Σχολή Θετικών Επιστημών και Τεχνολογίας

Πρόγραμμα Σπουδών Πληροφορικής

Πτυχιακή Εργασία

Αξιοποίηση δεδομένων TCP/IP Traffic σε Μορφή Γραφήματος Γνώσης για τη Βελτίωση της Κυβερνοασφάλειας

Πέτρος-Γιάννης Πεταλάς

Επιβλέπων καθηγητής: Βάιος Παπαϊωάννου

Αξιοποίηση δεδομένων TCP/IP Traffic σε Μορφή Γραφήματος Γνώσης για τη Βελτίωση της Κυβερνοασφάλειας

Πέτρος-Γιάννης Πεταλάς

Επιτροπή Επίβλεψης Πτυχιακής Εργασίας

Επιβλέπων Καθηγητής:

Βάιος Παπαϊωάννου

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Πανεπιστήμιο Πελοποννήσου

Συνεργαζόμενο Εκπαιδευτικό Προσωπικό (ΣΕΠ)

Ελληνικό Ανοικτό Πανεπιστήμιο (ΕΑΠ)

Συν-Επιβλέπων Καθηγητής:

Πάτρα, Μάιος 2026

Αφιερώνεται στη γυναίκα μου, για την αδιάλειπτη ηθική και συναισθηματική στήριξη που μου προσέφερε μέσα σε πολύ δύσκολες περιόδους.

Περίληψη

Η παρούσα εργασία εξετάζει μία πιθανή εφαρμογή των γραφημάτων γνώσης, σε συνδυασμό με Μεγάλα Γλωσσικά Μοντέλα(Large Language Model - LLM), για την ανίχνευση μοτίβων επίθεσης σε υπολογιστικά δίκτυα. Τα γραφήματα γνώσης συνιστούν εναλλακτικό παράδειγμα αναπαράστασης πληροφορίας, στο οποίο οι οντότητες κωδικοποιούνται ως κόμβοι και οι μεταξύ τους σχέσεις ως ακμές ενός γράφου - σε αντιδιαστολή προς τις σχεσιακές βάσεις δεδομένων, όπου η πληροφορία οργανώνεται σε πίνακες και συσχετίσεις.

Επιχειρούμε στα πλαίσια της εργασίας να μοντελοποιήσουμε την διαδικτυακή κίνηση σε επίπεδο TCP/IP βάσει ενός πρωτότυπου οντολογικού πλαισίου. Για το σκοπό της έρευνας έχει δημιουργηθεί το πρόγραμμα PcapToKG, το οποίο εκτελεί εξαγωγή πληροφορίας από αρχεία .pcap, τα εμπλουτίζει με άντληση πληροφορίας από API γεωεντοπισμού και τέλος τα αποθηκεύει σε γράφημα γνώσης. Το παραγόμενο γράφημα διασυνδέεται στη συνέχεια με LLM μέσω εξυπηρετητή MCP, επιτρέποντας την εκτέλεση ερωτημάτων σε φυσική γλώσσα προς τη βάση δεδομένων.

Η υλοποίηση έγινε με αντικειμενοστραφή προγραμματισμό σε γλώσσα Java, χρησιμοποιώντας τη μηχανή γραφημάτων Neo4j για τη διαχείριση του γραφήματος γνώσης. Ως LLM χρησιμοποιήθηκε το μοντέλο Claude της Anthropic.

Η αξιολόγηση της αποδοτικότητας της προτεινόμενης μεθόδου διενεργήθηκε μέσω σύγκρισης των εξόδων του LLM με αναμενόμενες κατηγοριοποιήσεις, για αρχεία με γνωστή εκ των προτέρων ετικέτα (κακόβουλη ή καλοήθης κίνηση). Τα προκαταρκτικά αποτελέσματα καταδεικνύουν ακρίβεια (accuracy) της τάξεως του 0,73 έως 0,83, ανάλογα με την ύπαρξη ή μη δεδομένων ασφαλείας. Ωστόσο, το μέγεθος του δείγματος παραμένει περιορισμένο, και κρίνεται αναγκαία η επανάληψη του πειράματος σε εκτενέστερο σύνολο δεδομένων για την εξαγωγή στατιστικά αξιόπιστων συμπερασμάτων, καθώς και σύγκριση με σημεία αναφοράς.

Λέξεις κλειδιά:

LLM, Γραφήματα Γνώσης, Κυβερνοασφάλεια, Δίκτυα Υπολογιστών Java, Neo4j

Abstract

This paper examines a potential application of knowledge graphs, in combination with Large Language Models (LLMs), for the detection of attack patterns in computer networks. Knowledge graphs represent an alternative method for information representation, in which entities are encoded as nodes and the relationships between them as edges of a graph - in contrast to relational databases, where information is organized into tables and relations.

Within the scope of this work, we attempt to model internet traffic at the TCP/IP level based on an original ontological framework. For the purposes of the research, the program PcapToKG was developed, which extracts information from .pcap files, enriches them by retrieving data from a geolocation API, and finally stores them in a knowledge graph. The resulting graph is then connected to an LLM through an MCP server, enabling the execution of natural language queries against the database.

The implementation was carried out using object-oriented programming in Java, utilizing the Neo4j graph engine for knowledge graph management. The LLM used was Anthropic’s Claude.

The efficiency of the proposed method was evaluated by comparing the LLM's outputs against expected classifications, using files with a known label assigned in advance (malicious or benign traffic). Preliminary results demonstrate an overall accuracy of approximately 0.73 to 0.83, depending on the presence of Security information. However, the sample size remains limited, and it is deemed necessary to repeat the experiment on a more extensive data corpus in order to draw statistically reliable conclusions, as well as comparison with baseline.

Keywords:

LLM, Knowledge Graphs, Cybersecurity, Computer Networks, Java, Neo4j

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Συντομογραφίες & Ακρωνύμια

LLM	**Large Language Model**
TCP	**Transmission Control Protocol**
IP	**Internet Protocol**
UCO	**Unified Cybersecurity Ontology**
STUCCO	**Situation and Threat Understanding by Correlating Contextual Observations**
MCP	**Model Context Protocol**
API	**Application Programming Interface**
DNS	**Domain Name System**
VPN	**Virtual Private Network**
NAT	**Network Address Translation**
DGA	**Domain Generation Algorithm**
OSI	**Open Systems Interconnection**
QUIC	**Quick UDP Internet Connections**
RDBMS	**Relational DataBase Management System**
POJO	**Plain Old Java Object**
JVM	**Java Virtual Machine**

Ηθική και Δεοντολογία Χρήσης

Στο πλαίσιο της παρούσας πτυχιακής εργασίας, η χρήση εμπορικών υπηρεσιών γεωεντοπισμού IP (IP geolocation) για την άντληση στοιχείων όπως hostname και γεωγραφική τοποθεσία πραγματοποιήθηκε αποκλειστικά για ακαδημαϊκούς και ερευνητικούς σκοπούς. Τα δεδομένα IP που αναλύθηκαν προέρχονται από ανοικτά, δημόσια διαθέσιμα αρχεία καταγραφής δικτυακής κίνησης (.pcap), τα οποία έχουν δημοσιευθεί από τους κατόχους τους για εκπαιδευτική και ερευνητική χρήση. Η επεξεργασία περιορίστηκε σε τεχνικά μεταδεδομένα δικτύου και δεν επιδιώχθηκε η ταυτοποίηση φυσικών προσώπων ούτε η συσχέτιση των διευθύνσεων IP με συγκεκριμένα άτομα· τα στοιχεία γεωεντοπισμού αντιμετωπίστηκαν ως δεδομένα σε επίπεδο δικτύου και όχι ως προσωπικά δεδομένα ταυτοποιήσιμων χρηστών. Καθ’ όλη τη διάρκεια της εργασίας τηρήθηκαν οι αρχές της ελαχιστοποίησης δεδομένων, της αναλογικότητας και του σκοπού, σύμφωνα με το πνεύμα του Γενικού Κανονισμού για την Προστασία Δεδομένων (ΓΚΠΔ / GDPR), ενώ η χρήση των εμπορικών υπηρεσιών έγινε σε συμμόρφωση με τους όρους χρήσης τους. Τα στοιχεία που προέκυψαν αναφέρονται αποκλειστικά για την επιστημονική τεκμηρίωση της μεθοδολογίας και των ευρημάτων, χωρίς πρόθεση παρακολούθησης, στοχοποίησης ή βλάβης οποιουδήποτε φορέα ή προσώπου.

1. Εισαγωγή

Η διαρκής επέκταση των υπολογιστικών δικτύων και η μεταφορά ολοένα περισσότερων υπηρεσιών σε διαδικτυακά περιβάλλοντα έχουν αυξήσει σημαντικά τον όγκο και την πολυπλοκότητα της παραγόμενης δικτυακής κίνησης. Παράλληλα, οι κυβερνοεπιθέσεις εξελίσσονται τόσο ως προς τις τεχνικές που χρησιμοποιούν όσο και ως προς την ικανότητά τους να αποκρύπτουν την κακόβουλη δραστηριότητα μέσα σε μεγάλο πλήθος φυσιολογικών επικοινωνιών. Για τον λόγο αυτό, η έγκαιρη αναγνώριση ύποπτων μοτίβων απαιτεί τη συλλογή, τη συσχέτιση και την ερμηνεία δεδομένων που προέρχονται από διαφορετικές πηγές.

Τα αρχεία καταγραφής πακέτων, όπως τα αρχεία `.pcap`, περιέχουν αναλυτικές πληροφορίες για την κίνηση ενός δικτύου. Σε αυτές περιλαμβάνονται οι διευθύνσεις IP προέλευσης και προορισμού, οι χρησιμοποιούμενες θύρες και τα πρωτόκολλα, ο χρόνος επικοινωνίας, ο αριθμός και το μέγεθος των πακέτων, καθώς και άλλα χαρακτηριστικά των δικτυακών συνδέσεων. Η εξέταση των δεδομένων αυτών μπορεί να αποκαλύψει ενδείξεις ασυνήθιστης ή κακόβουλης δραστηριότητας. Ωστόσο, η ανάλυση μεμονωμένων πακέτων και η διαχείριση μεγάλου όγκου καταγραφών αποτελούν σύνθετες διαδικασίες, ιδιαίτερα όταν απαιτείται η συσχέτιση των τεχνικών δεδομένων με εξωτερικές πληροφορίες και γνωστά μοτίβα επιθέσεων.

Μια εναλλακτική προσέγγιση είναι η αναπαράσταση των δεδομένων σε μορφή γραφήματος γνώσης. Σε ένα γράφημα γνώσης, οι βασικές οντότητες αναπαρίστανται ως κόμβοι και οι μεταξύ τους συσχετίσεις ως ακμές. Με αυτόν τον τρόπο, οι διευθύνσεις IP, οι υπολογιστές, οι θύρες, οι δικτυακές συνομιλίες, οι γεωγραφικές τοποθεσίες και τα ονόματα DNS δεν εξετάζονται ως ανεξάρτητες εγγραφές, αλλά ως συνδεδεμένα στοιχεία ενός ενιαίου πλαισίου. Η γραφοκεντρική αναπαράσταση επιτρέπει την αποτύπωση των συμφραζομένων της δικτυακής δραστηριότητας και διευκολύνει την αναζήτηση σχέσεων και μοτίβων που ενδέχεται να μην είναι άμεσα εμφανή κατά την εξέταση των αρχικών καταγραφών.

Στην παρούσα εργασία σχεδιάζεται και υλοποιείται το σύστημα **PcapToKG**, το οποίο μετασχηματίζει δεδομένα δικτυακής κίνησης από αρχεία `.pcap` σε γράφημα γνώσης. Το σύστημα αναλύει τα καταγεγραμμένα πακέτα, συγκεντρώνει τις επιμέρους επικοινωνίες σε δικτυακές συνομιλίες και εξάγει επιλεγμένες πληροφορίες σχετικές με τις διευθύνσεις IP, τις θύρες, τα πρωτόκολλα και τον όγκο της κίνησης. Στη συνέχεια, οι πληροφορίες αυτές εμπλουτίζονται με πρόσθετα δεδομένα, όπως γεωγραφική τοποθεσία, hostname, στοιχεία DNS, πιθανή χρήση VPN και διαθέσιμες ενδείξεις ασφαλείας. Τα δεδομένα οργανώνονται σύμφωνα με ένα πρωτότυπο οντολογικό πλαίσιο και αποθηκεύονται στη βάση γραφημάτων Neo4j.

Η αποθήκευση της δικτυακής πληροφορίας σε γράφημα γνώσης επιτρέπει τη συγκέντρωση διαφορετικών τεχνικών και περιγραφικών χαρακτηριστικών σε μία κοινή αναπαράσταση. Για παράδειγμα, μια διεύθυνση IP μπορεί να συνδεθεί με έναν υπολογιστή, μία γεωγραφική περιοχή, έναν πάροχο, ένα όνομα DNS, συγκεκριμένες θύρες και τις συνομιλίες στις οποίες συμμετέχει. Οι συσχετίσεις αυτές παρέχουν ένα ευρύτερο πλαίσιο για την ερμηνεία της δραστηριότητας και μπορούν να βοηθήσουν στη διάκριση μεταξύ μιας μεμονωμένης ύποπτης ένδειξης και ενός πιο σύνθετου μοτίβου συμπεριφοράς.

Παράλληλα, η ανάπτυξη των Μεγάλων Γλωσσικών Μοντέλων δημιουργεί νέες δυνατότητες για την αλληλεπίδραση με σύνθετες πηγές δεδομένων μέσω φυσικής γλώσσας. Ένα τέτοιο μοντέλο μπορεί να συνδυάζει πληροφορίες, να εντοπίζει πιθανές συσχετίσεις και να διατυπώνει επεξηγήσεις σχετικά με τα δεδομένα που εξετάζει. Η χρήση ενός LLM χωρίς πρόσβαση σε συγκεκριμένα και ελέγξιμα δεδομένα συνοδεύεται, ωστόσο, από κινδύνους, όπως η παραγωγή γενικών ή μη τεκμηριωμένων απαντήσεων. Για τον περιορισμό αυτού του προβλήματος, στην εργασία το γλωσσικό μοντέλο δεν λαμβάνει απευθείας το σύνολο των αρχικών αρχείων καταγραφής, αλλά αποκτά ελεγχόμενη πρόσβαση στο οργανωμένο γράφημα γνώσης.

Η διασύνδεση πραγματοποιείται μέσω εξυπηρετητή **Model Context Protocol (MCP)**. Μέσω του MCP, το μοντέλο Claude της Anthropic μπορεί να υποβάλλει ερωτήματα προς τη Neo4j και να ανακτά επιλεκτικά τις πληροφορίες που είναι αναγκαίες για την ανάλυση κάθε καταγραφής. Έτσι, εξετάζεται η δυνατότητα ενός LLM να αξιοποιεί τις σχέσεις του γραφήματος, να συνδυάζει δεδομένα δικτυακής κίνησης και εμπλουτισμού

και να εκτιμά αν μια καταγραφή εμφανίζει ενδείξεις κακόβουλης ή φυσιολογικής δραστηριότητας.

Η προτεινόμενη προσέγγιση αξιολογείται σε αρχεία `.pcap` με γνωστή εκ των προτέρων κατηγοριοποίηση. Οι απαντήσεις του μοντέλου συγκρίνονται με τον πραγματικό χαρακτηρισμό κάθε καταγραφής και εξετάζονται με βάση τις περιπτώσεις αληθώς θετικών, ψευδώς θετικών, αληθώς αρνητικών και ψευδώς αρνητικών αποτελεσμάτων. Επιπλέον, διερευνώνται η επίδραση των πρόσθετων δεδομένων ασφαλείας, η επαναληψιμότητα των απαντήσεων και η εμφάνιση μη τεκμηριωμένων ισχυρισμών ή παραισθήσεων. Με αυτόν τον τρόπο, η εργασία δεν περιορίζεται στην τεχνική υλοποίηση του συστήματος, αλλά εξετάζει και τον βαθμό στον οποίο η συνδυασμένη χρήση γραφήματος γνώσης και LLM μπορεί να υποστηρίξει αξιόπιστα την ανάλυση δικτυακής δραστηριότητας.

Η παρούσα μελέτη αποτελεί ερευνητική απόδειξη εφικτότητας και όχι πλήρες επιχειρησιακό σύστημα ανίχνευσης εισβολών. Η προτεινόμενη μέθοδος δεν αποσκοπεί στην αντικατάσταση των εξειδικευμένων εργαλείων ανάλυσης πακέτων, των συστημάτων ανίχνευσης επιθέσεων ή της κρίσης ενός αναλυτή κυβερνοασφάλειας. Αντίθετα, διερευνά κατά πόσο ένα γράφημα γνώσης μπορεί να λειτουργήσει ως ενδιάμεσο επίπεδο οργάνωσης και συσχέτισης της δικτυακής πληροφορίας και κατά πόσο ένα LLM μπορεί να αξιοποιήσει αυτό το επίπεδο για την παραγωγή κατανοητών και τεκμηριωμένων εκτιμήσεων.

1.1 Στόχοι

Στόχος της εργασίας είναι η δημιουργία ενός συστήματος που θα παράγει ένα γράφημα γνώσης στο οποίο θα αποτυπώνονται στοχευμένες πληροφορίες που μπορούν να εξαχθούν από .pcap αρχεία βάσει ενός πρωτότυπου οντολογικού πλαισίου και η αξιολόγηση του συστήματος με τη χρήση LLM, το οποίο θα κληθεί να αποφασίσει αν κατά την καταγραφή υπάρχει πιθανή επίθεση ή όχι.

Η εξαντλητική ανάλυση πακέτων είναι μια ιδιαίτερα ακριβή διαδικασία όσον αφορά τη χρήση υπολογιστικών πόρων. Για αυτό, στην παρούσα εργασία θα εξετάσουμε κατά πόσο είναι εφικτό να διαπιστώσει ένα LLM την ύπαρξη κακόβουλου λογισμικού σε έναν υπολογιστή ή δίκτυο αν, αντί της εξαντλητικής ανάλυσης, εξάγουμε κομμάτια πληροφορίας από την καταγεγραμμένη κίνηση, τα εμπλουτίσουμε με εξωτερικές πληροφορίες και τα τοποθετήσουμε σε γράφημα γνώσης. Οι αποκρίσεις του LLM θα εξεταστούν σε πειραματικές καταγραφές με ή χωρίς την ύπαρξη κακόβουλου λογισμικού.

Από τα αποτελέσματα της πειραματικής διαδικασίας αναμένουμε να εξάγουμε συμπεράσματα για το κατά πόσο η καταγραφή σε γράφημα γνώσης είναι ικανή, σε συνδυασμό με διαθέσιμη πληροφορία για μοτίβα και τακτικές επιθέσεων, να συνεισφέρει στην αναγνώριση επιθέσεων σε υπολογιστικά συστήματα, καθώς και τους περιορισμούς που προκύπτουν από τη μέθοδο αυτή. Επιπλέον θα εξεταστεί η συμπεριφορά του μοντέλου σε φυσιολογική διαδικτυακή κίνηση, καθώς και η επαναληψιμότητα των αποτελεσμάτων.

1.2 Διάρθρωση Εργασίας

Η παρούσα εργασία οργανώνεται σε εννέα κεφάλαια, τα οποία ακολουθούν τη διαδοχική πορεία από τη θεωρητική θεμελίωση και τον σχεδιασμό του συστήματος έως την υλοποίηση, την πειραματική αξιολόγηση και την αποτίμηση των αποτελεσμάτων.

Στο **Κεφάλαιο 1** παρουσιάζονται το αντικείμενο, το ερευνητικό πλαίσιο και οι βασικοί στόχοι της εργασίας. Περιγράφεται συνοπτικά η προτεινόμενη προσέγγιση, σύμφωνα με την οποία δεδομένα δικτυακής κίνησης μετασχηματίζονται σε γράφημα γνώσης και αξιοποιούνται από Μεγάλο Γλωσσικό Μοντέλο για την αναγνώριση πιθανών ενδείξεων κακόβουλης δραστηριότητας.

Στο **Κεφάλαιο 2** αναπτύσσεται το θεωρητικό υπόβαθρο που είναι απαραίτητο για την κατανόηση της προτεινόμενης μεθόδου. Αρχικά παρουσιάζονται βασικές έννοιες των δικτύων υπολογιστών, όπως η δομή των πακέτων TCP/IP, τα αρχεία καταγραφής δικτυακής κίνησης, το DNS, ο γεωεντοπισμός διευθύνσεων IP και η χρήση εικονικών ιδιωτικών δικτύων. Στη συνέχεια εξετάζονται βασικά ζητήματα κυβερνοασφάλειας και παρουσιάζεται το πλαίσιο MITRE ATT&CK. Το κεφάλαιο ολοκληρώνεται με την παρουσίαση των οντολογιών κυβερνοασφάλειας, των γραφημάτων γνώσης και του Model Context Protocol, το οποίο χρησιμοποιείται για τη διασύνδεση του γραφήματος με το Μεγάλο Γλωσσικό Μοντέλο.

Στο **Κεφάλαιο 3** περιγράφεται ο σχεδιασμός του συστήματος **PcapToKG**. Παρουσιάζεται το οντολογικό μοντέλο που χρησιμοποιείται για την αναπαράσταση των δεδομένων, καθώς και η συνολική αρχιτεκτονική και η ροή λειτουργίας του συστήματος. Αναλύονται οι βασικές λειτουργικές μονάδες που είναι υπεύθυνες για την ανάγνωση των αρχείων `.pcap`, τη συγκέντρωση των πακέτων σε δικτυακές συνομιλίες, τον εμπλουτισμό των διευθύνσεων IP και την αποθήκευση των δεδομένων στη βάση γραφημάτων Neo4j. Παρουσιάζεται επίσης η διασύνδεση της βάσης με το LLM μέσω εξυπηρετητή MCP.

Στο **Κεφάλαιο 4** παρουσιάζονται η μεθοδολογία ανάπτυξης και τα εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση του συστήματος. Ειδικότερα, περιγράφονται το

Wireshark, η γλώσσα προγραμματισμού Java, το IntelliJ IDEA, το Maven, η βιβλιοθήκη Pcap4j, οι βιβλιοθήκες επικοινωνίας με διαδικτυακές υπηρεσίες, η PostgreSQL, η Neo4j και το μοντέλο Claude της Anthropic.

Στο **Κεφάλαιο 5** αναλύεται η υλοποίηση του συστήματος. Παρουσιάζονται τα προαπαιτούμενα εκτέλεσης και οι βασικές κλάσεις της εφαρμογής, καθώς και οι μεταξύ τους αλληλεπιδράσεις. Περιγράφεται επίσης ένα ολοκληρωμένο παράδειγμα εκτέλεσης, από την επεξεργασία ενός αρχείου καταγραφής έως τη δημιουργία του αντίστοιχου γραφήματος γνώσης. Τέλος, παρουσιάζονται η εγκατάσταση του MCP Server και η κατάλληλη διαμόρφωση του Claude για την υποβολή ερωτημάτων προς τη βάση.

Στο **Κεφάλαιο 6** περιγράφονται η μεθοδολογία αξιολόγησης και τα πειραματικά δεδομένα. Αναλύεται η διαδικασία επιλογής των αρχείων καταγραφής και καθορίζονται τα κριτήρια με βάση τα οποία αξιολογούνται οι αποκρίσεις του μοντέλου. Εξετάζονται οι περιπτώσεις αληθώς θετικών, ψευδώς θετικών, αληθώς αρνητικών και ψευδώς αρνητικών αποτελεσμάτων, καθώς και η εμφάνιση παραισθήσεων, η επίδραση των δεδομένων ασφαλείας και η επαναληψιμότητα των απαντήσεων.

Στο **Κεφάλαιο 7** παρουσιάζονται τα αποτελέσματα της πειραματικής διαδικασίας και η ερμηνεία τους. Περιλαμβάνονται τα αναλυτικά και τα συγκεντρωτικά αποτελέσματα, οι πίνακες σύγχυσης και οι παρατηρήσεις που προέκυψαν για κάθε εξεταζόμενο αρχείο. Επιπλέον, αξιολογούνται η ακρίβεια και η σταθερότητα των αποκρίσεων, η συνεισφορά των δεδομένων ασφαλείας, η ομοιομορφία των απαντήσεων και η συνολική συμβολή του γλωσσικού μοντέλου στη διαδικασία ανάλυσης.

Στο **Κεφάλαιο 8** καταγράφονται οι βασικές προκλήσεις και οι περιορισμοί της προτεινόμενης προσέγγισης. Εξετάζονται προβλήματα που σχετίζονται με την καταγραφή και την ανάλυση της δικτυακής κίνησης, την ποιότητα των δεδομένων γεωεντοπισμού και ασφάλειας, την αποθήκευση στο γράφημα γνώσης, τον οντολογικό σχεδιασμό, το περιορισμένο μέγεθος του πειραματικού δείγματος και τη

συμπεριφορά του LLM. Παράλληλα, διατυπώνονται προτάσεις για τη βελτίωση και τη μελλοντική επέκταση του συστήματος.

Τέλος, στο **Κεφάλαιο 9** παρατίθενται τα παραρτήματα της εργασίας. Σε αυτά περιλαμβάνονται ο πηγαίος κώδικας των κύριων κλάσεων της εφαρμογής, ενδεικτικές αποκρίσεις της υπηρεσίας γεωεντοπισμού, αποτελέσματα και αποκρίσεις του LLM, στοιχεία διαμόρφωσης του Claude και η αναλυτική παρουσίαση της οντολογίας. Η εργασία ολοκληρώνεται με τη βιβλιογραφία που χρησιμοποιήθηκε για τη θεωρητική και τεχνική τεκμηρίωσή της.

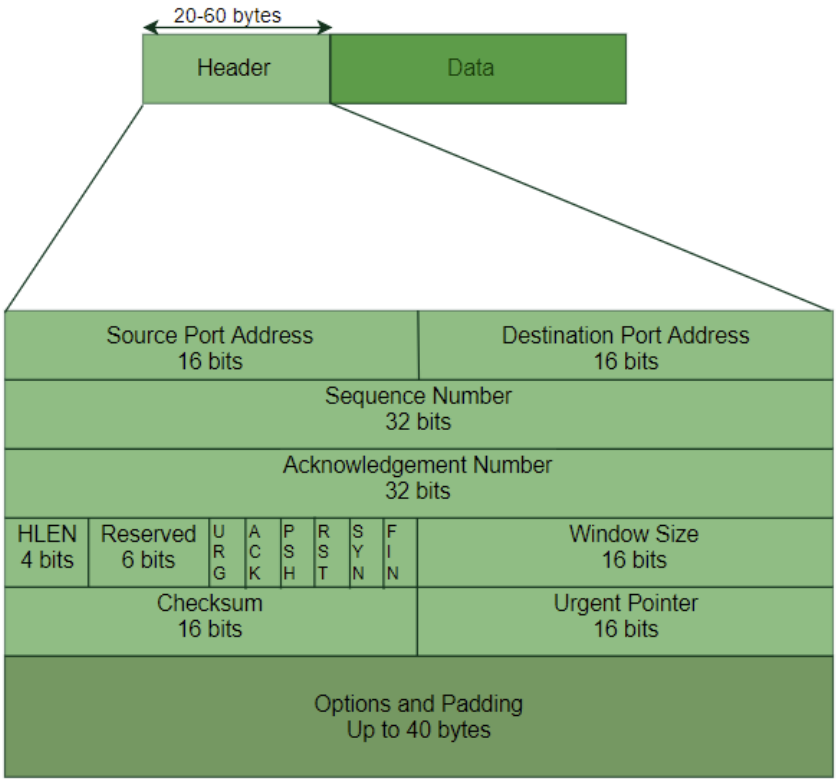
2. Υπόβαθρο

2.1 Δίκτυα Υπολογιστών

Οι βασικές τεχνολογίες και τα πρωτόκολλα που υποστηρίζουν την επικοινωνία μεταξύ υπολογιστικών συστημάτων θεωρούνται γνωστά. Στην παρούσα ενότητα παρουσιάζονται συνοπτικά μόνο εκείνα τα στοιχεία που συνδέονται άμεσα με τη σχεδίαση και τη λειτουργία του συστήματος PcapToKG. Ειδικότερα, εξετάζονται τα πεδία των πακέτων IP και των τμημάτων TCP που αξιοποιούνται κατά την επεξεργασία, ο γεωεντοπισμός διευθύνσεων IP, ο ρόλος του DNS και των VPN στην ερμηνεία της κίνησης, καθώς και η μορφή των αρχείων καταγραφής .pcap.

2.1.1 Ανατομία ενός πακέτου IP/TCP

Η δομή ενός πακέτου IP και ενός τμήματος TCP είναι ευρέως γνωστή. Ωστόσο, κρίνεται χρήσιμη μια συνοπτική αναφορά στα πεδία που παρουσιάζουν ιδιαίτερο ενδιαφέρον για την παρούσα εργασία.



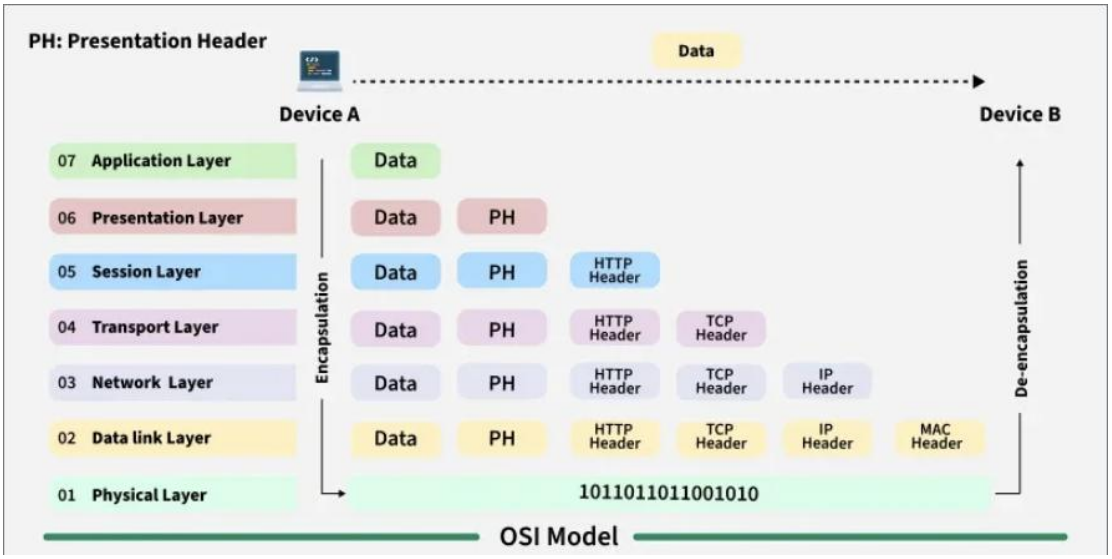
Εικόνα 1: Δομή τμήματος TCP (GeeksForGeeks)

Σε ένα τμήμα TCP ενθυλακώνονται τα δεδομένα που έχουν παραχθεί και κωδικοποιηθεί από τα ανώτερα επίπεδα της στοίβας πρωτοκόλλων. Παράλληλα, η επικεφαλίδα TCP περιλαμβάνει τα ακόλουθα βασικά πεδία:

1. Θύρα προέλευσης (Source Port).
2. Θύρα προορισμού (Destination Port).
3. Αριθμός σειράς (Sequence Number).
4. Αριθμός επιβεβαίωσης (Acknowledgement Number).
5. Μέγεθος επικεφαλίδας (HLEN).
6. Σημαίες ελέγχου (Control Flags).
7. Μέγεθος παραθύρου (Window Size).
8. Άθροισμα ελέγχου (Checksum).
9. Δείκτης επείγοντος (Urgent Pointer).

Από τα παραπάνω πεδία, ιδιαίτερο ενδιαφέρον για το σύστημα παρουσιάζουν οι θύρες προέλευσης και προορισμού, καθώς μπορούν να λειτουργήσουν ως ενδείξεις για τις υπηρεσίες ή τις εφαρμογές που συμμετέχουν σε μια επικοινωνία. Τα υπόλοιπα πεδία

δεν αξιοποιούνται άμεσα στην παρούσα υλοποίηση, όχι επειδή στερούνται γενικότερης σημασίας, αλλά επειδή δεν εντάσσονται στο επιλεγμένο μοντέλο αναπαράστασης και δεν παρέχουν, μεμονωμένα, επαρκή πληροφορία για την ταυτότητα των εμπλεκόμενων συστημάτων ή για τον σκοπό της επικοινωνίας.



Εικόνα 2: Μοντέλο OSI (GeeksForGeeks)

Αντίστοιχα, από την επικεφαλίδα IP αξιοποιούνται κυρίως οι διευθύνσεις IP προέλευσης και προορισμού, καθώς και το μέγεθος του πακέτου. Οι διευθύνσεις επιτρέπουν την αναγνώριση των εμπλεκόμενων άκρων της επικοινωνίας, ενώ το μέγεθος συμβάλλει στην εκτίμηση του όγκου δεδομένων που ανταλλάσσεται μεταξύ των αντίστοιχων υπολογιστών (hosts). Οι πληροφορίες αυτές αποτελούν τη βάση για τη συγκρότηση των συνομιλιών που αποτυπώνονται αργότερα στο γράφημα γνώσης.

2.1.2 Γεωεντοπισμός IP και χρήση Εικονικών Ιδιωτικών Δικτύων (VPN)

Όταν είναι διαθέσιμη μια διεύθυνση IP, μπορούν να χρησιμοποιηθούν εμπορικές ή ανοικτές βάσεις δεδομένων και διεπαφές προγραμματισμού εφαρμογών (APIs), οι οποίες παρέχουν εκτιμήσεις για τη γεωγραφική της τοποθεσία. Οι υπηρεσίες αυτές αξιοποιούν δεδομένα που προέρχονται, μεταξύ άλλων, από παρόχους υπηρεσιών διαδικτύου (Internet Service Providers - ISPs). Χαρακτηριστικό παράδειγμα αποτελεί η προσέγγιση Geofeeds, μέσω της οποίας οι πάροχοι μπορούν να δημοσιεύουν πληροφορίες για τις γεωγραφικές περιοχές που αντιστοιχούν στα σύνολα διευθύνσεων IP που διαχειρίζονται (Kline et al., 2020).

Η ακρίβεια του γεωεντοπισμού IP είναι περιορισμένη και συνήθως δεν υπερβαίνει το επίπεδο της πόλης, της ευρύτερης γεωγραφικής περιοχής ή της χώρας. Κατά κανόνα, δεν παρέχει αξιόπιστη πληροφορία σε επίπεδο γειτονιάς ή οδού. Στην παρούσα εργασία, ωστόσο, το ζητούμενο δεν είναι η ακριβής φυσική τοποθέτηση μιας διεύθυνσης IP, αλλά η παραγωγή πρόσθετων συμφραζομένων που μπορούν να συνεκτιμηθούν μαζί με τα υπόλοιπα χαρακτηριστικά της δικτυακής δραστηριότητας.

Για παράδειγμα, ένας οργανισμός που δραστηριοποιείται στην Αθήνα είναι αναμενόμενο να λαμβάνει σημαντικό μέρος της κίνησής του από διευθύνσεις IP που γεωεντοπίζονται στην Αττική, ενώ είναι φυσιολογικό να εμφανίζονται κατά διαστήματα και επικοινωνίες από άλλες χώρες. Ένας ασυνήθιστα υψηλός όγκος κίνησης από γεωγραφική περιοχή που δεν σχετίζεται εμφανώς με τη δραστηριότητα του οργανισμού μπορεί να αποτελέσει ένδειξη προς περαιτέρω διερεύνηση. Η ένδειξη αυτή δεν συνιστά αυτοτελή απόδειξη κακόβουλης ενέργειας, αλλά αποκτά αξία όταν συνδυάζεται με τις θύρες, τη συχνότητα των επικοινωνιών, τα χαρακτηριστικά των συνομιλιών και τα υπόλοιπα διαθέσιμα δεδομένα ασφαλείας.

Τα τελευταία χρόνια έχει επίσης εδραιωθεί η χρήση Εικονικών Ιδιωτικών Δικτύων (Virtual Private Networks - VPNs) (Rames, Vyas, & Ensafi, 2022). Μέσω ενός VPN, η δικτυακή κίνηση του χρήστη δρομολογείται μέσω εξυπηρετητών του παρόχου, οι οποίοι ενδέχεται να βρίσκονται σε διαφορετική γεωγραφική περιοχή. Κατά συνέπεια, η δημόσια διεύθυνση IP που παρατηρείται από μια απομακρυσμένη υπηρεσία δεν αντιστοιχεί απαραίτητα στην πραγματική τοποθεσία του χρήστη. Η χρήση VPN είναι απολύτως θεμιτή για λόγους ιδιωτικότητας ή ασφαλούς απομακρυσμένης πρόσβασης· ταυτόχρονα, όμως, μπορεί να αξιοποιηθεί και για την απόκρυψη της προέλευσης μιας επίθεσης. Για τον λόγο αυτό, η ένδειξη χρήσης VPN αντιμετωπίζεται ως πληροφορία συμφραζομένων και όχι ως άμεσος χαρακτηρισμός κακόβουλης δραστηριότητας.

2.1.3 DNS και η θέση του στην ασφάλεια υπολογιστών

Το Domain Name System (DNS) επιτρέπει την αντιστοίχιση αναγνώσιμων ονομάτων περιοχής σε διευθύνσεις IP. Παράλληλα, μπορεί να αξιοποιηθεί και στο πλαίσιο κακόβουλων ενεργειών. Ενδεικτικά, οι αλγόριθμοι δημιουργίας ονομάτων περιοχής (Domain Generation Algorithms - DGAs) παράγουν αυτοματοποιημένα

μεγάλο αριθμό ονομάτων, τα οποία μπορούν να χρησιμοποιηθούν για την επικοινωνία μολυσμένων συστημάτων με εξυπηρετητές διοίκησης και ελέγχου (command-and-control) (Drichel et al., 2021).

Η συσχέτιση μιας διεύθυνσης IP με όνομα υπολογιστή (hostname) είναι δυνατή όταν υπάρχει αντίστοιχη εγγραφή αντίστροφης αναζήτησης (reverse lookup) στον εξυπηρετητή DNS. Παρόμοια πληροφορία μπορεί να παρέχεται και από ορισμένα APIs γεωεντοπισμού. Το όνομα DNS που συσχετίζεται με μια διεύθυνση IP δεν επαρκεί από μόνο του για την αξιολόγησή της, μπορεί όμως να χρησιμοποιηθεί για τη διασταύρωση των υπόλοιπων στοιχείων ασφαλείας και για την καλύτερη κατανόηση του ρόλου της στην καταγεγραμμένη επικοινωνία.

Για παράδειγμα, ορισμένες διευθύνσεις IP που ανήκουν σε παρόχους υπηρεσιών νέφους ενδέχεται να έχουν καταχωριστεί σε πηγές φήμης ως κακόβουλες, επειδή στο παρελθόν χρησιμοποιήθηκαν από διαφορετικό σύστημα ή χρήστη. Αυτό μπορεί να οδηγήσει σε ψευδώς θετικές εκτιμήσεις για νόμιμες υπηρεσίες που φιλοξενούνται στην ίδια υποδομή. Η ύπαρξη αναγνωρίσιμου hostname ή εγγραφής αντίστροφης αναζήτησης προσθέτει επομένως χρήσιμα συμφραζόμενα, χωρίς να αποτελεί από μόνη της εγγύηση νομιμότητας. Αντίστοιχα, ονόματα περιοχής που παράγονται με ασυνήθιστα ή απρόβλεπτα μοτίβα (Drichel et al., 2025), καθώς και η απουσία αναμενόμενων εγγραφών DNS, μπορούν να αποτελέσουν πρόσθετες ενδείξεις, οι οποίες πρέπει να αξιολογούνται σε συνδυασμό με τη συνολική δικτυακή συμπεριφορά.

2.1.4 Αρχεία καταγραφής πακέτων (.pcap)

Τα αρχεία Packet Capture (.pcap) χρησιμοποιούνται για την αποθήκευση δικτυακής κίνησης που έχει καταγραφεί μέσω τεχνικών σύλληψης πακέτων (packet sniffing). Αξιοποιούνται κυρίως από μηχανικούς δικτύων και ομάδες ασφάλειας για την ανάλυση της επικοινωνίας ενός δικτύου, τον εντοπισμό προβλημάτων συνδεσιμότητας και τη διερεύνηση πιθανών κενών ασφαλείας.

Τα αρχεία αυτά μπορούν να δημιουργηθούν με εργαλεία όπως το Wireshark και το TShark. Για κάθε πακέτο καταγράφονται σημαντικές πληροφορίες, όπως οι διευθύνσεις IP προέλευσης και προορισμού, το χρησιμοποιούμενο πρωτόκολλο, ο

χρόνος καταγραφής και, ανάλογα με τη διαδικασία σύλληψης, το ωφέλιμο φορτίο (payload) (Sysdig, 2026). Η πληρότητα των διαθέσιμων πεδίων εξαρτάται από το σημείο καταγραφής, τα εφαρμοζόμενα φίλτρα και το αν η επικοινωνία είναι κρυπτογραφημένη.

Στην παρούσα εργασία, τα αρχεία .pcap αποτελούν την πρωτογενή πηγή δεδομένων. Το σύστημα δεν επιχειρεί πλήρη επιθεώρηση του περιεχομένου κάθε πακέτου, αλλά εξάγει επιλεγμένα μεταδεδομένα και τα συγκεντρώνει σε συνομιλίες μεταξύ διευθύνσεων IP. Η επιλογή αυτή περιορίζει την πολυπλοκότητα της επεξεργασίας και επιτρέπει την αποτύπωση των βασικών σχέσεων της κίνησης στο γράφημα γνώσης.

2.2 Ασφάλεια Υπολογιστών και Δικτύων

Η ανάλυση της δικτυακής κίνησης αποκτά ουσιαστικό νόημα όταν εντάσσεται στο ευρύτερο πλαίσιο της κυβερνοασφάλειας. Στην ενότητα αυτή παρουσιάζονται συνοπτικά οι σύγχρονες προκλήσεις του πεδίου και το πλαίσιο MITRE ATT&CK, το οποίο παρέχει κοινή ορολογία για την περιγραφή τακτικών και τεχνικών επίθεσης και αξιοποιείται κατά την ερμηνεία των πειραματικών αποτελεσμάτων.

2.2.1 Η τρέχουσα κατάσταση της κυβερνοασφάλειας

Η ευρεία διάδοση των Μεγάλων Γλωσσικών Μοντέλων (Large Language Models - LLMs) έχει επιταχύνει την παραγωγή πληροφορίας και την ανάπτυξη εφαρμογών που βασίζονται στις δυνατότητές τους. Η εξέλιξη αυτή συνοδεύεται από νέες προκλήσεις για την κυβερνοασφάλεια. Από τη μία πλευρά, τα μοντέλα μπορούν να διευκολύνουν τη δημιουργία ή την προσαρμογή εργαλείων επίθεσης. Από την άλλη, χρησιμοποιούνται συχνά για ταχεία ανάπτυξη λογισμικού από χρήστες που δεν διαθέτουν πάντοτε την απαιτούμενη εμπειρία σε θέματα ασφαλούς σχεδίασης. Η πρακτική αυτή αναφέρεται συχνά ως «vibe coding» (IBM, 2025) και μπορεί να οδηγήσει σε εφαρμογές με ανεπαρκείς ελέγχους ή μη εμφανείς ευπάθειες.

Παράλληλα, τα LLMs και γενικότερα οι τεχνικές Τεχνητής Νοημοσύνης δημιουργούν νέες δυνατότητες για την υποστήριξη της κυβερνοάμυνας, όπως η ανάλυση μεγάλου όγκου συμβάντων, ο εντοπισμός πιθανών κενών ασφαλείας και η

συσχέτιση ενδείξεων κακόβουλης δραστηριότητας. Οι δυνατότητες αυτές δεν αναιρούν την ανάγκη ανθρώπινης εποπτείας, αλλά μπορούν να λειτουργήσουν συμπληρωματικά προς τα καθιερωμένα εργαλεία και την κρίση του αναλυτή.

Η ανάγκη ανάπτυξης νέων εργαλείων προστασίας των πληροφοριακών συστημάτων είναι ιδιαίτερα σημαντική, καθώς τα συστήματα αυτά αποτελούν αναπόσπαστο μέρος της σύγχρονης καθημερινότητας. Οι συνέπειες ενός κενού κυβερνοασφάλειας μπορεί να είναι εξαιρετικά σοβαρές, ιδίως όταν επηρεάζονται νοσοκομεία, στρατιωτικοί ή κυβερνητικοί οργανισμοί, αεροδρόμια, παραγωγικές μονάδες, υποδομές διαχείρισης πόρων και τραπεζικά συστήματα (Kutub et al., 2016).

Η κυβερνοασφάλεια αποτελεί ένα πολυσύνθετο πεδίο, τόσο σε εννοιολογικό όσο και σε πρακτικό επίπεδο. Περιλαμβάνει τον προσδιορισμό της κακόβουλης δραστηριότητας, την κατανόηση των διαφορετικών μορφών επίθεσης και την αποτίμηση των πιθανών επιπτώσεών τους. Παράλληλα, η διαρκής εξέλιξη του λογισμικού, του υλικού και των δικτυακών υποδομών εισάγει νέες ευπάθειες και μεταβάλλει συνεχώς την επιφάνεια επίθεσης. Η πολυπλοκότητα αυτή καθιστά ιδιαίτερα σημαντική τη σύνδεση των μεμονωμένων τεχνικών ενδείξεων με ένα οργανωμένο πλαίσιο γνώσης.

2.2.2 Το πλαίσιο MITRE ATT&CK

Το MITRE ATT&CK αποτελεί εκτενή βάση γνώσης για καταγεγραμμένες συμπεριφορές αντιπάλων, οργανωμένες σε τακτικές, τεχνικές και υποτεχνικές. Παρέχει κοινή ορολογία για την περιγραφή των στόχων και των ενεργειών που μπορεί να εμφανιστούν κατά τη διάρκεια μιας κυβερνοεπίθεσης. Η βάση οργανώνεται σε τρεις βασικούς τομείς:

- **Enterprise:** αφορά υπολογιστικά συστήματα οργανισμών, όπως εμπορικούς και κυβερνητικούς φορείς. Περιλαμβάνει κυρίως τεχνικές που στοχεύουν φυσικούς υπολογιστές, εικονικές μηχανές (Virtual Machines - VMs), δίκτυα, βάσεις δεδομένων και άλλες πληροφοριακές οντότητες.
- **Mobile:** αφορά φορητές συσκευές, όπως τα smartphones, και περιλαμβάνει τεχνικές επίθεσης που στοχεύουν προσωπικές κινητές συσκευές.

- **ICS:** αφορά βιομηχανικά συστήματα ελέγχου και περιλαμβάνει τεχνικές επίθεσης σε μονάδες παραγωγής, υποδομές διαχείρισης ενέργειας ή υδάτων και τα αντίστοιχα κέντρα ελέγχου.

Στο πλαίσιο της παρούσας εργασίας, το ενδιαφέρον επικεντρώνεται στον τομέα Enterprise, ο οποίος είναι και ο εκτενέστερος από τους τρεις (Εικόνα 3).

Εικόνα 3: Τμήμα του πίνακα MITRE ATT&CK για εταιρικά συστήματα

Στο σκέλος Enterprise του MITRE ATT&CK οι τεχνικές οργανώνονται σε δεκατέσσερις τακτικές, καθεμία από τις οποίες αντιστοιχεί σε έναν ευρύτερο επιχειρησιακό στόχο του επιτιθέμενου. Οι τακτικές που παρουσιάζονται στην εργασία είναι οι ακόλουθες:

- **Reconnaissance:** περιλαμβάνει τεχνικές συλλογής πληροφοριών για τον οργανισμό-στόχο. Σε αυτό το στάδιο, ο επιτιθέμενος αξιοποιεί δημόσια διαθέσιμες πληροφορίες από πολλαπλές πηγές.
- **Resource Development:** περιλαμβάνει τεχνικές που χρησιμοποιούνται για τη δημιουργία υποδομών και πόρων επίθεσης. Οι πόροι αυτοί μπορεί να περιλαμβάνουν υποδομές που δημιουργούνται ειδικά για την επίθεση, καθώς και ήδη παραβιασμένους λογαριασμούς, domains ή VPNs.
- **Initial Access:** περιλαμβάνει τεχνικές μέσω των οποίων ο επιτιθέμενος αποκτά αρχική πρόσβαση στα δίκτυα ή στους υπολογιστές του θύματος.

- **Execution:** περιλαμβάνει τεχνικές εκτέλεσης κακόβουλου λογισμικού στα συστήματα του θύματος. Συχνά συνδέεται με άλλες τακτικές, όπως η αναγνώριση του δικτύου ή η κλοπή δεδομένων.
- **Persistence:** περιλαμβάνει τεχνικές που επιτρέπουν στον επιτιθέμενο να διατηρήσει την πρόσβασή του παρά τις αλλαγές που θα μπορούσαν να τη διακόψουν, όπως επανεκκινήσεις συστημάτων ή αλλαγές κωδικών.
- **Privilege Escalation:** περιλαμβάνει τεχνικές απόκτησης αυξημένων δικαιωμάτων, όπως δικαιώματα διαχειριστή ή χρήστη root.
- **Defense Evasion:** περιλαμβάνει τεχνικές απόκρυψης της κακόβουλης δραστηριότητας, αποφυγής ανίχνευσης και εξουδετέρωσης ή αποδυνάμωσης μηχανισμών ασφαλείας.
- **Credential Access:** περιλαμβάνει τεχνικές υποκλοπής ή απόκτησης διαπιστευτηρίων.
- **Discovery:** περιλαμβάνει τεχνικές μέσω των οποίων ο επιτιθέμενος συλλέγει πληροφορίες για την αρχιτεκτονική του δικτύου και τα επιμέρους συστήματα του οργανισμού.
- **Lateral Movement:** περιλαμβάνει τεχνικές που επιτρέπουν την πρόσβαση ή την παραβίαση απομακρυσμένων συστημάτων μέσα στο ίδιο δίκτυο.
- **Collection:** περιλαμβάνει τεχνικές συλλογής πληροφοριών που σχετίζονται με τους στόχους του επιτιθέμενου.
- **Command and Control:** περιλαμβάνει τεχνικές επικοινωνίας με ήδη παραβιασμένα συστήματα στο δίκτυο του θύματος.
- **Exfiltration:** περιλαμβάνει τεχνικές εξαγωγής των επιθυμητών δεδομένων από τον οργανισμό.
- **Impact:** περιλαμβάνει τεχνικές που αποσκοπούν στην καταστροφή ή αλλοίωση συστημάτων και δεδομένων του οργανισμού.

Αρκετές από τις παραπάνω τεχνικές μπορούν να δημιουργήσουν αναγνωρίσιμα μοτίβα στη δικτυακή κίνηση ενός οργανισμού. Για παράδειγμα, η τεχνική T1046 - Network Service Discovery, η οποία εντάσσεται στην τακτική Discovery, απαιτεί από ένα παραβιασμένο σύστημα να αποστείλει πακέτα σε εύρη θυρών άλλων υπολογιστών. Η δραστηριότητα αυτή μπορεί να αποκαλύψει ανοικτές θύρες και υπηρεσίες και, παράλληλα, να δημιουργήσει μοτίβα που υποδεικνύουν διαδικασία σάρωσης.

Αντίθετα, ορισμένες τεχνικές δεν είναι δυνατό να ανιχνευθούν αποκλειστικά μέσω της δικτυακής κίνησης. Πολλές τεχνικές της τακτικής Reconnaissance δεν απαιτούν άμεση αλληλεπίδραση με τα συστήματα του οργανισμού-θύματος. Χαρακτηριστικό παράδειγμα είναι η τεχνική T1589 - Gather Victim Identity Information. Επιπλέον, ακόμη και τεχνικές που εκτελούνται εντός του οργανισμού δεν δημιουργούν πάντοτε σαφώς αναγνωρίσιμα δικτυακά ίχνη. Επομένως, η αντιστοίχιση μιας καταγραφής με το MITRE ATT&CK πρέπει να αντιμετωπίζεται ως διαδικασία ερμηνείας ενδείξεων και όχι ως μηχανισμός αυτόματης απόδειξης μιας επίθεσης.

Στην πειραματική διαδικασία της εργασίας, το MITRE ATT&CK χρησιμοποιείται ως πλαίσιο αναφοράς για την οργάνωση και την αιτιολόγηση των παρατηρήσεων του LLM. Η τελική εκτίμηση βασίζεται στη συνδυαστική εξέταση της δομής του γραφήματος, των χαρακτηριστικών της κίνησης και των δεδομένων εμπλουτισμού, ενώ λαμβάνεται υπόψη ότι η διαθέσιμη τηλεμετρία δικτύου καλύπτει μόνο ένα μέρος της συνολικής δραστηριότητας ενός συστήματος.

2.3 Οντολογίες, Γραφήματα Γνώσης και Model Context Protocol

Η αξιοποίηση ετερογενών δεδομένων δικτύου απαιτεί έναν τρόπο αναπαράστασης που να διατηρεί όχι μόνο τις επιμέρους τιμές, αλλά και τις σχέσεις μεταξύ τους. Οι οντολογίες παρέχουν το εννοιολογικό πλαίσιο αυτής της αναπαράστασης, τα γραφήματα γνώσης υλοποιούν τη διασύνδεση των οντοτήτων και το Model Context Protocol επιτρέπει την ελεγχόμενη πρόσβαση ενός LLM στα δεδομένα του γραφήματος.

2.3.1 Ορισμός της οντολογίας και υφιστάμενες οντολογίες κυβερνοασφάλειας

Ο όρος «οντολογία» προέρχεται από τον χώρο της μεταφυσικής. Στη σύγχρονη Επιστήμη των Υπολογιστών, όμως, μια οντολογία ορίζεται ως τυπική και ρητή περιγραφή των εννοιών ενός πεδίου γνώσης, των ιδιοτήτων που χαρακτηρίζουν κάθε έννοια και των περιορισμών που διέπουν τις ιδιότητες και τις μεταξύ τους σχέσεις (Noy & McGuinness). Στο πλαίσιο της παρούσας εργασίας, η οντολογία λειτουργεί κυρίως

ως εννοιολογικό σχήμα για τη συνεπή οργάνωση των κόμβων, των ιδιοτήτων και των σχέσεων του γραφήματος.

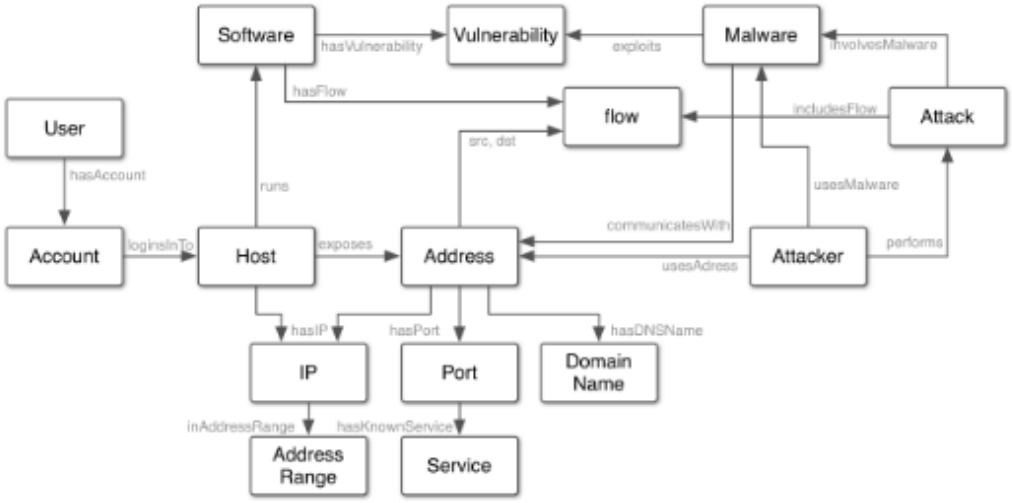
Δύο υφιστάμενες οντολογίες στον χώρο της ασφάλειας υπολογιστικών συστημάτων είναι η Unified Cybersecurity Ontology (UCO) (Syed et al., 2015) και η Situation and Threat Understanding by Correlating Contextual Observations (STUCCO) (Iannacone et al., 2015).

UCO

Η Unified Cybersecurity Ontology (UCO) αναπτύχθηκε με σκοπό την περιγραφή ενός ευρέος συνόλου οντοτήτων και σχέσεων κυβερνοασφάλειας. Σε αυτές περιλαμβάνονται οι τρόποι επίθεσης, οι επιπτώσεις, οι επιτιθέμενοι, οι ενέργειες εκμετάλλευσης, οι στόχοι και οι σχετικές ενδείξεις. Με αυτόν τον τρόπο παρέχει ένα γενικό πλαίσιο για τη μοντελοποίηση μιας κυβερνοεπίθεσης σε πολλαπλά στάδια και για τη σύνδεση τεχνικών παρατηρήσεων με ευρύτερες έννοιες ασφαλείας.

STUCCO

Η οντολογία STUCCO (Situation and Threat Understanding by Correlating Contextual Observations) έχει σχεδιαστεί για τη συγκέντρωση και τη συσχέτιση ετερογενών παρατηρήσεων κυβερνοασφάλειας μέσα σε ένα ενιαίο πλαίσιο. Στο πλαίσιο αυτό, δεδομένα δικτυακής δραστηριότητας μπορούν να συνδέονται με διευθύνσεις, υπολογιστές, domains, γνωστούς τύπους κακόβουλου λογισμικού και μοτίβα επιθέσεων. Η έμφαση στις σχέσεις μεταξύ παρατηρήσεων καθιστά τη STUCCO ιδιαίτερα συναφή με τον στόχο της παρούσας εργασίας.



Εικόνα 4: Η οντολογία STUCCO (Iannacone et al., 2015)

Η οντολογία που αναπτύσσεται στην παρούσα εργασία εστιάζει στο τμήμα της STUCCO που περιλαμβάνει τις έννοιες Host, Address, IP, Port, Domain Name και Address Range. Παράλληλα, εισάγονται πρόσθετες έννοιες που είναι απαραίτητες για την αποτύπωση των διαθέσιμων δεδομένων γεωεντοπισμού, των συνομιλιών και των στοιχείων ασφαλείας. Η επιλογή αυτή δεν αποσκοπεί στην πλήρη αναπαραγωγή της STUCCO, αλλά στην προσαρμογή ενός κατάλληλου υποσυνόλου της στις ανάγκες και στους περιορισμούς του συστήματος PcapToKG. Η αναλυτική παρουσίαση της προτεινόμενης οντολογίας πραγματοποιείται στο επόμενο κεφάλαιο.

2.3.2 Γραφήματα γνώσης και εφαρμογές

Τα γραφήματα γνώσης αποτελούν τρόπο συγκέντρωσης, οργάνωσης και αναπαράστασης πληροφορίας με έμφαση στις σχέσεις μεταξύ των δεδομένων. Μέσω ενός οντολογικού πλαισίου επιτρέπουν τη συνεπή σύνδεση οντοτήτων που προέρχονται από διαφορετικές πηγές και τη διατήρηση των συμφραζομένων τους. Σύμφωνα με τον ορισμό των Hogan et al. (2020), ένα γράφημα γνώσης αποσκοπεί στη συγκέντρωση και αναπαράσταση γνώσης από τον πραγματικό κόσμο, με τις κορυφές του να αναπαριστούν οντότητες ενδιαφέροντος και τις ακμές να αποτυπώνουν τις μεταξύ τους σχέσεις.

Γράφημα γνώσης ή σχεσιακή βάση δεδομένων;

Ένα εύλογο ερώτημα είναι γιατί να επιλεγεί ένα γράφημα γνώσης αντί μιας συμβατικής σχεσιακής βάσης δεδομένων. Οι σχεσιακές βάσεις παρέχουν ισχυρούς μηχανισμούς ακεραιότητας και είναι ιδιαίτερα αποτελεσματικές όταν το σχήμα και οι σχέσεις των δεδομένων είναι σαφώς προκαθορισμένα. Στην παρούσα περίπτωση, όμως, το κύριο ενδιαφέρον εστιάζεται στη διασύνδεση ετερογενών οντοτήτων και στη διερεύνηση μεταβαλλόμενων σχέσεων μέσα σε συνεχή, δυναμική και συχνά ελλιπή δικτυακή κίνηση. Το γραφηματικό μοντέλο αποτυπώνει αυτές τις σχέσεις με πιο άμεσο τρόπο και διευκολύνει τη διάσχιση από μια οντότητα προς τα συνδεδεμένα συμφραζόμενά της.

Ένα γράφημα γνώσης επιτρέπει επίσης την προσθήκη νέων οντοτήτων, ιδιοτήτων και συσχετίσεων με περιορισμένες αλλαγές στο υφιστάμενο μοντέλο. Η ευελιξία αυτή δεν σημαίνει απουσία σχεδίασης ή ελέγχων, αλλά δυνατότητα σταδιακής επέκτασης του σχήματος καθώς προκύπτουν νέες πηγές πληροφορίας. Επιπλέον, πρώιμα ερευνητικά αποτελέσματα δείχνουν ότι η αξιοποίηση γραφημάτων γνώσης από LLMs σε εφαρμογές εκπαιδευτικών συστάσεων μπορεί να βελτιώσει τις αποκρίσεις, περιορίζοντας την επιβάρυνση του παραθύρου συμφραζομένων (context window) και την πιθανότητα παραισθήσεων (hallucinations) (Abu-Rasheed, Weber, & Fathi, 2024). Η παρατήρηση αυτή παρέχει ένα επιπλέον κίνητρο για τη διερεύνηση της σύνδεσης γραφήματος γνώσης και LLM στο διαφορετικό πεδίο της κυβερνοασφάλειας.

Ως παράδειγμα, θεωρούνται οι οντότητες IP(address: String) και City(name: String). Σε μια σχεσιακή βάση δεδομένων, η σχέση is_located_in μπορεί να αποτυπωθεί με τον ακόλουθο κώδικα SQL:

```
CREATE TABLE is_located_in (
    IP VARCHAR(45) NOT NULL,
    City VARCHAR(255) NOT NULL,
    PRIMARY KEY (IP, City),
    FOREIGN KEY (IP) REFERENCES IP(address),
    FOREIGN KEY (City) REFERENCES City(name)
);

INSERT INTO is_located_in (IP, City)
VALUES ('192.168.1.42', 'Berlin');
```

Ο παραπάνω κώδικας δημιουργεί τον πίνακα is_located_in, ορίζει τα γνωρίσματά του, τις αναφορές προς τα κλειδιά άλλων πινάκων και τους απαραίτητους περιορισμούς. Στη συνέχεια, εισάγει μια εγγραφή που συσχετίζει τη συγκεκριμένη διεύθυνση IP με την πόλη Berlin.

Η ίδια συσχέτιση μπορεί να αποδοθεί στη γλώσσα Cypher ως εξής:

```
MERGE (:IP {address: '192.168.1.42'})-[:IS_LOCATED_IN]->(:City {name: 'Berlin'})
```

Στην περίπτωση αυτή, η σχέση εκφράζεται άμεσα ως ακμή μεταξύ των κόμβων IP και City, χωρίς να απαιτείται ξεχωριστός πίνακας συσχέτισης. Αν οι κόμβοι με τις τιμές 192.168.1.42 και Berlin δεν υπάρχουν ήδη, η εντολή MERGE τους δημιουργεί μαζί με τη σχέση που τους συνδέει. Το παράδειγμα δεν υποδηλώνει ότι το γραφηματικό μοντέλο δεν χρειάζεται περιορισμούς, αλλά αναδεικνύει ότι η σχέση αποτελεί στοιχείο πρώτης τάξης και μπορεί να αναπαρασταθεί με τρόπο πλησιέστερο στη σημασιολογία του προβλήματος.

Το παράδειγμα αναδεικνύει την ευελιξία του γραφηματικού μοντέλου στην αναπαράσταση δεδομένων. Η ευελιξία αυτή είναι ιδιαίτερα χρήσιμη όταν οι διαθέσιμες καταγραφές είναι ελλιπείς ή περιέχουν θόρυβο, όπως συμβαίνει συχνά σε πραγματικά δεδομένα δικτύου. Νέες ιδιότητες μπορούν να προστεθούν επιλεκτικά στους κόμβους για τους οποίους υπάρχουν διαθέσιμες τιμές, χωρίς να απαιτείται η συμπλήρωσή τους σε κάθε οντότητα του ίδιου τύπου. Για παράδειγμα, μια νέα διεύθυνση IP μπορεί να περιλαμβάνει, εκτός από τη διεύθυνση, και πληροφορία για τη μάσκα υποδικτύου:

```
CREATE (ip:IP {address: '192.168.1.24', subnet_mask: '255.255.255.252'})
```

Σε μια σχεσιακή βάση δεδομένων, η αποθήκευση της αντίστοιχης πληροφορίας θα απαιτούσε συνήθως την τροποποίηση του σχήματος με την προσθήκη νέας στήλης και, στη συνέχεια, την εισαγωγή της επιθυμητής τιμής. Η διαφορά αυτή είναι σημαντική για το PcapToKG, καθώς το σύνολο των διαθέσιμων χαρακτηριστικών δεν είναι

πάντοτε ίδιο για όλες τις διευθύνσεις IP και μπορεί να μεταβάλλεται ανάλογα με την απόκριση των εξωτερικών υπηρεσιών εμπλουτισμού.

2.3.3 Model Context Protocol: επικοινωνία με LLM

Το Model Context Protocol (MCP) είναι ένα πρωτόκολλο που επιτρέπει τη σύνδεση εφαρμογών Τεχνητής Νοημοσύνης, όπως τα LLMs και τα ολοκληρωμένα περιβάλλοντα ανάπτυξης (Integrated Development Environments - IDEs), με εξωτερικά δεδομένα και εργαλεία. Η σύνδεση πραγματοποιείται μέσω σαφώς καθορισμένων λειτουργιών που εκτίθενται από έναν εξυπηρετητή MCP, ώστε το μοντέλο να μπορεί να αναζητεί πληροφορίες ή να εκτελεί επιτρεπόμενες ενέργειες με ελεγχόμενο τρόπο.

Ένας προγραμματιστής μπορεί να υλοποιήσει έναν εξυπηρετητή MCP για την εφαρμογή του, ώστε ένα μοντέλο Τεχνητής Νοημοσύνης να χρησιμοποιεί τις λειτουργίες που εκτίθενται από τον εξυπηρετητή και να εκτελεί συγκεκριμένες ενέργειες (Model Context Protocol, 2026).

Η χρήση του MCP παρουσιάζει σημαντικά πλεονεκτήματα σε σχέση με την άμεση και απεριόριστη πρόσβαση μέσω εργαλείων γραμμής εντολών (Command Line Interface - CLI). Στο μοντέλο παρέχεται μόνο η πρόσβαση που απαιτείται για τις προκαθορισμένες λειτουργίες, ενώ τμήματα της εφαρμογής που δεν είναι αναγκαία μπορούν να παραμένουν μη διαθέσιμα. Επιπλέον, το μοντέλο ανακτά μόνο τα δεδομένα που απαιτούνται για κάθε ερώτημα, χωρίς να χρειάζεται να φορτωθεί ολόκληρο το σύνολο δεδομένων στο παράθυρο συμφραζομένων του. Τέλος, το MCP μπορεί να υποστηρίξει τη μετάφραση ερωτημάτων φυσικής γλώσσας σε ερωτήματα προς μια βάση δεδομένων (Shah Drishti, 2025).

Στην αρχιτεκτονική της παρούσας εργασίας, το MCP λειτουργεί ως το ενδιάμεσο επίπεδο μεταξύ του Claude και της Neo4j. Το LLM δεν λαμβάνει αυτούσιο το αρχείο .pcap ούτε ολόκληρο το περιεχόμενο του γραφήματος. Αντίθετα, διατυπώνει ερωτήματα και ανακτά επιλεγμένα δεδομένα, τα οποία χρησιμοποιεί για να συνθέσει την τελική του εκτίμηση σχετικά με την καταγεγραμμένη δικτυακή δραστηριότητα.

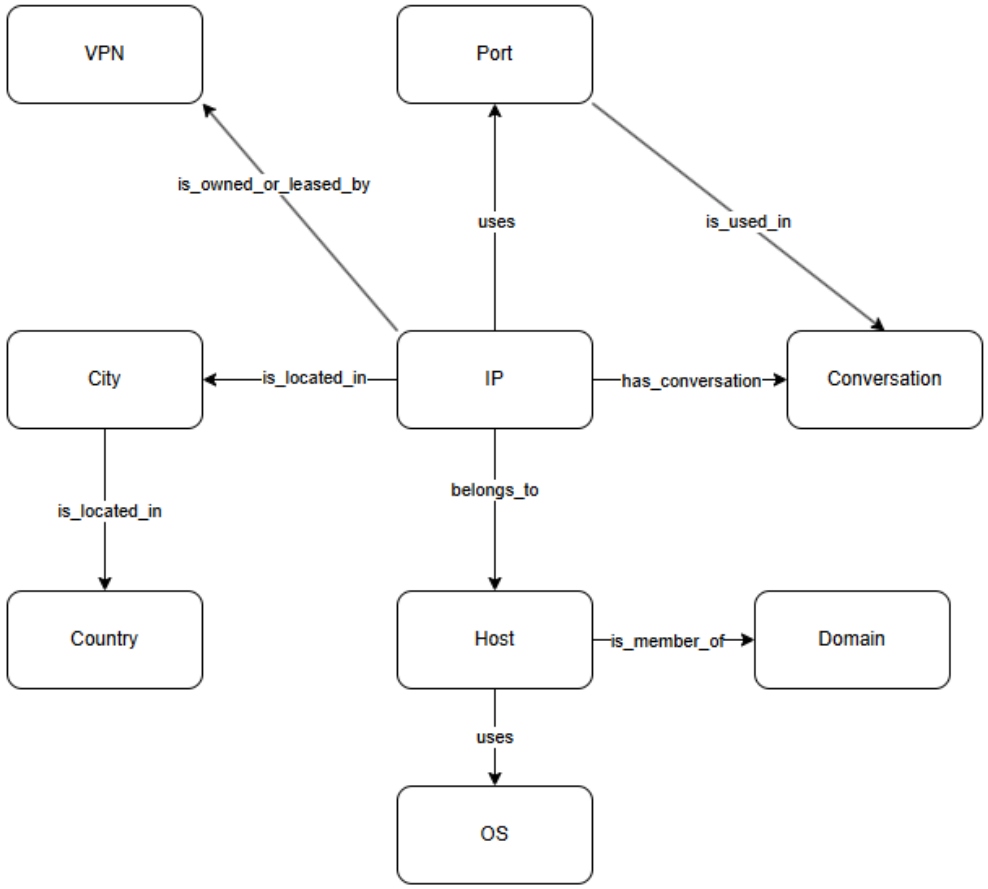
3. Σχεδιασμός συστήματος

3.1 Ορισμός Οντολογίας

Η οντολογία που αναπτύσσουμε βασίζεται στην STUCCO, με ορισμένες επεκτάσεις και ορισμένες ελλείψεις, οι οποίες οφείλονται στη διαθεσιμότητα εργαλείων και στην αδυναμία υλοποίησης στα πλαίσια της εργασίας. Η μορφή της οντολογίας φαίνεται στην Εικόνα 5.

Περιγράφουμε τη δομή ως ακολούθως:

- IP: μια οντότητα που αντιπροσωπεύει μια διεύθυνση IP
    - Host: κάθε IP αντιστοιχίζεται με κάποιον Host, χαρακτηριζόμενο από hostname. Αν δεν υπάρχει hostname, τότε αντ' αυτού χρησιμοποιούμε την IP.
        - OS: κάθε Host θεωρούμε ότι χρησιμοποιεί κάποιο λειτουργικό σύστημα
        - Domain: Εφόσον υπάρχει hostname, αφαιρούμε το πρώτο κομμάτι του FQDN για να βρούμε το άμεσο domain στο οποίο ανήκει ο υπολογιστής.
    - Conversation: μια συνομιλία μεταξύ δύο IP. Σαν όνομα θέτουμε την IP πηγής και την IP προορισμού από το πρώτο πακέτο της συνομιλίας, χωρισμένες με το σύμβολο πλην (-)
    - Port: κάθε IP χρησιμοποιεί θύρες(ports) κατά τη διάρκεια της κίνησης. Δημιουργούμε έναν κόμβο port για κάθε μία. Επίσης η θύρα συσχετίζεται και με την συνομιλία στην οποία χρησιμοποιείται.
    - City: Βάσει των δεδομένων γεωεντοπισμού, δημιουργούμε την οντότητα city που δείχνει την πόλη στην οποία βρίσκεται η IP.
        - Country: τοποθετούμε την πόλη στην αντίστοιχη χώρα.
    - VPN: για ορισμένες IP μπορεί να υπάρχουν στοιχεία για το VPN που χρησιμοποιούν. Εφόσον υπάρχουν, δημιουργείται ένας αντίστοιχος κόμβος.



Εικόνα 1:Οντολογία συστήματος

Η οντολογία βασίζεται σε τμήμα της STUCCO με αρκετές παραλλαγές. Στην παρούσα οντολογία, καθότι δεν αποτυπώνουμε μόνο επιθέσεις, αλλά το σύνολο της κίνησης σε ένα δίκτυο, δεν έχουμε οντότητες όπως Attack, Flow και Attacker. Αντιθέτως, ο χαρακτηρισμός is_known_attacker αποδίδεται, εφόσον υπάρχουν σχετικά στοιχεία από το API γεωεντοπισμού, σαν ιδιότητα της κλάσης IP. Επίσης, λόγω αντικειμενικής δυσκολίας εύρεσης σχετικών εργαλείων, δεν αποτυπώνουμε πληροφορίες για τις εφαρμογές ή υπηρεσίες που χρησιμοποιούν θύρες. Έχουν προστεθεί οι κλάσεις OS, City, Country, VPN, οι οποίες μπορούν να δώσουν περαιτέρω πληροφορίες για τη συνολική εικόνα της κίνησης.

Ο ορισμός της οντολογίας σε OWL XML μορφή δίνεται στο παράρτημα 9.5.

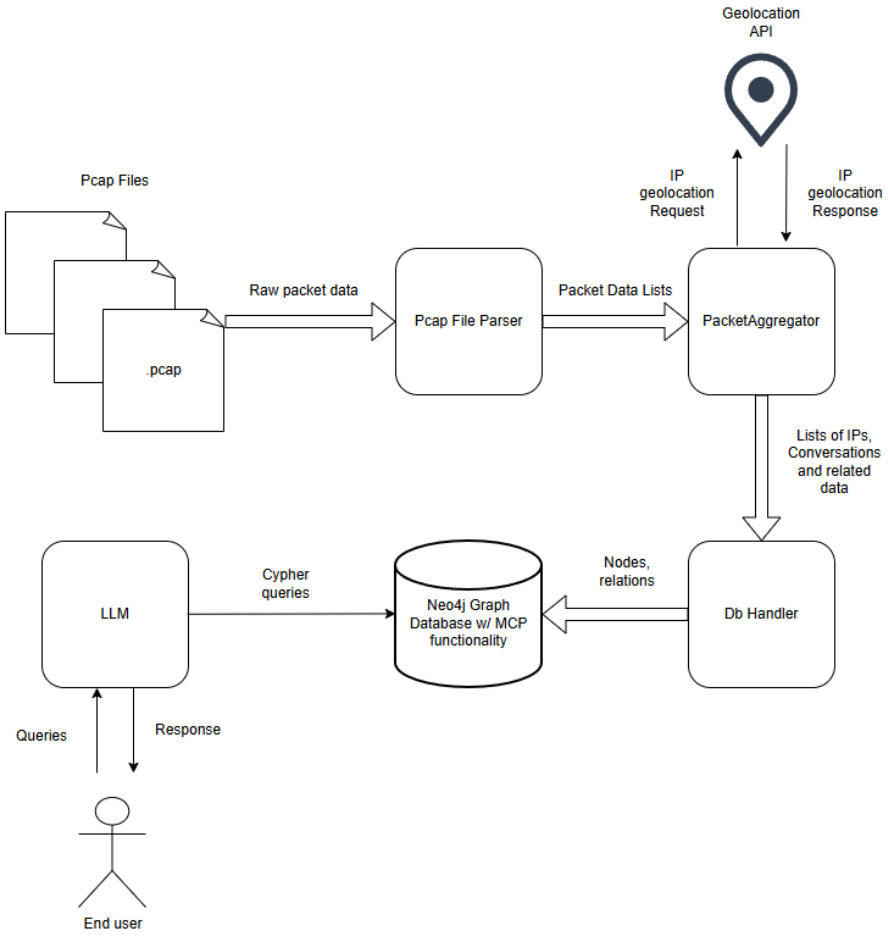
3.2 Επισκόπηση λειτουργίας PcapToKG

Καλούμαστε να σχεδιάσουμε ένα σύστημα το οποίο θα λαμβάνει στοιχεία διαδικτυακής κίνησης και θα τα τοποθετεί σε ένα γράφημα γνώσης, κατόπιν εμπλουτισμού τους με στοιχεία ασφαλείας γεωεντοπισμού. Στα πλαίσια της παρούσας θα αναπτύξουμε το πρόγραμμα PcapToKG, το οποίο θα εκτελείται σε ένα αρχείο .pcap, θα αποσπά από κάθε πακέτο τη χρήσιμη πληροφορία, θα συγκεντρώνει το πλήθος στοιχείων που ορίζονται στην οντολογία που περιγράφουμε στην ενότητα 3.1, και τέλος θα εισάγει τα στοιχεία στο γράφημα γνώσης.

Ο τρόπος λειτουργίας περιγράφεται επίσης στην Εικόνα 6 και απαρτίζεται από τα κάτωθι στοιχεία:

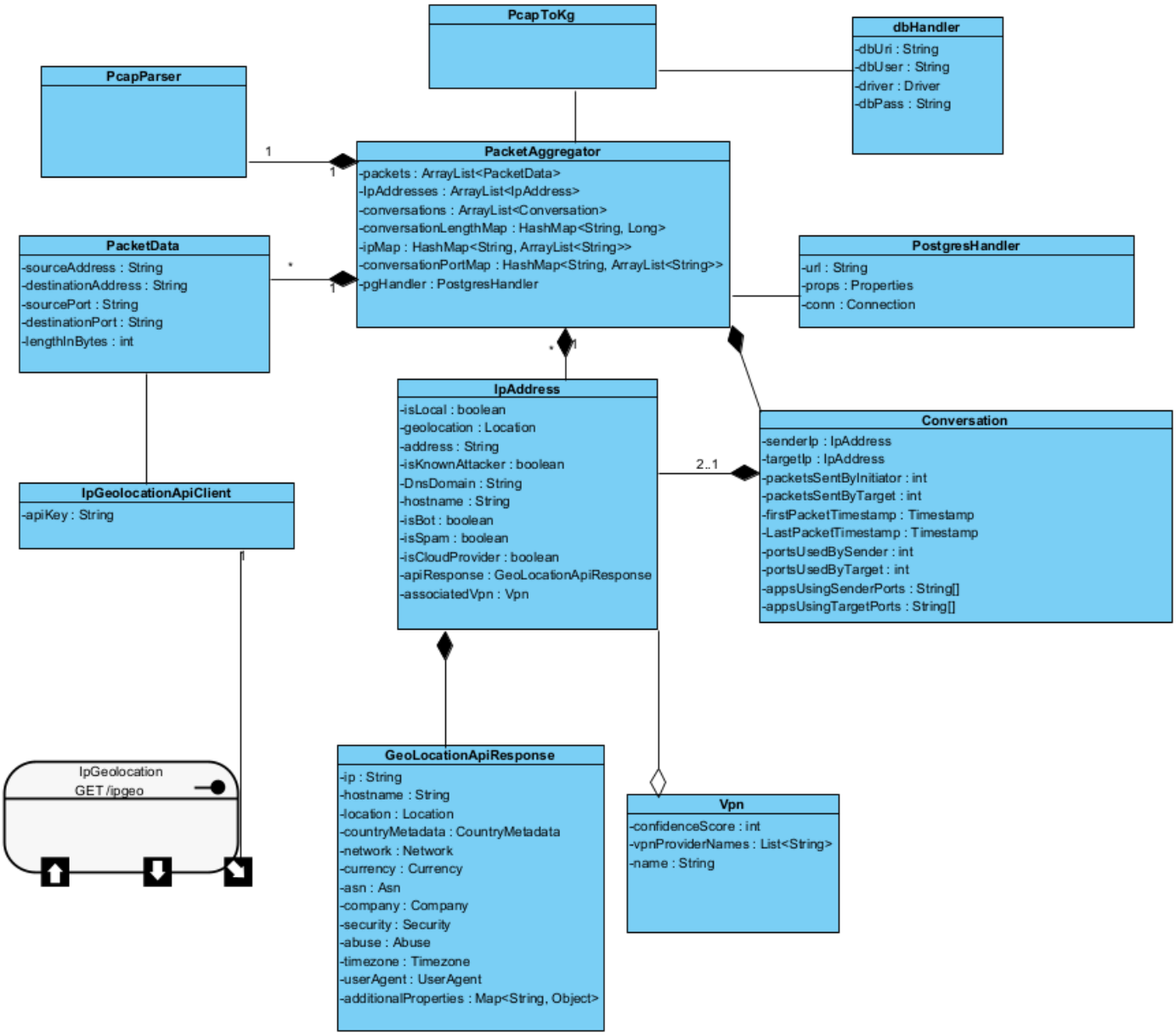
1. Αρχεία .pcap: η παραγωγή των αρχείων packet capture θα γίνει από εξωτερικό πρόγραμμα (π.χ. Wireshark) και από διαθέσιμες πηγές στο διαδίκτυο.
2. Pcap File Parser: η κλάση αυτή έχει ως σκοπό να σαρώσει τα αρχεία .pcap που θα εισάγει ο χρήστης και να δημιουργήσει Λίστες στιγμιοτύπων PacketData.
    - Το PacketData θα είναι μια κλάση που θα αναπαριστά τις ιδιότητες του πακέτου που μας αφορούν (IP αφετηρίας, IP προορισμού, πρωτόκολλο, μέγεθος, χρόνος άφιξης κλπ).
    - Κύρια λειτουργία ενός αντικειμένου PacketData είναι τα δεδομένα που παίρνουμε από τον Parser να μορφοποιούνται κατάλληλα αφαιρώντας επιπλέον χαρακτήρες από τις τιμές που λαμβάνουμε από κάθε πακέτο, ή με καταχώρηση default τιμών.
    - Τα στοιχεία που συλλέγουμε σε αυτή τη φάση για κάθε πακέτο είναι τα ακόλουθα:
        i. sourceAddress - η IP αποστολέα
        ii. destinationAddress - η IP προορισμού
        iii. lengthInBytes - το μέγεθος του πακέτου σε Byte
        iv. sourcePort - η θύρα αποστολέα
        v. destinationPort - η θύρα προορισμού

3. Την παραπάνω λίστα τροφοδοτούμε στην κλάση PacketAggregator, όπου θα αναλύσουμε τα δεδομένα σε που λαμβάνουμε από το αρχείο .pcap και θα εξάγουμε Ips και Conversations.
    - Η κλάση IP αντιστοιχεί σε μια IP που έχουμε εντοπίσει στο αρχείο .pcap. Πολλές IP μπορεί να αντιστοιχούν σε έναν Host, όμως λόγω της δομής των πακέτων και το γεγονός ότι η εύρεση hostname δεν είναι πάντοτε εφικτή, η IP θα είναι η κύρια κλάση στην οποία θα ανάγουμε τα στοιχεία γεωεντοπισμού, στοιχεία ασφαλείας κ.α. δεδομένα που μπορούμε να εξάγουμε από εξωτερικά API.
    - Η κλάση Conversation αντιστοιχεί σε ένα σύνολο πακέτων που έχουν πομπό και δέκτη δύο συγκεκριμένες IP. Πρόκειται για ένα μοντέλο που αντικατοπτρίζει μια συνομιλία μεταξύ δύο host όπου έχουμε αποστολή και αποδοχή πακέτων.
        i. Σημειώση: η συνομιλία θα αποτυπωθεί στο γράφημα με τα γνωρίσματα “sender” και “receiver”, τα οποία θα συλλέγονται ανάλογα με το πρώτο πακέτο της συνομιλίας. Είναι πιθανό η συνομιλία να έχει ξεκινήσει πριν την καταγραφή, αλλά δεχόμαστε ως παραδοχή ότι ο πρώτος αποστολέας αντιστοιχεί στην IP πηγής του πρώτου πακέτου.
4. Db Handler: η κλάση αυτή έχει το ρόλο της ενημέρωσης του γραφήματος γνώσης. Υλοποιεί μεθόδους που εισάγουν, ενημερώνουν ή διαγράφουν στοιχεία από τον γράφο.
5. LLM: Εφόσον έχουμε καταγράψει τα στοιχεία που μας ενδιαφέρουν στο γράφημα γνώσης, η βάση δεδομένων μας δίνει τη δυνατότητα να δημιουργήσουμε έναν MCP server, ο οποίος δίνει τα δεδομένα που βρίσκονται εντός του γράφου ως συμφραζόμενα (context) για κάποιο LLM.
6. Χρήση: Ο χρήστης θα μπορεί να επικοινωνεί με το LLM, προσπαθώντας να εξάγει χρήσιμα στοιχεία από την καταγεγραμμένη κίνηση.



Εικόνα 2 : High-level flow diagram

Η υλοποίηση θα γίνει με αντικειμενοστραφή προγραμματισμό σε γλώσσα Java, και αντιστοιχεί στο ακόλουθο διάγραμμα κλάσεων[Εικόνα 7]:



Εικόνα 3: Class Diagram

3.3 Σχεδιασμός Pcap Parser

Το πρώτο κομμάτι του συστήματος θα είναι ο Parser για αρχεία .pcap. Θα χρησιμοποιήσουμε την pcap4j βιβλιοθήκη για εξάγουμε πληροφορίες από κάθε

πακέτο που έχει εντοπιστεί στο αρχείο. Για τους σκοπούς ανάπτυξης, οι ροές θα εξαχθούν, αρχικά, από το Wireshark που τρέχει σε προσωπικό υπολογιστή. Δεν πρόκειται να τεθούν περιορισμοί στις πηγές δεδομένων, μπορεί κάλλιστα να αναλυθούν αρχεία packet capture που δημιουργήθηκαν από οποιαδήποτε κάρτα δικτύου. Αυτή η επιλογή μας επιτρέπει να έχουμε ιδιαίτερα ευρύ φάσμα δεδομένων, που δεν περιορίζεται στο εύρος δεδομένων του οργανισμού που το χρησιμοποιεί, αλλά μπορεί να μοντελοποιήσει την κίνηση σε ένα ευρύ φάσμα του internet. Ωστόσο, αντίστοιχα προκύπτει το πρόβλημα των τοπικών IPv4 διευθύνσεων, που, ενώ μπορεί να έχουν την ίδια IP σε σχέση με κάποια κάρτα δικτύου, στην πραγματικότητα ανήκουν σε εντελώς διαφορετικούς υπολογιστές. Στην παρούσα εργασία δεν θα ασχοληθούμε με την αντιμετώπιση αυτού του ζητήματος, η οποία μπορεί να έγκειται, π.χ. στη χρήση των public IP με NAT, και δεχόμαστε ότι το πρόγραμμα θα τρέξει υπολογιστές ενός τοπικού δικτύου.

Σε αυτή την φάση, προκύπτει ένας σημαντικός σχεδιαστικός κόμβος, συγκεκριμένα το πώς ακριβώς θα μοντελοποιηθεί ένα πακέτο και τι θα σταλθεί στην επόμενη φάση της ανάλυσης της κίνησης. Επιλέγουμε μια δομή που θα ονομάσουμε PacketData η οποία θα αποτελείται από δεδομένα που προκύπτουν από την ανάλυση των πακέτων, όπως φαίνεται στο διάγραμμα κλάσεων:

- sourceAddress
- destinationAddress
- sourcePort
- destinationPort
- lengthInBytes

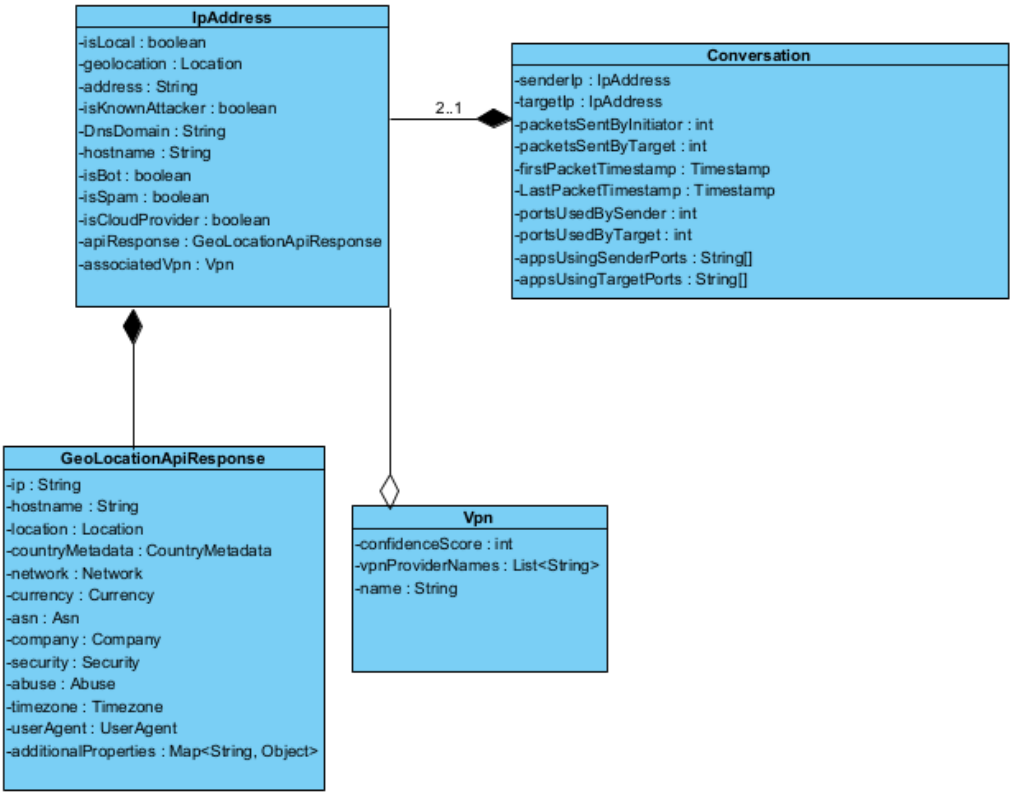
Σημειώνεται ότι στην τρέχουσα υλοποίηση, αποσπώνται στοιχεία μόνο για κίνηση IPv4, όχι για κίνηση IPv6 και όχι για διαφορετικά πρωτόκολλα (π.χ. QUIC, DNS, UDP, ICMP).

3.4 Σχεδιασμός λειτουργίας PacketAggregator

Ο στόχος της κλάσης αυτής θα είναι να δημιουργήσει τις τελικές δομές δεδομένων που θα αποθηκευτούν στο γράφημα γνώσης μας. Συγκεκριμένα, θα παίρνει σαν είσοδο μια λίστα αντικειμένων PacketData (όπως φαίνεται στην ενότητα 3.2) και θα δημιουργεί λίστες με δομές IP και Conversation, οι οποίες θα αποτυπώνουν τα κάτωθι:

- Ποιες συνομιλίες γίνονται μεταξύ ποιών IP
- Αν μια IP είναι local ή public
- Τι όγκος δεδομένων εστάλη από κάθε IP σε κάθε συνομιλία
- Ποιες θύρες χρησιμοποιήθηκαν για κάθε IP
- Που βρίσκεται (γεωγραφικά) η κάθε IP
- Στοιχεία ασφαλείας για κάθε IP
- Αν μπορούμε να βρούμε DNS entry για την εκάστοτε IP
- Αν ανήκει μια IP σε VPN
- Ποιο είναι το σκορ εμπιστοσύνης για το VPN

Προκύπτουν δυο κύριες κλάσεις: Conversation και IPaddress, οι οποίες είναι και οι κεντρικές οντότητες της οντολογίας μας. Επιπλέον, αν οι αποκρίσεις του API δώσουν στοιχεία για VPN, αυτά θα συνδέονται με την κλάση IP, καθώς και το αντικείμενο που απεικονίζει την πλήρη απόκριση του API. Οι κλάσεις αποδίδονται σχηματικά στην Εικόνα 8.



Εικόνα 4:Conversation-IP Class Diagram

Τα στοιχεία γεωεντοπισμού, στοιχεία ασφαλείας και στοιχεία hostname και VPN λαμβάνονται από το API της IPGeolocation, τα οποία είναι διαθέσιμα επί πληρωμή. Για τα πειράματα μας, και ώστε να αποφευχθούν τα υπέρογκα κόστη από επαναλαμβανόμενες ερωτήσεις, δημιουργήθηκε μια τοπική βάση δεδομένων Postgres στην οποία αποθηκεύουμε τις αποκρίσεις του API, ώστε να εξοικονομήσουμε credits χρήσης, αλλά και χρόνο σε κάθε ερώτημα. Για το λόγο αυτό χρησιμοποιήθηκε η κλάση PostgresHandler, στην οποία υλοποιούνται μέθοδοι για την εισαγωγή και ανάκτηση δεδομένων από την τοπική βάση δεδομένων.

3.5 Ανάλυση λειτουργίας DbHandler

Η λειτουργία του DbHandler θα είναι να λάβει λίστες αντικειμένων IP και Conversation, όπως αυτά θα έχουν διαμορφωθεί από τα προηγούμενα στάδια επεξεργασίας και να παράγει τα κατάλληλα queries ώστε να εισάγονται στο γράφημα γνώσης με τη μορφή που έχει οριστεί στην ενότητα 3.1. Στην παρούσα υλοποίηση, η βάση δεδομένων είναι η Neo4j με γλώσσα Cypher, τα οποία θα παρουσιαστούν στο επόμενο κεφάλαιο.

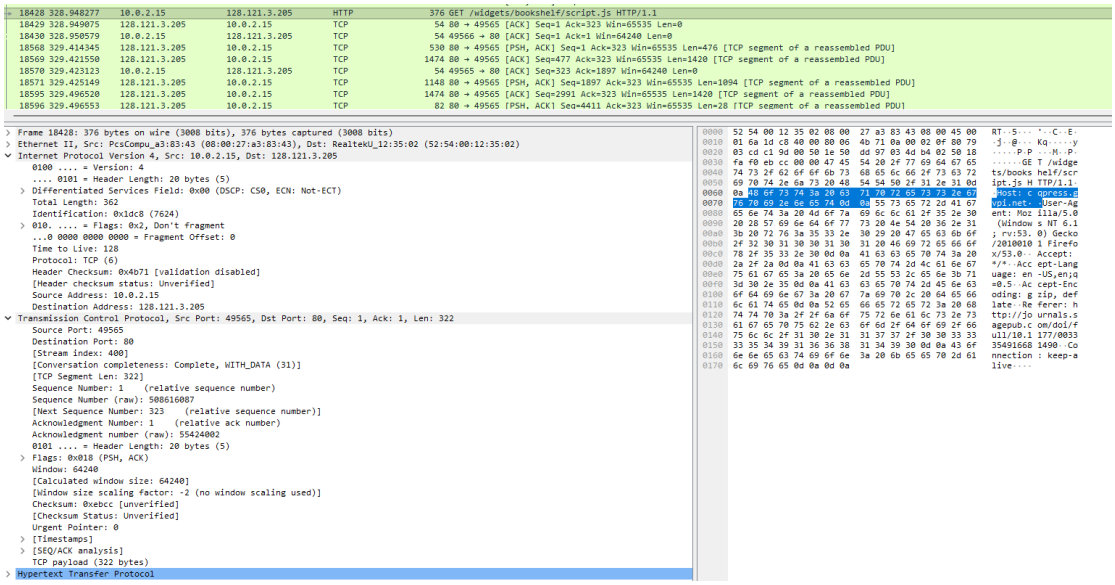
3.6 MCP server - σύνδεση με LLM

Εφόσον έχει δημιουργηθεί το γράφημα γνώσης, θα ενεργοποιήσουμε τη λειτουργία του MCP server που παρέχει η Neo4j, η οποία επιτρέπει την προσπέλαση από LLM. Εφόσον έχει παραμετροποιηθεί σωστά ο MCP Server, είναι πλέον εφικτό να κάνουμε ερωτήματα στο LLM ώστε να λάβουμε συμπεράσματα σχετικά με την κίνηση που έχει καταγραφεί.

4. Μεθοδολογία & εργαλεία

4.1 Wireshark

Το WireShark αποτελεί ένα από τα πιο διαδεδομένα προγράμματα ανάλυσης πακέτων. Παρέχει στο χρήστη τη δυνατότητα να καταγράψει την κίνηση που περνάει από τον υπολογιστή του ανά πακέτο, καταγράφοντας πληροφορίες όπως διευθύνσεις και θύρες πηγής και προορισμού, πρωτόκολλα, φορτία (Wireshark, 2025). Παρέχει εκτενή πληροφορία για όλα τα επίπεδα ενθυλάκωσης που έχει ανιχνεύσει στο πακέτο, όπως φαίνεται στην εικόνα 9.



Εικόνα 5: Wireshark GUI

Ο λόγος χρήσης του Wireshark είναι η αξιοπιστία του και το γεγονός ότι είναι πρόγραμμα ανοιχτού κώδικα. Στην παρούσα εργασία θα χρησιμοποιηθεί για την καταγραφή ορισμένων δειγμάτων τοπικής κίνησης προς επικύρωση λειτουργίας του προγράμματος PcapToKG.

4.2 Java-IntelliJ IDEA-Maven

Η Java είναι από τις πιο ευρέως χρησιμοποιούμενες γλώσσες παγκοσμίως, διδακτέα στα περισσότερα πανεπιστήμια σε προγράμματα πληροφορικής και με πλήρες οικοσύστημα βιβλιοθηκών. Η έκδοση που θα χρησιμοποιήσουμε είναι η έκδοση 17, για λόγους συμβατότητας με τους οδηγούς της βάσης δεδομένων.

Η Java είναι μια πλήρως αντικειμενοστραφής γλώσσα προγραμματισμού που δημιουργήθηκε το 1995 από την Sun Microsystems και αργότερα εξαγοράστηκε από την Oracle.

Ένα από τα κύρια χαρακτηριστικά της Java είναι το γεγονός ότι ο κώδικας μπορεί να εκτελεστεί χωρίς παραλλαγές σε οποιοδήποτε λειτουργικό σύστημα, καθώς εκτελείται στο JVM-Java Virtual Machine (Java, 2025).

Η επιλογή της γλώσσας έγινε για τους παρακάτω λόγους:

1. Οικειότητα, καθώς είναι η γλώσσα προγραμματισμού την οποία δύναμαι να χρησιμοποιήσω καλύτερα.
2. Συμβατότητα με Neo4j. Η βάση δεδομένων Neo4j είναι επίσης γραμμένη σε Java και παρέχει άρτια τεκμηριωμένο driver για τη σύνδεση μεταξύ του προγράμματος και της βάσης δεδομένων.
3. Μεγάλη κοινότητα χρηστών και διαθεσιμότητα εργαλείων. Η Java είναι από τις πιο διαδεδομένες γλώσσες παγκοσμίως, κάτι που διευκολύνει την εύρεση βιβλιοθηκών καθώς και βοήθεια για τυχόν προκλήσεις κατά την υλοποίηση.

Το IDE επιλογής μας είναι το IntelliJ IDEA της JetBrains και το project θα χτιστεί μέσω Maven. Το Intellij IDEA αποτελεί το πιο διαδεδομένο IDE (Integrated Development Environment) για Java, κατέχοντας το 84% της αγοράς (Jrebel, 2026). Παρέχει εκτενή λειτουργικότητα για debugging, αυτόματη συμπλήρωση κειμένου και λειτουργικότητα refactoring, μεταξύ άλλων.

Το Maven αποτελεί ένα εργαλείο διαχείρισης εξαρτήσεων (dependencies) κατά τη διαδικασία κατασκευής του εκτελέσιμου προγράμματος. Δημιουργήθηκε από την Apache με σκοπό την βελτίωση της διαδικασίας κατασκευής εκτελέσιμων και την παροχή μιας ενός ενιαίου συστήματος κατασκευής. Η εισαγωγή βιβλιοθηκών είναι ιδιαίτερα εύκολη, με την εισαγωγή ενός μπλοκ κώδικα xml στο pom.xml που βρίσκεται στο φάκελο του project. Οι βιβλιοθήκες βρίσκονται στο αποθετήριο του Maven (Maven Repository) και βρίσκονται αυτόματα. (Apache, 2026).

4.3 Βιβλιοθήκη Pcap4j

Η Pcap4j είναι η πιο πρόσφατη ανοικτή βιβλιοθήκη που επιτρέπει την προσπέλαση αρχείων .pcap σε γλώσσα Java. Δημιουργήθηκε και συντηρείται από τον Kaito Yamada, ελλείψει κατάλληλων βιβλιοθηκών για καταγραφή pcap για Java, ώστε να βοηθήσει στη δημιουργία ενός προσομοιωτή SNMP δικτύου. (Pcap4J, 2022)

Η Pcap4j μας δίνει τη δυνατότητα καταγραφής πακέτων σε πραγματικό χρόνο, αλλά και την ανάλυση αρχείων .pcap. Για τη χρήση της, αρκεί να εισάγουμε τη σχετική εξάρτηση στο pom.xml του project.

4.4 IPGeolocation API

Για τον εμπλουτισμό δεδομένων θα χρειαστεί να χρησιμοποιήσουμε κάποιο εξωτερικό API. Υπάρχουν αρκετές βάσεις δεδομένων γεωεντοπισμού και ασφαλείας IP. Στην περίπτωση μας, θα χρησιμοποιήσουμε το app.ipgeolocation.io. Η επιλογή έγινε λόγω ευκολίας χρήσης και χαμηλού κόστους συγκριτικά με άλλες παρόμοιες λύσεις. Παρέχει πολλαπλά API, εκ των οποίων θα χρησιμοποιήσουμε τα:

- IP Location API
- IP Security API
- UserAgent API

Τα API δίνουν απαντήσεις σε μορφή Json. Οφείλουμε να σημειώσουμε ότι τα δεδομένα που επιστρέφονται από το API δεν εξετάζονται προς την ορθότητα τους και δεν έχουμε μέθοδο αξιολόγησης της εγκυρότητας. Ενδέχεται να επιστρέφονται λανθασμένες ή ανακριβείς ενδείξεις για κάποιες από τις IP.

4.5 Jackson API - Okhttp

Η OkHttp αποτελεί μια σύγχρονη βιβλιοθήκη πελάτη HTTP για τη γλώσσα Java, η οποία αναπτύσσεται από την **Square**. Σχεδιάστηκε με στόχο την αποδοτική, ασφαλή και αξιόπιστη επικοινωνία εφαρμογών με απομακρυσμένες διαδικτυακές υπηρεσίες μέσω των πρωτοκόλλων HTTP/1.1, HTTP/2 και WebSocket (Square, 2026). Με τη χρήση της βιβλιοθήκης θα αποστέλλουμε αιτήματα στο Geolocation API και θα διαχειριζόμαστε τις αποκρίσεις.

Για την σειριοποίηση και αποσειριοποίηση των απαντήσεων, θα χρησιμοποιήσουμε τη βιβλιοθήκη Jackson. Η βασική λειτουργία της Jackson είναι η μετατροπή αντικειμένων Java σε μορφή JSON και αντίστροφα, διευκολύνοντας την ανταλλαγή δεδομένων μεταξύ εφαρμογών και διαδικτυακών υπηρεσιών. Θα έχουμε ήδη δημιουργήσει POJO που αντιστοιχούν στη μορφή του JSON ώστε η αποσειριοποίηση των απαντήσεων να αντιστοιχίζεται σε στιγμιότυπο αντικειμένου.

4.6 PostgreSQL

Η χρήση της PostgresSQL δεν ήταν προσχεδιασμένη, αλλά προέκυψε ως ανάγκη κατά τη φάση της αξιολόγησης, όπως αναφέρουμε στην ενότητα 3.3.

Η PostgreSQL είναι ένα ανοικτού κώδικα Σχεσιακό Σύστημα Διαχείρισης Βάσεων Δεδομένων (RDBMS) που αναπτύσσεται συνεχώς από μια διεθνή κοινότητα προγραμματιστών από το 1996. Βασίζεται στο ερευνητικό έργο POSTGRES του Πανεπιστημίου της Καλιφόρνιας στο Berkeley και θεωρείται μία από τις πιο ώριμες και αξιόπιστες βάσεις δεδομένων γενικής χρήσης.

4.7 Neo4j Graph Database

Το εργαλείο επιλογής μας για την δημιουργία γραφήματος γνώσης στην παρούσα εργασία θα είναι η Neo4J. Ο λόγος επιλογής είναι η δημοφιλία του εργαλείου, η εκτενής τεκμηρίωση τόσο για τη χρήση της βάσης όσο και για τη γλώσσα Cypher την οποία χρησιμοποιεί για τις συναλλαγές.

Η Neo4j είναι ένα σύστημα διαχείρισης βάσεων δεδομένων γράφων (graph database management system), σχεδιασμένο για την αποθήκευση, αναζήτηση και ανάλυση δεδομένων που παρουσιάζουν έντονες διασυνδέσεις. Αντί να οργανώνει τα δεδομένα σε πίνακες και σχέσεις μέσω ξένων κλειδιών, όπως οι σχεσιακές βάσεις δεδομένων, η Neo4j υλοποιεί το μοντέλο property graph, στο οποίο τα δεδομένα αναπαρίστανται ως κόμβοι (nodes), ακμές (relationships) και ιδιότητες (properties).(Neo4j, 2026).

Η Neo4j παρέχει επίσης δυνατότητα MCP την οποία θα χρησιμοποιήσουμε για την σύνδεση της βάσης με το Claude.

4.8 Claude

Το Claude είναι ένα Μεγάλο Γλωσσικό Μοντέλο (Large Language Model, LLM) που αναπτύχθηκε από την εταιρεία Anthropic, μια αμερικανική εταιρεία τεχνητής νοημοσύνης που ιδρύθηκε το 2021. Η Anthropic ιδρύθηκε από πρώην μέλη της OpenAI, μεταξύ των οποίων οι Dario και Daniela Amodei.

Το Claude βασίζεται στην αρχιτεκτονική Transformer, την οποία χρησιμοποιούν τα περισσότερα σύγχρονα LLM. Εκπαιδεύτηκε με συνδυασμό τεχνικών:

- Supervised Fine-Tuning (SFT): Εκπαίδευση σε ανθρώπινα επισημειωμένα παραδείγματα.
- Reinforcement Learning from Human Feedback (RLHF): Βελτιστοποίηση μέσω ανθρώπινης αξιολόγησης απαντήσεων.
- Constitutional AI (CAI): Ιδιόκτητη μέθοδος της Anthropic που χρησιμοποιεί ένα σύνολο αρχών («σύνταγμα») για να καθοδηγεί τη συμπεριφορά του μοντέλου, μειώνοντας την εξάρτηση από ανθρώπινη επίβλεψη σε κάθε βήμα.

Χρήσιμα χαρακτηριστικά του Claude είναι το ευρύ παράθυρο συμφραζομένων, η πολυτροπικότητα και η χρήση εργαλείων μέσω του πρωτοκόλλου MCP. (Claude, 2026).

5. Υλοποίηση

5.1 Προαπαιτούμενα

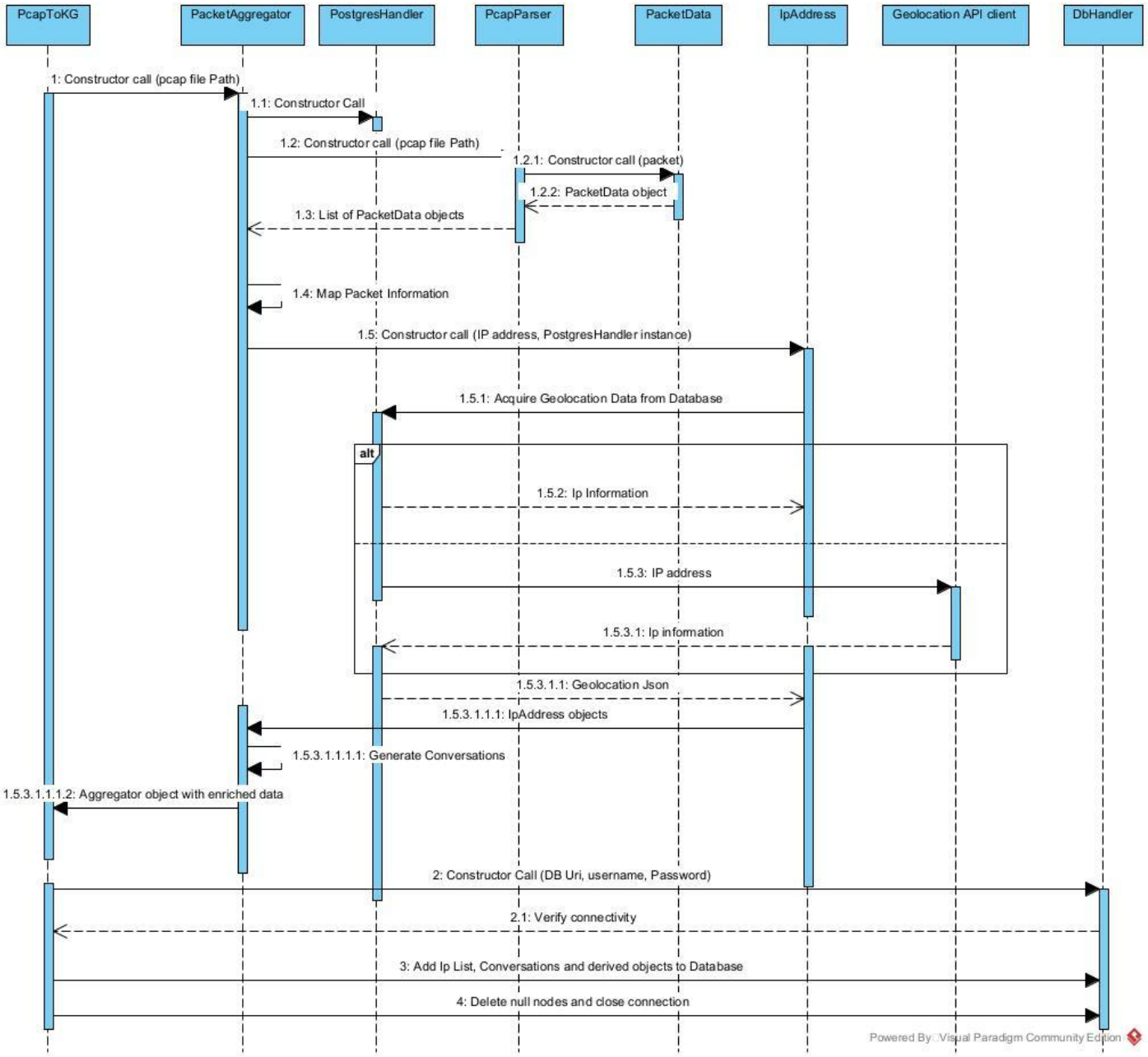
Προκειμένου να λειτουργήσει το πρόγραμμα, χρειάζεται να έχουμε στη διάθεση μας τα παρακάτω:

1. Εγκατεστημένη τοπική βάση δεδομένων Neo4J
2. Προαιρετικά Κλειδί συνδρομής API για το geolocationAPI ή
3. Εγκατεστημένη τοπική βάση δεδομένων PostgreSQL
4. Java JDK 17

Όλες οι εκτελέσεις έχουν γίνει εντός του IDE, με τις τιμές των παραπάνω παραμέτρων να αποτελούν μέρος του κώδικα σε σχετικά σχολιασμένες μεταβλητές. Δεν διαβάζονται ως arguments.

5.2 Παρουσίαση κλάσεων PcapToKG

Ο τρόπος λειτουργίας αποδίδεται στο διάγραμμα αλληλουχίας που φαίνεται στην Εικόνα 10.



Εικόνα 6: Sequence Diagram

Παρακάτω θα παρουσιάσουμε τις λειτουργικές κλάσεις του προγράμματος με τη σειρά κλήσης, όπως παρουσιάζεται στο διάγραμμα αλληλουχίας. Στο παράρτημα 9.1 δίνονται όλοι οι κώδικες των κλάσεων που φαίνονται στο διάγραμμα. Παραλείπονται οι κλάσεις POJO.

5.2.1 PcapToKG

Η PcapToKG περιέχει αποκλειστικά τη μέθοδο main. Στην main, δημιουργούμε ένα στιγμιότυπο PacketAggregator με γνώρισμα το path του αρχείου .pcap που θα αναλυθεί. Στον PacketAggregator, όπως θα δούμε στη συνέχεια, αποθηκεύουμε λίστες με τις IP και τις συνομιλίες τις οποίες επιθυμούμε να εισάγουμε στη βάση δεδομένων. Το path του αρχείου δίνεται ως ακολούθως:

PacketAggregator aggregator = new PacketAggregator ("**src/main/resources/sample.pcap**");

Εφόσον ολοκληρωθεί η κλήση του constructor και συγκεντρωθούν οι λίστες, δημιουργείται ένα στιγμιότυπο DbHandler. Πρώτα καλείται η μέθοδος εισαγωγής IP στο γράφημα γνώσης για κάθε IP του PacketAggregator και κατόπιν η μέθοδος εισαγωγής Conversation στο γράφημα, για κάθε Conversation του PacketAggregator.

Κατόπιν, διαγράφονται όλες οι εγγραφές με όνομα “null”, καλώντας τη σχετική μέθοδο του DbHandler.

Τέλος, κλείνει η σύνδεση στη βάση δεδομένων και το πρόγραμμα ολοκληρώνεται.

5.2.2 PacketAggregator

Η κλάση PacketAggregator είναι η πιο σύνθετη του προγράμματος. Χρησιμοποιούμε αυτή την κλάση ώστε να δημιουργήσουμε τις τελικές λίστες με αντικείμενα IpAddress και Conversation τα οποία εισάγουμε στο γράφημα. Η κλάση λειτουργεί ως εξής:

Με την κλήση του constructor δημιουργεί ένα στιγμιότυπο Postgres Handler που θα περαστεί αργότερα στις IP, και έναν parser δίνοντας ως γνώρισμα την τοποθεσία αρχείου, όπως αυτή δόθηκε από τη main. Καλεί την μέθοδο parsePacketData() η οποία επιστρέφει μια λίστα με στιγμιότυπα PacketData. Από κάθε πακέτο εξάγουμε πληροφορία και την τοποθετούμε σε δομή HashMap. Έχουμε ορίσει τρία HashMap:

- ipMap: HashMap <String, ArrayList<String>> : χάρτης που αντιστοιχεί κάθε IP με μια λίστα από Ports που έχει χρησιμοποιήσει σε συνομιλίες.
- conversationLengthMap: HashMap<String, Long>: χάρτης που αντιστοιχεί κάθε IP με το πλήθος byte που έχει στείλει
    - Σημείωση: για κάθε συνομιλία όπου έχουν στείλει και οι δύο υπολογιστές πακέτα, υπάρχουν δύο keys σε αυτό το map. Το ένα έχει μορφή "{Ip1}-{Ip2}": X, ενώ το δεύτερο έχει μορφή "{Ip2}-{Ip1}":Y, όπου Ip1 η διεύθυνση αποστολέα του πρώτου πακέτου που εστάλη, Ip2 η διεύθυνση του παραλήπτη του πρώτου πακέτου, X,Y τα αθροίσματα σε byte των απεσταλμένων ή παραληφθέντων πακέτων. Όταν δημιουργούμε τη δομή Conversation, αυτές οι πληροφορίες θα συγχωνευθούν και θα μετατραπούν σε sender, receiver, bytesSent και bytesReceived,
- conversationPortMap: HashMap<String, ArrayList<String>>: χάρτης που αντιστοιχεί μια συνομιλία με λίστα Ports που χρησιμοποιούνται σε αυτήν
    - Και πάλι έχουμε διπλές εγγραφές σε αυτό το Map, οι οποίες συγχωνεύονται αργότερα.

Εφόσον ολοκληρωθεί η ανάλυση των πακέτων και τα δεδομένα πλέον έχουν τοποθετηθεί στα HashMaps, καλείται η μέθοδος aggregatePacketInfo(), η οποία δημιουργεί την τελική λίστα με δομές IP και την λίστα με δομές Conversation που αποθηκεύονται στον PacketAggregator:

- Για τη δημιουργία των IP εκτελεί διαπέραση του κάθε map μέσω του keyset, και καλέι τον constructor της κλάσης IpAddress, με γνώρισμα την IP και τον PostgresHandler του PacketAggregator που την καλεί. Κατόπιν περνάει τη

λίστα Ports απευθείας στη νέα δομή IP μέσω σχετικού setter και η δομή αποθηκεύεται στη λίστα ipAddresses του PacketAggregator

- Για τη δημιουργία των δομών Conversation λειτουργούμε με παρόμοιο τρόπο, με τη διαφορά ότι για κάθε conversation ψάχνουμε δύο κλειδιά στα HashMap: το κλειδί μορφής Ip1-Ip2 και το κλειδή μορφής Ip2-Ip1. Η τελική ονομασία της δομής θα είναι Ip1-Ip2.

Εφόσον ολοκληρωθεί η δημιουργία της λίστας, ολοκληρώνεται η κλήση του constructor.

5.2.3 PcapParser

Η κλάση PcapParser αποτελείται από τον constructor, ο οποίος δέχεται σαν γνώρισμα το path του αρχείου .pcap, και τη μέθοδο ArrayList<PacketData> ParsePacketData().

Η μέθοδος ParsePacketData() δημιουργεί ένα PcapHandle από τη βιβλιοθήκη Pcap4j και καλεί τον Constructor της δομής PacketData για κάθε πακέτο. Το παραχθέν πακέτο αποθηκεύεται σε λίστα την οποία επιστρέφει τελικά η μέθοδος.

5.2.4 PacketData

Στη δομή PacketData ορίζουμε τα παρακάτω γνωρίσματα:

```
private String      sourceAddress;
private String      destinationAddress;
private Integer     lengthInBytes;
private String      sourcePort;
private String      destinationPort;
```

Με την κλήση του Constructor, καλούνται μέθοδοι υπεύθυνες για την εξαγωγή και την τροποποίηση της κάθε τιμής από τη δομή Packet που μας παρέχει η Pcap4j.

Οι μέθοδοι είναι επίσης υπεύθυνες για την διαχείριση εσφαλμένων ή null τιμών από το Packet που έχει δώσει η βιβλιοθήκη.

Σημείωση: υπήρχε πρόβλεψη για την εξαγωγή Timestamp, αλλά η τρέχουσα λειτουργικότητα της βιβλιοθήκης Pcap4j επιτρέπει τα Timestamps μόνο για ταυτόχρονη καταγραφή.

5.2.5 IpAddress

Η κλάση IpAddress δημιουργείται με κλήση από τον PacketAggregator. Με την κλήση του constructor, εξετάζουμε αν η δοθείσα IP δεν είναι null και εφόσον δεν είναι, εξετάζουμε αν ανήκει bogon εύρος διευθύνσεων. Εάν ανήκει, τότε θεωρούμε τη διεύθυνση ως local και επιστρέφουμε την οντότητα.

Σε διαφορετική περίπτωση, προχωράμε σε εμπλουτισμό της οντότητας με στοιχεία γεωεντοπισμού.

Χρησιμοποιούμε τη μέθοδο inputToDb του PostgresHandler η οποία επιστρέφει την τιμή που βρίσκεται στην βάση δεδομένων. Εάν δεν βρίσκεται στη βάση δεδομένων (αφού πρώτα την λάβει από το API). Η μέθοδος μας επιστρέφει ένα Json των δεδομένων, το οποίο μετατρέπουμε σε δομή GeoLocationApiResponse ώστε να εξάγουμε τις πληροφορίες που χρειαζόμαστε για την IP.

Οι πληροφορίες που εξάγουμε είναι οι παρακάτω:

```
private String address
private ArrayList<String> usedPorts
private boolean isLocal
private String hostName
private String dnsDomain
private boolean isAnonymous
private boolean isBot
private boolean isSpam
private boolean isCloudProvider
private String hostOs
private GeoLocationApiResponse apiResponse
private Location geolocation
```

```
private Vpn associatedVpn
private boolean isKnownAttacker
```

Ιδιαίτερη σημασία έχουν οι δομές Location και Vpn. Η δομή Location αντιστοιχεί στην αντίστοιχη δομή του Json που επιστρέφεται, ενώ τη δομή Vpn τη δημιουργούμε από τα δεδομένα ασφαλείας που βρίσκονται στην υποκλάση Security.

Σημείωση: Με τα τρέχοντα δεδομένα είναι αδύνατο να γίνει αναζήτηση των Domain όπου ανήκει ένα hostname από την IP. Ο τρόπος που εκτελούμε τον υπολογισμό είναι η αποκοπή του μέρους του hostname μετά την πρώτη τελεία (.). Αναγνωρίζεται ότι η μέθοδος μπορεί να οδηγήσει σε εσφαλμένο υπολογισμό του Domain και παραβλέπει την πιθανότητα ύπαρξης περισσότερων Domain από μία IP.

Έχει γίνει πρόβλεψη ώστε αν δεν υπάρχει τοπική ΒΔ PostgreSQL, να επιχειρούμε την απόκτηση του JSON από το API.

5.2.6 PostgresHandler

Σε αυτή την κλάση υλοποιούμε δύο μεθόδους: getIpResponseFromDb και InputIpResponseToDb. Η δεύτερη καλεί την πρώτη σε περίπτωση κενής απάντησης, καλεί τη μέθοδο getRawResponse του IpGeolocationApiClient και εισάγει την απάντηση στη βάση δεδομένων για μελλοντική χρήση. Τέλος, επιστρέφεται η τιμή που πλέον βρίσκεται στη βάση δεδομένων.

Η κλάση αυτή δημιουργήθηκε εξ' ανάγκης και δεν αποτελούσε μέρος του αρχικού σχεδιασμού. Τα διαπιστευτήρια για τη σύνδεση με τη βάση δεδομένων είναι hardcoded στις παρακάτω μεταβλητές:

```
String url = "jdbc:postgresql://localhost/ipgeolocationdb";

props.setProperty("user", "postgres");
props.setProperty("password", "***");
```

5.2.7 IpGeolocationApiClient

Η παρούσα κλάση υλοποιεί τη στατική μέθοδο parseApiResponse(String Ip) η οποία επιχειρεί να λάβει την απόκριση του API για τη δοθείσα IP. Επιστρέφει το Json σε μορφή object GeoLocationApiResponse.Επιπλέον υλοποιεί τη στατική μέθοδο getRawResponse η οποία αντί για object επιστρέφει string με αυτούσια την απόκριση του API. Στην τρέχουσα υλοποίηση, το κλειδί είναι hardcoded σε μεταβλητή, η οποία φαίνεται παρακάτω, όπως και το url κλήσης του API:

```
String ApiKey = "***";
String url =
"https://api.ipgeolocation.io/v3/ipgeo?apiKey="+ApiKey+"&ip="+Ip+"&in
clude=security,user_agent,hostname";
```

5.2.8 GeolocationApiResponse

Η παρούσα κλάση αποτελεί POJO που αντιπροσωπεύει την απόκριση του IP Geolocation API. Δεν έχουν υλοποιηθεί μέθοδοι πέραν από getters και setters. Η μορφή του Json φαίνεται στο παράρτημα 9.2. Οι υποκλάσεις που έχουν δημιουργηθεί είναι οι:

- Abuse
- Asn
- Company
- CountryMetadata
- Currency
- Device
- DstEnd
- DstStart
- Engine
- Location
- Network
- OperatingSystem

- Security
- Timezone
- UserAgent

Εξ αυτών, χρησιμοποιούνται οι υποκλάσεις Security, OperatingSystem, UserAgent στο υπόλοιπο πρόγραμμα. Οι λοιπές δεν χρησιμοποιούνται σε αυτή τη φάση. Οι κλάσεις έχουν δημιουργηθεί με βάση την επίσημη, πλήρη απάντηση. Στο παράρτημα 9.2 παρατίθεται JSON που περιέχει μόνο τα σχετικά για το πρόγραμμα μας πεδία.

5.2.9 Conversation

Η υλοποίηση της κλάσης Conversation είναι ιδιαίτερα απλή, καθώς η προεργασία στα δεδομένα έχει γίνει στον PacketAggregator. Συνεπώς εδώ υλοποιούνται μόνο διάφοροι Constructors και Getters/Setters.

Η πληροφορία που αποτυπώνεται είναι η ακόλουθη:

```
private long lengthOfConversationInBytes;
private ArrayList<String> portsUsed;
private int amountOfPacketsSentFromIp1;
private long lengthOfPacketsSentFromIp1;
private ArrayList<String> portsUsedOnIp1;
private int amountOfPacketsSentFromIp2;
private long lengthOfPacketsSentFromIp2;
private ArrayList<String> portsUsedOnIp2;
private String Ip1str;
private String Ip2str;
```

Τα δεδομένα αυτά θα χρησιμοποιηθούν από τον DbHandler για την τελική τοποθέτηση σε γράφημα γνώσης.

5.2.10 DbHandler

Στην παρούσα κλάση τοποθετούμε τις δομές IpAddress, Conversation στο γράφημα γνώσης. Υλοποιείται η addIpToDb (IpAddress ip) και η addConversationToDb(Conversation conversation). Ο Constructor δημιουργεί μια

σύνδεση στην βάση, και κατόπιν για κάθε κλήση δημιουργείται κατάλληλο query το οποίο εκτελείται μέσω του driver της Neo4j.

Τα διαπιστευτήρια της Neo4j βρίσκονται δίνονται στον constructor:

```
DbHandler handler = new DbHandler("neo4j://127.0.0.1:7687", "neo4j",
"password");
```

Για τις IP, το Query που δημιουργείται είναι της ακόλουθης μορφής:

```
MERGE (ip:IP {address: $ipAddress, is_Bot: $isBot, is_Local:
$isLocal, is_known_attacker: $isKnownAttacker, is_spam: $isSpam })
MERGE (city:City {name: $city})
MERGE (ip)-[:IS_LOCATED_IN]->(city)
MERGE (country:Country{name: $country})
MERGE (city)-[:IS_LOCATED_IN]->(country)
MERGE (host: Host {hostname: $hostname})
MERGE (ip)-[:BELONGS_TO]->(host)
MERGE (hostOs: OS {title: $hostOs})
MERGE (host)-[:USES]->(hostOs)
MERGE (port0: PORT{port_Number: '443 (HTTPS)'})
MERGE (ip)-[:USES_PORT]->(port0) ',
parameters={isBot: FALSE, country: "United Kingdom", city: "London",
hostOs: "Cloud", vpnScore: "null", ipAddress: "172.187.86.74",
associatedVpn: "null", isKnownAttacker: FALSE, dnsDomain: "null",
isLocal: FALSE, port0: "443 (HTTPS)", hostname: "172.187.86.74",
isSpam: FALSE}
```

Για τις δομές Conversation, αντίστοιχα, δημιουργείται το ακόλουθο query:

```
MATCH (b:IP {address: $destinationAddress})
MERGE (conversation:CONVERSATION{id: $conversationId,
bytes_sent:$bytesSent, bytes_received:$bytesReceived, sender:
$sourceAddress, receiver: $destinationAddress})

MERGE (a)-[:HAS_CONVERSATION]->(conversation)
MERGE (b)-[:HAS_CONVERSATION]->(conversation)
MERGE (port0: PORT{port_Number: '59133 (unknown)'})
```

```
MERGE (port0)-[:IS_USED_IN]->(conversation)
MERGE (port1: PORT{port_Number: '443 (HTTPS)'})MERGE (port1)-
[:IS_USED_IN]->(conversation)
MERGE (port2: PORT{port_Number: '59134 (unknown)'})
MERGE (port2)-[:IS_USED_IN]->(conversation)
MERGE (port3: PORT{port_Number: 'null'})
MERGE (port3)-[:IS_USED_IN]->(conversation)
MERGE (port4: PORT{port_Number: '443 (HTTPS)'})
MERGE (port4)-[:IS_USED_IN]->(conversation)
MERGE (port5: PORT{port_Number: '59134 (unknown)'})
MERGE (port5)-[:IS_USED_IN]->(conversation)
MERGE (port6: PORT{port_Number: '59133 (unknown)'})
MERGE (port6)-[:IS_USED_IN]->(conversation) MERGE (port7:
PORT{port_Number: 'null'})
MERGE (port7)-[:IS_USED_IN]->(conversation) ',
parameters={destinationAddress: "157.240.30.35", sourceAddress:
"10.0.0.2", conversationId: "10.0.0.2-157.240.30.35", port7: NULL,
port5: "59134 (unknown)", bytesSent: 27908, port6: "59133 (unknown)",
port3: NULL, port4: "443 (HTTPS)", port1: "443 (HTTPS)", port2:
"59134 (unknown)", port0: "59133 (unknown)", bytesReceived: 199069}}
```

Σημειώνουμε ότι για κάθε port, δημιουργείται νέα εντολή Merge η οποία δίνεται στο τελικό query που εκτελείται. Επειδή σε ορισμένες περιπτώσεις ανάλυσης προέκυψαν IPs και Conversations που χρησιμοποιούσαν χιλιάδες ports, οι ports εισάγονται στο γράφημα ανά 100, ώστε να μην υπερφορτώνεται το query.
Τέλος, υλοποιείται ξεχωριστή μέθοδος απόσβεσης όλων των κόμβων με κενά (null) δεδομένα ταυτοποίησης.

5.3 Παράδειγμα εκτέλεσης - γράφημα

Ακολουθούν στιγμιότυπα από την εκτέλεση του προγράμματος σε διάφορες φάσεις:

- Parsing pcap (Εικόνα 11)
- Τέλος parsing και εξαγωγή δεδομένων από το πακέτο (Εικόνα 12)
- Δημιουργία λίστας IP και λίστας conversation (Εικόνα 13)
- Εισαγωγή δεδομένων στο γράφημα γνώσης/εκτέλεση Query (Εικόνα 14)

Όλη η έξοδος δίνεται σε κονσόλα, δεν έχει υλοποιηθεί γραφικό περιβάλλον χρήσης ή οποιαδήποτε αλληλεπίδραση με τον χρήστη.

```
PacketData{lengthInBytes=1460, sourceAddress='40.126.31.3', destinationAddress='10.0.0.2', sourcePort='443 (HTTPS)', destinationPort='50146 (unknown)'}
Parsing next packet: 1015
PacketData{lengthInBytes=40, sourceAddress='10.0.0.2', destinationAddress='40.126.31.3', sourcePort='50146 (unknown)', destinationPort='443 (HTTPS)'}
Parsing next packet: 1016
PacketData{lengthInBytes=44, sourceAddress='40.126.31.3', destinationAddress='10.0.0.2', sourcePort='443 (HTTPS)', destinationPort='50146 (unknown)'}
Parsing next packet: 1017
Failed to parse source port from packet
Failed to parse destination port from packet
PacketData{lengthInBytes=77, sourceAddress='10.0.0.2', destinationAddress='10.0.0.138', sourcePort='null', destinationPort='null'}
Parsing next packet: 1018
Failed to parse source port from packet
Failed to parse destination port from packet
PacketData{lengthInBytes=211, sourceAddress='10.0.0.138', destinationAddress='10.0.0.2', sourcePort='null', destinationPort='null'}
Parsing next packet: 1019
```

Εικόνα 7: Φάση parsing αρχείου .pcap

```
Parsing next packet: 3045
Exhausted pcap file.
java.io.EOFException Create breakpoint
    at org.pcap4j.core.PcapHandle.getNextRawPacketEx(PcapHandle.java:715)
    at org.pcap4j.core.PcapHandle.getNextPacketEx(PcapHandle.java:668)
    at org.pcaptokg.PcapParser.ParsePacketData(PcapParser.java:52)
    at org.pcaptokg.PacketAggregator.<init>(PacketAggregator.java:35)
    at org.pcaptokg.PcapToKg.main(PcapToKg.java:7)
15:13:19.238 [main] INFO org.pcap4j.core.PcapHandle -- Closed.
Processing packet 50 out of 3044
Processing packet 100 out of 3044
Processing packet 150 out of 3044
Processing packet 200 out of 3044
Processing packet 250 out of 3044
Processing packet 300 out of 3044
Processing packet 350 out of 3044
Processing packet 400 out of 3044
```

Εικόνα 8:Τέλος parsing και ανάλυση πακέτων

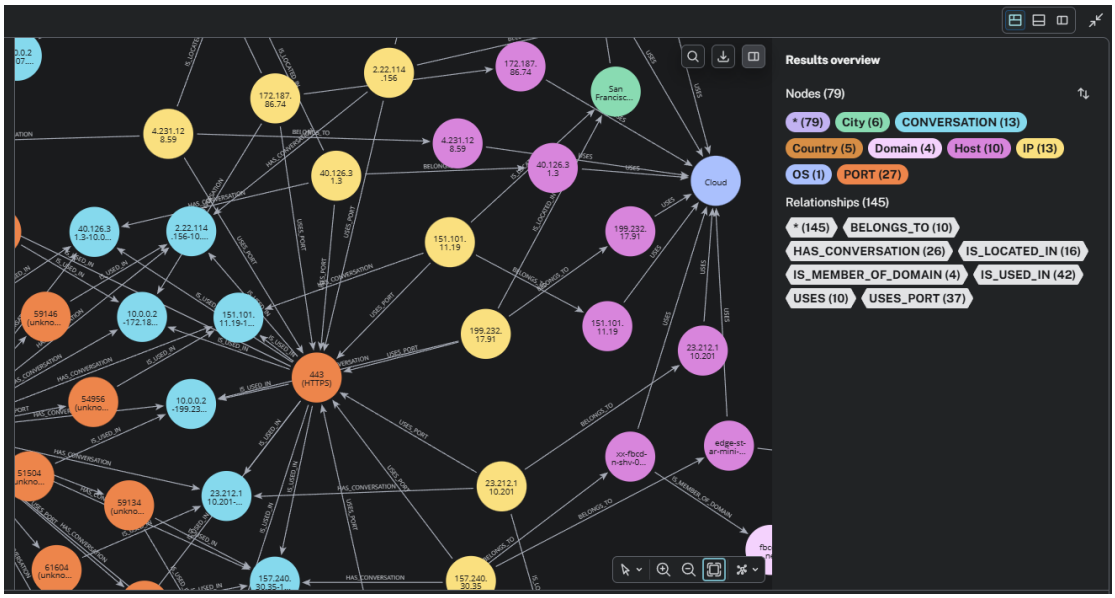
```
Ip 2.22.114.156 info already in database
Ip 40.126.31.3 info already in database
Ip 172.187.86.74 info already in database
Ip 199.232.17.91 info already in database
IP list build finished. Building conversation list
Successfully created conversation 10.0.0.2-172.187.86.74
Successfully created conversation 40.126.31.3-10.0.0.2
Successfully created conversation 10.0.0.138-10.0.0.2
Successfully created conversation 23.212.110.201-10.0.0.2
Successfully created conversation 10.0.0.2-199.232.17.91
Successfully created conversation 151.101.11.19-10.0.0.2
Successfully created conversation 10.0.0.2-34.107.221.82
Successfully created conversation 10.0.0.2-4.231.128.59
Successfully created conversation 10.0.0.2-157.240.30.27
Successfully created conversation 2.22.114.156-10.0.0.2
Successfully created conversation 10.0.0.138-239.255.255.250
Successfully created conversation 10.0.0.2-157.240.30.35
```

Εικόνα 9: Δημιουργία λίστας IP, Conversation

```
InternalResultSummary{query=Query{text='MERGE (ip:IP {address: $ipAddress, is_Bot: $isBot, is_Local: $isLocal, is_known_attacker: $isKnownAttacker, is_spam: $isSpam })
 MERGE (city:City {name: $city}) MERGE (ip)-[:IS_LOCATED_IN]->(city) MERGE (country:Country{name: $country}) MERGE (city)-[:IS_LOCATED_IN]->(country) MERGE (host: Host {hos
Adding IP 11 out of 13 to graph
true
InternalResultSummary{query=Query{text='MERGE (ip:IP {address: $ipAddress, is_Bot: $isBot, is_Local: $isLocal, is_known_attacker: $isKnownAttacker, is_spam: $isSpam })
 MERGE (port0: PORT{port_Number: 'null'})MERGE (ip)-[:USES_PORT]->(port0) ', parameters={isBot: FALSE, country: "null", city: "null", hostOs: "null", vpnScore: "null", ipAd
Adding IP 12 out of 13 to graph
true
InternalResultSummary{query=Query{text='MERGE (ip:IP {address: $ipAddress, is_Bot: $isBot, is_Local: $isLocal, is_known_attacker: $isKnownAttacker, is_spam: $isSpam })
 MERGE (port0: PORT{port_Number: '54956 (unknown)'})MERGE (ip)-[:USES_PORT]->(port0) MERGE (port1: PORT{port_Number: '55687 (unknown)'})MERGE (ip)-[:USES_PORT]->(port1) MER
Adding IP 13 out of 13 to graph
true
InternalResultSummary{query=Query{text='MERGE (ip:IP {address: $ipAddress, is_Bot: $isBot, is_Local: $isLocal, is_known_attacker: $isKnownAttacker, is_spam: $isSpam })
 MERGE (city:City {name: $city}) MERGE (ip)-[:IS_LOCATED_IN]->(city) MERGE (country:Country{name: $country}) MERGE (city)-[:IS_LOCATED_IN]->(country) MERGE (host: Host {hos
Adding conversation 1 out of 13 to graph
true
InternalResultSummary{query=Query{text='MATCH (a:IP {address: $sourceAddress})
MATCH (b:IP {address: $destinationAddress})
MERGE (conversation:CONVERSATION{id: $conversationId, bytes_sent:$bytesSent, bytes_received:$bytesReceived, sender: $sourceAddress, receiver: $destinationAddress})
MERGE (a)-[:HAS_CONVERSATION]->(conversation)
MERGE (b)-[:HAS_CONVERSATION]->(conversation)
 MERGE (port0: PORT{port_Number: '55687 (unknown)'})MERGE (port0)-[:IS_USED_IN]->(conversation) MERGE (port1: PORT{port_Number: '443 (HTTPS)'})MERGE (port1)-[:IS_USED_IN]->
Adding conversation 2 out of 13 to graph
true
```

Εικόνα 10: Εισαγωγή δεδομένων στο γράφημα

Πλέον το γράφημα έχει λάβει την τελική του μορφή, την οποία μπορούμε να δούμε από την εφαρμογή Neo4j Desktop, όπως φαίνεται στην Εικόνα 15:

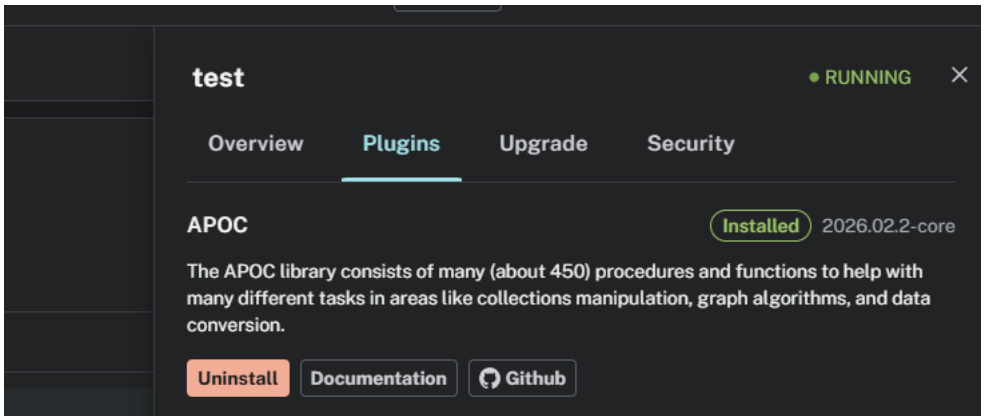


Εικόνα 11: Γράφημα γνώσης

Αφού έχουμε επιβεβαιώσει ότι οι κορυφές και οι ακμές έχουν επιτυχώς εισαχθεί στο γράφημα, είμαστε πλέον σε θέση να συνδέσουμε τη μηχανή της Neo4j με το Claude ή να εκτελέσουμε απευθείας queries, ώστε να λάβουμε τη χρήσιμη σε εμάς πληροφορία.

5.4 Εγκατάσταση MCP Server και διαμόρφωση Claude

Η διαδικασία εγκατάστασης του MCP Server είναι αρκετά απλή και περιγράφεται στην σχετική τεκμηρίωση. (MCP-neo4j, 2026). Αρχικά εγκαθιστούμε το APOC plugin στην Neo4j:



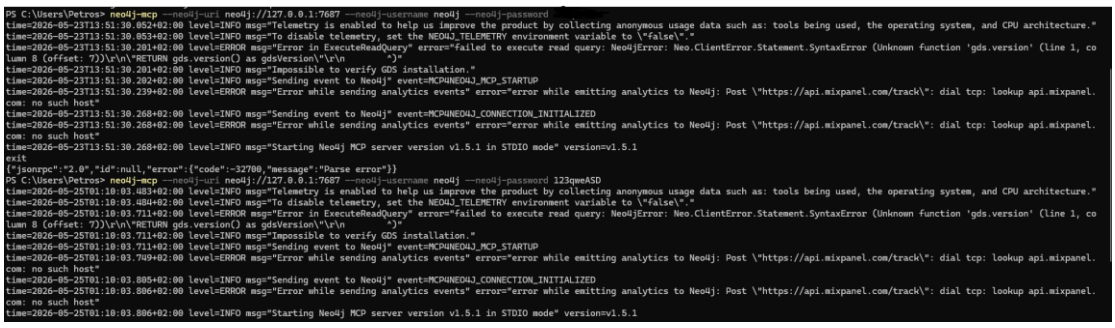
Εικόνα 12:Εγκατάσταση APOC

Κατόπιν, εγκαθιστούμε το εκτελέσιμο του MCP server. Επιπλέον εγκαθιστούμε το uv μέσω CLI:

```
pip install uv
```

Πλέον είμαστε έτοιμοι να ξεκινήσουμε τον server:

```
neo4j-mcp --neo4j-uri neo4j://127.0.0.1:7687 --neo4j-username neo4j --neo4j-password ***
```

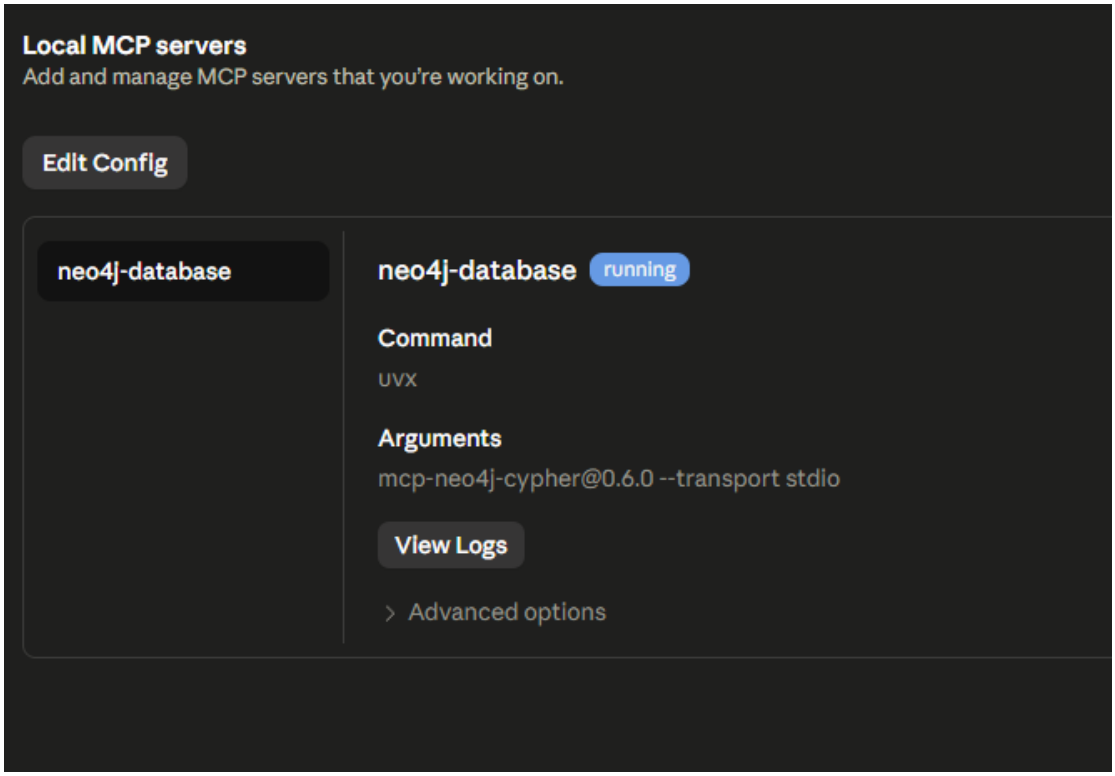


Εικόνα 13: Εκκίνηση MCP server

Το μόνο που μένει είναι η διαμόρφωση του Claude. Εισάγουμε τις παρακάτω παραμέτρους στο αρχείο claude_desktop_config.json και επανεκκινούμε το Claude.

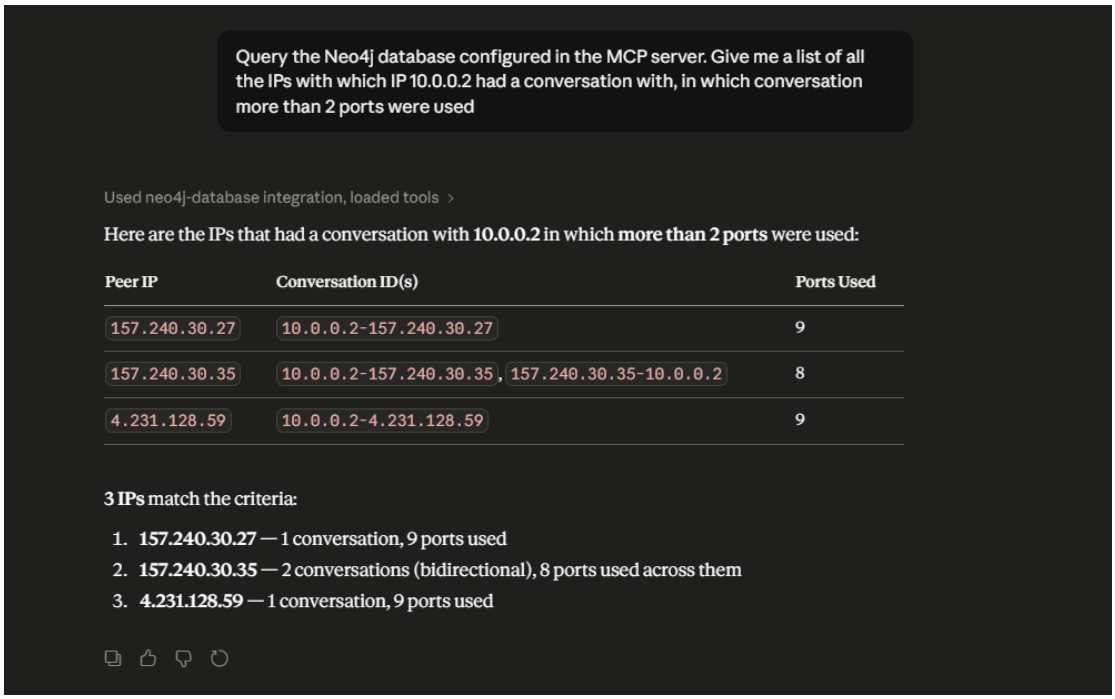
```
{
  "mcpServers": {
    "neo4j-database": {
      "command": "uvx",
      "args": [ "mcp-neo4j-cypher@0.6.0", "--transport", "stdio"  ],
      "env": {
        "NEO4J_URI": "bolt://localhost:7687",
        "NEO4J_USERNAME": "neo4j",
        "NEO4J_PASSWORD": "<your-password>",
        "NEO4J_DATABASE": "neo4j"
      }
    }
  }
}
```

Πλέον το σύστημα είναι έτοιμο και το Claude μπορεί να εκτελέσει queries απευθείας στο γράφημα γνώσης.



Εικόνα 14: Επιτυχής εγκατάσταση MCP Server

Εκτελούμε ένα Query για να εξετάσουμε την λειτουργικότητα:



Εικόνα 15: Πειραματικό query

6. Μέθοδος αξιολόγησης - πειραματικά δεδομένα

Σε αυτή την ενότητα θα εξετάσουμε τη μέθοδο αξιολόγησης επάνω σε πραγματικά δεδομένα

6.1 Εύρεση δεδομένων

Θα ανατρέξουμε στο διαδίκτυο για πηγές δεδομένων με αρχεία .pcap από προσομοιωμένες ή πραγματικές επιθέσεις. Στην περίπτωση μας, θα χρησιμοποιήσουμε αρχεία .pcap που βρίσκουμε στο Stratosphere IPS (Stratosphere, 2015). Θα κάνουμε συνολικά 10 αναλύσεις πάνω σε πειραματικά δεδομένα, εκ των οποίων 4 θα προέρχονται από Datasets με ύπαρξη κακόβουλου λογισμικού (Malware Captures), 4 θα προέρχονται από Datasets με φυσιολογική κίνηση (Normal Captures) και 2 θα προέρχονται από μικτές καταγραφές (Mixed Captures). Όπως αναφέρεται στη σχετική σελίδα του Stratosphere, τα Malware Captures περιέχουν δεδομένα από πραγματική κίνηση, προς υπολογισμό True Positive και False Negative εκτιμήσεων, τα Normal Captures περιέχουν δεδομένα από φυσιολογική διαδικτυακή κίνηση, προς υπλογισμό True Negative και False Positive, ενώ τα Mixed Captures έχουν μια αλληλουχία καθαρής κίνησης, κατόπιν μόλυνσης με κακόβουλο λογισμικό και τέλος καθαρισμού της μόλυνσης.

Επιλογή Dataset

Η Stratosphere παρέχει 349 καταγραφές με ύπαρξη κακόβουλης κίνησης, 32 καταγραφές φυσιολογικής κίνησης και 5 μικτές καταγραφές.

Η επιλογή για Malware Captures θα γίνει τυχαία με τη χρήση RNG της Google. Η επιλογή για Normal και Mixed θα γίνει σειριακά, αγνοώντας captures που έχουν αλλοιωμένα δεδομένα χάριν ιδιωτικότητας.

1. Malware captures:
    a. 46-Virut
    b. 210-Trojan.WisdomEyes
    c. 66-Sality

   d. 90-Conficker
2. Normal Captures:
   a. Normal Capture 28 - Linux notebook at home
   b. Normal Capture 26 - Windows computer, file execution
   c. Normal Capture 24 - Webpage traffic
   d. Normal Capture 20 - Normal HTTPS traffic
3. Mixed Captures
   a. Normal and infected with MD5 a0840a39ec90e1f603e2f4be42a87026
   b. Normal and infected with MD5 4d9838607597427f2dd6b1d2092f1e76

6.2 Εκτίμηση πειραμάτων

Για αρχή, τα δεδομένα που βρίσκονται στο γράφημα γνώσης θα διαγράφονται από σχετικό query της Neo4J. Στη συνέχεια, θα εκτελούμε το πρόγραμμα PcapToKg ώστε το γράφημα μας να περιλαμβάνει μόνο στοιχεία από το προς εξέταση Dataset. Εφόσον το γράφημα έχει δημιουργηθεί για ένα Dataset, θα εκτελούμε το ίδιο prompt σε νέα συνομιλία με το Claude για κάθε ένα από τα γραφήματα, ώστε να μην υπάρχουν συμφραζόμενα από προηγούμενες εκτελέσεις. Αναμένουμε για κάθε γράφημα μια αναλυτική απάντηση που να συνδέει την απεικόνιση της κίνησης με τακτικές επίθεσης βάσει του MITRE ATT&CK, αν και εφόσον υπάρχει, και μια καταληκτική απάντηση (θετική, αρνητική ή ουδέτερη) επάνω στο ερώτημα.

Εκτίμηση True Positive

Για τα Malware Captures και τα Mixed Captures, αναμένουμε από το LLM θετική αποτίμηση κακόβουλης κίνησης. Δεδομένου ότι το μοντέλο παράγει κείμενο και όχι ένα απλό «ναι» ή «όχι», για το χαρακτηρισμό True Positive αναμένουμε το κείμενο που θα παραχθεί θα πληροί τις παρακάτω προϋποθέσεις:

1. Σαφή θετική ανταπόκριση για την ύπαρξη κακόβουλης κίνησης

2. Αναφορές στον πίνακα MITRE ATT&CK με κωδικούς αναφοράς σε σχετικές τακτικές επίθεσης
3. Σύνδεση με τα στοιχεία του γραφήματος

Εκτίμηση False Positive

Σε περίπτωση που σε φυσιολογική καταγραφή (Normal Capture) το LLM δώσει θετική απόκριση, με ή χωρίς αναφορά στον πίνακα στον πίνακα ATT&CK, το αποτέλεσμα θα καταγράφεται ως False Positive. Υπάρχει η πιθανότητα το κείμενο που θα παραχθεί να μην παίρνει σαφή θετική θέση, αλλά να χρησιμοποιεί αβέβαιη γλώσσα.

Τέτοιες αποκρίσεις θα καταγράφονται ως False Positive μόνον εάν δούμε ότι ο λόγος που παράγεται «ωθεί» τον αναγνώστη σε υποψία για κακόβουλη κίνηση.

Εκτίμηση True Negative

Σε περίπτωση που σε φυσιολογική καταγραφή το LLM αποκριθεί ότι δεν υπάρχουν ενδείξεις επίθεσης, τότε το αποτέλεσμα θα καταγράφεται ως True Negative. Το μοντέλο, υπάρχει η πιθανότητα να αποκριθεί με εκτενή ανάλυση και να κάνει αναφορά στον πίνακα ATT&CK. Ωστόσο, εφόσον η κατάληξη (verdict) είναι αρνητική, θα καταγράφουμε το στιγμιότυπο σαν True Negative.

Εκτίμηση False Negative

Σε περίπτωση που το LLM αποκριθεί ότι δεν υπάρχουν ξεκάθαρες ενδείξεις επίθεσης μετά από ανάλυση Malware Capture, τότε θα θεωρήσουμε το αποτέλεσμα ως False Negative.

Εκτίμηση παραισθήσεων(Hallucinations)

Για κάθε επανάληψη του πειράματος, θα υπάρχει η δυνατότητα καταγραφής του αποτελέσματος ως παραίσθηση, ανεξάρτητα αν έχουμε True Positive, True Negative, False Positive ή False Negative εκτίμηση. Οι παραισθήσεις είναι ιδιαίτερα συχνό φαινόμενο που παρουσιάζεται κατά τη χρήση LLM και αποτελεί έναν από τους

μεγαλύτερους κινδύνους της τεχνολογίας αυτής. Στην παρούσα εργασία, λόγω χρονικών περιορισμών και αντικειμενικής δυσκολίας διασταύρωσης, **δεν** θα γίνει εξέταση παραισθήσεων, αλλά οφείλει να γίνει σε μελλοντικό χρόνο για τη διαπίστωση της εγκυρότητας των αποτελεσμάτων.

Εκτίμηση με ή χωρίς στοιχεία ασφάλειας

Το Geolocation API παρέχει στοιχεία ασφαλείας, όπως το flag “isKnownAttacker”. Θεωρούμε σημαντικό τέτοια στοιχεία να αποθηκεύονται στο γράφημα γνώσης, όμως είναι αναμενόμενο να επηρεάσουν την κρίση του μοντέλου. Για αυτό το λόγο, θα επαναλάβουμε για κάθε στιγμιότυπο δύο πειράματα: ένα που θα περιέχει στοιχεία ασφαλείας στο γράφημα, και ένα σε γράφημα που δεν θα περιέχει τέτοια στοιχεία στους αντίστοιχους κόμβους.

Επαναληψιμότητα

Τα LLM είναι μη-ντετερμινιστικά συστήματα (Ati et al, 2024), εννοώντας πως για το ίδιο ερώτημα μπορεί να δώσουν διαφορετικές απαντήσεις. Για αυτό το λόγο, είναι σημαντική η επανάληψη του πειράματος για κάθε στιγμιότυπο.

6.3 Μέθοδος αξιολόγησης

Δεδομένων όλων το παραμέτρων εκτίμησης που αναλύσαμε στην ενότητα 6.2, η τελική μεθοδολογία αξιολόγησης θα είναι η ακόλουθη:

1. Το γράφημα θα απαλείφεται στην αρχή κάθε πειράματος
2. Για κάθε Dataset θα γίνονται δύο ειδών πειράματα:
    - Το γράφημα περιέχει δεδομένα ασφαλείας
    - Το γράφημα δεν περιέχει δεδομένα ασφαλείας
3. Κάθε πείραμα θα επαναλαμβάνεται 3 φορές, ώστε να είναι φανερές οι πιθανές μεταβολές στην εκτίμηση του μοντέλου.
4. Συνολικά, για 10 datasets, θα γίνουν 20 καταγραφές στο γράφημα και 60 ερωτήματα στο LLM
5. Το ερώτημα θα είναι κάθε φορά το ίδιο
6. Θα αναπτύξουμε έναν πίνακα αποτελεσμάτων με τις ακόλουθες ενδείξεις: TP(True Positive), TN(True Negative), FP(False Positive), FN(False Negative).

Το ερώτημα που κάνουμε σε κάθε στιγμιότυπο θα έχει την ακόλουθη μορφή:

The database configured in Neo4j MCP contains a model of the traffic captured in a computer network over an amount of time. Query the MITRE ATT&CK matrix and the given graph, and produce a verdict on whether or not there are indications of malware or other cyber-attack in the computers of the local network, during the time of capture.

Δεν έχουν χρησιμοποιηθεί περαιτέρω παράμετροι, skills ή shared context. Το μοντέλο που χρησιμοποιείται είναι το Opus 4.7 σε περιβάλλον Claude Desktop με τις ρυθμίσεις που φαίνονται στο παράρτημα 9.4. Η αναζήτηση του Claude στον πίνακα ATT&CK είναι αδιαφανής και γίνεται από search του ίδιου του εργαλείου - δεν εισάγεται σαν context.

6.4 Ερμηνεία αναμενόμενων αποτελεσμάτων

Δεδομένης της περιορισμένης έκτασης της έρευνας, τα αποτελέσματα μπορούν να αποτελέσουν μόνον ενδείξεις για την αποτελεσματικότητα του συστήματος και δεν απαντάνε κατηγορηματικά στην αποτελεσματικότητα του. Ωστόσο, διακρίνουμε τις παρακάτω περιπτώσεις:

a) Το μοντέλο απαντάει σωστά στην πλειοψηφία (>50%) των στιγμιοτύπων, χωρίς σημαντικές διαφορές στην απόδοση μεταξύ των πειραμάτων που διενεργούνται όταν το γράφημα έχει δεδομένα ασφαλείας ή όχι.
b) Το μοντέλο απαντάει σωστά κατά πλειοψηφία(>50%) στα στιγμιότυπα που έχουν δεδομένα ασφαλείας
c) Το μοντέλο απαντάει λάθος κατά πλειοψηφία, ανεξάρτητα δεδομένων ασφαλείας.
d) Το μοντέλο απαντάει λάθος κατά πλειοψηφία, μόνο όταν υπάρχουν (ή δεν υπάρχουν) δεδομένα ασφαλείας.
e) Το μοντέλο παράγει αποτελέσματα που δεν συγκλίνουν για κάθε στιγμιότυπο.

Επίσης σημαντικό είναι να εξετάσουμε τις απαντήσεις - ακόμα και αν είναι έγκυρα θετικές ή αρνητικές, για μεταξύ τους αποκλίσεις.

7. Εκτέλεση πειραμάτων, ερμηνεία αποτελεσμάτων

Σε αυτή την ενότητα θα ασχοληθούμε με την πραγματική απόδοση του συστήματος σε πειραματικά δεδομένα και τα συμπεράσματα που μπορούμε να εξάγουμε.

7.1 Αποτελέσματα

Στους πίνακες που ακολουθούν σημειώνονται τα αποτελέσματα του πειράματος. Στη συνέχεια θα σχολιάσουμε για κάθε Dataset τα αποτελέσματα.

Σημειώνουμε ότι η επιλογή Dataset έχει γίνει τυχαία, συνεπώς είναι πιθανό ορισμένα από τα κακόβουλα αρχεία να έχουν πλήρως τοπική συμπεριφορά (άρα να μην αφήνουν διαδικτυακό αποτύπωμα). Ανεξάρτητα από την συμπεριφορά του εκάστοτε κακόβουλου λογισμικού, κάνουμε τις παρακάτω παραδοχές:

- Για κάθε Dataset Malware Capture (label MC-XXX) θεωρούμε ότι το αναμενόμενο αποτέλεσμα είναι θετικό.
- Για κάθε Dataset Normal Capture (label NC-XX) θεωρούμε ότι το αναμενόμενο αποτέλεσμα είναι αρνητικό
- Για κάθε Dataset Mixed Capture (label MX-X) θεωρούμε ότι το αναμενόμενο αποτέλεσμα είναι θετικό.

Σημειώνεται ότι το αναμενόμενο αποτέλεσμα επιλέγεται βάσει της ύπαρξης ή μη malware στον υπολογιστή, ανεξάρτητα αν το συγκεκριμένο malware αφήνει διαδικτυακό αποτύπωμα.

Στον πίνακα 1 καταγράφουμε όλα τα αποτελέσματα για κάθε ένα από τα 60 πειράματα. Στον πίνακα 2, συγκεντρώνουμε στατιστικά δεδομένα: αθροίσματα για κάθε ένδειξη (TP, TN, FP, FN, H) και λόγο προς το σύνολο.

Στους πίνακες 3,4,5 παρουσιάζεται το confusion matrix για κάθε υπότυπο και συνολικά, και υπολογίζεται η ακρίβεια (Accuracy - A), ειδικότητα (Specificity-S) και η ανάκληση (Recall-R) ως ακολούθως:

$$A=\frac{TP+TN}{P+N}, S=\frac{TN}{N}, R=\frac{TP}{P}$$

Στον πίνακα 6 υπολογίζουμε τον δείκτη συμφωνίας των αποτελεσμάτων με τη συνάρτηση διακύμανσης πληθυσμού:

$$\sigma=\sqrt{\frac{\sum_{i=1}^{n}\left(X_i\text{-}X_{avg}\right)^2}{n}}$$

Όπου σ η τιμή του δείκτη συμφωνίας, X_i η τιμή του αποτελέσματος (1 εάν το Χ ανήκει στην πλειοψηφία για ένα πείραμα, 0 εάν ανήκει στη μειοψηφία), X_{avg} ο μέσος όρος των αποτελεσμάτων και n το πλήθος στοιχείων. Σημειώνεται ότι η χρησιμότητα του δείκτη δεν έχει στατιστικό βάρος, καθότι το δείγμα επαναλήψεων είναι ιδιαίτερα μικρό (3 ή 6 πειράματα ανά dataset), και δίνεται σαν ποσοτικοποίηση της συμφωνίας των αποτελεσμάτων.

Οι υπολογισμοί γίνονται ξεχωριστά για τις περιπτώσεις με στοιχεία ασφαλείας (S) και για περιπτώσεις χωρίς στοιχεία ασφαλείας (NS) και συγκεντρωτικά (S+NS). Όλες οι τιμές έχουν στρογγυλοποίηση 2 δεκαδικών.

Σημειώνουμε ότι με 60 μετρήσεις το δείγμα είναι ενδεικτικό. Τα αποτελέσματα αφορούν τα πειράματα που διεξήχθησαν κατά την παρούσα έρευνα και δεν αποτελούν αντιπροσωπευτική εικόνα της αξιοπιστίας του μοντέλου.

Πίνακας 1: Αποτελέσματα

	Security Data in Graph			No Security Data provided			Baseline	Actual
Attempt	S1	S2	S3	NS1	NS2	NS 3	Trivial - always Positive	**Expected**
MC-66	TP	TP	TP	TP	TP	TP	TP	P
MC-90	TP	TP	TP	TP	TP	TP	TP	P
MC-210	**FN**	**FN**	**FN**	**FN**	**FN**	TP	TP	P
MC-46	TP	TP	TP	TP	TP	TP	TP	P
NC-20	TN	TN	**FP**	**FP**	TN	TN	FP	N
NC-24	TN	TN	TN	TN	TN	TN	FP	N
NC-26	TN	TN	TN	TN	TN	TN	FP	N
NC-28	TN	TN	TN	FP	FP	FP	FP	N
Mix-3	TP	TP	TP	FN	TP	TP	TP	P
Mix-4	TP	TP	FN	TP	FN	TP	TP	P

Πίνακας 2: Συγκεντρωμένα αποτελέσματα

	S	NS	S+NS	BL	S/TOTAL	NS/TOTAL	S+NS/TOTAL
TP	14	14	28	6	0,47	0,47	0,47
TN	11	8	19	0	0,37	0,27	0,32
FP	1	4	5	4	0,03	0,13	0,08
FN	4	4	8	0	0,13	0,13	0,13
TOTAL	30	30	60	10	1	1	1

Πίνακας 3: Confusion Matrix (S)

S	Verdict Positive	Verdict Negative
Actual Positive	14	4
Actual Negative	1	11
Accuracy	0,83	
Recall	0,78	
Specificity	0,91	

Πίνακας 4: Confusion Matrix (NS)

NS	Verdict Positive	Verdict Negative
Actual Positive	14	4
Actual Negative	4	8
Accuracy	0,73	
Recall	0,78	
Specificity	0,67	

Πίνακας 5: Confusion Matrix (S+NS)

S+NS	Calculated Positive	Calculated Negative
Actual Positive	28	8
Actual Negative	5	19
Accuracy	0,78	
Recall	0,78	
Specificity	0,79	

Πίνακας 6: Δείκτης συμφωνίας

	σ(S)	σ(NS)	σ(S+NS)
MC-66	0	0	0
MC-90	0	0	0
MC-210	0	0,47	0,37
MC-46	0	0	0
NC-20	0,47	0,47	0,47
NC-24	0	0	0
NC-26	0	0	0
NC-28	0	0	0,5
Mix-3	0	0,47	0,37
Mix-4	0,47	0,47	0,47
Average	0,09	0,18	0,21

7.2 Παρατηρήσεις επί των πειραμάτων

Όλες οι αναλύσεις του LLM παρατίθενται στο επισυναπτόμενο technical report.

MC-90

Ανάλυση του Stratosphere: https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-90/

Για την συγκεκριμένη καταγραφή το LLM βρήκε τα δεδομένα με ιδιαίτερη επιτυχία, εξιχνιάζοντας επιτυχώς το μολυσμένο υπολογιστή και δίνοντας συγκεκριμένες αναφορές στον πίνακα MITRE ATT&CK για τις τακτικές που χρησιμοποιήθηκαν στη συγκεκριμένη επίθεση. Αξίζει να σημειώσουμε ότι στις αναλύσεις δύο από τις τακτικές εμφανίστηκαν και στις 3 συνομιλίες, με όλες τις υπόλοιπες παραμέτρους, συμπεριλαμβανομένου του prompt ίδιες.

MC-46

Ανάλυση του Stratosphere: https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-46/

Για τη συγκεκριμένη καταγραφή είχαμε πλήρως True Positive αποτελέσματα, με ή χωρίς δεδομένα ασφαλείας. Το LLM ήταν σε θέση, με τα δεδομένα του γραφήματος, να εξιχνιάσει τον μολυσμένο υπολογιστή και να προτείνει βήματα ασφαλείας.

MC-66

Ανάλυση του Stratosphere: https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-66-1/

Το συγκεκριμένο capture είχε μια σημαντική διαφορά σε σχέση με τα υπόλοιπα: είχε το μεγαλύτερο αριθμό IPs (περίπου 160,000) και το μεγαλύτερο όγκο. Σύμφωνα με το Stratosphere είναι καταγραφή κίνησης 45 ημερών. Για το λόγο αυτό, η εισαγωγή έπρεπε να γίνει σε δύο φάσεις, αρχικά εισάγοντας τις IP και σε δεύτερη εκτέλεση εισάγοντας τα στοιχεία συνομιλιών. Επίσης ήταν σημαντικά μεγαλύτερος ο χρόνος εισαγωγής των δεδομένων (περίπου 4 ώρες, σε αντίθεση με <5 λεπτά για τις περισσότερες καταγραφές).

Το γράφημα που προέκυψε είναι ιδιαίτερα πολύπλοκο, αλλά το LLM έδωσε αρκετά περιεκτική και έγκυρη απόκριση σε όλες τις επαναλήψεις του πειράματος, με ή χωρίς δεδομένα ασφαλείας.

MC-210

Ανάλυση του Stratosphere: https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-210-1/

Για την συγκεκριμένη καταγραφή έχουμε False Negative αποτέλεσμα, σε όλες τις επαναλείψεις. Αξίζει όμως να σημειωθεί ότι η συμπεριφορά του συγκεκριμένου λογισμικού δεν φαίνεται να αφήνει διαδικτυακό αποτύπωμα και η ανάλυση του Stratosphere δεν είναι κατατοπιστική. Επιπλέον, στη σχετική σελίδα για το

VirusTotal, βλέπουμε ότι το συγκεκριμένο αρχείο δεν ανιχνεύεται ως malware από κανένα εμπορικό λογισμικό AntiVirus.

https://www.virustotal.com/gui/file/53ab071876dd528939b770eec5371681c3ff5a0ceca8774c4efe4f129392e885/detection

NC-20

Ανάλυση του Stratosphere: https://mcfp.felk.cvut.cz/publicDatasets/CTU-Normal-20/

Οι πρώτες εξετάσεις το NC-20 επέστρεψαν επιτυχώς True Negative αποτέλεσμα, όμως στην Τρίτη εξέταση το LLM συμπέρανε ότι υπάρχουν ενδείξεις κακόβουλης κίνησης. Αξίζει ωστόσο να σημειωθεί ότι το LLM σε αυτή την περίπτωση σωστά ανίχνευσε ότι ο υπολογιστής με IP 10.0.2.15 δεν είναι μολυσμένος (όπως αναφέρεται και στο Stratosphere), αλλά ο υπολογιστής 10.0.2.200 είναι πιθανώς μολυσμένος. Ανατρέχοντας στη βάση για συνομιλίες που περιέχουν την IP 10.0.2.200 βλέπουμε, όπως φαίνεται στην Εικόνα 20 ότι χρησιμοποιούνται πολλές θύρες σε συνομιλίες με IP οι οποίες είναι χαρακτηρισμένες ως γνωστοί επιτιθέμενοι.

Εικόνα 16:NC-20, Επικοινωνίες IP 10.0.2.200

NC-24

Ανάλυση του Stratosphere: https://mcfp.felk.cvut.cz/publicDatasets/CTU-Normal-24/

Το LLM ορθώς δεν ανίχνευσε ύποπτη κίνηση σε κανέναν υπολογιστή του δικτύου. Αξίζει να σημειωθεί ότι το LLM αναφέρει τακτικές οι οποίες δεν μπορούν να αποτυπωθούν στο τρέχον μοντέλο.

NC-26

Ανάλυση του Stratosphere: https://mcfp.felk.cvut.cz/publicDatasets/CTU-Normal-26/

Το Claude επιτυχώς εξιχνιάζει τον υπολογιστή του οποίου την κίνηση αναλύουμε και ορθώς αποφαίνεται ότι δεν υπάρχει ένδειξη επίθεση σε όλες τις επαναλήψεις.

NC-28

Ανάλυση του Stratosphere: https://mcfp.felk.cvut.cz/publicDatasets/CTU-Normal-28/

Με τα στοιχεία που εισάγαμε στο γράφημα γνώσης το LLM ήταν ικανό να αναγνωρίσει ότι δεν υπήρξε κακόβουλη κίνηση, αλλά μόνο εφόσον υπήρχαν λεπτομέρειες ασφαλείας στο γράφημα γνώσης. Εδώ βλέπουμε ότι τα στοιχεία ασφαλείας θα ήταν δυνατό να αποτρέψουν την ύπαρξη False Positive. Εξετάζοντας την απόκριση του LLM, βλέπουμε τα κάτωθι:

Indicators of concern

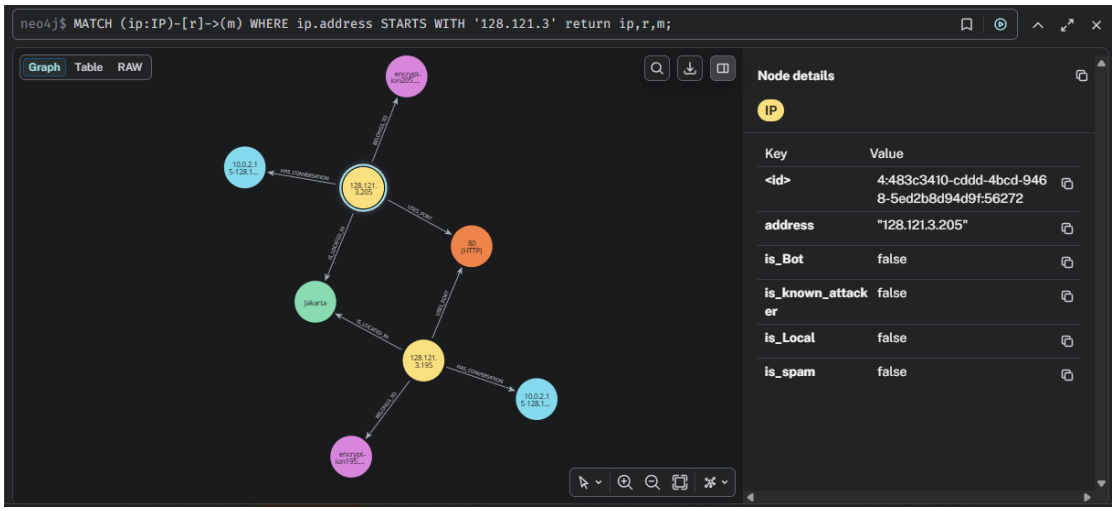
The notable findings cluster around a small number of atypical destinations:

Suspicious wildcard/randomized domains. *The host repeatedly contacted several IPs in the* ***185.42.119.0/24*** *block (Issy-les-Moulineaux, France) whose hosts resolve to wildcard records on randomly-named domains — *.darkishaeromedical.com and *.scatsideview.net. Random, semantically meaningless domain names combined with wildcard DNS and a shared netblock are a classic signature of malware C2 / domain-generation-algorithm infrastructure. The 185.42.119.x range has a long-standing reputation in threat intelligence for hosting exploit-kit and malware delivery infrastructure.*

Plain-HTTP contact with "encryption"-named hosts. *The host exchanged HTTP traffic with encryption195.afodgk128.jebeloutlet.com and encryption205.afodgk128.jebeloutlet.com (128.121.3.x, Jakarta), downloading ~47 KB over unencrypted port 80 from a host with a deceptive, machine-generated subdomain — a pattern consistent with second-stage payload delivery.*

Other oddly-named endpoints *include vps-109-123-210-154.nlooud.com (a VPS in Prague) and 204.2.197.204 (alt-e.pipelane.net).*

Με ένα απλό query στο γράφημα μας, βλέπουμε, για παράδειγμα, το κάτωθι σχήμα (Εικόνα 21), εκτελώντας το query MATCH (ip:IP)-[r]->(m) WHERE ip.address STARTS WITH '128.121.3' return ip,r,m;:



Εικόνα 17:Query results

Εδώ φαίνεται ότι το LLM παρουσιάζει παραίσθηση, καθότι αναφέρεται στο hostname “encryption205.afodgk128.jebeloutlet.com" ως machine-generated, ενώ φαίνεται να είναι host εμπορικού domain (jebel είναι όνομα, encryption205.afodgk128 φαίνονται να είναι ονόματα host του δικτύου). Δεν γνωρίζουμε με βεβαιότητα τα περιεχόμενα της συνομιλίας, και εξετάζοντας τα με Wireshark δεν προέκυψε κάποιο οριστικό συμπέρασμα για τα περιεχόμενα της κίνησης.

Η ερμηνεία του Stratosphere για τα περιεχόμενα της καταγραφής είναι ότι πρόκειται για συνδέσεις browser με site στις θέσεις 61-120 των πιο δημοφιλών site.

MX-3

Ανάλυση του Stratosphere: https://mcfp.felk.cvut.cz/publicDatasets/CTU-Mixed-Capture-3/

Η ερμηνεία των μικτών καταγραφών είναι και η δυσκολότερη. Σύμφωνα με το Stratosphere, η καταγραφή περιλαμβάνει δεδομένα από μόλυνση με adware. Ανατρέχοντας στο VirusTotal, η συντριπτική πλειοψηφία των παρόχων προγραμμάτων Anti-Virus δεν αναφέρουν το αρχείο ως κακόβουλο λογισμικό.

Τα αποτελέσματα ήταν θετικά όταν υπάρχουν δεδομένα ασφαλείας στο γράφημα.

Απόσπασμα από την ανάλυση του LLM κατά το πείραμα MX-3-S2:

Conclusion: Yes. There is a credible indication of malicious software on the local network — specifically the commercial spyware/stalkerware "mSpy" — on host 10.0.2.200.

What the graph contains

The capture models traffic for a single monitored workstation, 10.0.2.200, on the 10.0.2.0/24 subnet (gateway 10.0.2.2, plus broadcast/multicast addresses). It recorded 442 conversations with external IPs over HTTP and HTTPS only, totaling ~178 MB.

The key finding

The host communicated with mta-06.email.mspy.com (IP 82.196.8.230, Amsterdam, over HTTP). mSpy is a phone/computer surveillance app that promotes itself as a way to track children or monitor employees, but, like most spyware, is also widely used to monitor people without their consent — which is why these apps are also known as "stalkerware." It runs in the background to stay hidden on the target device and uploads collected data — messages, location, browsing, media — to an online dashboard. TechCrunchMageplaza

A clean workstation has no legitimate reason to contact mSpy's mail infrastructure. This is the kind of artifact mSpy generates when it is installed and reporting.

Mapping to MITRE ATT&CK

The presence of stalkerware corresponds to a recognizable behavioral chain:

- ***TA0009 – Collection / TA0010 – Exfiltration:** Stalkerware's core function is gathering local data and sending it to an attacker-controlled dashboard.*
- ***T1041 – Exfiltration Over C2 Channel:** Attackers leverage existing channels to transfer sensitive data back to attacker-controlled infrastructure, commonly*

abusing legitimate protocols such as HTTP/S to exfiltrate data unnoticed. The mSpy contact is over plain HTTP. Garnet

- ***T1020 – Automated Exfiltration:*** *mSpy uploads collected data automatically without operator interaction.*
- ***TA0011 – Command and Control:*** *The host maintains an outbound channel to the spyware operator's servers.*

Important caveats

The 13 IPs flagged in the database as is_known_attacker or is_Bot are ***not*** *strong evidence on their own. On inspection they resolve to Facebook, Google (1e100.net), Yahoo, and the Dotomi ad network — these are legitimate services, and the flags appear to be low-quality blocklist labels (likely tied to ad/tracker reputation). The traffic to them is single, ordinary HTTP/HTTPS conversations with no beaconing.*

Similarly, the overall traffic profile is otherwise benign: only ports 80/443 are used, there are no unusual service ports, no C2 beaconing pattern (almost every external IP appears in exactly one conversation), and the byte ratio is heavily inbound (~175 MB down vs ~3.4 MB up) — consistent with normal web browsing and streaming, not bulk data theft.

So the verdict rests specifically and solely on the ***mSpy contact****, which is unambiguous and does not have an innocent explanation.*

Παρατηρούμε ότι η ανάλυση και η αιτιολόγηση αναγνωρίζει ότι τα security flags, στη συγκεκριμένη περίπτωση είναι αναξιόπιστα και εστιάζει στις εγγραφές DNS.

Όταν αφαιρούμε τα flags από το γράφημα, τα αποτελέσματα παρεκκλίνουν μεταξύ τους. Στο στιγμιότυπο MX-3-NS2, το LLM βγάζει παρόμοιο συμπέρασμα με το παρατεθέν παραπάνω. Ωστόσο, στο στιγμιότυπο MX-3-NS1, βλέπουμε διαφορετική ανάλυση και κατάληξη:

Verdict

No clear indication of malware or cyber-attack on the local network during the capture. *The traffic profile is consistent with ordinary web browsing by a single host.*

What the graph shows

The capture models a single active local endpoint, ***10.0.2.200*** *(a 10.0.2.0/24 VirtualBox-style NAT network — the other "local" IPs are just the gateway 10.0.2.2*

and broadcast 10.0.2.255). That host had 442 conversations with 262 distinct external peers, downloading ~166 MB and uploading ~12 MB total.

Every indicator I checked points to normal activity:

***Protocols/ports** — All traffic is on **port 80 (HTTP)** and **443 (HTTPS)** only. No SSH, RDP, SMB, IRC, DNS-tunneling-style ports, or non-standard listening ports. No internal lateral movement (only one real local host exists).*

***Traffic direction** — Once the byte-direction convention is applied correctly (bytes_sent/bytes_received are relative to the conversation's sender field), the host is overwhelmingly a **downloader**: largest true outbound flow is ~448 KB, and inbound dwarfs outbound (~14:1). This is the opposite of the upload-heavy pattern expected from data exfiltration.*

***Destinations** — The heaviest peers resolve to **Akamai CDN, Google (1e100.net), Amazon AWS, CloudFront, Facebook/fbcdn, Dropbox, and CESNET** (Czech academic network). These are mainstream content-delivery and web services. The large multi-megabyte inbound transfers came from Akamai and Google — i.e., legitimate content downloads.*

***No beaconing** — The apparent "repeated" conversations are just paired bidirectional records of the same flow; no peer shows the regular, periodic, low-volume callback pattern characteristic of C2.*

***Geography** — Spread across the US, Western Europe, Singapore, etc. — typical for CDN-served browsing. A handful of single brief connections touch Russia, Iran, and Ukraine, but each is tiny (<4 KB) and isolated, with no follow-on activity — normal noise from ad networks and embedded web content, not a campaign.*

How this maps to MITRE ATT&CK

I evaluated the traffic against the ATT&CK techniques most relevant to network-only telemetry:

T1071.001 – Application Layer Protocol: Web Protocols *(Command & Control): adversaries hide C2 inside HTTP/HTTPS to blend with normal traffic. Not substantiated — while the host uses only HTTP/HTTPS, there is no asymmetric callback pattern, no persistent low-volume beacon, and destinations are reputable CDNs/services rather than uncategorized infrastructure.*

T1041 – Exfiltration Over C2 Channel*: stolen data leaves over an existing channel, identifiable by unusual outbound data volume. Not substantiated — outbound volume is small and inbound-dominated; there is no upload-heavy flow to an unexplained destination.*

Caveats

This is a confident-but-bounded conclusion. The dataset is ***flow-level metadata only*** *— no payloads, no timestamps, no DNS query names, no HTTP URLs or TLS SNI, and no process/host telemetry. That means a low-and-slow or well-disguised implant (small periodic beacons to a CDN-fronted or domain-fronted C2, which T1071/T1041 explicitly enable) could not be definitively ruled out from this graph alone. The absence of timestamps in particular prevents true beacon-interval analysis. Roughly 235 peers also had no reverse DNS, which is unremarkable for cloud/CDN IPs but does limit attribution.*

So: ***based on the available evidence, the capture shows benign web browsing with no signs of compromise.*** *For a definitive all-clear, you'd want packet-level capture (or Zeke/Suricata logs) with timestamps and DNS/TLS metadata to support proper beacon-timing and destination-reputation analysis.*

Παρατηρούμε ότι οι δύο αναλύσεις παρουσιάζουν σημαντικές διαφορές, με τη δεύτερη να μην αναγνωρίζει την ύπαρξη spyware από την κίνηση. Σημειώνουμε ότι στο πείραμα υπήρξε μόνο μία ανάλυση που χαρακτηρίζεται ως false negative.

MX-4

Ανάλυση του Stratosphere: https://mcfp.felk.cvut.cz/publicDatasets/CTU-Mixed-Capture-4/

Στο συγκεκριμένο Dataset είχαμε ως επί το πλείστον επιτυχή αναγνώριση ύπαρξης κακόβουλου λογισμικού. Υπάρχουν περιπτώσεις False Negative, τόσο με την δεδομένα ασφαλείας όσο και χωρίς. Έχει ιδιαίτερο ενδιαφέρον ότι για παρόμοιες IPs το LLM φαίνεται να δηλώνει διαφορετικές ερμηνείες κατά την επανάληψη του prompt:

Verdict: The host 10.0.2.200 shows strong indications of malware infection — specifically the Sality peer-to-peer botnet.

The local network

*The capture covers a small local network, 10.0.2.0/24, with effectively one active workstation: **10.0.2.200** (the rest are the gateway 10.0.2.2, broadcast, and multicast addresses). All 162 outbound conversations originate from this single host.*

Primary indicator — Sality P2P botnet traffic

*The decisive finding is **7 conversations on destination port 12350**, the signature port of the **Sality** peer-to-peer botnet. 10.0.2.200 exchanged small, roughly symmetric payloads (a few hundred to ~2,000 bytes each) with **10 different peer IPs** in the 91.190.216.x, 91.190.218.x, and 91.190.219.x ranges, located in Luxembourg and Ireland. This pattern — many short, bidirectional exchanges with a scattered set of peers on a single uncommon port — is exactly how a Sality bot maintains its peer list. Systems infected with Sality communicate over a peer-to-peer network to form a botnet to relay spam, proxy communications, exfiltrate sensitive data, and coordinate distributed tasks. In the Sality protocol, a peer can ask another peer for its list of URLs, send its own list, or ask for the coordinates (IP, port) of a third-party peer. WikipediaBotnets*

Supporting indicators

- ***Botnet peers lack reverse DNS.** Every 91.190.x peer has no hostname and no domain — the IP is its own "hostname." More broadly, 97 of 156 external destinations contacted by the host have no proper reverse DNS, consistent with Sality's habit of fetching payloads from disposable, non-named infrastructure.*

- ***Threat-intel flags.*** *Five external IPs are flagged in the graph (is_Bot / is_known_attacker). On inspection these resolve to legitimate services (Yandex, Facebook, Google, Yahoo), so I treat them as* ***false positives*** *from generic blocklists rather than evidence — but they show the host's traffic was broad enough to brush against flagged ranges.*
- ***No spam/SMTP activity observed.*** *There is no port 25/465/587 traffic, so the spam-relay capability of Sality was not exercised during this capture. The null-port conversations are benign (DNS to 8.8.8.8/8.8.4.4, multicast, Microsoft update servers).*

MITRE ATT&CK mapping

The observed behavior maps cleanly to these techniques:

- ***T1071 – Application Layer Protocol*** *and* ***T1095 – Non-Application Layer Protocol****: C2 communication over the custom port 12350.*
- ***T1090.003 – Proxy: Multi-hop / Peer-to-Peer Proxy****: the defining trait — C2 distributed across a P2P mesh of bot peers rather than a central server, making takedown difficult.*
- ***T1105 – Ingress Tool Transfer****: Sality's core function is fetching URL lists from peers and downloading further payloads.*
- ***T1571 – Non-Standard Port****: use of port 12350.*

Συγκρίνοντας με το στιγμιότυπο της προσπάθειας MX-4-S3:

Verdict: No indications of malware or cyber-attack

After querying the network graph against the MITRE ATT&CK framework, my assessment is that the captured traffic shows ***no credible evidence of malware infection or a cyber-attack*** *affecting the local network during the capture window.*

What the network actually is

The "local network" is a single sandbox/VM host: ***10.0.2.200*** *(the 10.0.2.x subnet with 10.0.2.2 as the NAT gateway and 10.0.2.255/224.0.0.x/239.255.255.250 being broadcast and multicast addresses — a textbook VirtualBox NAT layout). It is one workstation, not a corporate LAN, and it generated 286 conversations with 268 external IPs over the capture.*

Walking the MITRE ATT&CK tactics

I checked the graph against each adversary objective in the ATT&CK Enterprise matrix:

***Command & Control (TA0011)** — The 3 IPs flagged is_known_attacker (31.13.84.36, 31.13.84.4, 172.217.16.98) and the 2 flagged is_Bot (87.250.250.91, 77.238.185.35) resolve to **Facebook, Google, Yandex, and Yahoo** infrastructure. All traffic to them is ordinary HTTP/HTTPS on ports 80/443 with small, balanced byte counts. These flags are noisy threat-intel labels, not genuine C2 indicators — no beaconing pattern, no rare-port persistence, no long-lived low-volume channel to a single suspect endpoint.*

***Exfiltration (TA0010)** — The aggregate ratio is ~38 MB sent vs ~126 MB received: strongly inbound-dominant, the signature of a host consuming web content. The single largest upload is ~82 KB to an AWS EC2 address — consistent with a form POST, not bulk data theft.*

***Lateral Movement (TA0008) / Discovery (TA0007)** — There is only one host, so there is nothing to move laterally to. The only internal conversations are with the NAT gateway and the broadcast address — normal. No internal scanning.*

***Initial Access / Execution / Persistence** — Destinations are dominated by mainstream infrastructure: Google (1e100.net), Amazon AWS, Akamai CDN, Facebook, CloudFront, Dropbox. No connections to known malware-distribution or exploit-kit infrastructure.*

The one item worth a footnote

*Seven conversations use destination **port 12350** to a cluster of IPs in 91.190.216.0/24 and 91.190.218.0/24 (Luxembourg/Dublin). Port 12350 and that IP range are associated with the **Hotspot Shield / AnchorFree VPN service**. The traffic is small and bidirectional — consistent with VPN handshake/keepalive activity. This indicates a VPN client was running on the host. That is a policy consideration (it can be used to evade network monitoring), but it is not malware and not an attack — it's user software.*

Αξίζει να σημειώσουμε ότι σύμφωνα με την ανάλυση του Stratosphere, πρόκειται για μόλυνση locky Ransomware, αλλά το LLM αναφέρει ότι είναι πιθανή μόλυνση με Sality.

7.3 Ερμηνεία αποτελεσμάτων

Κρίνοντας από την θετική ή αρνητική απόκριση του LLM και μόνο, η απεικόνιση της διαδικτυακής κίνησης σε γράφημα γνώσης φαίνεται να αποτελεί ικανό εργαλείο ώστε το μοντέλο να διαπιστώσει την ύπαρξη επίθεσης ή μη, χωρίς να χρειάζεται εξαντλητική μελέτη όλων των πακέτων που έχουν κινηθεί μεταξύ των υπολογιστών στους οποίους έχουμε την καταγραφή. Φυσικά, τα συμπεράσματα δεν μπορούν να θεωρηθούν καταληκτικά και χρειάζεται περαιτέρω εμπεριστατωμένη εξέταση σε πειραματικά και πραγματικά δεδομένα και καταγραφές.

Ακρίβεια αποτελεσμάτων

Το σύστημα κατέγραψε ακρίβεια 0,83 στην περίπτωση που υπήρχαν δεδομένα ασφαλείας και 0,73 όταν αυτά απουσίαζαν. Αυτό δείχνει μια αύξηση 0,13 έως 0,23 σε σύγκριση με την ακρίβεια της τετριμμένης Positive-Only πρόβλεψης.

Στην πλειοψηφία των καταγραφών, το LLM απάντησε σωστά, ιδιαίτερα όταν το γράφημα περιείχε flags ασφαλείας. Κάτι που δεν περιμέναμε, είναι τα flags να βοηθήσουν το LLM να μειώσει τον αριθμό False Positive αποτελεσμάτων, αφού η μεγαλύτερη διαφορά, σύμφωνα με τον πίνακα 3, σημειώθηκε στην ακριβή απόφαση για True Negative αποτελέσματα. Η ακρίβεια των αποτελεσμάτων, φυσικά, εξαρτάται και από το κατά πόσο μια επίθεση είναι διαδικτυακά διακριτή από φυσιολογική κίνηση. Για παράδειγμα, στην καταγραφή MC-210, το κακόβουλο λογισμικό έχει ασαφή συμπεριφορά, δεν είναι καταγεγραμμένο από κανένα εμπορικό AntiVirus λογισμικό ως κακόβουλο και πιθανότατα δεν επιχειρεί καμία επαφή με εξωτερικές IP ή με άλλους υπολογιστές του δικτύου (εκτελώντας για παράδειγμα port scanning). Είναι αναμενόμενο το σύστημα να μην είναι σε θέση να ανιχνεύσει τέτοιες επιθέσεις.

Απόκλιση αποτελεσμάτων

Από τα δεδομένα του πίνακα 6 διαπιστώνουμε ότι ο δείκτης συμφωνίας είναι σημαντικά μεγαλύτερος στα πειράματα χωρίς δεδομένα ασφαλείας στο γράφημα γνώσης. Λόγω περιορισμένου dataset, η μέγιστη τιμή του δείκτη συμφωνίας είναι

0,47 για τα πειράματα S, NS και 0,5 συνολικά, όμως ο μέσος όρος, τόσο για τα πειράματα S, όσο και για τα πειράματα NS και συνολικά ήταν χαμηλός. Από αυτό συμπεραίνουμε ότι, αρχικά, τα δεδομένα ασφαλείας τείνουν να συνεισφέρουν σε λιγότερο αποκλίνοντα αποτελέσματα και ότι το LLM σε γενικές γραμμές καταλήγει σε παρόμοια συμπεράσματα όταν εξετάζει δεδομένα σε γράφημα γνώσης.

Αξίζει να σημειωθεί ότι η διαδικασία με την οποία το Claude αποφάσισε την εκτέλεση queries στο γράφημα γνώσης είναι αδιαφανής, συνεπώς είναι πιθανό σε κάθε πείραμα να εκτελέστηκαν διαφορετικά queries στη Neo4j, τα οποία δύνανται να ωθούν το μοντέλο σε διαφορετικά συμπεράσματα.

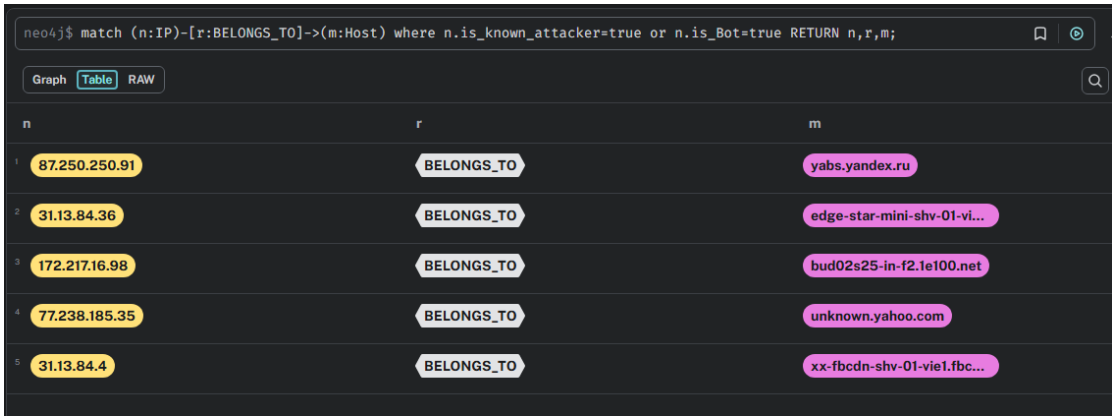
Τα συμπεράσματα είναι μη-καταληκτικά, λόγω εύρους πειραμάτων.

Αξία δεδομένων ασφαλείας

Αξίζει να σημειωθεί ότι πολλές φορές τα flags είναι αναξιόπιστα και το μοντέλο φαίνεται να είναι ικανό να ανιχνεύσει τέτοια ενδεχόμενα. Για παράδειγμα, στο παράδειγμα MX-4-S2 έχουμε την παρακάτω απόκριση:

- **Threat-intel flags.** Five external IPs are flagged in the graph (is_Bot / is_known_attacker). On inspection these resolve to legitimate services (Yandex, Facebook, Google, Yahoo), so I treat them as **false positives** from generic blocklists rather than evidence — but they show the host's traffic was broad enough to brush against flagged ranges.

Όπου, κάνοντας αντίστοιχα query στο γράφημα, παρατηρούμε ότι πράγματι IPs από νόμιμες υπηρεσίες έχουν χαρακτηριστεί ως bot ή ως κακόβουλες στις βάσεις δεδομένων του IPgeolocation API, όπως βλέπουμε στην Εικόνα 22



Εικόνα 18: Legitimate hosts flagged as attackers/bots

Κρίνοντας από το περιεχόμενο των απαντήσεων, το LLM ήταν σε θέση να αναγνωρίσει ότι τα flags είναι false-positive παρουσία στοιχείων hostname και domain, κάτι που επιβεβαιώνει τη χρησιμότητα των δεδομένων αυτών στο γράφημα γνώσης.

Κάτι που δεν φάνηκε να έχει ιδιαίτερη επιρροή στο αποτέλεσμα, είναι η ύπαρξη δεδομένων Λειτουργικού Συστήματος και VPN. Τα δεδομένα λειτουργικού συστήματος, ως επί το πλείστον, είχαν την ένδειξη “Cloud”. Αντιθέτως, δεδομένα VPN δεν προέκυψαν για κάποια από τις καταγραφές. Αυτό μπορεί να οφείλεται στο γεγονός ότι οι επιθέσεις δεν χρησιμοποίησαν VPN αλλά, πιθανόν, άλλες μεθόδους απόκρυψης IP ή τοποθεσίας (παραδείγματος χάριν χρήση proxy) ή πράγματι δεν υπήρξε τέτοια προσπάθεια.

Τα δεδομένα τοποθεσίας, φάνηκε να αναγνωρίζονται, αλλά δεν είναι εμφανές το κατά πόσο οδήγησαν το LLM σε επιτυχή ή μη απόφαση.

Ομοιομορφία απαντήσεων

Το prompt που δώσαμε στο LLM είναι απλό και δεν θέτει περιορισμούς για τη μορφή των απαντήσεων. Παρατηρούμε ότι σε όλες τις απαντήσεις έδωσε καταληκτική παράγραφο “Verdict”, όμως τα περιεχόμενα των απαντήσεων, ακόμα και σε διαδοχικά ερωτήματα για το ίδιο Dataset παρουσιάζουν διαφορές στη δομή, αλλά και στην ουσία, ήτοι οι αναφορές στον πίνακα ATT&CK δεν είναι πάντοτε οι ίδιες. Αυτό μας ωθεί σε περαιτέρω προβληματισμό, καθώς δεν είναι σαφές ποιες τακτικές εξετάζονται σε κάθε συνομιλία.

Για την αξιοποίηση των απαντήσεων, θα είναι προτιμότερο να συνταχθεί ένα σαφές πρότυπο απάντησης ως skill για το LLM, ώστε η μορφή κάθε απάντησης να είναι παρόμοια.

Συνεισφορά του μοντέλου

Αξιοσημείωτο είναι το γεγονός ότι στην πλειοψηφία των περιπτώσεων, το LLM ήταν σε θέση να αναγνωρίσει ότι ορισμένα από τα security flags που εισήχθησαν στο γράφημα είναι αναξιόπιστα, παρουσία δεδομένων DNS name και πιθανών αναζητήσεων του ίδιου του μοντέλου. Επίσης σημαντική είναι η ανίχνευση του spyware μέσω της επικοινωνίας του υπολογιστή ενδιαφέροντος με IP η οποία είχε DNS Name αντιστοιχιζόμενη με το λογισμικό mSpy, κάτι που δείχνει τόσο την αξία της πληροφορίας, όσο και την ικανότητα του μοντέλου να εξάγει συμπεράσματα και να χρησιμοποιήσει αιτιολογικές μεθόδους.

8. Συμπεράσματα, προκλήσεις - περιορισμοί κατά την ανάπτυξη, σημεία επέκτασης

Στο παρόν κεφάλαιο συνοψίζονται τα συμπεράσματα που προέκυψαν από την πειραματική αξιολόγηση του συστήματος και αποτυπώνονται συστηματικά οι περιορισμοί του, ανά στάδιο της αρχιτεκτονικής: από την καταγραφή και ανάλυση της δικτυακής κίνησης, έως την αποτύπωσή της στο γράφημα γνώσης και την αξιοποίησή της από το γλωσσικό μοντέλο. Κάθε περιορισμός συνοδεύεται από αντίστοιχη πρόταση μελλοντικής επέκτασης, ώστε το κεφάλαιο να λειτουργεί και ως οδικός χάρτης για τη συνέχιση της έρευνας.

8.1 Συμπεράσματα

Η μέθοδος καταχώρησης της κίνησης σε γράφημα γνώσης, σύμφωνα με τα πειραματικά αποτελέσματα της παρούσας εργασίας, δείχνει υποσχόμενη για την ανίχνευση ύποπτης κίνησης από LLM. Η εισαγωγή δεδομένων ασφαλείας φαίνεται να επηρεάζει θετικά την ακρίβεια του μοντέλου. Η οντολογία που περιγράψαμε στην ενότητα 3.1, φαίνεται να επαρκεί για μία πρώτη βελτίωση, αλλά κρίνεται απαραίτητη η εισαγωγή επιπλέον στοιχείων κίνησης, όπως στοιχεία χρονισμού (Timestamps).

Επιπλέον, κρίνεται απαραίτητη η διασταύρωση στοιχείων ασφαλείας από τα API ή δημιουργία μηχανισμού whitelist καθώς υπήρξαν στιγμιότυπα κατά τα οποία επέστρεφε False Positive ή False Negative αποτελέσματα. Είναι ωστόσο αξιοσημείωτο ότι στην πλειοψηφία των περιπτώσεων False Positive, το LLM ήταν σε θέση να διακρίνει ότι πρόκειται για σφάλμα των δεδομένων ασφαλείας.

Από τα παραπάνω συμπεραίνουμε ότι τα γραφήματα γνώσης δύνανται να αποτελέσουν ένα ισχυρό εργαλείο στον τομέα της κυβερνοασφάλειας, όμως τα δεδομένα της έρευνας δεν μπορούν να θεωρηθούν ακόμα καταληκτικά, καθότι το δείγμα χρήσης ήταν ιδιαίτερα μικρό.

8.2 Περιορισμοί κατά την καταγραφή της κίνησης

Ένα σύστημα ασφαλείας μπορεί να αναγνωρίζει επιθέσεις του παρελθόντος, από ήδη υπάρχοντα αρχεία και γνωστούς επιτιθέμενους, αλλά η πραγματική αξία του συστήματος φαίνεται σε πραγματικά δεδομένα και σε πραγματικό χρόνο. Το σύστημα που αναπτύχθηκε αποτελεί ένα πρωτότυπο το οποίο λειτουργεί μόνο σε ήδη καταγεγραμμένη κίνηση. Επιπλέον, δεν έχει υλοποιηθεί παράλληλη ανάλυση πολλών αρχείων (όπως θα γινόταν σε πραγματικές συνθήκες στο υπολογιστικό σύστημα ενός οργανισμού).

Προς μελλοντική επέκταση: επέκταση λειτουργικότητας για καταγραφή κίνησης σε πραγματικό χρόνο, υλοποίηση με threads για τη δυνατότητα παράλληλης καταγραφής και ανάλυσης κίνησης.

8.3 Περιορισμοί κατά την ανάλυση των πακέτων

Η βιβλιοθήκη Pcap4j, αν και κεντρικής σημασίας στην παρούσα μελέτη, παρουσιάζει ορισμένους λειτουργικούς περιορισμούς, εν μέρει λόγω της περιορισμένης πλέον συντήρησης του έργου.

Ένα αρκετά σημαντικό στοιχείο που δεν έχει αντιμετωπιστεί είναι η έλλειψη δεδομένων χρόνου για τα πακέτα που αναλύουμε, συνεπώς δεν έχουμε οπτική της κίνησης μεταξύ διευθύνσεων στο χρόνο. Αυτό συμβαίνει καθώς η βιβλιοθήκη Pcap4j δεν μπορεί να εξάγει αυτή την πληροφορία από αρχεία .pcap, αλλά μόνο από ζωντανή κίνηση.

Επιπλέον, μεγάλο πρόβλημα αποτελεί το γεγονός ότι μεγάλο μέρος της κίνησης δεν χρησιμοποιεί το πρωτόκολλο TCP, αλλά διαφορετικά πρωτόκολλα (όπως π.χ. το QUIC), τα οποία φέρουν διαφορετικές κεφαλίδες. Επίσης, δεν έχει γίνει πρόβλεψη για χρήση IPv6.

Προς μελλοντική επέκταση: εύρεση καταλληλότερης βιβλιοθήκης, ή συμβολή (ή fork) στην Pcap4j ώστε να είναι δυνατή η μελέτη χρονικής διάρκειας συνομιλιών ασύγχρονα. Μελέτη διαφορετικών πρωτοκόλλων επικοινωνίας και ανάλυση διαφορετικών τύπων πακέτων.

8.4 Περιορισμοί κατά την εύρεση στοιχείων γεωεντοπισμού και στοιχείων ασφαλείας

Το geolocation API δίνει πολλές πληροφορίες για τις IP, όμως δεν υπάρχει διασταύρωση των στοιχείων και δεν είναι σαφές το πώς προκύπτουν τα στοιχεία που μας δίνει. Επιπλέον, τα στοιχεία που δίνονται από το API μπορεί να φέρουν πολλές φορές false-positives, όπως για παράδειγμα όταν έχουμε κίνηση από public IP που ανήκουν σε υπολογιστικά νέφη (Cloud). Εάν κάποιος κακόβουλος χρήστης κάνει επιθέσεις χρησιμοποιώντας τέτοια IP, τότε θα καταγραφεί ως κακόβουλη. Ωστόσο, η IP αυτή, πιθανότατα θα χρησιμοποιείται και από μη κακόβουλους οργανισμούς ή υπηρεσίες. Το σύστημα μας δεν είναι σε θέση να διακρίνει αν πρόκειται για φυσιολογική ή κακόβουλη κίνηση. Επιπλέον, χρειάζεται υλοποίηση μεθόδου εύρεσης Domain, πιθανόν από διαφορετικό API.

Προς μελλοντική επέκταση: διασταύρωση στοιχείων με διαφορετικά API. Εύρεση context που θα επιτρέψει στο γλωσσικό μοντέλο να αποφανθεί αν η κίνηση είναι κακόβουλη ή μη.

8.5 Περιορισμοί κατά την καταγραφή στο γράφημα γνώσης

Το σύστημα διαχείρισης του γραφήματος γνώσης είναι σε στάδιο πρωτότυπου και έχει γίνει πρόβλεψη μόνο για την καταγραφή, αλλά όχι για την ενημέρωση και τη διαγραφή στοιχείων. Η Neo4j μέσω της εντολής Merge είναι σε θέση να ενημερώσει ήδη υπάρχοντες κόμβους, όμως αυτό είναι δυνατό να οδηγήσει σε διπλότυπους κόμβους εφόσον δεν υπάρχουν σαφείς όροι ενημέρωσης. Δεν αποτέλεσε πρόβλημα όταν στο γράφημα έχει πρόσβαση ένα μόνο στιγμιότυπο, αλλά σε περίπτωση

ταυτόχρονης σύνδεσης στην ίδια βάση από πολλά στιγμιότυπα του PcapToKg, σε διαφορετικούς υπολογιστές, το αποτέλεσμα θα ήταν αβέβαιο και πιθανώς θα δυσχέραινε την επίδοση του γλωσσικού μοντέλου.

Προς μελλοντική επέκταση: επέκταση λειτουργικότητας DbHandler, χρήση concurrent μεθόδων, δημιουργία uuid ή hash για κάθε κόμβο του γραφήματος, δημιουργία μεθόδων ενημέρωσης γραφήματος με νέα δεδομένα, δημιουργία μεθόδων διαγραφής κόμβων, βάσει κριτηρίων χρησιμότητας, έλεγχος διπλότυπων.

8.6 Περιορισμοί οντολογικού σχεδιασμού

Η οντολογία που έχουμε αναπτύξει βασίζεται στην οντολογία STUCCO με ορισμένες επεκτάσεις, αλλά και αρκετές ελλείψεις σε σχέση με την πρωτότυπη. Συγκεκριμένα, στο γράφημα μας δεν έχουμε συμφραζόμενα για τύπους Malware, Exploit σε γνωστές εφαρμογές, καθώς επίσης και δεδομένα για πρωτόκολλα, χρόνους και άλλα στοιχεία ενδιαφέροντος. Χρειάζεται διεύρυνση του πεδίου πειραματισμού με πιο πλήρη συμφραζόμενα και εκτενέστερο οντολογικό πλαίσιο. Ωστόσο η επέκταση της οντολογίας οφείλει να γίνει σε συνάρτηση με τη βελτίωση των πειραματικών αποτελεσμάτων, καθώς έγινε σαφές πως δεν βοηθούν όλα τα δεδομένα στην εξιχνίαση των επιθέσεων. Επιπλέον, ένα κενό είναι οι Bogon IP Addresses, οι οποίες περιλαμβάνουν Multicast/Broadcast διευθύνσεις, οι οποίες επίσης πρέπει να επισημανθούν στο παραχθέν σχήμα.

Προς μελλοντική επέκταση: προσθήκη οντοτήτων και σχέσεων στην οντολογία με έμφαση σε γνωστά κενά ασφαλείας εφαρμογών και επιπλέον στοιχείων διαδικτυακής κίνησης.

8.7 Περιορισμοί κατά την ερμηνεία των αποτελεσμάτων

Καθότι ο όγκος δεδομένων που παρήχθησαν από το LLM ήταν ιδιαίτερα μεγάλος και δεδομένου ότι η διασταύρωση εγκυρότητας είναι αρκετά χρονοβόρα, η εικόνα που

έχουμε για τις παραισθήσεις του LLM δεν είναι πλήρης. Χρειάζεται επιπλέον έρευνα και διασταύρωση των αποτελεσμάτων με τα γραφήματα, ώστε να αποφανθούμε για την ύπαρξη παραισθήσεων και να τεθούν κριτήρια για την απόφαση TP/TN/FP/FN που να αφορούν τη συνολική απάντηση του LLM.

Προς μελλοντική επέκταση: συστηματική διασταύρωση των αποκρίσεων του LLM με το γράφημα γνώσης και θέσπιση σαφών κριτηρίων ταξινόμησης TP/TN/FP/FN σε επίπεδο συνολικής απάντησης.

8.8 Περιορισμοί πειραματικών δεδομένων

Για την περαιτέρω αξιολόγηση του μοντέλου, απαιτούνται εμπεριστατωμένες συγκρίσεις του συστήματος με σημεία αναφοράς ως προς την ακρίβεια των αποτελεσμάτων, το κόστος υλοποίησης και την χρονική απόδοση. Χρειάζεται να μελετηθεί η απόδοση ενός LLM στην ανάλυση αρχείων καταγραφής πακέτων χωρίς την αποτύπωση σε γράφημα γνώσης, καθώς και η απόδοση όταν τα δεδομένα εισάγονται σε σχεσιακή βάση δεδομένων.

Προς μελλοντική επέκταση: διεύρυνση πεδίου πειραματισμού: μετρήσεις κόστους, χρόνου, συγκρίσεις με σημεία αναφοράς και εναλλακτικές αποτυπώσεις διαδικτυακής κίνησης.

8.9 Περιορισμοί LLM

Για την πλήρη αξιοποίηση του LLM είναι απαραίτητο να αναπτυχθούν ικανότητες (skills) και να εξεταστούν διαφορετικά prompts, τα οποία θα εξαλείψουν τις αποκλίσεις στη μορφή των απαντήσεων. Επιπλέον, χρειάζονται σαφείς οδηγίες για την ανίχνευση συγκεκριμένων τεχνικών επίθεσης.

Προς μελλοντική επέκταση: δημιουργία skills και κατάλληλων prompts για τη βελτιστοποίηση της μορφής των απαντήσεων.

8.10 Συνολική αποτίμηση

Παρά τους περιορισμούς που καταγράφηκαν στις προηγούμενες ενότητες, η παρούσα εργασία σχεδίασε και υλοποίησε ένα πλήρες, λειτουργικό σύστημα από άκρο σε άκρο: από τη σύλληψη και ανάλυση δικτυακής κίνησης με τη βιβλιοθήκη Pcap4j, τον εμπλουτισμό της με στοιχεία γεωεντοπισμού και ασφαλείας, την αποτύπωσή της σε γράφημα γνώσης στη Neo4j βάσει οντολογικού σχήματος θεμελιωμένου στην οντολογία STUCCO, έως και τη διασύνδεση του γραφήματος με Μεγάλο Γλωσσικό Μοντέλο μέσω του πρωτοκόλλου MCP. Τα πειραματικά ευρήματα, μολονότι βασίζονται σε περιορισμένο δείγμα, τεκμηριώνουν την κεντρική υπόθεση της εργασίας: ότι η δομημένη αναπαράσταση της δικτυακής κίνησης ως γράφημα γνώσης, ιδίως όταν εμπλουτίζεται με δεδομένα ασφαλείας, βελτιώνει μετρήσιμα την ικανότητα ενός γλωσσικού μοντέλου να εντοπίζει ύποπτη δραστηριότητα και να αιτιολογεί τα συμπεράσματά του — έως και τη διάκριση αναξιόπιστων ενδείξεων ασφαλείας ή τον εντοπισμό κακόβουλου λογισμικού. Υπό το πρίσμα αυτό, οι περιορισμοί που προηγήθηκαν δεν συνιστούν αδυναμίες της προσέγγισης, αλλά έναν σαφή ερευνητικό οδικό χάρτη: το σύστημα που αναπτύχθηκε αποτελεί επεκτάσιμη υποδομή αναφοράς, επάνω στην οποία μπορούν να θεμελιωθούν μελλοντικές εργασίες στη διασταύρωση γραφημάτων γνώσης, γλωσσικών μοντέλων και κυβερνοασφάλειας.

9.Παραρτήματα

9.1 Κώδικας

9.1.1 PcapToKG

```
package org.pcaptokg;

public class PcapToKg {
    public static void main(String[] args){
        PacketAggregator aggregator = new
PacketAggregator("src/main/resources/sample.pcap");

        DbHandler handler = new DbHandler("neo4j://127.0.0.1:7687",
"neo4j", "123qweASD");

        handler.getDriver().verifyConnectivity();

        int size = aggregator.getIpAddresses().size();
        int counter = 0;
        for (IpAddress ip : aggregator.getIpAddresses()){
            counter++;
            System.out.println("Adding IP "+counter+ " out of "
+size+ " to graph");
            handler.addIpToDb(ip);
        }
        size=aggregator.getConversations().size();
        counter=0;
        for (Conversation conversation :
aggregator.getConversations()){
            counter++;
            System.out.println("Adding conversation "+counter+ " out
of " +size + " to graph");
            handler.addConversation(conversation);
        }
        handler.deleteNullNodes();
        handler.getDriver().close();
    }
}
```

9.1.2 Packet Aggregator

```
package org.pcaptokg;

import java.util.ArrayList;
import java.util.concurrent.ConcurrentHashMap;

public class PacketAggregator {

    private PostgresHandler pgHandler;

    private ArrayList<PacketData> packets = new ArrayList<>();
    private ArrayList<IpAddress> ipAddresses = new ArrayList<>();
    private ArrayList<Conversation> conversations = new
ArrayList<>();

    private ConcurrentHashMap<String, Long> conversationLengthMap =
new ConcurrentHashMap<>();
    private ConcurrentHashMap<String, ArrayList<String>> ipMap = new
ConcurrentHashMap<>();
    private ConcurrentHashMap<String, ArrayList<String>>
conversationPortMap = new ConcurrentHashMap<>();

    public ConcurrentHashMap<String, Long> getConversationLengthMap()
{
        return conversationLengthMap;
    }

    public ConcurrentHashMap<String, ArrayList<String>> getIpMap() {
        return ipMap;
    }

    public PacketAggregator(String pcapFile){
        this.pgHandler=new PostgresHandler();
        PcapParser parser = new PcapParser(pcapFile);
        this.packets = parser.ParsePacketData();

        int packetCount = 0;
        int packetSize = packets.size();
        for(PacketData packet : this.packets){
```

```
            packetCount++;
            if(packetCount%50==0){System.out.println("Processing
packet "+packetCount+" out of "+packetSize );}
            mapPacket(packet);

        }
        this.aggregatePacketInfo();
        try{
            this.pgHandler.conn.close();
        }catch (Exception e){
            e.printStackTrace();
        }

    }



    public void addPortsToIpMap(String address, String port){
        if (this.ipMap.containsKey(address)){

if(!(this.ipMap.get(address).contains(port))&&(port!=null)){
                ArrayList<String> ipPorts = this.ipMap.get(address);
                ipPorts.add(port);
                this.ipMap.put(address, ipPorts);
            }
        }
        else{
            ArrayList<String>ipPorts=new ArrayList<>();
            ipPorts.add(port);
            this.ipMap.put(address,ipPorts);
        }
    }

    public void mapPacket(PacketData packet){
        String sourceAddress = packet.getSourceAddress();
        String destinationAddress = packet.getDestinationAddress();
        String sourcePort= (packet.getSourcePort()!=null) ?
packet.getSourcePort() : null;
        String destinationPort = (packet.getDestinationPort()!=null)
? packet.getDestinationPort() : null;
```

```
        Integer packetLength = (packet.getLengthInBytes()!=null) ?
packet.getLengthInBytes() : 0;
        String conversationTitle = packet.getSourceAddress() + "-" +
packet.getDestinationAddress();


if((sourceAddress.equals("Unknown"))||(destinationAddress.equals("Unk
nown"))){return;}
        if (this.conversationLengthMap.get(conversationTitle)==null){
            this.conversationLengthMap.put(conversationTitle,
Long.valueOf(packetLength));
        }else{
            this.conversationLengthMap.put(conversationTitle,
this.conversationLengthMap.get(conversationTitle)+packetLength);
        }



        ArrayList<String> convoPorts = new ArrayList<>();
        if (this.conversationPortMap.containsKey(conversationTitle)){

convoPorts.addAll(this.conversationPortMap.get(conversationTitle));
            if (!convoPorts.contains(sourcePort)){
                convoPorts.add(sourcePort);

conversationPortMap.put(conversationTitle,convoPorts);
            }
            if (!convoPorts.contains(destinationPort)){
                convoPorts.add(destinationPort);

conversationPortMap.put(conversationTitle,convoPorts);
            }
        }else{
            convoPorts.add(sourcePort);
            convoPorts.add(destinationPort);
            this.conversationPortMap.put(conversationTitle,
convoPorts );
        }

        this.addPortsToIpMap(sourceAddress,sourcePort);
```

```
        this.addPortsToIpMap(destinationAddress,destinationPort);

    }

    public IpAddress generateIpFromString(String Ip, String port){
        boolean found = false;
        IpAddress ipAdresss = null;
        for (IpAddress ip : this.ipAddresses){
            if (ip.getAddress().equals(Ip)){
                found=true;
                ipAdresss=ip;
                ip.addPort(port);
            }
        }
        if (!found){
            ipAdresss=new IpAddress(Ip, this.pgHandler);
            ipAdresss.addPort(port);
            this.ipAddresses.add(ipAdresss);
        }
        return ipAdresss;
    }

    public void  aggregatePacketInfo(){


        System.out.println("Packets mapped successfully. Building IP
list");
        int size = this.ipMap.keySet().size();
        int counter = 0;
        for (String ip : this.ipMap.keySet()){
            counter++;
            if (counter%20==0){System.out.println("Processing ip
"+counter + " out of " + size);}
            IpAddress currIp = new IpAddress(ip, this.pgHandler);
            currIp.setUsedPorts(this.ipMap.get(ip));
            this.ipAddresses.add(currIp);
        }

        System.out.println("IP list build finished. Building
conversation list");
```

```
        size = this.conversationLengthMap.keySet().size();
        counter = 0;
        for (String conversation :
this.conversationLengthMap.keySet()){
            counter++;
            if (counter%20==0){System.out.println("Processing
conversation "+counter + " out of " + size);}

            if (conversation==null){continue;}

            String[] convoParts =conversation.split("-");
            String reverseConversation=convoParts[1]+"-
"+convoParts[0];
            Long convoLength =
this.conversationLengthMap.get(conversation)!=null?
this.conversationLengthMap.get(conversation):0;
            Long reverseConvoLength =
this.conversationLengthMap.get(reverseConversation)!=null?
this.conversationLengthMap.get(reverseConversation):0;
            ArrayList<String> reverseConvoPorts =
this.conversationPortMap.get(reverseConversation)!=null?this.conversa
tionPortMap.get(reverseConversation):new ArrayList<>();


            Conversation currConv =new
Conversation(convoParts[0],convoParts[1],
                                        convoLength,
                                        reverseConvoLength,

this.conversationPortMap.get(conversation),
                                        reverseConvoPorts);
            this.conversationPortMap.remove(reverseConversation);
            this.conversationLengthMap.remove(reverseConversation);
            conversations.add(currConv);
        }

    }

    public ArrayList<PacketData> getPackets() {
        return packets;
```

```
    }

    public ArrayList<IpAddress> getIpAddresses() {
        return ipAddresses;
    }

    public ArrayList<Conversation> getConversations() {
        return conversations;
    }

}
```

9.1.3 Pcap Parser

```
package org.pcaptokg;

import java.io.EOFException;
import java.sql.Timestamp;
import java.util.ArrayList;

import org.pcap4j.core.NotOpenException;
import org.pcap4j.core.PcapHandle;
import org.pcap4j.core.PcapNativeException;
import org.pcap4j.core.Pcaps;
import org.pcap4j.packet.*;

@SuppressWarnings("javadoc")
public class PcapParser {


    private static final String PCAP_FILE_KEY =
PcapParser.class.getName() + ".pcapFile";
    private static String PCAP_FILE;

    public PcapParser(String PcapFile) {
        this.PCAP_FILE = PcapFile;

    }


    public ArrayList<PacketData> ParsePacketData() {
```

```
        ArrayList<PacketData> parsedPacketData = new ArrayList<>();

        PcapHandle handle;

        try {
            handle = Pcaps.openOffline(this.PCAP_FILE);

        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }

        long counter = 0;
        while (true) {
            try {
                counter++;
                System.out.println("Parsing next packet: "+counter);
                Packet packet = handle.getNextPacketEx();

                PacketData currentPacketData = new
PacketData(packet);
                System.out.println(currentPacketData);
                parsedPacketData.add(currentPacketData);
            } catch (EOFException e) {
                e.printStackTrace();
                System.out.println("Exhausted pcap file.");
                break;
            } catch (Exception e) {
                e.printStackTrace();

                System.err.println("There were errors in parsing
Packet information");
            }
        }
        handle.close();
        return parsedPacketData;
    }

}
```

9.1.4 PacketData

```
package org.pcaptokg;

import org.pcap4j.packet.IpV4Packet;
import org.pcap4j.packet.Packet;
import org.pcap4j.packet.TcpPacket;

import java.security.Timestamp;

public class PacketData {

    private String      sourceAddress;
    private String      destinationAddress;
    private Integer     lengthInBytes;
    private String      sourcePort;
    private String      destinationPort;
    private Timestamp       timestamp;

    private String   sourceIp;
    private String   destinationIp;



    public PacketData (Packet packet){
        this.sourceAddress=parsePacketSrcAddr(packet);
        this.destinationAddress=parsePacketDstAddr(packet);
        this.lengthInBytes=parsePacketlength(packet);
        this.sourcePort=parsePacketSrcPort(packet);
        this.destinationPort=parsePacketDstPort(packet);
        this.sourceIp=parsePacketSrcAddr(packet);
        this.destinationIp=parsePacketDstAddr(packet);

    }


    public Integer getLengthInBytes() {
        return lengthInBytes;
    }
```

```
public void setLengthInBytes(Integer lengthInBytes) {
    this.lengthInBytes = lengthInBytes;
}

public String getSourceAddress() {
    if (this.sourceAddress==null){return "Unknown";}
    return sourceAddress;
}

public void setSourceAddress(String sourceAddress) {
    this.sourceAddress = sourceAddress;
}

public String getDestinationAddress() {
    if (this.destinationAddress==null){
        return "Unknown";
    }
    return destinationAddress;
}

public String getSourceIp() {
    return sourceIp;
}

public void setSourceIp(String sourceIp) {
    this.sourceIp = sourceIp;
}

public String getDestinationIp() {
    return destinationIp;
}

public void setDestinationIp(String destinationIp) {
    this.destinationIp = destinationIp;
}

public void setDestinationAddress(String destinationAddress) {
    this.destinationAddress = destinationAddress;
}
```

```
    public String getSourcePort() {
        return sourcePort;
    }

    public void setSourcePort(String sourcePort) {
        this.sourcePort = sourcePort;
    }

    public String getDestinationPort() {
        return destinationPort;
    }

    public void setDestinationPort(String destinationPort) {
        this.destinationPort = destinationPort;
    }

    @Override
    public String toString() {
        return "PacketData{" +
                "lengthInBytes=" + lengthInBytes +
                ", sourceAddress='" + sourceAddress + '\'' +
                ", destinationAddress='" + destinationAddress + '\''
+
                ", sourcePort='" + sourcePort + '\'' +
                ", destinationPort='" + destinationPort + '\'' +
                '}';
    }



    public String parsePacketSrcAddr (Packet packet){
        String parsedSrcDst=null;
        try{

parsedSrcDst=packet.get(IpV4Packet.class).getHeader().getSrcAddr().to
String().substring(1);

        }catch(Exception e){
            //e.printStackTrace();
            System.out.println("Failed to parse source address from
```

```
packet");
        }

        return parsedSrcDst;
    }

    public Integer parsePacketlength (Packet packet){
        Integer length=0;
        try{
            length=packet.get(IpV4Packet.class).length();
        }catch(Exception e){
            //e.printStackTrace();
            System.out.println("Failed to parse packet length");
        }

        return length;
    }

    public String parsePacketDstAddr (Packet packet){
        String parsedDstAddr=null;
        try{

parsedDstAddr=packet.get(IpV4Packet.class).getHeader().getDstAddr().t
oString().substring(1);
        }catch(Exception e){
            //e.printStackTrace();
            System.out.println("Failed to parse destination address
from packet");
        }
        return parsedDstAddr;
    }

    public String parsePacketSrcPort (Packet packet){
        String parsedSrcPort=null;
        try{

parsedSrcPort=packet.get(TcpPacket.class).getHeader().getSrcPort().to
String();
        }catch(Exception e){
            //e.printStackTrace();
```

```
            System.out.println("Failed to parse source port from
packet");
        }
        return parsedSrcPort;
    }

    public String parsePacketDstPort (Packet packet){
        String parsedDstPort=null;
        try{

parsedDstPort=packet.get(TcpPacket.class).getHeader().getDstPort().to
String();
        }catch(Exception e){
            //e.printStackTrace();
            System.out.println("Failed to parse destination port from
packet");
        }
        return parsedDstPort;
    }
}
```

9.1.5 IpAddress

```
package org.pcaptokg;

import org.pcaptokg.pojo.GeoLocationApiResponse;
import org.pcaptokg.pojo.Location;
import org.pcaptokg.pojo.Security;
import tools.jackson.databind.ObjectMapper;

import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import java.util.concurrent.TimeUnit;

public class IpAddress {
    private String address;
    private ArrayList<String> usedPorts = new ArrayList<>();
    private boolean isLocal;
    private String hostName;
```

```
    private String dnsDomain=null;
    private boolean isAnonymous=false;
    private boolean isBot=false;
    private boolean isSpam=false;
    private boolean isCloudProvider=false;
    private String hostOs=null;
    private GeoLocationApiResponse apiResponse=null;
    private Location geolocation=null;
    private Vpn associatedVpn=null;
    private boolean isKnownAttacker=false;
    private PostgresHandler pgHandler;

//From api response


    public void setUsedPorts(ArrayList<String> usedPorts) {
        this.usedPorts = usedPorts;
    }

    public void addPort(String port){
        if (!this.usedPorts.contains(port)&&(port!=null)){
            this.usedPorts.add(port);
        }
    }

    public IpAddress(String address, PostgresHandler pgHandler) {
        this.address = "null";

        ObjectMapper mapper = new ObjectMapper();

        if (address!=null){

            this.address=address;
            this.isLocal=decideIfIpIsLocal(address);
            if (!this.isLocal){
                //PostgresHandler pgHandler=new PostgresHandler();
                boolean firstTry=true;
                boolean failedAttempt=false;
                int attemptNo=1;
                String dbResponse;
```

```
                do{
                    if (attemptNo==5){
                        failedAttempt=true;
                        System.out.println("Failed to find info for
IP "+this.address);
                    }
                     dbResponse=
pgHandler.inputIpResponseToDb(address);
                    if (firstTry){
                        firstTry=false;
                    }
                    else{
                        System.out.println("Attempt to gather info
failed, trying again...");
                        attemptNo++;

                    }

                }while (dbResponse==null&&(!failedAttempt));

                if(dbResponse!=null){
                    this.apiResponse=mapper.readValue(dbResponse,
GeoLocationApiResponse.class);
                }else{
                    System.out.println("Failed to import data in
local db. Attempting with regular API response");
                    this.apiResponse =
IpGeolocationApiClient.parseApiResponse(this.address);
                }



                if(this.apiResponse!=null){
                    this.hostName=apiResponse.getHostname();
                    this.geolocation =
this.apiResponse.getLocation();


                    Security secInfo =
this.apiResponse.getSecurity();
```

```
                    if (secInfo !=null){
                        Vpn vpn = new
Vpn(secInfo.getVpnProviderNames(),
                                secInfo.getVpnConfidenceScore());
                        if
(vpn.getName()!=null){this.associatedVpn=vpn;}

this.isKnownAttacker=secInfo.getIsKnownAttacker();
                        this.isCloudProvider=
secInfo.getIsCloudProvider();
                        this.isBot=secInfo.getIsBot();
                    }



                    this.hostOs=this.apiResponse.getUserAgent()==null
? null :
this.apiResponse.getUserAgent().getOperatingSystem().getName() ;

                    if
(!this.address.equals(this.hostName)&&(this.address!=null)){
                        try {
                            this.dnsDomain =
this.apiResponse.getHostname().replaceFirst("[.]",
",").split(",")[1];
                        } catch (Exception e) {
                            this.dnsDomain = "null";
                        }
                    }
                }
//                  try {
//                      pgHandler.conn.close();
//                  } catch (SQLException e) {
//                      throw new RuntimeException(e);
//                  }

            }
        }


    }
```

```
    public boolean decideIfIpIsLocal(String address){
        if (address==null){return false;}
        String[] strings = address.split("\\.");
        Integer[] ints = null;
        try{
            Integer[] mints = {Integer.parseInt(strings[0]),
Integer.parseInt(strings[1]),Integer.parseInt(strings[2]),Integer.par
seInt(strings[3]) };
            ints=mints;
        }catch(Exception e){
            return false;
        }
        //Private networks
//        if (ints[0]==10){return true;}
//        if ((ints[0]==172)&&(ints[1]>=16)&&(ints[1]<=31)){return
true;}
//        if ((ints[0]==192)&&(ints[1]==168)){return true;}
//        if ((ints[0]==127)){return true;}
//        if((ints[0]==169)&&(ints[1]==254)){return true;}
//        //Multicast
//        if ((ints[0]>239)&&ints[0]<239){return true;}
        int a = ints[0];
        int b = ints[1];
        int c = ints[2];
        int d = ints[3];

        return (
                // 0.0.0.0/8
                (a == 0) ||

                        // 10.0.0.0/8 (private)
                        (a == 10) ||

                        // 100.64.0.0/10 (carrier-grade NAT)
                        (a == 100 && b >= 64 && b <= 127) ||

                        // 127.0.0.0/8 (loopback)
                        (a == 127) ||
```

```
                        // 169.254.0.0/16 (link local)
                        (a == 169 && b == 254) ||

                        // 172.16.0.0/12 (private)
                        (a == 172 && b >= 16 && b <= 31) ||

                        // 192.168.0.0/16 (private)
                        (a == 192 && b == 168) ||

                        // 192.0.2.0/24 (TEST-NET-1)
                        (a == 192 && b == 0 && c == 2) ||

                        // 198.18.0.0/15 (benchmark testing)
                        (a == 198 && (b == 18 || b == 19)) ||

                        // 198.51.100.0/24 (TEST-NET-2)
                        (a == 198 && b == 51 && c == 100) ||

                        // 203.0.113.0/24 (TEST-NET-3)
                        (a == 203 && b == 0 && c == 113) ||

                        // 224.0.0.0/4 (multicast)
                        (a >= 224 && a <= 239) ||

                        // 240.0.0.0/4 (reserved)
                        (a >= 240) ||

                        // 255.255.255.255 (broadcast)
                        (a == 255 && b == 255 && c == 255 && d ==
255)
        );
    }

    private ArrayList<Conversation> conversationsWithIp= new
ArrayList<>();



    public Location getGeolocation() {
        return geolocation;
```

```
}

public void setGeolocation(Location geolocation) {
    this.geolocation = geolocation;
}

public String getHostName() {
    return hostName;
}

public void setHostName(String dnsName) {
    hostName = dnsName;
}

public boolean isLocal() {
    return isLocal;
}

public void setLocal(boolean local) {
    isLocal = local;
}

public ArrayList<String> getUsedPorts() {
    return usedPorts;
}

public String getAddress() {
    return address;
}

public void setAddress(String address) {
    this.address = address;
}

public String getDnsDomain() {
    return dnsDomain;
}

public void setDnsDomain(String dnsDomain) {
    this.dnsDomain = dnsDomain;
```

```
}

public boolean isAnonymous() {
    return isAnonymous;
}

public void setAnonymous(boolean anonymous) {
    isAnonymous = anonymous;
}

public boolean isBot() {
    return isBot;
}

public void setBot(boolean bot) {
    isBot = bot;
}

public boolean isSpam() {
    return isSpam;
}

public void setSpam(boolean spam) {
    isSpam = spam;
}

public boolean isCloudProvider() {
    return isCloudProvider;
}

public void setCloudProvider(boolean cloudProvider) {
    isCloudProvider = cloudProvider;
}

public String getHostOs() {
    return hostOs;
}

public void setHostOs(String hostOs) {
    this.hostOs = hostOs;
```

```
}

public GeoLocationApiResponse getApiResponse() {
    return apiResponse;
}

public void setApiResponse(GeoLocationApiResponse apiResponse) {
    this.apiResponse = apiResponse;
}

public Vpn getAssociatedVpn() {
    return associatedVpn;
}

public void setAssociatedVpn(Vpn associatedVpn) {
    this.associatedVpn = associatedVpn;
}

public boolean isKnownAttacker() {
    return isKnownAttacker;
}

public void setKnownAttacker(boolean knownAttacker) {
    isKnownAttacker = knownAttacker;
}


@Override
public String toString() {
    return "IpAddress{" +
            "address='" + address + '\'' +
            ", usedPorts=" + usedPorts +
            ", isLocal=" + isLocal +
            ", hostName='" + hostName + '\'' +
            ", dnsZone='" + dnsDomain + '\'' +
            ", isAnonymous=" + isAnonymous +
            ", isBot=" + isBot +
            ", isSpam=" + isSpam +
            ", isCloudProvider=" + isCloudProvider +
            ", hostOs='" + hostOs + '\'' +
```

```
                ", apiResponse=" + apiResponse +
                ", geolocation=" + geolocation +
                ", associatedVpn=" + associatedVpn +
                ", isKnownAttacker=" + isKnownAttacker +
                ", conversationsWithIp=" + conversationsWithIp +
                '}';
    }


}
```

9.1.6 PostgresHandler

```
package org.pcaptokg;


import org.pcaptokg.pojo.GeoLocationApiResponse;

import java.sql.*;
import java.util.Properties;

public class PostgresHandler {
    String url = "jdbc:postgresql://localhost/ipgeolocationdb";
    Properties props = new Properties();
    Connection conn = null;


    public PostgresHandler(){
        props.setProperty("user", "postgres");
        props.setProperty("password", "***");
        props.setProperty("ssl", "false");
        try{
            this.conn = DriverManager.getConnection(url, props);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }


    public static void main(String[] args){
```

```
        String url = "jdbc:postgresql://localhost/ipgeolocationdb";
        Properties props = new Properties();

        props.setProperty("user", "postgres");
        props.setProperty("password", "123qweASD");
        props.setProperty("ssl", "false");
        PostgresHandler handler = new PostgresHandler();
        //handler.inputIpResponseToDb("123.123.123.123");
        String ip = "123.122.123.12";
        System.out.println(handler.getIpResponseFromDb(ip));
        handler.inputIpResponseToDb(ip);

    }

    public String getIpResponseFromDb(String ip){
        String result = null;

        try{
            Statement st = conn.createStatement();
            ResultSet rs = st.executeQuery("SELECT api_response from
ip_response WHERE " +
                    "ip_address = '"+ip+"'");


            while (rs.next()){

                result=rs.getString(1);

            }
            st.close();
            rs.close();

        } catch (Exception e) {
            throw new RuntimeException(e);
        }
        return result;
    }


    public String inputIpResponseToDb(String ip){
```

```
        String result = null;
        String response = null;
        String inDb = this.getIpResponseFromDb(ip);

        if (inDb==null){

            try{
                System.out.println("Attempting to add IP "+ip+" to
local database");
                response=IpGeolocationApiClient.getRawResponse(ip);
                response = response.replaceAll("'","''");

                Statement st = conn.createStatement();

                st.executeUpdate("INSERT INTO ip_response
(ip_address, api_response)" +
                        "VALUES ('" + ip +"','"+response+"') ON
CONFLICT DO NOTHING");
                System.out.println("IP "+ip+" info successfully added
to the local database.");
                result= this.getIpResponseFromDb(ip);
                st.close();

            } catch (Exception e) {
                e.printStackTrace();

            }
        }else {
            System.out.println("Ip "+ip+" info already in database");
            return inDb;
        }
        return result;

    }
}
```

9.1.7 IpGeolocationApiClient

```
package org.pcaptokg;


import okhttp3.*;

import java.util.Map;
import java.util.HashMap;
//import com.fasterxml.jackson.*;
import tools.jackson.databind.ObjectMapper;
import org.pcaptokg.pojo.*;

public class IpGeolocationApiClient {

    public static final String apiKey="***";


    public static GeoLocationApiResponse parseApiResponse( String
Ip){

        String ApiKey = "***";

        Map<String, String> ipResponseMap = new HashMap <String,
String>();
        GeoLocationApiResponse responseObject = new
GeoLocationApiResponse();
        String url =
"https://api.ipgeolocation.io/v3/ipgeo?apiKey="+ApiKey+"&ip="+Ip+"&in
clude=security,user_agent,hostname";
        try{
            ObjectMapper mapper = new ObjectMapper();
            OkHttpClient client = new OkHttpClient().newBuilder()
                    .build();
            MediaType mediaType =
MediaType.parse("application/json");
            RequestBody body = RequestBody.create(mediaType, "");
            Request request = new Request.Builder()
                    .url(url)
                    .method("GET", null)
```

```
                    .build();
            Response response = client.newCall(request).execute();
            String jsonString = response.body().string();
            responseObject = mapper.readValue(jsonString,
GeoLocationApiResponse.class);

//ipResponseMap=mapper.readValue(jsonString,GeoLocationApiResponse);


        }
        catch (Exception e){
            e.printStackTrace();
        }



        return responseObject;
    }



    public static String getRawResponse(String Ip){
        String ApiKey = "***";

        Map<String, String> ipResponseMap = new HashMap <String,
String>();
        GeoLocationApiResponse responseObject = new
GeoLocationApiResponse();
        String url =
"https://api.ipgeolocation.io/v3/ipgeo?apiKey="+ApiKey+"&ip="+Ip+"&in
clude=security,user_agent,hostname";
        String jsonString=null;
        try{
            OkHttpClient client = new OkHttpClient().newBuilder()
                    .build();
            MediaType mediaType =
MediaType.parse("application/json");
            RequestBody body = RequestBody.create(mediaType, "");
            Request request = new Request.Builder()
                    .url(url)
```

```
                    .method("GET", null)
                    .build();
            Response response = client.newCall(request).execute();
            jsonString = response.body().string();

        }
        catch (Exception e){
            e.printStackTrace();
        }
        return jsonString;
    }
}
```

9.1.8 GeolocationApiResponse

```
package org.pcaptokg.pojo;

import java.util.LinkedHashMap;
import java.util.Map;
import javax.annotation.processing.Generated;
import com.fasterxml.jackson.annotation.JsonAnyGetter;
import com.fasterxml.jackson.annotation.JsonAnySetter;
import com.fasterxml.jackson.annotation.JsonIgnore;
import com.fasterxml.jackson.annotation.JsonInclude;
import com.fasterxml.jackson.annotation.JsonProperty;
import com.fasterxml.jackson.annotation.JsonPropertyOrder;

@JsonInclude(JsonInclude.Include.NON_NULL)
@JsonPropertyOrder({
    "ip",
    "hostname",
    "location",
    "country_metadata",
    "network",
    "currency",
    "asn",
    "company",
    "security",
    "abuse",
    "time_zone",
```

```
    "user_agent"
})
@Generated("jsonschema2pojo")
public class GeoLocationApiResponse {

    @JsonProperty("ip")
    private String ip;
    @JsonProperty("hostname")
    private String hostname;
    @JsonProperty("location")
    private Location location;
    @JsonProperty("country_metadata")
    private CountryMetadata countryMetadata;
    @JsonProperty("network")
    private Network network;
    @JsonProperty("currency")
    private Currency currency;
    @JsonProperty("asn")
    private Asn asn;
    @JsonProperty("company")
    private Company company;
    @JsonProperty("security")
    private Security security;
    @JsonProperty("abuse")
    private Abuse abuse;
    @JsonProperty("time_zone")
    private TimeZone timeZone;
    @JsonProperty("user_agent")
    private UserAgent userAgent;
    @JsonIgnore
    private Map<String, Object> additionalProperties = new
LinkedHashMap<String, Object>();

    /**
     * No args constructor for use in serialization
     *
     */
    public GeoLocationApiResponse() {
    }
```

```
    public GeoLocationApiResponse(String ip, String hostname,
Location location, CountryMetadata countryMetadata, Network network,
Currency currency, Asn asn, Company company, Security security, Abuse
abuse, TimeZone timeZone, UserAgent userAgent) {
        super();
        this.ip = ip;
        this.hostname = hostname;
        this.location = location;
        this.countryMetadata = countryMetadata;
        this.network = network;
        this.currency = currency;
        this.asn = asn;
        this.company = company;
        this.security = security;
        this.abuse = abuse;
        this.timeZone = timeZone;
        this.userAgent = userAgent;
    }

    @JsonProperty("ip")
    public String getIp() {
        return ip;
    }

    @JsonProperty("ip")
    public void setIp(String ip) {
        this.ip = ip;
    }

    @JsonProperty("hostname")
    public String getHostname() {
        return hostname;
    }

    @JsonProperty("hostname")
    public void setHostname(String hostname) {
        this.hostname = hostname;
    }

    @JsonProperty("location")
```

```
public Location getLocation() {
    return location;
}

@JsonProperty("location")
public void setLocation(Location location) {
    this.location = location;
}

@JsonProperty("country_metadata")
public CountryMetadata getCountryMetadata() {
    return countryMetadata;
}

@JsonProperty("country_metadata")
public void setCountryMetadata(CountryMetadata countryMetadata) {
    this.countryMetadata = countryMetadata;
}

@JsonProperty("network")
public Network getNetwork() {
    return network;
}

@JsonProperty("network")
public void setNetwork(Network network) {
    this.network = network;
}

@JsonProperty("currency")
public Currency getCurrency() {
    return currency;
}

@JsonProperty("currency")
public void setCurrency(Currency currency) {
    this.currency = currency;
}

@JsonProperty("asn")
```

```
public Asn getAsn() {
    return asn;
}

@JsonProperty("asn")
public void setAsn(Asn asn) {
    this.asn = asn;
}

@JsonProperty("company")
public Company getCompany() {
    return company;
}

@JsonProperty("company")
public void setCompany(Company company) {
    this.company = company;
}

@JsonProperty("security")
public Security getSecurity() {
    return security;
}

@JsonProperty("security")
public void setSecurity(Security security) {
    this.security = security;
}

@JsonProperty("abuse")
public Abuse getAbuse() {
    return abuse;
}

@JsonProperty("abuse")
public void setAbuse(Abuse abuse) {
    this.abuse = abuse;
}

@JsonProperty("time_zone")
```

```
    public TimeZone getTimeZone() {
        return timeZone;
    }

    @JsonProperty("time_zone")
    public void setTimeZone(TimeZone timeZone) {
        this.timeZone = timeZone;
    }

    @JsonProperty("user_agent")
    public UserAgent getUserAgent() {
        return userAgent;
    }

    @JsonProperty("user_agent")
    public void setUserAgent(UserAgent userAgent) {
        this.userAgent = userAgent;
    }

    @JsonAnyGetter
    public Map<String, Object> getAdditionalProperties() {
        return this.additionalProperties;
    }

    @JsonAnySetter
    public void setAdditionalProperty(String name, Object value) {
        this.additionalProperties.put(name, value);
    }

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();

sb.append(GeoLocationApiResponse.class.getName()).append('@').append(
Integer.toHexString(System.identityHashCode(this))).append('[');
        sb.append("ip");
        sb.append('=');
        sb.append(((this.ip == null)?"<null>":this.ip));
        sb.append(',');
        sb.append("hostname");
```

```
        sb.append('=');
        sb.append(((this.hostname == null)?"<null>":this.hostname));
        sb.append(',');
        sb.append("location");
        sb.append('=');
        sb.append(((this.location == null)?"<null>":this.location));
        sb.append(',');
        sb.append("countryMetadata");
        sb.append('=');
        sb.append(((this.countryMetadata ==
null)?"<null>":this.countryMetadata));
        sb.append(',');
        sb.append("network");
        sb.append('=');
        sb.append(((this.network == null)?"<null>":this.network));
        sb.append(',');
        sb.append("currency");
        sb.append('=');
        sb.append(((this.currency == null)?"<null>":this.currency));
        sb.append(',');
        sb.append("asn");
        sb.append('=');
        sb.append(((this.asn == null)?"<null>":this.asn));
        sb.append(',');
        sb.append("company");
        sb.append('=');
        sb.append(((this.company == null)?"<null>":this.company));
        sb.append(',');
        sb.append("security");
        sb.append('=');
        sb.append(((this.security == null)?"<null>":this.security));
        sb.append(',');
        sb.append("abuse");
        sb.append('=');
        sb.append(((this.abuse == null)?"<null>":this.abuse));
        sb.append(',');
        sb.append("timeZone");
        sb.append('=');
        sb.append(((this.timeZone == null)?"<null>":this.timeZone));
        sb.append(',');
```

```
        sb.append("userAgent");
        sb.append('=');
        sb.append(((this.userAgent ==
null)?"<null>":this.userAgent));
        sb.append(',');
        sb.append("additionalProperties");
        sb.append('=');
        sb.append(((this.additionalProperties ==
null)?"<null>":this.additionalProperties));
        sb.append(',');
        if (sb.charAt((sb.length()- 1)) == ',') {
            sb.setCharAt((sb.length()- 1), ']');
        } else {
            sb.append(']');
        }
        return sb.toString();
    }

}
```

9.1.9 Conversation

```
package org.pcaptokg;

import java.util.ArrayList;

public class Conversation {
    String conversationId;

    private IpAddress Ip1;
    private IpAddress Ip2;
    private long lengthOfConversationInBytes;

    private ArrayList<String> portsUsed=new ArrayList<>();

    private int amountOfPacketsSentFromIp1;
    private long lengthOfPacketsSentFromIp1;
    private ArrayList<String> portsUsedOnIp1=new ArrayList<>();

    private int amountOfPacketsSentFromIp2;
```

```
    private long lengthOfPacketsSentFromIp2;
    private ArrayList<String> portsUsedOnIp2=new ArrayList<>();

    private String Ip1str;
    private String Ip2str;


    public boolean containsIp(String address){
        if (address==null){return false;}
        boolean
containsIp=((address.equals(Ip1.getAddress()))||(address.equals(Ip2.g
etAddress())));

        return containsIp;
    }

    public ArrayList<String> getPortsUsed() {
        return portsUsed;
    }

    public String getConversationId() {
        return conversationId;
    }

    public void addPortToList(String port, boolean usedOnIp1){
        if (!this.portsUsed.contains(port)){portsUsed.add(port);}
        if (usedOnIp1){
            if (this.portsUsedOnIp1.isEmpty()){
                portsUsedOnIp1.add(port);
            }else{
                if (!portsUsedOnIp1.contains(port)){
                    portsUsedOnIp1.add(port);
                }
            }
        } else{
            if (this.portsUsedOnIp2.isEmpty()){
                portsUsedOnIp2.add(port);
            }else{
                if (!portsUsedOnIp2.contains(port)){
                    portsUsedOnIp2.add(port);
```

```
                }
            }
        }

    }




    public IpAddress getIp1() {
        return Ip1;
    }

    public void setIp1(IpAddress ip1) {
        Ip1 = ip1;
    }

    public IpAddress getIp2() {
        return Ip2;
    }

    public void setIp2(IpAddress ip2) {
        Ip2 = ip2;
    }

    public long getLengthOfConversationInBytes() {
        return lengthOfConversationInBytes;
    }

    public void setLengthOfConversationInBytes(long
lengthOfConversationInBytes) {
        this.lengthOfConversationInBytes =
lengthOfConversationInBytes;
    }

    public int getAmountOfPacketsSentFromIp1() {
        return amountOfPacketsSentFromIp1;
    }

    public void setAmountOfPacketsSentFromIp1(int
```

```
amountOfPacketsSentFromIp1) {
        this.amountOfPacketsSentFromIp1 = amountOfPacketsSentFromIp1;
    }

    public long getLengthOfPacketsSentFromIp1() {
        return lengthOfPacketsSentFromIp1;
    }

    public void setLengthOfPacketsSentFromIp1(long
lengthOfPacketsSentFromIp1) {
        this.lengthOfPacketsSentFromIp1 = lengthOfPacketsSentFromIp1;
    }

    public int getAmountOfPacketsSentFromIp2() {
        return amountOfPacketsSentFromIp2;
    }

    public void setAmountOfPacketsSentFromIp2(int
amountOfPacketsSentFromIp2) {
        this.amountOfPacketsSentFromIp2 = amountOfPacketsSentFromIp2;
    }

    public long getLengthOfPacketsSentFromIp2() {
        return lengthOfPacketsSentFromIp2;
    }

    public void setLengthOfPacketsSentFromIp2(long
lengthOfPacketsSentFromIp2) {
        this.lengthOfPacketsSentFromIp2 = lengthOfPacketsSentFromIp2;
    }

    public Conversation(IpAddress ip1, IpAddress ip2, long
lengthOfConversationInBytes, int amountOfPacketsSentFromIp1, long
lengthOfPacketsSentFromIp1, int amountOfPacketsSentFromIp2, long
lengthOfPacketsSentFromIp2) {
        Ip1 = ip1;
        Ip2 = ip2;
        this.lengthOfConversationInBytes =
lengthOfConversationInBytes;
        this.amountOfPacketsSentFromIp1 = amountOfPacketsSentFromIp1;
```

```
        this.lengthOfPacketsSentFromIp1 = lengthOfPacketsSentFromIp1;
        this.amountOfPacketsSentFromIp2 = amountOfPacketsSentFromIp2;
        this.lengthOfPacketsSentFromIp2 = lengthOfPacketsSentFromIp2;

        this.Ip1str=Ip1.getAddress();
        this.Ip2str=Ip2.getAddress();
    }

    public Conversation(IpAddress Ip1, IpAddress Ip2) {
        this.Ip1 = Ip1;
        this.Ip2 = Ip2;

        this.Ip1str=Ip1.getAddress();
        this.Ip2str=Ip2.getAddress();

        this.conversationId=Ip1.getAddress()+"-"+Ip2.getAddress();
    }


    public Conversation(String ip1str, String ip2str, long
lengthOfPacketsSentFromIp1, long lengthOfPacketsSentFromIp2) {
        Ip1str = ip1str;
        Ip2str = ip2str;
        this.lengthOfPacketsSentFromIp1 = lengthOfPacketsSentFromIp1;
        this.lengthOfPacketsSentFromIp2 = lengthOfPacketsSentFromIp2;
        this.conversationId=Ip1str+"-"+Ip2str;
    }

    public Conversation(String ip1str, String ip2str, long
lengthOfPacketsSentFromIp1, long lengthOfPacketsSentFromIp2,
ArrayList<String> portsUsedOnIp1, ArrayList<String> portsUsedOnIp2) {
        Ip1str = ip1str;
        Ip2str = ip2str;
        this.lengthOfPacketsSentFromIp1 = lengthOfPacketsSentFromIp1;
        this.lengthOfPacketsSentFromIp2 = lengthOfPacketsSentFromIp2;
        this.portsUsedOnIp1 = portsUsedOnIp1;
        this.portsUsedOnIp2 = portsUsedOnIp2;


if(portsUsedOnIp1!=null){this.portsUsed.addAll(portsUsedOnIp1);}
```

```
if(portsUsedOnIp2!=null){this.portsUsed.addAll(portsUsedOnIp2);}
        this.conversationId=Ip1str+"-"+Ip2str;
        System.out.println("Successfully created conversation "+
conversationId);
    }

    public String getIp2str() {
        return Ip2str;
    }

    public String getIp1str() {
        return Ip1str;
    }

    public ArrayList<String> getPortsUsedOnIp2() {
        return portsUsedOnIp2;
    }

    public ArrayList<String> getPortsUsedOnIp1() {
        return portsUsedOnIp1;
    }
}
```

9.1.10 DbHandler

```
package org.pcaptokg;

import org.neo4j.driver.*;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;


public class DbHandler {

    private String dbUri;
    private String dbUser;
    private String dbPass;
```

```
    private Driver driver=null;

    public DbHandler(String dbUri, String dbUser, String dbPass){
        this.dbUri=dbUri;
        this.dbUser=dbUser;
        this.dbPass=dbPass;

        try{
            this.driver = GraphDatabase.driver(this.dbUri,
AuthTokens.basic(this.dbUser, this.dbPass));


        }catch (Exception e){
            e.printStackTrace();
        }
    }




    public Driver getDriver(){
        return this.driver;
    }

    public EagerResult addIpToDb(IpAddress ip){
        String currIp = ip.getAddress();
        EagerResult result = null;

        if(currIp.equals("-1")){return null;}
        HashMap<String,Object> params = new
HashMap<>(Map.ofEntries(Map.entry("ipAddress", ip.getAddress()),
                Map.entry("isLocal", ip.isLocal()),
                Map.entry("isSpam", ip.isSpam()),
                Map.entry("isBot", ip.isBot()),
                Map.entry("isKnownAttacker", ip.isKnownAttacker()),



                Map.entry("hostname", ip.getHostName()==null ? "null"
```

```
: ip.getHostName()),

                Map.entry("dnsDomain", ip.getDnsDomain()==null ?
"null" : ip.getDnsDomain()),


                Map.entry("city", ip.getGeolocation()==null ? "null"
: ip.getGeolocation().getCity()),

                Map.entry("country",  ip.getGeolocation()==null ?
"null" : ip.getGeolocation().getCountryName()),

                Map.entry("associatedVpn",
ip.getAssociatedVpn()==null ? "null" :
ip.getAssociatedVpn().getName()),
                Map.entry("vpnScore", ip.getAssociatedVpn()==null ?
"null" : ip.getAssociatedVpn().getConfidenceScore()),

                Map.entry("hostOs", ip.getHostOs()==null ? "null" :
ip.getHostOs())
        ));

        String appendedQuery = " ";


        appendedQuery+=params.get("city")=="null"? "" : "MERGE
(city:City {name: $city}) " +
                "MERGE (ip)-[:IS_LOCATED_IN]->(city) ";
        appendedQuery+=params.get("country")=="null"? "" : "MERGE
(country:Country{name: $country}) " +
                "MERGE (city)-[:IS_LOCATED_IN]->(country) ";
        appendedQuery+=params.get("hostname")=="null"? "" : "MERGE
(host: Host {hostname: $hostname}) " +
                "MERGE (ip)-[:BELONGS_TO]->(host) ";
        appendedQuery+=params.get("dnsDomain")=="null"? "" : "MERGE
(domain: Domain {domain_name: $dnsDomain}) " +
                "MERGE (host)-[:IS_MEMBER_OF_DOMAIN]->(domain) ";
        appendedQuery+=params.get("associatedVpn")=="null"? "" :
"MERGE (vpn: VPN {name: $associatedVpn, confidence_score: $vpnScore})
" +
```

```
                "MERGE (ip)-[:IS_OWNED_OR_LEASED_BY]->(vpn) ";
        appendedQuery+=params.get("hostOs")=="null"? "" : "MERGE
(hostOs: OS {title: $hostOs}) " +
                "MERGE (host)-[:USES]->(hostOs) ";

        int portSize = ip.getUsedPorts().size();

        boolean hasMoreThan100Ports = portSize>100;

        if (hasMoreThan100Ports){
            result = driver.executableQuery("""
                MERGE (ip:IP {address: $ipAddress, is_Bot: $isBot,
is_Local: $isLocal, is_known_attacker: $isKnownAttacker, is_spam:
$isSpam })

"""+appendedQuery).withConfig(QueryConfig.builder().withDatabase("neo
4j").build()).
                    withParameters(params)
                    .execute();
            var summary=result.summary();
            System.out.println(summary.counters().containsUpdates());
            System.out.println(summary.toString());

            ArrayList<String> appendedQueries=new ArrayList<>();

            for (int i=0; i<ip.getUsedPorts().size();i++){
                String currPort = ip.getUsedPorts().get(i);
                params.put("port"+i,currPort);
                appendedQueries.add("MERGE (port"+i+":
PORT{port_Number: '"+currPort+"'})" + "MERGE (ip)-[:USES]-
>(port"+i+") ");
            }

            while(!appendedQueries.isEmpty()){

                String query = "MATCH (ip:IP {address: '" + currIp +
"'}) ";
                for (int i=0; i<100; i++){
                    query=query.concat(appendedQueries.get(0));
                    appendedQueries.remove(0);
```

```
                    if (appendedQueries.isEmpty()){break;}
                }
                result = driver.executableQuery(query)

.withConfig(QueryConfig.builder().withDatabase("neo4j").build()).
                        withParameters(params)
                        .execute();
                summary=result.summary();

System.out.println(summary.counters().containsUpdates());
                System.out.println(summary.toString());
            }

        }else{
            for (int i=0; i<ip.getUsedPorts().size();i++){
                String currPort = ip.getUsedPorts().get(i);
                params.put("port"+i,currPort);
                appendedQuery+="MERGE (port"+i+": PORT{port_Number:
'"+currPort+"'})" + "MERGE (ip)-[:USES_PORT]->(port"+i+") ";
            }
            result = driver.executableQuery("""
                MERGE (ip:IP {address: $ipAddress, is_Bot: $isBot,
is_Local: $isLocal, is_known_attacker: $isKnownAttacker, is_spam:
$isSpam })

"""+appendedQuery).withConfig(QueryConfig.builder().withDatabase("neo
4j").build()).
                    withParameters(params)
                    .execute();
            var summary=result.summary();
            System.out.println(summary.counters().containsUpdates());
            System.out.println(summary.toString());
        }

        return result;
    }

    public EagerResult deleteNullNodes(){
        var result = driver.executableQuery(
                """
```

```
                        MATCH (city:City WHERE city.name = "null")
DETACH DELETE city
                        MATCH (country:Country WHERE country.name =
"null") DETACH DELETE country
                        MATCH (vpn:VPN WHERE vpn.name = "null")
DETACH DELETE vpn
                        MATCH (host:Host WHERE host.hostname =
"null") DETACH DELETE host
                        MATCH (os:OS WHERE os.title = "null") DETACH
DELETE os
                        MATCH (ip:IP WHERE ip.address = "null")
DETACH DELETE ip
                        MATCH (port:PORT WHERE port.port_Number =
"null") DETACH DELETE port
                        MATCH (domain:Domain WHERE domain.domain_name
= "null") DETACH DELETE domain
                        """

        ).execute();

        var summary=result.summary();
        System.out.println(summary.counters().containsUpdates());
        System.out.println(summary.toString());

        return result;

    }

    public boolean addConversation(Conversation conversation){

//        IpAddress Ip1 = conversation.getIp1();
//        IpAddress Ip2 = conversation.getIp2();
        String ip1=conversation.getIp1str();
        String ip2=conversation.getIp2str();

        if((ip1==null)||(ip2==null)){return false;}

        HashMap<String,Object> params = new HashMap<>(Map.ofEntries(
                Map.entry("sourceAddress",ip1),
                Map.entry("destinationAddress", ip2),
```

```
Map.entry("bytesSent",conversation.getLengthOfPacketsSentFromIp1()),

Map.entry("bytesReceived",conversation.getLengthOfPacketsSentFromIp2(
)),
                Map.entry("conversationId",
conversation.getConversationId())));


        String appendedQuery=" ";
        for (int i=0; i<conversation.getPortsUsed().size();i++){
            String currPort = conversation.getPortsUsed().get(i);
            params.put("port"+i,currPort);
            appendedQuery+="MERGE (port"+i+": PORT{port_Number:
'"+currPort+"'})" + "MERGE (port"+i+")-[:IS_USED_IN]->(conversation)
";
        }

        var result = driver.executableQuery("""
                MATCH (a:IP {address: $sourceAddress})
                MATCH (b:IP {address: $destinationAddress})
                MERGE (conversation:CONVERSATION{id: $conversationId,
bytes_sent:$bytesSent, bytes_received:$bytesReceived, sender:
$sourceAddress, receiver: $destinationAddress})
                MERGE (a)-[:HAS_CONVERSATION]->(conversation)
                MERGE (b)-[:HAS_CONVERSATION]->(conversation)

"""+appendedQuery).withConfig(QueryConfig.builder().withDatabase("neo
4j").build()).
                withParameters(params)
                .execute();

        var summary=result.summary();
        System.out.println(summary.counters().containsUpdates());
        System.out.println(summary.toString());

        return true;


    }
```

```
}
```


9.2 Απόκριση IP Geolocation API


```
{
    "ip": "1.0.155.152",
    "hostname": "node-5g8.pool-1-0.dynamic.nt-isp.net",
    "location": {
        "continent_code": "AS",
        "continent_name": "Asia",
        "country_code2": "TH",
        "country_code3": "THA",
        "country_name": "Thailand",
        "country_name_official": "Kingdom of Thailand",
        "country_capital": "Bangkok",
        "state_prov": "Bangkok",
        "state_code": "TH-10",
        "district": "Sai Mai",
        "city": "Bangkok",
        "zipcode": "10210",
        "latitude": "13.88807",
        "longitude": "100.57449",
        "is_eu": false,
        "country_flag":
"https://ipgeolocation.io/static/flags/th_64.png",
        "geoname_id": "10120526",
        "country_emoji": "TH"
    },
    "country_metadata": {
        "calling_code": "+66",
        "tld": ".th",
        "languages": [
            "th",
            "en"
        ]
    },
    "network": {
        "connection_type": "Broadband",
```

```
        "route": "1.0.144.0/20",
        "is_anycast": false
    },
    "currency": {
        "code": "THB",
        "name": "Baht",
        "symbol": "฿"
    },
    "asn": {
        "as_number": "AS23969",
        "organization": "TOT Public Company Limited",
        "country": "TH",
        "type": "BUSINESS",
        "domain": "ntplc.co.th",
        "date_allocated": "",
        "rir": "APNIC"
    },
    "company": {
        "name": "TOT Public Company Limited",
        "type": "ISP",
        "domain": "tot.co.th"
    },
    "security": {
        "threat_score": 0,
        "is_tor": false,
        "is_proxy": false,
        "proxy_provider_names": [
        ],
        "proxy_confidence_score": 0,
        "proxy_last_seen": "",
        "is_residential_proxy": false,
        "is_vpn": false,
        "vpn_provider_names": [
        ],
        "vpn_confidence_score": 0,
        "vpn_last_seen": "",
        "is_relay": false,
        "relay_provider_name": "",
        "is_anonymous": false,
        "is_known_attacker": false,
```

```
        "is_bot": false,
        "is_spam": false,
        "is_cloud_provider": false,
        "cloud_provider_name": ""
    },
    "time_zone": {
        "name": "Asia/Bangkok",
        "offset": 7,
        "offset_with_dst": 7,
        "current_time": "2026-05-12 12:23:08.390+0700",
        "current_time_unix": 1778563388.39,
        "current_tz_abbreviation": "ICT",
        "current_tz_full_name": "Indochina Time",
        "standard_tz_abbreviation": "ICT",
        "standard_tz_full_name": "Indochina Time",
        "is_dst": false,
        "dst_savings": 0,
        "dst_exists": false,
        "dst_tz_abbreviation": "",
        "dst_tz_full_name": "",
        "dst_start": {
        },
        "dst_end": {
        }
    },
    "user_agent": {
        "user_agent_string": "okhttp/5.3.2",
        "name": "Okhttp",
        "type": "Robot",
        "version": "5.3.2",
        "version_major": "5",
        "device": {
            "name": "Square Okhttp",
            "type": "Robot",
            "brand": "Square",
            "cpu": "Unknown"
        },
        "engine": {
            "name": "okhttp",
            "type": "Robot",
```

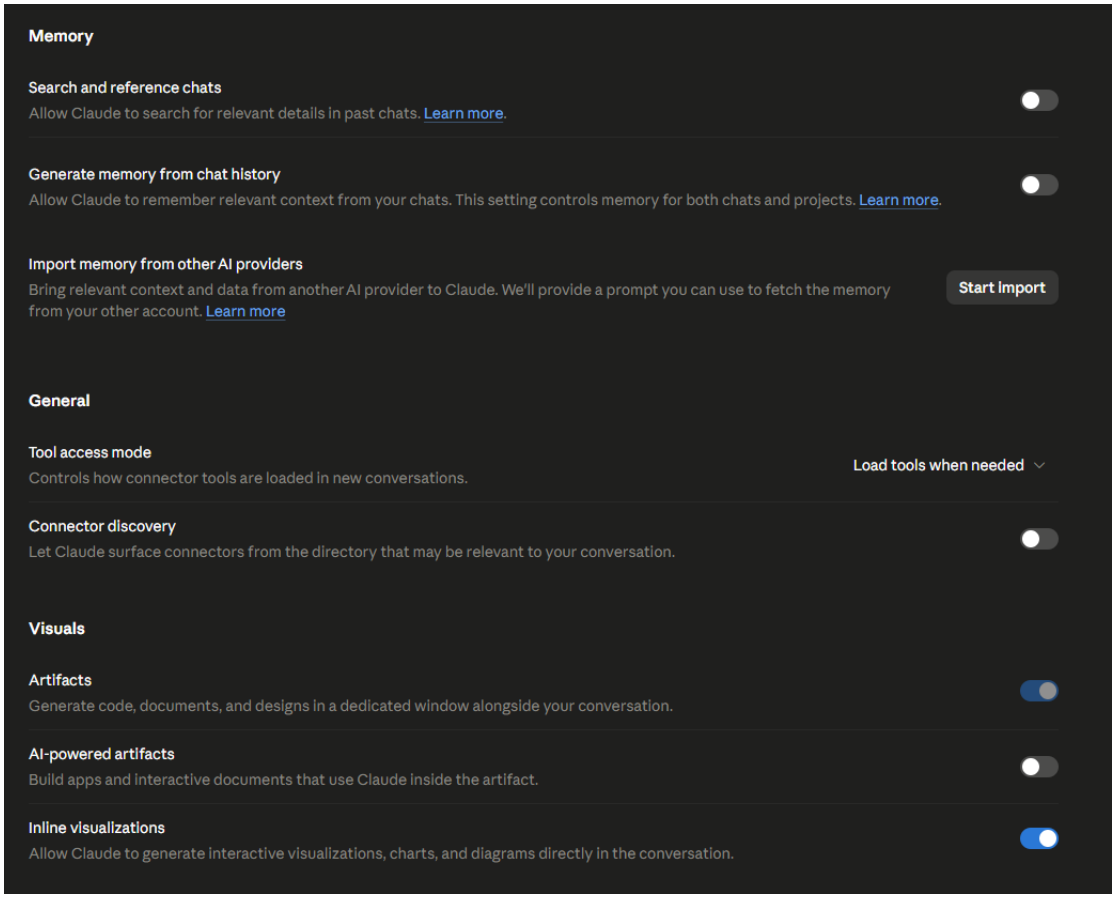
```
            "version": "5.3.2",
            "version_major": "5"
        },
        "operating_system": {
            "name": "Cloud",
            "type": "Cloud",
            "version": "??",
            "version_major": "??",
            "build": "??"
        }
    }
}
```


9.3 Αποτελέσματα-Αποκρίσεις LLM

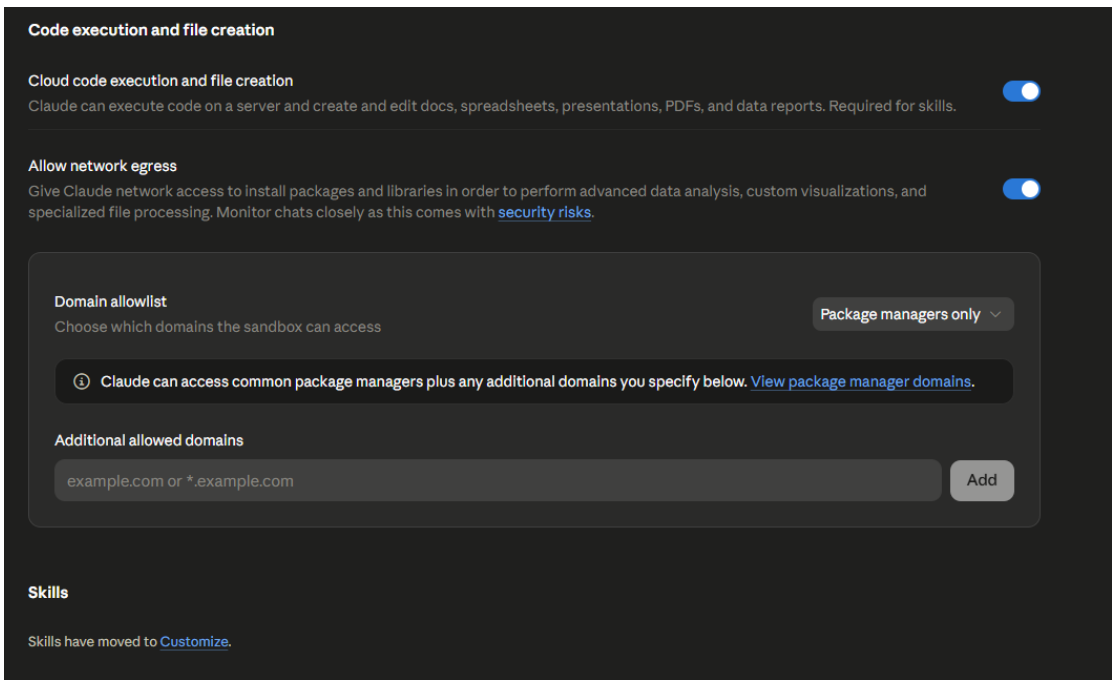
Λόγω έκτασης, παρατίθενται σε επισυναπτόμενη τεχνική αναφορά.

9.4 Διαμόρφωση Claude

Παρατίθενται Screenshots από τις ρυθμίσεις που χρησιμοποιήθηκαν:



Εικόνα 19: Claude Desktop Settings (1)



Εικόνα 20: Claude Desktop Settings (2)

9.5 Οντολογία

Η οντολογία που ορίζουμε στην 3.1 δίνεται αναλυτικά παρακάτω:

```
<?xml version="1.0"?>
<Ontology xmlns="http://www.w3.org/2002/07/owl#"
     xml:base="http://www.semanticweb.org/petros-
petalas/ontologies/2026/5/pcap-to-kg-ontology/0.1.0"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:xml="http://www.w3.org/XML/1998/namespace"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     ontologyIRI="http://www.semanticweb.org/petros-
petalas/ontologies/2026/5/pcap-to-kg-ontology/0.1.0">
    <Prefix name="" IRI="http://www.semanticweb.org/petros-
petalas/ontologies/2026/5/pcap-to-kg-ontology/0.1.0/"/>
    <Prefix name="owl" IRI="http://www.w3.org/2002/07/owl#"/>
    <Prefix name="rdf" IRI="http://www.w3.org/1999/02/22-rdf-syntax-
ns#"/>
    <Prefix name="xml" IRI="http://www.w3.org/XML/1998/namespace"/>
    <Prefix name="xsd" IRI="http://www.w3.org/2001/XMLSchema#"/>
    <Prefix name="rdfs" IRI="http://www.w3.org/2000/01/rdf-schema#"/>
    <Declaration>
        <Class IRI="#City"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Conversation"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Country"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Domain"/>
    </Declaration>
    <Declaration>
        <Class IRI="#External_IP"/>
    </Declaration>
    <Declaration>
        <Class IRI="#Host"/>
    </Declaration>
```

```
<Declaration>
    <Class IRI="#IP"/>
</Declaration>
<Declaration>
    <Class IRI="#Local_IP"/>
</Declaration>
<Declaration>
    <Class IRI="#OS"/>
</Declaration>
<Declaration>
    <Class IRI="#Port"/>
</Declaration>
<Declaration>
    <Class IRI="#VPN"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="#belongs_to"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="#has_conversation"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="#is_located_in"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="#is_member_of"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="#is_owned_or_leased_by"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="#is_used_in"/>
</Declaration>
<Declaration>
    <ObjectProperty IRI="#uses"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#address"/>
</Declaration>
<Declaration>
```

```
    <DataProperty IRI="#bytes_received"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#bytes_sent"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#confidence_score"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#domain_name"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#hostname"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#id"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#is_bot"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#is_known_attacker"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#is_local"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#is_spam"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#name"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#port_number"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#receiver"/>
</Declaration>
<Declaration>
    <DataProperty IRI="#sender"/>
```

```
</Declaration>
<SubClassOf>
    <Class IRI="#External_IP"/>
    <Class IRI="#IP"/>
</SubClassOf>
<SubClassOf>
    <Class IRI="#Local_IP"/>
    <Class IRI="#IP"/>
</SubClassOf>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#belongs_to"/>
    <Class IRI="#IP"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#has_conversation"/>
    <Class IRI="#IP"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#is_located_in"/>
    <Class IRI="#City"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#is_located_in"/>
    <Class IRI="#IP"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#is_member_of"/>
    <Class IRI="#Host"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#is_owned_or_leased_by"/>
    <Class IRI="#IP"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#is_used_in"/>
    <Class IRI="#Port"/>
</ObjectPropertyDomain>
<ObjectPropertyDomain>
    <ObjectProperty IRI="#uses"/>
    <Class IRI="#Host"/>
```

```
</ObjectPropertyDomain>
<ObjectPropertyRange>
    <ObjectProperty IRI="#belongs_to"/>
    <Class IRI="#Host"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#has_conversation"/>
    <Class IRI="#Conversation"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#is_located_in"/>
    <Class IRI="#City"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#is_located_in"/>
    <Class IRI="#Country"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#is_member_of"/>
    <Class IRI="#Domain"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#is_owned_or_leased_by"/>
    <Class IRI="#VPN"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#is_used_in"/>
    <Class IRI="#Conversation"/>
</ObjectPropertyRange>
<ObjectPropertyRange>
    <ObjectProperty IRI="#uses"/>
    <Class IRI="#OS"/>
</ObjectPropertyRange>
<DataPropertyDomain>
    <DataProperty IRI="#address"/>
    <Class IRI="#IP"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#bytes_received"/>
    <Class IRI="#Conversation"/>
```

```
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#bytes_sent"/>
    <Class IRI="#Conversation"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#confidence_score"/>
    <Class IRI="#VPN"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#domain_name"/>
    <Class IRI="#Domain"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#hostname"/>
    <Class IRI="#Host"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#id"/>
    <Class IRI="#Conversation"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#is_bot"/>
    <Class IRI="#IP"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#is_known_attacker"/>
    <Class IRI="#IP"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#is_local"/>
    <Class IRI="#IP"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#is_spam"/>
    <Class IRI="#IP"/>
</DataPropertyDomain>
<DataPropertyDomain>
    <DataProperty IRI="#name"/>
    <Class IRI="#City"/>
```

```
    </DataPropertyDomain>
    <DataPropertyDomain>
        <DataProperty IRI="#name"/>
        <Class IRI="#Country"/>
    </DataPropertyDomain>
    <DataPropertyDomain>
        <DataProperty IRI="#name"/>
        <Class IRI="#VPN"/>
    </DataPropertyDomain>
    <DataPropertyDomain>
        <DataProperty IRI="#port_number"/>
        <Class IRI="#Port"/>
    </DataPropertyDomain>
    <DataPropertyDomain>
        <DataProperty IRI="#receiver"/>
        <Class IRI="#Conversation"/>
    </DataPropertyDomain>
    <DataPropertyDomain>
        <DataProperty IRI="#sender"/>
        <Class IRI="#Conversation"/>
    </DataPropertyDomain>
    <DataPropertyDomain>
        <DataProperty abbreviatedIRI="owl:topDataProperty"/>
        <Class IRI="#Host"/>
    </DataPropertyDomain>
    <DataPropertyDomain>
        <DataProperty abbreviatedIRI="owl:topDataProperty"/>
        <Class IRI="#IP"/>
    </DataPropertyDomain>
</Ontology>
```

Βιβλιογραφία

- IBM. (2025).What is vibe coding? https://www.ibm.com/think/topics/vibe-coding
- Kutub Thakur; Md Liakat Ali; Ning Jiang; Meikang Qiu. (2016). Impact of Cyber-Attacks on Critical Infrastructure. https://ieeexplore.ieee.org/abstract/document/7502286
- MITRE. (2025). ATT&CK https://attack.mitre.org/
- Iannacone Michael, Bohn Shawn, Nakamura Grant, Gerth John, Huffer Kelly, Bridges Robert, Ferragut Erik, Goodall John. (2015). Developing an Ontology for Cyber Security Knowledge Graphs https://www.osti.gov/servlets/purl/1424501
- Zareen Syed, Ankur Padia, Tim Finin, Lisa Mathews and Anupam Joshi. (2015). UCO: A Unified Cybersecurity Ontology UCO: A Unified Cybersecurity Ontology
- Natalya F. Noy and Deborah L. McGuinness. (2001). Ontology Development 101: A Guide to Creating Your First Ontology.
- AIDAN HOGAN, IMFD, DCC, EVA BLOMQVIST, MICHAEL COCHEZ, CLAUDIA D'AMATO,GERARD DE MELO, CLAUDIO GUTIERREZ,JOSÉ EMILIO LABRA GAYO, SABRINA KIRRANE, SEBASTIAN NEUMAIER, AXEL POLLERES, ROBERTO NAVIGLI, AXEL-CYRILLE NGONGA NGOMO, DICE, SABBIR M. RASHID,ANISA RULA, LUKAS SCHMELZEISEN, JUAN SEQUEDA, STEFFEN STAAB, ANTOINE ZIMMERMANN. (2021). Knowledge Graphs https://arxiv.org/pdf/2003.02320
- GeeksForGeeks. (2025). Services and Segment Structure in TCP. https://www.geeksforgeeks.org/computer-networks/services-and-segment-structure-in-tcp/

- GeeksForGeeks.(2025). What is OSI Model? - Layers of OSI Model. https://www.geeksforgeeks.org/computer-networks/open-systems-interconnection-model-osi/
- E. Kline, Loon LLC, K. Duleba, Google, Z. Szamonek, S. Moser, Google Switzerland GmbH, W. Kumari. (August 2020) A Format for Self-Published IP Geolocation Feeds https://datatracker.ietf.org/doc/rfc8805/
- Stratosphere. (2020). Stratosphere Laboratory Datasets. Retrieved March 13, 2020, from https://www.stratosphereips.org/datasets-overview
- Model Context Protocol. (2026) https://modelcontextprotocol.io/docs/getting-started/intro
- Hasan Abu-Rasheed, Christian Weber, Madjid Fathi. (2024). Knowledge Graphs as Context Sources for LLM-Based Explanations of Learning Recommendations https://arxiv.org/abs/2403.03008
- Berk Ati , Sarp Aykent, Alexa Chittams, Lisheng Fu, Rebecca J. Passonneau, Evan Radcliffe, Guru Rajan Rajagopal, Adam Sloan, Tomasz Tudrej, Ferhan Ture, Zhe Wu, Lixinyu Xu2, Breck Baldwin2Penn State University, Comcast AI Technologies (2024). Non-Determinism of "Deterministic" LLM Settings https://arxiv.org/html/2408.04667v5#bib
- Thebestvpn. (2026)What Percentage of People use a VPN? https://thebestvpn.com/statistics/what-percentage-of-people-use-a-vpn/
- Reethika Ramesh, Anjali Vyas, Roya Ensafi. (2022). "All of them claim to be the best": Multi-perspective study of VPNusers and VPN providers https://arxiv.org/pdf/2208.03505
- Drishti Shah (2025).Benefits of using MCP over traditional integration methods https://portkey.ai/blog/benefits-of-mcp-over-traditional-integration/
- Sysdig (2026). What is a PCAP file? https://www.sysdig.com/learn-cloud-native/what-is-a-pcap-file#:~:text=A%20PCAP%20(Packet%20Capture)%20file,issues%2C%20and%20detect%20security%20threats.
- Wireshark (2025). Introduction https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html#ChIntroWhatIs

- Java (2025). What is Java https://www.java.com/en/download/help/whatis_java.html
- Apache (2026). What is Maven https://maven.apache.org/what-is-maven.html
- Pcap4J (2022). Github Repository of Pcap4j https://github.com/kaitoy/pcap4j
- Square(2026) OkHttp https://square.github.io/okhttp/
- PostgreSQL(2026). https://www.postgresql.org/
- Neo4j(2026). What is Neo4j https://neo4j.com/docs/getting-started/whats-neo4j/
- Claude (2026)Overview https://code.claude.com/docs/en/overview
- MCP-Neo4j(2026)mcp-neo4j/servers/mcp-neo4j-cypher at main · neo4j-contrib/mcp-neo4j
- Arthur Drichel, Benedikt Holmes, Justus von Brandt, Ulrike Meyer. (2021) The More, the Better?AStudy on Collaborative Machine Learning for DGA Detection https://arxiv.org/pdf/2109.11830