμπέρασμα, ότι όσο αυξάνεται η ετερογένεια των δεδομένων μεταξύ των συσκευών που συμμετέχουν στην εκπαίδευση, είναι πιθανότερο να μειώνεται η ακρίβεια του τελικού γενικού μοντέλου. Αυτό είναι αναμενόμενο, διότι τα τοπικά μοντέλα που προκύπτουν, έχουν προσαρμόσει τα βάρη τους ώστε να διαχωρίζουν διαφορετικού τύπου δεδομένα σε κάθε συσκευή, με αποτέλεσμα οι παράμετροί τους να διαφέρουν σημαντικά και ως εκ τούτου, ο αλγόριθμος του Federated Averaging να αργεί να συγκλίνει. Ωστόσο, εκτός ακραίων περιπτώσεων ετερογένειας, στις περισσότερες εφαρμογές, η τάξη μείωσης της ακρίβειας δεν καθιστά τη χρήση του Federated Learning απαγορευτική. Μάλιστα, τα οφέλη που προκύπτουν από τη μεγαλύτερη συμμετοχή στη διαδικασία της εκπαίδευσης και άρα τη μεγαλύτερη ποικιλία και όγκο δεδομένων, είναι πιθανό να υπερκεράσουν τον περιορισμό αυτόν. Ένα άλλο βασικό χαρακτηριστικό μιας πραγματικής εφαρμογής Federated Learning, είναι η ετερογένεια που παρατηρείται στο hardware. Όπως είναι λογικό, σε ένα δίκτυο, κάθε συσκευή, ή στην περίπτωσή μας κάθε recursive DNS θα έχει διαφορετικές υπολογιστικές δυνατότητες και διαφορετικό φόρτο εργασίας κάθε χρονική στιγμή. Αντιπροσωπευτικό παράδειγμα αποτελεί η περίπτωση των strugglers που προαναφέρθηκε.

2.5.4 Ενισχυμένοι Αλγόριθμοι Συμψηφισμού Τοπικών Μοντέλων

Σε μια προσπάθεια για την ελαχιστοποίηση της αρνητικής επίδρασης των παραγόντων που αναλύθηκαν, οι Li et al. (2019), ανέπτυξαν έναν βελτιωμένο αλγόριθμο συμψηφισμού των μοντέλων, σε σχέση με τον παραδοσιακό Federated Averaging, τον FedProx [20]. Ο FedProx αποτέλει μια τροποποίηση του Federated Averaging, ως προς δύο σημεία. Αρχικά εισάγεται μια παράμετρος $\gamma \in [0, 1]$, η οποία εκφράζει την ανοχή σε λιγότερο ακριβή τοπικά μοντέλα. Η παράμετρος αυτή ορίζεται διαφορετική για κάθε client σε κάθε γύρο εκπαίδευσης και ο σκοπός της είναι να μειώσει την απώλεια πληροφορίας που οφείλεται στους strugglers. Αύξηση της τιμής της παραμέτρου συνεπάγεται αυξημένη ανοχή, οπότε και ένα πιο ανακριβές μοντέλο, αναλόγως του client για τον οποίο ορίστηκε. Ουσιαστικά, σκοπός είναι η καταπολέμηση της ετερογένειας στο hardware που χαρακτηρίζει την αρχιτεκτονική του Federated Learning. Επομένως, αντί να αποβάλλονται τα μοντέλα των strugglers από τον συμψηφισμό, προσδιορίζεται γι' αυτούς μια υψηλότερη ανοχή γ, ώστε να υπολογίσουν ένα τοπικό μοντέλο, έστω χαμηλότερης ακριβείας από αυτό που θα υπολόγιζαν κανονικά, εντός όμως της χρονικής προθεσμίας που έχει οριστεί. Κατά συνέπεια, μειώνεται η πληροφορία

που στερούμαστε κατά τον συμψηφισμό. Το δεύτερο σημείο στο οποίο διαφέρει ο FedProx είναι η εισαγωγή μιας ποινής $\mu \in [0, 1]$, ώστε κατά την εκπαίδευση των τοπικών μοντέλων οι ενημερώσεις να προκαλούν μικρότερες μεταβολές στα βάρη, για να μην απέχουν πολύ από το γενικό μοντέλο. Με τον τρόπο αυτό το γενικό μοντέλο αργεί να συγκλίνει, αλλά όταν τα δεδομένα χαρακτηρίζονται από μεγάλη ετερογένεια, επιτυγχάνει μεγαλύτερη τελική ακρίβεια, καθώς αποφεύγεται η εισαγωγή θορύβου. Ο σχετικός κώδικας είναι διαθέσιμος στον σύνδεσμο: https://github.com/litian96/FedProx.

Με τη σειρά τους, οι Wang et al. (2020), πρότειναν έναν καινούριο αλγόριθμο [21], ο οποίος ισχυρίζονται ότι πετυχαίνει καλύτερη ακρίβεια κατά τον συμψηφισμό, τόσο από τον Federated Averaging, όσο και από τον FedProx. Ο αλγόριθμός αυτός ονομάζεται FedMa (Federated Matching) και η υλοποίησή του διαφέρει σημαντικά από αυτή του Federated Averaging, εμφανίζοντας μάλιστα αυξημένη υπολογιστική πολυπλοκότητα. Όπως φαίνεται όμως από τα πειράματα που εκτελέστηκαν στη συγκεκριμένη μελέτη, πράγματι τα αποτελέσματα είναι εντυπωσιακά, αφού κατά τη σύγκριση των τριών αλγορίθμων πάνω στις ίδιες εφαρμογές, παρουσιάζεται μια βελτίωση 2-4% στην ακρίβεια σε σχέση με τον ήδη ενισχυμένο FedProx. Συνοπτικά, η κεντρική ιδέα του FedMa είναι να συμπεριλαμβάνονται κατά τον συμψηφισμό των τοπικών μοντέλων, τα βάρη εκείνα που συμβάλλουν περισσότερο στην απόδοση του μοντέλου. Με τη συγκεκριμένη προσέγγιση, παρατηρείται καθοριστική ελλάττωση στον αρνητικό παράγοντα της ετερογένειας των δεδομένων. Παρόλα αυτά, αν και ο FedMa αποτελεί μια πολλά υποσχόμενη τεχνική, πρέπει να λαμβάνουμε υπόψη μας την αυξημένη υπολογιστική πολυπλοκότητα που τον χαρακτηρίζει, καθώς και τη δύσκολη γενικά υλοποίησή του, συγκριτικά με τους δύο προηγούμενους αλγορίθμους που εξετάστηκαν. Ο σχετικός κώδικας είναι διαθέσιμος στον σύνδεσμο: https://github.com/IBM/FedMA.

2.5.5 Ζητήματα Ασφαλείας και Ιδιωτικότητας

Ένα καίριο ζήτημα που πρέπει να καταπολεμήσει το Federated Learning είναι η προστασία της ιδιωτικότητας των δεδομένων. Όπως έχει ήδη εξηγηθεί, η συγκεκριμένη αρχιτεκτονική αποτελεί ένα μεγάλο βήμα προς αυτήν την κατεύθυνση, αφού τα δεδομένα των clients παραμένουν αποθηκευμένα στις τοπικές συσκευές καθ' όλη τη διάρκεια της εκπαίδευσης, χωρίς να υπάρχει κίνδυνος έκθεσης και υποκλοπής τους από κακόβουλους. Ωστόσο, πρόσφατες έρευνες έχουν δείξει, ότι το Federated Learning από μόνο του δεν μπορεί να εγγυηθεί την απόλυτη ιδιωτικότητα των δεδομένων. Οι Wand et al. (2019) [22] προκειμένου να αναδείξουν πιθανές αδυναμίες του, επικεντρώθηκαν στο γεγονός πως οι clients θεωρούν ότι ο κεντρικός server που είναι υπεύθυνος για τον συμψηφισμό των τοπικών μοντέλων παραμένει αξιόπιστος. Αντιθέτως, στην έρευνά τους θεώρησαν ένα σενάριο όπου ο aggregating server εκτελεί μεν σωστά τη διαδικασία για την οποία είναι υπεύθυνος, αλλά ταυτόχρονα χαρακτηρίζεται από την περιέργειά του να αποκτήσει πληροφορίες σχετικά με τη φύση των δεδομένων των clients (honest-but-curious server). Ανέπτυξαν ένα σύστημα, το mGAN-AI, το οποίο εκτελείται στον aggregating server και μελετώντας το τοπικό μοντέλο ενός client και τη συμβολή αυτού στην απόδοση του γενικού μοντέλου, εξάγει συμπεράσματα σχετικά με τη δομή και τη φύση των δεδομένων του. Μάλιστα, σε ορισμένες περιπτώσεις υπάρχει ακόμα και η δυνατότητα της κατά προσέγγιση ανακατασκευής ορισμένων δειγμάτων του dataset.

Η συγκεκριμένη επίθεση δεν γίνεται αντιληπτή, αφού δεν επιφέρει κάποια καθυστέρηση ή μείωση στην απόδοση.

Γίνεται λοιπόν αντιληπτό, ότι αν δεν υπάρχουν αδιάσειστα στοιχεία ότι ο aggregating server είναι απολύτως αξιόπιστος, υπάρχει το ρίσκο, έστω και με μικρή πιθανότητα, έκθεσης των δεδομένων ενός client. Επίσης, κατά την αποστολή των τοπικών μοντέλων, θα πρέπει να συνυπολογιστεί η πιθανή παρεμβολή ενός κακόβουλου που θα παρακολουθεί την κίνηση και θα υποκλέπτει τα μηνύματα που προορίζονται για τον aggregating server (gradient leakage). Για την εξάλειψη αυτής της πιθανότητας, έχουν προταθεί μια σειρά από τεχνικές, όπως το Differential Privacy, που μπορούν να συνδυαστούν με το Federated Learning, ώστε ένας honest-but-curious server να έχει πλήρη άγνοια για τα δεδομένα των clients. Το Differential Privacy συνιστά μια ειδική μορφή κρυπτογραφίας και εφαρμόζεται στην περίπτωσή μας, για να επιτευχθεί ο στόχος που στη βιβλιογραφία είναι γνωστός ως secure aggregation, ή αλλιώς ασφαλής συμψηφισμός. Η ιδέα του βασίζεται στην εισαγωγή θορύβου στα δεδομένα, με τη χρήση ειδικών συναρτήσεων (π.χ. Laplace noise), ώστε γνωρίζοντας την έξοδο ενός υπολογισμού, να μην είναι εφικτός ο προσδιορισμός της εισόδου (εν προκειμένου των παραμέτρων των τοπικών μοντέλων που χρησιμοποιούνται για τον συμψηφισμό) [23]. Οι Geyer et al. (2017) [24] μελέτησαν τη συμβολή του Differential Privacy στο Federated Learning και συμπέραναν ότι με με ένα μικρό trade-off στην ακρίβεια (1%), μειώνεται στο ελάχιστο η πιθανή απειλή ενός κακόβουλου aggregating server.

Μία άλλη απειλή που καλείται να αντιμετωπίσει το Federated Learning είναι οι λεγόμενες επιθέσεις data poisoning και model poisoning. Οι επιθέσεις αυτές αφορούν αντίστοιχα στην εσκεμμένη αλλαγή των δεδομένων (π.χ. αλλάζοντας το label ορισμένων εγγραφών, ώστε να ανήκουν σε λάθος κλάση) και στην τροποποίηση των παραμέτρων ενός τοπικού μοντέλου, ώστε να μειώνεται η ακρίβειά του. Οι τροποποιήσεις γίνονται πολύ προσεκτικά και σε περιορισμένο βαθμό, ώστε να μην γίνονται αντιληπτές οι ανωμαλίες που προκύπτουν σε σχέση με τα γνήσια τοπικά μοντέλα. Τέτοιες ενέργειες εκτελούνται συνήθως από κακόβουλους clients που συμμετέχουν στο συνεργατικό σχήμα, με αποκλειστικό σκοπό την υποβάθμιση της ποιότητας της εκπαίδευσης και της ακρίβειας του τελικού γενικού μοντέλου. Στην περίπτωσή μας, όπου εφαρμόζουμε ως μέθοδο το Federated Learning για τον εντοπισμό κακόβουλων ονομάτων DGA, ένας client που θα είχε συμφέρον από την υποβάθμιση του γενικού μοντέλου, θα μπορούσε να είναι ένας botmaster, ο οποίος συμμετέχει από έναν παραβιασμένο recursive DNS. Αν και η πιθανότητα αυτή είναι μικρή, οι Bhagoji et al. [25] μελέτησαν διάφορες στρατηγικές για την ενορχήστρωση τέτοιων επιθέσεων και συμπέραναν ότι ακόμα και ένας κακόβουλος να υπάρχει σε ένα αρκετά μεγαλύτερο σύνολο από clients, μπορεί να προκληθεί μια σεβαστή μείωση στην ακρίβεια του τελικού μοντέλου. Επιπλέον συμπέραναν ότι οι επιθέσεις model poisoning είναι πιο αποδοτικές και πιο δύσκολο να εντοπιστούν. Για την καταπολέμησή τους έχουν προταθεί συστήματα, όπως το Auror [26], που εντοπίζει τα αλλοιωμένα τοπικά μοντέλα και τα αποκλείει από τον συμψηφισμό. Μάλιστα, οι εμπνευστές του (Shen et al.) κατάφεραν στα πειράματά τους, χρησιμοποιώντας το Auror, να περιορίσουν τη μείωση στην ακρίβεια στο 3%, ακόμα και όταν το 30% των clients είναι κακόβουλοι. Από την άλλη πλευρά, οι Chen et al. (2020), για να εξαλείψουν την παρουσία κακόβουλων

clients, ανέπτυξαν ένα πρωτόκολλο [27] για εκτέλεση του Federated Learning σε ένα αξιόπιστο περιβάλλον υπολογισμού (Trusted Execution Environment). Κάθε client ελέγχεται κατά τον συμψηφισμό για το αν εφαρμόζει το πρωτόκολλο ασφαλείας που προτείνεται, μέσω ενός ειδικού signature. Για την εκπαίδευση του τοπικού του μοντέλου, κάθε client, υποχρεώνεται από το πρωτόκολλο να δεσμεύσει μια απομονωμένη περιοχή μνήμης στη συσκευή του, όπου θα εκτελούνται οι απαραίτητοι υπολογισμοί, χωρίς την παρεμβολή άλλων προγραμμάτων. Εφόσον η προϋπόθεση αυτή τηρείται, το μήνυμα προς αποστολή σφραγίζεται με το signature που υποδηλώνει την ακεραιότητα και γνησιότητα των υπολογισμών.

Σε κάθε περίπτωση, στη συγκεκριμένη διπλωματική θα περιοριστούμε στη θεωρητική αναφορά των ζητημάτων ιδιωτικότητας και ασφαλείας του Federated Learning, καθώς σε πρώτο στάδιο μας ενδιαφέρει να εξετάσουμε την απόδοσή του, συγκριτικά με τις κλασσικές μεθοδολογίες Distributed Deep Learning για συστήματα εντοπισμού DGA-based botnet.

Κεφάλαιο 3

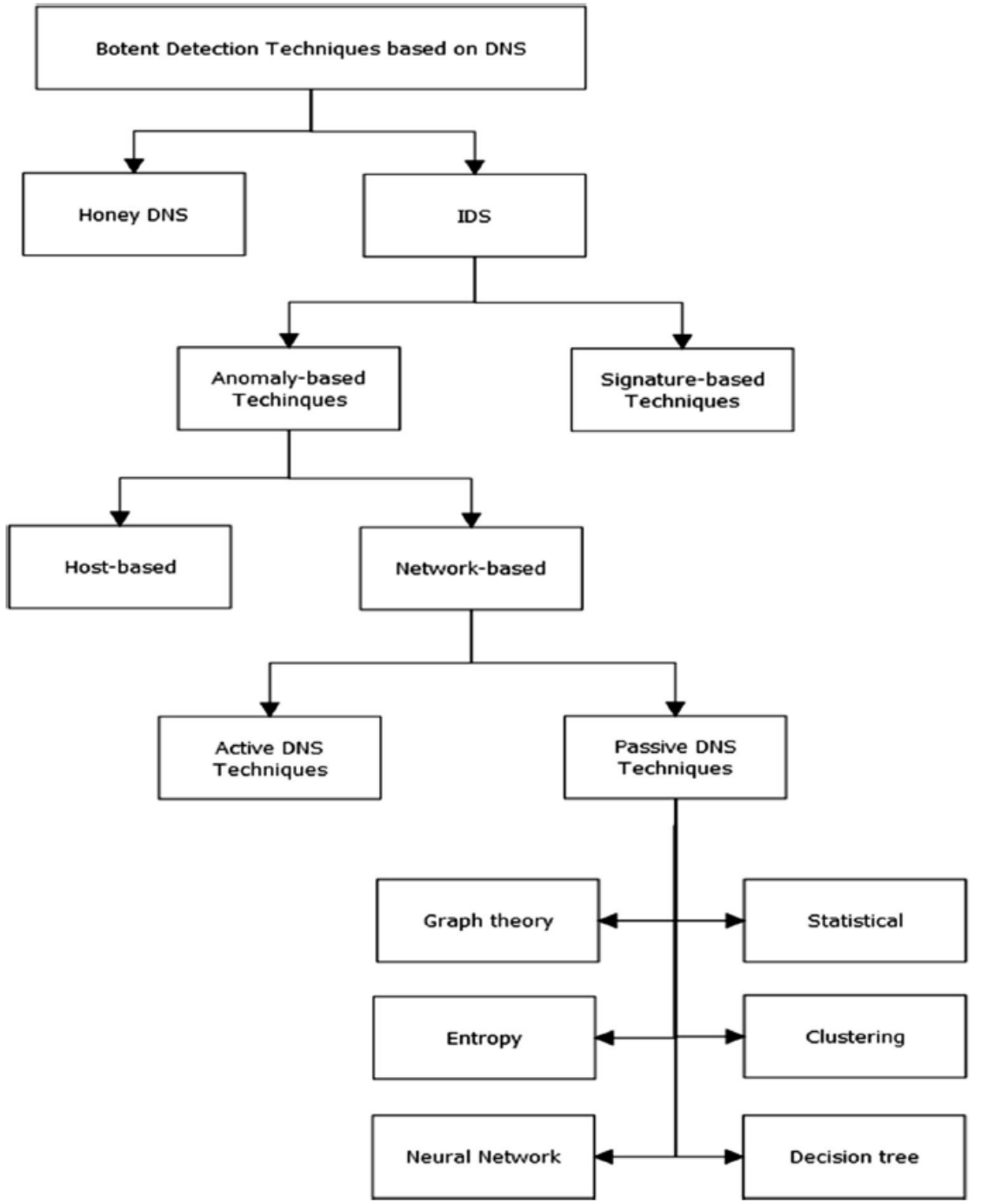
Σχετική Έρευνα που Προηγείται

Η μελέτη των Plohmann et al. (2016) [7] περιγράφει λεπτομερώς τον ρόλο των DGAs στα σύγχρονα botnets. Η μελέτη τους επικεντρώθηκε στην ανάλυση και αξιολόγηση 43 διαφορετικών botnets, παρατηρώντας ότι 23 από τα 43 χρησιμοποιούν DGAs ως μοναδικό μηχανισμό επικοινωνίας με τους C&C servers τους. Οι Barabosch et al. (2012) [8] καθόρισαν μια ταξινόμηση των DGA βάσει δύο ιδιοτήτων, του χρόνου (time-dependence) και της αιτιότητας (determinism). Η χρονική διάσταση καταγράφει εάν τα seeds είναι σταθερά ή αν παράγονται δυναμικά, με το πέρας κάποιας περιόδου. Επιπλέον, τα seeds μπορούν να είναι ντετερμινιστικά (υπολογισμένα μέσω μιας σταθερής διαδικασίας), ή μη ντετερμινιστικά (απρόβλεπτα, χρησιμοποιώντας π.χ. μετεωρολογικές προβλέψεις ή τιμές χρηματιστηρίου). Οι Alieyan et al. (2015) [28] κατηγοριοποίησαν και εξήγησαν αναλυτικά όλες τις προσεγγίσεις για τον εντοπισμό κίνησης botnet, όπως παρουσιάζεται στην Εικόνα 3.1, που χρησιμοποιήθηκε μέσα από τη μελέτη τους. Όπως υποστηρίζουν και οι ίδιοι, η πιο μοντέρνα και scalable, προσέγγιση είναι αυτή των τεχνικών που εκμεταλλεύονται τα δεδομένα των Passive DNS, για την ανάπτυξη αποδοτικών ανιχνευτών.

Θα ήταν λογικό να διακρίνουμε τις προηγούμενες προσεγγίσεις για την ανίχνευση DGA domain names σε δύο κατηγορίες:

1. **Ανασκοπικές (Retrospective):** βασίζονται συνήθως στην ομοιότητα των NXDomain responses που επιστρέφονται σε δύο η περισσότερους hosts. Πιο συγκεκριμένα, μολυσμένες συσκευές (bots) που ανήκουν στο ίδιο botnet και εκτελούν τον ίδιο DGA, θα λαμβάνουν συχνότερα αποκρίσεις σφάλματος NXDomain (σε σχέση για παράδειγμα με καλόβουλους hosts που εκτέλεσαν λάθος DNS query λόγω τυπογραφικού) και μάλιστα αυτές θα είναι παρόμοιες μεταξύ τους με μεγάλη πιθανότητα. Επομένως, αναλύοντας τα NXDomain responses και συγκρίνοντάς τα, εξάγεται ένα σύνολο κοινών στατιστικών χαρακτηριστικών, με βάση τα οποία γίνεται η κατηγοριοποίηση των ερωτηθέντων domain names σε legit ή DGA.

2. **Σε πραγματικό χρόνο (Real-time):** ανίχνευση domain names αποκλειστικά με πληροφορίες από τη μορφή των domain names καθαυτών.



Εικόνα 3.1: *Κατηγοριοποίηση Τεχνικών Εντοπισμού Botnet με Βάση τα Χαρακτηριστικά της Κίνησης DNS*

3.1 Ανασκοπική (Retrospective) Ανίχνευση DGA

Οι McGrath και Gupta (2008) [29] εξέτασαν διάφορα χαρακτηριστικά δικτύου, όπως διευθύνσεις IP, εγγραφές ¨Whois¨ και λεξιλογικά χαρακτηριστικά URL, για domains που ταξινομούνται σε γνήσιους, ή κακόβουλους ιστότοπους για phishing. Παρατήρησαν ότι κάθε κλάση εμφανίζει διαφορετική κατανομή αλφαβήτου. Το συμπέρασμά τους ήταν, ότι τα κακόβουλα domain names είναι συντομότερα από τα γνήσια, τα οποία χρησιμοποιούν ως

επί το πλείστον λιγότερα φωνήεντα και παρουσιάζουν σημαντικές διαφορές στις αλφαβητικές κατανομές τους. Το κίνητρό τους ήταν να βρουν χρήσιμα ευρετικά για το φιλτράρισμα phishing μηνυμάτων και για τον εντοπισμό κίνησης botnet.

Οι ερευνητές της Cisco ανέπτυξαν το 2015 μία πρώτη εκδοχή ενός συστήματος ανίχνευσης DGA [30], που εξέλιξαν και χρησιμοποίησαν στη συνέχεια. Παρουσίασαν έναν αλγόριθμο που βασίζεται στα γλωσσικά χαρακτηριστικά για τον εντοπισμό των αλγοριθμικά δημιουργημένων ονομάτων. Ο αλγόριθμος εκχωρεί μια βαθμολογία τυχαιότητας σε κάθε domain name προκειμένου να αποφασίσει εάν δημιουργείται αλγοριθμικά ή όχι. Για την εκτίμηση αυτής της βαθμολογίας, δημιούργησαν πρώτα ένα μεγάλο σύνολο λεξικών που αντιπροσωπεύουν διάφορες γλώσσες, π.χ. αγγλικά, γαλλικά, κινέζικα, κ.λ.π., καθώς και αγγλικά ονόματα, domain names από την Alexa και ακρωνύμια. Τα παραπάνω χρησιμοποιήθηκαν για την εύρεση σημαντικών ακολουθιών σε γνήσια domain names, τα οποία είναι απίθανο να εμφανίζονται σε ένα όνομα που δημιουργείται από DGA. Για κάθε όνομα που ελέγχθηκε, εξήχθησαν όλες οι υπακολουθίες και υπολογίστηκαν διάφορα χαρακτηριστικά, όπως ο αριθμός των υπακολουθιών που εμφανίζονταν στα λεξικά, το αντίστοιχο μήκος τους και ο αριθμός των διαφορετικών γλωσσών που χρησιμοποιούνταν. Από τα εξαγόμενα χαρακτηριστικά δημιούργησαν ένα γραμμικό μοντέλο που υπολογίζει τη βαθμολογία τυχαιότητας. Πέτυχαν μάλιστα ένα ποσοστό ψευδών θετικών προβλέψεων μεταξύ 0 και 2% για εννέα διαφορετικούς DGAs.

Οι Yadav et. al. (2012) [31] παρουσίασαν μια μεθοδολογία για την ανίχνευση domain-fluxing, εξετάζοντας την κατανομή των αλφαριθμητικών χαρακτήρων, χρησιμοποιώντας τόσο unigrams όσο και bigrams. Αυτή η μεθοδολογία βασίζεται στην υπόθεση, ότι υπάρχει μια σημαντική διαφορά μεταξύ των domain names που δημιουργούνται από τον άνθρωπο και από τους DGAs, αναφορικά με την κατανομή των αλφαριθμητικών χαρακτήρων. Αρχικά ομαδοποίησαν ερωτήματα DNS που μοιράζονται το ίδιο second level domain. Στη συνέχεια, σε κάθε ομάδα υπολόγισαν την απόσταση KL-Divergence και Jaccard Index σε unigrams και bigrams των ονομάτων. Το N-gram μιας συμβολοσειράς είναι μια ομάδα υπακολουθιών μεγέθους N, τα οποία εξάγονται χρησιμοποιώντας ένα κυλιόμενο παράθυρο μήκους N από την αρχή της συμβολοσειράς έως το τέλος. Για παράδειγμα, τα unigrams της λέξης "domain" θα είναι "d", "o", "m", "a", "i", "n" και τα bigrams αυτής, "do", "om", "ma", "ai", "in". Αξιολόγησαν τα πειράματά τους στα αποτελέσματα δικτύου μιας ημέρας, του Tier-1 ISP της Ασίας και της Νότιας Αμερικής και εντόπισαν το Conficker, καθώς και ορισμένα άλλα άγνωστα ως τότε botnets. Οι ίδιοι ισχυρίστηκαν ότι πέτυχαν ποσοστό ακρίβειας που προσεγγίζει το 100%, με ψευδώς θετικά λιγότερα από 6%. Ωστόσο, υπάρχουν κάποια μειονεκτήματα στην προσέγγισή τους: Αρχικά, οι μετρήσεις που προτείνονται στην εργασία τους απαιτούν έναν ελάχιστο αριθμό domain names σε κάθε ομάδα για την επίτευξη ακριβών αποτελεσμάτων. Ωστόσο, πολλές ομάδες σε περιπτώσεις real-life εφαρμογών δεν πληρούν αυτήν την απαίτηση, με αποτέλεσμα πολλά domain names να μην ταξινομηθούν σωστά, αφού οι κλάσεις στις οποίες ανήκουν θα απορρίπτονταν λόγω ελλειπών μετρήσεων. Μία άλλη αδυναμία εντοπίζεται σε έναν από τους δείκτες που χρησιμοποιούνται για τον εντοπισμό DGA. Ο λόγος γίνεται για τον δείκτη Jaccard (JI). Εδώ ο JI εφαρμόζεται σε σύνολα bigram, των ονομάτων. Όσο

μεγαλώνει το μέγεθος των bigrams set , θα επιτυγχάνεται μεγαλύτερη ακρίβεια, με τίμημα όμως όλο και μεγαλύτερης κατανάλωσης μνήμης και επεξεργαστικής ισχύος.

Οι Antonakakis et al. (2012) [1] πρότειναν ένα σύστημα ανίχνευσης που ονομάζεται Pleiades. Στην εργασία τους εκμεταλλεύονται για πρώτη φορά το γεγονός ότι η απόκριση σε ερωτήματα που αφορούν DGA domain names θα είναι ως επί το πλείστον NXDomain και ότι οι μολυσμένες συσκευές που ανήκουν στο ίδιο botnet και εκτελούν τον ίδιο DGA, θα λάβουν παρόμοια απόκριση σφάλματος NXDomain. Περιορίστηκε έτσι ο φόρτος εργασίας του συστήματος μόνο στις συγκεκριμένες αποκρίσεις. Συνδυάζουν τεχνικές ταξινόμησης, για να ομαδοποιήσουν domain names με παρόμοια χαρακτηριστικά συμβολοσειρών και στη συνέχεια να προσδιορίσουν ποιον DGA εκτελούν. Το Pleiades συνδυάζει τα ονόματα που αντιστοιχούν σε NXDomain (πιθανά ονόματα DGA) με λίστες γνωστών και νόμιμων domain names, καθώς και με άλλες που αποτελούνται από ήδη γνωστά ονόματα που δημιουργήθηκαν από κάποιον DGA. Παράλληλα, για την ενίσχυση του συστήματος, χρησιμοποιήθηκε ένα σύνολο στατιστικών χαρακτηριστικών (π.χ. μήκος ονομάτων, επίπεδο τυχαιότητας και κατανομή συχνότητας χαρακτήρων). Ένα Hidden Markov Model χρησιμοποιήθηκε τελικά για κάθε έναν DGA για την ομαδοποίηση και τον εντοπισμό νέων αλγοριθμικά παραγόμενων ονομάτων. Αφού ανακάλυψαν τα ονόματα που παράγονται από τον ίδιο DGA, ανέπτυξαν μια μέθοδο για τον εντοπισμό του C&C server. Σε διάστημα 15 μηνών, βρήκαν 12 DGAs, εκ των οποίων μόνο οι μισοί ήταν ήδη γνωστοί. Στα πειράματά τους πέτυχαν ακρίβεια 95-99% και ένα ποσοστό ψευδών θετικών 0,1-0,7%. Ομοίως, οι Zhou et al. (2013) [32] κατέγραψαν τις αποκρίσεις NXDomain και ομαδοποίησαν τα DGA domain names, με κριτήριο τον χρόνο ζωής του κάθε ονόματος, καθώς και με το μοτίβο επισκεψιμότητάς του (πόσα DNS queries αντιστοιχούν στο κάθε όνομα σε διάφορες χρονικές περιόδους).

Οι Nguyen et al. (2015) [33] παρουσίασαν ένα σύστημα μεγάλης κλίμακας, για τον εντοπισμό botnet που βασίζονται σε DGA. Αυτό το σύστημα έχει τη δυνατότητα να ανιχνεύει κίνηση botnet, αναλύοντας τα αρχεία καταγραφής κίνησης DNS του δικτύου. Ισχυρίζονται ότι το σύστημά τους μπορεί να εντοπίσει νέες εκδόσεις ενός DGA, το οποίο αποτελούσε παραδοσιακά πρόβλημα σε μεθόδους reverse-engineering. Η νέα μέθοδος που πρότειναν βασίζεται σε μια πλατφόρμα Big Data και χρησιμοποιεί συνεργατικό φιλτράρισμα και ομαδοποίηση βάσει πυκνότητας πιθανότητας. Ο αλγόριθμός τους βασίζεται στην ομοιότητα της χαρακτηριστικής κατανομής των αλγοριθμικά παραγόμενων ονομάτων, για την αφαίρεση του θορύβου και την ομαδοποίηση αυτών. Η τεχνική τους απέδωσε ψευδώς θετικό ποσοστό 18% και ψευδώς αρνητικό ποσοστό 22%. Τα αποτελέσματα αυτά υποδεικνύουν σημαντική μείωση της απόδοσης του μοντέλου εντοπισμού, σε σχέση με προηγούμενες μεθόδους. Το κυριότερο μειονέκτημα όμως είναι, ότι πριν από την ανάλυση των τομέων, πρέπει να καταγραφεί ολόκληρη η κίνηση DNS από τα αρχεία καταγραφής των χρηστών. Στην ιδανική περίπτωση όμως, θα μας ενδιέφερε μια υλοποίηση που λαμβάνει μόνο τα domain names ως είσοδο, χωρίς επιπλέον πληροφορίες, οι οποίες επιβαρύνουν το σύστημα.

Επιχειρώντας να επιλύσουν τους παραπάνω προβληματισμούς, οι Schiavoni et al. (2014) [34] ανέπτυξαν το Phoenix, που βασίστηκε σε έναν συνδυασμό γλωσσικών χαρακτηριστικών και σε IP addresses για τη διάκριση των κακόβουλων ονομάτων και αναγνώριση των DGA από

τους οποίους δημιουργούνται. Δυστυχώς, το σύστημα αυτό έχει ένα σημαντικό μειονέκτημα που εντοπίζεται στη χρήση, μεταξύ άλλων, διευθύνσεων IP, που οδηγεί στη μείωση της ακρίβειας και σε αυξημένο FPR, αφού μηχανισμοί κοινής χρήσης IP, όπως το πρωτόκολλο NAT, προκαλούν σύγχηση στη διαδικασία κατηγοριοποίησης. Επιπλέον, το Phoenix λειτουργεί μόνο για ονόματα που δημιουργούνται τυχαία, αλλά όχι για εκείνα που δημιουργούνται από συνένωση πραγματικών λέξεων (wordlist-based).

Πολλές από τις μελέτες που περιγράφηκαν έως τώρα, απαιτούν τη συλλογή κίνησης DNS από πολλά δίκτυα και την παρουσία πολλαπλών bots, από το ίδιο botnet. Αυτοί οι περιορισμοί δυσχεραίνουν τον εντοπισμό των bots όταν δεν υπάρχει πρόσβαση σε δεδομένα έξω από ένα ενιαίο δίκτυο. Με αυτό ως δεδομένο, οι Kwon et al. (2015) [35] ανέπτυξαν μια scalable προσέγγιση, το PsyBoG. Ο αλγόριθμός τους αξιοποίησε την ανάλυση φασματικής πυκνότητας ισχύος (PSD), για να μελετήσει τις κύριες συχνότητες, που δίδονται από τα περιοδικά ερωτήματα DNS των botnets. Το PsyBoG έχει την ιδιαιτερότητα, ότι αντιστοιχίζει τις μετρήσεις του σε διευθύνσεις IP και όχι στα domain names, όπως φαίνεται να συνηθίζουν οι πιο μοντέρνες τεχνικές. Η ιδέα πίσω από αυτήν την επιλογή ήταν η αντιμετώπιση του προβλήματος που προκύπτει από την ταχεία αυξανόμενη κίνηση DNS, με τη συνεχή επέκταση του Internet και των domain names. Έτσι, για να αποφευχθεί ο επιπλέον φόρτος εργασίας, κατηγοριοποιώντας ξεχωριστά έναν μεγάλο αριθμό από ονόματα, επιλέχθηκε η ανίχνευση σε επίπεδο IP, των οποίων ο αριθμός παραμένει σταθερός. Πέτυχε ένα ποσοστό ανίχνευσης 95% και εντόπισε 23 άγνωστες και 26 γνωστές οικογένειες κακόβουλου λογισμικού με 0,1% ψευδώς θετικές προβλέψεις (FPR). Αν και τα αποτελέσματα που αναφέρουν είναι ικανοποιητικά, φαίνεται εν τέλει, ότι το σύστημά τους μπορεί να παρακάμπτεται εύκολα από τους κακόβουλους. Κατά πρώτον, όπως εξηγήθηκε και προηγουμένως, η κατηγοριοποίηση σε επίπεδο IP επηρεάζεται αρνητικά από την χρήση πρωτοκόλλων κοινής χρήσης IP, όπως το NAT, ή ακόμα και από τη χρήση Proxy και VPN. Σε αυτήν την περίπτωση, ο εντοπισμός των bots, είναι πολύ δύσκολος και χρονοβόρος. Μία δεύτερη αδυναμία αυτής της τεχνικής είναι η επιλογή ως κριτήριο κατηγοριοποίησης, της συχνότητας των περιοδικών DNS queries. Το κριτήριο αυτό όταν χρησιμοποιείται ως το κύριο μέσο κατηγοριοποίησης δεν είναι επαρκές, καθώς ένας κακόβουλος μπορεί εύκολα να ρυθμίσει το botnet του, ώστε να εκτελεί τα ερωτήματα με μεγαλύτερο ενδιάμμεσο χρονικό περιθώριο. Κατ' αυτόν τον τρόπο δεν θα είναι πλέον ξεκάθαρος ο διαχωρισμός τους από την γνήσια κίνηση. Το BotDigger (2016) [36] αναλύοντας τη συχνότητα των DNS queries και χρησιμοποιώντας επιπλέον γλωσσικά και στατιστικά χαρακτηριστικά εντόπισε όλα τα bots του Kraken malware και το 99,8% των Conficker bots. Τελικά, προς αυτήν την κατεύθυνση, οι Wang et al. (2016) ανέπτυξαν το DBod [37]. Το τελευταίο έχει τη δυνατότητα να εντοπίζει κίνηση botnet χωρίς καμία προηγούμενη εκπαίδευση, ή πληροφορία. Χρησιμοποιώντας τις αποκρίσεις NXDomain και εξάγοντας στατιστικά στοιχεία για την κίνηση DNS, πετυχαίνει ακρίβεια μεγαλύτερη απο 99% για κάποιους DGAs, ενώ το FPR δεν ξεπερνά το 0,5%. Μάλιστα, οι δημιουργοί του ισχυρίζονται ότι είναι ικανό να εντοπίζει ακόμα και κρυπτογραφημένη κακόβουλη κίνηση.

Όλες οι παραπάνω προσεγγίσεις για ανίχνευση κίνησης botnet χαρακτηρίζονται ως ανασκοπικές. Το κύριο πρόβλημα αυτής της φιλοσοφίας υλοποίησης είναι, ότι για τη διαδικασία

της κατηγοριοποίησης απαιτείται η εκ των πρωτέρων ύπαρξη ενός συνόλου (batch) domain names, γεγονός που σε εφαρμογές πραγματικού χρόνου δεν είναι πάντα εφικτό.

3.2 Ανίχνευση DGA σε Πραγματικό Χρόνο

Μετά από μια εις βάθος έρευνα [38] και σύγκριση των παραπάνω τεχνικών, προέκυψαν δύο σημαντικά συμπεράσματα. Αφενός μεν, οι ανασκοπικές προσεγγίσεις έχουν μεγάλες απαιτήσεις σε χρόνο, ενώ οι περισσότερες πραγματικές εφαρμογές που βασίζονται στη συγκεκριμένη φιλοσοφία, συχνά χρειάζονται ώρες για τον εντοπισμό κακόβουλων ονομάτων. Το γεγονός αυτό καθιστά τις ανασκοπικές τεχνικές ανίχνευσης προβληματικές, αν αναλογιστούμε ότι ένας C&C server ενός DGA-based Botnet μπορεί να αλλάζει domain name έως και αρκετές φορές μέσα στη μέρα. Αφετέρου δε, η απόδοση αυτών των συστημάτων σε πραγματικές εφαρμογές είναι αρκετά μειωμένη σε ότι αφορά την ακρίβεια και το FPR.

3.2.1 Ανίχνευση με Εξαγωγή Χαρακτηριστικών από τον Άνθρωπο

Λαμβάνοντας υπόψη τους τους παραπάνω περιορισμούς, ορισμένοι ερευνητές προσπάθησαν να ανιχνεύσουν και να κατηγοριοποιήσουν κίνηση botnet, με βάση μεμονωμένα domain names. Οι Bilge et al. (2011)[39] πρότειναν ένα σύστημα, το EXPOSURE, που υιοθετεί τεχνικές passive DNS μεγάλης κλίμακας για την ανίχνευση κακόβουλης κίνησης DGA. Οι συγγραφείς χρησιμοποίησαν 15 χαρακτηριστικά που εξήχθησαν, αναλύοντας την κίνηση DNS από τα διαφορετικά χαρακτηριστικά και τις ιδιότητες των DGA domain names, καθώς και από το χρονικό μοτίβο εκτέλεσης των σχετικών DNS ερωτημάτων. Ύστερα χρησιμοποιείται ένας δυαδικός ανιχνευτής που κατηγοριοποιεί τα domain names σε νόμιμα και κακόβουλα. Ο προτεινόμενος ανιχνευτής δημιουργήθηκε ως αλγόριθμος δέντρου αποφάσεων (decision tree) J48, που είναι μια εφαρμογή του αλγορίθμου C4.5. Οι Krishnan et al. (2013) [38] παρουσίασαν έναν αλγόριθμο, ο οποίος εξήγαγε, από NXDomain responses, μοτίβα κίνησης traffic για μεμονωμένους υπολογιστές. Στη συνέχεια, η μελέτη των Raghuram et al. [40] βασιζόμενη στις διαφορετικές κατανομές χαρακτήρων που παρουσιάζουν τα γνήσια-νόμιμα domain names με τα αλγοριθμικά παραγόμενα, εισήγαγε ένα πιθανοτικό μοντέλο ανίχνευσης.

Με τη συνεχόμενη ανάπτυξη του Machine Learning τα τελευταία χρόνια, έχει παρατηρηθεί μια στροφή προς πιο μοντέρνες προσεγγίσεις για την ανίχνευση κίνησης botnet. Έχουν προταθεί πολλές τεχνικές που βασίζονται σε Machine Learning, για την εκπαίδευση ενός μοντέλου που λειτουργεί ως ταξινομητής ονομάτων ανάμεσα σε γνήσια και αλγοριθμικά παραγόμενα. Για την εκπαίδευσή τους χρησιμοποιούνται datasets που κατασκευάζονται εξάγοντας γλωσσικά χαρακτηριστικά από τα ονόματα που λαμβάνονται από NXDomain αποκρίσεις (πιθανά DGA ονόματα), καθώς και από γνήσια ονόματα. Το πλεονέκτημα αυτών των μοντέλων είναι η υποσχόμενη επεκτασιμότητά τους, καθώς και η ικανότητά τους να ανιχνεύουν πολύ γρήγορα, με σχετικά χαμηλότερους υπολογιστικούς πόρους από τις προηγούμενες έρευνες, μεμονωμένα ονόματα DGA, σε πραγματικό χρόνο. Αναλυτικότερα, οι Chen et al. (2018) [41], χρησιμοποίησαν ένα μοντέλο Support Vector Machine (SVM), για

την ανίχνευση των κακόβουλων ονομάτων, πετυχαίνοντας ένα ποσοστό αληθώς θετικών προβλέψεων (True Positive Rate) της τάξης του 95%, ενώ το FPR περιορίστηκε στο 1%. Για την εκπαίδευση του μοντέλου βασίστηκαν αποκλειστικά σε γλωσσικά χαρακτηριστικά των domain names, όπως το μήκος του ονόματος, τη συχνότητα εμφάνισης φωνηέντων και άλλα. Με τη σειρά τους, οι Huang et al. (2019) [42], εκπαίδευσαν ένα ενισχυμένο μοντέλο SVM, πετυχαίνοντας ακρίβεια περίπου 97,5%. Ενώ το συγκεκριμένο μοντέλο φαίνεται να έχει ικανοποιητική απόδοση, έχει το μειονέκτημα πως έχει μεγάλες απαιτήσεις σε υπολογιστικούς πόρους, σε σχέση με άλλες τεχνικές Machine Learning. Μεταξύ άλλων, στη μελέτη των Mac et al. (2017) [43] έχουν δοκιμαστεί και μοντέλα, όπως Hidden Markov Models και Bayesian Networks, χωρίς όμως να πετυχαίνουν καλή απόδοση (TPR<90%). Μία άλλη ενδιαφέρουσα προσέγγιση είναι η ανίχνευση κίνησης botnet με υλοποίηση Random Forest [44], για την ικανότητά του να συνδυάζει υψηλή ακρίβεια (περίπου 93,5%) και γενίκευση. Κατά συνέπεια, μπορεί να ανιχνεύει ένα σημαντικό ποσοστό ονομάτων που δημιουργούνται από DGAs, πάνω στους οποίους δεν έχει εκπαιδευτεί και να ανακαλύπτει έτσι νέες οικογένειες DGA. Όπως θα δούμε στη συνέχεια όμως, υπάρχουν συγκριτικά πολύ πιο αποδοτικές τεχνικές.

3.2.2 Ανίχνευση με Αυτόματη Εξαγωγή Χαρακτηριστικών (Featureless)

Όλες οι μελέτες που εξετάστηκαν έως τώρα παρουσιάζουν τα εξής κοινά σημεία:

- Βασίζονται σε χαρακτηριστικά που εξάγονται και δημιουργούνται από τον άνθρωπο.
- Εκπαιδεύονται και αξιολογούνται σε συνθετικά σύνολα δεδομένων (π.χ. Alexa και μια λίστα αλγοριθμικά παραγόμενων ονομάτων που λαμβάνονται από διαθέσιμα στο κοινό datasets), αντί να χρησιμοποιείται πραγματική κίνηση DNS.

Ωστόσο, η χρήση χαρακτηριστικών παραγόμενων από τον άνθρωπο, καθιστά τον ανιχνευτή ευάλωτο, καθώς είναι εύκολο για έναν botmaster να τροποποιήσει τον DGA που χρησιμοποιεί, ώστε να μην είναι πλέον εφικτή η διάκρισή του από τη νόμιμη κίνηση. Παράλληλα, τα datasets που βασίζονται σε τέτοιου είδους δεδομένα, έχουν παραχθεί για συγκεκριμένους γνωστούς DGAs, με αποτέλεσμα το εκπαιδευόμενο μοντέλο να μην γενικεύει σωστά για νεοεμφανιζόμενους DGAs που πιθανώς να μην έχουν τα ίδια χαρακτηριστικά. Τέλος, τα συνθετικά datasets, δεν αναπαριστούν πάντα με σωστό τρόπο την πραγματική κίνηση σε ένα δίκτυο. Επομένως, η αξιολόγηση που βασίζεται αποκλειστικά σε τέτοια σύνολα δεδομένων ενδέχεται να μην αντικατοπτρίζει το πόσο καλά αυτοί οι ανιχνευτές θα αποδίδουν σε μια πραγματική κατάσταση.

Σε αντίθεση με τις μεθόδους αυτές, η επιβλεπόμενη βαθιά μηχανική μάθηση (Supervised Deep Learning) έχει τη δυνατότητα αυτόματης εύρεσης και εξαγωγής σχετικών χαρακτηριστικών. Έχει αποδειχθεί επίσης σε σχετικές μελέτες, πως βελτιώνουν την ακρίβεια ανίχνευσης κίνησης botnet, σε σύγκριση με τις παραδοσιακές μεθόδους μηχανικής μάθησης που βασίζονται σε χαρακτηριστικά που εξάγονται από τον άνθρωπο. Το 2019, οι Spooren et al. [44], συνέκριναν έναν ανιχνευτή Random Forest (για τον οποίον έγινε λόγος και παραπάνω) με έναν ανιχνευτή Long Short Term Memory Network (LSTM) που βασίζεται σε Deep Learning και κατέληξαν στην ξεκάθαρη υπεροχή του δευτέρου, ο οποίος πέτυχε ακρίβεια 98,5% και FPR μικρότερο του 2%.

Πρώτοι οι Woodbridge et al. (2016) [45] έδειξαν, ότι ένα δίκτυο LSTM που χειρίζεται τα domain names σε επίπεδο χαρακτήρα, μπορεί να ανιχνεύει κακόβουλα ονόματα με εξαιρετική ακρίβεια. Παράλληλα απέδειξαν, ότι ένα μοντέλο Deep Learning μπορεί να εκτελεί προβλέψεις σχετικές με ανίχνευση κίνησης botnet, χωρίς τη χρήση κατασκευασμένων από τον άνθρωπο χαρακτηριστικών, σε πραγματικό χρόνο και με μεγάλη ακρίβεια. Παρόμοια αποτελέσματα πέτυχαν και οι Palak et al. (2020) [46], οι οποίο έχουν ακρίβεια 98%, χρησιμοποιώντας ένα LSTM. Εμπνευσμένοι από τη μελέτη των Woodbridge et al., οι Yu et al. (2017) [47] επιχείρησαν να βελτιώσουν τα αποτελέσματά τους, προτείνοντας ένα καινούριο μοντέλο LSTM και ένα Convolutional Neural Network (CNN), εκπαιδεύοντάς τα πάνω σε δεδομένα που προήλθαν από πραγματικό traffic. Από τη μεταξύ τους σύγκριση το πρώτο μοντέλο αποδείχθηκε ελάχιστα πιο αποδοτικό. Παράλληλα, τροποποιώντας τα μοντέλα τους, έδειξαν ότι εκτός από κατηγοριοποίηση ονομάτων ανάμεσα σε γνήσια και DGA, μπορούσαν να χρησιμοποιηθούν με μεγάλη επιτυχία και για κατηγοριοποίηση των κακόβουλων ονομάτων στη συγκεκριμένη οικογένεια malware στην οποία ανήκει. Έδωσαν μεγάλη βάση στη διαδικασία της εκπαίδευσης, ώστε αυτή να γίνεται με δεδομένα πραγματικού traffic και επιχείρησαν να μειώσουν στο ελάχιστο το FPR, θέτοντας κατάλληλα κατώφλια εξόδου. Από την έρευνα αυτή προέκυψαν οι εξής περιορισμοί σχετικά με την απόδοση του Deep Learning, στην ανίχνευση κίνησης DGA-based botnet:

- Τα μοντέλα που προτάθηκαν προς την κατεύθυνση του Deep Learning, πάσχουν από σχετικά υψηλά ποσοστά FPR (περίπου 2%) και η επιβολή κατωφλίων για τη μείωσή τους, επιφέρει μετά από ένα σημείο εκθετική πτώση στην ακρίβεια.

- Η κατηγοριοποίηση κακόβουλων ονομάτων στις εκάστοτε οικογένειες DGA, στις οποίες ανήκουν, επηρεάζεται σημαντικά όταν κατά την εκπαίδευση το dataset δεν είναι ισορροπημένο.

- Τα προτεινόμενα μοντέλα αδυνατούν να ανιχνεύσουν αποτελεσματικά κάποιες οικογένεις DGA, ιδίως τις wordlist-based, για τις οποίες η ακρίβεια μπορεί να προσεγγίζει ακόμα και το 0%, αν στο training-set δεν συμπεριλάβουμε ανάλογα ονόματα.

Η αδυναμία ανίχνευσης wordlist-based DGAs αποτέλεσε αντικείμενο προβληματισμού πολλών πρόσφατων ερευνών. Οι κακόβουλοι υιοθέτησαν παρόμοιους DGAs, για να μοιάζουν τα ονόματα των botnets τους όσο γίνεται πιο γνήσια. Οι Liu et al. (2018) [48] σχεδίασαν έναν νέο αλγόριθμο για την καταπολέμηση αυτού του προβλήματος. Ανέλυσαν χαρακτηριστικά όπως η συχνότητα εμφάνισης συγκεκριμένων λέξεων σε έναν αριθμό ονομάτων, τη συσχέτιση των λέξεων και εξήγαγαν 16-διάστατα χαρακτηριστικά. Χρησιμοποιήθηκαν τέσσερα διαδεδομένα μοντέλα Machine Learning. Ο Random Forest είχε την καλύτερη απόδοση. Ύστερα, τα συγκριτικά αποτελέσματα δόθηκαν ξεχωριστά σε ένα CNN και σε ένα LSTM μοντέλο, για ανίχνευση σε δεύτερο στάδιο. Τα πειραματικά αποτελέσματα έδειξαν, ότι ο προτεινόμενος αλγόριθμος πέτυχε ακρίβεια 70% για wordlist-based DGAs. Ωστόσο, το σύστημα αυτό, για λόγους που έχουν προαναφερθεί στις προηγούμενες υποενότητες του κεφαλαίου, πάσχει από υψηλή καθυστέρηση, υψηλές απαιτήσεις σε υπολογιστικούς πόρους και μειωμένη ικανότητα ανακάλυψης καινούριων DGA. Την ίδια χρονιά οι Koh et al. [49] πρότειναν μια μέθοδο που

συνδυάζει ένα word-embedding μοντέλο ELMo με (logistic regression), πετυχαίνοντας έτσι ακρίβεια 91% για τον matsnu DGA. Τέλος, το 2019, οι Curtin et al. [50] επινόησαν τη βαθμολογία smashword για τη μέτρηση της ομοιότητας μεταξύ ονομάτων που δημιουργήθηκαν από wordlist-based DGAs και ενός συνόλου αγγλικών λέξεων. Στη συνέχεια, χρησιμοποίησαν ένα μοντέλο LSTM για τον εντοπισμό πιθανής κακόβουλης κίνησης.

3.3 Συνεισφορά της Παρούσας Διπλωματικής

Η συνεισφορά της διπλωματικής αυτής έγγειται στα παρακάτω σημεία:

1. Χρησιμοποιούμε τα μοντέλα που προτάθηκαν από τους Yu et al. [47] και τα βελτιώνουμε, ενώ παράλληλα μελετάμε εκ νέου το trade-off ανάμεσα σε μείωση του FPR και TPR. Εξετάζουμε επίσης την απόδοση ενός Bidirectional LSTM μοντέλου που βασίζεται στο απλό LSTM των Yu et al..

2. Προκειμένου να μελετηθούν οι ιδιομορφίες των wordlist-based DGAs και το πρόβλημα της μειωμένης ικανότητας των μοντέλων Deep Learning να εντοπίζουν τα σχετικά domain names, εκπαιδεύουμε τα μοντέλα μας πάνω σε δύο διαφορετικά datasets. Ένα που δεν συμπεριλαμβάνει wordlist-based DGAs και ένα δεύτερο που συμπεριλαμβάνει μεταξύ άλλων και αυτή την ιδιαίτερη ομάδα αλγοριθμικά παραγόμενων domain names.

3. Αναπτύσσουμε ένα πειραματικό σύστημα, προκειμένου να υλοποιήσουμε τους προηγούμενους ανιχνευτές σε ένα περιβάλλον Federated Learning, ώστε μέσα από μια συνεργατική διαδικασία εκπαίδευσης που σέβεται την ιδιωτικότητα των δεδομένων, να ενισχύσουμε και να διατηρήσουμε όσο γίνεται πιο ενημερωμένα τα μοντέλα μας.

Κεφάλαιο 4

Μεθοδολογία

Στο κεφάλαιο αυτό παρουσιάζεται η μεθοδολογία που ακολουθείται και τα επιμέρους στοιχεία που χρησιμοποιούνται για την ανάπτυξη του συστήματος ανίχνευσης κίνησης botnet που προτείνεται. Η προσομοίωση εφαρμογών Federated Learning συμβάλλει σημαντικά στην επιτάχυνση της σχετικής έρευνας πάνω σε αυτόν τον καινούριο τομέα. Οι προσομοιώσεις Federated Learning απαιτούν την αντιμετώπιση πολλαπλών ζητημάτων που δεν προκύπτουν στο παραδοσιακό Distributed Deep Learning. Για παράδειγμα, την αποτελεσματική επεξεργασία διαμερισμένων dataset, με τους υπολογισμούς να εκτελούνται σε διαφορετικές συσκευές, καθεμία με μεταβλητή ποσότητα ετερογενών δεδομένων. Η έρευνα FL απαιτεί επίσης διαφορετικές μετρήσεις, όπως ο αριθμός των byte που μεταφορτώνονται, καθώς και η δυνατότητα προσομοίωσης ζητημάτων, όπως η άφιξη διαφορετικών clients σε τυχαία χρονικά διαστήματα. Οι ανάγκες αυτές έχουν οδηγήσει στην ανάπτυξη πολλών frameworks για Federated Learning. Τα πιο γνωστά που λαμβάνουν και τη μεγαλύτερη αποδοχή από την επιστημονική κοινότητα είναι:

- PySyft [51]: Μια βιβλιοθήκη της Python για Federated Learning, με πολλές δυνατότητες προσομοίωσης, που παράλληλα παρέχει δυνατότητες για differential privacy και Secure Multiparty Computation [52]. Το PySyft αποτελεί επέκταση του PyTorch framework.

- TensorFlow Federated [53]: Μια βιβλιοθήκη της Python, που λειτουργεί ως extension του TensorFlow framework και παρέχει ένα ευέλικτο περιβάλλον προσομοίωσης για Federated Learning.

- FATE [54]: Βιβλιοθήκη της Python που στοχεύει περισσότερο στην ανάπτυξη εργαλείων για την πρακτική εφαρμογή του Federated Learning, παρά για προσομοίωση. Και εδώ παρέχονται επιπλέον δυνατότητες για Secure Multiparty Computation.

Στη συγκεκριμένη διπλωματική έχουμε επιλέξει το Pysyft για τους εξής λόγους. Αρχικά, το PySyft έχει τη μεγαλύτερη και πιο ενεργή κοινότητα από τα προαναφερθέντα frameworks, με αποτέλεσμα να εξελίσσεται και να βελτιώνεται με μεγαλύτερη ταχύτητα. Υπάρχει επαρκές documentation, σε αντίθεση με το FATE, που βοηθά στην καταννόηση των εννοιών και στην εξοικίωση με το εν λόγω εργαλείο. Ο κώδικας είναι απλός και καταννοητός, χωρίς όμως να γίνονται εκπτώσεις στη λειτουργικότητα και στην απόδοση. Αντιθέτως, όπως αναφέραμε,

δίνονται και επιπλέον δυνατότητες για ασφαλέστερες εφαρμογές Federated Learning, μέσω differential privacy και Secure Multiparty Computation, οι οποίες αν και δεν χρησιμοποιούνται στην παρούσα διπλωματική, θα μπορούσαν να φανούν χρήσιμες για μελλοντικές επεκτάσεις του συστήματος που προτείνουμε. Επιπλέον, αν και στη παρούσα φάση το PySyft μπορεί να χρησιμοποιηθεί επαρκώς μόνο για προσομοιώσεις, σύντομα θα μπορεί να χρησιμοποιηθεί και σε πραγματικές εφαρμογές, συνδυαστικά με το PyGrid [55]. Ουσιαστικά το PyGrid χρησιμοποιείται για την κατασκευή ενός δικτύου από clients που μπορούν να εκπαιδεύσουν ένα μοντέλο συνεργατικά, με Federated Learning χρησιμοποιώντας το PySyft. Πρέπει να αναφέρουμε πως όλα τα διαθέσιμα frameworks που έχουν αναπτυχθεί για Federated Learning βρίσκονται ακόμα σε αρχικό στάδιο και επιδέχονται πολλών βελτιώσεων και διορθώσεων. Για παράδειγμα, για το PySyft έχουν αναφερθεί στην κοινότητα bugs που αφορούν time overhead κατά την εκπαίδευση και πιθανά memory leaks. Παρόλα αυτά, επιλέξαμε το PySyft framework αφενός για την ήδη ευρεία συλλογή δυνατοτήτων που προσφέρει και αφετέρου για τις πολλά υποσχόμενες προοπτικές του.

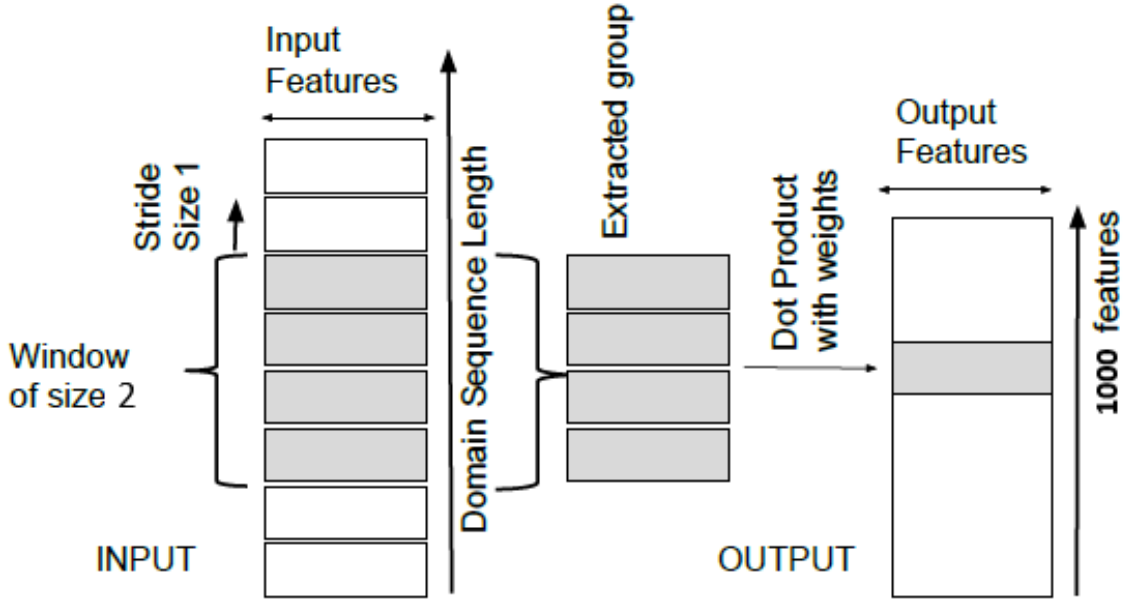
4.1 Υλοποίηση Deep Learning Μοντέλων-Ανιχνευτών

Όπως προαναφέρθηκε στο Κεφάλαιο 3, εξετάζουμε τα μοντέλα για Deep Learning των Yu et al. [47] (CNN, LSTM) και δοκιμάζουμε πιθανές βελτιώσεις πάνω σε αυτά. Επιπλέον κατασκευάζουμε ένα τρίτο μοντέλο (Bidirectional-LSTM), με βάση το LSTM που πρότειναν. Δίνοντας μεγάλη έμφαση στο κριτήριο της ακρίβειας και του χαμηλού FPR, στοχεύουμε στην εκπαίδευση των τριών μοντέλων για τον αποτελεσματικό εντοπισμό διαφορετικών οικογενειών DGA και την πιθανή ανακάλυψη νεοεμφανιζόμενων οικογενειών, για τις οποίες δεν υπάρχουν προηγούμενες πληροφορίες. Και τα τρία μοντέλα λαμβάνουν ως είσοδο μεμονωμένα domain names και ως έξοδο παράγουν την προβλεπόμενη κλάση στην οποία ανήκει κάθε όνομα, δηλαδή legit ή DGA. Τα μοντέλα έχουν αναπτυχθεί σε Python (version 3.6), με τη χρήση του PyTorch framework (version 1.4.0).

4.1.1 Convolutional Neural Network (CNN)

Τα CNN έχουν εφαρμοστεί με μεγάλη επιτυχία στη βιομηχανία για την επεξεργασία εικόνας και video. Τα δίκτυα βαθιάς μηχανικής μάθησης CNN χρησιμοποιούν φίλτρα για τον εντοπισμό μοτίβων που είναι σημαντικά για τη σωστή πρόβλεψη. Τα εν λόγω φίλτρα ονομάζονται kernels (πυρήνες) και βελτιστοποιούνται κατά τη διάρκεια του backpropagation. Ένα διαισθητικό παράδειγμα στην επεξεργασία εικόνας αφορά ένα φίλτρο που ανιχνεύει κάθετες ακμές. Γενικά, η εφαρμογή τους προτείνεται σε περιπτώσεις όπου τα δεδομένα εισόδου μπορούν να αναλυθούν σε μια grid-like τοπολογία, όπως για παράδειγμα το σύνολο των pixel μιας εικόνας. Ωστόσο, φαίνεται πως αποδίδουν καλά και στην επεξεργασία κειμένου, αν αντί για ένα δισδιάστατο Conv2D στρώμα που χρησιμοποιείται παραδοσιακά για επεξεργασία εικόνας, εφαρμόσουμε ένα μονοδιάστατο Conv1D συνελικτικό (convolutional) στρώμα. Ο εντοπισμός κακόβουλων domain names αποτελεί ένα πρόβλημα επεξεργασίας κειμένου και στην περίπτωση αυτή οι kernels ανιχνεύουν υπακολουθίες χαρακτήρων, ή N-grams, μέσα σε κάθε domain name. Δηλαδή τα domain names αντιμετωπίζονται ως ακολουθίες χαρα-

κτήρων. Η κύρια λειτουργία των CNN είναι η συνέλιξη που εκτελείται, με την εφαρμογή ενός κυλιόμενου παραθύρου κατά μήκος όλης της ακολουθίας της συμβολοσειράς του domain name. Τον ρόλο του παραθύρου παίζουν οι kernels, το μήκος των οποίων στο δικό μας μοντέλο είναι 2. Επομένως το μοντέλο μας εξάγει τα χαρακτηριστικά ενός domain name, υπολογίζοντας τα αντίστοιχα bigrams. Σε κάθε μετακίνηση ενός kernel, εκτελείται η πράξη του εσωτερικού γινομένου μεταξύ του ίδιου του kernel και της αντίστοιχης επικαλυπτόμενης υπακολουθίας και το αποτέλεσμα είναι η έξοδος που αναπαριστά μια ενεργοποίηση χάρτη, δηλαδή μια περιοχή της συμβολοσειράς όπου ανακαλύφθηκε ένα καινούριο χαρακτηριστικό. Το βήμα κύλισης του kernel αναφέρεται ως stride και στην περίπτωσή μας ισούται με 1. Στην Εικόνα 4.1 δίνεται μια γραφική αναπαράσταση της διαδικασίας εξαγωγής χαρακτηριστικών από ένα domain name μέσω του συνελικτικού στρώματος Conv1D που χρησιμοποιείται στο μοντέλο μας.



Εικόνα 4.1: *Εξαγωγή Χαρακτηριστικών με Μονοδιάστατο Συνελικτικό Στρώμα Conv1D*

Η αρχιτεκτονική του CNN μοντέλου που κατασκευάσαμε σε αυτή τη διπλωματική μπορεί να αναλυθεί σε 4 επίπεδα:

1. Embedding Layer
2. Conv1D Layer
3. Dense Layer 1 (Linear Regression)
4. Dense Layer 2 (Linear Regression)

Παρατηρούμε ότι εκτός από το μονοδιάστατο συνελικτικό στρώμα που εξετάσαμε, το μοντέλο μας δομείται επιπλέον από ένα Embedding και δύο Dense Layers. Το Embedding Layer χρησιμοποιείται για την χαρτογράφηση κάθε ενός από τα 38 επιτρεπόμενα σύμβολα που μπορούν να χρησιμοποιηθούν σε ένα domain name, σε ένα διάνυσμα 128 διαστάσεων. Η

χρήση του στρώματος αυτού βοηθάει το μοντέλο να μάθει τα κατάλληλα χαρακτηριστικά που αντιπροσωπεύουν κάθε επιτρεπόμενο σύμβολο. Σημειώνεται ότι τα 38 σύμβολα που χρησιμοποιούνται, αποτελούνται από τα 26 πεζά γράμματα του λατινικού αλφαβήτου, τα 10 αριθμητικά ψηφία (0-9), την παύλα και από ένα τελευταίο που αντιπροσωπεύει το padding. Το πρώτο Dense Layer λαμβάνει ως είσοδο τα εξαγόμενα χαρακτηριστικά που παράγονται από το Conv1D Layer και υπολογίζει τη πιθανότητα να ανήκει ένα domain name σε μία από τις δύο κλάσεις με βάση αυτά. Το δεύτερο Dense Layer χρησιμοποιείται για την εξαγωγή της πιθανότητας αυτής. Η συνάρτηση ενεργοποίησης σε όλα τα στρώματα εκτός του τελευταίου είναι η Rectified Linear Unit (ReLu). Στο στρώμα εξόδου η συνάρτηση ενεργοποίησης είναι η σιγμοειδής (sigmoid). Επιπλέον, πρέπει να αναφερθεί ότι μετά το Conv1D Layer εφαρμόζεται 50% Dropout, ώστε να αποφύγουμε τυχόν overfit. Ο optimizer που χρησιμοποιούμε είναι ο Adam, με learning rate 0.001.

Αλγοριθμος 4.1: *CNN MODEL in PYTORCH FRAMEWORK*

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.word_embeddings = nn.Embedding(num_embeddings=38,
        embedding_dim=128)
        self.conv1 = torch.nn.Conv1d(in_channels=128,
             out_channels=1000, kernel_size=2, stride=1)
        self.dropout = nn.Dropout(0.5)
        self.linear1 = nn.Linear(1000*46, 100)
        self.out = nn.Linear(100, 1)

    def forward(self, x):
        embedded = self.word_embeddings(x).permute(0, 2, 1)
        x = F.relu(self.conv1(embedded))
        x = self.dropout(x)
        x = x.view(-1, 1000*46)
        x = F.relu(self.linear1(x))
        x = F.sigmoid(self.out(x))
        return x

model = Net()
optimizer=optim.Adam(params=model.parameters(), lr=0.001)
scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer,
     'max', verbose=True, patience=2, factor=0.5)
loss = F.binary_cross_entropy()
```

Το μοντέλο που προτείνουμε σε σχέση με των Yu et al. διαφέρει σε δύο σημεία. Κατά πρώτον, χρησιμοποιούμε ένα vocabulary μεγέθους 38 χαρακτήρων, αντί για 256 (σύνολο ASCII χαρακτήρων), ώστε να περιοριστούμε μόνο στα επιτρεπόμενα σύμβολα και να μειώσουμε την επιβάρυνση σε μνήμη. Η δεύτερη διαφορά έγγειται στον τρόπο με τον οποίο ε-

λαττώνεται το learning rate κατά τη διαδικασία της εκπαίδευσης. Αναλυτικότερα, το learning rate παραμένει σταθερό σε κάθε γύρο εκπαίδευσης, μέχρι η ακρίβεια του εκπαιδευούμενου μοντέλου να μην αυξηθεί για περισσότερους από 2 συνεχόμενους γύρους εκπαίδευσης. Τότε μειώνουμε το learning rate στο μισό. Η μέθοδος αυτή οδήγησε σε μια ελάχιστη (0.15%), αλλά επιθυμητή αύξηση στην ακρίβεια του τελικού μοντέλου, σε σχέση με το σταθερά ελλατούμενο learning rate του μοντέλου των Yu et al.. Ακολουθεί η παραμετρική ανάλυση του μοντέλου μας και οι απαιτήσεις του σε μνήμη.

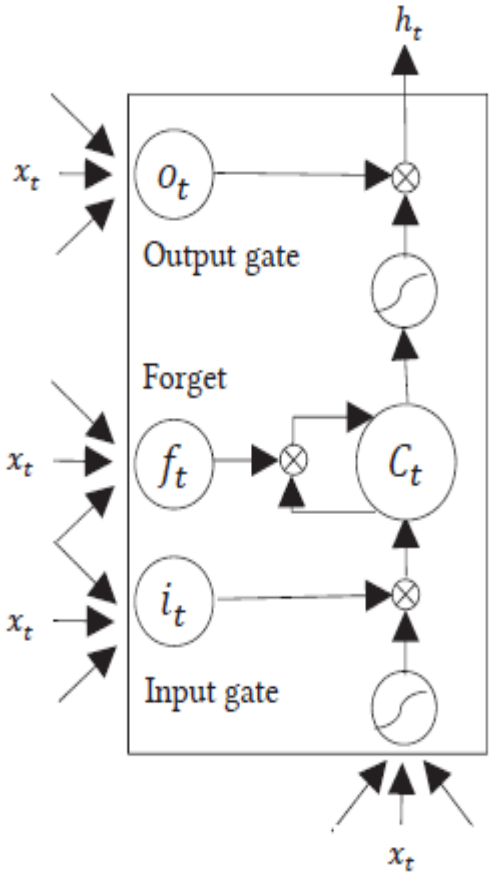
```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
         Embedding-1              [-1, 47, 128]           4,992
            Conv1d-2             [-1, 1000, 46]         257,000
           Dropout-3             [-1, 1000, 46]               0
            Linear-4                  [-1, 100]       4,600,100
            Linear-5                    [-1, 1]             101
               Net-6                    [-1, 1]               0
================================================================
Total params: 4,862,193
Trainable params: 4,862,193
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): -0.00
Forward/backward pass size (MB): 0.75
Params size (MB): 18.55
Estimated Total Size (MB): 19.30
----------------------------------------------------------------
```

Εικόνα 4.2: *CNN model: Παραμετρική Ανάλυση-Memory Overhead*

4.1.2 Long Short-Term Memory Network (LSTM)

Ένα μοντέλο LSTM αποτελεί ειδική περίπτωση ενός Recursive Neural Network (RNN) μοντέλου. Τα RNN έχουν χρησιμοποιηθεί στο παρελθόν για εφαρμογές επεξεργασίας κειμένου, για την ανίχνευση των εξαρτήσεων μεταξύ μεμονωμένων συμβόλων μιας ακολουθίας, που στην περίπτωσή μας είναι μία λέξη (domain name). Σε ένα RNN η έξοδος είναι συνάρτηση τόσο της ισχύουσας εισόδου, όσο και των προηγούμενων ενεργοποιήσεων του RNN που προέκυψαν από παρελθοντικές εισόδους, δηλαδή από προηγούμενα σύμβολα της ακολουθίας. Δημιουργείται έτσι ένας χάρτης συμφραζομένων για κάθε ακολουθία. Ωστόσο, εξαιτίας των μακρών αλυσιδωτών αναδρομών που προκύπτουν σε ένα domain name μεγάλου μήκους, η έξοδος που παράγεται σε ένα παραδοσιακό RNN μπορεί είτε να μειώνεται συνέχως μέχρι που τείνει να εξαφανιστεί (vanishing gradient problem), ή να αυξάνεται συνεχώς προκαλώντας το γνωστό πρόβλημα exploding gradient. Το γεγονός αυτό καθιστά την εκμάθηση μακροπρόθεσμων αλληλοεξαρτήσεων σε ένα domain name δύσκολο [45].

Το πρόβλημα που αναφέρεται στη βιβλιογραφία ως vanishing gradient problem, απο-



Εικόνα 4.3: *LSTM Memory Block*

τέλεσε σημείο έμπνευσης για την ανάπτυξη των LSTM. Ένα LSTM block διαφέρει σε σχέση με ένα κλασσικό RNN block στο γεγονός πως εκ κατασκευής περιέχει επιπλέον ένα κελί μνήμης, στο οποίο μπορεί να διαβάζεται, να γράφεται, ή και να γίνεται επαναφορά (reset) μιας κατάστασης, μέσω προγραμματιζόμενων λογικών πυλών. Το κελί μνήμης είναι συνδεδεμένο με έναν αυτο-αναδρομικό βρόχο, ο οποίος επιτρέπει την αποθήκευση μιας κατάστασης εξόδου, μεταξύ δύο διαφορετικών χρονικών βημάτων (δύο διαδοχικά σύμβολα). Όμως η κατάσταση αυτή μπορεί να μεταβάλλεται κατά τη διάρκεια εκτέλεσης, μέσω μιας ειδικής πύλης εισόδου (input gate), η οποία πολλαπλασιάζει την είσοδο με έναν αριθμό από 0 έως 1 (sigmoid activation), ή από -1 έως 1 (tanh activation). Ομοίως, η τιμή της κατάστασης πολλαπλασιάζεται εκ νέου μέσω ενός αναδρομικά συνδεδεμένου forget gate, με μια τιμή από 0 έως 1 (sigmoid activation). Έτσι, αν για παράδειγμα η πύλη εισόδου πολλαπλασιάσει την είσοδο με 0 και η πύλη επαναφοράς forget gate πολλαπλασιάσει την τιμή του κελιού με 1, τότε η αποθηκευμένη κατάσταση δεν μεταβάλλεται και παραμένει αυτούσια. Από την άλλη πλευρά, αν η πύλη εισόδου πολλαπλασιάσει την είσοδο με 1 και η πύλη επαναφοράς πολλαπλασιάσει την τιμή του κελιού με 0, τότε στο κελί γράφεται εκ νέου η ισχύουσα είσοδος. Αν και οι δύο πύλες πολλαπλασιάσουν τις αντίστοιχες τιμές με 0, η τιμή του κελιού μηδενίζεται. Τελικά, μια πύλη εξόδου πολλαπλασιάζει την τιμή του κελιού με μια τιμή ανάλογη της συνεισφοράς κατάστασης του συγκεκριμένου κελιού και διαδίδει το αποτέλεσμα στο επόμενο επίπεδο (αν υπάρχει) του LSTM μοντέλου. Στη μελέτη των Mac et al. [43] περιγράφεται μεταξύ άλλων,

μέσω των παρακάτω εξισώσεων, πως προκύπτει η έξοδος του LSTM block της Εικόνας 4.3. Οι sigmoid και tanh ενεργοποιήσεις αναπαριστούνται αντίστοιχα με σ_g και σ_h.

$$f_t = \sigma_g(W_{fx}X_t + W_{fh}h_{t-1} + b_f) \tag{4.1}$$

$$i_t = \sigma_g(W_{ix}X_t + W_{ih}h_{t-1} + b_i) \tag{4.2}$$

$$o_t = \sigma_g(W_{ox}X_t + W_{oh}h_{t-1} + b_o) \tag{4.3}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \sigma_h(W_{cx}X_t + W_{ch}h_{t-1} + b_c) \tag{4.4}$$

$$h_t = o_t \odot \sigma_h(c_t) \tag{4.5}$$

Συνεπώς, η δομή του κελιού σε ένα LSTM memory block επιτρέπει την πρόσβαση και την αποθήκευση ενδιάμμεσων καταστάσεων σε περιπτώσεις μεγάλων ακολουθιών, αντιμετωπίζοντας το πρόβλημα του vanishing gradient. Στην περίπτωσή μας, το κελί χρησιμοποιείται για την προσωρινή αποθήκευση συνδυασμών χαρακτήρων ενός domain name για την εξαγωγή χρήσιμων χαρακτηριστικών που βοηθούν στην κατηγοριοποίηση των ονομάτων σε legit ή DGA. Στο πρόβλημά μας, όπως θα δούμε στο κεφάλαιο 6, το μοντέλο LSTM εμφανίζει μεγαλύτερη ακρίβεια σε σχέση με το CNN. Αυτό είναι αναμενόμενο, τόσο από τη σχετική βιβλιογραφία, όσο και από την ικανότητα των LSTM μοντέλων να γενικεύουν την ιδέα εξαγωγής χαρακτηριστικών μέσω N-grams. Πιο συγκεκριμένα, το LSTM μοντέλο μας ανιχνεύει τις εξαρτήσεις μεταξύ διαφόρων υποσυνόλων συμβόλων (bigrams, 3-grams, 4-grams,...) ενός domain name ταυτόχρονα. Επιπλέον, τα σύμβολα αυτά μπορεί να είναι διαδοχικά, ή να βρίσκονται σε διαφορετικές θέσεις μέσα στην ακολουθία. Επομένως, ένα LSTM εμφανίζει αυξημένη ευελιξία ως προς την ανίχνευση και εξαγωγή χαρακτηριστικών με βάση τις εξαρτήσεις των συμβόλων μιας ακολουθίας και για τον λόγο αυτό συνιστάται η χρήση τους για προβλήματα επεξεργασίας κειμένου, όπως είναι και η ανίχνευση DGA ονομάτων.

Η αρχιτεκτονική του LSTM μοντέλου που κατασκευάσαμε στη διπλωματική αυτή, εμπνεόμενοι από το αντίστοιχο των Yu et al., αποτελείται από 4 επίπεδα:

1. Embedding Layer
2. LSTM Layer
3. Dense Layer 1 (Linear Regression)
4. Dense Layer 2 (Linear Regression)

Το embedding layer κατασκευάζεται με τον ίδιο ακριβώς τρόπο με το αντίστοιχο του CNN, ενώ επίσης τα δύο τελευταία επίπεδα εξυπηρετούν τον ίδιο σκοπό. Το δεύτερο επίπεδο που είναι και το πιο σημαντικό, αποτελείται από 128 LSTM blocks, τα οποία λαμβάνουν ως είσοδο τις 128-διάστατες ακολουθίες που εξάγονται από το embedding layer και καθένα από

Αλγοριθμος 4.2: *LSTM MODEL in PYTORCH FRAMEWORK*

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.word_embeddings = nn.Embedding(num_embeddings=38,
        embedding_dim=128)
        self.lstm = nn.LSTM(input_size=128, hidden_size=128,
        num_layers=1, batch_first=True)
        self.dropout = nn.Dropout(0.5)
        self.linear1 = nn.Linear(47*128, 100)
        self.linear2 = nn.Linear(100, 1)

    def init_hidden(self):
        return (torch.zeros(self.num_layers, self.batch_size,
        self.hidden_dim),
                torch.zeros(self.num_layers, self.batch_size,
                self.hidden_dim))

    def forward(self, x):
        embedded = self.word_embeddings(x)
        x, (ht, ct) = self.lstm(embedded)
        x = x.contiguous().view(-1, 47*128)
        x = F.relu(x)
        x = self.dropout(x)
        x = F.relu(self.linear1(x))
        x = F.sigmoid(self.linear2(x))
        return x

model = Net()
optimizer=optim.Adam(params=model.parameters(), lr=0.001)
scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer,
     'max', verbose=True, patience=2, factor=0.5)
loss = F.binary_cross_entropy()
```

αυτά παράγει ένα χαρακτηριστικό. Άρα η έξοδος του LSTM είναι επίσης μία 128-διάστατη ακολουθία για κάθε σύμβολο του domain name. Μετά το LSTM layer εφαρμόζεται 50% Dropout, για την αποφυγή του overfit. Ο optimizer που χρησιμοποιούμε είναι και εδώ ο Adam, με learning rate 0.001. Η συνάρτηση ενεργοποίησης είναι για όλα τα επίπεδα, εκτός του τελευταίου η ReLU και για το τελευταίο είναι η σιγμοειδής sigmoid. Όμοια με το CNN μοντέλο το learning rate μειώνεται με τον ίδιο τρόπο, δηλαδή με υποδιπλασιασμό μετά από 3 συνεχόμενους γύρους μη βελτίωσης της ακρίβειας του μοντέλου. Οι διαφορές με το αντίστοιχο LSTM μοντέλο των Yu et al. είναι οι ίδιες που περιγράφηκαν στην προηγούμενη

υποενότητα. Παραθέτουμε εδώ 4.2 τον κώδικα του μοντέλου LSTM που αναπτύξαμε.

Επιπροσθέτως, παραθέτουμε επίσης την παραμετρική ανάλυση και τις απαιτήσεις σε μνήμη του LSTM μοντέλου μας. Εύκολα παρατηρούμε ότι συγκριτικά με το CNN, το LSTM καταναλώνει σημντικά λιγότερη μνήμη. Παράλληλα έχουμε λιγότερες παραμέτρους προς βελτιστοποίηση, γεγονός που καθιστά τη διαδικασία της εκπαίδευσης πολύ πιο γρήγορη. Σημειώνουμε τις παρατηρήσεις αυτές, ώστε σε επόμενο κεφάλαιο να τις συμπεριλάβουμε στην τελική σύγκριση των τριών μοντέλων που εξετάζουμε.

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
         Embedding-1           [-1, 47, 128]           4,992
              LSTM-2           [-1, 47, 128]         132096
           Dropout-3           [-1, 47, 128]              0
            Linear-4                [-1, 100]         601700
            Linear-5                  [-1, 1]            101
               Net-6                  [-1, 1]              0
================================================================
Total params: 738889
Trainable params: 738889
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): -0.00
Forward/backward pass size (MB): 0.05
Params size (MB): 2.82
Estimated Total Size (MB): 2.87
----------------------------------------------------------------
```

Εικόνα 4.4: *LSTM model: Παραμετρική Ανάλυση-Memory Overhead*

4.1.3 Bidirectional Long Short-Term Memory Network (Bi-LSTM)

Ένα Bidirectional LSTM αποτελεί μια επέκταση του παραδοσιακού LSTM. Αποτελείται ουσιαστικά από 2 LSTM σε στοίβα, ένα forward και ένα backward [43]. Έχει παρατηρηθεί ότι σε σχέση με το συμβατικό LSTM, επιτυγχάνει υψηλότερη ακρίβεια καθώς και γενίκευση, όταν χρησιμοποιείται σε προβλήματα κατηγοριοποίησης ακολουθιών όπως είναι ο εντοπισμός DGA domain names. Η βελτιωμένη απόδοση οφείλεται στο επιπλέον (backward) LSTM που συνιστά ένα Bidirectional LSTM. Ένα LSTM υπολογίζει τις εξαρτήσεις μιας ακολοθίας, χρησιμοποιώντας τις παρελθοντικές καταστάσεις που έχουν αποθηκευτεί στο κελί, αφού κάθε φορά λαμβάνει την επόμενη (μελλοντική) είσοδο (forward layer). Από την άλλη πλευρά, ένα Bi-LSTM προσπελαύνει την ακολουθία και από τις δύο κατευθύνσεις ταυτόχρονα (forward και backward), οπότε υπολογίζει τις εξαρτήσεις χρησιμοποιώντας τόσο τις παρελθοντικές, όσο και τις μελλοντικές καταστάσεις του κελιού σε σχέση με το απλό LSTM [43]. Με αφορμή τα πλεονεκτήματα του μοντέλου αυτού, εξετάζουμε ένα Bi-LSTM, βασισμένο στο LSTM μοντέλο που περιγράψαμε στην προηγούμενη υποενότητα. Βέβαια η αύξηση της ακρίβειας του

Bi-LSTM επιφέρει αυξημένες καθυστερήσεις στην εκπαίδευση και επιβάρυνση στη μνήμη. Παρακάτω φαίνεται ο κώδικας του μοντέλου σε PyTorch. Βλέπουμε ότι η μόνη διαφορά σε σχέση με το LSTM μοντέλο είναι το επιπλέον backward επίπεδο.

Αλγοριθμος 4.3: *Bi-LSTM MODEL in PYTORCH FRAMEWORK*

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.word_embeddings = nn.Embeddingnum_embeddings=38,
        embedding_dim=128)
        self.lstm = nn.LSTM(input_size=128, hidden_size=128,
        num_layers=1, batch_first=True, bidirectional=True)
        self.dropout = nn.Dropout(0.5)
        self.linear1 = nn.Linear(47*256, 100)
        self.linear2 = nn.Linear(100, 1)

    def init_hidden(self):
        return (torch.zeros(self.num_layers, self.batch_size,
        self.hidden_dim),
                torch.zeros(self.num_layers, self.batch_size,
                self.hidden_dim))

    def forward(self, x):
        embedded = self.word_embeddings(x)
        x, (ht, ct) = self.lstm(embedded)
        x = x.contiguous().view(-1, 47*256)
        x = F.relu(x)
        x = self.dropout(x)
        x = F.relu(self.linear1(x))
        x = F.sigmoid(self.linear2(x))
        return x

model = Net()
optimizer=optim.Adam(params=model.parameters(), lr=0.001)
scheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer,
     'max', verbose=True, patience=2, factor=0.5)
loss = F.binary_cross_entropy()
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
         Embedding-1           [-1, 47, 128]           4,992
              LSTM-2           [-1, 47, 128]         132096
              LSTM-3           [-1, 47, 128]         132096
           Dropout-4           [-1, 47, 128]              0
            Linear-5               [-1, 100]        1203300
            Linear-6                 [-1, 1]            101
               Net-7                 [-1, 1]              0
================================================================
Total params: 1,472,585
Trainable params: 1,472,585
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): -0.00
Forward/backward pass size (MB): 0.07
Params size (MB): 5.62
Estimated Total Size (MB): 5.69
----------------------------------------------------------------
```

Εικόνα 4.5: *Bi-LSTM model: Παραμετρική Ανάλυση-Memory Overhead*

4.1.4 Υλοποίηση Ανιχνευτών

Καθένα από τα τρία μοντέλα Deep Learning που περιγράψαμε μπορεί να χρησιμοποιηθεί ως ένας ανεξάρτητος ταξινομητής για domain names, κατηγοριοποιώντας τα ανάμεσα σε legit και DGA. Οι προβλέψεις γίνονται σε επίπεδο μεμονωμένου domain name. Κατά την πρόβλεψη, κάθε χαρακτήρας του domain name μετατρέπεται σε πεζό, αφού ένας DNS είναι case insensitive. Στην πορεία γίνεται η χαρτογράφηση κάθε συμβόλου, μέσω του embedding layer και με βάση τα χαρακτηριστικά που εξάγονται, υπολογίζεται τελικά στην έξοδο του μοντέλου η ανάλογη πιθανότητα, να ανήκει ένα domain name σε μία από τις δύο κατηγορίες. Αν θέλουμε να μειώσουμε το FPR (False Positive Rate), δηλαδή το ποσοστό των legit ονομάτων που κατηγοριοποιούνται λανθασμένα ως DGA, αυξάνουμε το κατώφλι της εξόδου. Δηλαδή απαιτούμε από τον ταξινομητή μας να κατηγοριοποιήσει ένα domain name ως DGA (κακόβουλο), μόνο αν η σχετική πιθανότητα είναι επαρκώς υψηλή (υψηλότερη από το κατώφλι που ορίζουμε). Η ελαχιστοποίηση του FPR είναι πολύ σημαντική αν ο ανιχνευτής πρόκειται να αποκόπτει την κακόβουλη κίνηση σε πραγματικό χρόνο.

4.2 Προσέγγιση Federated Learning

Η πειραματική διάταξη που υλοποιούμε για την εκπαίδευση στα πρότυπα του Federated Learning των μοντέλων που αναπτύξαμε, αποτελείται από 3 virtual clients (Alice, Bob, Charlie) και έναν κεντρικό aggregating server που τρέχουν στο ίδιο Linux-like μηχάνημα. Ο κεντρικός server είναι υπεύθυνος για τη διαδικασία του συμψηφισμού των τοπικών μοντέλων και την περιοδική αξιολόγηση του γενικού μοντέλου που προκύπτει σε κάθε γύρο. Ο συμψη-

φισμός γίνεται με τον παραδοσιακό αλγόριθμο του Federated Averaging. Για το deployment της εφαρμογής ανοίγουμε 4 διαφορετικές διεργασίες στη συσκευή που χρησιμοποιούμε για την εκτέλεση των πειραμάτων, 3 για τους clients και μια για τον server. Κάθε διεργασία τρέχει σε ένα ξεχωριστό port, στο οποίο και ακούει. Οι clients συνδέονται με τον aggregating server, μέσω TCP websocket connections για την μεταξύ τους επικοινωνία. Ύστερα εκτελούμε ένα πρόγραμμα για τη διαμέριση των δεδομένων ανάμεσα στους 3 clients, με τρόπο που εμείς ορίζουμε σε κάθε πείραμα που θα εκτελέσουμε στο κεφάλαιο 6. Ένα μέρος των δεδομένων (20%) στέλνουμε και στον aggregating server, ώστε πάνω σε αυτά να εκτελείται η αξιολόγηση του γενικού μοντέλου. Τονίζεται εδώ, ότι σε μια πραγματική εφαρμογή Federated Learning τα δεδομένα είναι ήδη αποθηκευμένα στις τοπικές συσκευές (στην περίπτωσή μας στους recursive DNS). Πράγματι, η διαδικασία που περιγράφουμε παραπάνω δεν συμπεριλαμβάνεται στη διαδικασία της εκπαίδευσης και ο σκοπός της είναι αποκλειστικά, η εξυπηρέτηση του setup. Μόλις διεκπεραιωθεί η διαδικασία αυτή ξεκινάμε την εκπαίδευση, ακολουθώντας πιστά τα βήματα που περιγράφονται στον αλγόριθμο 2.1, αλλά αντί να διαλέγουμε σε κάθε γύρο εκπαίδευσης ένα τυχαίο υποσύνολο από clients, συμπεριλαμβάνουμε πάντα και τους 3 στη διαδικασία της εκπαίδευσης, αφού ο αριθμός τους είναι μικρός για χάρη της απλότητας των πειραμάτων. Επίσης, το γενικό μοντέλο σε κάθε γύρο εκπαίδευσης προκύπτει από τον απλό μέσο όρο των παραμέτρων των τοπικών μοντέλων και όχι από τον σταθμισμένο μέσο όρο αυτών. Ο λόγος για την επιλογή μας αυτή εξηγείται στην υποενότητα 2.5.2. Εκτελείται λοιπόν η διαδικασία εκπαίδευσης, όπως περιγράφεται στη συνέχεια:

1. **Βήμα 1:** Ο aggregating server αρχικοποιεί το μοντέλο προς εκπαίδευση (CNN, LSTM, Bi-LSTM) και το στέλνει στους 3 clients.

2. **Βήμα 2:** Οι 3 clients (Alice, Bob, Charlie) εκπαιδεύουν παράλληλα και ασύγχρονα τα τοπικά τους μοντέλα, υπό την έννοια ότι μόλις ένας client ολοκληρώσει το έργο του μεταβαίνει στο επόμενο βήμα, χωρίς να περιμένει τους υπόλοιπους να ολοκληρώσουν.

3. **Βήμα 3:** Με το πέρας της τοπικής εκπαίδευσης κάθε client συμπιέζει σε ένα αρχείο τις ενημερωμένες παραμέτρους του μοντέλου και τις στέλνει στον aggregating server. Η συμπίεση γίνεται με χρήση του αλγορίθμου LZ4 [56], [57], ο οποίος επικεντρώνεται στην ταχύτητα τόσο της συμπίεσης, όσο και της αποσυμπίεσης των δεδομένων, με σκοπό την επιτάχυνση της επικοινωνίας. Ο συγκεκριμένος αλγόριθμος έχει ενσωματωθεί ως εργαλείο του PySyft framework.

4. **Βήμα 4:** Μόλις ο aggregating server λάβει τις ενημερώσεις και των τριών client αποσυμπιέζει τα σχετικά μηνύματα και αξιολογεί τα τοπικά μοντέλα πάνω στο δικό του dataset. Το σημείο αυτό δεν είναι απαραίτητο και εξυπηρετεί απλώς τα πειράματά μας. Ύστερα συμψηφίζει τα τοπικά μοντέλα με Federated Averaging και τέλος αξιολογεί το νέο γενικό μοντέλο.

5. **Βήμα 5:** Ο aggregating server συμπιέζει το γενικό μοντέλο και το στέλνει εκ νέου στους 3 clients. Επαναλαμβάνεται έτσι ο κύκλος από το Βήμα 2 έως το Βήμα 5, μέχρις ότου συμπληρωθούν 30 γύροι εκπαίδευσης, ή μέχρι η ακρίβεια Accuracy του γενικού μοντέλου να μην βελτιωθεί για περισσότερους από 3 γύρους (αν δηλαδή μετά

τον υποδιπλασιασμό του learning rate δεν υπάρξει βελτίωση). Βέβαια όπως θα δούμε στην πορεία από τα πειράματα που θα εκτελέσουμε, συνήθως αρκούν πολύ λιγότεροι από 30 γύροι εκπαίδευσης για να συγκλίνει το μοντέλο.

Ο απαραίτητος κώδικας έχει γραφτεί σε Python (version 3.6) και έχει χρησιμοποιηθεί το PySyft framework (version 0.2.4). Περισσότερες πληροφορίες για τον κώδικα που αναπτύξαμε μπορούμε να βρούμε στο παράρτημα Α΄ της διπλωματικής.

Κεφάλαιο 5

Περιγραφή και Τρόπος Δημιουργίας των Dataset Εκπαίδευσης και Αξιολόγησης

Στο Κεφάλαιο αυτό εξηγείται πώς κατασκευάστηκαν τα datasets που θα χρησιμοποιήσουμε για τη διαδικασία της εκπαίδευσης και της αξιολόγησης των μοντέλων. Παραδοσιακά ένα dataset που προορίζεται για εκπαίδευση (training dataset), διαμερίζεται ομοιογενώς σε δύο μέρη, ένα train-set και ένα validation-set. Στην εργασία μας τα σχετικά ποσοστά διαμέρισης είναι 80% για το train-set και 20% για το validation-set. Το train-set χρησιμοποιείται για την εκπαίδευση των μοντέλων καθαυτή. Ο ρόλος του validation-set είναι η αξιολόγηση σε κάθε ενδιάμμεσο γύρο εκπαίδευσης, ώστε να ελέγχεται η ταχύτητα σύγκλισης, η ακρίβεια και μια σειρά επιπλέον παραμέτρων που χαρακτηρίζουν την απόδοση του μοντέλου. Η διαδικασία αυτή είναι πολύ χρήσιμη για να ελέγχουμε σε κάθε γύρο αν ικανοποιούνται ορισμένες συνθήκες, προκειμένου να εκτελεστούν επιπλέον ενέργειες, όπως για παράδειγμα υποδιπλασιασμός του learning rate (στην περίπτωσή μας), ή ακόμα και πρόωρος τερματισμός της εκπαίδευσης αν το μοντέλο έχει πετύχει τη βέλτιστη ακρίβειά του. Ωστόσο, για την ακέραια αξιολόγηση του τελικού μοντέλου συνίσταται η χρήση ενός διαφορετικού dataset (testing dataset). Ο σκοπός της διαδικασίας είναι ο έλεγχος της ικανότητας του μοντέλου να γενικεύει αποτελεσματικά, δηλαδή η δυνατότητά του να εντοπίζει ονόματα που παράγονται από οικογένειες DGA, που δεν συμπεριλήφθηκαν στο dataset εκπαίδευσης και κατ' επέκταση να ανακαλύπτει εγκαίρως νέες οικογένειες DGA σε μια πραγματική εφαρμογή. Ταυτόχρονα, με τον τρόπο αυτό μπορεί να παρατηρηθεί τυχόν overfit των μοντέλων μας.

5.1 Training Dataset

Για χάρη των πειραμάτων που θα ακολουθήσουν, χρησιμοποιούμε δύο διαφορετικά σύνολα από domain names. Στην πρώτη περίπτωση, σε αντίθεση με τη δεύτερη, χρησιμοποιούνται μόνο Non-wordlist-based DGAs. Το πρώτο dataset, αποτελείται από 220.000 εγγραφές-domain names. Τα μισά από αυτά προήλθαν από τη συλλογή Alexa Top 1 Million με επιλογή των δημοφιλέστερων ονομάτων και τα υπόλοιπα δημιουργήθηκαν με την εκτέλεση συγκεκριμένων αλγορίθμων DGA. Πιο αναλυτικά, εκτελέστηκαν 11 διαφορετικοί DGAs και δημιουργήθηκαν 10.000 ονόματα για κάθε οικογένεια. Το σχετικό dataset, καθώς και οι DGAs που εκτελέστηκαν για την παραγωγή του διατίθενται στο github repository [58].

Επιπλέον, παρατηρήσαμε ότι το ίδιο dataset έχει χρησιμοποιηθεί στη μελέτη των Palac et al. [46] για την εκπαίδευση των μοντέλων τους.

Το dataset πάνω στο οποίο εκπαιδεύονται τα μοντέλα στη δεύτερη περίπτωση (Wordlist-Based DGAs Included), είναι το ίδιο με το προηγούμενο, αλλά έχει εμπλουτιστεί με 100.000 επιπλέον εγγραφές-domain names. Τα μισά από αυτά (50.000) προέρχονται από τη συλλογή Alexa Top 1 Million, ενώ τα υπόλοιπα δημιουργήθηκαν με την εκτέλεση 5 διαφορετικών Wordlist-based DGAs, από την οποία παρήχθησαν 10.000 ονόματα για κάθε οικογένεια. Οι σχετικοί DGAs που εκτελέστηκαν είναι διαθέσιμοι στο github repository [59]. Οι DGAs που εκτελέστηκαν για την παραγωγή του κάθε dataset, καθώς και πληροφορίες γι' αυτούς φαίνονται στους παρακάτω πίνακες.

DGA Family	Time Dependent	Deterministic	Generation Scheme
Ramnit		✓	Alphanumeric
Kraken		✓	Alphanumeric
Simda		✓	Alphanumeric
Banjori		✓	Alphanumeric
Pykspa	✓	✓	Alphanumeric
Ramdo		✓	Alphanumeric
QakBot	✓	✓	Alphanumeric
Cryptolocker	✓	✓	Alphanumeric
Locky	✓	✓	Alphanumeric
CoreBot	✓	✓	Alphanumeric
DirCrypt		✓	Alphanumeric

Πίνακας 5.1: *Training Dataset 1 (Non-Wordlist-Based DGA Domain Names)*

DGA Family	Time Dependent	Deterministic	Generation Scheme
Ramnit		✓	Alphanumeric
Kraken		✓	Alphanumeric
Simda		✓	Alphanumeric
Banjori		✓	Alphanumeric
Pykspa	✓	✓	Alphanumeric
Ramdo		✓	Alphanumeric
QakBot	✓	✓	Alphanumeric
Cryptolocker	✓	✓	Alphanumeric
Locky	✓	✓	Alphanumeric
CoreBot	✓	✓	Alphanumeric
DirCrypt		✓	Alphanumeric
Matsnu	✓	✓	Wordlist
Suppobox	✓	✓	Wordlist
Gozi	✓	✓	Wordlist
Nymaim v2	✓	✓	Wordlist
Pizd	✓	✓	Wordlist

Πίνακας 5.2: *Training Dataset 2 (Wordlist-Based DGAs Included)*

5.2 Testing Dataset

Το dataset αξιολόγησης αποτελείται από 918.000 domain names. Από αυτά τα μισά είναι legit ονόματα (διαφορετικά από τα αντίστοιχα του training dataset), προερχόμενα από τη συλλογή Alexa Top 1 Million, ενώ τα υπόλοιπα έχουν δημιουργηθεί μετά από εκτέλεση 34 διαφορετικών DGA. Από αυτούς οι 18 δεν έχουν συμπεριληφθεί στο dataset εκπαίδευσης. Ωστόσο, ακόμα και για αυτούς που έχουν συμπεριληφθεί, χρησιμοποιούμε διαφορετικά seeds (ή wordlists) κατά την εκτέλεσή τους, ώστε τα παραγόμενα domain names να είναι διαφορετικά από αυτά που χρησιμοποιήθηκαν για την εκπαίδευση. Κάθε DGA παράγει 13.500 domain names.

DGA Family	Time Dependent	Deterministic	Generation Scheme
Ramnit		✓	Alphanumeric
Kraken		✓	Alphanumeric
Simda		✓	Alphanumeric
Banjori		✓	Alphanumeric
Pykspa	✓	✓	Alphanumeric
Ramdo		✓	Alphanumeric
QakBot	✓	✓	Alphanumeric
Cryptolocker	✓	✓	Alphanumeric
Locky	✓	✓	Alphanumeric
CoreBot	✓	✓	Alphanumeric
DirCrypt		✓	Alphanumeric
Matsnu	✓	✓	Wordlist
Suppobox	✓	✓	Wordlist
Gozi	✓	✓	Wordlist
Nymaim v2	✓	✓	Wordlist
Pizd	✓	✓	Wordlist
Ranbyus	✓	✓	Alpanumeric
MoneroDownloader	✓	✓	Hash (MD5)
Symmi	✓	✓	Alpanumeric
Emotet	✓	✓	Alpanumeric
Fobber		✓	Alpanumeric
Pushdo		✓	Hash (MD5)
Qadars	✓	✓	Alpanumeric
Necurs	✓	✓	Alpanumeric
Conficker	✓	✓	Alpanumeric
Tinba	✓	✓	Alpanumeric
Murofet	✓	✓	Alpanumeric
Rovnix		✓	Alpanumeric
Shiotob		✓	Alpanumeric
Proslikefan	✓	✓	Alpanumeric
Padcrypt	✓	✓	Hash (SHA256)
GameoverZeus	✓	✓	Hash (MD5)
DnsChanger		✓	Alpanumeric
MyDoom	✓	✓	Alpanumeric

Πίνακας 5.3: *Testing Dataset DGAs*

Το υπό μελέτη dataset δημιουργήθηκε συνδυαστικά ως εξής. Σε πρώτη φάση χρησιμοποιήθηκε το ήδη υπάρχον dataset [60], το οποίο περιέχει 675.000 εγγραφές με τις μισές να είναι legit (Alexa Top 1 Million) και τις υπόλοιπες να αποτελούν κακόβουλα domain names που δημιουργήθηκαν από 25 διαφορετικές οικογένειες DGA. Σε κάθε οικογένεια αντιστοιχούν 13.500 domain names. Οι δημιουργοί του άντλησαν τα δεδομένα από το Netlab Opendata Project [61], που αποτελεί ένα εναποθετήριο DGA ονομάτων, προερχόμενα από ανάλυση σε εγγραφές Passive DNS. Τελικά εμπλουτίσαμε το συγκεκριμένο dataset με 9 επιπλέον οικογένειες DGA, εκτελώντας τους σχετικούς αλγορίθμους από το project [59], ενώ ταυτόχρονα προσθέσαμε ισάριθμες εγγραφές από την Alexa, ώστε να το εξισορροπήσουμε.

Κεφάλαιο 6

Εκτέλεση των Πειραμάτων

Στο κεφάλαιο αυτό εξετάζεται η αποτελεσματικότητα των μοντέλων που περιγράψαμε στο κεφάλαιο 4 για την ανίχνευση DGA domain names και ελέγχεται η απόδοση του σχήματος εκπαίδευσης Federated Learning που προτείνουμε, μέσα από την εκτέλεση μιας σειράς πειραμάτων. Στην πορεία του κεφαλαίου σχολιάζονται τα αποτελέσματα που προκύπτουν, καθώς και οι περιορισμοί του συστήματός μας που αναδεικνύονται.

6.1 Κριτήρια Αξιολόγησης της Απόδοσης

Για την αξιολόγηση των μοντέλων στον ευρύτερο τομέα της Μηχανικής Μάθησης χρησιμοποιούνται ευρέως οι ακόλουθες 4 μετρήσεις:

1. **Accuracy:** Είναι η πιο κοινή μέτρηση απόδοσης και εκφράζει την αναλογία των συνολικών σωστών προβλέψεων προς τον συνολικό αριθμό των προβλέψεων. Η ακρίβεια είναι μια εξέχουσα μέτρηση όταν έχουμε ένα ισορροπημένο σύνολο δεδομένων.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{6.1}$$

2. **Precision:** Εκφράζει την αναλογία των προβλέψεων προς το σύνολο των θετικών προβλέψεων (θετικών και ψευδών). Μεγαλύτερες τιμές αυτής της παραμέτρου υποδηλώνουν χαμηλότερο FPR. Επομένως είναι σημαντικό για συστήματα εντοπισμού DGA domain names που αποκόβουν την κακόβουλη κίνηση σε πραγματικό χρόνο, η τιμή της να είναι όσο το δυνατόν υψηλότερη.

$$Precision = \frac{TP}{TP + FP} \tag{6.2}$$

3. **Recall:** Για μία κλάση (π.χ. DGA domain names) εκφράζει το σύνολο των αληθών θετικών προβλέψεων, προς τις συνολικές προβλέψεις που αφορούν δεδομένα (domain names) της συγκεκριμένης κλάσης.

$$Recall = \frac{TP}{TP + FN} \tag{6.3}$$

4. **F1 score:** Είναι ο σταθμισμένος μέσος όρος του Precision και του Recall (Ανάκληση). Ως εκ τούτου, δίνεται έμφαση στις ψευδώς αρνητικές και ψευδώς θετικές προβλέψεις.

Ως μέτρηση είναι συνήθως πιο πολύτιμη από το Accuracy (Ακρίβεια), ειδικά όταν έχουμε ένα μη εξισορροπημένο dataset.

$$F1score = \frac{2 \times (Recall \times Precision)}{Recall + Precision} \tag{6.4}$$

6.2 Πειράματα

Αρχικά εξετάζουμε την απόδοση των 3 μοντέλων (CNN, LSTM, Bi-LSTM) στην παραδοσιακή εκδοχή Non-Federated Learning, όπου όλα τα δεδομένα που χρησιμοποιούνται στη διαδικασία της εκπαίδευσης βρίσκονται συγκεντρωμένα σε έναν κεντρικό server. Μελετούμε δηλαδή τη βέλτιστη απόδοση υπό συνθήκες που δεν παρατηρείται η αρνητική επίδραση του παράγοντα της ετερογένειας των δεδομένων των clients που συναντάται στο Federated Learning. Στη συνέχεια συγκρίνουμε τα αποτελέσματα αυτά με τα αντίστοιχα που προκύπτουν κατά την εκπαίδευση με Federated Learning. Μελετούμε ενδελεχώς την αρνητική επίδραση της ετερογένειας των δεδομένων των clients στην ακρίβεια του τελικού γενικού μοντέλου και στοχεύουμε στον περιορισμό αυτής, μέσω της αύξησης της συχνότητας εκτέλεσης συμψηφισμού με Federated Averaging ανά εποχή εκπαίδευσης. Τελικά αναλύουμε το κέρδος των εικονικών clients από τη συμμετοχή τους στο συνεργατικό περιβάλλον εκπαίδευσης Federated Learning που υλοποιήσαμε και εξάγουμε τα απαραίτητα συμπεράσματα.

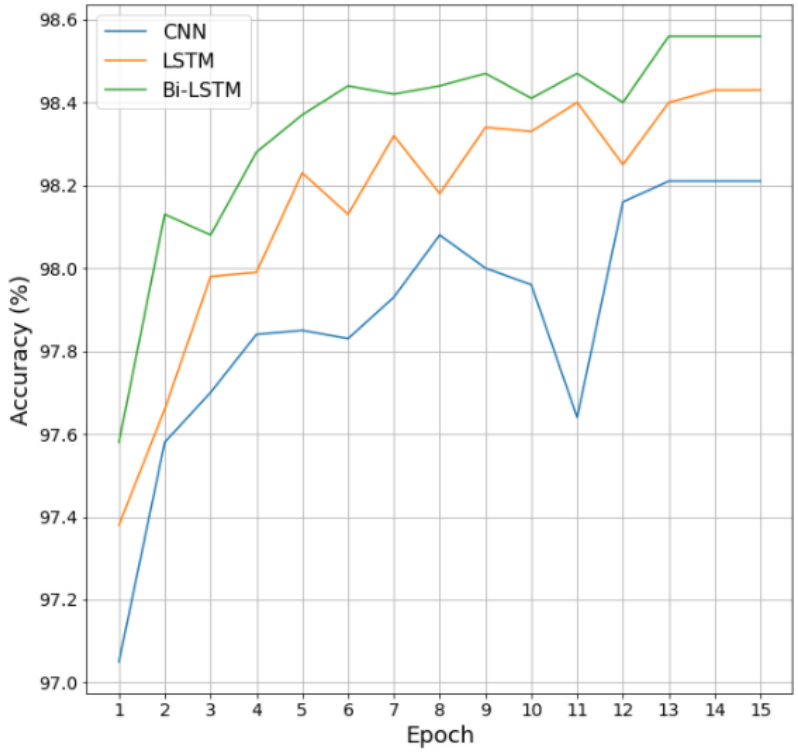
6.2.1 Απόδοση των Μοντέλων σε Αρχιτεκτονική Non-Federated Learning

Στο πρώτο πείραμα αυτής της υποενότητας εκπαιδεύουμε τα μοντέλα μας πάνω σε δύο διαφορετικά datasets. Στην πρώτη περίπτωση χρησιμοποιούμε το dataset 5.1, οπότε δεν συμπεριλαμβάνουμε wordlist-based DGAs στην εκπαίδευση. Στη δεύτερη περίπτωση η εκπαίδευση γίνεται πάνω στο dataset 5.2, άρα συμπεριλαμβάνουμε και τους wordlist-based DGAs. Ύστερα αναλύουμε την απόδοση των μοντέλων μας με βάση τα κριτήρια αξιολόγησης που παρουσιάστηκαν στην προηγούμενη ενότητα. Σκοπός αυτού του διαχωρισμού είναι να μελετήσουμε το πρόβλημα της χαμηλής ακρίβειας των ανιχνευτών Deep Learning για τους wordlist-based DGAs. Παράλληλα εξετάζουμε την επίδραση (είτε θετική, είτε αρνητική) που επιφέρει στην απόδοση των μοντέλων μας η συμπερίληψη των εν λόγω DGA domain names στο training dataset.

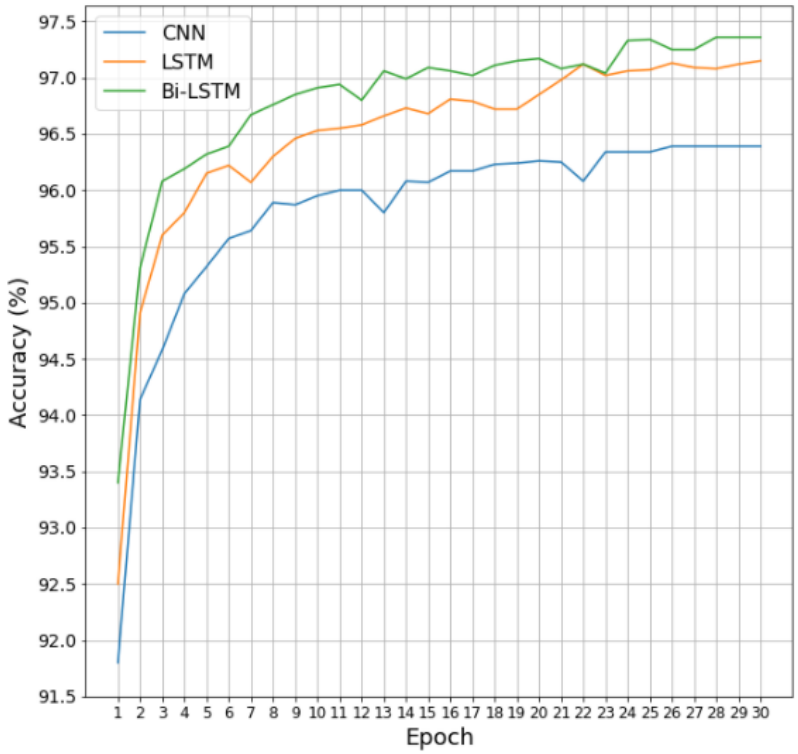
Στον παρακάτω πίνακα 6.1 φαίνεται το Accuracy που πετυχαίνει το βέλτιστο μοντέλο σε κάθε περίπτωση εκπαίδευσης. Επιπροσθέτως, στα διαγράμματα που ακολουθούν αναλύεται η ακρίβεια (Accuracy) κάθε μοντέλου, συναρτήσει των εποχών εκπαίδευσης. Οι μετρήσεις αυτές αφορούν το validation-set του dataset εκπαίδευσης και τις λαμβάνουμε μετά από κάθε εποχή.

Models Trained on Dataset 5.1			**Models Trained on Dataset 5.2**		
CNN	**LSTM**	**Bi-LSTM**	**CNN**	**LSTM**	**Bi-LSTM**
98.21	98.43	98.56	96.39	97.15	97.36

Πίνακας 6.1: *Accuracy (%) of Optimized Models for Validation-set*



Εικόνα 6.1: *Validation-set Accuracy (%): Models Trained on Non-Wordlist-Based DGAs Dataset 5.1*



Εικόνα 6.2: *Validation-set Accuracy (%): Models Trained on Dataset 5.2 (Wordlist-Based DGAs Included)*

Παρατηρούμε ότι από τα τρία μοντέλα το Bidirectional LSTM πετυχαίνει τελικά τη μεγαλύτερη ακρίβεια, ξεπερνώντας έτσι το απλό LSTM μοντέλο που προτάθηκε από τους Yu et al. και που βελτιώθηκε στην πορεία σε αυτή τη διπλωματική. Αποδεικνύεται δηλαδή, η βελτιωμένη διακριτική ικανότητα του Bidirectional LSTM σε σχέση με το συμβατικό LSTM για την ανίχνευση DGA domain names. Από την άλλη πλευρά, το CNN μοντέλο αν και πετυχαίνει εντυπωσιακά αποτελέσματα, φαίνεται να υστερεί σε σχέση με τα άλλα δύο μοντέλα και ειδικά στη δεύτερη περίπτωση εκπαίδευσης, όπου συμπεριλαμβάνονται και wordlist-based DGAs στη διαδικασία. Γενικά όμως, όπως προκύπτει για την περίπτωση αυτή, η ακρίβεια των μοντέλων μας για το validation-set μειώνεται λόγω σημαντικής αύξησης του FPR και του FNR (False Negative Rate). Το γεγονός αυτό οφείλεται στην ομοιότητα των legit domain names με αυτά που έχουν παραχθεί από wordlist-based DGAs, με αποτέλεσμα η σωστή κατηγοριοποίηση να γίνεται πιο δύσκολη. Επίσης όπως είναι αναμενόμενο, από τη στιγμή που το dataset 5.2 διαθέτει μεγαλύτερο όγκο ονομάτων, επιβραδύνεται αναλόγως και η διαδικασία της εκπαίδευσης στη συγκεκριμένη περίπτωση.

Αξιολογούμε τώρα τους ανιχνευτές που εκπαιδεύσαμε σε κάθε περίπτωση, πάνω στο testing dataset που περιγράφηκε στο κεφάλαιο 5. Επιδιώκουμε με αυτόν τον τρόπο να αξιολογήσουμε τη γενικότερη ανιχνευτική ικανότητα των μοντέλων μας. Μελετάμε δηλαδή την ικανότητα των ταξινομητών μας να εντοπίζουν γενικά DGA domain names, ακόμα και αν αυτά ανήκουν σε οικογένειες που δεν έχουν συμπεριληφθεί στην εκπαίδευση. Το χαρακτηριστικό αυτό είναι πολύ σημαντικό, καθώς σχετίζεται έμμεσα με την προοπτική ανακάλυψης καινούριων οικογενειών DGA.

	CNN			**LSTM**			**Bi-LSTM**		
	Precision	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**
Legit	0.76	0.97	0.85	0.78	0.98	0.87	0.78	0.98	0.87
DGA	0.96	0.69	0.80	0.97	0.72	0.83	0.97	0.73	0.83
Macro Avg	0.86	0.83	0.83	0.88	0.85	0.85	0.88	0.85	0.85
Weighted Avg	0.86	0.83	0.83	0.88	0.85	0.85	0.88	0.85	0.85
Accuracy	0.8308			0.8492			0.8536		

Πίνακας 6.2: *Classification Report for Models Trained on Dataset 5.1 (Non-Wordlist DGAs)*

	CNN			**LSTM**			**Bi-LSTM**		
	Precision	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**
Legit	0.84	0.89	0.86	0.88	0.90	0.89	0.88	0.92	0.90
DGA	0.88	0.83	0.85	0.90	0.88	0.89	0.91	0.87	0.89
Macro Avg	0.86	0.86	0.86	0.89	0.89	0.89	0.89	0.89	0.89
Weighted Avg	0.86	0.86	0.86	0.89	0.89	0.89	0.89	0.89	0.89
Accuracy	0.8575			0.8901			0.8936		

Πίνακας 6.3: *Classification Report for Models Trained on Dataset 5.2*

Αναλυτικά για κάθε DGA οικογένεια παρουσιάζουμε την Ανάκληση (Recall) που πετυχαίνει κάθε μοντέλο στον πίνακα που ακολουθεί.

	Models Trained on Dataset 5.1			Models Trained on Dataset 5.2		
DGA Family	**CNN**	**LSTM**	**Bi-LSTM**	**CNN**	**LSTM**	**Bi-LSTM**
Ramnit	97.91	98.08	98.52	97.78	98.23	98.21
Kraken	89.59	90.31	92.70	89.53	90.52	91.04
Simda	84.12	97.11	90.99	91.85	92.45	90.06
Banjori	11.93	39.70	57.33	100.00	99.98	100.00
Pykspa	85.81	87.61	87.96	86.14	89.50	89.31
Ramdo	99.92	99.75	99.94	99.91	99.93	99.99
QakBot	98.76	98.93	99.19	98.78	98.87	99.05
Cryptolocker	98.93	99.06	99.31	98.67	99.28	99.44
Locky	96.30	96.53	97.54	96.29	96.79	96.80
CoreBot	100.00	99.90	99.93	99.99	99.83	99.82
DirCrypt	98.61	98.70	98.90	98.50	98.60	98.76
Matsnu	3.42	1.36	0.82	96.91	96.79	96.75
Suppobox	0.57	0.44	0.32	80.44	87.11	85.99
Gozi	6.19	6.56	4.56	80.63	88.32	89.27
Nymaim v2	1.55	1.29	1.39	81.56	85.26	83.95
Pizd	0.59	0.50	0.61	73.42	67.68	64.65
Ranbyus	99.58	99.73	99.79	99.72	99.87	99.88
MoneroDownloader	2.70	2.33	2.67	1.93	16.85	18.37
Symmi	57.98	61.21	60.34	62.87	69.58	68.10
Emotet	99.56	99.59	99.74	99.70	99.85	99.84
Fobber	96.41	96.79	97.36	96.64	96.64	96.79
Pushdo	35.73	38.84	36.39	39.39	38.47	39.39
Qadars	41.38	64.70	61.58	32.64	83.27	74.93
Necurs	97.33	97.81	98.13	97.35	97.71	97.70
Conficker	55.95	55.93	59.68	55.87	57.10	57.50
Tinba	99.00	98.87	99.20	98.91	99.37	99.20
Murofet	86.33	94.66	98.95	95.21	99.79	97.45
Rovnix	81.29	97.61	99.27	63.33	99.52	99.61
Shiotob	62.99	67.73	69.49	43.71	68.29	61.42
Proslikefan	88.74	88.79	91.64	88.18	88.92	89.47
Padcrypt	95.34	96.20	97.45	95.64	98.30	98.30
GameoverZeus	79.35	88.90	97.30	92.90	96.81	90.78
DnsChanger	96.36	96.86	97.33	96.40	96.56	97.00
MyDoom	87.56	89.51	90.32	86.87	90.18	90.64

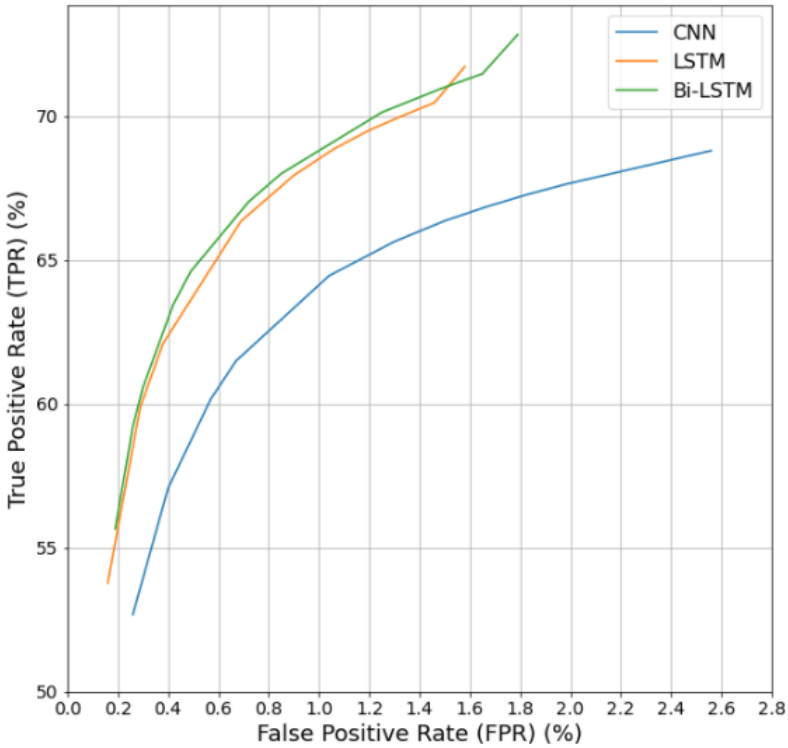
Πίνακας 6.4: *Recall (%) Achieved for each DGA Family on Testing Dataset*

Από τους παραπάνω πίνακες συμπεραίνουμε τα εξής. Στην πρώτη περίπτωση εκπαίδευσης, όπου δεν έχουν συμπεριληφθεί wordlist-based DGAs στο training dataset, γίνεται αισθητό το πρόβλημα της αδυναμίας εντοπισμού των οικογενειών αυτών. Μάλιστα η ανάκληση (Recall) για τις συγκεκριμένες οικογένειες DGA (Matsnu, Suppobox, Gozi, Nymaim v2, Pizd) τείνει σε κάποιες περιπτώσεις στο 0%. Αυτό σημαίνει ότι οι ανιχνευτές μας θεωρούν τα ονόματα αυτά με ελάχιστες εξαιρέσεις ως νόμιμα. Παρόλα αυτά φαίνεται ότι για τους υπόλοιπους 23 DGAs που δεν συμμετέχουν στην εκπαίδευση, το Recall είναι ως επί το πλείστον ικανοποιητικό. Ένα άλλο πλεονέκτημα των ανιχνευτών που εκπαιδεύονται πάνω σε Non-wordlist-based DGAs είναι το χαμηλό FPR ($< 1.5\%$). Για τους 11 DGAs που συνιστούν

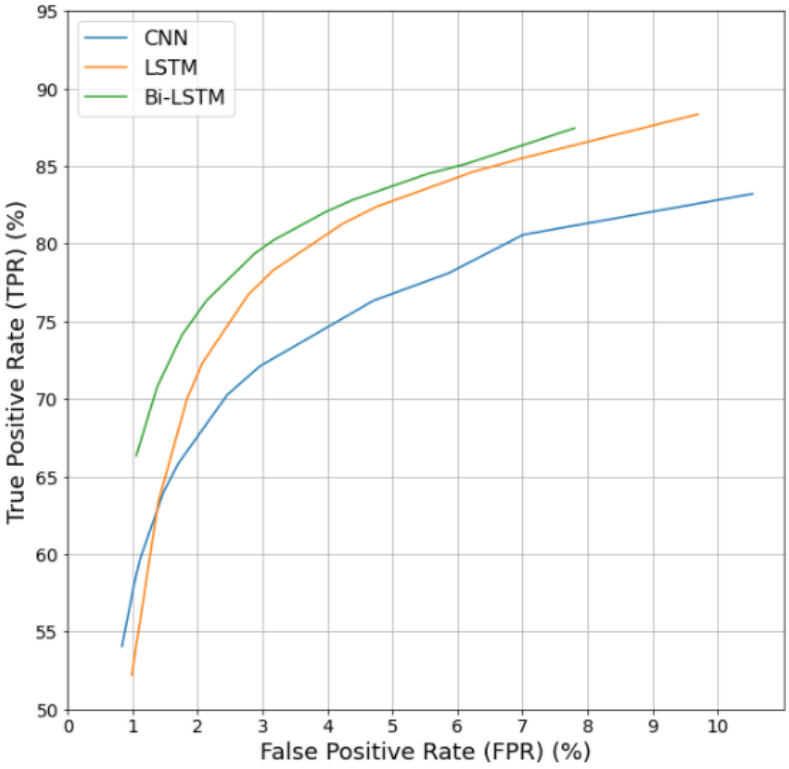
το training dataset στην πρώτη περίπτωση εκπαίδευσης, τα αποτελέσματα δεν διαφέρουν σε σχέση με αυτά του validation-set. Για τις περισσότερες οικογένειες επιτυγχάνεται Recall που ξεπερνά το 98%. Επιβεβαιώνεται έτσι, ότι ανεξαρτήτως των διαφορετικών seeds που χρησιμοποιήθηκαν για τη δημιουργία του Testing Dataset, οι ανιχνευτές μας έχουν εκπαιδευτεί σωστά, ώστε να εντοπίζουν τα συγκεκριμένα domain names. Εξαίρεση αποτελεί η οικογένεια Banjori, για την οποία παρόλο που στο validation-set πετυχαίνεται Recall άνω του 99%, εδώ τα αποτελέσματα είναι απογοητευτικά. Αιτία γι' αυτό αποτελεί ο τρόπος με τον οποίο λειτουργεί ο Banjori [62], ο οποίος λαμβάνει ως seed μία αγγλική λέξη και παράγει domain names αντικαθιστώντας τα 4 πρώτα γράμματα της λέξης με διαφορετικούς συνδυασμούς ψευδοτυχαίων γραμμάτων. Συνεπώς, χρησιμοποιώντας διαφορετικά seeds για τον Banjori στο Testing Dataset, τα ονόματα που παράγονται, εφόσον το μήκος της λέξης που χρησιμοποιείται ως seed είναι επαρκώς μεγάλο, διατηρούν τελικά μια ομοιότητα με legit domain names, αφού αν εξαιρέσουμε τα πρώτα 4 γράμματα, η υπόλοιπη λέξη παραμένει απαράλλαχτη. Το πρόβλημα αντιμετωπίζεται κατά τη δεύτερη περίπτωση εκπαίδευσης, με τη συμπερίληψη wordlist-based DGAs στο training dataset, όπου για τον Banjori έχουμε Recall 100%.

Σε αντίθεση με την πρώτη περίπτωση εκπαίδευσης, όταν συμπεριλαμβάνουμε wordlist-based DGAs στην εκπαίδευση, οι ανιχνευτές μας εντοπίζουν αποδοτικά τα σχετικά domain names. Στο Testing Dataset έχουμε Recall άνω του 85% (με εξαίρεση τον Pizd) για τους ανιχνευτές LSTM και Bidirectinal LSTM και μάλιστα για τον Matsnu επιτυγχάνεται ένα εντυπωσιακό ποσοστό της τάξης του 96%. Η προσθήκη των wordlist-based DGAs επιδρά μεταξύ άλλων θετικά στον εντοπισμό και των υπολοίπων οικογενειών και ειδικά για όσες δεν έχουν συμπεριληφθεί στο training dataset. Ενισχύεται δηλαδή κατ' αυτόν τον τρόπο η ικανότητα των μοντέλων να γενικεύουν και να εντοπίζουν καινούριους DGAs. Η παρατήρηση αυτή αντανακλάται μέσα από την αύξηση του TPR κατά 15% περίπου σε σχέση με την προηγούμενη περίπτωση εκπαίδευσης. Υπάρχει όμως και ένα μειονέκτημα που πρέπει να λαμβάνουμε υπόψη μας και αφορά στα αυξημένα ποσοστά του FPR, τα οποία προσεγγίζουν το 10%. Ο αριθμός αυτός είναι απαγορευτικός για ανιχνευτές που αποκόβουν την κακόβουλη κίνηση σε πραγματικό χρόνο.

Επομένως, αν επιθυμούμε απλά τον εντοπισμό της κακόβουλης κίνησης, χωρίς αυτή να αποκόβεται αμέσως, μπορούμε να επιδείξουμε μεγαλύτερη ανοχή στο FPR και να εκμεταλλευτούμε τα οφέλη από τη συμπερίληψη των wordlist-based DGAs στην εκπαίδευση. Από την άλλη πλευρά, όταν μας ενδιαφέρει το FPR να παραμένει σε χαμηλά επίπεδα, τότε είναι προτιμότερο να μην συμπεριλαμβάνουμε αυτού του τύπου τους DGAs στο training dataset, με ό,τι αυτό συνεπάγεται. Με αφορμή την παρατήρηση αυτή περιγράφουμε στους πίνακες που ακολουθούν το trade-off ανάμεσα σε TPR και FPR για τα μοντέλα που προκύπτουν σε καθεμιά από τις δύο περιπτώσεις εκπαίδευσης. Διαπιστώνουμε ότι μπορούμε να περιορίσουμε δραστικά το FPR (έως και στο μισό) για όλα τα μοντέλα, αν θυσιάσουμε ένα ποσοστό του TPR. Η σχετική μεταβολή είναι μέχρι ενός σημείου γραμμική. Εντούτοις αν το ξεπεράσουμε, το ποσοστό του TPR μειώνεται πλέον εκθετικά σε σχέση με το αντίστοιχο του FPR.



Εικόνα 6.3: *FPR-TPR Trade-Off: Models Trained on Dataset 5.1 (Non-Wordlist DGAs)*



Εικόνα 6.4: *FPR-TPR Trade-Off: Models Trained on Dataset 5.2 (Wordlist-Based DGAs Included)*

Σε επίπεδο μοντέλων, επαληθεύεται η βελτιωμένη ακρίβεια του Bidirectional LSTM σε σχέση με το συμβατικό LSTM. Βέβαια η βελτίωση αυτή κοστίζει σε μνήμη, καθώς όπως αναλύεται στις εικόνες 4.5 και 4.4 αντίστοιχα, το Bidirectional LSTM καταναλώνει 5.69 MB έναντι 2.87 MB του LSTM. Την επιβάρυνση αυτή πρέπει να λαμβάνουμε σοβαρά υπόψη μας, ιδιαίτερα για το συνεργατικό περιβάλλον Federated Learning που προτείνουμε, αφού η χρήση Bidirectional LSTM οδηγεί σε διπλασιασμό του κόστους επικοινωνίας μεταξύ clients και aggregating server. Παράλληλα το Bidirectional LSTM εμφανίζει και χρονικές καθυστερήσεις σε ό,τι αφορα τον μέσο χρόνο πρόβλεψης. Πιο αναλυτικά, χρειάζονται κατά μέσο όρο 4.26 ms για κάθε πρόβλεψη, δηλαδή διπλάσιος χρόνος σε σχέση με τα 2.12 ms του απλού LSTM. Από την άλλη πλευρά, το CNN μοντέλο υστερεί σημαντικά σε ότι αφορά την αποδοτική γενίκευση, σε σχέση με τα άλλα δύο μοντέλα που εξετάζουμε. Μάλιστα παρουσιάζει συγκριτικά μειωμένη ακρίβεια της τάξης του 2% και του 3.5% αντίστοιχα για τις δύο περιπτώσεις εκπαίδευσης. Ο μέσος χρόνος πρόβλεψης είναι 5.25 ms, ενώ αυξημένες είναι και οι απαιτήσεις σε μνήμη στα 19.30 MB, όπως προκύπτει από την ανάλυση 4.2. Σημειώνεται εδώ ότι για την εξαγωγή της μέσης χρονικής καθυστέρησης για κάθε πρόβλεψη, εκτελέστηκαν τα απαραίτητα πειράματα σε Linux like (Ubuntu 18.04) συσκευή με επεξεργαστή (CPU) AMD Ryzen 5 3400G.

6.2.2 Απόδοση των Μοντέλων σε Αρχιτεκτονική Federated Learning

Στην παρούσα υποενότητα μελετούμε την απόδοση και τη συμπεριφορά των ανιχνευτών μας, όταν εκπαιδεύονται στο πειραματικό περιβάλλον (3 clients και 1 aggregating server) συνεργατικής εκπαίδευσης Federated Learning που αναπτύξαμε. Πιο αναλυτικά, εξετάζουμε πως επιδρούν οι ακόλουθοι δύο παράγοντες στην απόδοση του τελικού μοντέλου:

1. **Ετερογένεια των δεδομένων:** Όπως ήδη έχουμε εξηγήσει (2.5.3), όσο αυξάνεται η ετερογένεια των δεδομένων εκπαίδευσης μεταξύ των clients, τόσο αποκλίνει η ακρίβεια του τελικού μοντέλου από το ιδανικό αντίστοιχο που θα προέκυπτε μέσω του παραδοσιακού Non-Federated Learning. Για να προσδιορίσουμε τον βαθμό της επίδρασης αυτής στην εφαρμογή μας, εκπαιδεύουμε τα μοντέλα μας πάνω σε 3 διαφορετικές κατανομές δεδομένων που αναπτύσσουμε για καθένα από τα δύο training dataset. Οι εν λόγω κατανομές A (υψηλή ετερογένεια), B (μέτρια ετερογένεια) και C (χαμηλή ετερογένεια) παρουσιάζονται στους πίνακες 6.5 και 6.6.

2. **Συχνότητα συμψηφισμού (FedAvg) ανά εποχή εκπαίδευσης:** Ο όρος αυτός σχετίζεται άμεσα με τη μέγιστη ποσότητα δεδομένων που θα χρησιμοποιήσει ο κάθε client για την εκπαίδευση του τοπικού μοντέλου πριν το στείλει στον aggregating server, για την εκτέλεση του συμψηφισμού με τη διαδικασία του Federated Averaging. Όπως θα δείξουμε στη συνέχεια, όσο πιο συχνά εκτελείται το Federated Averaging, δηλαδή όσο μειώνεται ο αριθμός των δεδομένων (domain names) που χρησιμοποιεί κάθε client σε κάθε γύρο εκπαίδευσης, τόσο αυξάνεται η ακρίβεια του τελικού μοντέλου και η ταχύτητα σύγκλισης αυτού. Για να επαληθεύσουμε τον συγκεκριμένο ισχυρισμό, δημιουργούμε δύο διαφορετικά σενάρια εκπαίδευσης με τη βοήθεια του PySyft. Στο πρώτο σενάριο ο συμψηφισμός εκτελείται με το πέρας κάθε εποχής, οπότε κάθε client

εκπαιδεύει το τοπικό του μοντέλο πάνω σε όλα τα δεδομένα που διαθέτει πριν το στείλει στον aggregating server. Αντιθέτως, στο δεύτερο σενάριο ο συμψηφισμός μπορεί να εκτελείται παραπάνω από μια φορά σε κάθε εποχή, αναλόγως του όγκου δεδομένων που διαθέτει ο κάθε client. Για την ακρίβεια, κατά την εκπαίδευση που εκτελείται τοπικά σε κάθε έναν από τους 3 clients το training dataset έχει χωριστεί σε batches αποτελούμενα από 64 domain names έκαστο. Μόλις οι 3 clients χρησιμοποιήσουν 200 batches, δηλαδή 12800 ονόματα, από τα αντίστοιχα τοπικά datasets που διαθέτουν, τότε η εκπαίδευση διακόπτεται, ώστε να γίνει ο συμψηφισμός. Το νέο γενικό μοντέλο επιστρέφεται στους clients και αυτοί με τη σειρά τους χρησιμοποιούν τα επόμενα 200 batches, μέχρι να εξαντληθούν όλα τα διαθέσιμα δεδομένα και να μεταβούμε στην επόμενη εποχή εκπαίδευσης.

Distribution	**A**			**B**			**C**		
Client	**Alice**	**Bob**	**Charlie**	**Alice**	**Bob**	**Charlie**	**Alice**	**Bob**	**Charlie**
Ramnit (#)	100%			100%			30%	43%	27%
Kraken (#)	100%			75%		25%	26%	49%	25%
Simda (#)	100%			50%	50%		30%	58%	12%
Banjori (#)		100%		50%	50%		50%	15%	35%
Pykspa (#)		100%		34%	66%		34%	19%	47%
Ramdo (#)		100%			100%		44%	36%	20%
QakBot (#)		100%			89%	11%	43%	46%	11%
Cryptolocker (#)		100%			58%	42%	24%	34%	42%
Locky (#)			100%	38%		62%	29%	28%	43%
CoreBot (#)			100%			100%	38%	38%	24%
DirCrypt (#)			100%		44%	56%	35%	40%	25%

Πίνακας 6.5: *Non-Wordlist-Based DGA Dataset 5.1 Distributions per Client*

Distribution	**A**			**B**			**C**		
Client	**Alice**	**Bob**	**Charlie**	**Alice**	**Bob**	**Charlie**	**Alice**	**Bob**	**Charlie**
Ramnit (#)	100%			100%			30%	43%	27%
Kraken (#)	100%			75%		25%	26%	49%	25%
Simda (#)	100%			50%	50%		30%	58%	12%
Banjori (#)		100%		50%	50%		50%	15%	35%
Pykspa (#)		100%		34%	66%		34%	19%	47%
Ramdo (#)		100%			100%		44%	36%	20%
QakBot (#)		100%			89%	11%	43%	46%	11%
Cryptolocker (#)		100%			58%	42%	24%	34%	42%
Locky (#)			100%	38%		62%	29%	28%	43%
CoreBot (#)			100%			100%	38%	38%	24%
DirCrypt (#)			100%		44%	56%	35%	40%	25%
Matsnu (#)			100%	25%	50%	25%	23%	27%	50%
Suppobox (#)			100%			100%	38%	31%	31%
Gozi (#)		100%		25%	56%	19%	38%	38%	24%
Nymaim v2 (#)	100%			25%	44%	31%	44%	38%	18%
Pizd (#)	100%			100%			34%	41%	25%

Πίνακας 6.6: *Dataset 5.2 Distributions per Client (Wordlist-Based DGAs Included)*

Στη συνέχεια παρουσιάζουμε την ακρίβεια των βέλτιστων μοντέλων που προκύπτουν κατά τη διαδικασία της συνεργατικής εκπαίδευσης. Εξετάζουμε όλους τους πιθανούς συνδυασμούς (6 στο σύνολο) που αφορούν τις συνθήκες εκπαίδευσης. Μελετούμε δηλαδή την ακρίβεια (Accuracy) για κάθε κατανομή δεδομένων (A, B, C) και για κάθε συχνότητα συμψηφισμού (per Epoch FedAvg, Per 200 Batches FedAvg). Σημειώνεται εδώ ότι οι σχετικές μετρήσεις αφορούν το validation-set.

Training Type	**CNN**	**LSTM**	**Bi-LSTM**
Per Epoch FedAvg A	94.36	94.79	94.80
Per 200 Batches FedAvg A	96.71	97.72	97.47
Per Epoch FedAvg B	90.74	95.40	92.00
Per 200 Batches FedAvg B	97.38	97.84	97.11
Per Epoch FedAvg C	96.75	92.90	92.87
Per 200 Batches FedAvg C	97.54	97.96	97.51
Non-Federated Learning	98.21	98.43	98.56

Πίνακας 6.7: *Accuracy (%) of Optimized Federated Models for Validation-Set - Training on Dataset 5.1 (Non-Wordlist DGA)*

Training Type	**CNN**	**LSTM**	**Bi-LSTM**
Per Epoch FedAvg A	87.24	89.07	86.61
Per 200 Batches FedAvg A	91.83	95.06	95.69
Per Epoch FedAvg B	87.83	87.89	87.22
Per 200 Batches FedAvg B	92.41	95.99	96.45
Per Epoch FedAvg C	85.35	85.83	85.33
Per 200 Batches FedAvg C	92.48	95.96	96.47
Non-Federated Learning	96.39	97.15	97.36

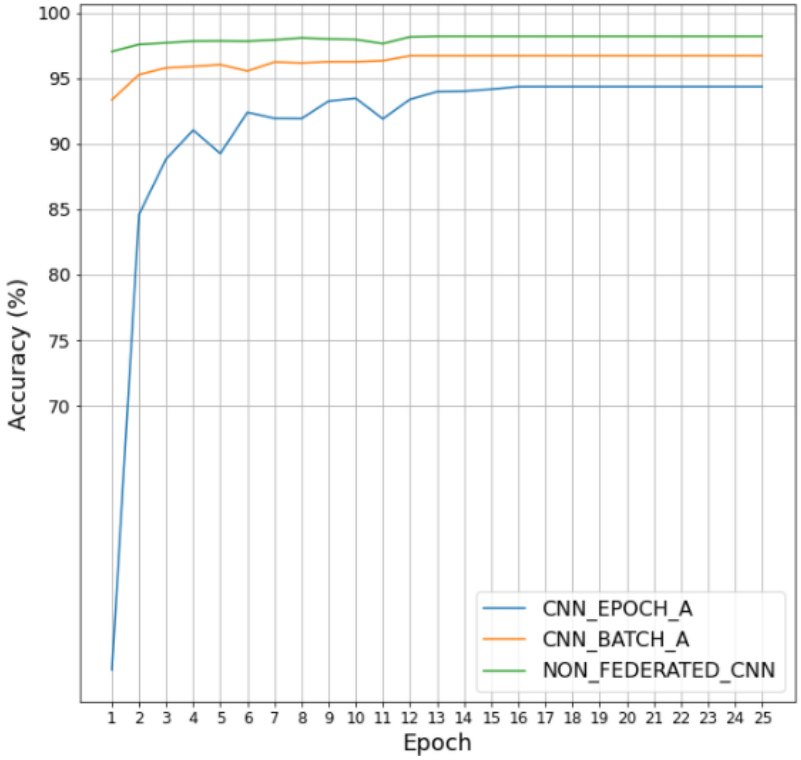
Πίνακας 6.8: *Accuracy (%) of Optimized Federated Models for Validation-Set - Training on Dataset 5.2 (Wordlist-Based DGAs Included)*

Με βάση λοιπόν τα παραπάνω αποτελέσματα οδηγούμαστε στα εξής σημαντικά συμπεράσματα. Αρχικά, για τη συνεργατική εκπαίδευση Deep Learning ανιχνευτών για τον εντοπισμό DGA domain names, δεν συνιστάται ο συμψηφισμός των μοντέλων ανά εποχή, αφού η απόδοση των τελικών μοντέλων σε ό,τι αφορά την ακρίβεια, εμφανίζεται δραστικά μειωμένη. Η εν λόγω διαφορά γίνεται ιδιαίτερα αισθητή για τους ανιχνευτές που εκπαιδεύονται στο dataset που συμπεριλαμβάνει wordlist-based DGAs. Αντιθέτως, όταν ο συμψηφισμός των μοντέλων εκτελείται με υψηλότερη συχνότητα και εν προκειμένου ανά 200 batches των 64 domain names, τα επιτυγχάνεται αρκετά ικανοποιητική σύγκλιση των Federated ανιχνευτών με τους αντίστοιχους ιδανικούς που εκπαιδεύονται υπό συνθήκες Non-Federated Learning. Χαρακτηριστικά, στην πρώτη περίπτωση η απόκλιση των μοντέλων από το ιδανικό κυμαίνεται από 3.5-5.5% για εκπαίδευση στο training dataset 5.1 (Non-wordlist DGAs) και από 8-12% για εκπαίδευση στο training dataset 5.2. Σαφώς και αυτή η ποινή που εισάγεται στα μοντέλα μας κατά την εκπαίδευση με Federated Learning, λόγω της κλιμακούμενης ετερογένειας των δεδομένων, περιορίζεται σημαντικά όταν αυξάνουμε την συχνότητα εκτέλεσης του Federated Averaging με τον τρόπο που εξηγήσαμε. Πράγματι, στη δεύτερη περίπτωση, η

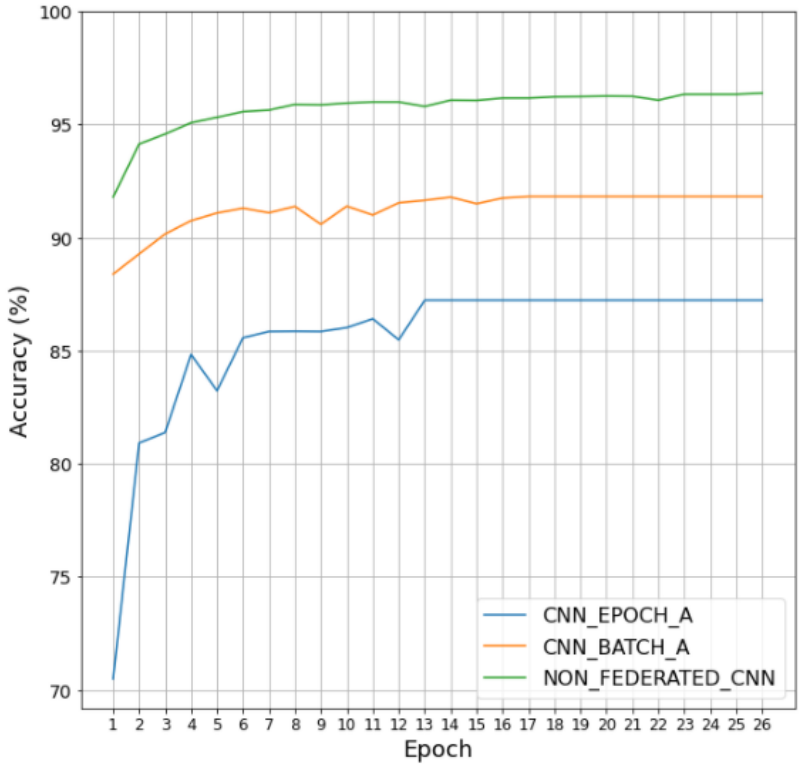
απόκλιση στο Accuracy των μοντέλων από το ιδανικό περιορίζεται στο 0.5-1.5% για τους για εκπαίδευση στο training dataset 5.1 (Non-wordlist DGAs) και από 0.9-4.5% για εκπαίδευση στο training dataset 5.2.

Αναλύοντας προσεκτικά τα προηγούμενα ποσοστά, μπορούμε να αντιληφθούμε ότι στην περίπτωση που συμπεριλαμβάνουμε wordlist-based DGAs στην εκπαίδευση, οι ανιχνευτές είναι πιο ευαίσθητοι στην ετερογένεια των domain names, δεδομένων των αυξημένων αποκλίσεων που εμφανίζουν συγκριτικά με τους ανιχνευτές που εκπαιδεύονται στο Non-Wordlist-Based DGA dataset. Ειδικά για την πρώτη περίπτωση, το CNN μοντέλο κρίνεται ακατάλληλο ως ανιχνευτής, αφού ακόμα και για την κατανομή χαμηλής ετερογένειας C με συχνότητα συμψηφισμού ανά 200 batches, παρουσιάζει μειωμένη ακρίβεια που προσεγγίζει το 4%. Μάλιστα, για τις ίδιες συνθήκες εκπαίδευσης η απόκλιση αυτή είναι υπερτριπλάσια από αυτήν των μοντέλων LSTM και Bidirectional LSTM που είναι 1.2% και 0.9% αντίστοιχα. Γενικά όμως, επιβεβαιώνουμε ότι ακόμα και υπό συνθήκες υψηλής ετερογένειας, όταν εκτελείται Federated Averaging ανά 200 batches, τα μοντέλα μας προσεγγίζουν τα αντίστοιχα ιδανικά, τα οποία δεν υπόκεινται σε ποινές συμψηφισμού. Υπό αυτές τις συνθήκες τα μοντέλα LSTM και Bidirectional LSTM προσεγγίζουν αποτελεσματικά τη βέλτιστη ακρίβεια, με μέγιστη απόκλιση 1.09% για εκπαίδευση στο Non-wordlist-based DGA dataset και 2.09% σε διαφορετική περίπτωση.

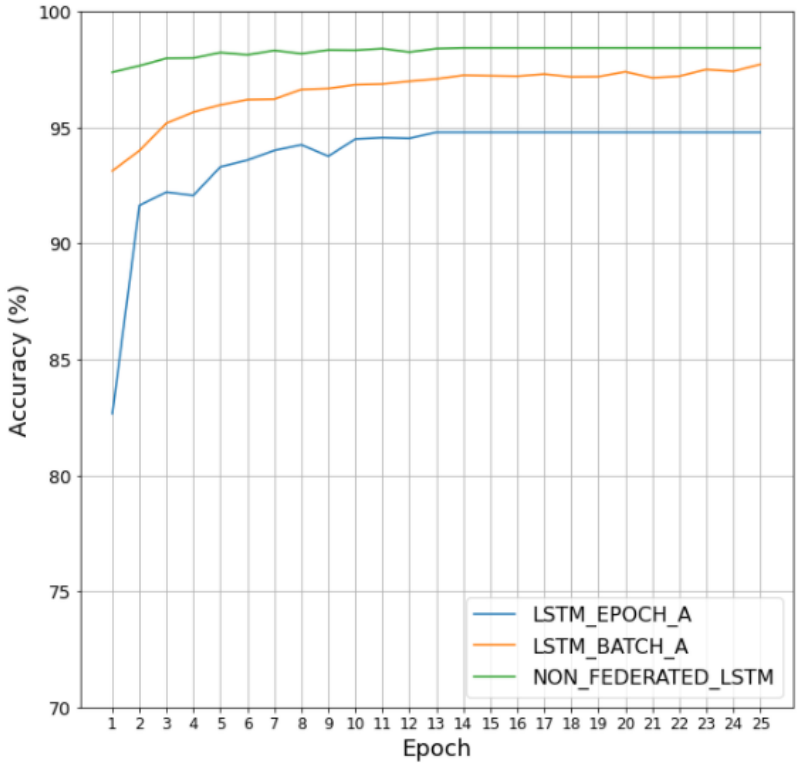
Ένα δίλημμα που προέκυψε κατά τη διάρκεια της μελέτης του συστήματος εκπαίδευσης Federated Learning που αναπτύξαμε, ήταν για το αν άξιζε η αύξηση της συχνότητας συμψηφισμού των τοπικών μοντέλων, δεδομένου ότι η ενέργεια αυτή επιφέρει επιβάρυνση κόστους επικοινωνίας πολλαπλάσια των Federated Averaging εκτελέσεων ανά εποχή, καθώς και χρονική επιβάρυνση που οφείλεται στις επιπλέον εκτελέσεις συμψηφισμού. Όπως ήδη εξηγήσαμε, το κέρδος από τη σύγκλιση των μοντέλων μας είναι από μόνο του ένα πολύ ισχυρό επιχείρημα υπέρ της αύξησης της συχνότητας συμψηφισμού. Κατά την εκτέλεση των πειραμάτων ωστόσο, διαπιστώσαμε παράλληλα, ότι το κόστος επικοινωνίας όχι μόνο δεν αυξάνεται, αλλά σε βάθος χρόνου μειώνεται κιόλας. Ο ισχυρισμός μας αυτός επαληθεύεται από τα διαγράμματα που ακολουθούν για την εκπαίδευση με Federated Learning με την κατανομή υψηλής ετερογένειας A. Σε αυτά πρέπει να παρατηρήσουμε ένα σημείο κλειδί. Αυτό είναι ότι κατά την εκπαίδευση με Federated Averaging ανά 200 batches, τα μοντέλα μας πετυχαίνουν από την πρώτη κιόλας εποχή ακρίβεια μεγαλύτερη από αυτή που πετυχαίνουν τα μοντέλα που εκπαιδεύονται με Federated Averaging ανά εποχή καθ' όλη τη διάρκεια τις διαδικασίας. Αντιπροσωπευτικό είναι το διάγραμμα 6.8, όπου το LSTM που εκπαιδεύεται με συμψηφισμό ανά εποχή, προσεγγίζει μετά από 30 εποχές την ακρίβεια που πέτυχε το ίδιο μοντέλο στην πρώτη κιόλας εποχή κατά την εκπαίδευση με συμψηφισμό ανά 200 batches. Αναφέρουμε εδώ, ότι για την εκπαίδευση του δεύτερου ανιχνευτή εκτελούνται 7 συμψηφισμοί ανά εποχή. Αυτό σημαίνει τελικά, ότι μετά την έβδομη εποχή, το κόστος επικοινωνίας κατά την εκπαίδευση με ανά εποχή συμψηφισμό, ξεπερνά το αντίστοιχο κόστος της εκπαίδευσης με συμψηφισμό ανα 200 batches. Φυσικά, το ίδιο ισχύει και για τη χρονική καθυστέρηση από τη δεύτερη κιόλας εποχή. Αξίζει να αναφέρουμε επιπλέον, ότι στα πειράματά μας παρατηρήσαμε ότι περαιτέρω αύξηση της συχνότητας εκτέλεσης του Federated Averaging ανά εποχή (π.χ. ανά 100 batches) επιδρά θετικά στα εκπαιδευόμενα μοντέλα.



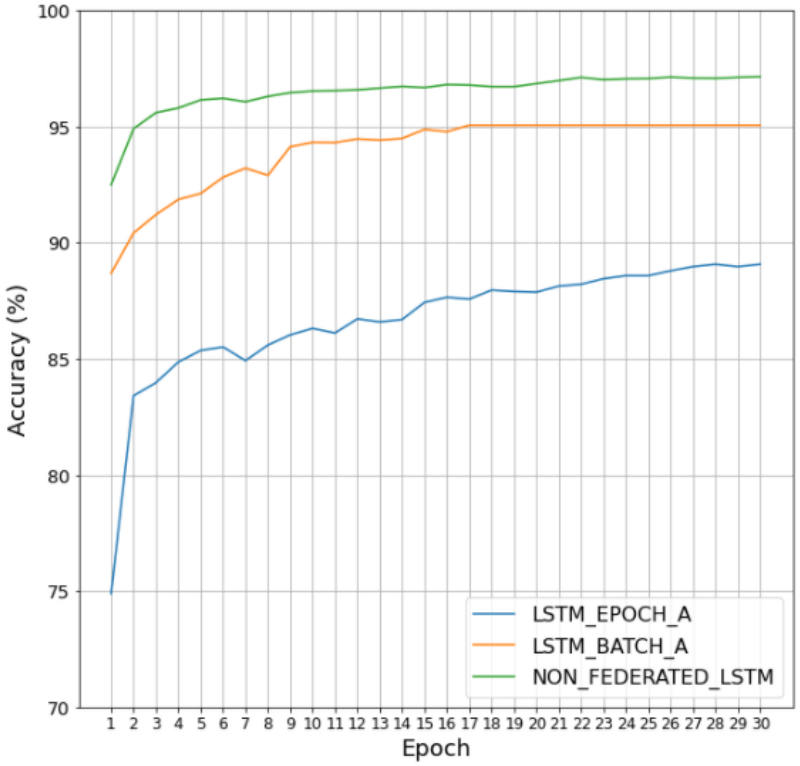
Εικόνα 6.5: *Validation-set Accuracy (%): CNN Trained on Non-Wordlist-Based DGAs Dataset 5.1*



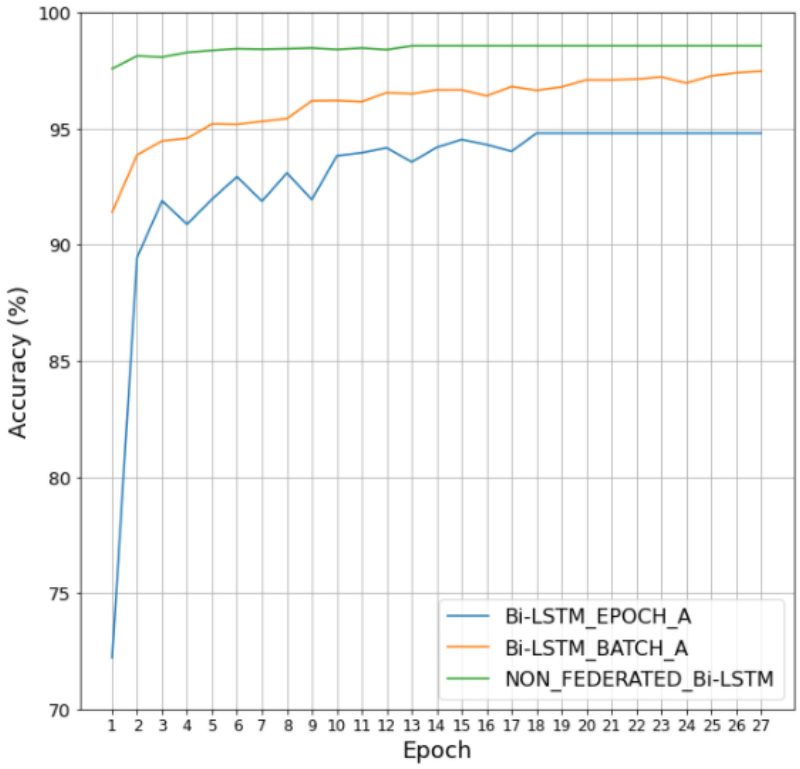
Εικόνα 6.6: *Validation-set Accuracy (%): CNN Trained on Dataset 5.2 (Wordlist-Based DGAs Included)*



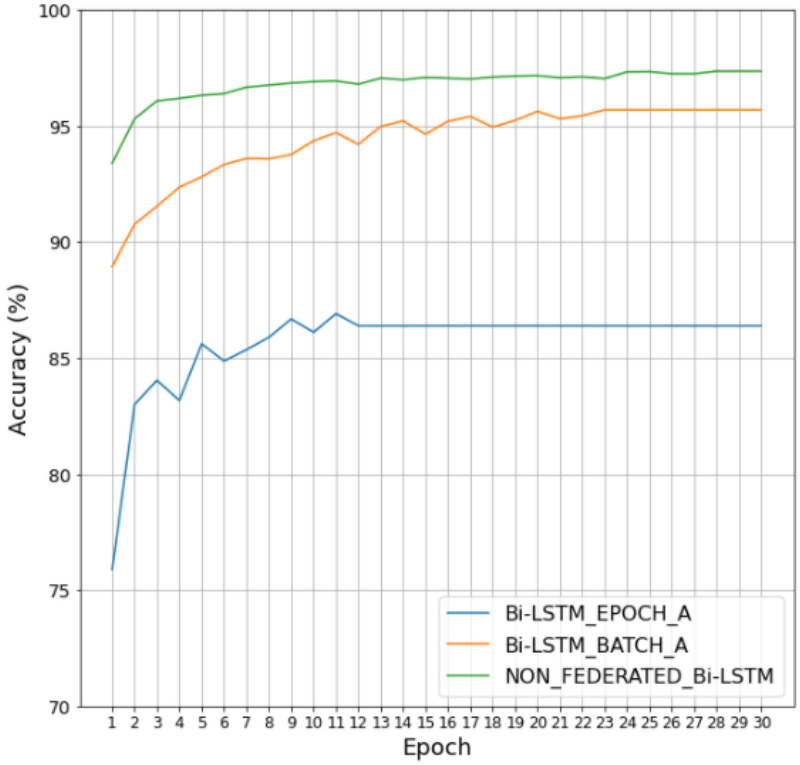
Εικόνα 6.7: *Validation-set Accuracy (%): LSTM Trained on Non-Wordlist-Based DGAs Dataset 5.1*



Εικόνα 6.8: *Validation-set Accuracy (%): LSTM Trained on Dataset 5.2 (Wordlist-Based DGAs Included)*

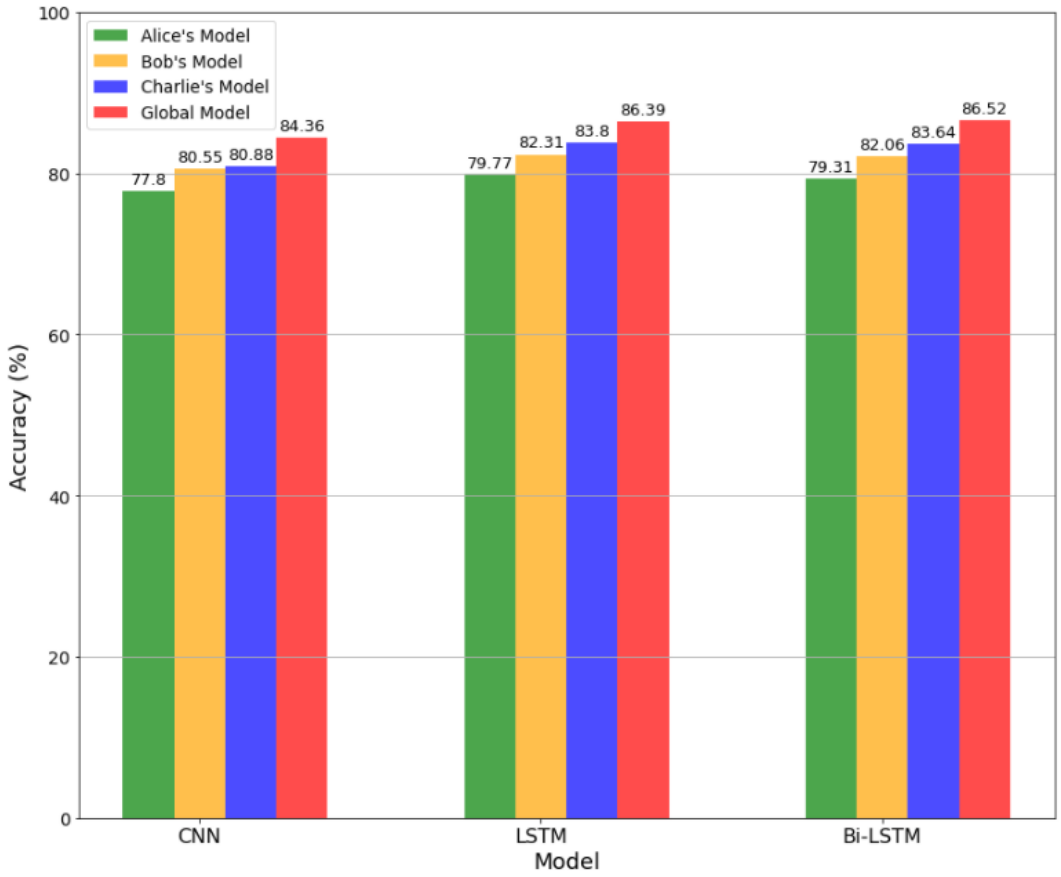


Εικόνα 6.9: *Validation-set Accuracy (%): Bi-LSTM Trained on Non-Wordlist-Based DGAs Dataset 5.1*



Εικόνα 6.10: *Validation-set Accuracy (%): Bi-LSTM Trained on Dataset 5.2 (Wordlist-Based DGAs Included)*

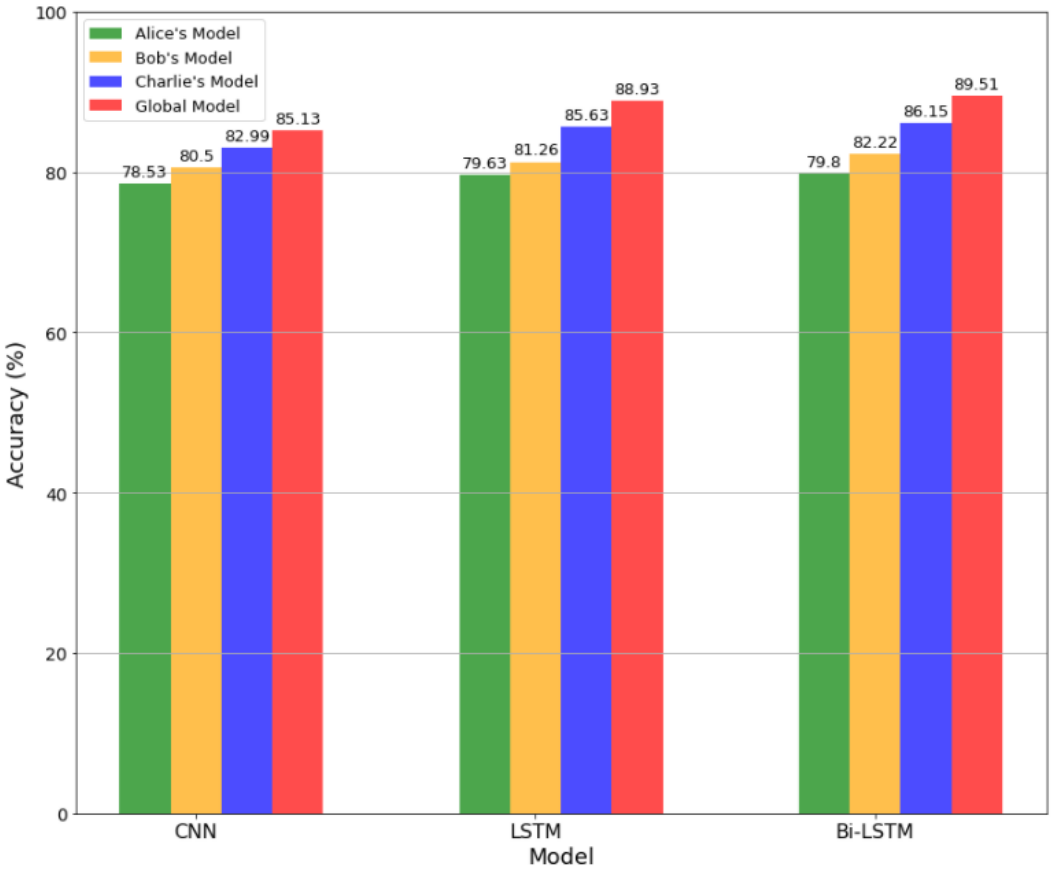
Θα αξιολογήσουμε τώρα τους Federated Learning ανιχνευτές μας που εκπαιδεύτηκαν με συμψηφισμό ανά 200 batches, μιας και εν τέλει υιοθετούμε τη συγκεκριμένη τεχνική. Οι μετρήσεις μας αφορούν τους ανιχνευτές που εκπαιδεύονται πάνω στις κατανομές υψηλής ετερογένειας A, αφού αυτό το σενάριο παρουσιάζει και την μεγαλύτερη πρόκληση. Για τον σκοπό μας χρησιμοποιούμε το Testing Dataset που κατασκευάσαμε. Αναδεικνύουμε πρώτα ένα από τα βασικά κίνητρα για τους clients να συμμετέχουν στο περιβάλλον συνεργατικής εκπαίδευσης που προτείνουμε. Συγκρίνουμε δηλαδή το accuracy των μοντέλων που θα προέκυπταν αν οι clients δεν συνεργάζονταν και χρησιμοποιούσαν μόνο τα δεδομένα που διαθέτουν οι ίδιοι (Distribution A) για τη διαδικασία της εκπαίδευσης, με την ακρίβεια των μοντέλων που εκπαιδεύονται με Federated Learning.



Εικόνα 6.11: *Per Model Accuracy Gain on Testing-set - Trained on Dataset 5.1*

	CNN		**LSTM**		**Bi-LSTM**	
	Accuracy	**Improvement**	**Accuracy**	**Improvement**	**Accuracy**	**Improvement**
Alice's Model	77.80	+ 6.56	79.77	+ 6.62	79.31	+ 7.21
Bob's Model	80.55	+ 3.81	82.31	+ 4.08	82.06	+ 4.46
Charlie's Model	80.88	+ 3.48	83.80	+ 2.59	83.64	+ 2.88
Global Federated Model	84.36	-	86.39	-	86.52	-
Non-Federated Model	83.08	-	84.92	-	85.36	-

Πίνακας 6.9: *Federated Learning: Improved Accuracy (%) of Local Models - Trained on Dataset 5.1*



Εικόνα 6.12: *Per Model Accuracy Gain on Testing-set - Trained on Dataset 5.2*

	CNN		**LSTM**		**Bi-LSTM**	
	Accuracy	**Improvement**	**Accuracy**	**Improvement**	**Accuracy**	**Improvement**
Alice's Model	78.53	+ 6.60	79.63	+ 9.30	79.80	+ 9.71
Bob's Model	80.50	+ 4.63	81.26	+ 7.67	82.22	+ 7.22
Charlie's Model	82.99	+ 2.14	85.63	+ 3.30	86.15	+ 3.36
Global Federated Model	85.13	-	88.93	-	89.51	-
Non-Federated Model	85.75	-	89.01	-	89.36	-

Πίνακας 6.10: *Federated Learning: Improved Accuracy (%) of Local Models - Trained on Dataset 5.2*

Από τα προηγούμενα διαγράμματα γίνεται ξεκάθαρο, ότι η συμμετοχή των 3 clients στο συνεργατικό περιβάλλον εκπαίδευσης Federated Learning που αναπτύξαμε, υπόσχεται σημαντικά ενισχυμένους ανιχνευτές. Πράγματι, βλέπουμε ότι για τους ανιχνευτές που εκπαιδεύονται πάνω στην κατανομή A του training dataset 5.1 (Non-wordlist DGAs), υπάρχει μια βελτίωση στην ακρίβεια από 2% περίπου στη χειρότερη περίπτωση, έως και 7% στην καλύτερη. Για τους ταξινομητές που εκπαιδεύονται πάνω στην κατανομή A του training dataset 5.2, έχουμε αντίστοιχη βελτίωση που κυμαίνεται από 2-10%. Αναλυτικά οι μετρήσεις για τα μοντέλα που εκπαιδεύονται με Federated Learning παρουσιάζονται στους πίνακες 6.11 και 6.12.

	CNN			LSTM			Bi-LSTM		
	Precision	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**
Legit	0.77	0.98	0.86	0.80	0.97	0.88	0.80	0.97	0.88
DGA	0.97	0.71	0.82	0.97	0.76	0.85	0.96	0.76	0.85
Macro Avg	0.87	0.84	0.84	0.88	0.86	0.86	0.88	0.87	0.86
Weighted Avg	0.87	0.84	0.84	0.88	0.86	0.86	0.88	0.87	0.86
Accuracy	0.8436			0.8639			0.8652		

Πίνακας 6.11: *Classification Report for Models Trained on Non-Wordlist DGA Dataset 5.1 (FedAvg per 200 Batches - Distribution A)*

	CNN			LSTM			Bi-LSTM		
	Precision	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**
Legit	0.82	0.90	0.86	0.85	0.95	0.90	0.86	0.94	0.90
DGA	0.89	0.81	0.84	0.94	0.83	0.88	0.94	0.85	0.89
Macro Avg	0.85	0.85	0.85	0.89	0.89	0.89	0.90	0.90	0.89
Weighted Avg	0.85	0.85	0.85	0.89	0.89	0.89	0.90	0.90	0.89
Accuracy	0.8513			0.8893			0.8951		

Πίνακας 6.12: *Classification Report for Models Trained on Dataset 5.2 (FedAvg per 200 Batches - Distribution A)*

Τα αποτελέσματα που προκύπτουν είναι πολύ ενθαρρυντικά. Πιο συγκεκριμένα, αναφορικά με το Federated Learning, παρατηρούμε ότι για τους ανιχνευτές που εκπαιδεύονται πάνω στο Non-wordlist DGA dataset 5.1, η ακρίβεια σε σχέση με τα αντίστοιχα Non-Federated μοντέλα όχι μόνο δεν ελαττώνεται, αλλά παρουσιάζει μια αύξηση της τάξης του 1.2-1.4%. Η μεταβολή αυτή οφείλεται ως επί το πλείστον στο αυξημένο TPR. Η συμπεριφορά αυτή πιθανώς να οφείλεται στο Federated Averaging καθαυτό. Η λογική του ισχυρισμού αυτού βασίζεται στο ενδεχόμενο, ο αλγόριθμος του Federated Averaging να προσδίδει στα μοντέλα μας ένα χαρακτηριστικό εποικοδομητικής γενίκευσης κατά τον συμψηφισμό. Η ιδέα αυτή γίνεται ακόμα πιο λογική αν σταθούμε στις αναλυτικές μετρήσεις της ανάκλησης (Recall) ανά DGA οικογένεια του πίνακα 6.13. Επαληθεύεται μια μικρή πτώση στα ποσοστά της ανάκλησης για τους DGAs που συμπεριλαμβάνονται στο training dataset, σε συνδυασμό όμως με τα βελτιωμένα ποσοστά των περισσότερων DGAs που δεν συμπεριλαμβάνονται σε αυτό.

Από την άλλη πλευρά για τους ανιχνευτές που εκπαιδεύονται πάνω στο dataset 5.2, αν εξαιρέσουμε κάποιες ελάχιστες μεταβολές, η ακρίβεια παραμένει στα ίδια επίπεδα. Αυτό γίνεται, διότι ενώ υπάρχει μια σημαντική αύξηση του TNR, μειώνεται σχεδόν ισόποσα και το FPR. Αξίζει όμως να λάβουμε υπόψη μας, ότι η μείωση του FPR οφείλεται περισσότερο στη μειωμένη διακριτική ικανότητα των Federated Learning ανιχνευτών σχετικά με τους wordlist-based DGAs. Εξαίρεση αποτελεί η οικογένεια Matsnu. Στον πίνακα που ακολουθεί παρουσιάζονται λεπτομερώς οι παρατηρήσεις αυτές.

	Models Trained on Dataset 5.1			Models Trained on Dataset 5.2		
DGA Family	**CNN**	**LSTM**	**Bi-LSTM**	**CNN**	**LSTM**	**Bi-LSTM**
Ramnit	97.24	98.54	98.25	97.15	97.70	97.99
Kraken	83.35	89.11	88.47	82.09	85.50	86.84
Simda	69.56	86.08	87.61	76.04	80.82	80.11
Banjori	0.13	85.33	65.81	99.58	99.81	99.93
Pykspa	73.39	83.01	82.31	77.44	83.87	83.26
Ramdo	99.05	99.98	99.96	97.81	99.85	99.93
QakBot	98.30	99.12	99.13	98.41	98.54	98.75
Cryptolocker	98.49	99.47	99.33	98.52	99.20	98.96
Locky	95.47	97.62	97.10	95.22	95.95	96.57
CoreBot	99.96	99.80	99.88	99.84	99.49	99.78
DirCrypt	98.05	99.04	98.79	98.10	98.28	98.49
Matsnu	20.24	1.92	3.40	96.83	96.13	96.54
Suppobox	0.31	0.67	1.04	59.35	72.90	82.54
Gozi	5.40	6.93	13.44	61.61	64.39	71.19
Nymaim v2	1.19	1.56	1.47	44.70	41.95	57.56
Pizd	0.26	0.51	0.30	54.43	45.84	50.67
Ranbyus	99.53	99.80	99.68	99.73	99.84	99.76
MoneroDownloader	44.07	18.19	43.93	4.07	13.33	17.78
Symmi	52.87	63.60	66.01	58.71	57.03	62.98
Emotet	99.44	99.76	99.73	99.62	99.76	99.79
Fobber	95.71	97.51	97.16	95.76	96.01	96.10
Pushdo	30.59	36.93	33.47	38.35	26.76	32.70
Qadars	69.36	92.80	94.52	52.20	86.59	83.12
Necurs	96.50	98.11	97.71	96.41	96.87	97.28
Conficker	49.44	54.53	54.47	47.36	51.85	52.83
Tinba	98.65	99.42	99.24	98.66	98.98	98.95
Murofet	99.81	99.57	99.93	96.64	99.82	99.90
Rovnix	99.44	99.83	99.93	97.30	99.72	99.90
Shiotob	76.10	85.51	90.27	68.15	76.35	68.52
Proslikefan	81.60	88.08	87.56	79.79	83.92	85.89
Padcrypt	92.31	97.84	97.81	89.11	96.88	97.92
GameoverZeus	100.00	98.41	100.00	96.64	97.88	99.59
DnsChanger	95.24	97.50	97.13	95.50	95.67	95.86
MyDoom	83.90	91.47	88.30	87.15	86.85	89.12

Πίνακας 6.13: *Recall (%) Achieved for each DGA Family on Testing Dataset with Federated Models (FedAvg per 200 Batches - Distribution A)*

Κεφάλαιο 7

Επίλογος

7.1 Συμπεράσματα

Σε αυτήν την διπλωματική εργασία, αφού εξετάσαμε τη φιλοσοφία και τη δομή των DGA-based botnets, μελετήσαμε τις υπάρχουσες τεχνικές για την ανίχνευση DGA domain names. Στην πορεία επικεντρωθήκαμε στις πιο πρόσφατες έρευνες που εκμεταλλεύονται το γεγονός, ότι η μορφή των domain names που παράγονται από DGAs διαφέρει σημαντικά από αυτήν των legit domain names (με εξαίρεση τους wordlist-based DGAs), για την εκπαίδευση Deep Learning μοντέλων-ανιχνευτών. Από τη μελέτη μας καταλήξαμε στο ότι οι ανιχνευτές που βασίζονται σε τεχνικές Deep Learning, υποστηρίζοντας τη δυνατότητα ακριβών προβλέψεων για μεμονωμένα domain names σε πραγματικό χρόνο, καθώς και την αυτόματη εξαγωγή χαρακτηριστικών κατά την εκπαίδευση, είναι ιδανικοί για την ανίχνευση κίνησης DGA. Επιπροσθέτως, οι προοπτικές κλιμακωσιμότητας (scalability) και η απλότητα των υπό διερεύνηση ανιχνευτών, τους καθιστά κατάλληλους για εγκατάσταση σε πραγματικά συστήματα ασφαλείας.

Ωστόσο, όπως εξηγήσαμε, ο εντοπισμός των C&C servers και κατ' επέκταση η εξάρθρωση ενός DGA-based botnet συχνά απαιτεί τεράστιο όγκο πληροφοριών και συνεπώς τη συνεργασία πολλών οργανισμών που ασχολούνται με υπηρεσίες του διαδικτύου. Για να δημιουργήσουμε το κίνητρο προς αυτήν την κατεύθυνση, προτείναμε και αναπτύξαμε μια πειραματική διάταξη για τη συνεργατική εκπαίδευση των ταξινομητών χρησιμοποιώντας τη σύγχρονη αρχιτεκτονική του Federated Learning. Η μέθοδος αυτή σέβεται πλήρως την ιδιωτικότητα των δεδομένων με τα οποία συμβάλλει κάθε client στην εκπαίδευση, ικανοποιώντας έτσι την απαίτηση για ενισχυμένους ανιχνευτές σε συνδυασμό όμως με τη διατήρηση του απορρήτου. Παράλληλα, κατά την εκπαίδευση με Federated Learning μειώνεται και το κόστος επικοινωνίας και οι καθυστερήσεις που προκύπτουν από τη μεταφορά των δεδομένων σε κεντρικούς servers σε Non-Federated αρχιτεκτονικές.

Αντλώντας έμπνευση από τη μελέτη των Yu et al. [47], αναπτύξαμε 3 ανιχνευτές Deep Learning (CNN, LSTM, Bi-LSTM) και τους εκπαιδεύσαμε στο πειραματικό περιβάλλον Federated Learning που σχεδιάσαμε. Παρατηρήσαμε ότι το LSTM και το Bidirectional LSTM υπερτερούν του CNN μοντέλου. Επίσης, αν και η απόδοσή τους δεν διαφέρει σημαντικά, το Bidirectional LSTM φαίνεται να πετυχαίνει καλύτερη ακρίβεια από το LSTM, με την αντίστοιχη βέβαια επιβάρυνση σε μνήμη και χρονική καθυστέρηση. Για τον λόγο αυτό προτείνεται η

χρήση του LSTM για τον εντοπισμό κακόβουλων ονομάτων σε πραγματικό χρόνο και η χρήση του Bidirectional LSTM σε διαφορετική περίπτωση. Συμπερασματικά, τα αποτελέσματα που προέκυψαν κατά την εκπαίδευση είναι πολύ ενθαρρυντικά και υποδεικνύουν τις προοπτικές εξέλιξης που υπάρχουν για την ανάπτυξη ενισχυμένων ανιχνευτών που θα εκπαιδεύονται περιοδικά πάνω σε μια τεράστια ποικιλία από domain names και θα παραμένουν ενημερωμένοι και έτοιμοι να ανακαλύπτουν καινούριες οικογένειες DGA. Πιο συγκεκριμένα, αυξάνοντας τη συχνότητα με την οποία εκτελείται ο συμψηφισμός των τοπικών μοντέλων στον aggregating server (με Federated Averaging), καταφέραμε να περιορίσουμε δραστικά την αρνητική επίδραση της ετερογένειας των δεδομένων των clients και να πετύχουμε σύγκλιση των μοντέλων μας πολύ κοντά στα ιδανικά αντίστοιχα που εκπαιδεύονται με παραδοσιακές τεχνικές Non-Federated Learning. Τέλος, όταν συμπεριλαμβάνουμε wordlist-based DGAs στην εκπαίδευση, αυξάνεται το κέρδος σε ακρίβεια και πιο συγκεκριμένα σε TPR. Η βελτίωση αυτή όμως συνοδεύεται και με αύξηση του FPR καθιστώντας τους ανιχνευτές μας ακατάλληλους για real-time αποκοπή κίνησης.

7.2 Μελλοντικές Επεκτάσεις

Το σύστημα που αναπτύχθηκε στα πλαίσια αυτής της διπλωματικής εργασίας θα μπορούσε να βελτιωθεί και να επεκταθεί μελλοντικά, τουλάχιστον ως προς τρεις κατευθύνσεις. Συγκεκριμένα, αναφέρονται τα ακόλουθα:

- Όπως πιθανώς να έγινε αντιληπτό, για την εκπαίδευση των ανιχνευτών μας θα μπορούσαν να χρησιμοποιηθούν πολύ περισσότερες οικογένειες DGA βελτιώνοντας έτσι τη διακριτική ικανότητά τους. Ωστόσο, συνειδητά επιλέξαμε να μην συμπεριλάβουμε επιπλέον οικογένειες στο training dataset για να εξετάσουμε δυνητικά την ικανότητα των ανιχνευτών μας να ανακαλύπτουν καινούριες οικογένειες και να γενικεύουν αποδοτικά στις ήδη υπάρχουσες. Φυσικά, σε μελλοντική μας έρευνα δεν θα περιοριστούμε στο κομμάτι αυτό, καθώς θα στοχεύσουμε στη βέλτιστη δυνατή παροχή πληροφορίας κατά την εκπαίδευση.

- Στην υποενότητα 2.5.4 αναφερθήκαμε σε δύο ενισχυμένους αλγορίθμους συμψηφισμού, τον FedProx και τον FedMa. Θα ήταν ενδιαφέρον να τους χρησιμοποιήσουμε, αντί του FedAvg με σκοπό να μελετήσουμε πιθανή βελτίωση στη σύγκλιση των μοντέλων μας κατά την εκπαίδευση με Federated Learning.

- Έχουμε αναφέρει ότι το PySyft framework παρέχει δυνατότητες προσαρμογής differential privacy στη διαδικασία εκμάθησης του Federated Learning. Σε ένα σύστημα συνεργατικής εκπαίδευσης που η φιλοσοφία του βασίζεται στην προστασία του απορρήτου των δεδομένων, είναι ζωτικής σημασίας να μελετήσουμε επιπλέον τρόπους να το θωρακίσουμε από τυχόν κακόβουλους.

Φυσικά οι επεκτάσεις που προτείνουμε για μελλοντική έρευνα, θα εφαρμοστούν σε ένα περιβάλλον εκπαίδευσης με πολλούς περισσότερους clients (πιθανώς αρκετές δεκάδες η ακόμα και εκατοντάδες), ώστε να παραπέμπει σε ένα πραγματικό σενάριο Federated Learning,

στο οποίο θα συμμετέχει ένας μεγάλος αριθμός από recursive DNS servers με τεράστιο όγκο εγγραφών domain names έκαστος.

Τέλος, μια πολύ ενδιαφέρουσα κατεύθυνση για μελλοντική έρευνα πάνω στο αντικείμενο που μελετούμε, έχει προταθεί ήδη σε πολύ πρόσφατες (2019), σχετικές μελέτες [63], [64], [65] και αφορούν αποκεντρωμένες αρχιτεκτονικές Federated Learning. Στην εργασία τους [64] οι Lalitha et al. προτείνουν μια αποκεντρωμένη αρχιτεκτονική Federated Learning, στην οποία ο κεντρικός aggregating server απουσιάζει. Σε αντίθεση με το παραδοσιακό Federated Learning ο συμψηφισμός των τοπικών μοντέλων δεν εκτελείται σε έναν κεντρικό server, αλλά σε κάθε client ξεχωριστά. Πιο συγκεκριμένα, κάθε client που συμμετέχει στην εκπαίδευση, συμψηφίζει το δικό του τοπικό μοντέλο με αυτά των γειτονικών μόνο clients κι έτσι οι ενημερώσεις των τοπικών μοντέλων «διαδίδονται» σταδιακά σε όλους τους συμμετέχοντες. Οι υποστηρικτές αυτής της αρχιτεκτονικής ισχυρίζονται, ότι το τελικό γενικό μοντέλο που προκύπτει για κάθε client συγκλίνει ικανοποιητικά με το αντίστοιχο του παραδοσιακού Federated Learning. Παράλληλα εξηγούν ότι η μέθοδος αυτή παρέχει σημαντική ευελιξία στο περιβάλλον εκπαίδευσης και αποδοτικότερη χρήση του bandwidth του δικτύου.

Θα επιθυμούσαμε λοιπόν να εξετάσουμε μια παρόμοια αρχιτεκτονική σε μελλοντική μας εργασία, ώστε η διαδικασία της εκπαίδευσης να καταστεί ανεξάρτητη από την παρουσία του κεντρικού aggregating server, τον οποίο οι clients εμπιστεύονται αναγκαστικά μέχρι τώρα. Η γνησιότητα των τοπικών μοντέλων και η αξιοπιστία ενός τέτοιου περιβάλλοντος εκπαίδευσης θα μπορούσαν να εξασφαλιστούν ιδανικά, επιστρατεύοντας την τεχνολογία του blockchain [66].

Παραρτήματα

Παράρτημα Α΄

Παράρτημα

Ο κώδικας που αναπτύξαμε για την υλοποίηση της πειραματικής διάταξης Federated Learning που εξετάσαμε στη διπλωματική αυτή είναι διαθέσιμος στο github repository https://github.com/georgesoul/DGA-Based_BotnetDetection_with_FederatedLearning. Σε αυτόν τον σύνδεσμο βρίσκονται και τα datasets που χρησιμοποιήσαμε για την εκπαίδευση και την αξιολόγηση των μοντέλων μας.

Βιβλιογραφία

[1] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee and D. Dagon. *From throw-away traffic to bots: detecting the rise of DGA- based Malware. 21th USENIX Security Symposium (USENIX Security 12)*, 2012.

[2] *HTTP-Botnets: The Dark Side of a Standard Protocol!* https://securityaffairs.co/wordpress/13747/cyber-crime/http-botnets.html. Access Date: 2-9-2020.

[3] R. Perdisci, W. Lee and N. Feamster. *Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces. USENIX Symposium on Networked Systems Design Implementation (NSDI 2010)*, 2010.

[4] J. Goebel and T. Holz. *Rishi: Identify Bot Contaminated Hosts by IRC Nickname Evaluation. In USENIX HotBots*, 2007.

[5] B. Stock, J. Göbel, M. Engelberth, F. C. Freiling and T. Holz. *Walowdac - Analysis of a Peer-to-Peer Botnet. 2009 European Conference on Computer Network Defense*, 2009.

[6] S. Stover, D. Dittrich, J. Hernandez and S. Dietrich. *Analysis of the storm and nugache trojans: P2P is here. In Proc. of USENIX 2007*, 2007.

[7] D. Plohmann, K. Yakdan, M. Klatt, J. Bader and E. Gerhards-Padilla. *A Comprehensive Measurement Study of Domain Generating Malware, 25th USENIX Security Symposium (USENIX Security 16)*, 2016. pp. 263–278.

[8] T. Barabosch, A. Wichmann, F. Leder and E. Gerhards-Padilla. *Automatic extraction of domain name generation algorithms from current malware. NATO Symposium IST-111 on Information Assurance and Cyber Defense*, 2012.

[9] *What is DNS Tunneling?* https://www.plixer.com/blog/what-is-dns-tunneling/. Access Date: 2-9-2020.

[10] C. J. Dietrich, C. Rossow, F. C. Freiling, H. Bos, M.v. Steen and N. Pohlmann. *On Botnets That Use DNS for Command and Control. 2011 Seventh European Conference on Computer Network Defense*, 2011.

[11] F. Weimer. *Passive DNS replication. In Proc. of the FIRST conference on computer security incident*, 2005.

[12] Conficker Working Group. *Conficker Working Group: Lessons Learned*, Conficker Working Group, 2011. http://docplayer.net/16497189-Conficker-working-group-lessons-learned.html.

[13] T. Yang, G. Andrew, H. E. Haicheng Sun, W. Li, N. Kong, D. Ramage and F. Beaufays. *APPLIED FEDERATED LEARNING: IMPROVING GOOGLE KEYBOARD QUERY SUGGESTIONS*, Google LLC, 2018. https://arxiv.org/abs/1812.02903.

[14] H. B. McMahan, E. Moore, D. Ramage, S. Hampson and B. A. Arcas. *Communication-efficient learning of deep networks from decentralized data.* In Proc. of the 20th International Conference on Artificial Intelligence and Statistics, 2016.

[15] Peter Kairouz, H. Brendan Mcmahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis and et al. *Advances and Open Problems in Federated Learning*, Google LLC, 2019. https://hal.inria.fr/hal-02406503.

[16] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtarik, Ananda Theertha Suresh and Dave Bacon. *Federated Learning: Strategies for Improving Communication Efficiency*. NIPS Workshop on Private Multi-Party Machine Learning, 2016.

[17] F. Sattler, S. Wiedemann, K. R. Müller and W. Samek. *Robust and Communication-Efficient Federated Learning from Non-IID Data*. IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, 2020.

[18] Kevin Hsieh, Amar Phanishayee, Onur Mutlu and Phillip B. Gibbons. *THE NON-IID DATA QUAGMIRE OF DECENTRALIZED MACHINE LEARNING*. In Proc. of the International Conference on Machine Learning (ICML), 2019.

[19] Xiang Li, Wenhao Yang, Zhihua Zhang, Kaixuan Huang and Shusen Wang. *ON THE CONVERGENCE OF FEDAVG ON NON-IID DATA.* In Proc. of the International Conference on Learning Representations, 2020.

[20] T. Li, A. Kumar, S. M. Zaheer, M. Sanjabi, A. Talwalkar and V. Smith. *Federated Optimization in Heterogeneous Networks*. In Proc. of the 3rd MLSys Conference, 2019.

[21] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos and Yasaman Khazaeni. *Federated Learning with Matched Averaging*. In Proc. of the 38th Annual IEEE International Conference on Computer Communications (INFOCOM 2019), 2019.

[22] Zhibo Wang, Mengkai Song, Zhifei Zhang, Yang Song, Qian Wang and Hairong Qi. *Beyond Inferring Class Representatives: User-Level Privacy Leakage From Federated Learning*. International Conference on Learning Representations, 2020.

[23] R. Shokri and V. Shmatikov. *Privacy-preserving deep learning*. In Proc. of the 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton), 2015.

[24] R. C. Geyer, T. Klein and M. Nabi. *Differentially Private Federated Learning: A Client Level Perspective*. In Proc. of the 31st Conference on Neural Information Processing Systems (NIPS 2017), 2017.

[25] A. N. Bhagoji, S. Chakraborty, P. Mittal and S.Calo. *Analyzing Federated Learning through an Adversarial Lens*, *Proceedings of the 36th International Conference on Machine Learning*, 2019. vol. 97. pp. 634–643.

[26] Shiqi Shen, S. Tople and P. Saxena. *Auror: defending against poisoning attacks in collaborative deep learning systems.* *In Proc. of the 32nd Annual Conference on Computer Security Applications*, 2016.

[27] Yu Chen, Fang Luo, Tong Li, Tao Xiang, Zheli Liu and Jin Li. *A training-integrity privacy-preserving federated learning scheme with trusted execution environment*, *Information Sciences*, 2020. vol. 522. pp. 69–79.

[28] Kamal Alieyan, Ammar ALmomani, Ahmad Manasrah and Mohammed M. Kadhum. *A survey of botnet detection based on DNS.*, *Neural Comput Applic (2017).*, 2017. vol. 28. pp. 1541–1558.

[29] D. K. Mcgrath and M. Gupta. *"Behind Phishing: An Examination of Phisher Modi Operandi.* *LEET*, vol. 8, 2008.

[30] M. Namazifar and Y. Pan. *Research Spotlight: Detecting Algorithmically Generated Domains*, Cisco, 2015. Available: http://blogs.cisco.com/security/talos/detecting-dga.

[31] S. Yadav, A. K. K. Reddy, A. L. N. Reddy and S. Ranjan. *Detecting algorithmically generated domain-flux attacks with DNS traffic analysis.*, *IEEE/ACM Trans. Netw.*, 2012. vol. 20. pp. 1663–1677.

[32] Y. Zhou, Q. S. Li, Q. Miao and K. Yim. *DGA-Based Botnet Detection Using DNS Traffic.*, *Journal of Internet Services and Information Security*, 2013. vol. 3. pp. 116–123.

[33] T. D. Nguyen, T. D. CAO and l. G. Nguyen. *DGA Botnet detection using Collaborative Filtering and Density-based Clustering.* *SoICT 2015 Proceedings of the Sixth International Symposium on Information and Communication Technology*, 2015.

[34] S. Schiavoni, F. Maggi, L. Cavallaro and S. Zanero. *Phoenix: DGAbased botnet tracking and intelligence.* *in Proc. Int. Conf. Detection Intrusions Malware, Vulnerability Assessment*, 2014.

[35] Jonghoon Kwon, Jehyun Lee, Heejo Lee and Adrian Perrig. *PsyBoG:AscalablebotnetdetectionmethodforlargeDNStraffic.*, *Computer Networks (The International Journal of Computer and Telecommunications Networking)*, 2016. vol. 97. pp. 48–73.

[36] Han Zhang, Manaf Gharaibeh, Spiros Thanasoulas and Christos Papadopoulos. *BotDigger: Detecting DGA Bots in a Single Network.* *Proceedings of the 2016 Network Traffic Measurement and Analysis Conference (TMA)*, 2016.

[37] Tzy Shiah Wang, Hui Tang Lin, Wei Tsung Cheng and Chang Yu Chen. *DBod: Clustering and detecting DGA-based botnets using DNS traffic analysis.*, *Computers and Security*, 2016. vol. 64. pp. 1–15.

[38] S. Krishnan, T. Taylor, F. Monrose and J. McHugh. *Crossing the threshold: Detecting network malfeasance via sequential hypothesis testing.* *43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2013.

[39] Bilge L., Kirda E., Kruegel C. and Balduzzi M. *EXPOSURE: finding malicious domains using passive DNS analysis.* *In Proc. of the Network and Distributed System Security Symposium (NDSS)*, 2011.

[40] J. Raghuram, D. J. Miller and G. Kesidis. *Unsupervised, low latency anomaly detection of algorithmically generated domain names by generative probabilistic modeling.*, *Journal of Advanced Research*, 2014. vol. 5. pp. 423–433.

[41] Yu Chen, Sheng Yan, Tianyu Pang and Rui Chen. *Detection of DGA Domains Based on Support Vector Machine.* *Third International Conference on Security of Smart Cities, Industrial Control System and Communications (SSIC)*, 2018.

[42] G. Zhang J. Huang and Y. Shen. *DGA Domain Name Detection Based on SVM Under Grey Wolf optimization Algorithm.* *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)*, 2019.

[43] Hieu Mac, Duc Tran, Van Tong, Linh Giang Nguyen and Hai Anh Tran. *DGA Botnet Detection Using Supervised Learning Methods.* *In SoICT 17: 8th International Symposium on Information and Communication Technology*, 2017.

[44] Jan Spooren, Davy Preuveneers, Lieven Desmet, Peter Janssen and Wouter Joosen. *Detection of Algorithmically Generated Domain Names used by Botnets: A Dual Arms Race.* *SAC 2019: The 34th ACM/SIGAPP Symposium On Applied Computing*, 2019.

[45] Jonathan Woodbridge, Hyrum S. Anderson, Anjum Ahuja and Daniel Grant. *Predicting Domain Generation Algorithms with Long Short-Term Memory Networks*, Endgame Inc., 2016. arXiv:1611.00791 [cs.CR] https://arxiv.org/abs/1611.00791.

[46] Palak Vij, Sayali Nikam and Ashutosh Bhatia. *Detection of Algorithmically Generated Domain Names using LSTM.* *2020 12th International Conference on Communication Systems Networks (COMSNETS)*, 2020.

[47] Bin Yu, Daniel L. Gray, Jie Pan, Martine De Cock and Anderson C. A. Nascimento. *Inline DGA Detection with Deep Networks.* *2017 IEEE International Conference on Data Mining Workshops*, 2017.

[48] G. Liu, J. Zhai, Y. Dai, Z. Yan, Y. Zou and W. Huang. *A Novel Detection Method for Word-Based DGA.* *International Conference on Cloud Computing and Security (ICCCS 2018)*, 2018.

[49] Joewie J. Koh and Barton Rhodes. *Inline Detection of Domain Generation Algorithms with Context-Sensitive Word Embeddings*, Optfit LLC, 2018. arXiv:1811.08705 [cs.CR] https://arxiv.org/abs/1811.08705.

[50] R. R. Curtin, A. B. Gardner, S. Grzonkowski, A. Kleymenov and A. Mosquera. *Detecting DGA domains with recurrent neural networks and side information. 14th International Conference on Availability, Reliability and Security (ARES 2019)*, 2019.

[51] *PySyft*. https://github.com/OpenMined/PySyft. Access Date: 3-9-2020.

[52] *Secure multi-party computation*. https://en.wikipedia.org/wiki/Secure_multi-party_computation. Access Date: 30-9-2020.

[53] *TensorFlow Federated*. https://github.com/tensorflow/federated. Access Date: 3-9-2020.

[54] *FATE*. https://github.com/FederatedAI/FATE. Access Date: 3-9-2020.

[55] *PyGrid*. https://github.com/OpenMined/PyGrid. Access Date: 3-9-2020.

[56] *LZ4 (compression algorithm)*. https://en.wikipedia.org/wiki/LZ4_(compression_algorithm). Access Date: 30-9-2020.

[57] *LZ4 (compression algorithm)*. https://ticki.github.io/blog/how-lz4-works/. Access Date: 30-9-2020.

[58] *DgaDetect*. https://github.com/cckuailong/DgaDetect. Commit: ca4620ff06821cb3cdabab40d0c6effb3ccfaf58.

[59] *Domain Generation Algorithms*. https://github.com/baderj/domain_generation_algorithms. Commit: a092045416fe55a033b0a5913e584076a1e144bb.

[60] *DGA Domains Dataset*. https://github.com/chrmor/DGA_domains_dataset. Commit: 7b51f84a4db6c0828f85142df274ba3caaf9ba78.

[61] *Netlab DGA Project*. https://data.netlab.360.com/dga/. Access Date: 20-8-2020.

[62] *The DGA of Banjori*. https://johannesbader.ch/blog/the-dga-of-banjori/. Access Date: 14-9-2020.

[63] Jingyan Jiang Chenghao Hu and Zhi Wang. *Decentralized Federated Learning: A Segmented Gossip Approach*. *In Proc. of the 1st International Workshop on Federated Machine Learning for User Privacy and Data Confidentiality (FML'19)*, 2019.

[64] Tara Javidi Anusha Lalitha, Osman Cihan Kilinc and Farinaz Koushanfar. *Peer-to-peer Federated Learning on Graphs*, Department of Electrical Computer Engineering, University of California, San Diego, USA, 2019. https://arxiv.org/abs/1901.11173.

[65] Osman Kilinc Yongxi Lu Tara Javidi Anusha Lalitha, Xinghan Wang and Farinaz Koushanfar. *Decentralized Bayesian Learning over Graphs*, Department of Electrical Computer Engineering, University of California, San Diego, USA, 2019. https://arxiv.org/abs/1905.10466.

[66] *Blockchain*. https://en.wikipedia.org/wiki/Blockchain. Access Date: 1-10-2020.