Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Μεταπτυχιακή Διατριβή
Στην Ασφάλεια Υπολογιστών και Δικτύων

Υλοποίηση Πρωτοκόλλων Διαμοιρασμού Πληροφορίας που Σχετίζονται με Απειλές του Κυβερνοχώρου

Ζησιάδης Γεώργιος

Επιβλέπων Καθηγητής
Μαυρίδης Ιωάννης

Μάιος 2017

Ανοικτό Πανεπιστήμιο Κύπρου

Σχολή Θετικών και Εφαρμοσμένων Επιστημών

Υλοποίηση Πρωτοκόλλων Διαμοιρασμού Πληροφορίας που Σχετίζονται με Απειλές του Κυβερνοχώρου

Ζησιάδης Γεώργιος

Επιβλέπων Καθηγητής
Μαυρίδης Ιωάννης

Η παρούσα μεταπτυχιακή διατριβή υποβλήθηκε
προς μερική εκπλήρωση των απαιτήσεων για απόκτηση

μεταπτυχιακού τίτλου σπουδών
στην Ασφάλεια Υπολογιστών και Δικτύων

από τη Σχολή Θετικών και Εφαρμοσμένων Επιστημών
του Ανοικτού Πανεπιστημίου Κύπρου

Μάιος 2017

Περίληψη

Στόχος της συγκεκριμένης μεταπτυχιακής διατριβής, είναι η δημιουργία μιας εφαρμογής λογισμικού για την ανταλλαγή πληροφοριών ασφαλείας σχετικών με απειλές κυβερνοχώρου. Για την υλοποίηση χρησιμοποιήθηκε το σύστημα ανάλυσης κακόβουλου λογισμικού Cuckoo Sandbox, καθώς και το πρότυπο ανταλλαγής μηνυμάτων TAXII.

Στα κεφάλαια που ακολουθούν παρουσιάζονται οι υπηρεσίες και οι μέθοδοι που υποστηρίζονται από το TAXII, καθώς και η μεθοδολογία που ακολουθήθηκε για την υλοποίηση της εφαρμογής ανταλλαγής μηνυμάτων. Ακόμη παρουσιάζεται το σύστημα ανάλυσης κακόβουλου λογισμικού Cuckoo Sandbox με αναφορά στις δυνατότητες του, στον τρόπο λειτουργίας του καθώς και στον τύπο των αναφορών που εξάγονται από την εκτέλεση του. Τέλος παρουσιάζεται μια ανασκόπηση της υλοποίησης που ακολουθήθηκε, όπου επισημαίνονται προβλήματα που εντοπίστηκαν και προτείνονται πιθανές λύσεις για το μέλλον.

Summary

The goal of this M.Sc. dissertation is the creation of a software application, for the exchange of security information related to cyber threats. For the implementation Cuckoo Sandbox malware analysis system and TAXII message exchanging standard were used.

The following chapters illustrate the services and methods supported by TAXII, as well as the methodology used to implement the message exchanging application. Also is presented the Cuckoo Sandbox malware analysis system, with reference to its capabilities, its mode of operation and the type of reports extracted from its execution. Finally, there is a review of the implementation that followed, highlighting identified problems and suggesting possible solutions for the future.

Ευχαριστίες

Θα ήθελα να εκφράσω τις θερμότατες ευχαριστίες μου στον επιβλέποντα καθηγητή κ. Ιωάννη Μαυρίδη όπως και στον κ. Γιώργο Σακελλαρίου, καθώς χωρίς την πολύτιμη και συνεχή βοήθεια τους, η εκπόνηση της παρούσας διπλωματικής διατριβής δεν θα ήταν δυνατή.

Επίσης θα ήθελα να πω ένα πολύ μεγάλο ευχαριστώ στην οικογένεια μου και την σύντροφο μου, για όλο αυτό τον καιρό που με στήριξαν αλλά και με ανεχτήκαν κάποιες φορές, σε αυτό το δύσκολο αλλά ταυτόχρονα μαγικό ταξίδι.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

Οι κυβερνοεπιθέσεις αποτελούν μια από τις μεγαλύτερες μάστιγες της εποχής. Οι οργανισμοί συνήθως κάνουν χρήση διαφόρων μέτρων και πολιτικών αφού πέσουν θύματα κάποιας επίθεσης. Επομένως προκύπτει ότι θα πρέπει η σωστή αντιμετώπιση των κυβερνοεπιθέσεων να αφορά την πρόληψη. Σκοπός είναι η αναγνώριση των αδυναμιών, πριν την εκμετάλλευση αυτών από κάποιον κακόβουλο. Η συλλογή και η χρήση πληροφοριών σχετικών με απειλές που μπορεί να εξελιχθούν σε κυβερνοεπιθέσεις, αποτελεί τον καλύτερο τρόπο προστασίας για κάποιον οργανισμό έναντι των αντιπάλων του. Η συλλογή και η ανάλυση πληροφοριών σχετικών με κυβερνοαπειλές είναι σήμερα γνωστή με τον όρο cyber intelligence (CTI)(SANS 2015).

Για παράδειγμα το θέμα ή το περιεχόμενο κάποιου phishing email, οι κακόβουλοι σύνδεσμοι που μπορεί να εμπεριέχονται σε αυτό, όπως διευθύνσεις IP εχθρικών εξυπηρετητών (command and control servers), είναι μια τυπική διαδικασία CTI. Με τη χρήση των παραπάνω πληροφοριών οι αναλυτές κυβερνοεπιθέσεων μπορούν να συνδέσουν συγκεκριμένες δραστηριότητες με συγκεκριμένους φορείς απειλών (threat actors), έτσι ώστε να εφαρμόσουν στρατηγικές για τον μετριασμό ή την εξουδετέρωση της δράσης τους αλλά και την πρόβλεψη μελλοντικών επιθέσεων.

Με βάση τα παραπάνω γίνεται ξεκάθαρη η σημασία της ανταλλαγής μεταξύ των διαφόρων οργανισμών, πληροφοριών σχετικών με απειλές του κυβερνοχώρου, με σκοπό την έγκυρη ενημέρωση και προετοιμασία για αντιμετώπιση των κυβερνοαπειλών. Ωστόσο, η διαδικασία ανταλλαγής πληροφοριών είναι αρκετές φορές χρονοβόρα και ανακριβής ή περιλαμβάνει πληροφορίες που δεν γίνονται άμεσα επεξεργάσιμες από υπολογιστικές διατάξεις χωρίς την μεσολάβηση του ανθρώπινου παράγοντα. Γεγονός που εισάγει περαιτέρω καθυστέρηση στην επεξεργασία και περιορισμό της σαφήνειας και της ακρίβειας των πληροφοριών.

Κεφάλαιο 2

Πρότυπο Διαμοιρασμού Πληροφορίας TAXII

Ο διαμοιρασμός πληροφορίας (Information Sharing) αναφέρεται στην ανταλλαγή δεδομένων ανάμεσα σε οργανισμούς**.** Ο διαμοιρασμός πληροφοριών σχετικών με ψηφιακές απειλές, είναι πάρα πολύ σημαντικός στην διαδικασία αντιμετώπισής τους. Σε ένα ρεαλιστικό περιβάλλον κανένας οργανισμός δεν θα μπορούσε να αντιμετωπίσει τους κινδύνους που μπορούν να προκύψουν παραμένοντας απομονωμένος. Ο εντοπισμός μιας απειλής για κάποιον οργανισμό μπορεί να είναι η πρόληψη για κάποιον άλλον. Ο διαμοιρασμός πληροφοριών σχετικών με ψηφιακές απειλές μπορεί να αποτελέσει τροχοπέδη στους επιτιθέμενους καθώς διαφορές τεχνικές επιθέσεων μπορούν να αποκαλυφθούν με συνέπεια να μην είναι το ίδιο αποτελεσματικές. Ωστόσο η διαδικασία διαμοιρασμού πληροφοριών είναι ιδιαίτερα χρονοβόρα και πολλές φορές συναντάει προβλήματα στις διαφορετικές τεχνικές ενημέρωσης που χρησιμοποιεί ο κάθε οργανισμός. Αυτά τα προβλήματα προσπαθεί να αντιμετωπίσει το πρότυπο TAXII, δίνοντας την δυνατότητα σε χρήστες να ανταλλάσουν πληροφορίες σχετικές με ψηφιακές απειλές μέσω ενός αυτοματοποιημένου τρόπου.

Το πρότυπο TAXII (Trusted Automated Exchange of Indicator Information) ορίζει ένα σύνολο υπηρεσιών και μηνυμάτων για την ανταλλαγή πληροφοριών ανάμεσα σε οργανισμούς, σχετικές με απειλές του κυβερνοχώρου. Θα πρέπει να γίνει σαφές ότι σκοπός του TAXII δεν είναι η σύναψη σχέσεων εμπιστοσύνης ανάμεσα στους οργανισμούς αλλά η ενημέρωση των οργανισμών για τις απειλές, καθώς και ο εύκολος διαμοιρασμός τους μεταξύ των υπολογιστικών συστημάτων τους(Connolly et al. 2014).

Οι κύριοι στόχοι του TAXII είναι οι εξής:

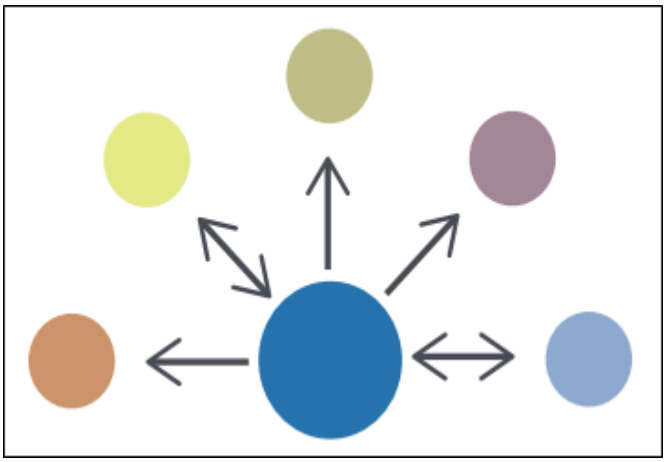
- Ο γρήγορος και ασφαλής διαμοιρασμός πληροφοριών ψηφιακών απειλών.
- Η ανταλλαγή μεγάλου όγκου πληροφοριών.
- Η υποστήριξη μεγάλου εύρους περιπτώσεων χρήσης και τεχνικών σχετικών με διαμοιρασμό πληροφοριών ψηφιακών απειλών.
- Η υποστήριξη των υφιστάμενων μηχανισμών διαμοιρασμού, με στόχο την ελαχιστοποίηση των αλλαγών.
- Η μακροπρόθεσμη έγκριση από τους οργανισμούς τυποποίησης.

2.1 Μοντέλα Διαμοιρασμού

Το πρότυπο TAXII υποστηρίζει τα ακόλουθα μοντέλα διαμοιρασμού πληροφοριών:

HUB - SPOKE

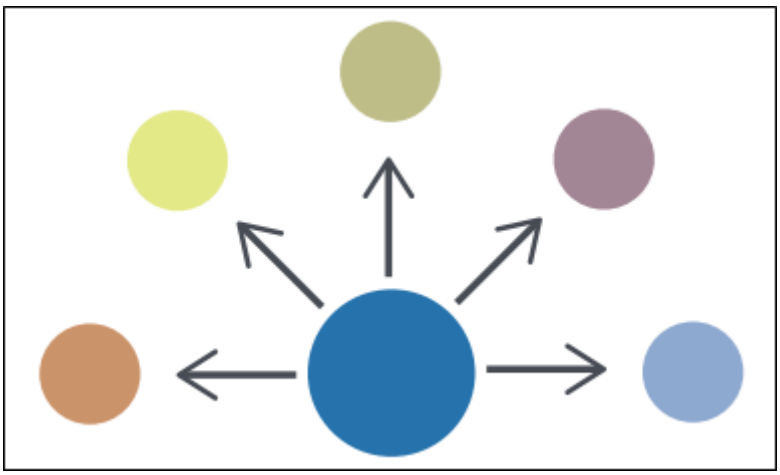
Σε αυτό το μοντέλο ένας οργανισμός συμπεριφέρεται ως διαμοιραστής (hub) των πληροφοριών προς όλους τους υπόλοιπους οργανισμούς. Κάθε οργανισμός (spoke) μοιράζεται πληροφορίες με τον διαμοιραστή οργανισμό και αυτός με την σειρά του προωθεί τις πληροφορίες στους υπόλοιπους οργανισμούς. Ο διαμοιραστής μπορεί να επιλέξει εάν θα προωθήσει τις πληροφορίες αυτούσιες ή εάν θα εφαρμόσει διάφορα φίλτρα πριν τον διαμοιρασμό τους. Η ροή πληροφοριών γίνεται αμφίδρομα (και προς τις δυο κατευθύνσεις).



Εικόνα 2.1:Διάγραμμα επικοινωνίας hub - spoke(Connolly et al. 2014)

SOURCE - SUBSCRIBER

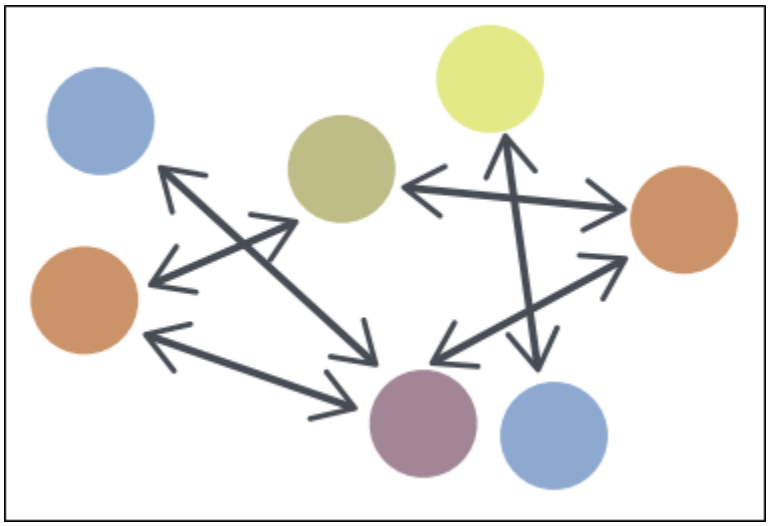
Σε αυτό το μοντέλο ένας οργανισμός συμπεριφέρεται σαν πηγή πληροφοριών για όλους τους υπόλοιπους οργανισμούς. Η ροή της πληροφορίας επιτρέπεται μόνο από τον οργανισμό πηγή (source) προς τους εγγεγραμένους οργανισμούς (subscriber).



Εικόνα 2.2:Διάγραμμα επικοινωνίας Source - Subscriber(Connolly et al. 2014)

PEER TO PEER

Σε αυτό το μοντέλο, κάθε οργανισμός μπορεί να είναι αποστολέας ή παραλήπτης πληροφοριών. Όπως γίνεται σαφές επιτρέπεται να υπάρχει ροη πληροφοριών από όλους προς όλους.



Εικόνα 2.3:Διάγραμμα επικοινωνίας PEER TO PEER(Connolly et al. 2014)

2.2 Ορολογία

Οι παρακάτω όροι χρησιμοποιούνται για τον ορισμό των κεντρικών εννοιών του TAXII, των υπηρεσιών καθώς των ρόλων που μπορούν να έχουν οι εμπλεκόμενοι κατά την επικοινωνία μέσω του προτύπου TAXII:

- TAXII Data Collection είναι μια ονομαστική συλλογή πληροφοριών η οποία μπορεί να μεταφερθεί μέσω του TAXII. Κάθε συλλογή μπορεί να είναι είτε τύπου Data feed είτε τύπου Data set. TAXII Data Feed είναι μια διατεταγμένη συλλογή πληροφοριών όπου η διάταξη των εγγραφών στην συλλογή επιτυγχάνεται με την χρήση χρονοσφραγίδων. TAXII Data Set είναι μια μη διατεταγμένη συλλογή πληροφοριών.

- TAXII Message είναι ένα διακριτό σύνολο πληροφοριών το οποίο μεταφέρεται από μια οντότητα σε μια άλλη μέσω του δικτύου. Μια ακολουθία μηνυμάτων ανάμεσα σε 2 πλευρές (συνήθως με την μορφή αίτησης-απάντησης), ονομάζεται TAXII Message Exchange.

- TAXII Service είναι μια λειτουργία οι οποία επιτυγχάνεται μέσω της ανταλλαγής μηνυμάτων TAXII, ενώ TAXII Capability είναι μια δραστηριότητα υψηλού επιπέδου η οποία επιτυγχάνεται μέσω της χρήσης των υπηρεσιών TAXII.

Στο πρότυπο TAXII οι εμπλεκόμενες οντότητες (οργανισμοί) μπορούν κάθε στιγμή να έχουν έναν από τους ακόλουθους ρόλους :

- Παραγωγός (Producer): Χαρακτηρίζει την οντότητα που είναι η πηγή της πληροφορίας.

- Καταναλωτής (Consumer): Χαρακτηρίζει την οντότητα που είναι ο παραλήπτης της πληροφορίας.

Με τον όρο *οντότητα αναφερόμαστε σε οτιδήποτε μπορεί να υπάρχει σαν διακριτή μονάδα.* Επίσης θα πρέπει να σημειωθεί ότι οι οντότητες δεν έχουν αποκλειστικά κάποιον ρόλο. Μια οντότητα μπορεί να είναι παραγωγός κάποιας πληροφορίας και ταυτόχρονα καταναλωτής κάποιας άλλης.

Οι ακόλουθοι όροι χρησιμοποιούνται κατά την υλοποίηση ενός μοντέλου TAXII Client Server:

- TAXII Implementation είναι η υλοποίηση μιας αρχιτεκτονικής TAXII.

- TAXII Daemon είναι το μέρος μιας υλοποίησης TAXII, το οποίο παρέχει μια ή περισσότερες υπηρεσίες TAXII.

- TAXII Client είναι το μέρος μιας υλοποίησης το οποίο επικοινωνεί με έναν TAXII Daemon έτσι ώστε να γίνει κάτοχος μιας υπηρεσίας TAXII.

Οι υπηρεσίες TAXII αντιπροσωπεύουνένα σύνολο απαραίτητων μηχανισμών για την υποστήριξη των δυνατοτήτων του TAXII. Οι υπηρεσίες που υποστηρίζονται από το πρότυπο TAXII είναι οι ακόλουθες:

- Discovery Service: Είναι ο μηχανισμός που ενημερώνει τον αιτούντα για τις διαθέσιμες υπηρεσίες TAXII, καθώς και για το πώς μπορεί να τις χρησιμοποιήσει. Για παράδειγμα ενημερώνει τον αιτούντα με την διεύθυνση του TAXII Daemon που υλοποιεί την υπηρεσία για την οποία ενδιαφέρεται. Μια υπηρεσία ανακάλυψης(Discovery Service) μπορεί να ενημερώσει για υπηρεσίες που βρίσκονται σε διαφορετικούς παρόχους. Ο κάτοχος της υπηρεσίας ανακάλυψης είναι αυτός που ορίζει το εύρος των διαθέσιμων υπηρεσιών. Επίσης δεν είναι απαραίτητο κάθε φορά η υπηρεσία ανακάλυψης να γνωστοποιεί όλες τις διαθέσιμες υπηρεσίες σε όλους τους αιτούντες υπηρεσιών TAXII. Η ταυτότητα του αιτούντα και κατά συνέπεια τα δικαιώματα του είναι αυτά που καθορίζουν το εύρος των υπηρεσιών TAXII στις οποίες θα έχει πρόσβαση.

- Collection Management Service: Η υπηρεσία διαχείρισης συλλογών είναι ο μηχανισμός μέσω του οποίου κάποιος καταναλωτής μπορεί να αποκτήσει πρόσβαση σε συλλογές δεδομένων του TAXII, να αιτηθεί την εγγραφή του σε κάποια συγκεκριμένη συλλογή δεδομένων, ή να ζητήσει την διαγραφή του από κάποια συλλογή. Θα πρέπει να τονιστεί ότι η υπηρεσία διαχείρισης συλλογών δεν διανέμει το περιεχόμενο των συλλογών στους καταναλωτές. Για την διανομή του περιεχομένου απαιτείται η υλοποίηση της υπηρεσίας εισερχομένων(Inbox Service). Οι εγγραφές μπορεί να περιέχουν διάφορα ερωτήματα (queries) μέσω των οποίων φιλτράρονται τα διαθέσιμα δεδομένα των συλλογών.

- Inbox Service: Είναι ο μηχανισμός που εξυπηρετεί την ανταλλαγή μηνυμάτων ανάμεσα σε έναν παραγωγό και έναν καταναλωτή. Κάποιος καταναλωτής μπορεί να υλοποιήσει την συγκεκριμένη υπηρεσία για να αποκτήσει πρόσβαση στο περιεχόμενο μιας συλλογής δεδομένων TAXII.

- Poll Service: Είναι η υπηρεσία μέσω της οποία κάποιος παραγωγός δίνει άδεια σε κάποιον καταναλωτής να "τραβήξει" μια συλλογή δεδομένων που του ανήκει. Ο καταναλωτής επικοινωνεί με την υπηρεσία (Poll Service), ζητώντας το περιεχόμενο μιας συλλογής δεδομένων TAXII. Ο καταναλωτής μπορεί να χρησιμοποιεί ερωτήματα (queries) κατά την διάρκεια της επικοινωνίας του με την υπηρεσία, έτσι ώστε να αποκτήσει συγκεκριμένα δεδομένα από το ευρύτερο φάσμα της συλλογής.

2.3 Μηνύματα

Σε αυτή την ενότητα παρουσιάζονται οι διαφορετικοί τύποι των μηνυμάτων TAXII, καθώς και κάποια βασικά χαρακτηριστικά τους.

- TAXII Status Message: Χρησιμοποιείται για να δηλώσει κάποιο λάθος κατά την μετάδοση ή για να επιβεβαιώσει την σωστή μετάδοση.

- TAXII Discovery Request: Αίτηση για πληροφορίες σχετικά με τις διαθέσιμες υπηρεσίες TAXII.

- TAXII Discovery Response: Απάντηση σε μια αίτηση τύπου TAXII Discovery Request.

- TAXII Collection Information Request: Αίτηση για πληροφορίες σχετικά με τις διαθέσιμες συλλογές δεδομένων TAXII.

- TAXII Collection Information Response: Απάντηση σε μια αίτηση τύπου TAXII Collection Information Request.

- TAXII Manage Collection Subscription Request: Αίτηση για την δημιουργία εγγραφής ή την διαχείριση κάποιας υπάρχουσας.

- TAXII Manage Collection Subscription Response: Απάντηση σε ένα αίτημα τύπου TAXII Manage Collection Subscription Request . Παρουσιάζει τις εγγραφές για μια συγκεκριμένη συλλογή δεδομένων TAXII.

- TAXII Poll Request: Αίτηση για το περιεχόμενο μιας συλλογής δεδομένων TAXII.

- TAXII Poll Response: Απάντηση σε ένα αίτημα τύπου TAXII Poll Request. Περιλαμβάνει το συσχετιζόμενο με την συλλογή δεδομένων TAXII περιεχόμενο.

- TAXII Inbox Message: Χρησιμοποιείται για την προώθηση περιεχομένου μιας συλλογής σε κάποιον παραλήπτη.

- TAXII Poll Fulfillment Request: Χρησιμοποιείται για να αιτηθεί τη ολοκλήρωση μιας πολυτμηματικής (multi-part) αποστολής δεδομένων ή την αποστολή δεδομένων που έχουν καθυστερήσει να παραδοθούν.

Κάθε μήνυμα TAXII περιλαμβάνει ένα πεδίο αναγνωριστικού μηνύματος. Τα αναγνωριστικά των μηνυμάτων χρησιμοποιούνται για την σύνδεση αιτήσεων με απαντήσεις στις συγκεκριμένες αιτήσεις. Για παράδειγμα εάν ένα μήνυμα Β είναι η απάντηση σε ένα μήνυμα Α, το πεδίο του Β "απάντηση σε " (in response to), θα πρέπει να περιέχει το αναγνωριστικό μηνύματος του Α.

Τα ονόματα των συλλογών δεδομένων που προέρχονται από τον ίδιο παραγωγό θα πρέπει να έχουν διαφορετικά αναγνωριστικά. Ονόματα συλλογών από διαφορετικούς παραγωγούς μπορούν να είναι ίδια. Σε περίπτωση που κάποιος καταναλωτής αιτείται συλλογές δεδομένων από διαφορετικούς παραγωγούς θα πρέπει να καταγράφει τόσο το όνομα της συλλογής, όσο και το όνομα του παραγωγού καθώς ενδέχεται το όνομα της συλλογής να είναι ίδιο για δύο παραγωγούς.

Οι εγγραφές καταναλωτών σε συλλογές δεδομένων TAXII, θα πρέπει να έχουν μοναδικά αναγνωριστικά. Εγγραφές καταναλωτή σε πάνω από μια συλλογές παραγωγού θα πρέπει επίσης να έχουν διαφορετικά αναγνωριστικά. Μετά την πραγματοποίηση της εγγραφής παραγωγός και καταναλωτής αναφέρονται στην συγκεκριμένη εγγραφή με το μοναδικό αναγνωριστικό της. Ομοίως τα αποτελέσματα από κάποια αίτηση (Poll Request) θα πρέπει να έχουν μοναδικά αναγνωριστικά.

Δεδομένα που βρίσκονται σε διατεταγμένες συλλογές θα πρέπει να έχουν μια ετικέτα χρονοσφραγίδας (timestamp labels). Ο λόγος ύπαρξης των ετικετών χρονοσφραγίδας είναι η ευκολότερη αναζήτηση των δεδομένων μέσα στην συλλογή και όχι η αναφορά στον αληθινό χρόνο αυτούσιο. Νέα δεδομένα που προστίθενται σε κάποια διατεταγμένη συλλογή δεδομένων, θα πρέπει να έχουν ετικέτα χρονοσφραγίδας μεταγενέστερη από τα ήδη υπάρχοντα δεδομένα. Κάθε χρονοσφραγίδα μπορεί να έχει 0 έως 6 δεκαδικά (xx:xx:xx.(0-6)).

Τα μηνύματα TAXII ενδέχεται να περιλαμβάνουν και ερωτήματα (queries). Το TAXII διαθέτει μια γενική φόρμα για την δημιουργία ερωτημάτων. Παρόλα αυτά δίνει την δυνατότητα σε κάθε χρήστη να δημιουργήσει την δίκια του φόρμα ερωτημάτων. Κάθε φόρμα ερωτημάτων θα πρέπει να διαθέτει ένα μοναδικό αναγνωριστικό.

Τα αναγνωριστικά της έκδοσης TAXII χρησιμοποιούνται σε κάποια μηνύματα για να δείξουν τις προδιαγραφές της συγκεκριμένης έκδοσης.

2.4 Ανταλλαγή Μηνυμάτων

Σε αυτή την ενότητα περιγράφονται οι ανταλλαγές μηνυμάτων TAXII, που απαιτούνται για την υποστήριξη των υπηρεσιών TAXII που ορίστηκαν προηγουμένως. Θα πρέπει να επισημανθεί ότι οι διαδικασίες ανταλλαγής μηνυμάτων TAXII, δεν έχουν κάποια γνώση σχετική με τα πρωτόκολλα δικτύου μέσω των οποίων μεταφέρονται. Πιθανώς να απαιτείται επιπλέον επικοινωνία ανάμεσα στις εμπλεκόμενες πλευρές πριν την ανταλλαγή μηνυμάτων TAXII.

Inbox Exchange: Ένα μήνυμα εισερχομένων (Inbox Message) αποστέλλεται από τον πελάτη TAXII στην σύνδεση εισερχομένων TAXII (inbox daemon). Εάν η σύνδεση εισερχομένων εντοπίσει κάποιο σφάλμα δεν επεξεργάζεται το μήνυμα και απαντάει με το κατάλληλο μήνυμα κατάστασης. Η αποδοχή του μηνύματος απλά σημαίνει ότι το μήνυμα είναι συμβατό με τις απαιτήσεις του TAXII και έχει αποσταλεί σωστά μέσω της σύνδεσης. Ο παραλήπτης του μηνύματος μπορεί ανά πάσα στιγμή να απορρίψει το μήνυμα χωρίς να ενημερώσει τον αποστολέα.

Pushing Content to TAXII Data Collections: Πολλές φορές ο σκοπός της προώθησης δεδομένων σε έναν παραλήπτη είναι η ενημέρωση των συλλογών δεδομένων του. Για παράδειγμα σε μια

επικοινωνία τύπου hub-spoke, προωθούνται δεδομένα σε κάποια συλλογή δεδομένων του διαμοιραστή και αυτός αυτόματα διαθέτει το περιεχόμενο των συλλογών δεδομένων στους υπόλοιπους κόμβους. Μια αυτόματη δρομολόγηση τέτοιου τύπου μπορεί να επιτευχθεί με δύο τρόπους. Στην πρώτη περίπτωση ο παραλήπτης ρυθμίζει την υπηρεσία εισερχομένων του έτσι ώστε το περιεχόμενο που λαμβάνει να προστίθεται σε συγκεκριμένες συλλογές δεδομένων. Στην δεύτερη περίπτωση ο αποστολέας πρέπει να διασαφηνίσει μέσω του μηνύματος που στέλνει σε ποία συλλογή δεδομένων επιθυμεί να προστεθεί το περιεχόμενο του μηνύματος του.

Στην προώθηση πληροφοριών σε συλλογές δεδομένων TAXII ισχύει ότι και στην ανταλλαγή εισερχομένων. Εάν η σύνδεση εισερχομένων του παραλήπτη αποστείλει μήνυμα κατάστασης διαφορετικό του "success" το περιεχόμενο του μηνύματος διαγράφεται. Σε περίπτωση που το μήνυμα προωθηθεί στον παραλήπτη δεν σημαίνει ότι θα προστεθεί σε κάποια συλλογή δεδομένων. Ο παραλήπτης είναι αυτός που παίρνει την τελική απόφαση για το αν θα κρατήσει ή θα διαγράψει το περιεχόμενο ενός μηνύματος.

Discovery exchange: Σε αυτήν την ανταλλαγή ο πελάτης TAXII στέλνει ένα αίτημα ανακάλυψης υπηρεσιών στον δαίμονα της υπηρεσίας ανακάλυψης (Discovery Daemon). Σε περίπτωση ανίχνευσης σφάλματος η σύνδεση αποστέλλει το κατάλληλο μήνυμα στον πελάτη. Σε αντίθετη περίπτωση το αίτημα προωθείται. Ο κάτοχος των υπηρεσιών αποφασίζει για την λίστα των υπηρεσιών που θα αποστείλει στον πελάτη. Κάθε πελάτης ανάλογα με τα δικαιώματα του έχει πρόσβαση σε διαφορετικό εύρος υπηρεσιών. Σε περίπτωση που το αίτημα προωθηθεί αλλά δεν επιστραφεί μια λίστα με τις διαθέσιμες υπηρεσίες στον πελάτη, θα πρέπει να αιτιολογηθεί ο λόγος της απόρριψης.

Collection Information Excahnge: Σε αυτήν την ανταλλαγή ο πελάτης αιτείται πληροφορίες σχετικά με τις διαθέσιμες συλλογές δεδομένων. Ο πελάτης στέλνει το αντίστοιχο αίτημα στην σύνδεση διαχείρισης συλλογών. Εάν η σύνδεση εντοπίσει κάποιο σφάλμα, αποστέλλει το αντίστοιχο μήνυμα. Σε διαφορετική περίπτωση το αίτημα τίθεται προς επεξεργασία. Ο πελάτης δέχεται μια απάντηση με τις διαθέσιμες συλλογές δεδομένων. Οι διαθέσιμες συλλογές δεδομένων για κάθε πελάτη διαφέρουν ανάλογα με τα δικαιώματα του πελάτη. Εάν δεν επιστραφούν πληροφορίες σχετικά με τις διαθέσιμες συλλογές δεδομένων θα πρέπει να αιτιολογείται στον πελάτη με το αντίστοιχο μήνυμα ο λόγος της απόφασης.

Subscription Management Exchange: Ο πελάτης στέλνει ένα αίτημα για την δημιουργία εγγραφής σε κάποια συλλογή δεδομένων. Εάν η σύνδεση διαχείρισης συλλογών εντοπίσει

κάποιο σφάλμα στέλνει το αντίστοιχο μήνυμα. Διαφορετικά το αίτημα προωθείται. Ο πελάτης δέχεται μια απάντηση με την επιβεβαίωση της εγγραφής του στην συλλογή δεδομένων ή με τον λόγο για τον οποίον το αίτημα του απορρίφθηκε.

Poll Exchange: Ο πελάτης αιτείται δεδομένα από την αντίστοιχη συλλογή του παραγωγού. Εάν εντοπιστεί κάποιο σφάλμα αποστέλλεται στον πελάτη το αντίστοιχο μήνυμα σφάλματος, σε διαφορετική περίπτωση αξιολογείται το αίτημα και αποστέλλεται στον πελάτη το αντίστοιχο μήνυμα απάντησης, με το περιεχόμενο που αιτήθηκε. Εάν αποφασιστεί να μην δοθεί άδεια πρόσβασης, ο πελάτης θα πρέπει να ενημερωθεί με το αντίστοιχο μήνυμα. Κάποιες φορές το περιεχόμενο προς αποστολή είναι μεγάλου μεγέθους και δεν γίνεται να σταλεί με ένα μόνο μήνυμα TAXII. Σε αυτές τις περιπτώσεις ο παραγωγός χρησιμοποιεί μια πολυτμηματική υλοποίηση (Multi-Part Poll Exchange). Το περιεχόμενο διαιρείται σε τμήματα τα οποία αριθμούνται με αρχική τιμή αρίθμησης το 1. Επίσης προστίθεται στην κάθε απάντηση μια σημαία “More” με τιμές true ή false, για να υποδείξει στον πελάτη ότι υπάρχουν επιπλέον τμήματα προς αποστολή. Μόλις ο παραλήπτης λάβει την πρώτη απάντηση αιτείται το τμήμα με την αμέσως μεγαλύτερη τιμή αρίθμησης. Όταν ο παραλήπτης δεχτεί μήνυμα και η σημαία “More” πάρει την τιμή false αντιλαμβάνεται ότι το συγκεκριμένο μήνυμα είναι το τελευταίο της ακολουθίας.

Πέραν της άμεσης απάντησης σε ένα αίτημα Poll Request, το πρότυπο TAXII παρέχει μια επιπλέον δυνατότητα η οποία ονομάζεται Asynchronous Polling. Η συγκεκριμένη περίπτωση ανταλλαγής χρησιμοποιείται σε περίπτωση που ο παραλήπτης του αιτήματος είναι διατεθειμένος να δεχτεί το αίτημα αλλά όχι άμεσα. Ο πελάτης TAXII στέλνει ένα αίτημα στον κάτοχο των δεδομένων ενημερώνοντας τον, ότι είναι διατεθειμένος να υποστηρίξει ασύγχρονη μετάδοση δεδομένων. Η σύνδεση του παρόχου δέχεται το αίτημα όπως θα συνέβαινε και στην απλή αίτηση δεδομένων(Poll Request). Ο πάροχος απαντάει με ένα μήνυμα κατάστασης “Εκκρεμή”, το οποίο περιέχει και έναν δείκτη για το πότε αναμένεται να είναι έτοιμη η αποστολή των δεδομένων, όπως και για τον τρόπο που θα γίνει η αποστολή τους. Υπάρχουν δύο τρόποι για αποστολή των αποτελεσμάτων στον αιτούντα:

- Pulling Asynchronous Poll Requests: Οποιαδήποτε στιγμή μετά την παραλαβή του μηνύματος κατάστασης “Εκκρεμή”, ο αιτών μπορεί να στείλει στον πάροχο ένα μήνυμα ολοκλήρωσης της αποστολής (Poll Fulfillment Request Message). Αν η σύνδεση του παρόχου εντοπίσει κάποια σφάλμα κατά την αποστολή του αιτήματος, απαντάει με το κατάλληλο μήνυμα κατάστασης. Σε περίπτωση που ο αιτών διορθώσει το λάθος του τα

αποτελέσματα της αίτησης του θα πρέπει να είναι διαθέσιμα για να του σταλούν. Εάν για κάποιο λόγο τα αποτελέσματα δεν είναι διαθέσιμα, η σύνδεση του παρόχου θα πρέπει να απαντήσει με το μήνυμα κατάστασης "Asynchronous Poll Error". Εάν δεν υπάρχει κάποιο σφάλμα αλλά τα αποτελέσματα δεν είναι ακόμα διαθέσιμα, ο πάροχος θα πρέπει να αποστείλει ένα μήνυμα κατάστασης "Εκκρεμή" με τον νέο εκτιμώμενο χρόνο αποστολής. Τέλος αν τα αποτελέσματα είναι έτοιμα για αποστολή, η σύνδεση του παρόχου αποστέλλει το κατάλληλο μήνυμα στον αιτούντα με το περιεχόμενο που ζήτησε.

- Pushing Asynchronous Poll Requests: Σε αυτήν την περίπτωση ο αιτών δηλώνει την πρόθεση του στον παραγωγό να δεχτεί αποστολή δεδομένων σε κάποια συγκεκριμένη υπηρεσία εισερχομένων του (inbox service). Πληροφορίες σχετικά με τη υπηρεσία στην οποία θα γίνει η αποστολή, περιέχονται στο αρχικό μήνυμα αίτησης δεδομένων. Η σύνδεση του παραγωγού μόλις παραλάβει το αίτημα εξετάζει την υπηρεσία στην οποία θα γίνει η αποστολή. Για να πραγματοποιηθεί η αποστολή ο παραγωγός θα πρέπει να είναι διατεθειμένος να αποστείλει δεδομένα. Η υπηρεσία στην οποία θα γίνει η αποστολή, θα πρέπει να ορίζεται στο αρχικό μήνυμα αίτησης του πελάτη και ο παραγωγός θα πρέπει να είναι συμβατός με την αντίστοιχη υπηρεσία του πελάτη. Εάν ικανοποιούνται οι παραπάνω απαιτήσεις ο παραγωγός στέλνει ένα μήνυμα κατάστασης ¨ Εκκρεμή ¨. Η διαδικασία που ακολουθείται μετά είναι η ίδια όπως και στην περίπτωση της απλής ασύγχρονης αίτησης δεδομένων με την μόνη διαφορά ότι τα αποτελέσματα στέλνονται στη συγκεκριμένη υπηρεσία εισερχομένων που δηλώθηκε στο αρχικό μήνυμα του πελάτη.

2.5 Αναπαράσταση Μηνυμάτων

Κάθε μήνυμα TAXII αποτελείται από την επικεφαλίδα και το σώμα. Η επικεφαλίδα περιέχει πληροφορίες σχετικές με όλους τους τύπους σώματος μηνυμάτων, ενώ το σώμα περιέχει πληροφορίες για έναν συγκεκριμένο τύπο μηνύματος.

Η κεφαλίδα ενός μηνύματος περιέχει τα εξής πεδία:

- Message ID Ένα αναγνωριστικό του μηνύματος.

- Message Body Type: Τον τύπο του μηνύματος TAXII.

- In Response To: Το αναγνωριστικό του μηνύματος στο οποίο απαντάει το συγκεκριμένο μήνυμα.

- Extended Header: Επεκτάσεις επικεφαλίδων που δεν ορίζονται από το TAXII, αλλά από τρίτους.

- Signature: Κρυπτογραφημένες υπογραφές για κάποιο συγκεκριμένο μήνυμα TAXII.

Τα πεδία του σώματος ενός μηνύματος TAXII διαφέρουν ανάλογα με τον τύπο του μηνύματος τον οποίον αντιπροσωπεύουν. Στο παράρτημα Β παρουσιάζονται οι τύποι πεδίων σώματος.

2.6 Τρόποι Χειρισμού Μηνυμάτων

Σε αυτήν την ενότητα περιγράφονται οι απαιτούμενοι τρόποι χειρισμού του περιεχομένου από τους παραγωγούς TAXII. Το πρότυπο TAXII θέτει κάποιους περιορισμούς σχετικά με την διαχείριση του περιεχομένου, έτσι ώστε να επιτυγχάνεται συμβατότητα ανάμεσα στις αρχιτεκτονικές διαφορετικών παραγωγών.

Έλεγχοι πρόσβασης

Τα μηνύματα TAXII περιέχουν σημαντικές πληροφορίες σχετικά με απειλές στο κυβερνοχώρο. Αυτό από μόνο του αποτελεί έναν σημαντικό λόγο για να γίνεται έλεγχος στο ποιοι έχουν πρόσβαση σε τέτοιου είδους μηνύματα. Το TAXII δεν ορίζει ελέγχους πρόσβασης, ωστόσο κάνει κάποιες υποθέσεις για την επίδραση των πολιτικών ελέγχου πρόσβασης στην διάδοση του περιεχομένου των μηνυμάτων.

- Οι παραγωγοί έχουν τον πλήρη έλεγχο της πληροφορίας που μοιράζονται με τους καταναλωτές TAXII. Έχουν τη δυνατότητα να επεξεργαστούν, να τροποποιήσουν ή να αποκρύψουν εντελώς το περιεχόμενο TAXII από τους καταναλωτές. Μπορούν να

αποκρύψουν την ύπαρξη κάποιων υπηρεσιών σε ένα αίτημα ανακάλυψης υπηρεσιών ή την ύπαρξη κάποιων διαθέσιμων συλλογών δεδομένων κατά το αντίστοιχο αίτημα. Επίσης δεν είναι υποχρεωμένοι να γνωστοποιούν στους καταναλωτές την απόκρυψη ή τροποποίηση αυτών των πληροφοριών. Το εύρος των πληροφοριών που θα είναι διαθέσιμο βρίσκεται στην διακριτική ευχέρεια των παραγωγών.

- Σε περίπτωση που τα δικαιώματα πρόσβασης χρηστών αλλάξουν, δεδομένα που πριν δεν ήταν διαθέσιμα στον χρήστη τώρα μπορούν να γίνουν. Το TAXII δεν περιλαμβάνει κάποιο μήνυμα για την ενημέρωση των χρηστών σχετικά με την αλλαγή των δικαιωμάτων τους. Υπάρχουσες εγγραφές χρηστών θα πρέπει να παραμένουν ενεργές κατά την διάρκεια αλλαγής δικαιωμάτων.

Συλλογές Δεδομένων και Περιεχόμενο

Με τον όρο συλλογές δεδομένων, αναφερόμαστε στον τρόπο με τον οποίον οι παραγωγοί διαθέτουν περιεχόμενο. Παρακάτω παρουσιάζονται οι βασικές απαιτήσεις ανάμεσα στο περιεχόμενο TAXII και τις συλλογές δεδομένων TAXII:

- Το TAXII δεν προσφέρει λεπτομέρειες σχετικά με τα δεδομένα που μεταφέρονται μέσων των μηνυμάτων του. Πληροφορίες σχετικά με την μορφή του περιεχομένου ή τον τρόπο διαχείρισης του είναι άγνωστα προς το TAXII.

- Το TAXII υποστηρίζει δύο τύπους συλλογών δεδομένων. Τις διατεταγμένες συλλογές δεδομένων (Data Feed) και τις μη διατεταγμένες συλλογές δεδομένων (Data Set). Εγγραφές και στους δύο τύπους συλλογών γίνεται με παρόμοιο τρόπο. Η βασική διαφορά ανάμεσα στους δυο τύπους συλλογών είναι ότι στις μη διατεταγμένες συλλογές δεδομένων, οποιαδήποτε αλλαγή συμβαίνει σε δεδομένα κάποιας συλλογής θα πρέπει να εξετάζεται από την αρχή, καθώς δεν υπάρχει κάποιος μηχανισμός στα μηνύματα TAXII για να ελέγχει εάν η συγκεκριμένη εγγραφή αποτελεί καινούργια εγγραφή ή επανάληψη κάποιας παλιάς.

- Κατά την διαχείριση αιτημάτων τύπου Poll Request μια επιπλέον διαφορά που εμφανίζεται ανάμεσα στους δύο τύπους συλλογών δεδομένων, είναι η δυνατότητα περιορισμού του εύρους των αποτελεσμάτων στις διατεταγμένες συλλογές. Αυτό μπορεί να επιτευχθεί με την χρήση των ετικετών χρονοσφραγίδας. Ο αιτών ορίζει το εύρος των

δεδομένων που επιθυμεί να αποκτήσει και ο αποστολέας περιορίζει το σύνολο των δεδομένων ανάμεσα στα όρια που έχουν τεθεί. Όπως γίνεται αντιληπτό μια τέτοια διαδικασία δεν μπορεί να πραγματοποιηθεί σε μη διατεταγμένες συλλογές δεδομένων, καθώς οι ετικέτες χρονοσφραγίδας δεν υποστηρίζονται.

- Επίσης κατά την διάρκεια πολύτμηματικής μετάδοσης δεδομένων που ανήκουν σε διατεταγμένες συλλογές είναι απαραίτητο να διατηρείται η διάταξη των δεδομένων. Το σύνολο των δεδομένων χωρίζεται σε τμήματα και σε κάθε τμήμα δίνεται μια ετικέτα χρονοσφραγίδας έτσι ώστε να διατηρηθεί η σωστή σειρά των δεδομένων κατά την μετάδοση. Στις μη διατεταγμένες συλλογές δεν υπάρχει καμία τέτοια απαίτηση. Τα δεδομένα αποστέλλονται με οποιαδήποτε σειρά.

- Ένα επιπλέον χαρακτηριστικό των διατεταγμένων συλλογών δεδομένων είναι η αμεταβλητότητα των δεδομένων. Οι πάροχοι των συλλογών δεν θα πρέπει να επεμβαίνουν στα δεδομένα των συλλογών αντιθέτως εάν επιθυμούν να πραγματοποιήσουν κάποιες αλλαγές σε δεδομένα, αυτό θα πρέπει να γίνεται μόνο με την προσθήκη νέων εγγραφών στις συλλογές δεδομένων. Οι καινούργιες εγγραφές θα πρέπει να έχουν ετικέτες χρονοσφραγίδας μεταγενέστερες των είδη υπαρχουσών εγγραφών. Αυτό μπορεί να γίνει ευκολότερα αντιληπτό εάν αναλογιστούμε ότι κάποιος πελάτης δεν θα ξαναζητούσε δεδομένα που ανήκουν σε ένα συγκεκριμένο εύρος χρονοσφραγίδων, καθώς το περιεχόμενο τους είναι ήδη γνωστό σε αυτόν. Έτσι εάν ο παραγωγός επιθυμούσε την αλλαγή του περιεχομένου για κάποιο λόγο, ο μόνος τρόπος να γίνει το περιεχόμενο ορατό στους πελάτες θα ήταν να προστεθεί σε μια καινούργια εγγραφή στη συλλογή δεδομένων.

- Όπως προαναφέρθηκε το TAXII έχει την δυνατότητα χρήσης Inbox exchange για να προωθήσει δεδομένα σε μια συλλογή. Σε μια τέτοια ανταλλαγή ο κάτοχος της συλλογής είναι αυτός θα που θα αποφασίσει εάν θα συμπεριλάβει τα δεδομένα στην συλλογή ή όχι. Επίσης από αυτόν εξαρτάται το πότε θα τα συμπεριλάβει στην συλλογή, όπως επίσης εάν θα χρησιμοποιήσει τις ετικέτες χρονοσφραγίδας που έχουν τα δεδομένα ή αν θα αποφασίσει να τους δώσει καινούργιες ετικέτες χρονοσφραγίδας. Πολλές φορές τα μέρη που συμμετέχουν στην ανταλλαγή, διατυπώνουν κάποιους κανόνες για τον τρόπο που θα γίνει ο χειρισμός των δεδομένων.

- Μέχρι στιγμής οι συλλογές δεδομένων περιγράφηκαν σε συνάρτηση με τους πελάτες και το πώς μπορούν να αποκτήσουν δεδομένα μέσα από αυτές. Θα πρέπει να τονιστεί ωστόσο ότι ορισμένες συλλογές δεδομένων λειτουργούν απλά σαν δεξαμενές δεδομένων για τον ιδιοκτήτη τους χωρίς τα αποτελέσματα τους να γίνονται ορατά σε άλλους και χωρίς να δέχονται εγγραφές.

Εμφώλευση περιεχομένου και κρυπτογράφηση

Κατά την μεταφορά περιεχομένου από τον παραγωγό στον πελάτη, το πεδίο Content Binding δείχνει τον τύπο των δεδομένων προς μεταφορά. Πολλές φορές τα δεδομένα μπορούν να χρησιμοποιηθούν κατευθείαν από τον πελάτη, ωστόσο κάποιες φορές απαιτείται η μετατροπή τους σε άλλου τύπου δεδομένα πριν την χρήση τους. Για παράδειγμα κρυπτογραφημένα δεδομένα θα πρέπει πρώτα να αποκρυπτογραφηθούν πριν χρησιμοποιηθούν. Παρακάτω παρουσιάζονται 3 τρόποι για την κρυπτογράφηση και μεταφορά κρυπτογραφημένου περιεχομένου.

- Τυφλή Εμφώλευση: Σε αυτή την περίπτωση το πεδίο Content Binding δίνει πληροφορίες μόνο για τον τύπο δεδομένων του εξωτερικού περιβλήματος. Για παράδειγμα σε μια περίπτωση κρυπτογραφίας το πεδίο Content Binding θα είχε τιμή EncStr αλλά από εκεί και πέρα δεν θα γνωρίζαμε τίποτα για το περιεχόμενο μέσα στην κρυπτογραφημένη δομή.

- Ρητή Εμφώλευση: Σε αυτήν την περίπτωση το πεδίο Content Binding περιέχει πληροφορίες για τους τύπους δεδομένων όλων των στοιχείων της εμφωλευμένης δομής. Σύμφωνα με το προηγούμενο παράδειγμα των κρυπτογραφημένων δεδομένων το πεδίο Content Binding θα έχει τιμή EncStr | ThreatInfo. Μια τέτοια μορφή εμφώλευσης, βοηθάει τον παραλήπτη να γνωρίζει για την δομή του μηνύματος που δέχεται. Αυτό όμως συνεπάγεται, ότι η δομή του μηνύματος μπορεί να γίνει ορατή και σε κάποιον εξωτερικό παρατηρητή.

- Εμφώλευση τμήματος περιεχομένου: Αποτελεί την καλύτερη μορφή εμφώλευσης καθώς συνδυάζει τις δύο παραπάνω μεθόδους. Με βάση το παράδειγμα που χρησιμοποιήθηκε, το πεδίο Content Binding θα είχε τιμή EncStr|ContentBlock. Με αυτή την μέθοδο ο παραλήπτης μπορεί να γνωρίζει το περιεχόμενο του μηνύματος αφού προβεί στις

απαραίτητες ενέργειες αποκρυπτογράφησης, αλλά πέρα από αυτόν κανένας άλλος δεν έχει σαφή εικόνα για τα δεδομένα που περιέχονται.

Υποχρεώσεις Παραγωγών

Κατά την αποστολή δεδομένων από έναν παραγωγό σε έναν καταναλωτή υπάρχουν κάποιοι κανόνες που πρέπει να τηρούνται από την πλευρά του παραγωγού. Τα δεδομένα που στέλνονται στον καταναλωτή θα πρέπει να είναι υποστηριζόμενου τύπου από αυτόν. Κατά το αίτημα του, ο καταναλωτής ορίζει τι τύπους δεδομένων μπορεί να υποστηρίξει. Ο παραγωγός θα πρέπει να απαντάει με δεδομένα αυτών των τύπων. Επίσης στο αρχικό αίτημα του καταναλωτή μπορεί να συμπεριλαμβάνεται κάποιο ερώτημα (query). Ο παραγωγός θα πρέπει να απαντήσει μόνο με δεδομένα που ταιριάζουν με το συγκεκριμένο ερώτημα.

2.7 HTTP Protocol Binding Specification

Οι προδιαγραφές δέσμευσης του TAXII HTTP πρωτοκόλλου, ορίζουν ένα σύνολο απαιτήσεων, για την χρήση του πρωτοκόλλου μεταφοράς υπερκειμένου (HTTP) στην ανταλλαγή μηνυμάτων TAXII. Οι προδιαγραφές αυτές παρέχουν κανονιστικό κείμενο, σχετικά με την μετάδοση μηνυμάτων TAXII μέσω των πρωτοκόλλων μεταφοράς υπερκειμένου HTTP και HTTPS.

Σύμφωνα με την συγκεκριμένη προδιαγραφή ορίζονται δύο αναγνωριστικά της έκδοσης του πρωτοκόλλου TAXII. Το πρώτο αναγνωριστικό αφορά την μεταφορά μέσω του πρωτοκόλλου HTTP και το δεύτερο την μεταφορά μέσω του πρωτοκόλλου HTTPS.

TAXII PROTOCOL VERSION ID	HTTP VERSION
urn:taxii.mitre.org:protocol:http:1.0	HTTP:1.0
urn:taxii.mitre.org:protocol:https:1.0	HTTPS:1.0

Πίνακας 2.1: Αντιστοίχηση εκδόσεων πρωτοκόλλου TAXII και HTTP
(Mark Davidson and Charles Schmidt, 2013)

Με τον όρο TAXII HTTP headers αναφερόμαστε σε επικεφαλίδες HTTP, οι οποίες μπορούν να πάρουν τιμές μόνο μέσα από την προδιαγραφή του πρωτοκόλλου TAXII HTTP. Στον ακόλουθο πίνακα παρουσιάζονται οι τύποι επικεφαλίδων TAXII και οι απαιτήσεις για τον κάθε τύπο.

Τύπος Header	**Απαιτήσεις**
ACCEPT	Ορίζει τους αποδεκτούς τύπους δεδομένων κατά την απάντηση σύμφωνα με τα ακόλουθα κριτήρια: • Μόνο οι τύποι δεδομένων "application" γίνονται αποδεκτοί. • Όλοι οι τύποι δεδομένων θα πρέπει να έχουν έναν υποτύπο δεδομένων, ο οποίος θα ορίζεται στο πίνακα MIME(Media Types IANA Table), ως υποτύπος δεδομένων application. • Εάν υπάρχει επικεφαλίδα τύπου X-TAXII-ACCEPT, όλοι οι υποτύποι δεδομένων θα πρέπει να συμφωνούν με μια τουλάχιστον τιμή της επικεφαλίδας X-TAXII-ACCEPT.
Content-Type	Ορίζει τους αποδεκτούς τύπους δεδομένων σώματος, σύμφωνα με τα ακόλουθα κριτήρια: • Μόνο οι τύποι δεδομένων "application" γίνονται αποδεκτοί. • Όλοι οι τύποι δεδομένων θα πρέπει να έχουν έναν υποτύπο δεδομένων ο οποίος θα ορίζεται στο πίνακα MIME(Media Types IANA Table) ως υποτύπος δεδομένων application. • Εάν υπάρχει επικεφαλίδα τύπου X-TAXII-ACCEPT, όλοι οι υποτύποι δεδομένων θα πρέπει να συμφωνούν με μια τουλάχιστον τιμή της

	επικεφαλίδας X-TAXII-ACCEPT.
X-TAXII-ACCEPT	Η επικεφαλίδα X-TAXII-ACCEPT είναι παρόμοια με την επικεφαλίδα ACCEPT. Η μόνη διαφορά είναι ότι η επικεφαλίδα accept χρησιμοποιεί τον πίνακα MIME(Media Types IANA Table) για τις αποδεκτούς τύπους των απαντήσεων, ενώ η X-TAXII-ACCEPT αποδεκτούς τύπους μηνυμάτων TAXII(TAXII Message Bindings). Η επικεφαλίδα X-TAXII-ACCEPT θα πρέπει να περιέχει τα αναγνωριστικά των αποδεκτών τύπων μηνυμάτων (TAXII Message Binding Version IDs). Εάν δεν υπάρχει επικεφαλίδα X-TAXII-ACCEPT υποθέτουμε ότι όλοι οι τύποι μηνυμάτων είναι δεκτοί
X-TAXII-Content-Type	Η επικεφαλίδα X-TAXII-Content-Type είναι παρόμοια με την επικεφαλίδα Content-Type. Η μόνη διαφορά είναι ότι η επικεφαλίδα Content-Type χρησιμοποιεί τον πίνακα MIME(Media Types IANA) για τον ορισμό της μορφής του σώματος των δεδομένων, ενώ στην X-TAXII-Content-Type χρησιμοποιείται το TAXII Message Binding για το περιεχόμενο του σώματος των δεδομένων.
X-TAXII-Protocol	Δείχνει τον τύπο πρωτοκόλλου TAXII, σύμφωνα με τον οποίον συμμορφώνεται το HTTP μήνυμα.
X-TAXII-Services	Η επικεφαλίδα X-TAXII-Services προσδιορίζει την έκδοση των υπηρεσιών TAXII με την οποία συμφωνεί το αίτημα HTTP. Εάν μια επικεφαλίδα τύπου X-TAXII-Services είναι παρούσα σε ένα HTTP μήνυμα, ο παραλήπτης μπορεί να καταλάβει ότι περιέχεται κάποιο

	TAXII μήνυμα, σε διαφορετική περίπτωση θα υποθέσει ότι απλά είναι ένα μήνυμα HTTP.
HTTP REQUESTS	Αιτήματα HTTP που αποστέλλονται κατά την ανταλλαγή μηνυμάτων TAXII θα πρέπει να συμμορφώνονται με του κανόνες των επικεφαλίδων HTTP TAXII Headers που περιγράφηκαν νωρίτερα. Θα πρέπει να χρησιμοποιούν μέθοδο αίτησης POST και να περιέχουν κάποιο TAXII μήνυμα στο σώμα της αίτησης.
HTTP RESPONSES:	Απαντήσεις HTTP που αποστέλλονται κατά την ανταλλαγή μηνυμάτων TAXII θα πρέπει να συμμορφώνονται με του κανόνες των επικεφαλίδων HTTP TAXII Headers και να περιέχουν κάποιο TAXII μήνυμα στο σώμα της απάντησης, σε περίπτωση που το αίτημα απαντάται επιτυχώς.

Πίνακας 2.2: TAXII Headers

2.7.1 Διαχείριση Απαντήσεων Χωρίς Μηνύματα TAXII

Σε κάποιες περιπτώσεις οι χρήστες TAXII, δεν συμπεριλαμβάνουν μηνύματα TAXII στις απαντήσεις του. Κάποιο σφάλμα που έχει πραγματοποιηθεί από κάποιον ο οποίος δεν γνωρίζει την δομή τα μηνυμάτων TAXII, είναι μια τέτοια περίπτωση. Ο χρήστης TAXII θα πρέπει να είναι σε θέση να απαντήσει χωρίς την χρήση μηνυμάτων TAXII. Ακολουθεί ένας πινάκας με την αντιστοιχία μεταξύ κωδικών κατάστασης HTTP και TAXII.

HTTP Status Code	**TAXII Status Type**
400 - Bad Request	Bad Message
401 – Unauthorized	Unauthorized
403 – Forbidden	Unauthorized
406 - Not Acceptable	Unsupported Message Binding

407 - Proxy Authentication Required	Unauthorized
413 - Request Entity Too Large	Bad Message
415 - Unsupported Media Type	Unsupported Message Binding
All other Status Codes	Failure

Πίνακας 2.3: Αντιστοιχία κωδικών κατάστασης HTTP και TAXII (Davidson & Schmidt 2013)

Το TLS (Transport Layer Security) είναι ένα πρωτόκολλο κρυπτογράφησης το οποίο παρέχει ασφαλέστερη επικοινωνία. Στην περίπτωση χρήσης TLS υπάρχει αντιστοιχία μεταξύ κωδικών κατάστασης TLS και TAXII.

TLS Alert Description	TAXII Status Type
40 - Handshake Failure	Unauthorized
41 - No Certificate	Unauthorized
42 - Bad Certificate	Unauthorized
43 - Unsupported Certificate	Unauthorized
48 - Unknown CA	Unauthorized
49 - Access Denied	Unauthorized
All other codes	Failure

Πίνακας2.4: Αντιστοιχία κωδικών κατάστασης TLS και TAXII (Davidson & Schmidt 2013)

Θα πρέπει να τονιστεί πως τα μηνύματα TAXII δεν μεταφέρουν πληροφορίες ελέγχου ταυτότητας. Η ταυτοποίηση των χρηστών γίνεται από τα εκάστοτε πρωτόκολλα μεταφοράς. Ενώ το TAXII περιέχει δυνατότητα κρυπτογράφησης περιεχομένου, βασίζεται στην κρυπτογράφηση σε επίπεδο δικτύου για την προστασία του μηνύματος TAXII κατά την μεταφορά. Οι μηχανισμοί ελέγχου ταυτότητας των πρωτοκόλλων μεταφοράς είναι αυτοί που φροντίζουν για την ασφαλή μεταφορά των μηνυμάτων TAXII.

Πρωτόκολλο Μεταφοράς	**Δυνατότητες**
HTTP	• Client Authentication
HTTPS	• Server Authentication • Client Authentication

	• Encryption

Πίνακας 2.5: Δυνατότητες Πρωτοκόλλων Μεταφοράς

2.8 XML Message Binding

Στην ακόλουθη ενότητα περιγράφεται η έκφραση των μηνυμάτων TAXII μέσω του συντακτικού XML. Η XML (Extensible Markup Language) δημιουργήθηκε για την αποθήκευση και την μεταφορά δεδομένων. Η συσχέτισης της έκδοσης μηνυμάτων TAXII, με την έκδοση της XML περιγράφεται ως ***urn:taxii.mitre.org:message:xml:1.1*** (Davidson & Schmidt 2014e)**.**

Δομή

Για κάθε διαφορετικό τύπο μηνυμάτων TAXII ορίζεται ένα διαφορετικό στοιχείο XML. Κάθε XML στοιχείο αντιπροσωπεύει κάποιο μήνυμα το οποίο παρουσιάζεται σαν στοιχείο root σε ένα XML σχήμα. Σε ένα XML έγγραφο ενός μηνύματος TAXII, δεν υπάρχει κάποια ξεχωριστή περιοχή για τις επικεφαλίδες του μηνύματος και κάποια ξεχωριστή για το περιεχόμενο, αντίθετα και οι δύο τύποι πεδίων αντιμετωπίζονται ως ισάξιοι. Επίσης ορίζεται αυστηρή διάταξη των στοιχείων. Σκοπός είναι ο ταχύτερος εντοπισμός των διαφόρων πεδίων μέσα στο έγγραφο. Τα χαρακτηρίστηκα του κάθε στοιχείου μπορούν να εμφανίζονται με οποιαδήποτε σειρά. Δεν υπάρχει καμία υποχρέωση ελέγχου των XML σχημάτων πριν την αποστολή τους. Εάν ο παραλήπτης του μηνύματος αντιληφθεί κάποιο σφάλμα στο XML σχήμα που περιέχεται στο μήνυμα, είναι υποχρεωμένος να απαντήσει με ένα μήνυμα κατάστασης "BAD MESSAGE".

Απαιτήσεις XML Πεδίων

Κάθε πεδίο σε ένα μήνυμα TAXII έχει μια συγκεκριμένη δομή. Αντιστοίχως και τα πεδία ενός XML σχήματος έχουν κάποιες συγκεκριμένες απαιτήσεις τιμών.

Πεδίο	**Επιτρεπόμενες Τιμές**
Timestamps	Το TAXII XML BINDING απαιτεί πεδία που

	χρησιμοποιούν ετικέτες χρονοσφραγίδας να παίρνουν τιμές τύπου XML dateTime. Επίσης κάθε χρονοσφραγίδα μπορεί να έχει 0 έως 6 δεκαδικά (xx:xx:xx.(0-6)).
Extended Headers	Το TAXII επιτρέπει την προσθήκη επεκτάσεων επικεφαλίδων σε όλα τα μηνύματα TAXII. Οι επεκτάσεις επικεφαλίδων, ορίζονται από τους χρήστες και δεν ανήκουν στις προδιαγραφές του TAXII. Οι επεκτάσεις επικεφαλίδων αναπαριστώνται ως ζευγάρια ονόματος-τιμής.
Status Details	Τα μηνύματα κατάστασης TAXII, μπορούν επιπρόσθετα να έχουν κάποιο πεδίο, με πρόσθετες λεπτομέρειες σχετικά με το μήνυμα που αποστέλλεται. Όπως και πριν, το πεδίο λεπτομερειών κατάστασης αποτελείται από ζευγάρια ονόματος τιμής.

Πίνακας 2.6: XML Field Value

Ονόματα και Αναγνωριστικά

Το TAXII χρησιμοποιεί διάφορες κλάσεις αναγνωριστικών, τα οποία θεωρούνται δεσμευμένες λέξεις και δεν μπορούν να χρησιμοποιηθούν, πέραν του σκοπού δημιουργίας τους. Σε αυτές περιλαμβάνονται:

- Extended Header names
- Status Detail subfield names

- Query Format IDs
- Content Binding IDs
- Content Binding Subtype IDs
- Protocol Binding Version IDs
- Message Binding Version IDs
- TAXII Services Version IDs

Κατά την δημιουργία ενός αναγνωριστικού από κάποιον τρίτο, θα πρέπει να ορίζεται η αρχή που είναι υπεύθυνη για αυτό το αναγνωριστικό, προκειμένου να αποφευχθούν συγκρούσεις ανάμεσα σε αναγνωριστικά που έχουν δημιουργηθεί από διαφορετικές μεριές.

Αναγνωριστικά τα οποία δεν χρειάζεται να είναι μοναδικά για όλους θα πρέπει επίσης να συμμορφώνονται με τις απαιτήσεις μορφοποίησης URI (Uniform Resource Identifier). Ωστόσο υπεύθυνος για την αποφυγή συγκρούσεων στον τομέα ευθύνης του, είναι ο δημιουργός τους.

2.9 Ερωτήματα

Το πρότυπο TAXII δίνει την δυνατότητα χρήσης ερωτημάτων. Μέσω της χρήσης ερωτημάτων κάποιος πελάτης μπορεί να αιτηθεί συγκεκριμένο περιεχόμενο μιας συλλογής δεδομένων το οποίο θα ικανοποίει τα συγκεκριμένα ερωτήματα. Το αναγνωριστικό της μορφής των ερωτημάτων TAXII είναι **urn:taxii.mitre.org:query:default:1.0**(Davidson & Schmidt 2014c)**.**

Η ορολογία που χρησιμοποιείται κατά την σύνταξη ερωτημάτων TAXII περιγράφεται παρακάτω:

- Capability Module: Είναι ένα καθορισμένο σύνολο συσχετίσεων (ισότητα, μεγαλύτερο από, μικρότερο από) οι οποίες μπορούν να χρησιμοποιηθούν κατά την δημιουργία κριτηρίων επιλογής.

- Targeting Expression: Μια έκφραση η οποία διευκρινίζει την περιοχή της εγγραφής όπου θα γίνει η αναζήτηση.

- Targeting Expression Vocabulary: Ένα προκαθορισμένο σύνολο εκφράσεων, το οποίο χρησιμοποιείται σε εκφράσεις τύπου Targeting Expressions.

- Node: Ένα στοιχείο του λεξιλογίου για εκφράσεις τύπου Targeting Expressions.

Ορίζονται 3 τύποι κατάστασης(status types), οι οποίοι χρησιμοποιούνται σε περιπτώσεις σφαλμάτων σχετιζόμενων με ερωτήματα TAXII.

Status Type	**Περιγραφή**
Unsupported Capability Module	Ορίζονται συσχετίσεις οι οποίες δεν υποστηρίζονται από την υπηρεσία TAXII.
Unsupported Targeting Expression	Ορίζεται έκφραση τύπου Targeting Expression η οποία δεν υποστηρίζεται από την υπηρεσία TAXII.
Unsupported Targeting Expression Vocabulary	Ορίζεται λεξιλόγιο για Targeting Expressions το οποίο δεν υποστηρίζεται.

Πίνακας 2.7 : Τύποι κατάστασης

2.9.1 Δομή Ερωτημάτων

Το πρότυπο TAXII ορίζει μια προεπιλεγμένη δομή η οποία θα πρέπει να ακολουθείται κατά την σύνταξη των ερωτημάτων TAXII. Σύμφωνα με αυτήν την δομή κάποιο ερώτημα TAXII πρέπει να έχει την παρακάτω μορφή:

Πεδίο	**Περιγραφή**
Default Query	Περιέχει ένα ερώτημα TAXII.
Targeting Expression Vocabulary ID	Ορίζει το λεξιλόγιο για τις εκφράσεις του ερωτήματος.
Criteria	Περιέχει τα κριτήρια σύμφωνα με τα οποία ικανοποιείται το ερώτημα.
Operator	Περιέχει τους λογικούς τελεστές που χρησιμοποιούνται για την σύνταξη των ερωτημάτων.
Target	Περιέχει την έκφραση στόχο του ερωτήματος. Υποδεικνύει την περιοχή της συλλογής την οποία αιτείται ο πελάτης με το συγκεκριμένο ερώτημα.
Test	Περιέχει την δοκιμή της περιοχής της συλλογής την οποία αιτείται ο πελάτης με το συγκεκριμένο ερώτημα.
Capability ID	Περιέχει το αναγνωριστικό ενός συγκεκριμένου μοντέλου συσχετίσεων.
Relationship	Περιέχει τις συσχετίσεις που προκύπτουν μέσω του Capability ID.

Πίνακας 2.8: Δομή Ερωτημάτων

2.9.2 Αξιολόγηση Ερωτημάτων

Μόλις κάποιος παραγωγός λάβει κάποιο ερώτημα θα πρέπει να το αξιολογήσει πριν προχωρήσει στον διαμοιρασμό περιεχομένου κάποιας συλλογής του. Σε αυτήν την ενότητα παρουσιάζονται τα βασικά κριτήρια αξιολόγησης ερωτημάτων.

Με την δημιουργία ερωτημάτων ο πελάτης αιτείται εγγραφές της συλλογής δεδομένων που συμφωνούν με τα κριτήρια που αυτός έθεσε. Σε περίπτωση που ο παραγωγός δεν μπορεί ή δεν θέλει να απαντήσει στο ερώτημα, θα πρέπει να απαντήσει με ένα μήνυμα κατάστασης (Query Not Supported).

Τα ερωτήματα θα πρέπει να απαντώνται με βάση την σειρά βημάτων:

1. Έλεγχος του ερωτήματος για τυχόν σφάλματα ή μη υποστηριζόμενα χαρακτηριστικά. Αν εντοπιστεί κάτι από τα δύο, απάντηση με το κατάλληλο μήνυμα κατάστασης.

2. Για κάθε εγγραφή της συλλογής γίνεται αξιολόγηση των κριτηρίων. Εάν τα κριτήρια είναι αληθή, η συγκεκριμένη εγγραφή προστίθεται στην σύνολο του αποτελέσματος.

Τα κριτήρια θα πρέπει να απαντώνται με την δημιουργία μιας λίστας υποκριτηρίων και στην συνέχεια με αξιολόγησης τους, με βάση τους παρακάτω κανόνες:

A. Αν το υποκριτήριο είναι αληθές και συνδέεται με κάποιο άλλο υποκριτήριο με τον τελεστή OR, το βασικό κριτήριο αξιολογείται ως αληθές.

B. Αν το υποκριτήριο είναι αληθές και συνδέεται με κάποιο άλλο υποκριτήριο με τον τελεστή AND, συνεχίζεται ο υπολογισμός έως ότου δεν υπάρχουν άλλα υποκριτήρια. Το βασικό κριτήριο αξιολογείται ως αληθές.

C. Αν το υποκριτήριο είναι ψευδές και συνδέεται με κάποιο άλλο υποκριτήριο με τον τελεστή AND, το βασικό κριτήριο αξιολογείται ως ψευδές.

D. Αν το υποκριτήριο είναι ψευδές και συνδέεται με κάποιο άλλο υποκριτήριο με τον τελεστή OR, συνεχίζεται ο υπολογισμός έως ότου δεν υπάρχουν άλλα υποκριτήρια. Το βασικό κριτήριο αξιολογείται ως ψευδές.

2.9.3 Σύνταξη Εκφράσεων

Οι εκφράσεις (Targeting Expressions), περιέχονται στο πεδίο Target ενός ερωτήματος. Για την δημιουργία μιας έκφρασης απαιτείται ο συνδυασμός ενός λεξιλογίου εκφράσεων και του κατάλληλου συντακτικού.

Όλες οι εκφράσεις χρησιμοποιούν τον τρόπο σύνταξης Slash Notation. Σύμφωνα με αυτήν την μορφή σύνταξης, όλες οι εκφράσεις αποτελούνται από κόμβους (nodes), χωρισμένους με τον ειδικό χαρακτήρα (/). Ο ειδικός χαρακτήρας(*) αντιπροσωπεύει έναν οποιοδήποτε κόμβο και ο ειδικός χαρακτήρας (**) ένα ή περισσότερους κόμβους.

Το λεξιλόγιο εκφράσεων (Targeting Expression vocabulary), ορίζει ποιοι κόμβοι επιτρέπονται για την δημιουργία εκφράσεων, την ιεραρχία τους, καθώς και ποιοι ειδικοί χαρακτήρες επιτρέπονται.

Λεξιλόγια εκφράσεων που δημιουργούνται από τρίτους, θα πρέπει να περιέχουν τις εξής πληροφορίες:

- Το αναγνωριστικό του λεξιλογίου σε μορφή URI.
- Το σύνολο των επιτρεπόμενων κόμβων.
- Την ιεραρχία των επιτρεπόμενων κόμβων.
- Επεξήγηση των ειδικών χαρακτήρων.
- Τουλάχιστον ένα παράδειγμα υλοποίησης έκφρασης.

2.10 Αναγνωριστικά περιεχομένου

Τα αναγνωριστικά περιεχομένου (Content Binding IDs) χρησιμοποιούνται για τον προσδιορισμό του τύπου περιεχομένου που χρησιμοποιείται στις συλλογές δεδομένων, του τύπου περιεχομένου που επεξεργάζονται οι υπηρεσίες TAXII, ή για το φιλτράρισμα του περιεχομένου που δέχεται κάποιος πελάτης στα πλαίσια μιας εγγραφής.

STIX

Το STIX αποτελεί μια προσπάθεια ανάπτυξης μιας τυποποιημένης γλώσσας για την αναπαράσταση πληροφοριών σχετικά με απειλές του κυβερνοχώρου. Το STIX δημιουργήθηκε από την εταιρία MITRE για λογαριασμό του υπουργείου εσωτερικής ασφάλειας της Αμερικής. Τα περιεχόμενα του STIX αναπαριστώνται με την χρήση XML.

Τα αναγνωριστικά περιεχομένου του STIX έχουν την παρακάτω μορφή

"urn:stix.mitre.org:" + format + ":" + version(Davidson & Schmidt 2014b)

Το format στις περισσότερες των περιπτώσεων παίρνει τιμή XML, ενώ το version την εκάστοτε έκδοση του STIX. Παρακάτω παρουσιάζονται τα αναγνωριστικά περιεχομένου για τις διάφορες εκδόσεις του STIX.

STIX Version	**Content Binding ID**
STIX XML 1.0 http://stix.mitre.org/language/version1.0/	urn:stix.mitre.org:xml:1.0
STIX XML 1.0.1 http://stix.mitre.org/language/version1.0.1/	urn:stix.mitre.org:xml:1.0.1
STIX XML 1.1 http://stix.mitre.org/language/version1.1/	urn:stix.mitre.org:xml:1.1

Πίνακας 2.9: Αναγνωριστικά περιεχομένου STIX(Davidson & Schmidt 2014b)

Αναγνωριστικά περιεχομένου από τρίτους

Το TAXII επιτρέπει τον ορισμό αναγνωριστικών περιεχομένου από τρίτους. Η κοινότητα του TAXII ενθαρρύνει τους χρήστες να υποβάλλουν τα αναγνωριστικά που αυτοί ορίζουν, με στόχο την ανάπτυξη της διαλειτουργικότητας μεταξύ των χρηστών TAXII.

Κεφάλαιο 3
Ανάλυση Κακόβουλου Λογισμικού με το Cuckoo Sandbox

Στο παρόν κεφάλαιο παρουσιάζεται η λειτουργία του συστήματος ανάλυσης κακόβουλου λογισμικού Cuckoo Sandbox. Πριν προχωρήσουμε στην ανάλυση του Cuckoo και του τρόπου λειτουργιάς του καλό θα ήταν να αναφέρουμε μερικά πράγματα σχετικά με την ανάλυση κακόβουλου λογισμικού(Malware Analysis). Με τον όρο ανάλυση κακόβουλου λογισμικού, αναφερόμαστε στις διαδικασίες εντοπισμού κακόβουλων συμπεριφορών, στον λόγο ύπαρξης τους καθώς και στους στόχους τους. Έχουν κυριαρχήσει δυο κύριες μεθοδολογίες γύρω από την ανάλυση κακόβουλου λογισμικού. Η στατική ανάλυση γνωστή ως ανάλυση κώδικα και η δυναμική ανάλυση ή αλλιώς ανάλυση συμπεριφοράς. Η πρώτη μεθοδολογία απαιτεί καλή γνώση προγραμματισμού, καθώς το προς ανάλυση λογισμικό δεν εκτελείται αλλά αναλύεται ο κώδικας του. Σε αντίθεση στην δυναμική ανάλυση το λογισμικό εκτελείται και μελετάται η συμπεριφορά του. Λαμβάνοντας υπόψη τις αρνητικές συνέπειες που μπορεί να έχει η εκτέλεση ενός κακόβουλου λογισμικού σε ένα υπολογιστικό σύστημα, (διαγραφή αρχείων, τροποποίηση αρχείων, αλλαγές registry, κλοπή προσωπικών δεδομένων) είναι ιδιαίτερα σημαντικό να εξασφαλίσουμε ότι η εκτέλεση του κακόβουλου λογισμικού θα γίνει σε ένα απομονωμένο περιβάλλον. Πάνω σε αυτήν την λογική βασίζεται η λειτουργία του Cuckoo Sandbox(Oktavianto 2013)

3.1 Cuckoo Sandbox

Στην επιστήμη της Ασφάλειας υπολογιστών ένα sandbox είναι ένας μηχανισμός ασφαλείας για τον διαχωρισμό εκτελούμενων προγραμμάτων. Χρησιμοποιείται πολλές φορές για την εκτέλεση κώδικα ο οποίος δεν έχει ελεγχθεί ή για την εκτέλεση προγραμμάτων τα οποία προέρχονται από μη διαπιστευμένες πηγές(Anon n.d.).

Ο μηχανισμός μπορεί να χρησιμοποιηθεί στην ανίχνευση κακόβουλου λογισμικού. Σκοπός του συγκεκριμένου συστήματος, είναι η εκτέλεση κάποιας άγνωστης εφαρμογής ή αρχείου, σε απομονωμένο περιβάλλον και η συγκέντρωση των πληροφοριών από την εκτέλεση.

Πριν την χρήση του Cuckoo είναι ιδιαίτερα σημαντικό να δημιουργήσουμε το κατάλληλο περιβάλλον μέσα στο οποίο θα εκτελεστεί το κακόβουλο λογισμικό. Αυτό το στάδιο είναι ιδιαίτερα σημαντικό, καθώς διαφορετικές υλοποιήσεις του απομονωμένου περιβάλλοντος, μπορούν να οδηγήσουν σε διαφορετικά συμπεράσματα σχετικά με τον τρόπο λειτουργίας του υπό εξέταση αρχείου. Κατά το στάδιο της υλοποίησης θα πρέπει πάντα να θυμόμαστε ότι ο στόχος είναι να δημιουργηθεί ένα σύστημα το οποίο θα είναι σύμφωνο με τις απαιτήσεις μας, αλλά συγχρόνως θα αποτελεί μια ρεαλιστική προσέγγιση.

Για παράδειγμα έστω ότι εξετάζεται κάποιο κακόβουλο αρχείο, το οποίο έχει σχεδιαστεί για να εκτελείται πάνω σε αρχείου τύπου 'pdf'. Εάν από το απομονωμένο σύστημα που δημιουργήθηκε απουσιάζουν τέτοιου είδους αρχεία, θα βγουν λάθος συμπεράσματα για τον τρόπο λειτουργίας του κακόβουλου λογισμικού.

Το Cuckoo είναι ένα open source σύστημα ανάλυσης κακόβουλου λογισμικού, το οποίο υλοποιεί δυναμική ανάλυση, καθώς τα αρχεία εκτελούνται και αναλύονται σε πραγματικό χρόνο. Χρησιμοποιείται για την εκτέλεση και αυτόματη ανάλυση ενός υπό εξέταση αρχείου παρέχοντας αναλυτικές πληροφορίες σχετικά με την δράση του αρχείου σε ένα απομονωμένο περιβάλλον.

Το Cuckoo παρέχει του συγκεκριμένους τύπους αποτελεσμάτων:

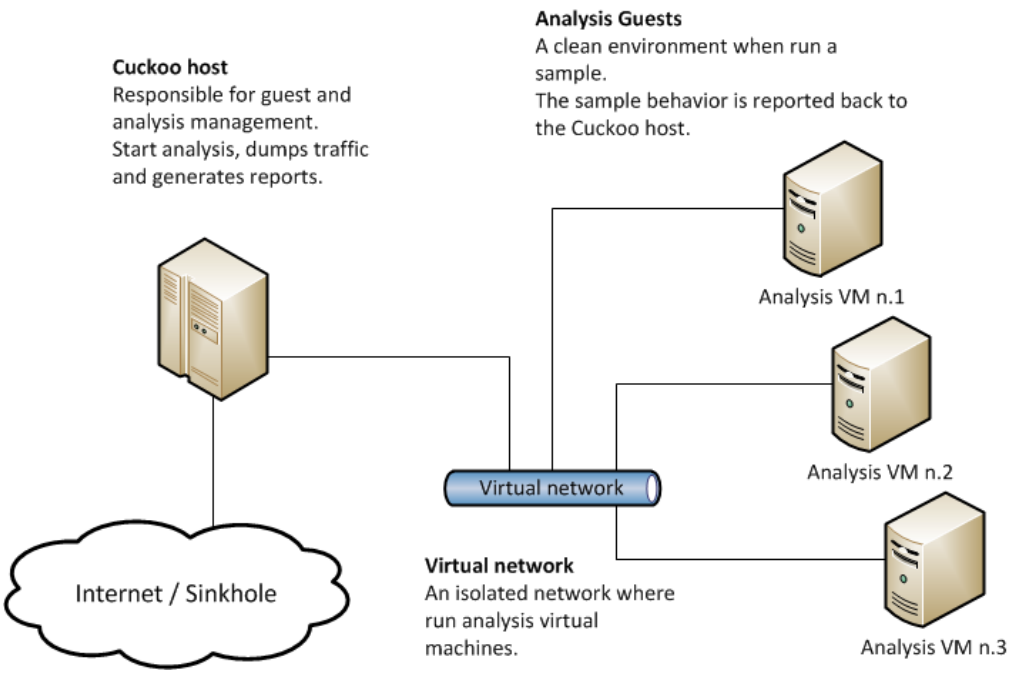
- Αρχεία που δημιουργήθηκαν ή διαγράφηκαν κατά την εκτέλεση του λογισμικού
- Απεικονίσεις μνήμης του συστήματος
- Απεικονίσεις μνήμης κακόβουλων διεργασιών
- Κίνηση δικτύου
- Απεικονίσεις της οθόνης του απομονωμένου συστήματος
- Ίχνη που δημιουργήθηκαν από την εκτέλεση του κακόβουλου λογισμικού

Το Cuckoo προσφέρει δυνατότητες ανάλυσης για τους παρακάτω τύπους αρχείων:

- Dll αρχεία
- Pdf έγγραφα
- Microsoft Office έγγραφα
- URL και HTML αρχεία
- CPL αρχεία
- Visual Basic scripts
- ZIP αρχεία
- Εκτελέσιμα αρχεία Windows
- Python αρχεία
- Java αρχεία

3.1.1 Αρχιτεκτονική

Το Cuckoo Sandbox αποτελείται από ένα λογισμικό κεντρικής διαχείρισης, το οποίο εκτελεί τα διάφορα δείγματα κακόβουλου λογισμικού στα συνδεδεμένα με αυτό απομονωμένα μηχανήματα. Τα μηχανήματα μπορούν να είναι είτε ψηφιακά είτε φυσικά. Ο κεντρικός κόμβος συνδέεται με τους κόμβους ανάλυσης μέσω ενός ψηφιακού δικτύου, για την εξασφάλιση της μέγιστης ασφάλειας κατά την εκτέλεση των εξεταζόμενων αρχείων. Μέσω αυτού του δικτύου συλλέγει τις αναφορές εκτέλεσης από τους κόμβους ανάλυσης.



Εικόνα 3.1:Συνδεσμολογία δικτύου Cuckoo(Oktavianto 2013)

3.1.2 Βιβλιοθήκες

Το Cuckoo περιέχει μια σειρά από βιβλιοθήκες , που εξασφαλίζουν την ορθότερη ανάλυση του κακόβουλου λογισμικού. Στον παρακάτω πίνακα παρουσιάζονται οι δυνατότητες των βιβλιοθηκών του Cuckoo.

Βιβλιοθήκη	**Δυνατότητες**
YARA	Αποτελεί ένα εργαλείο ομαδοποίησης κακόβουλων λογισμικών. Βασίζεται στην δημιουργία 'οικογενειών' κακόβουλων λογισμικών, σύμφωνα με το δυαδικό πρότυπο που ακολουθούν.
SSDEEP	Χρησιμοποιείται για τον υπολογισμός των τιμών 'Fuzzy hashing'
VOLATILITY	Χρησιμοποιείται στην ανάλυση της μνήμης του υπό εξέταση συστήματος. Ιδιαίτερα σημαντικό

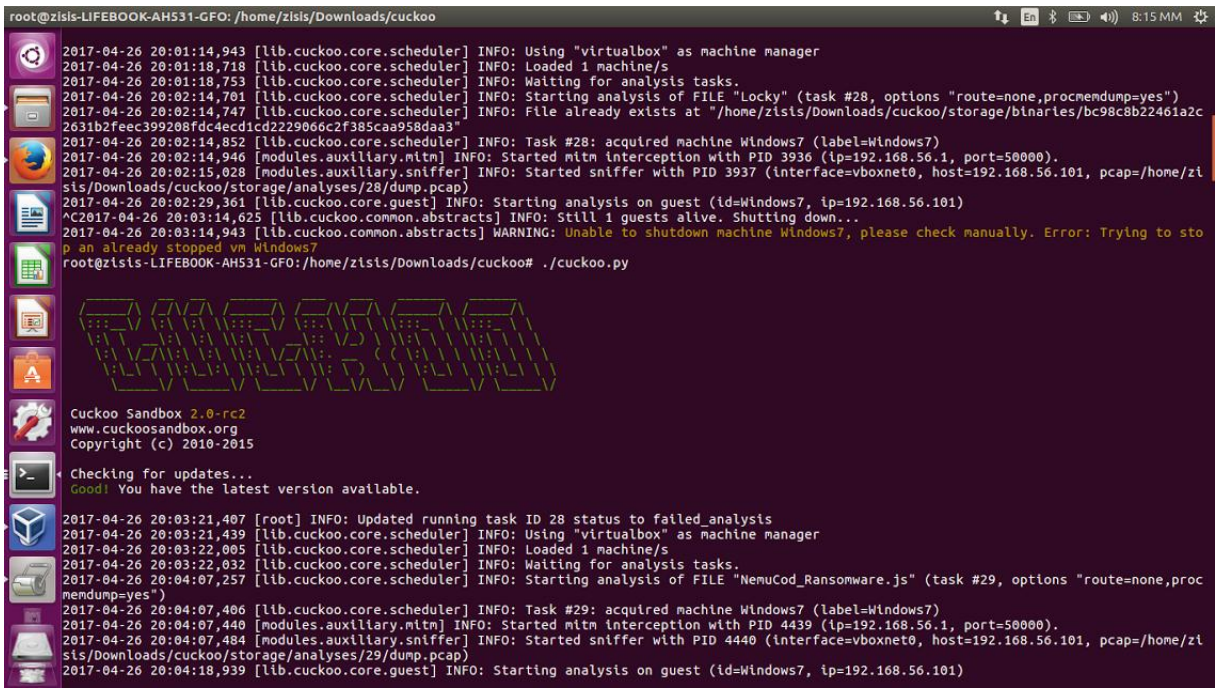
	εργαλείο για runtime αναλύσεις.
MAGIC	Χρησιμοποιείται για την αναγνώριση της μορφής των διαφόρων αρχείων.
DPKT	Ιδιαίτερα σημαντικό εργαλείο για την απόκτηση πληροφοριών από PCAP αρχεία.

Πίνακας 3.1: Βιβλιοθήκες Cuckoo(Oktavianto 2013)

3.2 Ανάλυση Κακόβουλου Λογισμικού

Αφού εγκατασταθεί το Cuckoo Sandbox και δημιουργηθεί το περιβάλλον ανάλυσης(Cuckoo Sandbox 2017; Oracle Corporation, 2017), ξεκινάει η διαδικασία εξέτασης κάποιου κακόβουλου δείγματος. Για την ανάλυση κάποιου κακόβουλο λογισμικού ακολουθούνται τα παρακάτω βήματα:

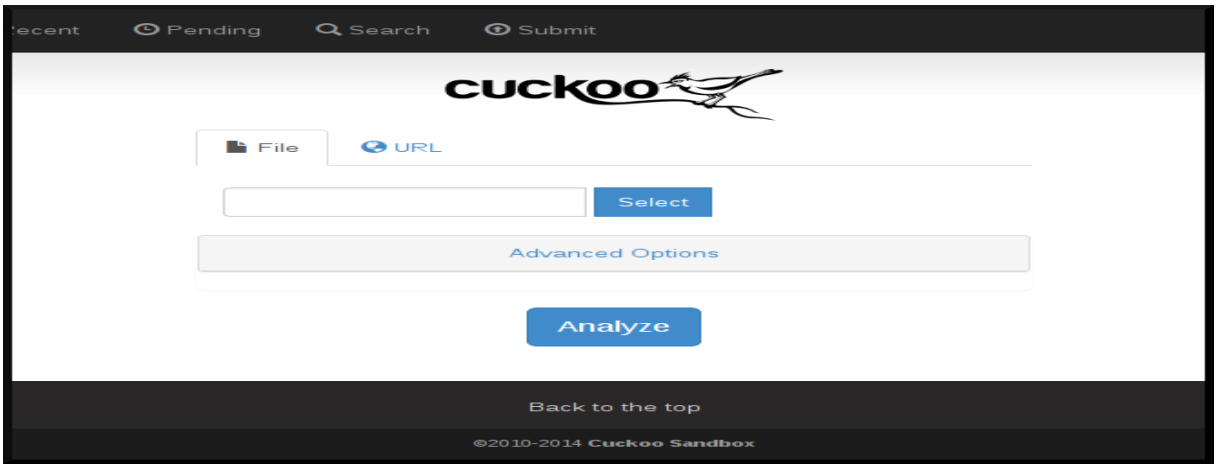
1. Εκτελείται το αρχείο cuckoo.py το οποίο βρίσκεται στο κατάλογο Cuckoo της εγκατάστασης.



Εικόνα 3.2: Εκκίνηση Cuckoo

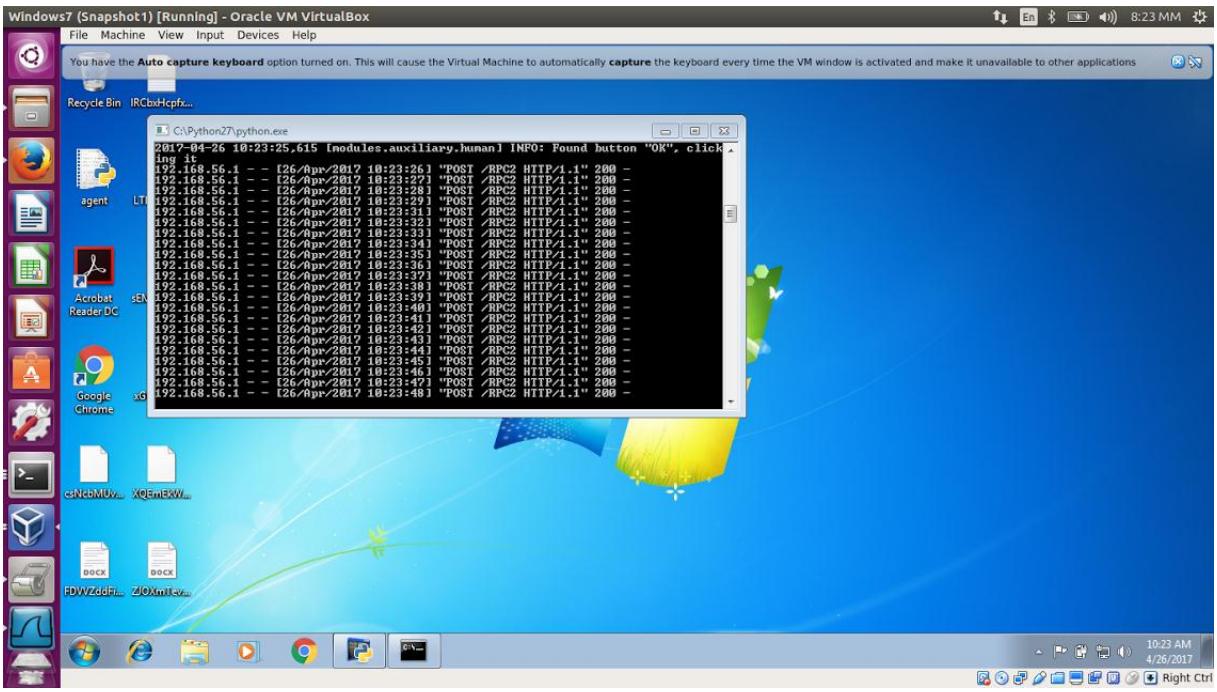
Όπως φαίνεται στην παραπάνω εικόνα το Cuckoo φορτώνει το εικονικό μηχάνημα που επιλέξαμε για να τεστάρει τα κακόβουλο λογισμικό.

2. Στην συνέχεια επιλέγεται κάποιο αρχείο προς εξέταση και είτε μέσω της γραμμή εντολών, εκτελώντας το αρχείο submit.py, είτε μέσω του web interface του Cuckoo, υποβάλλεται προς εξέταση.



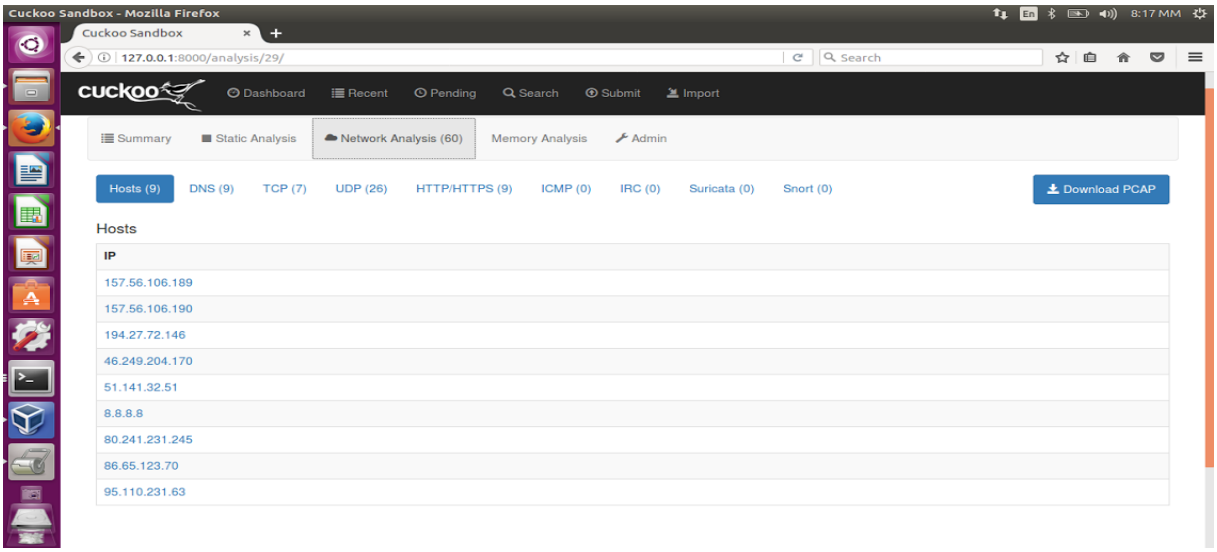
Εικόνα 3.3:Υποβολή αρχείου προς εξέταση

Ακολούθως το Cuckoo εκτελεί το υπό εξέταση αρχείο στο εικονικό μηχάνημα το οποίο είναι συνδεδεμένο με την εφαρμογή. Όπως φαίνεται και στην ακόλουθη εικόνα στην επιφάνεια εργασίας του εικονικού μηχανήματος δημιουργούνται διάφορα αρχεία, αποτέλεσμα της εκτέλεσης του κακόβουλου λογισμικού.



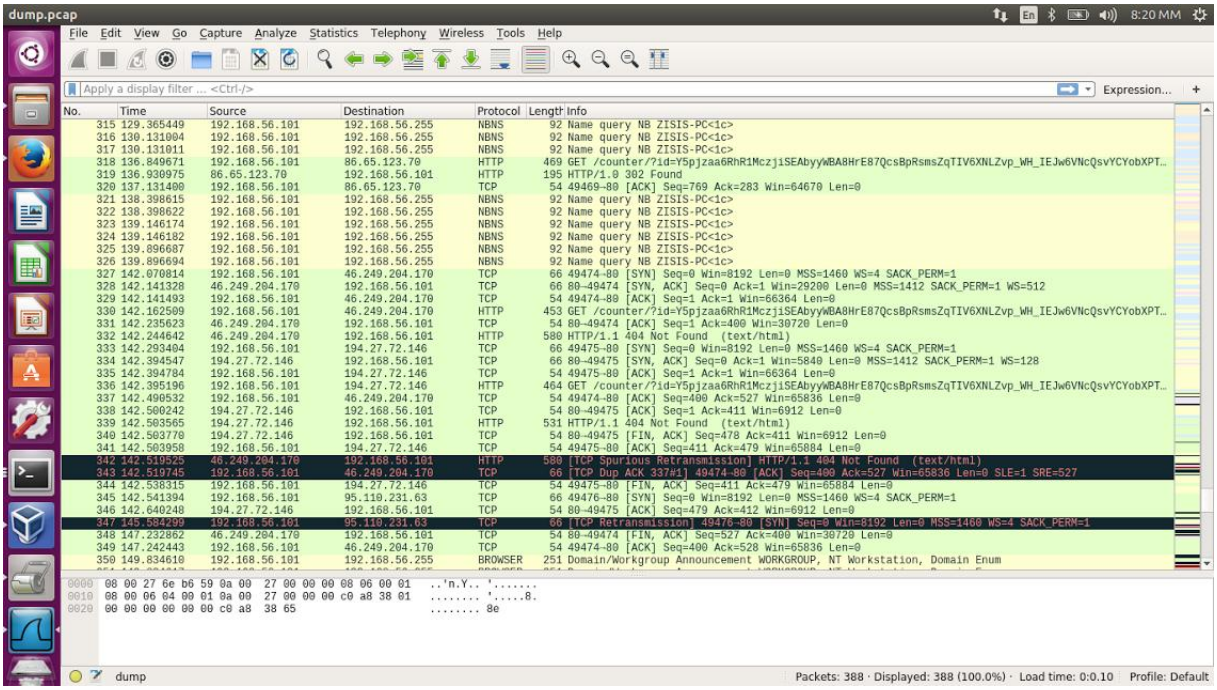
Εικόνα 3.4: Εκτέλεση κακόβουλου λογισμικού σε εικονικό μηχάνημα

Μόλις ολοκληρωθεί η εκτέλεση παράγεται μια αναφορά με τα αποτελέσματα που συνέλεξε η εφαρμογή. Παρακάτω φαίνονται οι διευθύνσεις που συνέλεξε η εφαρμογή, από την ανάλυση του δικτύου.



Εικόνα 3.5: Αναφορά Cuckoo

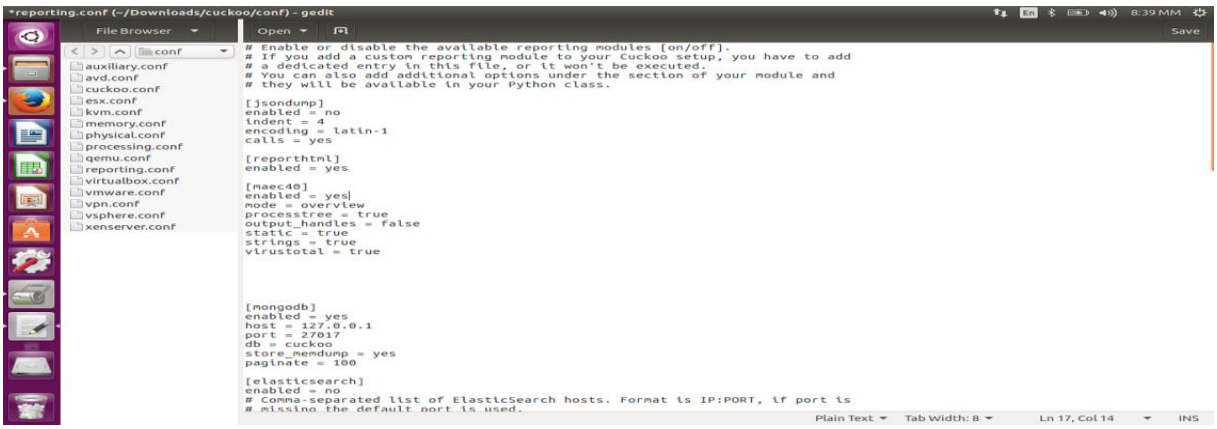
Μπορεί να χρησιμοποιηθεί το αρχείο της κίνησης δικτύου που παρήγαγε το Cuckoo και μέσω μια εφαρμογής ανάλυσης πακέτων να παραχθούν εκτενέστερα αποτελέσματα. Χρησιμοποιήθηκε το WireShark, στο οποίο εισήχθη το αρχείο dump.pcap που δημιούργησε το Cuckoo.



Εικόνα 3.6: Αναφορά Wireshark

Όπως φαίνεται, το κακόβουλο λογισμικό κατά την εκτέλεση του προσπαθεί να επικοινωνήσει με την διεύθυνση 46.249.204.170, η οποία βρίσκεται εξωτερικά του δικτύου.

Επίσης το Cuckoo δίνει την δυνατότητα επιλογής της μορφής που θα έχει η αναφορά που θα εξάγει, μετά το τέλος της ανάλυσης. Για παράδειγμα μπορεί να εξαχθεί κάποια HTML αναφορά είτε κάποια JSON αναφορά. Στις προηγούμενες εκδόσεις του Cuckoo, δινόταν η επιλογή εξαγωγής και μιας αναφοράς τύπου MAEC. Η επιλογή της αντίστοιχης αναφοράς γίνεται μέσω του αρχείου reporting.conf που βρίσκεται στον φάκελο conf του Cuckoo. Για την ενεργοποίηση του κάθε τύπου αναφοράς θα πρέπει το πεδίο enabled του συγκεκριμένου τύπου αναφοράς να πάρει τιμή YES.



Εικόνα 3.7 : Επιλογή τύπου αναφορών

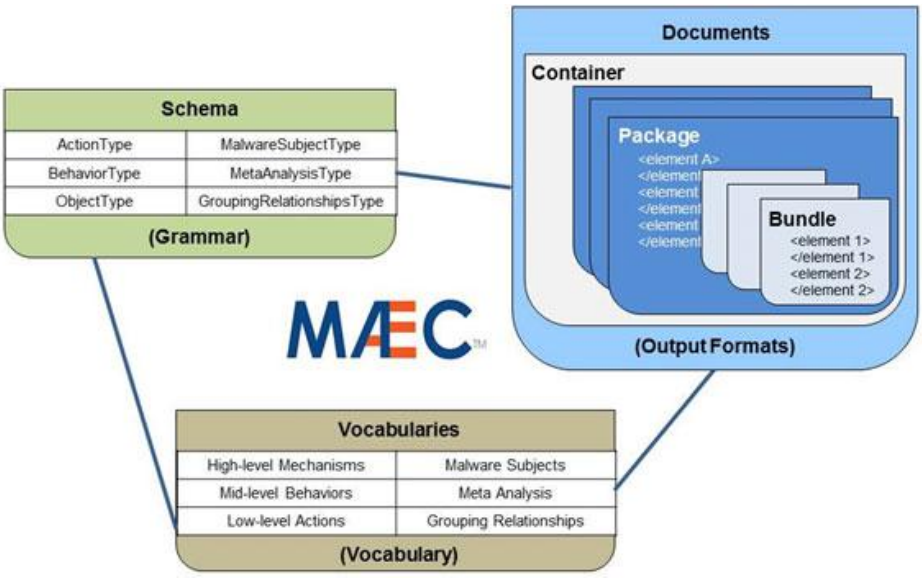
Κεφάλαιο 4

Γλώσσες Χαρακτηρισμού Κακόβουλου Λογισμικού

Σε αυτή την ενότητα παρουσιάζονται οι γλώσσες χαρακτηρισμού κακόβουλου λογισμικού MAEC και STIX καθώς και η σχέση που υπάρχει ανάμεσα τους.

4.1 MAEC

Το MAEC[1] (Malware Attribute Enumeration and Characterization) αποτελεί μια γλώσσα για την κωδικοποίηση και μετάδοση εμπίστων πληροφοριών σχετικά με κακόβουλα λογισμικά, βασιζόμενο σε χαρακτηριστικά όπως οι συμπεριφορές και οι μέθοδοι επίθεσης. Στόχος του είναι βελτιστοποίηση της επικοινωνίας ανάμεσα σε ανθρώπους σχετικά με κακόβουλα λογισμικά, όπως και αυτής ανάμεσα στα διάφορα εργαλεία ανάλυσης κακόβουλων λογισμικών. Το MAEC προσφέρει ένα τυποποιημένο λεξιλόγιο σχετικά με πληροφορίες κακόβουλων λογισμικών.



Εικόνα 4.1: Αρχιτεκτονική MAEC(Desiree Beck & Ivan Kirillov 2014)

[1] Ενότητα 3.2 Ανάλυση Κακόβουλου Λογισμικού

4.1.1 Λεξιλόγιο

Τα λεξιλόγια του MAEC αποτελούν ένα σύνολο προκαθορισμένων λέξεων τα οποία μπορούν να χρησιμοποιηθούν σε κάποιο MAEC μήνυμα. Επίσης εκτός από τα προκαθορισμένα λεξιλόγια MAEC, δίνεται η δυνατότητα στους χρήστες να ορίσουν τα δικά τους λεξιλόγια τα οποία θα πρέπει να συμφωνούν με τις αρχές του MAEC.

```
<maecBundle:Action id="maec-example-act-1">
<cybox:Name xsi:type="maecVocabs:NetworkActionNameVocab-1.0">download
file</cybox:Name></maecBundle:Action>
```

Πίνακας 4.1: Παράδειγμα χρήσης προκαθορισμένου λεξιλογίου MAEC(Desiree Beck & Ivan Kirillov 2013)

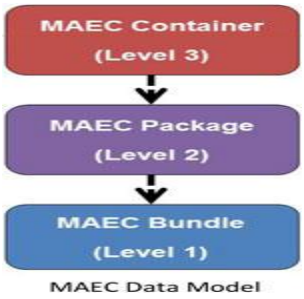
4.1.2 Πρότυπα

Η γλώσσα MAEC ορίζεται μέσα από ένα μοντέλο δεδομένων τριών επιπέδων, καθένα από τα οποία υλοποιείται μέσα στο δικό του XML σχήμα. Το πρότυπο MAEC BUNDLE αποτελεί το μοντέλο δεδομένων επιπέδου ένα, το πρότυπο MAEC PACKAGE το μοντέλο δεδομένων επιπέδου δύο και το πρότυπο MAEC CONTAINER το μοντέλο δεδομένων επιπέδου τρία. Υπάρχει αυτονομία ανάμεσα στα τρία μοντέλα έτσι ώστε καθένα από αυτά να μπορεί να χρησιμοποιηθεί χωρίς κάποιο από τα μοντέλα δεδομένων υψηλότερου επιπέδου.

MAEC BUNDLE: Το πρότυπο MAEC BUNDLE καταγράφει όλα τα χαρακτηριστικά που προκύπτουν από την ανάλυση μιας μεμονωμένης περίπτωσης κακόβουλου λογισμικού, συμπεριλαμβανομένων των συμπεριφορών, δράσεων και αντικειμένων τύπου MAEC.

MAEC PACKAGE: Το πρότυπο MAEC PACKAGE καταγράφει ένα ή περισσότερα αντικείμενα κακόβουλου λογισμικού, μαζί με την σχέση ομαδοποίησης ανάμεσα σε αυτά τα λογισμικά εάν αυτή υπάρχει. Για παράδειγμα κακόβουλα λογισμικά με παρόμοιες συμπεριφορές.

MAEC CONTAINER: Το πρότυπο MAEC CONTAINER αποτελεί μια δεξαμενή για όλες τις μορφές δεδομένων MAEC. Χρησιμοποιείται σαν μηχανισμός μεταφοράς πακέτων MAEC.



Εικόνα 4.2: Μοντέλο δεδομένων MAEC(Desiree Beck & Ivan Kirillov 2013)

4.2 STIX

Το STIX (Structured Thread Information Expression) είναι μια γλώσσα που χρησιμοποιείται για την περιγραφή, τον χαρακτηρισμό και τον διαμοιρασμό πληροφοριών από απειλές κυβερνοχώρου[2].

4.2.1 Αρχιτεκτονική

Στην παρούσα ενότητα παρουσιάζονται οι βασικές έννοιες του STIX και ο τρόπος συσχέτισης τους.

Για την περιγραφή των απειλών, το STIX χρησιμοποίει τις παρακάτω έννοιες:

- **Observables**: Αποτελούν την βασική έννοια της αρχιτεκτονικής STIX. Αποτελούν μετρήσιμα γεγονότα που σχετίζονται με την λειτουργία υπολογιστών και δικτύων. Πληροφορίες όπως το όνομα ενός αρχείου, το μέγεθος του ή τιμή ενός registry key είναι observables.

- **Indicators**: Οι δείκτες χρησιμοποιούνται για την μετάδοση παρατηρούμενων (observables), σε συνδυασμό με πληροφορίες ενδιαφέροντος, σχετικά με την ασφάλεια κυβερνοχώρου.

- **Incident**: Τα περιστατικά περιγράφουν περιπτώσεις εχθρικών πράξεων. Περιέχουν πληροφορίες όπως χρόνο, εμπλεκόμενα μέρη, επηρεασθέντα περιουσιακά στοιχεία και φορείς απειλής.

[2] Κεφάλαιο 1

- **Tactics Techniques Procedures**: Περιγράφουν πρότυπα επίθεσης, κακόβουλα λογισμικά, δομή εργαλείων για επιθέσεις, στόχευση θυμάτων και άλλες μεθόδους που μπορούν να χρησιμοποιηθούν για την εκμετάλλευση κάποιας ευπάθειας (Exploit Target).

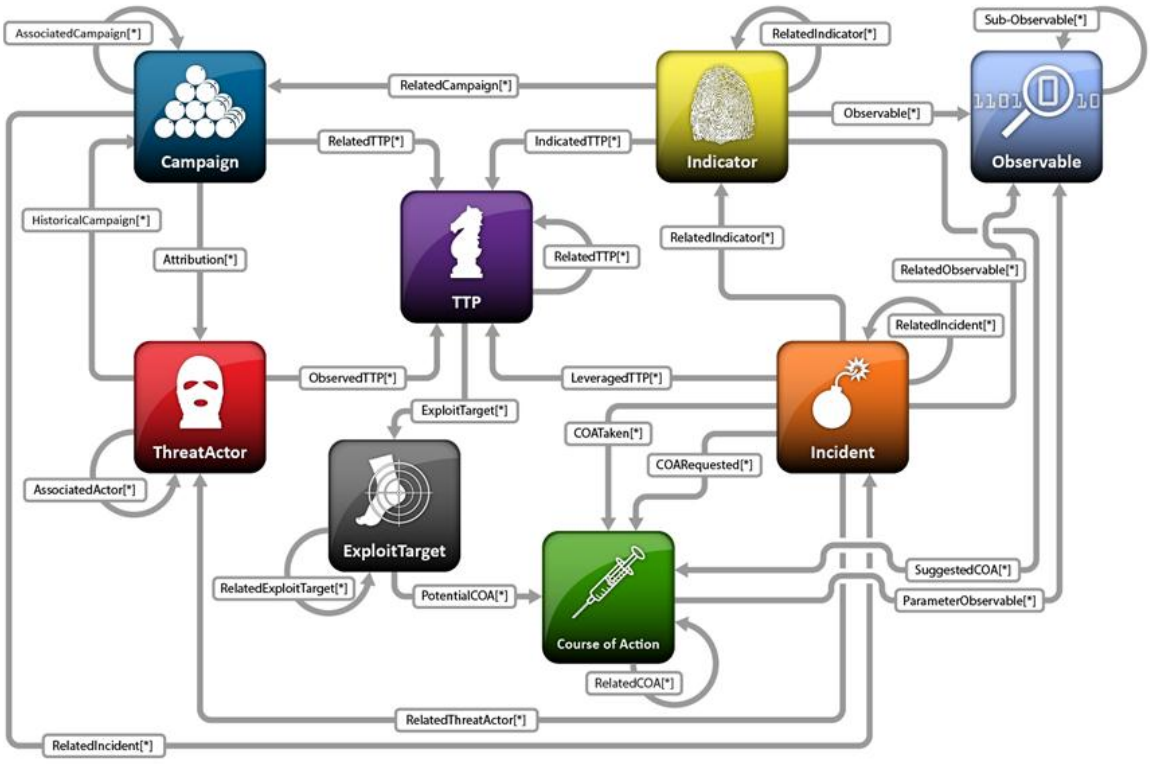
- **Exploit Target**: Περιγράφει αδυναμίες τις οποίες μπορεί να εκμεταλλευτεί κάποιος φορέας απειλής(Threat Actor) μέσω χρήσης των κατάλληλων τεχνικών (TTP).

- **Course of Action**: Περιγράφει ενέργειες που μπορούν να γίνουν ως αντίμετρα σε μια επίθεση ή σαν μέτρα πρόληψης,

- **Campaigns**: Περιγράφουν μια σειρά γεγονότων με σκοπό την εκμετάλλευση κάποιας ευπάθειας. Ένα περιστατικό (incident), μπορεί να αποτελεί μέρος μιας εκστρατείας.

- **Threat Actor**: Περιγράφει την ταυτότητα ενός εχθρού, ο οποίος μέσω τεχνικών (TTP) προσπαθεί να εκμεταλλευτεί κάποια ευπάθεια.

Στην παρακάτω εικόνα φαίνεται ο τρόπος συσχέτισης των εννοιών που παρουσιάστηκαν.



Εικόνα 4.3: Αρχιτεκτονική STIX(Barnum 2014)

4.3 Συνδυασμός MAEC-STIX

Σε αυτή την ενότητα παρουσιάζονται οι τύποι δεδομένων που μπορούν να μεταδοθούν με κάθε μια από της δύο γλώσσες, καθώς και τα πλεονεκτήματα που προσφέρει ο συνδυασμός τους.

Στον παρακάτω πίνακα παρουσιάζονται οι τύποι πληροφοριών που μπορούμε να μεταδοθούν με κάθε μια γλώσσα.

MAEC	**STIX**
• Αποτελέσματα στατικής ανάλυσης. • Αποτελέσματα δυναμικής ανάλυσης. • Μεταδεδομένα αναλύσεων όπως εργαλεία που χρησιμοποιήθηκαν, πληροφορίες σχετικές με τον αναλυτή, σχόλια που καταγράφηκαν κατά την διάρκεια της ανάλυσης. • Συσχετίσεις ανάμεσα σε διαφορετικά δείγματα που έχουν αναλυθεί.	• Τακτικές, τεχνικές και διαδικασίες (TTP) οι οποίες προσφέρουν μια συνοπτική περιγραφή του δείγματος κακόβουλου λογισμικού. • Περιστατικά που συνέβησαν κατά την εκτέλεση ενός δείγματος κακόβουλου λογισμικού (Incidents). • Εκστρατείες που σχετίζονται με κάποιο δείγμα κακόβουλου λογισμικού (Campaigns). • Χρήστες ενός κακόβουλου δείγματος (Threat Actors) • Δείκτες που αφορούν κάποιο δείγμα κακόβουλου λογισμικού (Indicators).

Πίνακας 4.2: Μεταδιδόμενες Πληροφορίες MAEC/STIX

Εμφωλεύοντας δεδομένα MAEC μέσα σε ένα αρχείο STIX μπορούν να μεταδοθούν:

- Εκτενείς πληροφορίες σχετικά με τις τακτικές, τις τεχνικές και τις διαδικασίες που χρησιμοποιεί το δείγμα κακόβουλου λογισμικού.

- Πληροφορίες συσχετίσεων ανάμεσα σε φορεία απειλής(threat actors) και δείγματα κακόβουλου λογισμικού.

- Συσχέτιση μεταδεδομένων κακόβουλων λογισμικών με φορείς απειλής.

Στην παρακάτω εικόνα αποτυπώνονται οι τύποι πληροφοριών που μπορούν να μεταδοθούν με την κάθε γλώσσα ξεχωριστά, καθώς και με τον συνδυασμό τους.

MAEC	STIX	STIX + MAEC
Captures structured, detailed malware information: • Capabilities • Behaviors • Actions • AV Classifications • Extracted Objects • Relationships • Associated Metadata	**Captures unstructured, basic malware information:** • Type • Name • Description	**Captures broad spectrum of malware information:** • Basic, descriptive information via STIX o Provides Identification • Detailed, structured information via MAEC o Provides broader understanding • E.g., a brief description of a malware family and detailed descriptions of several of its members
Provides analytical context • "What" does the malware do? • "How" does the malware operate?	**Provides surrounding context** • "Who" used the malware? • "Where" was the malware used?	**Provides surrounding AND analytical context** • Connects detailed malware information to broader threat context • E.g., "what" specific features of a malware instance are associated with a particular threat actor?
Target audience: • Malware Analysts/Reverse Engineers	**Target audience:** • Cyber Threat/Intelligence Analysts • SOC/CERT Operators • Incident Responders	**Target audience:** • Malware Analysts/Reverse Engineers • Cyber Threat/Intelligence Analysts • SOC/CERT Operators • Incident Responders

Εικόνα 4.4: Δυνατότητες συλλογής κακόβουλων πληροφοριών(Kirillov 2014)

Κεφάλαιο 5
Υλοποίηση

Σε αυτό το κεφάλαιο παρουσιάζεται η μεθοδολογία που ακολουθήθηκε για την υλοποίηση του προτύπου TAXII. Στις ακόλουθες ενότητες αναλύεται η διαδικασία μετατροπής αναφορών MAEC σε STIX, καθώς και ο τρόπος ανταλλαγής μηνυμάτων TAXII κατά την διάρκεια μιας επικοινωνίας τύπου πελάτη-διακομιστή (Client-Server).

5.1 Αρχιτεκτονική συστήματος

Στόχος του συστήματος που δημιουργήθηκε είναι ο διαμοιρασμός των αναφορών του Cuckoo Sandbox στους χρήστες του συστήματος, με σκοπό την ενημέρωση τους, σχετικά με διάφορα κακόβουλα λογισμικά.

Ο σχεδιασμός του συστήματος αποτελείται από τα παρακάτω βήματα:

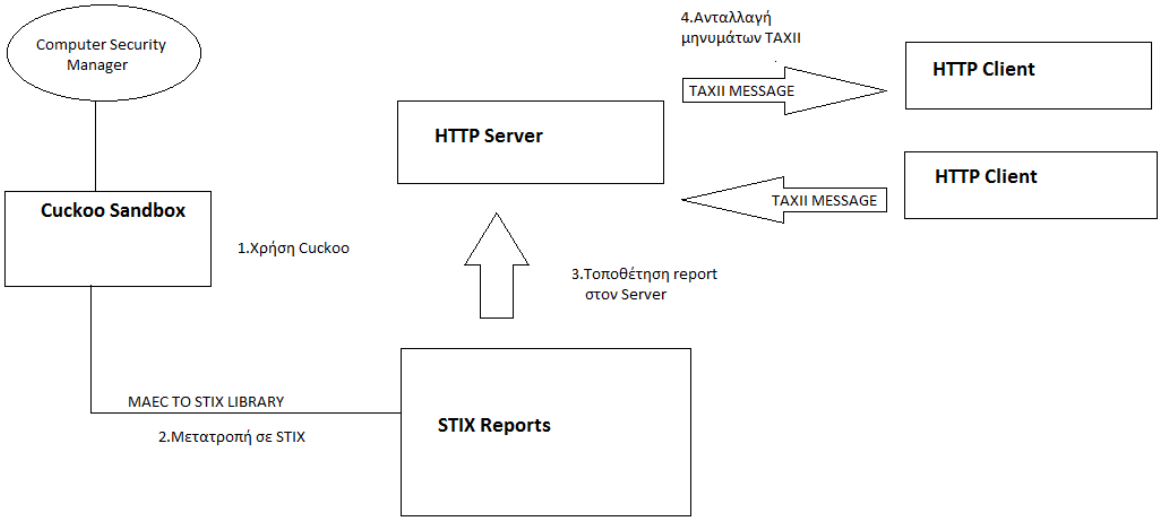
1. Ο υπεύθυνος ασφαλείας του συστήματος χρησιμοποιεί το Cuckoo Sandbox και εξάγει μια αναφορά MAEC.

2. Μετατρέπει την αναφορά MAEC σε STIX έτσι ώστε να μπορέσει να την μεταδώσει μέσω κάποιου μηνύματος TAXII.

3. Τοποθετεί το έγγραφο STIX στον Server του συστήματος, έτσι ώστε ο διαχειριστής του Server να μπορεί να το εισάγει σε κάποια συλλογή δεδομένων TAXII.

4. Οι χρήστες του συστήματος επικοινωνούν με τον Server και ανταλλάσουν μηνύματα TAXII.

Στην παρακάτω εικόνα παρουσιάζονται τα βήματα υλοποίησης του συστήματος ανταλλαγής μηνυμάτων TAXII.



Εικόνα 5.1: Αρχιτεκτονική συστήματος

5.2 Μετατροπή αναφοράς MAEC σε STIX

Η διαδικασία εξαγωγής μιας αναφοράς μέσω του Cuckoo περιγράφηκε σε προηγούμενο κεφάλαιο[3]. Σε αυτή την ενότητα παρουσιάζεται ο τρόπος μετατροπής μιας αναφοράς MAEC σε STIX, έτσι ώστε να χρησιμοποιηθεί σε ένα TAXII μήνυμα.

Για την μετατροπή χρησιμοποιήθηκε η βιβλιοθήκη της python maec-to-stix. Η βιβλιοθήκη maec-to-stix δίνει την δυνατότητα δημιουργίας μιας αναφοράς STIX από την αντίστοιχη αναφορά MAEC. Ποιο συγκεκριμένα μπορεί να εμφωλευθεί το περιεχόμενο MAEC σε ένα αρχείο STIX, ή να εξαχθούν δείκτες STIX από δεδομένα τύπου MAEC.

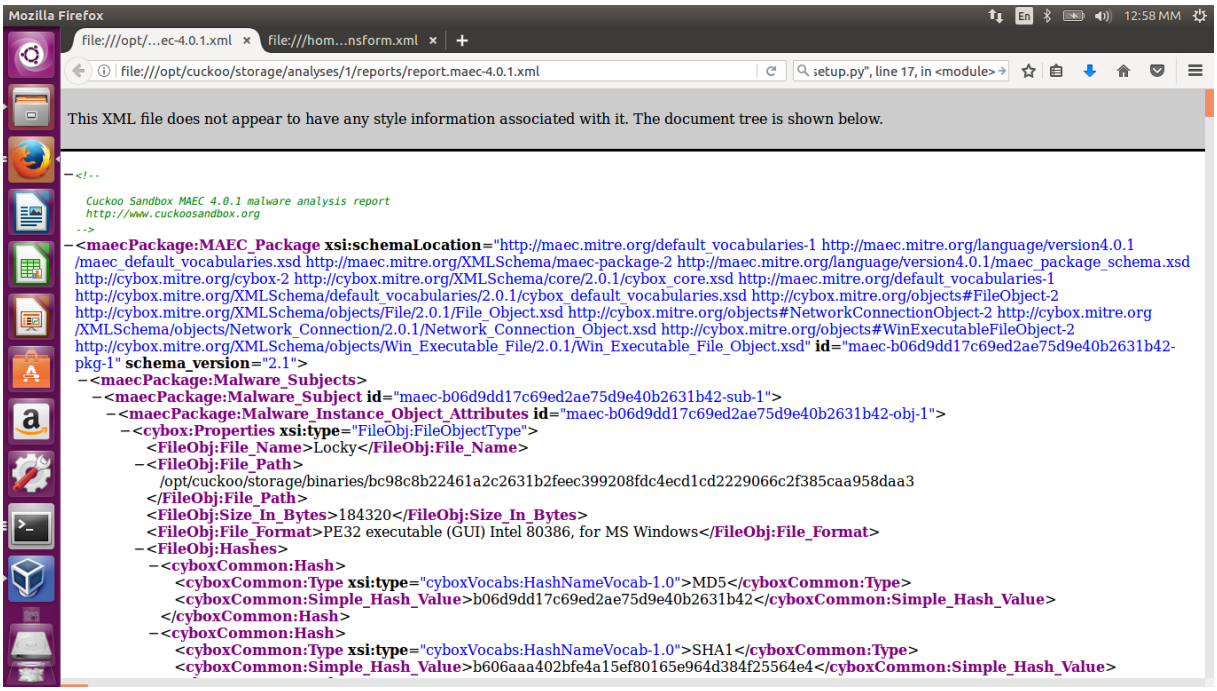
Η συγκεκριμένη βιβλιοθήκη αποτελείται από τα ακόλουθα script :

- **Maec_wrap**: Παίρνει ως όρισμα το αρχείο MAEC προς μετατροπή και δίνει ως έξοδο, ένα αρχείο STIX μέσα στο οποίο εμφωλεύεται το αρχείο MAEC.

[3] 3.2 Ανάλυση κακόβουλου λογισμικού

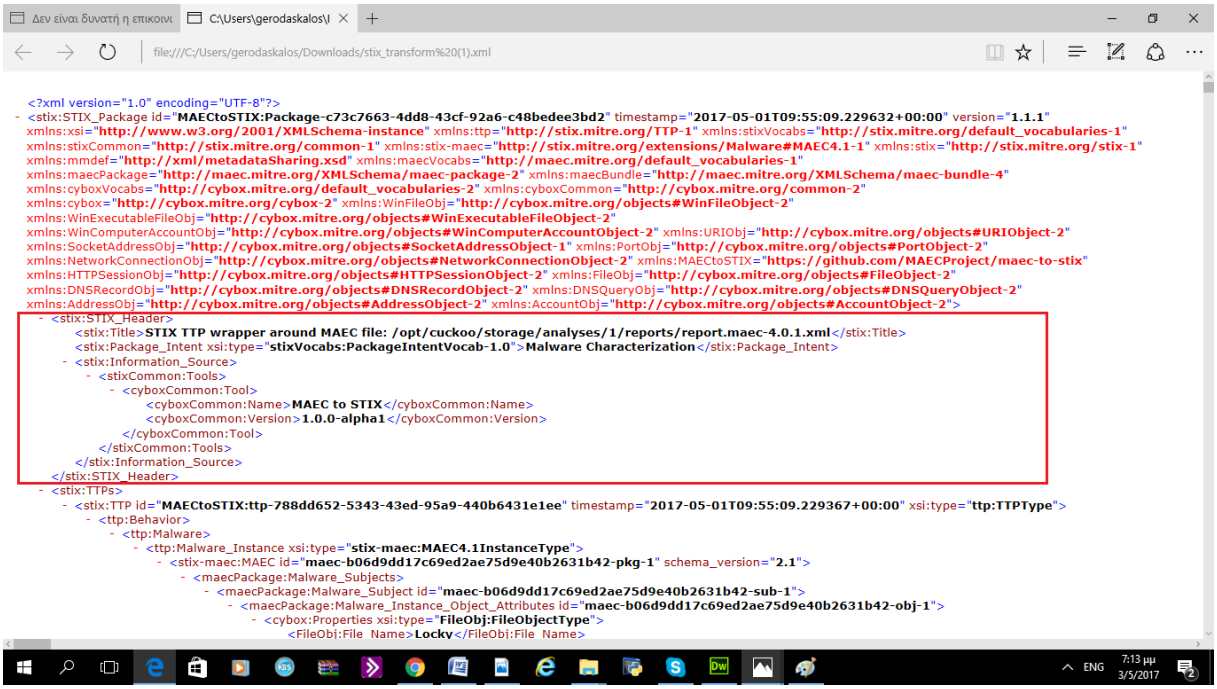
- **Maec_extract_indicators:** Παίρνει ως όρισμα ένα αρχείο MAEC και προσπαθεί να εξάγει δείκτες STIX, χρησιμοποιώντας τα αρχεία παραμετροποίησης JSON που παρέχει η βιβλιοθήκη.

Στην παρακάτω εικόνα φαίνεται η μορφή μιας αναφοράς MAEC.



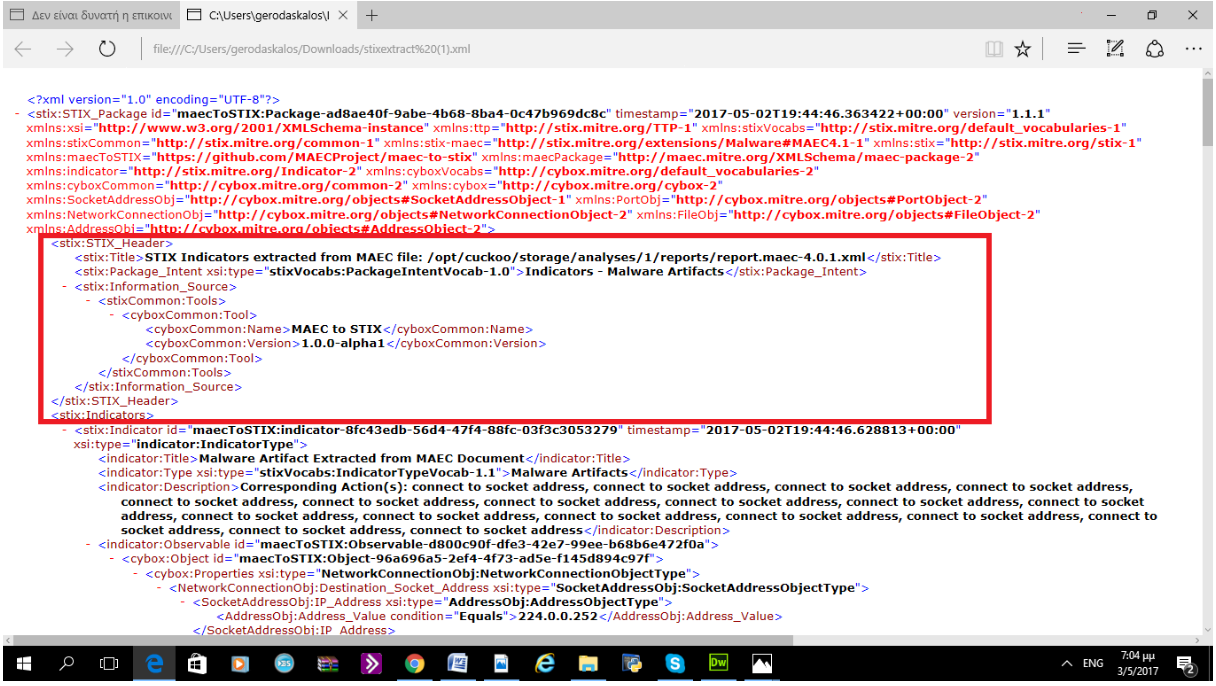
Εικόνα 5.2: MAEC report

Ακολουθεί η εικόνα μιας αναφοράς MAEC εμφωλευμένης σε STIX, με χρήση του Maec_wrap.py.



Εικόνα 5.3: MAEC wrapped in STIX

Τέλος στην παρακάτω εικόνα φαίνεται η εξαγωγή των δεικτών STIX από μια αναφορά MAEC, με χρήση του Maec_extract_indicators.py.



Εικόνα 5.4: MAEC report extracted Indicators

5.3 HTTP Server

Το επόμενο βήμα μετά την μετατροπή μιας αναφοράς MAEC σε STIX, είναι η δημιουργία ενός HTTP Server μέσω του οποίου θα υλοποιείται η διαδικασία ανταλλαγής μηνυμάτων.

Ένας HTTP Server είναι ένα υπολογιστικό σύστημα το οποίο διαχειρίζεται αιτήματα που μεταφέρονται μέσω του HTTP (HyperText Transfer Protocol). Το πρωτόκολλο μεταφοράς υπερκειμένου (HTTP) έχει σχεδιαστεί για να υποστηρίζει επικοινωνίες ανάμεσα σε Clients και Servers(Anon n.d.). Λειτουργεί σαν ένα πρωτόκολλο αίτησης – απάντησης (request-response). Για παράδειγμα κάποιος Client μπορεί να υποβάλει ένα αίτημα μέσω του πρωτοκόλλου μεταφοράς υπερκειμένου (HTTP request) σε κάποιον Server για χρήση κάποιας υπηρεσίας του Server (μια υπηρεσία TAXII π.χ.). Ο Server με την σειρά του απαντάει στο αίτημα του Client (HTTP response).

Για την υλοποίηση του HTTP Server χρησιμοποιήθηκε η βιβλιοθήκη της Python http. Δημιουργήθηκε η κλάση TAXIIHTTPRequestHandler που κληρονομεί την κλάση BaseHTTPRequestHandler, η οποία είναι υπεύθυνη για την διαχείριση των διαφόρων αιτημάτων που φτάνουν στον Server.

class TAXIIHTTPRequestHandler(http.server.BaseHTTPRequestHandler)

5.3.1 Η μέθοδος RUN

Η μέθοδος run είναι υπεύθυνη για την δημιουργία και εκκίνηση του Server. Δέχεται ως ορίσματα τις κλάσεις HTTPServer και BASEHTTPRequestHandler, οι οποίες είναι υπεύθυνες για την δημιουργία ενός HTTP Server καθώς και για την διαχείριση των διαφόρων αιτημάτων. Επίσης μέσα στην μέθοδο ορίζεται η διεύθυνση αλλά και η θύρα την οποία έχει διαθέσει ο Server, για την υποβολή αιτημάτων από τους Clients, όπως επίσης και το όνομα της κλάσης που είναι υπεύθυνη για την διαχείριση των αιτημάτων. Τέλος με την μέθοδο serve_forever() εξασφαλίζεται η αδιάκοπη λειτουργία του Server.

```
def run
(server_class=http.server.HTTPServer,handler_class=http.server.BaseHTTPRequestHandler):
print('http server is starting...')
# ip and port of server
server_address = ('192.168.1.68', 80)
httpd = server_class(server_address, TAXIIHTTPRequestHandler)
print('http server is running...')
httpd.serve_forever()
```

Πίνακας 5.2: Υλοποίηση μέθοδου run

5.3.2 Μέθοδοι Αίτησης - Απόκρισης

Οι δύο κύριες μέθοδοι που χρησιμοποιούνται σε μια επικοινωνία τύπου "request-response"ανάμεσα σε έναν Server και έναν Client είναι οι GET και POST. Η μέθοδος GET αιτείται την αναπαράσταση μιας συγκεκριμένης πηγής. Η μέθοδος POST υποβάλει δεδομένα προς επεξεργασία σε μια συγκεκριμένη πηγή. Η μέθοδος GET θεωρείται μια "ασφαλής μέθοδος" , καθώς εστιάζει στην απόκτηση πληροφοριών χωρίς να προκαλεί αλλαγές στην κατάσταση του

Server. Σε αντίθεση η μέθοδος Post θεωρείται μη ασφαλής καθώς μέσω των ενεργειών (actions) που περιλαμβάνει ενδέχεται να επιδράσει στην κατάσταση του Server.

5.3.2.1 Η μέθοδος do_GET

Μέσω της μεθόδου do_GET ο Server δέχεται τα διάφορα αιτήματα του Client. Δημιουργήθηκε μια μεταβλητή length, στην οποία ανατέθηκε η τιμή της κεφαλίδας 'Content-length'. Η μεταβλητή length υποδεικνύει το μέγεθος της εισερχόμενης ροής δεδομένων προς τον Server. Μέσω της μεταβλητής rfile της κλάσης BaseHTTPRequestHandler, δημιουργήθηκε μια εισερχόμενη ροή δεδομένων.

```
def do_GET(self):
    length=self.headers['content-length']
    data = self.rfile.read(int(length)).decode()
    return data
```

Πίνακας 5.3 : Υλοποίηση μεθόδου do_GET

5.3.2.2 Η μέθοδος do_POST

Μέσω της μεθόδου do_POST ο Server απαντάει στα εισερχόμενα αιτήματα. Στέλνει τον κωδικό κατάστασης 200 υποδεικνύοντας ότι το εισερχόμενο αίτημα ολοκληρώθηκε. Στην συνέχεια στέλνει τις κεφαλίδες 'Content-type', 'text-html' και ακολούθως δείχνει το τέλος των κεφαλίδων μέσω της μεθόδου end_headers(). Στην περίπτωση που ο Server δεν καταφέρει να βρει αυτό που του ζητήθηκε από τον Client, απαντάει με το κωδικό κατάστασης 404.

```
def do_POST(self):
   try:
      self.send_response(200)
       self.send_header('Content-type', 'text-html')
       self.end_headers()
       data=TAXIIHTTPRequestHandler.do_GET(self)
```

```
        print(data)
        header=TAXIIHTTPRequestHandler.getheader(data)

       if header=="discovery":
          TAXIIHTTPRequestHandler.disc_resp(self, data)
        elif header=="feed_info":
          TAXIIHTTPRequestHandler.feed_info(self,data)
        elif header=="poll_request":
          TAXIIHTTPRequestHandler.poll_Resp(self,data)
        elif header=="manage_feed":
          TAXIIHTTPRequestHandler.manfeed_subresp(self,data)
        elif header=="Inbox":
          TAXIIHTTPRequestHandler.Inbox_resp(self,data)
        else:
          return
      except IOError:
        self.send_error(404, 'file not found')
```

Πίνακας 5.4: Υλοποίησης μεθόδου do_POST

5.4 HTTP Client

Ο HTTP Client είναι υπεύθυνος για την αποστολή των διαφόρων αιτημάτων στον Server, με σκοπό την απόκτηση των υπηρεσιών TAXII.

Δημιουργία HTTP CLIENT

Χρησιμοποιήθηκε η κλάση HttpClient η οποία παρέχεται μέσω του module clients της βιβλιοθήκης libtaxii για την δημιουργίας του HTTP Client, μέσω του οποίου θα αποστέλλονται τα αιτήματα στον Server.

client=libtaxii.clients.HTTPClient()

5.5 Αιτήματα TAXII

Τα αιτήματα TAXII χρησιμοποιούνται με στόχο την απόκτηση υπηρεσιών TAXII. Σε αυτήν την ενότητα παρουσιάζονται τα αιτήματα TAXII και ο τρόπος υλοποίησης τους.

5.5.1 Discovery Request

Μέσω ενός αιτήματος τύπου Discovery Request, ο HTTP Client επιθυμεί να γίνει γνώστης των διαθέσιμων προς αυτόν υπηρεσιών TAXII. Για την υλοποίηση χρησιμοποιήθηκε το module messages_10 της βιβλιοθήκης libtaxii, δημιουργώντας ένα αντικείμενο τύπου DiscoveryRequest.

discovery_request=libtaxii.messages_10.DiscoveryRequest()

Ως ορίσματα χρησιμοποιήθηκαν το αναγνωριστικό μηνύματος (message_id), το οποίο δημιουργήθηκε μέσω της μεθόδου generate_message_id(), καθώς και ένα ζευγάρι επεκτάσεων επικεφαλίδων(extended headers). Το αναγνωριστικό μηνύματος είναι υποχρεωτικό σε κάθε μορφή αιτήματος TAXII, ενώ αντίθετα οι επεκτάσεις επικεφαλίδων είναι προαιρετικές.

```
def disc_Req():
  print("Requesting available services")

  discovery_request=tm10.DiscoveryRequest(message_id=tm10.generate_message_id()
                                          ,extended_headers={'service':'discovery'})
  rsp= client.call_taxii_service('192.168.1.68',"", VID_TAXII_XML_10,discovery_request.to_xml() ,
port=80 )
    data_received = rsp.read().decode()
    print(data_received)
```

Πίνακας 5.5: Υλοποίηση μεθόδου disc_Req

5.5.2 Feed Information Request

Ένα αίτημα τύπου Feed Information Request χρησιμοποιείται από τον HTTP Client για να αιτηθεί πληροφορίες σχετικά με διαθέσιμες συλλογές δεδομένων. Όπως και στην περίπτωση του

Discovery request χρησιμοποιήθηκε το module messages_10 της βιβλιοθήκης libtaxii. Δημιουργήθηκε ένα αντικείμενο τύπου FeedInformationRequest.

feed_information_request = *libtaxii.messages_10*.FeedInformationRequest()

Τα ορίσματα που χρησιμοποιούνται όπως και στα αιτήματα τύπου Discovery Request είναι το αναγνωριστικό μηνύματος και ένα ζευγάρι επεκτάσεων επικεφαλίδων.

```
def feed_InfoReq():
  print("Requesting feed informations")
  feed_information_request=tm10.FeedInformationRequest
  (message_id=tm10.generate_message_i     d(),extended_headers={'service':'feed_info'})
  rsp=client.call_taxii_service
('192.168.1.68',"",VID_TAXII_XML_10,feed_information_request.to_xml(),port=80 )

  data_received = rsp.read().decode()
  print(data_received)
```

Πίνακας 5.6: Υλοποίηση μεθόδου feed_InfoReq

5.5.3 Poll Request

Ένα αίτημα τύπου Poll Request χρησιμοποιείται από τον HTTP Client, για να αιτηθεί περιεχόμενο κάποιας συλλογής δεδομένων από τον HTTP Server. Χρησιμοποιήθηκε το module messages_10 της βιβλιοθήκης libtaxii και δημιουργήθηκε ένα αντικείμενο τύπου PollRequest.

poll_request= libtaxii.messages_10.PollRequest()

Ως ορίσματα χρησιμοποιήθηκαν το αναγνωριστικό μηνύματος (message_id) και το όνομα της συλλογής δεδομένων (feed_name), της οποίας περιεχόμενο επιθυμεί να αποκτήσει ο πελάτης. Οι δύο αυτοί τύποι ορισμάτων είναι υποχρεωτικοί. Επίσης χρησιμοποιήθηκε ένα ζευγάρι επεκτάσεων επικεφαλίδων (extended headers), ένα αναγνωριστικό εγγραφής (subscription_id), η μορφή των αιτούμενων δεδομένων (content_bindings) και δύο χρονοσφραγίδες (exclusive_begin_timestamp_label, inclusive_end_timestamp_label).

```
def poll_Req():
  print("Requesting a poll")
  poll_request=tm10.PollRequest(message_id=tm10.generate_message_id(),
                                feed_name='FeedPol',
                                exclusive_begin_timestamp_label=datetime.datetime.now(tzutc()),
```

```
        inclusive_end_timestamp_label=datetime.datetime.now(tzutc()),
        subscription_id='SubsId002'
       content_bindings=[CB_STIX_XML_10],
       extended_headers={'service':'poll_request'})
  rsp=client.callTaxiiService('192.168.1.68',"",VID_TAXII_XML_10,poll_request.to_xml(),port=80)
  data_received = rsp.read().decode()
  print(data_received)
```

Πίνακας 5.7: Υλοποίηση μεθόδου poll_Req

5.5.4 Manage Feed Subscription Request

Μέσω αυτής της μορφής αιτήματος, ο πελάτης μπορεί να αιτηθεί την εγγραφή του σε κάποια συλλογή δεδομένων ή την διαγραφή από κάποια συλλογή στην οποία είναι ήδη εγγεγραμμένος. Χρησιμοποιήθηκε το module messages_10 της βιβλιοθήκης libtaxii και δημιουργήθηκε ένα αντικείμενο τύπου ManageFeedSubscriptionRequest.

manage_feed_subscription_request1 = libtaxii.messages_10.ManageFeedSubscriptionRequest()

Στην περίπτωση της εγγραφής σε κάποια συλλογή δεδομένων τα υποχρεωτικά ορίσματα που χρησιμοποιούνται είναι το αναγνωριστικό μηνύματος (message_id), το όνομα της συλλογής (feed_name), στην οποία επιθυμεί να εγγραφεί ο πελάτης καθώς και η ενέργεια που επιθυμεί να πραγματοποιήσει (action). Στην περίπτωση διαγραφής από κάποια συλλογή δεδομένων πέραν των προαναφερθέντων ορισμάτων θα πρέπει να οριστεί και το αναγνωριστικό της εγγραφής (subscription_id), από την οποία επιθυμεί να διαγραφεί ο πελάτης. Επιπλέον ορίσματα που μπορούν να χρησιμοποιηθούν είναι επεκτάσεις επικεφαλίδων (extended_headers) καθώς και οι παράμετροι παράδοσης(delivery_parameters). Οι παράμετροι παράδοσης μπορούν να δημιουργηθούν με αντικείμενα τύπου *libtaxii*.messages_10. DeliveryParameters().

```
def man_Feed_Subreq():
  print("Requesting manage feed")
  delivery_parameters1 = tm10.DeliveryParameters(
    inbox_protocol=VID_TAXII_HTTP_10,
    inbox_address='http://example.com/inbox',
    delivery_message_binding=VID_TAXII_XML_10,
    content_bindings=[CB_STIX_XML_10])

  manage_feed_subscription_request1 = tm10.ManageFeedSubscriptionRequest(
    message_id=tm10.generate_message_id(),
```

```
    feed_name='Feed',
    action=ACT_UNSUBSCRIBE,
    subscription_id='SubsId056',
    delivery_parameters=delivery_parameters1,
    extended_headers={'service': 'manage_feed'}
    )
 rsp=client.callTaxiiService('192.168.1.68',"",VID_TAXII_XML_10,
manage_feed_subscription_request1.to_xml(),port=80 )
 data_received = rsp.read().decode()
 print(data_received)
```

Πίνακας 5.8: Υλοποίηση μεθόδου man_Feed_Subreq

5.5.5 INBOX MESSAGE

Εκτός από τις μορφές αιτημάτων που μπορεί να αποστείλει ο HTTP Client στον Server, έχει επίσης την δυνατότητα να γίνει ο δημιουργός της πληροφορίας (producer), αποστέλλοντας ένα μήνυμα εισερχομένων. Για την υλοποίηση χρησιμοποιήθηκε το module messages_10 της βιβλιοθήκης libtaxii και δημιουργήθηκε ένα αντικείμενο τύπου InboxMessage().

inbox_message1 = *libtaxii.messages_10*.InboxMessage()

Τα υποχρεωτικά ορίσματα που χρησιμοποιήθηκαν είναι το αναγνωριστικό μηνύματος(message_id). Επιπρόσθετα μπορούν να χρησιμοποιηθούν επεκτάσεις επικεφαλίδων (extended headers), τμήματα περιεχομένου(content block), κάποιο μήνυμα για τον Server(message), ή πληροφορίες εγγραφής (subscription_information) στην περίπτωση που το μήνυμα που αποστέλλεται αφορά μια υπάρχουσα εγγραφή σε συλλογή δεδομένων.

Δημιουργήθηκε ένα αντικείμενο τύπου ContentBlock με ορίσματα την μορφή των περιεχομένων προς αποστολή (content_binding), καθώς και το περιεχόμενο το οποίο ο πελάτης επιθυμεί να μοιραστεί.

Σε περίπτωση που το μήνυμα που αποστέλλεται αφορά μια ήδη υπάρχουσα συλλογή δεδομένων, χρησιμοποιείται η κλάση SubscriptionInformation για να δημιουργηθεί ένα αντικείμενο της εγγραφής στην συγκεκριμένη συλλογή. Ως ορίσματα χρησιμοποιούνται το όνομα της συλλογής δεδομένων (Feed_name) της οποίας περιεχόμενο αποστέλλεται και το αναγνωριστικό εγγραφής (subscription_id). Μπορούν επίσης να χρησιμοποιηθούν ετικέτες χρονοσφραγίδας (inclusive_begin_timestamp_label,inclusive_end_timestamp_label) με το εύρος της συλλογής δεδομένων που καλύπτει το συγκεκριμένο μήνυμα.

```
def InboxMesssage():

    print("Sending an inbox message to the server with Stix Content")
    xmldoc = xml.dom.minidom.parse('/home/ubuntu/Downloads/1.SXML')
    xmldoc3 = xmldoc.toxml().encode()

    cb1 = tm10.ContentBlock(content_binding=CB_STIX_XML_10, content="xmldoc3")

    subscription_information1 = tm10.SubscriptionInformation(
      feed_name='SomeFeedName',
      subscription_id='SubsId021',
      inclusive_begin_timestamp_label=datetime.datetime.now(tzutc()),
      inclusive_end_timestamp_label=datetime.datetime.now(tzutc()))

    inbox_message1 = tm10.InboxMessage(
      message_id=tm10.generate_message_id(),
      message=,
      subscription_information=subscription_information1,
      content_blocks=[cb1],
      extended_headers={'service': 'Inbox'}
      )
    rsp = client.call_taxi_service ('192.168.1.68', "", VID_TAXII_XML_10, inbox_message1.to_xml(),
     port=80)
    data_received = rsp.read().decode()
    print(data_received)
```

Πίνακας 5.9: Υλοποίηση μεθόδου InboxMessage

5.6 Αποκρίσεις TAXII

Μόλις ο Server λάβει κάποιο αίτημα TAXII το επεξεργάζεται και στέλνει το αντίστοιχο μήνυμα απάντησης. Σε αυτήν την ενότητα παρουσιάζονται οι τύποι μηνυμάτων απάντησης TAXII.

5.6.1 Discovery Response

Ο τύπος απόκρισης Discovery Response χρησιμοποιείται από τον HTTP Server για να απαντήσει σε ένα αίτημα τύπου Discovery Request. Μέσω του συγκεκριμένου τύπου απόκρισης, ο Server γνωστοποιεί στον πελάτη τις διαθέσιμες προς αυτόν υπηρεσίες TAXII. Θα πρέπει να σημειωθεί ότι ο Server δεν κοινοποιεί το σύνολο των υπηρεσιών τις οποίες κατέχει, παρά μόνο αυτές που είναι διαθέσιμες για τον συγκεκριμένο πελάτη.

Για την υλοποίηση χρησιμοποιήθηκε το module messages_10 της βιβλιοθήκης libtaxii, δημιουργώντας ένα αντικείμενο τύπου DiscoveryResponse.

discovery_response = libtaxii.messages_10.DiscoveryResponse()

Ως ορίσματα χρησιμοποιήθηκαν το αναγνωριστικό μηνύματος(message_id) το οποίο δημιουργήθηκε μέσω της μεθόδου generate_message_id(), το πεδίο απάντησης σε(in response to), το οποίο παίρνει ως τιμή το αναγνωριστικό μηνύματος του αιτήματος στο οποίο απαντάει ο Server, καθώς και μια αναφορά υπηρεσιών(Service instances). Το όρισμα Service instances αποτελεί μια λίστα αντικειμένων τύπου ServiceInstance.

Στην συνέχεια χρησιμοποιήθηκε η κλάση ServiceInstance, για την δημιουργία ενός αντικείμενου με τις διαθέσιμες υπηρεσίες οι οποίες θα συμπεριληφθούν στο πεδίο Service instances του Discovery Response.

service_instance = libtaxii.messages_10.ServiceInstance()

Ως ορίσματα χρησιμοποιήθηκαν οι τύποι υπηρεσιών(service_types), η έκδοση της υπηρεσίας(service version), το πρωτόκολλο υπηρεσιών(protocol_binding), η διεύθυνση της υπηρεσίας(service_address), η μορφή των μηνυμάτων(message_binding), οι τύποι περιεχομένων οι οποίοι γίνονται αποδεκτοί(Inbox_service_accepted_content), η διαθεσιμότητα της υπηρεσίας (available) και ένα μήνυμα με πληροφορίες σχετικές με την υπηρεσία.

Αφού δημιουργήθηκε το αντικείμενο ServiceInstance προστέθηκε στην λίστα αντικειμένων τύπου ServiceInstance του Discovery Response.

discovery_response.service_instances.append(service_instance)

Σε περίπτωση που εντοπιστεί κάποιο σφάλμα κατά την διάρκεια επεξεργασίας του αρχικού αιτήματος, ο Server απαντάει με το αντίστοιχο μήνυμα κατάστασης.

Για την δημιουργία του μηνύματος κατάστασης χρησιμοποιήθηκε η κλάση StatusMessage.

status_message1 = libtaxii.messages_10.StatusMessage()

Τα αντικείμενα της συγκεκριμένης κλάσης παίρνουν ως ορίσματα το αναγνωριστικό μηνύματος(message_id), το αναγνωριστικό μηνύματος στο οποίο απαντάει το μήνυμα κατάστασης(in_response_to) και τον τύπο μηνύματος κατάστασης(status_type). Προαιρετικά ορίσματα που μπορούν να συμπεριληφθούν είναι κάποιο μήνυμα με πληροφορίες σχετικές με το μήνυμα κατάστασης(message) όπως και πληροφορίες κατάστασης σχετικές με το συγκεκριμένο μήνυμα(status_details).

```
def disc_resp(self,msg):

    x =TAXIIHTTPRequestHandler.getid(msg)

   try:

      discovery_response = tm10.DiscoveryResponse(
      message_id=tm10.generate_message_id(),
      in_response_to=x,
      service_instances="")

      service_instance = tm10.ServiceInstance(
      service_type=SVC_DISCOVERY,
      services_version=VID_TAXII_SERVICES_10,
      protocol_binding=VID_TAXII_HTTPS_10,
      service_address='https://example.com/inbox/',
      message_bindings=[VID_TAXII_XML_10],
      inbox_service_accepted_content=[CB_STIX_XML_10],
      available=True,
      message='This is a sample inbox service instance')
      discovery_response.service_instances.append(service_instance)

      self.wfile.write(discovery_response.to_xml())


  except:
      status_message1 = tm10.StatusMessage(
        message_id=tm10.generate_message_id(),
        in_response_to=x,
        status_type=ST_BAD_MESSAGE,
        status_detail='Machine-processable info here!',
        message='This is a message.')
        self.wfile.write(status_message1.to_xml())
```

Πίνακας 5.10: Υλοποίηση μεθόδου disc_resp

5.6.2 Feed Information Response

Ο συγκεκριμένος τύπος απόκρισης χρησιμοποιείται ως απάντηση σε ένα αίτημα τύπου Feed Information Request. Ο HTTP Server δέχεται ένα αίτημα σχετικά με διαθέσιμες συλλογές δεδομένων και απαντάει γνωστοποιώντας στον Client, πληροφορίες σχετικά με τις συλλογές δεδομένων τις οποίες επιθυμεί να μοιραστεί μαζί του. Σε περίπτωση που ο Server εντοπίσει κάποιο σφάλμα κατά την διαδικασία επεξεργασίας του αιτήματος ή αν δεν επιθυμεί να μοιραστεί πληροφορίες συλλογών δεδομένων με τον συγκεκριμένο Client απαντάει με το αντίστοιχο μήνυμα κατάστασης.

Για την υλοποίηση χρησιμοποιήθηκε το module messages_10 της βιβλιοθήκης libtaxi,i δημιουργώντας ένα αντικείμενο τύπου FeedInformationResponse.

feed_information_response1 = libtaxii.messages_10.FeedInformationResponse()

Ως ορίσματα χρησιμοποιήθηκαν το αναγνωριστικό μηνύματος(message_id) και το αναγνωριστικό μηνύματος του αρχικού αιτήματος στο οποίο το συγκεκριμένο μήνυμα απαντάει(in_response_to). Προαιρετικά ορίσματα που μπορούν να χρησιμοποιηθούν είναι επεκτάσεις επικεφαλίδων(extended_headers) καθώς και μια λίστα με τις διαθέσιμες συλλογές δεδομένων(feed_informations).

Επιπρόσθετα δημιουργήθηκαν δύο αντικείμενα τύπου FeedInformations τα οποία αντιπροσωπεύουν συλλογές δεδομένων και συμπεριλήφθηκαν στο πεδίο feed_informations της FeedInformationResponse.

Για την υλοποίηση χρησιμοποιήσαμε το module messages_10 της βιβλιοθήκης libtaxii, δημιουργώντας δύο αντικείμενα τύπου FeedInformations.

feed1 = libtaxii.messages_10.FeedInformation()

Ως ορίσματα χρησιμοποιήθηκαν το όνομα της συλλογής δεδομένων(feed_name), η περιγραφή της συλλογής δεδομένων(feed_description) και οι τύποι περιεχόμενου που υποστηρίζονται από την συγκεκριμένη συλλογή δεδομένων(supported_contents). Προαιρετικά ορίσματα που χρησιμοποιήθηκαν είναι η διαθεσιμότητα της συλλογής δεδομένων(available), τα πρωτόκολλα προώθησης δεδομένων(push_methods), η διεύθυνση της υπηρεσίας προώθησης η οποία

κατέχει την συγκεκριμένης συλλογή(polling_service_instance), καθώς και τα πρωτόκολλα και η διεύθυνση του δαίμονα TAXII ο οποίος διαχειρίζεται εγγραφές για την συγκεκριμένη συλλογή δεδομένων(subscription_methods).

Δημιουργήθηκε ένα αντικείμενο τύπου PushMethod με ορίσματα το πρωτόκολλο προώθησης(push_protocol) και την μορφή μηνυμάτων που μπορούν να χρησιμοποιηθούν για την προώθηση περιεχομένου(push_message_bindings).

Επιπλέον δημιουργήθηκε ένα αντικείμενο τύπου PollingServiceInstance με ορίσματα το πρωτόκολλο απόκτησης(poll_protocol), την διεύθυνση της υπηρεσίας απόκτησης(poll_address) και την μορφή των υποστηριζόμενων μηνυμάτων από την υπηρεσία απόκτησης(poll_message_bindings).

Τέλος δημιουργήθηκε ένα αντικείμενο τύπου SubscriptionMethod με ορίσματα το πρωτόκολλο εγγραφής(subscription_protocol), την διεύθυνση του δαίμονα TAXII που κατέχει την υπηρεσία διαχείρισης συλλογών(subscription_address) και τον τύπο μηνυμάτων που υποστηρίζονται από την υπηρεσία διαχείρισης συλλογών (subscription_message_bindings).

Τα αντικείμενα τύπου PushMethod,PollingServiceInstance και SubscriptionMethod συμπεριλήφθηκαν στο αντικείμενο τύπου FeedInformationResponse.

Για την δημιουργία του μηνύματος κατάστασης σε περίπτωση που διαπιστωθεί κάποιο σφάλμα κατά την επεξεργασία του αιτήματος ή σε περίπτωση αρνητικής απάντησης του Server χρησιμοποιήθηκε η κλάση StatusMessage. Τα ορίσματα που χρησιμοποιήθηκαν είναι το αναγνωριστικό μηνύματος(message_id),το αναγνωριστικό μηνύματος στο οποίο απαντάει το μήνυμα κατάστασης(in_response_to) και ο τύπος μηνύματος κατάστασης(status_type). Επιπλέον χρησιμοποιήθηκαν πληροφορίες κατάστασης(status_details) και ένα μήνυμα με επιπλέον πληροφορίες(message).

```
def feed_info(self,msg):

    x = TAXIIHTTPRequestHandler.getid(msg)

    try:
      push_method1 = tm10.PushMethod(
      push_protocol=VID_TAXII_HTTP_10,
      push_message_bindings=[VID_TAXII_XML_10])
```

```
    polling_service1 = tm10.PollingServiceInstance(
    poll_protocol=VID_TAXII_HTTP_10,
    poll_address='http://example.com/PollService/',
    poll_message_bindings=[VID_TAXII_XML_10])

    subscription_service1 = tm10.SubscriptionMethod(
    subscription_protocol=VID_TAXII_HTTP_10,
    subscription_address='http://example.com/SubsService/',
    subscription_message_bindings=[VID_TAXII_XML_10])

    feed1 = tm10.FeedInformation(
    feed_name='Feed1',
    feed_description='Description of a feed',
    supported_contents=[CB_STIX_XML_10],
    available=True,
    push_methods=[push_method1],
    polling_service_instances=[polling_service1],
    subscription_methods=[subscription_service1])

    feed2 = tm10.FeedInformation(
    feed_name='Feed1',
    feed_description='Description of a feed',
    supported_contents=[CB_STIX_XML_10],
    available=True,
    push_methods=[push_method1],
    polling_service_instances=[polling_service1],
    subscription_methods=[subscription_service1])

    feed_information_response1 = tm10.FeedInformationResponse(

    message_id=tm10.generate_message_id(),
    in_response_to=x,
    feed_informations=(feed1, feed2))
    self.wfile.write(feed_information_response1.to_xml())
  except:
    status_message1 = tm10.StatusMessage(
    message_id=tm10.generate_message_id(),
    in_response_to=x,
    status_type=ST_BAD_MESSAGE,
    status_detail='Machine-processable info here!',
    message='This is a message.')
    self.wfile.write(status_message1.to_xml())
```

Πίνακας 5.11: Υλοποίηση μεθόδου feed_info

5.6.3 Poll Response

Ο τύπος απόκρισης PollResponse χρησιμοποιείται από τον Server ως απάντηση σε ένα αίτημα τύπου PollRequest. Ο HTTP Client μέσω του συγκεκριμένου αιτήματος γνωστοποιεί στον Server το όνομα της συλλογής δεδομένων για την οποία θέλει να αποκτήσει περιεχόμενο. Ο Server με την σειρά του επεξεργάζεται το αίτημα. Αν εντοπίσει κάποιο σφάλμα κατά την διαδικασία επεξεργασίας, απαντάει με το αντίστοιχο μήνυμα κατάστασης. Επίσης αν αποφασίσει ότι δεν επιθυμεί να ικανοποιήσει το αίτημα απαντάει πάλι με κάποιο μήνυμα κατάστασης. Στην περίπτωση που αποφασίσει να ικανοποιήσει το αίτημα απαντάει με ένα μήνυμα τύπου PollResponse, μέσα στο οποίο συμπεριλαμβάνει περιεχόμενο από την συλλογή δεδομένων που του ζητήθηκε.

Για την υλοποίηση χρησιμοποιήθηκε το module messages_10 της βιβλιοθήκης libtaxii δημιουργώντας ένα αντικείμενο τύπου PollResponse.

```
poll_response1 = libtaxii.messages_10.PollResponse()
```

Ένα αντικείμενο της κλάσης PollResponse δέχεται ως υποχρεωτικά ορίσματα το αναγνωριστικό μηνύματος(message_id), το αναγνωριστικό μηνύματος του αιτήματος στο οποίο απαντάει το συγκεκριμένο μήνυμα(in_response_to), το όνομα της συλλογής δεδομένων από την οποία θα αποκτηθεί περιεχόμενο(feed_name) και μια ετικέτα χρονοσφραγίδας με το τέλος του προς μετάδοση περιεχομένου (inclusive_end_timestamp_label). Προαιρετικά ορίσματα που μπορούν να χρησιμοποιηθούν είναι επεκτάσεις επικεφαλίδων(extended headers), μια ετικέτα χρονοσφραγίδας με την αρχή του προς μετάδοση περιεχομένου (inclusive_begin_timpestamp_label), το αναγνωριστικό εγγραφής της οποίας περιεχόμενο παρέχεται(subscription_id), κάποιο μήνυμα με επιπρόσθετες πληροφορίες και τέλος τμήματα περιεχομένου(content_blocks).

Εάν δεν διαπιστωθεί κάποιο σφάλμα κατά την διαδικασία επεξεργασίας του αιτήματος, ο Server ελέγχει εάν διαθέτει την συλλογής δεδομένων της οποίας περιεχόμενο αιτείται. Στην περίπτωση που το όνομα της συλλογής περιλαμβάνεται στις συλλογές δεδομένων του Server, αποστέλλεται ένα μήνυμα τύπου PollResponse με το αντίστοιχο περιεχόμενο. Σε αντίθετη περίπτωση ο Server απαντάει με το αντίστοιχο μήνυμα κατάστασης.

Για την υλοποίηση του μηνύματος κατάστασης χρησιμοποιήθηκε η κλάση StatusMessage. Τα ορίσματα που χρησιμοποιήθηκαν είναι το αναγνωριστικό μηνύματος(message_id), το αναγνωριστικό μηνύματος στο οποίο απαντάει το μήνυμα κατάστασης(in_response_to) και ο τύπος μηνύματος κατάστασης(status_type). Επιπλέον χρησιμοποιήθηκαν πληροφορίες κατάστασης(status_details) και ένα μήνυμα με επιπλέον πληροφορίες(message).

```
def poll_Resp(self,msg):


    x = TAXIIHTTPRequestHandler.getid(msg)

    y=TAXIIHTTPRequestHandler.getfeedname(msg)

    xmldoc = xml.dom.minidom.parse('/home/ubuntu/Downloads/1.SXML')
    xmldoc3 = xmldoc.toxml().encode()

    cb1 = tm10.ContentBlock(content_binding=CB_STIX_XML_10, content=xmldoc3)

    poll_response1 = tm10.PollResponse(
        message_id=tm10.generate_message_id(),
        in_response_to=x,
        feed_name="FeedPol",
        inclusive_begin_timestamp_label=datetime.datetime.now(tzutc()),
        inclusive_end_timestamp_label=datetime.datetime.now(tzutc()),
        subscription_id='SubsId001',
        message="Poll Response Example",
        content_blocks=[cb1])
    try:
      if poll_response1.feed_name == y:
        print("sending the xml")
        self.wfile.write(poll_response1.to_xml())



      else:
        status_message1 = tm10.StatusMessage(
          message_id=tm10.generate_message_id(),
          in_response_to=x,
          status_type=ST_NOT_FOUND,
          status_detail='Machine-processable info here!',
          message='Feed not found.')

        self.wfile.write(status_message1.to_xml())
    except:
      status_message2 = tm10.StatusMessage(
        message_id=tm10.generate_message_id(),
```

```
        in_response_to=x,
        status_type=ST_BAD_MESSAGE,
        status_detail='Machine-processable info here!',
        message='Bad Message')

    self.wfile.write(status_message1.to_xml())
```

Πίνακας 5.12: Υλοποίηση μεθόδου poll_Resp

5.6.4 Manage Feed Subscription Response

Ο συγκεκριμένος τύπος μηνυμάτων χρησιμοποιείται ως απάντηση σε ένα αίτημα τύπου Manage Feed Subscription Request. Μέσω του συγκεκριμένου αιτήματος ο Client μπορεί να αιτηθεί την εγγραφή του, ή την διαγραφή του από κάποια συλλογή δεδομένων. Ο Server επεξεργάζεται το αίτημα και αν διαπιστώσει κάποιο σφάλμα απαντάει με ένα μήνυμα κατάστασης. Εάν αποφασίσει να αποδεχτεί το αίτημα, απαντάει με ένα μήνυμα τύπου ManageFeedSubscriptionResponse. Σε αντίθετη περίπτωση απαντάει με κάποιο μήνυμα κατάστασης εξηγώντας τον λόγο για τον οποίον δεν αποδέχεται το αίτημα.

Ένα αντικείμενο της τάξης ManageFeedSubscriptionResponse δέχεται ως υποχρεωτικά ορίσματα το αναγνωριστικό μηνύματος(message_id), το αναγνωριστικό μηνύματος του αιτήματος στο οποίο απαντάει το συγκεκριμένο μήνυμα(in_response_to) και το όνομα της συλλογής στην οποία θα εφαρμοστεί η αντίστοιχη ενέργεια εγγραφής ή διαγραφής(feed_name). Επιπλέον ορίσματα που μπορούν να χρησιμοποιηθούν είναι επεκτάσεις επικεφαλίδων(extended_headers), κάποιο μήνυμα με επιπλέον πληροφορίες(message) και κάποια αναφορά εγγραφής (subscription instances).

Για την δημιουργία μιας αναφοράς εγγραφής χρησιμοποιήθηκε η κλάση SubscriptionInstance με ορίσματα το αναγνωριστικό εγγραφής (subscription_id), παραμέτρους παράδοσης(delivery_parameters) και μια λίστα με αναφορές υπηρεσιών απόκτησης(poll_instances).

Για την δημιουργία μιας αναφοράς υπηρεσίας απόκτησης, χρησιμοποιήθηκε η κλάση PollInstance, με ορίσματα το υποστηριζόμενο πρωτόκολλο της υπηρεσίας απόκτησης(poll_protocol), την διεύθυνση του δαίμονα TAXII που κατέχει την υπηρεσία

απόκτησης(poll_address) και τους τύπους μηνυμάτων που μπορούν να χρησιμοποιηθούν στην επικοινωνία με την συγκεκριμένη υπηρεσία(poll_message_bindings).

Κατά την υλοποίηση αρχικά ελέγχεται εάν η συλλογή στην οποία θέλει να εγγραφεί ή να διαγράφει ο πελάτης υπάρχει. Σε περίπτωση που ο Server δεν διαθέτει την συγκεκριμένη συλλογή, απαντάει με ένα μήνυμα κατάστασης(status_type=ST_NOT_FOUND). Εάν ο πελάτης θέλει να εγγραφεί σε κάποια διαθέσιμη συλλογή, ο Server απαντάει με το αντίστοιχο μήνυμα τύπου ManageFeedSubscriptionResponse, δημιουργώντας μια καινούργια εγγραφή. Στην περίπτωση που ο πελάτης αιτείται διαγραφή, ο Server ελέγχει εάν διαθέτει το αναγνωριστικό εγγραφής του πελάτη σε κάποια από τις συλλογές του και απαντάει με μήνυμα τύπου ManageFeedSubscriptionResponse.

```
def manfeed_subresp(self,msg):

    print("Replying to manage feed request")


    x = TAXIIHTTPRequestHandler.getid(msg)
    y = TAXIIHTTPRequestHandler.getfeedname(msg)
    z =TAXIIHTTPRequestHandler.getaction(msg)
    e =TAXIIHTTPRequestHandler.getsubscriptionid(msg)


    poll_instance1 = tm10.PollInstance(
      poll_protocol=VID_TAXII_HTTP_10,
      poll_address='http://example.com/poll',
      poll_message_bindings=[VID_TAXII_XML_10])

    subscription_instance1 = tm10.SubscriptionInstance(
      subscription_id=tm10.generate_message_id(),
      poll_instances=[poll_instance1])

    manage_feed_subscription_response1 = tm10.ManageFeedSubscriptionResponse(
      message_id=tm10.generate_message_id(),
      in_response_to=x,
      feed_name='Feed',
      message='Your request will ne honored',
      subscription_instances=[subscription_instance1])


    try:
      if y == manage_feed_subscription_response1.feed_name:
        print("the requested action is : " + z)
```

```
        if z=="UNSUBSCRIBE":

          if subscription_instance1.subscription_id == e:
            manage_feed_subscription_response1.subscription_instances = []
            self.wfile.write(manage_feed_subscription_response1.to_xml())
            print(manage_feed_subscription_response1.subscription_instances)

          else:
           status_message = tm10.StatusMessage(
           message_id=tm10.generate_message_id(),
          in_response_to=x,
          status_type=ST_NOT_FOUND,
          status_detail='Machine-processable info here!',
          message='wrong subscription ID.')
          self.wfile.write(status_message.to_xml())


        elif z=="SUBSCRIBE":

          subscription_instance2 = tm10.SubscriptionInstance(
          subscription_id=tm10.generate_message_id(),
          poll_instances=[poll_instance1])
          manage_feed_subscription_response1.subscription_instances=[subscription_instance2]
           print( manage_feed_subscription_response1.subscription_instances)
            self.wfile.write(manage_feed_subscription_response1.to_xml())


        else:
          status_message = tm10.StatusMessage(
          message_id=tm10.generate_message_id(),
          in_response_to=x,
          status_type=ST_NOT_FOUND ,
          status_detail='Machine-processable info here!',
          message='PENDING.')
          self.wfile.write(status_message.to_xml())


      else:

        status_message2 = tm10.StatusMessage(
        message_id=tm10.generate_message_id(),
        in_response_to=x,
        status_type=ST_NOT_FOUND,
        status_detail='Machine-processable info here!',
        message='Wrong feed name.')
        self.wfile.write(status_message2.to_xml())
    except:
      status_message3 = tm10.StatusMessage(
      message_id=tm10.generate_message_id(),
      in_response_to=x,
```

```
        status_type=ST_BAD_MESSAGE,
        status_detail='Machine-processable info here!',
        message='Bad Message.')
        self.wfile.write(status_message3.to_xml())
```

Πίνακας 5.13: Υλοποίηση μεθόδου manfeed_subresp

5.6.5 Inbox Response

Αυτό ο τύπος μηνυμάτων χρησιμοποιείται σαν απάντηση σε μηνύματα τύπου InboxMessage. Ο πελάτης στέλνει ένα μήνυμα περιεχομένου στον Server και αυτός απλά ενημερώνει τον πελάτη κατά πόσο το μήνυμα έχει ληφθεί η όχι.

Για την υλοποίηση ενός μηνύματος Inbox Response, χρησιμοποιήθηκε η κλάση StatusMessage, μέσω της οποίας δημιουργούνται τα αντίστοιχα μηνύματα κατάστασης τα οποία αποστέλλονται στον πελάτη.

Τα αντικείμενα της συγκεκριμένης κλάσης παίρνουν ως ορίσματα το αναγνωριστικό μηνύματος(message_id), το αναγνωριστικό μηνύματος στο οποίο απαντάει το μήνυμα κατάστασης(in_response_to) και τον τύπο μηνύματος κατάστασης (status_type). Προαιρετικά ορίσματα που μπορούν να συμπεριληφθούν είναι κάποιο μήνυμα με πληροφορίες σχετικές με το μήνυμα κατάστασης(message) όπως και πληροφορίες κατάστασης σχετικές με το συγκεκριμένο μήνυμα(status_details).

Στην περίπτωση της θετικής μετάδοσης του μηνύματος από τον πελάτη το πεδίο status_type παίρνει τιμή ST_SUCCESS, ενώ στην αντίθετη περίπτωση ST_FAILURE.

```
def Inbox_resp(self,msg):
    x = TAXIIRequestHandler.getid(msg)
    if msg:
      status_message1 = tm10.StatusMessage(
        message_id=tm10.generate_message_id(),
        in_response_to=x,
        status_type=ST_SUCCESS,
        status_detail='Machine-processable info here!',
        message='Message Received')
      self.wfile.write(status_message1.to_xml())
    else:
      status_message2 = tm10.StatusMessage(
        message_id=tm10.generate_message_id(),
        in_response_to=x,
```

```
        status_type=ST_FAILURE,
        status_detail='Machine-processable info here!',
        message='Failed')
    self.wfile.write(status_message2.to_xml())
```

Πίνακας 5.14: Υλοποίηση μεθόδου Inbox_Resp

5.7 Βοηθητικές Μέθοδοι

Για την διαχείριση των εισερχομένων μηνυμάτων TAXII, χρησιμοποιήθηκαν κάποιες επιπλέον μέθοδοι. Σε αυτήν την ενότητα παρουσιάζονται οι βοηθητικές μέθοδοι του TAXII.

5.7.1 Η μέθοδος call_taxii_service

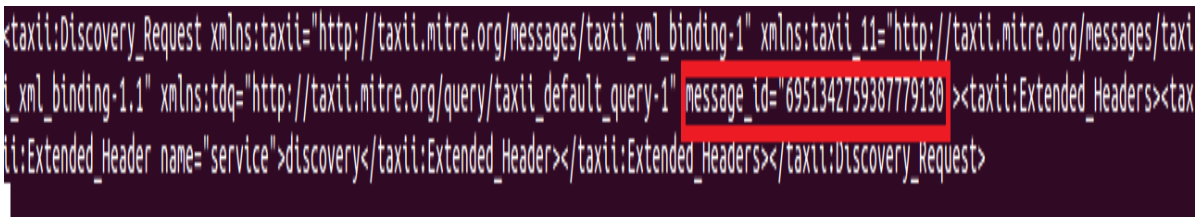
Η μέθοδος call_taxii_service βρίσκεται στο module clients της βιβλιοθήκης libtaxii. Η μέθοδος παίρνει ως ορίσματα την διεύθυνση του Server(host), την μορφή του μηνύματος το οποίο αποστέλλεται(message_binding), διάφορα δεδομένα τα οποία επιθυμεί ο πελάτης να μοιραστεί(post_data), καθώς και την θύρα του Server(port), στην οποία θέλει να συνδεθεί.

rsp= client.call_taxii_service('192.168.1.68',"", VID_TAXII_XML_10, discovery_request.to_xml())

5.7.2 Η μέθοδος getID

Κατά την λήψη ενός αιτήματος, ο Server λαμβάνει ένα μήνυμα μορφής xml. Για να απαντήσει ο Server στο αίτημα θα πρέπει να εξάγει κάποιες πληροφορίες από το εισερχόμενο μήνυμα. Μέσω της μεθόδου getID, ο Server εξάγει το αναγνωριστικό μηνύματος του ληφθέντος αιτήματος, με σκοπό να το χρησιμοποιήσει σαν τιμή στο πεδίο “in response to[4]”. Για την υλοποίηση της μεθόδου getID, χρησιμοποιείται η κλάση ElementTree. Με την βοήθεια του χαρακτηριστικού attrib εξάγεται η τιμή του αντίστοιχου χαρακτηριστικού.

[4] Ενότητα 5.5.1 Discovery Response



Εικόνα 5.5: Εισερχόμενο αίτημα Discovery Request

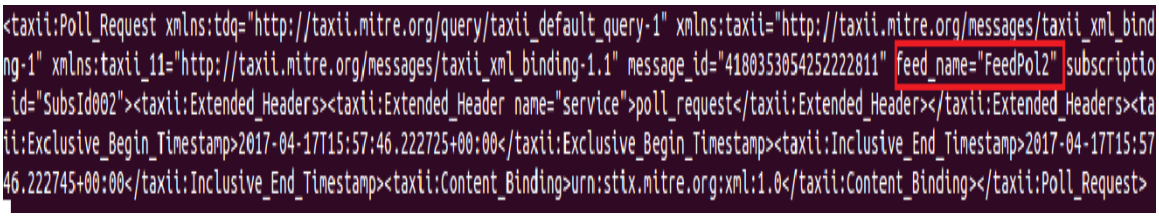
Στην παραπάνω εικόνα φαίνεται η μορφή ενός αιτήματος Discovery Request το οποίο καταφθάνει στον Server. Για να απαντήσει ο Server στο συγκεκριμένο αίτημα θα πρέπει να χρησιμοποιήσει το αναγνωριστικό μηνύματος(κόκκινο πλαίσιο). Μέσω της μεθόδου fromstring, εμφωλεύει το εισερχόμενο μήνυμα σε ένα αντικείμενο τύπου Element και στην συνέχεια εξάγει τη τιμή του χαρακτηριστικού message_id.

```
def getiD(msg):
    el = ET.fromstring(msg)
    return el.attrib['message_id']
```

Πίνακας 5.15: Υλοποίηση μεθόδου getID

5.7.3 Η μέθοδος getFeedName

Σε κάποιο αίτημα τύπου PollRequest, ο Server θα πρέπει να ελέγξει εάν διαθέτει την συγκεκριμένη συλλογή δεδομένων της οποίας περιεχόμενο αιτείται. Θα πρέπει λοιπόν να εξαχθεί το όνομα της συλλογής(κόκκινο πλαίσιο) από το εισερχόμενο αίτημα και να ελεγχθεί εάν είναι ίδιο με κάποιο όνομα συλλογής του Server.



Εικόνα 5.6: Εισερχόμενο αίτημα Poll Request

Για την υλοποίηση της μεθόδου getFeedName ακολουθείται η ίδια διαδικασία, με την διαφορά ότι εξάγεται η τιμή του χαρακτηριστικού feed_name.

```
def getFeedName(msg):
    el=ET.fromstring(msg)
    return el.attrib['feed_name']
```

Πίνακας 5.16: Υλοποίηση μεθόδου getFeedName

5.7.4 Οι μέθοδος getAction και getSubscriptionID

Σε ένα αίτημα τύπου Subscription Management Request, εκτός από το αναγνωριστικό μηνύματος και το όνομα της συλλογής δεδομένων, θα πρέπει να εξαχθεί η ενέργεια η οποία αιτείται να εφαρμοστεί σε κάποια συλλογή δεδομένων και το αναγνωριστικό εγγραφής, σε περίπτωση που ο πελάτης αιτείται διαγραφή.

Εικόνα 5.7: Εισερχόμενο Αίτημα Subscription Management Request

Όπως και στις προηγούμενες περιπτώσεις χρησιμοποιείται η κλάση ElementTREE εξάγοντας τα χαρακτηριστικά "action" και "subscription_id", τα οποία φαίνονται στα κόκκινα πλαίσια. Στην συνέχεα τα συγκεκριμένα χαρακτηριστικά θα χρησιμοποιηθούν στα αντίστοιχα πεδία της μεθόδου manfeed_subresp[5].

```
def getAction(msg):
    el = ET.fromstring(msg)
    return el.attrib['action']
```

Πίνακας 5.17: Υλοποίηση μεθόδου getAction

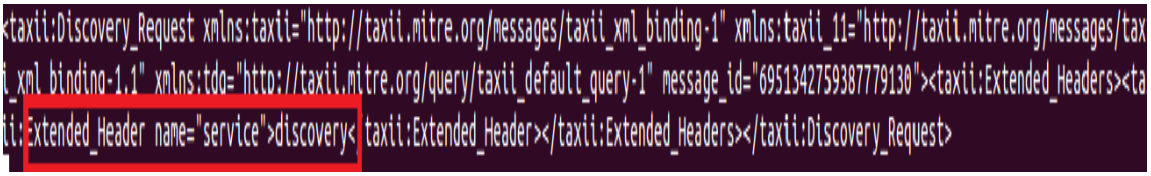
```
def getSubscriptionID(msg):
    el = ET.fromstring(msg)
    return el.attrib['subscription_id']
```

Πίνακας 5.18: Υλοποίηση μεθόδου getSubscriptionID

[5] 5.5.4 Manage Feed Subscription Response

5.7.5 Η μέθοδος getHeader

Μια ακόμα μορφή πληροφορίας που πρέπει να εξαχθεί από τα αιτήματα TAXII είναι οι επεκτάσεις επικεφαλίδων(extended headers). Κατά την δημιουργία ενός αιτήματος TAXII ο πελάτης συμπεριλαμβάνει στο αίτημα του, ένα ζευγάρι επεκτάσεων επικεφαλίδων. Η τιμή του ζευγαριού αποτελεί ένα αναγνωριστικό του αιτήματος που αποστέλλεται στον Server.



Εικόνα 5.8: Εισερχόμενο αίτημα Discovery Request

Όπως φαίνεται στο παραπάνω παράδειγμα το αίτημα περιέχει ένα ζευγάρι επικεφαλίδων, με όνομα service και τιμή discovery. Στην ουσία αποτελεί μια υπόδειξη στον Server για την υπηρεσία την οποία αιτείται ο πελάτης. Πριν ο Server απαντήσει σε κάποιο αίτημα ελέγχει την τιμή των επεκτάσεων επικεφαλίδων και αναλόγως χρησιμοποίει τον αντίστοιχο τύπο μηνυμάτων TAXII. Για την εξαγωγή των επεκτάσεων επικεφαλίδων χρησιμοποιήθηκαν οι μέθοδοι της κλάση ElementTRee.

```
def getHeader(msg):

    tree = ET.ElementTree(ET.fromstring(msg))
    root = tree.getroot()
    x = root[0][0].text
    return x
```

Πίνακας 5.19: Υλοποίηση μεθόδου getHeader

Με την μεταβλητή x γίνεται αναφορά στην τιμή του πρώτου παιδιού, του πρώτου παιδιού, του κόμβου ρίζας. Ο κόμβος ρίζα είναι ο taxii:Discovery_Request, το πρώτο παιδί του ο κόμβος taxii:Extended_headers και το πρώτο παιδί αυτού του κόμβου, είναι ο κόμβος taxii:Extended_header του οποίου το κείμενο(text)είναι “discovery”. Οπότε ο Server εξάγοντας την συγκεκριμένη τιμή αναγνωρίζει ότι πρέπει να εκτελέσει την μέθοδο discovery_response καθώς το αίτημα του Client είναι της μορφής Discovery_request.

Κεφάλαιο 6

Επισκόπηση Λειτουργίας Συστήματος

Η κεντρική ιδέα που ακολουθήθηκε κατά την διαδικασία υλοποίησης της εφαρμογής, αφορούσε τη δημιουργία ενός εικονικού επιχειρησιακού περιβάλλοντος. Ο υπεύθυνος ασφαλείας του οργανισμού, πραγματοποιεί ελέγχους ασφαλείας και ενημερώνει τους υπόλοιπους χρήστες του συστήματος για διάφορες απειλές τις οποίες έχει εντοπίσει. Η διαδικασία ενημέρωσης γίνεται μέσω των διαφόρων μορφών μηνυμάτων TAXII.

Σε πρώτη φάση ο υπεύθυνος ασφαλείας του οργανισμού, εντοπίζει κακόβουλο λογισμικό το οποίο υπάρχει σε κάποια αρχεία του οργανισμού και ξεκινάει την διαδικασία ανάλυσης του. Για την ανάλυση χρησιμοποιεί το σύστημα ανάλυσης κακόβουλου λογισμικού Cuckoo Sandbox παράγοντας τις αντίστοιχες αναφορές. Όπως επισημάνθηκε στο 3ο κεφάλαιο, κατά τη διαδικασία της ανάλυσης κάποιου κακόβουλου λογισμικού είναι ιδιαίτερα σημαντικό από την μία να εξασφαλιστεί η απομόνωση του περιβάλλοντος στο οποίο θα γίνει η ανάλυση και από την άλλη να εξασφαλιστεί ότι το περιβάλλον ανάλυσης θα είναι όσο το δυνατόν πιο ρεαλιστικό.

Το Cuckoo Sandbox μέσα από τις αναφορές που παράγει, προσφέρει μεγάλο εύρος πληροφοριών όπως ομαδοποίηση του προς εξέταση λογισμικού σε κατηγορίες κακόβουλων λογισμικών, κινήσεις που πραγματοποιούνται στο δίκτυο κατά την διάρκεια εκτέλεσης του λογισμικού, αναλύσεις μνήμης του συστήματος στο οποίο εκτελείται το λογισμικό και άλλα. Οι πληροφορίες αυτές μπορούν να χρησιμοποιηθούν σε ποιο εξειδικευμένα εργαλεία ανάλυσης.

Μόλις ολοκληρωθεί η διαδικασία ανάλυσης και μετατροπής των αναφορών του Cuckoo σε μορφή STIX, ο υπεύθυνος ασφαλείας του οργανισμού είναι έτοιμος να μοιραστεί τις πληροφορίες με τους υπόλοιπους χρήστες του συστήματος. Η διαδικασία της επικοινωνίας

γίνεται μέσω ανταλλαγής μηνυμάτων TAXII. Υποστηρίζονται πέντε διαφορετικοί τύποι επικοινωνίας.

Ενα μήνυμα τύπου Discovery Request αποστέλλεται από τον πελάτη στον Server για ανακαλύψει διαθέσιμες υπηρεσίες. Ο Server απαντάει με ένα μήνυμα τύπου Discovery Response γνωστοποιώντας στον πελάτη υπηρεσίες που αυτός επιθυμεί. Δύο πελάτες του συστήματος που αιτούνται μια υπηρεσία ανακάλυψης(Discovery Service) δεν θα πάρουν απαραίτητα την ίδια απάντηση. Αναλόγως με τα δικαιώματα που έχει ο κάθε χρήστης μπορεί να έχει πρόσβαση σε περισσότερες ή λιγότερες υπηρεσίες και συλλογές δεδομένων. Ο κάτοχος μια υπηρεσίας είναι αυτός που κάνει την επιλογή για το τι θα μοιραστεί και με ποιον.

Όταν ο πελάτης αποστέλλει ένα μήνυμα μορφής Feed Information request, αιτείται πληροφορίες σχετικά με διαθέσιμες συλλογές δεδομένων. Ο Server απαντάει με ένα μήνυμα τύπου Feed Information Response. Μέσω αυτού του τύπου μηνυμάτων ο Server στέλνει στον πελάτη πληροφορίες σχετικά με διαθέσιμες συλλογές δεδομένων. Όπως και προηγουμένως η απόφαση για το ποιες συλλογές θα γνωστοποιηθούν στον πελάτη είναι αποκλειστική απόφαση του Server.

Μέσω ενός μηνύματος Poll Request ο πελάτης αποστέλλει ένα αίτημα στον Server για απόκτηση δεδομένων μια συγκεκριμένης συλλογής. Στην περίπτωση που ο Server αποφασίσει να προωθήσει περιεχόμενο της συλλογής που του έχει αιτηθεί τότε απαντάει με ένα μήνυμα τύπου Poll Response.

Ο πελάτης μπορεί επίσης να στείλει ένα αίτημα τύπου Manage Feed Subscription Request. Μέσω ενός τέτοιου αιτήματος, ο πελάτης μπορεί να αιτηθεί την εγγραφή ή την διαγραφή από μια συλλογή δεδομένων. Ο Server απαντάει με ένα μήνυμα τύπου Manage Feed Subscription Response. Στην περίπτωση που το αιτούμενο είναι η εγγραφή σε κάποια συλλογή, εάν ο Server αποδεχτεί το αίτημα τότε δημιουργεί ένα αναγνωριστικό εγγραφής και απαντάει στο αίτημα. Εάν το αιτούμενο είναι η διαγραφή από κάποια συλλογή δεδομένων ο πελάτης συμπεριλαμβάνει στο αίτημα του και το αναγνωριστικό της εγγραφής από την οποία θέλει να διαγραφεί. Όπως και στις προηγούμενες περιπτώσεις ο Server είναι αυτός που παίρνει την τελική απόφαση για τον αν θα εγγράψει ή θα διαγράψει τον πελάτης σε μια συλλογή δεδομένων του.

Ο πελάτης εκτός από αποδέκτης των πληροφοριών μπορεί να γίνει και παραγωγός. Στην περίπτωση που ο πελάτης είναι κάτοχος πληροφοριών σχετικά με κάποιο κακόβουλο λογισμικό

και θέλει να μοιραστεί αυτές τις πληροφορίες με τον Server, προς ενημέρωση των υπολοίπων χρηστών του συστήματος, αποστέλλει ένα μήνυμα τύπου Inbox Message. Μέσα στο μήνυμα περιέχονται δεδομένα από κάποια αναφορά STIX. Ο Server απαντάει με ένα μήνυμα τύπου Inbox Message Response μέσω του οποίου ενημερώνει το πελάτη σχετικά με την σωστή λήψη του μηνύματος. Είναι στην δικαιοδοσία του Server να αξιολογήσει ένα εισερχόμενο μήνυμα και να αποφασίσει εάν θα το συμπεριλάβει σε κάποια συλλογή δεδομένων.

Κεφάλαιο 7

Επίλογος

Όπως επισημάνθηκε στα προηγούμενα κεφάλαια η ενημέρωση αποτελεί τον σημαντικότερο τρόπο πρόληψης ενάντια σε κακόβουλες επιθέσεις. Ο στόχος της συγκριμένης μεταπτυχιακής διατριβής ήταν η υλοποίηση ενός συστήματος επικοινωνίας ανάμεσα σε υπολογιστές, με κύριο σκοπό την ενημέρωση τους σχετικά με πληροφορίες κακόβουλων λογισμικών.

Μέσω της διαδικασίας που ακολουθήθηκε, δημιουργήθηκε ένα σύστημα επικοινωνίας ανάμεσα σε υπολογιστικούς κόμβους, με σκοπό την ανταλλαγή μηνυμάτων TAXII. Οι υπολογιστές έχουν την δυνατότητα να συνδέονται στον Server και να αποστέλλουν τα διάφορα αιτήματα τους με σκοπό να ενημερώνονται για τις διάφορες συλλογές δεδομένων του Server. Με την βοήθεια του συστήματος ανάλυσης κακόβουλου λογισμικού Cuckoo ο υπεύθυνος ασφαλείας μπορεί να ενημερώνει συνεχώς τους χρήστες του συστήματος για νέες απειλές.

Κατά την διάρκεια της υλοποίησης εντοπίστηκαν ζητήματα τα οποία μπορούν να βελτιωθούν στο μέλλον. Πρώτα απ όλα η μετατροπή των αναφορών MAEC σε STIX, γίνεται κάθε φορά μέσω της εκτέλεσης του αντίστοιχου Script. Θα μπορούσε να ενσωματωθεί η βιβλιοθήκη maec-to-stix στο αρχείο εξαγωγής αναφορών MAEC, έτσι ώστε η εξαγώγιμη αναφορά από το Cuckoo να είναι απευθείας σε μορφή STIX. Στην παρούσα φάση αυτό δεν καταστεί δυνατό, καθώς οι εκδόσεις των modules MAEC, CYBOX και STIX που απαιτούνται για την εξαγωγή μιας αναφοράς MAEC είναι διαφορετικές από αυτές που απαιτούνται για την μετατροπή μιας αναφοράς MAEC σε STIX.

Επίσης τα αρχεία STIX που χρησιμοποίει ο Server για να ενημερώσει τους πελάτες φορτώνονται χειροκίνητα. Στο μέλλον θα μπορούσε να δημιουργηθεί μια υλοποίηση η οποία θα ενσωμάτωνε κάθε νέα αναφορά που δημιουργούταν αυτομάτως σε μια λίστα δίνοντας της ένα αναγνωριστικό. Κάθε φορά που κάποιος πελάτης ζητούσε περιεχόμενο συλλογής δεδομένων με

το συγκεκριμένο αναγνωριστικό ο Server θα έψαχνε στην λίστα και θα του επέστρεφε την συγκεκριμένη αναφορά.

Τέλος στην συγκεκριμένη υλοποίηση κάθε χρήστης ο οποίος αποστέλλει σωστά ένα αίτημα για περιεχόμενο συλλογής, εφόσον το αίτημα ληφθεί από τον Server του αποστέλλεται συγκεκριμένο περιεχόμενο. Σε μια πιο ρεαλιστική προσέγγιση θα έπρεπε κάθε φορά που κάποιος χρήστης αιτούνταν περιεχόμενο να γίνεται πρώτα έλεγχος δικαιωμάτων και εφόσον πληρούνται τα κριτήρια να του αποστέλλεται το αντίστοιχο περιεχόμενο.

Βιβλιογραφία

[01] Anon, Sandbox (computer security) - Wikipedia. Available at: https://en.wikipedia.org/wiki/Sandbox_(computer_security). Last Visited May 20, 2017.

[02] Anon, Web server - Wikipedia. Available at: https://en.wikipedia.org/wiki/Web_server. Last Visited May 20, 2017.

[03] Barnum, S., 2014. Standardizing Cyber Threat Intelligence Information with the Structured Threat Information eXpression (STIX ™). , pp.1–22. Available at: https://stixproject.github.io/about/STIX_Whitepaper_v1.1.pdf.

[04] Connolly, J., Davidson, M. & Schmidt, C., 2014. The Trusted Automated eXchange of Indicator Information (TAXII™). Available at: http://taxii.mitre.org/about/documents/Introduction_to_TAXII_White_Paper_May_2014.pdf.

[05] Davidson, M. & Schmidt, C., 2014a. TAXII Overview Version 1.1. Available at: http://taxiiproject.github.io/releases/1.1/TAXII_Overview.pdf.

[06] Davidson, M. & Schmidt, C., 2014b. The TAXII Content Binding Reference. Available at: http://taxiiproject.github.io/releases/1.1/TAXII_ContentBinding_Reference_v3.pdf.

[07] Davidson, M. & Schmidt, C., 2014c. The TAXII Default Query Specification. , pp.0–23. Available at: http://taxiiproject.github.io/releases/1.1/TAXII_Default_Query_Specification.pdf.

[08] Davidson, M. & Schmidt, C., 2013. The TAXII HTTP Protocol Binding Specification. , pp.0–10. Available at: http://taxiiproject.github.io/releases/1.1/TAXII_HTTPProtocolBinding_Specification.pdf.

[09] Davidson, M. & Schmidt, C., 2014d. The TAXII Services Specification. Availabe at: http://taxiiproject.github.io/releases/1.1/TAXII_Services_Specification.pdf.

[10] Davidson, M. & Schmidt, C., 2014e. The TAXII XML Message Binding Specification. Availabe at: http://taxiiproject.github.io/releases/1.1/TAXII_XMLMessageBinding_Specification.pdf.

[11] Deitel, H.M., 2002. *PYTHON HOW TO PROGRAM*, Prentice Hall. Available at: ftp://www.mans.edu.eg/Learn/Python/pdf/Python%20How%20to%20Program%202002.pdf.

[12] Desiree Beck, Ivan Kirillov, Penny Chase, M., 2014. THE MAEC ™ LANGUAGE-OVERVIEW. Available at: http://maecproject.github.io/documentation/overview/MAEC_Overview.pdf.

[13] Desiree Beck, Ivan Kirillov, Penny Chase, M., 2013. THE MAEC ™ LANGUAGE VERSION 4.0.1 SPECIFICATION. Available at: https://maec.mitre.org/language/version4.0.1/MAEC_Language_Specification_11-15-2013.pdf.

[14] Kirillov, I.A., 2014. Characterizing Malware with MAEC and STIX. , pp.1–8. Available at: https://maecproject.github.io/documentation/characterize_malware/Characterizing_Malware_MAEC_and_STIX_v1.0.pdf.

[15] Oktavianto, D., 2013. *Cuckoo Malware Analysis Analyze malware using Cuckoo Sandbox*, Birmingham: Packt Publishing. Available at: http://gbv.eblib.com/patron/FullRecord.aspx?p=1389346.

[16] Sandbox, C., 2017. Cuckoo Sandbox Book. Available at: https://media.readthedocs.org/pdf/cuckoo/0.4.2/cuckoo.pdf.

[17] Virtualbox, R., 2017. Oracle VM VirtualBox. , pp.1–359. Available at: http://download.virtualbox.org/virtualbox/5.1.22/UserManual.pdf.

Παράρτημα Α
Κώδικας Υλοποίησης

Σε αυτό το παράρτημα παρουσιάζεται ο κώδικα υλοποίησης των HTTP Server και HTTP Client.

Α.1 Κώδικας Υλοποίησης HTTP Server

Ακολουθεί ο κώδικας υλοποίησης του HTTP Server, ο οποίος είναι υπεύθυνος για την διαχείριση των αιτημάτων TAXII.

```
1. import http.server  # Εισαγωγή Απαραίτητων Βιβλιοθηκών
2. import libtaxii.messages_10 as tm10
3. from libtaxii.constants import *
4. import xml.etree.ElementTree as ET
5. import xml.dom.minidom
6. import datetime
7. from dateutil.tz import tzutc
8. import http.server
9.
10. class TAXIIHTTPRequestHandler(http.server.BaseHTTPRequestHandler):  # Κλάση
     #διαχείρισης εισερχόμενων αιτημάτων
11.
12.
13.     def getid(msg):  # Μέθοδος απόκτησης αναγνωριστικού μηνύματος
14.         el = ET.fromstring(msg)
15.         return el.attrib['message_id']
16.
17.     def getfeedname(msg):  # Μέθοδος απόκτησης ονόματος συλλογής
18.         el = ET.fromstring(msg)
19.         return el.attrib['feed_name']
20.
21.     def getaction(msg):  # Μέθοδος απόκτησης ενέργειας
22.         el = ET.fromstring(msg)
23.         return el.attrib['action']
24.
25.     def getsubscriptionid(msg):  # Μέθοδος απόκτησης αναγνωριστικού εγγραφή
    ς
```

```
        el = ET.fromstring(msg)
        return el.attrib['subscription_id']

    def getheader(msg):  # Μέθοδος απόκτησης επεκτάσεων επικεφαλίδων
        tree = ET.ElementTree(ET.fromstring(msg))
        root = tree.getroot()
        x = root[0][0].text
        return x

    def disc_resp(self, msg):

        x = TAXIIHTTPRequestHandler.getid(msg)

        try:

            # Δημιουργία Αντικειμένου τύπου Discovery Response
            discovery_response = tm10.DiscoveryResponse(
                message_id=tm10.generate_message_id(),
                in_response_to=x,
                service_instances="")

            # Δημιουργία Αντικειμένου τύπου Service Instance
            service_instance = tm10.ServiceInstance(
                service_type=SVC_POLL,
                services_version=VID_TAXII_SERVICES_10,
                protocol_binding=VID_TAXII_HTTPS_10,
                service_address='https://example.com/inbox/',
                message_bindings=[VID_TAXII_XML_10],
                inbox_service_accepted_content=[CB_STIX_XML_10],
                available=True,
                message='This is a sample inbox service instance')
            # Ενσωμάτωση αντικειμένου service_instance στην λίστα υπηρεσιών
, του
            # αντικειμένου discovery_response
            discovery_response.service_instances.append(service_instance)

            # Αποστολή του μηνύματος
            self.wfile.write(discovery_response.to_xml())

        except:
            # Δημιουργία αντικειμένου τύπου Status Message
            status_message1 = tm10.StatusMessage(
                message_id=tm10.generate_message_id(),
                in_response_to=x,
                status_type=ST_BAD_MESSAGE,
                status_detail='Machine-processable info here!',
                message='This is a message.')
            # Αποστολή μηνύματος
            self.wfile.write(status_message1.to_xml())

    def feed_info(self, msg):

        x = TAXIIHTTPRequestHandler.getid(msg)

        try:

            # Δημιουργία αντικειμένου τύπου PushMethod
            push_method1 = tm10.PushMethod(
                push_protocol=VID_TAXII_HTTP_10,
                push_message_bindings=[VID_TAXII_XML_10])

            # Δημιουργία αντικειμένου τύπου Polling Service Instance
```

```
87.             polling_service1 = tm10.PollingServiceInstance(
88.                 poll_protocol=VID_TAXII_HTTP_10,
89.                 poll_address='http://example.com/PollService/',
90.                 poll_message_bindings=[VID_TAXII_XML_10])
91.
92.             # Δημιουργία αντικειμένου τύπου Subscription Method
93.             subscription_service1 = tm10.SubscriptionMethod(
94.                 subscription_protocol=VID_TAXII_HTTP_10,
95.                 subscription_address='http://example.com/SubsService/',
96.                 subscription_message_bindings=[VID_TAXII_XML_10])
97.
98.             # Δημιουργία αντικειμένου τύπου Feed Information
99.             feed1 = tm10.FeedInformation(
100.                    feed_name='Feed1',
101.                    feed_description='Description of a feed',
102.                    supported_contents=[CB_STIX_XML_10],
103.                    available=True,
104.                    push_methods=[push_method1],
105.                    polling_service_instances=[polling_service1],
106.                    subscription_methods=[subscription_service1])
107.
108.                feed2 = tm10.FeedInformation(
109.                    feed_name='Feed1',
110.                    feed_description='Description of a feed',
111.                    supported_contents=[CB_STIX_XML_10],
112.                    available=True,
113.                    push_methods=[push_method1],
114.                    polling_service_instances=[polling_service1],
115.                    subscription_methods=[subscription_service1])
116.
117.                # Δημιουργία αντικειμένου τύπου Feed Information Response

118.                feed_information_response1 = tm10.FeedInformationResponse
    (
119.                    message_id=tm10.generate_message_id(),
120.                    in_response_to=x,
121.                    feed_informations=(feed1,feed2))  # Ενσωμάτωση αντικε
    ιμένων τύπου  Feed Information
122.                self.wfile.write(feed_information_response1.to_xml())
123.
124.
125.            except:
126.                # Δημιουργία αντικειμένου τύπου Status Message
127.                status_message1 = tm10.StatusMessage(
128.                    message_id=tm10.generate_message_id(),
129.                    in_response_to=x,
130.                    status_type=ST_BAD_MESSAGE,
131.                    status_detail='Machine-processable info here!',
132.                    message='This is a message.')
133.                self.wfile.write(status_message1.to_xml())
134.
135.        def poll_Resp(self, msg):
136.
137.            # Εισαγωγή αρχείου STIX
138.
139.
140.            # Αποθήκευση αναγνωριστικού μηνύματος και ονόματος συλλογής σ
    ε μεταβλητές
141.            x = TAXIIHTTPRequestHandler.getid(msg)
142.            y = TAXIIHTTPRequestHandler.getfeedname(msg)
143.
144.            xmldoc = xml.dom.minidom.parse('/home/ubuntu/Downloads/1.SXML
```

```
    ')
                xmldoc3 = xmldoc.toxml().encode()

                # Δημιουργία αντικειμένου τύπου Content Block
                cb1 = tm10.ContentBlock(content_binding=CB_STIX_XML_10, content=xmldoc3)

                # Δημιουργία αντικειμένου Poll Response
                poll_response1 = tm10.PollResponse(
                    message_id=tm10.generate_message_id(),
                    in_response_to=x,
                    feed_name="FeedPoll",
                    inclusive_begin_timestamp_label=datetime.datetime.now(tzutc()),
                    inclusive_end_timestamp_label=datetime.datetime.now(tzutc()),
                    subscription_id='SubsId001',
                    message="Poll Response Example",
                    content_blocks=[cb1])
                try:
                    # Έλεγχος ονόματος συλλογής εισερχόμενου αιτήματος
                    if poll_response1.feed_name == y:
                        print("Feed Exists")
                        self.wfile.write(poll_response1.to_xml())



                    else:
                        # Δημιουργία μηνύματος κατάστασης για την περίπτωση που το όνομα συλλογής
                        # δεδομένων δεν υπάρχει .
                        status_message1 = tm10.StatusMessage(
                            message_id=tm10.generate_message_id(),
                            in_response_to=x,
                            status_type=ST_NOT_FOUND,
                            status_detail='Machine-processable info here!',
                            message='Feed not found.')

                        self.wfile.write(status_message1.to_xml())
                except:
                    # Δημιουργία αντικειμένου τύπου Status Message
                    status_message2 = tm10.StatusMessage(
                        message_id=tm10.generate_message_id(),
                        in_response_to=x,
                        status_type=ST_BAD_MESSAGE,
                        status_detail='Machine-processable info here!',
                        message='Bad Message')

                    self.wfile.write(status_message2.to_xml())


            def manfeed_subresp(self, msg):
                print("Replying to manage feed request")

            # Αποθήκευση αναγνωριστικού μηνύματος, ονόματος συλλογής, ενέργειας και
            # αναγνωριστικού εγγραφής σε μεταβλητές.
                x = TAXIIHTTPRequestHandler.getid(msg)
                y = TAXIIHTTPRequestHandler.getfeedname(msg)
                z = TAXIIHTTPRequestHandler.getaction(msg)
                e = TAXIIHTTPRequestHandler.getsubscriptionid(msg)
```

```
201.
202.          # Δημιουργία αντικειμένου τύπου Poll Instance
203.              poll_instance1 = tm10.PollInstance(
204.              poll_protocol=VID_TAXII_HTTP_10,
205.              poll_address='http://example.com/poll',
206.              poll_message_bindings=[VID_TAXII_XML_10])
207.
208.          # Δημιουργία αντικειμένου τύπου Subscription Instance
209.              subscription_instance1 = tm10.SubscriptionInstance(
210.              subscription_id='SubsId053',
211.              poll_instances=[poll_instance1])
212.
213.          # Δημιουργία αντικειμένου τύπου Manage Feed Subscription Response

214.              manage_feed_subscription_response1 = tm10.ManageFeedSubscript
   ionResponse(
215.              message_id=tm10.generate_message_id(),
216.              in_response_to=x,
217.              feed_name='Feed',
218.              message='Your request will ne honored',
219.              subscription_instances=[subscription_instance1])
220.
221.              try:
222.
223.              # Έλεγχος ονόματος συλλογής εισερχόμενου αιτήματος
224.                  if y == manage_feed_subscription_response1.feed_name:
225.                      print("the requested action is : " + z)
226.                  # Έλεγχος ενέργειας
227.                      if z == "UNSUBSCRIBE":
228.                  # Έλεγχος αναγνωριστικού εγγραφής
229.                          if subscription_instance1.subscription_id == e:
230.                              manage_feed_subscription_response1.subscripti
   on_instances = []
231.                              self.wfile.write(manage_feed_subscription_res
   ponse1.to_xml())
232.                              print(manage_feed_subscription_response1.subs
   cription_instances)
233.                          else:
234.                              print("Wrong Subscription ID")
235.                      # Δημιουργία αντικειμένου τύπου Status Message
236.                              status_message = tm10.StatusMessage(
237.                              message_id=tm10.generate_message_id(),
238.                              in_response_to=x,
239.                              status_type=ST_NOT_FOUND,
240.                              status_detail='Machine-
   processable info here!',
241.                              message='wrong subscription ID.')
242.                              self.wfile.write(status_message.to_xml())
243.                      elif z == "SUBSCRIBE":
244.
245.                          subscription_instance2 = tm10.SubscriptionInstanc
   e(
246.                              subscription_id=tm10.generate_message_id(),
247.                              poll_instances=[poll_instance1])
248.                          manage_feed_subscription_response1.subscription_i
   nstances=[subscription_instance2]
249.                          print( manage_feed_subscription_response1.subscri
   ption_instances)
250.                          self.wfile.write(manage_feed_subscription_respons
   e1.to_xml())
251.
252.                      else:
```

```
253.                    # Δημιουργία αντικειμένου τύπου Status Message
254.                            status_message = tm10.StatusMessage(
255.                            message_id=tm10.generate_message_id(),
256.                            in_response_to=x,
257.                            status_type=ST_NOT_FOUND,
258.                            status_detail='Machine-processable info here!',
259.                            message='PENDING.')
260.                            self.wfile.write(status_message.to_xml())
261.
262.
263.                    else:
264.
265.            # Δημιουργία αντικειμένου τύπου Status Message
266.                            status_message2 = tm10.StatusMessage(
267.                            message_id=tm10.generate_message_id(),
268.                            in_response_to=x,
269.                            status_type=ST_NOT_FOUND,
270.                            status_detail='Machine-processable info here!',
271.                            message='Wrong feed name.')
272.                            self.wfile.write(status_message2.to_xml())
273.
274.                except:
275.        # Δημιουργία αντικειμένου τύπου Status Message
276.                    status_message3 = tm10.StatusMessage(
277.                    message_id=tm10.generate_message_id(),
278.                    in_response_to=x,
279.                    status_type=ST_BAD_MESSAGE,
280.                    status_detail='Machine-processable info here!',
281.                    message='Bad Message.')
282.                    self.wfile.write(status_message3.to_xml())
283.
284.
285.            def Inbox_resp(self, msg):
286.        # Αποθήκευση αναγνωριστικού μηνύματος
287.                x = TAXIIHTTPRequestHandler.getid(msg)
288.                if msg:
289.        # Δημιουργία αντικειμένου τύπου Status Message
290.                    status_message1 = tm10.StatusMessage(
291.                    message_id=tm10.generate_message_id(),
292.                    in_response_to=x,
293.                    status_type=ST_SUCCESS,
294.                    status_detail='Machine-processable info here!',
295.                    message='Message Received')
296.                    self.wfile.write(status_message1.to_xml())
297.                else:
298.        # Δημιουργία αντικειμένου τύπου Status Message
299.                    status_message2 = tm10.StatusMessage(
300.                    message_id=tm10.generate_message_id(),
301.                    in_response_to=x,
302.                    status_type=ST_FAILURE,
303.                    status_detail='Machine-processable info here!',
304.                    message='Failed')
305.                    self.wfile.write(status_message2.to_xml())
306.
307.
308.            def do_GET(self):
309.            # Έλεγχος εισερχόμενης ροής δεδομένων
310.                length = self.headers['content-length']
311.                data = self.rfile.read(int(length)).decode()
312.                return data
313.
314.
```

```
315.          def do_POST(self):
316.              try:
317.                  self.send_response(200)
318.
319.                  self.send_header('Content-type', 'text-html')
320.
321.                  self.end_headers()
322.
323.                  data = TAXIIHTTPRequestHandler.do_GET(self)
324.                  print(data)
325.                  header = TAXIIHTTPRequestHandler.getheader(data)
326.
327.              # Έλεγχος επεκτάσεων επικεφαλίδων
328.                  if header == "discovery":
329.                      TAXIIHTTPRequestHandler.disc_resp(self, data)
330.                  elif header == "feed_info":
331.                      TAXIIHTTPRequestHandler.feed_info(self, data)
332.                  elif header == "poll_request":
333.                      TAXIIHTTPRequestHandler.poll_Resp(self, data)
334.                  elif header == "manage_feed":
335.                      TAXIIHTTPRequestHandler.manfeed_subresp(self, data)
336.                  elif header == "Inbox":
337.                      TAXIIHTTPRequestHandler.Inbox_resp(self, data)
338.                  else:
339.                      return
340.
341.              except IOError:
342.                  self.send_error(404, 'file not found')
343.
344.
345.
346.
347.              # Μέθοδος εκκίνησης Server
348.
349.
350.      def run(server_class=http.server.HTTPServer, handler_class=http.serve
   r.BaseHTTPRequestHandler):
351.              print('http server is starting...')
352.
353.              server_address = ('10.0.2.15', 80)
354.              httpd = server_class(server_address, TAXIIHTTPRequestHandler)

355.              print('http server is running...')
356.              httpd.serve_forever()
357.
358.
359.      if __name__ == '__main__':
360.          run()
```

Πίνακας Α.1: Κώδικας Υλοποίησης HTTP Server

A.2 Κώδικας Υλοποίησης HTTP Client

Ακολουθεί ο κώδικας υλοποίησης του HTTP Client, ο οποίος είναι υπεύθυνος για την αποστολή των αιτημάτων TAXII.

```
1. # Εισαγωγή απαραίτητων βιβλιοθηκών
2.
3. import datetime
4. from dateutil.tz import tzutc
5. import libtaxii.clients as tc
6. import libtaxii.messages_10 as tm10
7. from libtaxii.constants import *
8. import xml.dom.minidom
9. import sys
10.
11.
12.
13.
14.# Δημιουργία αντικειμένου HTTP Client
15.client = tc.HttpClient()
16.
17.
18.def disc_req():
19.    print("Requesting available services")
20.
21.
22.
23.# Δημιουργία αντικειμένου τύπου Discovery Request
24.    discovery_request = tm10.DiscoveryRequest(
25.        message_id=tm10.generate_message_id(),
26.        extended_headers={'service': 'discovery'}, )
27.
28.# Κλήση μεθόδου call_taxii_service
29.    rsp = client.call_taxii_service('10.0.2.15', "", VID_TAXII_XML_10, disc
   overy_request.to_xml(), port=80)
30.
31.# Αποθήκευση εισερχόμενης ροής
32.    data_received = rsp.read().decode()
33.    print(data_received)
34.
35.
36.def feed_inforeq():
37.    print("Requesting feed informations")
38.
39.# Δημιουργία αντικειμένου Feed Information request
40.    feed_information_request = tm10.FeedInformationRequest(
41.        message_id=tm10.generate_message_id(),
42.        extended_headers={'service': 'feed_info'})
43.
44. # Κλήση μεθόδου call_taxii_service
45.    rsp = client.call_taxii_service('10.0.2.15', "", VID_TAXII_XML_10, feed
   _information_request.to_xml(),port=80)
```

```
46.
47. # Αποθήκευση εισερχόμενης ροής
48.     data_received = rsp.read().decode()
49.     print(data_received)
50.
51.
52. def poll_req():
53.     print("Requesting a poll")
54.
55. # Δημιουργία αντικειμένου τύπου Poll Request
56.     poll_request1=tm10.PollRequest(
57.         message_id=tm10.generate_message_id(),
58.         feed_name='FeedPoll',
59.         exclusive_begin_timestamp_label=datetime.datetime.now(tzutc()),
60.         inclusive_end_timestamp_label = datetime.datetime.now(tzutc()),
61.         subscription_id = 'SubsId002', content_bindings = [CB_STIX_XML_10],

62.         extended_headers = {'service': 'poll_request'})
63.
64. # Κλήση μεθόδου call_taxii_service
65.     rsp = client.call_taxii_service('10.0.2.15', "", VID_TAXII_XML_10, poll
    _request1.to_xml(), port=80)
66.
67. # Αποθήκευση εισερχόμενης ροής
68.     data_received = rsp.read().decode()
69.     print(data_received)
70.
71.
72. def man_feed_subreq():
73.     print("Requesting manage feed")
74.
75. # Δημιουργία αντικειμένου τύπου Delivery Parameters
76.     delivery_parameters1 = tm10.DeliveryParameters(
77.         inbox_protocol=VID_TAXII_HTTP_10,
78.         inbox_address='http://example.com/inbox',
79.         delivery_message_binding=VID_TAXII_XML_10,
80.         content_bindings=[CB_STIX_XML_10])
81.
82. # Δημιουργία αντικειμένου Manage Feed Subscription Response
83.     manage_feed_subscription_request1 = tm10.ManageFeedSubscriptionRequest(

84.         message_id=tm10.generate_message_id(),
85.         feed_name='Feed',
86.         action=ACT_SUBSCRIBE,
87.         subscription_id='',
88.         delivery_parameters=delivery_parameters1,
89.         extended_headers={'service': 'manage_feed'})
90.
91.
92.
93.
94. # Κλήση μεθόδου call_taxii_service
95.     rsp=client.call_taxii_service('10.0.2.15',"",VID_TAXII_XML_10,manage_fe
    ed_subscription_request1.to_xml(),port=80 )
96.
97. # Αποθήκευση εισερχόμενης ροής
98.     data_received = rsp.read().decode()
99.     print(data_received)
100.
101.
102.       def InboxMes():
103.
```

```
104.      # Εισαγωγή αρχείου STIX
105.          print("Sending an inbox message to the server with Stix Content")

106.          xmldoc = xml.dom.minidom.parse('/home/ubuntu/Downloads/1.SXML')
107.          xmldoc3 = xmldoc.toxml().encode()
108.
109.      # Δημιουργία αντικειμένου τύπου Content Block
110.          cb1 = tm10.ContentBlock(content_binding=CB_STIX_XML_10, content=x
   mldoc3)
111.
112.      # Δημιουργία αντικειμένου τύπου Subscription Information
113.          subscription_information1 = tm10.SubscriptionInformation(
114.              feed_name='SomeFeedName',
115.              subscription_id='SubsId021',
116.              inclusive_begin_timestamp_label=datetime.datetime.now(tzutc()
   ),
117.              inclusive_end_timestamp_label=datetime.datetime.now(tzutc()))

118.
119.      # Δημιουργία αντικειμένου τύπου Inbox Message
120.
121.          inbox_message1 = tm10.InboxMessage(
122.              message_id=tm10.generate_message_id(),
123.              message="",
124.              subscription_information=subscription_information1,
125.              extended_headers={'service': 'Inbox'},
126.              content_blocks=[cb1])
127.
128.      # Κλήση μεθόδου call_taxii_service
129.
130.          rsp = client.call_taxii_service('10.0.2.15', "", VID_TAXII_XML_10
   , inbox_message1.to_xml(), port=80)
131.
132.      # Αποθήκευση εισερχόμενης ροής
133.
134.          data_received = rsp.read().decode()
135.          print(data_received)
136.
137.      if __name__ == '__main__':
138.
139.      # Κλήση μεθόδων
140.          globals()[sys.argv[1]]()
```

Πίνακας Α.2: Κώδικας Υλοποίησης HTTP Client

Παράρτημα Β
Πεδία σώματος μηνυμάτων TAXII

Στο ακόλουθο παράρτημα περιέχονται οι τύποι πεδίων σώματος για τους διαφορετικούς τύπους μηνυμάτων TAXII.

Τύπος Μηνύματος TAXII	Πεδία
Status Message	• Status Type: Παίρνει προκαθορισμένες τιμές μέσα από την ακόλουθη λίστα (Success, Asynchronous Poll Error, Denied, Bad Message, Destination Collection Error, Failure, Invalid Response Part, Network Error, Not Found, Pending, Polling Not Supported, Retry, Unauthorized, Unsupported Message Binding, Unsupported Content Binding, Unsupported Protocol Binding, Unsupported Query Format). • Status Details: Περιέχει επιπλέον πληροφορίες σχετικά με το μήνυμα κατάστασης σε μορφή γλώσσας μηχανής. • Message: Περιέχει επιπλέον πληροφορίες σχετικά με το μήνυμα κατάστασης. Συνήθως απευθύνεται σε ανθρώπους.
Discovery Request	Δεν περιέχει πεδία σώματος
Discovery Response	• Service Instance: Αντιπροσωπεύει μια διαθέσιμη υπηρεσία TAXII. • Service Type: Περιέχει τον τύπο της διαθέσιμης υπηρεσίας (Poll, Inbox, Collection Management, Discovery). • Service Version: Περιέχει την έκδοση της παρεχόμενης υπηρεσίας. • Protocol Binding: Περιέχει τον τύπο πρωτοκόλλου που υποστηρίζεται από

	την υπηρεσία. • Service Address: Περιέχει την διεύθυνση στην οποία βρίσκεται ο δαίμονας TAXII που κατέχει την συγκεκριμένη υπηρεσία. • Message Binding: Περιέχει την μορφή μηνυμάτων που υποστηρίζει η συγκριμένη υπηρεσία. • Supported Query: Περιέχει τον συγκεκριμένο τύπο ερωτημάτων που υποστηρίζει η υπηρεσία. • Query Format ID: Περιέχει το αναγνωριστικό του τύπου ερωτημάτων. • Inbox Service Accepted Content: Χρησιμοποιείται μόνο εάν η διαθέσιμη υπηρεσία είναι τύπου Inbox Service, για να υποδείξει τους τύπους περιεχομένου που δέχεται η συγκεκριμένη υπηρεσία. • Subtype: Περιέχει υποτύπους περιεχομένου για συγκεκριμένους τύπους περιεχομένου. • Available: Υποδεικνύει εάν ο αιτών έχει δικαίωμα να χρησιμοποιήσει την υπηρεσία. • Message: Περιέχει επιπλέον πληροφορίες σχετικά με την παρεχόμενη υπηρεσία.
Collection Information Request	Δεν περιέχει πεδία σώματος
Collection Information Response	• Collection Information: Ορίζει μια συλλογή δεδομένων. • Collection Name: Το όνομα της συλλογής δεδομένων. • Collection Type: Υποδεικνύει τον τύπο της συλλογής δεδομένων (Data Feed, Data Set). • Collection Description: Περιγραφή της συλλογής δεδομένων. • Collection Volume: Περιέχει τον αριθμό εγγραφών που προστίθενται καθημερινά στην συλλογή. • Supported Content: Περιέχει τα αναγνωριστικά των τύπων περιεχομένων που μπορούν να εντοπιστούν στην συγκεκριμένη συλλογή δεδομένων. • Subtype: Περιέχει υποτύπους περιεχομένου για συγκεκριμένους τύπους περιεχομένου.

	• Available: Υποδεικνύει εάν ο αιτών έχει δικαίωμα να χρησιμοποιήσει την συλλογή δεδομένων. • Push Method: Περιέχει τα πρωτόκολλα που μπορούν να χρησιμοποιηθούν για τη προώθηση περιεχομένου της συγκεκριμένης συλλογής. • Push Protocol: Περιέχει τους τύπους πρωτοκόλλων που υποστηρίζονται για την προώθηση περιεχομένου της συγκεκριμένης συλλογής. • Polling Service Instance: Περιέχει την διεύθυνση της υπηρεσίας Poll η οποία κατέχει την συγκεκριμένη συλλογή. • Poll Protocol: Περιέχει τις μορφές πρωτοκόλλων που υποστηρίζονται από την συγκεκριμένη υπηρεσία. • Poll Address: Περιέχει την διεύθυνση του δαίμονα TAXII που κατέχει την συγκεκριμένη υπηρεσία Poll. • Poll Message Binding: Περιέχει τους τύπους μηνυμάτων που υποστηρίζονται από την συγκεκριμένη υπηρεσία Poll. • Subscription Method: Περιέχει την διεύθυνση και τα πρωτόκολλα του δαίμονα TAXII ο οποίος είναι υπεύθυνος για την διαχείριση εγγραφών της συγκεκριμένης συλλογής. • Subscription Protocol: Περιέχει τους τύπους πρωτοκόλλων που υποστηρίζονται από την υπηρεσία διαχείρισης συλλογών. • Subscription Address: Περιέχει την διεύθυνση επικοινωνίας του δαίμονα TAXII που κατέχει την υπηρεσία διαχείρισης συλλογών. • Subscription Message Binding: Περιέχει τους τύπους μηνυμάτων που υποστηρίζονται από την υπηρεσία διαχείρισης συλλογών. • Receiving Inbox Service: Περιέχει την διεύθυνση μιας υπηρεσίας εισερχομένων, στην οποία μπορεί να συνδεθεί κάποιος για να προωθήσει περιεχόμενο σε κάποια συλλογή δεδομένων. • Inbox Protocol: Περιέχει τους τύπους πρωτοκόλλων που υποστηρίζονται από την υπηρεσία εισερχομένων.

	• Inbox Address: Περιέχει την διεύθυνση του δαίμονα TAXII που κατέχει την υπηρεσία εισερχομένων. • Inbox Message Binding: Περιέχει τους τύπους μηνυμάτων που υποστηρίζονται από την υπηρεσία εισερχομένων. • Supported Content: Περιέχει τους τύπους αναγνωριστικών περιεχομένου από τους οποίους δέχεται περιεχόμενο η υπηρεσία εισερχομένων. • Subtype: Περιέχει υποτύπους περιεχομένου για συγκεκριμένους τύπους περιεχομένου.
Manage Collection Subscription Request	• Collection Name: Περιέχει το όνομα της συλλογής δεδομένων. • Action:Περιέχει την αιτούμενη ενέργεια. Παίρνει τιμές μέσα από την λίστα (SUBSCRIBE, UNSUBSCRIBE, PAUSE, RESUME, STATUS) • Subscription ID: Περιέχει το αναγνωριστικό εγγραφής. • Subscription Parameters: Περιέχει τις παραμέτρους εγγραφής. • Content Binding: Περιέχει το αναγνωριστικό περιεχομένου. Το αναγνωριστικό περιεχομένου δείχνει των τύπο δεδομένων που επιθυμεί να αποκτήσει ο αιτών. • Query: Περιέχει κάποιο ερώτημα συσχετιζόμενο με την εγγραφή. • Delivery Parameters: Προσδιορίζει τις παραμέτρους για την προώθηση περιεχομένου. • Inbox Protocol: Προσδιορίζει το πρωτόκολλο που χρησιμοποιείται κατά την προώθηση περιεχομένου. • Inbox Address: Περιέχει την διεύθυνση που χρησιμοποιείται κατά την επικοινωνία με τον δαίμονα TAXII που κατέχει την υπηρεσία εισερχομένων. • Delivery Message Binding: Περιέχει την μορφή μηνυμάτων που χρησιμοποιείται κατά την προώθηση περιεχομένου από μια εγγραφή.
Manage Collection Subscription Response	• Collection Name: Περιέχει το όνομα της συλλογής δεδομένων. • Message: Ένα μήνυμα σχετικά με την

	απάντηση εγγραφής. • Subscription Instance: Περιέχει πληροφορίες σχετικές με υπάρχουσες εγγραφές του αιτούντα σε συγκεκριμένη συλλογή δεδομένων. • Subscription ID: Περιέχει το αναγνωριστικό εγγραφής. • Status: Περιέχει την κατάσταση της εγγραφής, Παίρνει τιμές μέσα από την λίστα (Active, Paused, Unsubscribed). • Subscription Parameters: Περιέχει ένα αντίγραφο των παραμέτρων εγγραφής, του αιτήματος Manage Collection Subscription. • Poll Instance: Αντιπροσωπεύει μια υπηρεσία Poll με την οποία μπορεί να επικοινωνήσει κάποιος, για να αποκτήσει περιεχόμενο σχετικό με την συγκεκριμένη εγγραφή. • Poll Protocol: Περιέχει τον τύπο πρωτοκόλλων που υποστηρίζεται από την υπηρεσία Poll. • Poll Address: Περιέχει την διεύθυνση του δαίμονα TAXII που περιέχει την υπηρεσία Poll. • Poll Message Binding: Περιέχει τύπους μηνυμάτων που μπορούν να χρησιμοποιηθούν κατά την επικοινωνία με την υπηρεσία Poll.
Poll Request	• Collection Name: Περιέχει το όνομα της συλλογής δεδομένων. • Exclusive Begin Timestamp Label: Περιέχει μια ετικέτα χρονοσφραγίδας υποδεικνύοντας την αρχή του εύρους των δεδομένων της συλλογής, που ο αιτών επιθυμεί να αποκτήσει. • Exclusive End Timestamp Label: Περιέχει μια ετικέτα χρονοσφραγίδας υποδεικνύοντας το τέλος του εύρους των δεδομένων της συλλογής, που ο αιτών επιθυμεί να αποκτήσει • Subscription ID: Περιέχει το αναγνωριστικό εγγραφής. • Poll Parameters: Χρησιμοποιείται όταν το πεδίο Subscription ID δεν υπάρχει υποδεικνύοντας το περιεχόμενο που πρέπει να σταλεί μέσω του Poll Response. • Content Binding: Περιέχει το αναγνωριστικό περιεχομένου. Το

	αναγνωριστικό περιεχομένου δείχνει των τύπο δεδομένων που επιθυμεί να αποκτήσει ο αιτών. • Query: Περιέχει κάποιο ερώτημα συσχετιζόμενο με την εγγραφή. • Delivery Parameters: Προσδιορίζει τις παραμέτρους για την προώθηση περιεχομένου. • Inbox Protocol: Προσδιορίζει το πρωτόκολλο που χρησιμοποιείται κατά την προώθηση περιεχομένου. • Inbox Address: Περιέχει την διεύθυνση που χρησιμοποιείται κατά την επικοινωνία με τον δαίμονα TAXII που κατέχει την υπηρεσία εισερχομένων. • Delivery Message Binding: Περιέχει την μορφή μηνυμάτων που χρησιμοποιείται κατά την προώθηση περιεχομένου από μια εγγραφή.
Poll Response	• Collection Name: Περιέχει το όνομα της συλλογής δεδομένων της οποίας περιεχόμενο προωθείται. • Subscription ID: Περιέχει το αναγνωριστικό εγγραφής της οποίας περιεχόμενο παρέχεται. • Exclusive Begin Timestamp: Περιέχει χρονοσφραγίδες. • Exclusive End Timestamp: Περιέχει χρονοσφραγίδες. • More: Χρησιμοποιείται στην περίπτωση πολυτμηματικής Υλοποίησης της υπηρεσίας Poll. • Message: Περιέχει επιπλέον πληροφορίες για τον παραλήπτη. • Content Block: Περιέχει τμήματα περιεχομένου μιας συλλογής δεδομένων.
Inbox Message	• Destination Collection Name: Περιέχει το όνομα της συλλογής δεδομένων για την οποία προορίζεται το μήνυμα. • Message: Περιέχει πληροφορίες για τον παραλήπτη του μηνύματος. • Subscription Information: Αυτό το πεδίο χρησιμοποιείται μόνο όταν το εισερχόμενο μήνυμα παρέχει περιεχόμενο μιας υπάρχουσας εγγραφής. • Collection Name: Περιέχει το όνομα της

	συλλογής δεδομένων της οποίας περιεχόμενο παρέχεται. • Subscription ID: Περιέχει το αναγνωριστικό εγγραφής της οποίας περιεχόμενο παρέχεται. • Exclusive Begin Timestamp: Περιέχει χρονοσφραγίδες έναρξης. • Exclusive End Timestamp: Περιέχει χρονοσφραγίδες λήξης.

Πίνακας Β.1: Πεδία σώματος μηνυμάτων TAXII