Πανεπιστήμιο Πειραιώς
Τμήμα Ψηφιακών Συστημάτων

Πρόγραμμα Μεταπτυχιακών Σπουδών
«Τεχνοοικονομική Διοίκηση & Ασφάλεια Ψηφιακών Συστημάτων»

Fuzzing

An XMPP Fuzzer

Διπλωματική Εργασία

Νάκος Παντελής
Α.Μ MTE 1119

Επιβλέπων
Δρ. Ξενάκης Χρήστος

Πειραιάς 2013

Περίληψη

Το fuzzing είναι μια τεχνική για την εύρεση αδυναμιών στο λογισμικό στέλνοντας ¨παραμορφωμένα¨ δεδομένα ως είσοδο σε αυτό χρησιμοποιώντας έναν αυτοματοποιημένο τρόπο. Η εύρεση και η κατάργηση των αδυναμιών αυτών είναι σημαντική για τους προγραμματιστές καθώς η εκμετάλλευσή τους από κακόβουλους χρήστες μπορεί να προκαλέσει κενά ασφάλειας. Το αποτέλεσμα από την εκμετάλλευση μιας αδυναμίας μπορεί να είναι η παραβίαση της εφαρμογής ή και του λειτουργικού συστήματος στο οποίο αυτή είναι εγκατεστημένη και η μη εξουσιοδοτημένη πρόσβαση σε δεδομένα και πληροφορίες. Το fuzzing αποτελεί ένα γρήγορο και σχετικά φθηνό εργαλείο για την εύρεση σφαλμάτων στο λογισμικό. Παρά το γεγονός ότι η φιλοσοφία λειτουργίας του fuzzing είναι απλή, η δημιουργία ενός αποδοτικού και ακριβείς fuzzer μπορεί να αποτελεί μία απαιτητική διαδικασία. Η αποδοτικότητα και η ακρίβεια του fuzzer συνήθως στηρίζονται στην ανάλυση του υπό εξέταση πρωτοκόλλου και στην αποτελεσματικότητα των ¨παραμορφωμένων¨ δεδομένων που θα χρησιμοποιηθούν. Σκοπός της εργασίας αυτής είναι να παρουσιαστούν οι μέθοδοι που χρησιμοποιούνται για το αποτελεσματικό fuzzing ενός πρωτοκόλλου καθώς και η ανάπτυξη ενός fuzzer για το Extensible Messaging and Presence Protocol (http://www.xmpp.org/) .

Το Extensible Messaging and Presence Protocol (XMPP) χρησιμοποιείται όλο και περισσότερο σε εφαρμογές για την ανταλλαγή μηνυμάτων σε πραγματικό χρόνο. Ενσωματώνει πολλά χαρακτηριστικά δικτυακών πρωτοκόλλων που σε συνδυασμό με την διαδεδομένη χρήση του το καθιστούν ελκυστικό ¨στόχο¨ για επιθέσεις κακόβουλων χρηστών.

Η ανάπτυξη του fuzzer έγινε με την χρήση της γλώσσας προγραμματισμού Python (http://www.python.org/) και της βιβλιοθήκης SleekXMPP (http://www.sleekxmpp.com/). Ο fuzzer διαβάζει από ένα αρχείο τις παραμορφωμένες ακολουθίες χαρακτήρων και με την χρήση της βιβλιοθήκης επικοινωνεί με τον εξυπηρετητή που προσφέρει υπηρεσίες XMPP. Ο έλεγχος που πραγματοποιεί του εργαλείο γίνεται σε δύο διαφορετικά σημεία όπου εισάγονται οι παραμορφωμένες ακολουθίες χαρακτήρων και εξετάζεται η συμπεριφορά του εξυπηρετητή.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

1. Εισαγωγή

Το fuzzing είναι μια δυναμική τεχνική για την εύρεση αδυναμιών στο λογισμικό στέλνοντας ¨τυχαία¨ δεδομένα σε αυτό. Το fuzzing γίνεται με την αποστολή ειδικά διαμορφωμένων πακέτων στην εφαρμογή για την εύρεση αδυναμιών, οι οποίες με άλλα εργαλεία δεν είναι δυνατόν να εντοπιστούν. Ακόμα και η είσοδος τυχαίων χαρακτήρων μπορεί να θεωρηθεί fuzzing. Το κλειδί για επιτυχημένο fuzzing είναι ο συνδυασμός της τυχαιότητας, της καλής γνώσης του πρωτοκόλλου υπό δοκιμή και της αυτοματοποίησης της διαδικασίας. Ο παραπάνω συνδυασμός έχει την δυνατότητα να αποκαλύψει περισσότερες αδυναμίες με μικρότερο κόστος σε σύγκριση με άλλες μεθόδους.

Ο όρος fuzzing αρχικά εμφανίστηκε το 1989. Ο καθηγητής Barton Miller και η ομάδα του ανέπτυξαν έναν fuzzer για να ¨εξετάσουν¨ την στιβαρότητα των εφαρμογών του UNIX. Σκοπός τους δεν ήταν να αξιολογήσουν την ασφάλεια των εφαρμογών αλλά γενικά να αξιολογήσουν την ποιότητα και την αξιοπιστία του κώδικα. Η ονομασία fuzzing αρχικά δόθηκε από ένα ερευνητικό πρόγραμμα του πανεπιστημίου Wisconsin-Madison. Στον ακαδημαϊκό χώρο ο πιο κοντινός όρος στο fuzzing είναι ο *Boundary Value Analysis* όπου μια εφαρμογή δοκιμάζεται με τις οριακές τιμές που μπορεί να δεχτεί σαν είσοδο.[1]

Σήμερα, το fuzzing χρησιμοποιείται από τους ερευνητές ασφάλειας για την εύρεση αδυναμιών και την διόρθωση τους και για την διασφάλιση της ποιότητας της εφαρμογής και από κακόβουλους χρήστες για την εύρεση αδυναμιών που θα τους δώσουν πρόσβαση σε δεδομένα. Οι εταιρείες παραγωγής λογισμικού έχουν ενσωματώσει το fuzzing στον κύκλο παραγωγής των εφαρμογών τους με σκοπό την παραγωγή αξιόπιστου και ασφαλέστερου λογισμικού.

1.1. Τύποι Fuzzing

Στο fuzzing διακρίνονται τρεις διαφορετικοί τύποι:

- White box Fuzzing: Στην περίπτωση αυτή ο πηγαίος κώδικας της εφαρμογής είναι γνωστός στον αναλυτή. Σε αυτό τον τύπο ανήκουν οι περιπτώσεις όπου το fuzzing διενεργείται από τις εταιρείες παραγωγής λογισμικού και σε εφαρμογές ανοιχτού κώδικα (open-source).

- Grey box Fuzzing: Κατά το grey box fuzzing ο πηγαίος κώδικας δεν είναι διαθέσιμος. Οι πληροφορίες για την λειτουργία της εφαρμογής γίνονται γνωστές με μεθόδους reverse engineering και την χρήση debuggers. Ο πηγαίος κώδικας των εμπορικών εφαρμογών συνήθως δεν είναι διαθέσιμος οπότε και χρησιμοποιείται το grey box fuzzing.

- Black box fuzzing: Στο black box fuzzing δεν είναι διαθέσιμος ούτε ο πηγαίος κώδικας ούτε η χρήση τεχνικών reverse engineering.

1.2. Σημεία επίθεσης

Πριν ξεκινήσει η διαδικασία του fuzzing πρέπει να βρεθούν τα σημεία εκείνα στα οποία μπορεί ο επιτιθέμενος να εισάγει κακόβουλα δεδομένα στην εφαρμογή πχ οι εξωτερικές διεπαφές. Τα σημεία αυτά συνήθως είναι οι parsers που χειρίζονται το πρωτόκολλο και τα διάφορα αρχεία όπως επίσης και οι διεπαφές προγραμματισμού των εφαρμογών (APIs). Ο

fuzzer που αναπτύχθηκε για τον σκοπό της εργασίας αυτής χρησιμοποιεί ως σημείο επίθεσης την απομακρυσμένη σύνδεση μέσω δικτύου.

1.3. Είδη fuzzer

Υπάρχουν δύο βασικά είδη fuzzer:

- Mutation based: Αυτού του είδους οι fuzzers ξεκινούν τις δοκιμές με ένα ¨έγκυρο¨ δείγμα του πρωτοκόλλου ή της δομής δεδομένων που εξετάζεται και εν συνεχεία μεταβάλει στο δείγμα αυτό κάθε byte, word, dword και συμβολοσειρά. Αυτό είναι το πιο απλό και εύκολο στην υλοποίηση είδος fuzzer. Ο fuzzer το μόνο που χρειάζεται να κάνει είναι να μεταβάλλει τα δεδομένα και να τα στείλει στην εφαρμογή. Το μειονέκτημα αυτού του είδους είναι ότι πολλά από τα πακέτα που θα σταλούν στην εφαρμογή θα απορριφθούν ως προς την εγκυρότητα τους με συνέπεια να μειώνεται η αποδοτικότητα του fuzzer.[1]

- Generation based: Αυτό είναι ένα πιο προχωρημένο είδος fuzzer όπου απαιτείται η καλή γνώση του πρωτοκόλλου που θα εξεταστεί έτσι ώστε τα δεδομένα που θα στέλνει ο fuzzer προς την εφαρμογή να μην απορρίπτονται λόγω λάθους στο πρωτόκολλο επικοινωνίας. Κατά την χρήση ενός πρωτοκόλλου κάποια πεδία παραμένουν σταθερά ενώ κάποια άλλα μεταβάλλονται. Ο fuzzer διατηρεί τα πεδία που χρειάζεται σταθερά και πραγματοποιεί δοκιμές στα υπόλοιπα. Η αποτελεσματικότητα του fuzzer στηρίζεται στην εύρεση των πεδίων εκείνων πού είναι πιο πιθανό να βρεθεί μία αδυναμία. Το μειονέκτημα αυτού του είδους είναι ότι απαιτεί αρκετό χρόνο για την προετοιμασία του fuzzer ώστε να χρησιμοποιεί χωρίς λάθη το πρωτόκολλο.[1]

1.4. Αξιολόγηση αδυναμιών

Παρά την ύπαρξη αυτοματοποιημένων εργαλείων για την κατανομή και αξιολόγηση των αδυναμιών που μπορεί να βρεθούν κατά το fuzzing, είναι απαραίτητη και η αξιολόγηση από τον ερευνητή για να διαπιστωθεί η σημαντικότητα της αδυναμίας. Για την αποφυγή της χρονοβόρας διαδικασίας της αξιολόγησης των αδυναμιών, ο fuzzer θα πρέπει να υλοποιηθεί έτσι ώστε να μην εντοπίζει τα ίδια σφάλματα επανειλημμένα. Αυτό μπορεί να επιτευχθεί με την ομαδοποίηση των εισόδων δεδομένων στην εφαρμογή που μπορεί να προκαλέσουν το ίδιο σφάλμα.

1.5. Εργαλεία fuzzing

Στο διαδίκτυο είναι διαθέσιμη μια μεγάλη ποικιλία από fuzzers οι οποίοι καλύπτουν μεγάλο αριθμό πρωτοκόλλων. Τα εργαλεία αυτά μπορούν να χωριστούν σε δύο κατηγορίες, η μια κατηγορία περιέχει τους fuzzers που εξετάζουν ένα συγκεκριμένο πρωτόκολλο και την κατηγορία με τα fuzzing frameworks. Τα fuzzing frameworks δίνουν την δυνατότητα ανάπτυξης ενός fuzzer χρησιμοποιώντας δοκιμασμένα παραμορφωμένα δεδομένα, τα οποία είναι πιθανό να αποκαλύψουν αδυναμίες. Επίσης προσφέρουν έναν ενσωματωμένο debugger για την αναγνώριση των σφαλμάτων, όπως επίσης ενσωματώνουν κάποια modules για την επικοινωνία με την εφαρμογή.

1.6. Σκοπός της εργασίας

Σκοπός της εργασίας είναι να παρουσιάσει τα πλεονεκτήματα και τους περιορισμούς που έχει το fuzzing. Για την καλύτερη κατανόηση αυτών στα πλαίσια της εργασίας αναπτύχθηκε ένας fuzzer για την αξιολόγηση των εξυπηρετητών (servers) που προσφέρουν υπηρεσίες XMPP.

Επίσης, περιγράφεται η διαδικασία που ακολουθήθηκε για την μελέτη του πρωτοκόλλου και την ανάπτυξη του fuzzer, έτσι ώστε να μπορεί να χρησιμοποιηθεί σαν ¨οδηγός¨ για περαιτέρω έρευνα.

1.7. Εύρος, Περιορισμοί και Όρια

Η παρούσα εργασία αναφέρεται στο fuzzing. Για την ανάπτυξη του fuzzer τέθηκαν οι εξής περιορισμοί:

- Το πρωτόκολλο που θα εξεταστεί θα είναι ανοιχτό
- Το εργαλείο θα πρέπει να αναπτυχθεί από την αρχή, χωρίς την χρήση κάποιου framework
- Το εργαλείο θα εξετάζει εξυπηρετητές που προσφέρουν υπηρεσίες μέσω του Internet
- Οι εξυπηρετητές που θα ελεγχθούν θα πρέπει να είναι ανοιχτού κώδικα
- Το εργαλείο θα πρέπει να είναι παραμετροποιήσιμο και επεκτάσιμο έτσι ώστε να είναι δυνατόν να προστεθούν λειτουργίες

1.8. Προσέγγιση του προβλήματος

Για την ολοκλήρωση της εργασίας, αρχικά έπρεπε να γίνει μια εκτεταμένη αναζήτηση πηγών σχετικά με το fuzzing έτσι ώστε να υπάρχει μία ολοκληρωμένη άποψη για το αντικείμενο αυτό. Στην συνέχεια επιλέχθηκε ένα πρωτόκολλο κατάλληλο για την εφαρμογή των τεχνικών του fuzzing. Εν συνεχεία επιλέχθηκαν οι εξυπηρετητές οι οποίοι χρησιμοποιούν το συγκεκριμένο πρωτόκολλο και είναι ανοιχτού κώδικα. Τέλος έγινε η ανάπτυξη του fuzzer για τον έλεγχο των επιλεγμένων εξυπηρετητών.

Με την ολοκλήρωση της ανάπτυξης του εργαλείου έγιναν εκτεταμένες δοκιμές για την ορθή λειτουργία του και την αποτελεσματικότητα του στους επιλεγμένους εξυπηρετητές.

1.9. Δομή της εργασίας

Σκοπός του κεφαλαίου αυτού είναι η εισαγωγή στις βασικές έννοιες του fuzzing καθώς και η σύντομη περιγραφή της διαδικασίας που ακολουθήθηκε για την ολοκλήρωση της εργασίας. Στο επόμενο κεφάλαιο θα ακολουθήσει η περιγραφή της διαδικασίας της αξιολόγησης και τα κριτήρια κάτω από τα οποία έγιναν οι επιλογές του πρωτοκόλλου, του fuzzer και των εξυπηρετητών. Στο τρίτο κεφάλαιο θα γίνει μια εκτεταμένη ανάλυση του πρωτοκόλλου που επιλέχθηκε. Στο τέταρτο κεφάλαιο θα ακολουθήσει η ανάλυση του fuzzer που αναπτύχθηκε και οι λειτουργίες που προσφέρει. Στο πέμπτο κεφάλαιο θα γίνει η τεχνική ανάλυση του fuzzer και του πηγαίου κώδικα και τέλος στο έκτο κεφάλαιο θα παρουσιαστούν τα συμπεράσματα και οι προοπτικές για την εξέλιξη του συγκεκριμένου εργαλείου.

2. Αξιολόγηση

2.1. Πρωτόκολλο

Η επιλογή του πρωτοκόλλου αποτελεί σημαντικό βήμα για την εξέλιξη της εργασίας καθώς επηρεάζει το σύνολο της. Η αξιολόγηση των επιλογών έγινε με τα κριτήρια που έχουν τεθεί στην παράγραφο 1.7. Στην συνέχεια αναλύονται τα κριτήρια και η αξιολόγηση των διαθέσιμων επιλογών έτσι ώστε να καταλήξουμε στην κατάλληλη επιλογή.

2.1.1. Κριτήρια

Τα κριτήρια για την επιλογή του κατάλληλου πρωτοκόλλου είναι τα εξής :

- Διαθεσιμότητα υλοποιήσεων: Η πιθανότητα εύρεσης αδυναμιών είναι μεγαλύτερη σε πρωτόκολλα λιγότερο διαδεδομένα αλλά αυτό περιορίζει των αριθμό των υλοποιήσεων που είναι διαθέσιμες. Αντιθέτως, πρωτόκολλα που είναι πιο διαδεδομένα έχουν υλοποιηθεί σε μεγάλο αριθμό εφαρμογών ανοιχτού κώδικα και προσφέρονται δωρεάν.

- Πολυπλοκότητα: Όσο περισσότερες δυνατότητες προσφέρει ένα πρωτόκολλο τόσο αυξάνεται και η πολυπλοκότητα του και κατά συνέπεια οι πιθανότητες εμφάνισης αδυναμιών. Το μέγεθος της πολυπλοκότητας μπορεί να υπολογιστεί προσεγγιστικά από τον αριθμό των εντολών, των καταστάσεων και των επιλογών που προσφέρει το πρωτόκολλο.

- Fuzzers: Η μη ύπαρξη fuzzer για το πρωτόκολλο που θα επιλεχθεί είναι πολύ σημαντική και θα αποτελέσει σημαντικό πλεονέκτημα.

- Βιβλιογραφία: Είναι προτιμότερα τα ανοιχτά και προτυποποιημένα πρωτόκολλα με διαθέσιμη βιβλιογραφία καθώς τα κλειστά πρωτόκολλα απαιτούν επιπλέον χρόνο για την ανάλυση τους.

- Αδυναμίες: Για την απόδειξη της αποτελεσματικότητας του fuzzer θα χρειαστεί να βρεθούν κάποιες αδυναμίες που έχουν αποκαλυφθεί με άλλες μεθόδους. Όσο περισσότερες γνωστές αδυναμίες υπάρχουν στο πρωτόκολλο τόσο μεγαλύτερες είναι και οι πιθανότητες να επαληθευτούν με την μέθοδο του fuzzing.

2.1.2.Αξιολόγηση

Τα επικρατέστερα πρωτόκολλα που πληρούν τα παραπάνω κριτήρια είναι το SIP και το XMPP. Είναι και τα δύο ανοιχτού κώδικα και έχουν μεγάλη πολυπλοκότητα. Η διαθέσιμη βιβλιογραφία τους καλύπτεται από τα διαθέσιμα RFCs που υπάρχουν και για τα δύο πρωτόκολλα καθώς και από μεγάλο πλήθος βιβλίων. Η ύπαρξη αδυναμιών είναι σαφώς υπέρ του πρωτοκόλλου SIP καθώς η αυξημένη διάδοση του έχει συντελέσει στην εύρεση πολλών αδυναμιών. Αδυναμίες είναι γνωστές και για το πρωτόκολλο XMPP αλλά όχι σε τόσο μεγάλο βαθμό. Τέλος η διάδοση του SIP έχει σαν αποτέλεσμα και την ύπαρξη πολλών fuzzers για το συγκεκριμένο πρωτόκολλο (KiF, sip- fuzzer, voiper) κάτι το οποίο μας αποθαρρύνει για την επιλογή του. Αντιθέτως, για το XMPP βρέθηκε μόνο ένας fuzzer (xmpp-fuzzer, https://code.google.com/p/xmpp-fuzzer/) με τελευταία ενημέρωση τον Σεπτέμβριο του 2009.

Αυτό αποτελεί ένδειξη ότι το πρωτόκολλο XMPP δεν έχει εξεταστεί επαρκώς με τις μεθόδους fuzzing και σε συνδυασμό με την παραπάνω αξιολόγηση το καθιστά κατάλληλο για τον σκοπό της παρούσας εργασίας.

2.1.3.Συμπέρασμα

Σύμφωνα με την αξιολόγηση που προηγήθηκε το πρωτόκολλο XMPP είναι καταλληλότερο για την ανάπτυξη ενός fuzzer. Λόγω του μεγάλου αριθμού υλοποιήσεων ανοιχτού κώδικα θα είναι δυνατός ο έλεγχος σε ένα μεγάλο αριθμό εφαρμογών δίνοντας την δυνατότητα εύρεσης περισσότερων αδυναμιών. Επίσης η αυξημένη πολυπλοκότητά του λόγω των πολλών δυνατοτήτων που προσφέρει αυξάνει τις πιθανότητες για την εύρεση αδυναμιών. Σχετικά με την διαθέσιμη βιβλιογραφία το πρωτόκολλο XMPP διαθέτει μία ιστοσελίδα με όλα τα σχετικά RFCs και τις προεκτάσεις τους, όπως επίσης και άλλα συγγράμματα τα οποία αφορούν το πρωτόκολλο. Επίσης η εύρεση ενός μόνο fuzzer για το συγκεκριμένο πρωτόκολλο αποτελεί ένδειξη ότι δεν έχει εξεταστεί επαρκώς με την μέθοδο του fuzzing. Τέλος, η αυξανόμενη χρήση του σε εφαρμογές όπως το Google Talk και Facebook Chat το καθιστούν ελκυστικό ¨στόχο¨.

2.2. Fuzzer

Σύμφωνα με τα κριτήρια που τέθηκαν στην παράγραφο 1.7 έγιναν και οι επιλογές για την ανάπτυξη του fuzzer. Αρχικά έγινε η επιλογή της γλώσσας προγραμματισμού στην οποία θα αναπτυχθεί το εργαλείο. Ως κατάλληλη γλώσσα επιλέχθηκε η Python λόγω της ευκολίας και της ευελιξίας που προσφέρει αλλά και λόγω προηγούμενης εμπειρίας που υπήρχε στην γλώσσα αυτή.

Η δεύτερη επιλογή αφορά το είδος του fuzzer. Το είδος των mutation-based fuzzers θα μπορούσε να υλοποιηθεί σχετικά σύντομα και χωρίς ιδιαίτερες δυσκολίες. Παρόλα αυτά το είδος αυτό δεν θεωρείται ιδιαιτέρως αποτελεσματικό για αυτό και απορρίφθηκε. Ο fuzzer που θα αναπτυχθεί θα είναι generation-based.

Για να ανήκει ο fuzzer στο είδος generation-based θα πρέπει να χρησιμοποιεί ¨σωστά¨ το πρωτόκολλο για να επικοινωνήσει με τον εξυπηρετητή. Για τον λόγο αυτό έπρεπε να επιλεχθεί μια βιβλιοθήκη για το πρωτόκολλο XMPP. Στον ιστότοπο του XMPP αναφέρει τις διαθέσιμες βιβλιοθήκες (http://xmpp.org/xmpp-software/libraries/) για κάθε γλώσσα προγραμματισμού. Για την Python είναι διαθέσιμες οι εξής: headstock, jabber.py, pyxmpp, SleekXMPP, Twisted Words και xmppy. Για την επιλογή της κατάλληλης βιβλιοθήκης ακολούθησε μια αξιολόγηση των διαθέσιμων επιλογών. Οι βιβλιοθήκες headstock και xmppy δεν διαθέτουν επαρκής βιβλιογραφία για την κατανόηση της χρήσης τους με αποτέλεσμα να είναι δύσκολες στην χρήση. Οι βιβλιοθήκες jabber.py και pyxmpp δεν έχουν ενημερωθεί πρόσφατα οπότε δεν αποτελούν αξιόπιστη επιλογή. Τέλος η βιβλιοθήκη Twisted Words προσφέρει την δυνατότητα δημιουργίας εφαρμογών για διάφορα πρωτόκολλα πχ IRC, XMPP κ.α κάτι το οποίο προσθέτει πολυπλοκότητα στην υλοποίηση του fuzzer η οποία δεν είναι απαραίτητη. Η πιο ολοκληρωμένη βιβλιοθήκη είναι η SleekXMPP (https://github.com/fritzy/SleekXMPP, http://sleekxmpp.com/) η οποία αναβαθμίζεται συνεχώς και διαθέτει παραδείγματα χρήσης τα οποία διευκολύνουν την χρήση της.

Συνοψίζοντας, ο fuzzer θα διαθέτει τα εξής χαρακτηριστικά:

- Γλώσσα προγραμματισμού: Python
- Είδος: Generation-based
- Χρήση βιβλιοθήκης: SleekXMPP

2.3. Εξυπηρετητής XMPP Ανοιχτού κώδικα

Για την πραγματοποίηση δοκιμών και την εύρεση αδυναμιών επιλέχθηκαν εξυπηρετητές ανοιχτού κώδικα. Από την διαθέσιμη λίστα των εξυπηρετητών XMPP που υπάρχει στην ιστοσελίδα του πρωτοκόλλου (http://xmpp.org/xmpp-software/servers/) επιλέχθηκαν τρεις διαφορετικές υλοποιήσεις. Οι διαφορές μεταξύ των εξυπηρετητών εντοπίζονται στην πλατφόρμα για την οποία προορίζονται αλλά και στην γλώσσα προγραμματισμού που έχουν αναπτυχθεί. Για τις δοκιμές χρησιμοποιήθηκαν οι εξυπηρετητές jabberd, jabberd2 και Openfire σε διάφορες εκδόσεις.

- Jabberd 1.4. Είναι η πρώτη υλοποίηση του πρωτοκόλλου XMPP από όταν αυτό ήταν ακόμα γνωστό ως Jabber. Η ανάπτυξη του εξυπηρετητή έγινε με βάση την ασφάλεια και την υποστήριξη των πρωτοκόλλων κρυπτογράφησης, την αυστηρή τήρηση των προδιαγραφών του πρωτοκόλλου XMPP, την ευκολία στην εγκατάσταση και την διαχείριση και την εύκολη ενσωμάτωση του σε ιστότοπους. Η γλώσσα προγραμματισμού που έχει γραφτεί είναι η C++. Είναι διαθέσιμος για πλατφόρμες Unix.

- Jabberd2. Ο jabberd2 είναι η επόμενη γενιά του προγράμματος jabberd. Έχει αναπτυχθεί από την αρχή έτσι ώστε να είναι επεκτάσιμος και να υποστηρίζει τις νεότερες επεκτάσεις του πρωτοκόλλου. Έχει αναπτυχθεί με την γλώσσα προγραμματισμού C. Είναι διαθέσιμος για τις πλατφόρμες Window, Linux και Solaris.

- Openfire. Ο Openfire είναι ένας εξυπηρετητής για το πρωτόκολλο XMPP. Είναι πολύ απλός στην εγκατάσταση και η διαχείριση του γίνετε μέσω ενός πολύ εύχρηστου web UI. Είναι διαθέσιμος για τις πλατφόρμες Linux, Mac OS X, Windows και Solaris. Έχει αναπτυχθεί με την γλώσσα προγραμματισμού Java.

3. Το πρωτόκολλο XMPP

Το XMPP είναι το ακρωνύμιο για το Extensible Messaging and Presence Protocol. Είναι ένα πρωτόκολλο για την ανταλλαγή μηνυμάτων σε πραγματικό χρόνο το οποίο βασίζεται στην XML για την μορφοποίηση των δεδομένων που ανταλλάσσονται και ήταν αρχικά γνωστό ως Jabber.

Το XMPP χρησιμοποιείται σε μεγάλο αριθμό εφαρμογών, με γνωστότερες όλων τα Google Talk και Facebook chat. Το πρωτόκολλο χωρίζεται σε δύο τμήματα. Το τμήμα των υπηρεσιών και το τμήμα των εφαρμογών. Το τμήμα των υπηρεσιών ορίζεται στις δύο εκδόσεις των προδιαγραφών του IETF (www.ietf.org) με τα RFC's και στις προδιαγραφές των επεκτάσεων τα οποία εκδίδονται από τον XMPP Standards Foundation (www.xmpp.org) μέσω των XEP's. Οι εφαρμογές αποτελούνται από το λογισμικό και από σενάρια υλοποίησης του πρωτοκόλλου τα οποία είναι διαθέσιμα προς τα ενδιαφερόμενα πρόσωπα και οργανισμούς.

Μια υπηρεσία του XMPP αποτελεί μία δυνατότητα που μπορεί να χρησιμοποιηθεί από οποιαδήποτε εφαρμογή. Οι υλοποιήσεις του XMPP συνήθως παρέχουν τις παρακάτω δυνατότητες:

- Κρυπτογράφηση καναλιού. Παρέχει κρυπτογράφηση της σύνδεσης μεταξύ του χρήστη και του εξυπηρετητή ή μεταξύ εξυπηρετητών, δίνοντας την δυνατότητα ανάπτυξης ασφαλών εφαρμογών.

- Αυθεντικοιποίηση. Εξασφαλίζει ότι κάθε οντότητα που θέλει να επικοινωνήσει μέσω του δικτύου είναι αυθεντικοποιημένη από τον εξυπηρετητή ο οποίος ελέγχει την πρόσβαση στο δίκτυο.

- Παρουσία. Παρέχει την δυνατότητα σε εγκεκριμένους χρήστες να ελέγχουν την διαθεσιμότητα άλλων χρηστών για επικοινωνία.

- Λίστα επαφών. Η υπηρεσία αυτή επιτρέπει την διατήρηση μίας λίστας επαφών στον εξυπηρετητή. Η λίστα παρέχει έναν άμεσο τρόπο ανταλλαγής μηνυμάτων και αποτελεί μία λίστα έμπιστων οντοτήτων.

- Ανταλλαγή μηνυμάτων. Δίνει την δυνατότητα ανταλλαγής μηνυμάτων μεταξύ δύο οντοτήτων.

- Ανταλλαγή μηνυμάτων με περισσότερους παραλήπτες. Δίνει την δυνατότητα ανταλλαγής μηνυμάτων μεταξύ πολλών χρηστών οι οποίοι μετέχουν σε ένα εικονικό «δωμάτιο επικοινωνίας» (chat room).

- Ανταλλαγή πολυμέσων. Δημιουργία συνεδρίας μεταξύ οντοτήτων για την ανταλλαγή αρχείων, επικοινωνίας με ήχο ή και εικόνα και αλληλεπίδραση σε πραγματικό χρόνο.

Αυτές είναι οι σημαντικότερες και πιο διαδεδομένες υπηρεσίες που προσφέρει το πρωτόκολλο από την στιγμή που ένας χρήσης ή μια εφαρμογή συνδεθεί σε ένα δίκτυο XMPP. Η συνεχής διάδοση και χρήση του πρωτοκόλλου επιτρέπει στην κοινότητα των προγραμματιστών να τις εξελίσσει και να δημιουργεί νέες υπηρεσίες για τις ανάγκες των χρηστών.

Με τις υπηρεσίες που είναι διαθέσιμες μέσω του πρωτοκόλλου XMPP, δίνετε η δυνατότητα ανάπτυξης μιας μεγάλης ποικιλίας εφαρμογών που βασίζονται στο πρωτόκολλο. Ενδεικτικά αναφέρονται οι παρακάτω κατηγορίες:

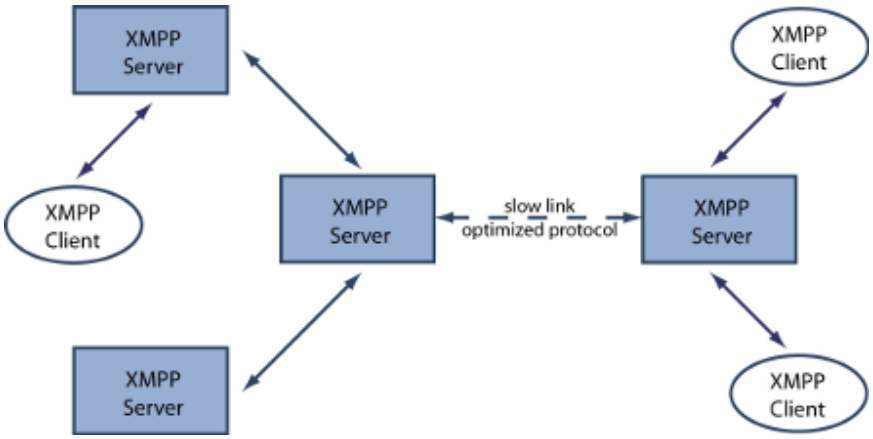
- Ανταλλαγή μηνυμάτων σε πραγματικό χρόνο. Με τον συνδυασμό των υπηρεσιών παρουσίας, λίστας επαφών και ανταλλαγής μηνυμάτων μπορεί να αναπτυχθεί μια εφαρμογή ανταλλαγής μηνυμάτων.

- Ανταλλαγή μηνυμάτων με πολλούς παραλήπτες. Με την χρήση της υπηρεσίας ανταλλαγής μηνυμάτων με περισσότερους παραλήπτες μπορούν να αναπτυχθούν εφαρμογές στα πρότυπα του IRC.

- Voice over IP (VoIP). Με την κυκλοφορία του Google Talk το 2005 έγινε ευρέως γνωστή η δυνατότητα του XMPP να προσφέρει υπηρεσίες φωνητικής συνομιλίας. Από την στιγμή εκείνη άρχισε η ανάπτυξη επεκτάσεων για την καλύτερη υποστήριξη της υπηρεσίας αυτής και συνεχίστηκε με την υιοθέτησή της από την Nokia και το πρόγραμμα One Laptop Per Child. Με την ίδια επέκτασή μπορεί να γίνει και ανταλλαγή αρχείων και video.

Αυτό είναι ένα μικρό δείγμα των δυνατοτήτων του πρωτοκόλλου XMPP. Μελλοντικά φαίνεται να ακολουθεί η υλοποίηση συστημάτων XMPP από μεγάλους οργανισμούς και παρόχους υπηρεσιών, αξιοποίηση των δυνατοτήτων του πρωτοκόλλου για την επίλυση επιχειρησιακών προβλημάτων και συνεχείς ανάπτυξη της κοινότητας των προγραμματιστών που ασχολούνται με την τεχνολογία του XMPP.

Η αρχιτεκτονική του XMPP είναι παρόμοια με αυτή του Παγκόσμιου Ιστού και του ηλεκτρονικού ταχυδρομείου (*Εικόνα 1*).



Εικόνα 1. Η Αρχιτεκτονική του XMPP

Χρησιμοποιεί την αποκεντρωμένη αρχιτεκτονική του πελάτη-εξυπηρετητή. Η αρχιτεκτονική αυτή έχει αρκετά πλεονεκτήματα, ανάμεσα στα οποία είναι και ο διαχωρισμός των εργασιών που πρέπει να γίνουν στον εξυπηρετητή και στον πελάτη, ευκολότερα στην διαχείριση συστήματα, δεν υπάρχουν μοναδικά σημεία αποτυχίας και τέλος δίνεται η δυνατότητα να εφαρμοστούν πολιτικές ασφάλειας στους εξυπηρετητές για την αυθεντικοποίηση των πελατών[2].

3.1. JID

Για την επικοινωνία μέσω του πρωτοκόλλου XMPP κάθε οντότητα που συμμετέχει πρέπει να διαθέτει μία διεύθυνση, όπως γίνεται σε όλα τα δίκτυα. Η διεύθυνση κάθε XMPP οντότητας καλείται Jabber Identification ή JID. Κάθε έγκυρο JID αποτελείται από τρία στοιχεία: το όνομα του κόμβου (node identifier), το όνομα του domain (domain identifier) και το όνομα της πηγής (resource identifier). Η μορφοποίηση του JID ορίζεται με βάση το Augmented Backus-Naur Form[3] και είναι η παρακάτω:

JID = [node"@"]domain["/"resource]
domain = FQDN / address-literal
FQDN = (sub-domain 1*("." sub-domain))
sub-domain = (internationalized domain label)
address-literal = IPv4 address / IPv6 address

Κατά την δημιουργία ενός λογαριασμού σε μία υπηρεσία XMPP ο χρήστης επιλέγει ένα «εικονικό» όνομα (node) που θα κατέχει και θα αναγνωρίζεται από το δίκτυο. Το όνομα δεν επηρεάζεται από πεζούς ή κεφαλαίους χαρακτήρες και υπάρχει η δυνατότητα το όνομα να παρέχεται από την υπηρεσία και να μην το επιλέγει ο χρήστης.

Το στοιχείο του domain στο JID οδηγεί σε ένα «έγκυρο» domain (fully qualified domain name, FQDN). Η επιλογή του ονόματος του domain γίνεται κατά την εγκατάσταση του εξυπηρετητή. Η σύνδεση με άλλους εξυπηρετητές που υπάρχουν στο δίκτυο γίνεται με την χρήση εγγραφών DNS. Και σε αυτή την περίπτωση το όνομα δεν επηρεάζεται από την επιλογή πεζών ή κεφαλαίων χαρακτήρων.

Όταν ο χρήστης συνδέεται σε ένα εξυπηρετητή XMPP είτε ερωτάται, είτε ανατίθεται από τον εξυπηρετητή ένα αναγνωριστικό στην πηγή από την οποία συνδέεται ο χρήστης. Το αναγνωριστικό αυτό χρησιμοποιείται για την προώθηση της κίνησης προς αυτή την πηγή σε περίπτωση που ο χρήστης έχει περισσότερες από μία ενεργές συνδέσεις με τον εξυπηρετητή. Η χρήση του αναγνωριστικού αυτού δίνει την δυνατότητα αλληλεπίδρασης με συγκεκριμένες συσκευές ή τοποθεσίες ανάλογα με το όνομα της πηγής. Κάθε πηγή είναι ανεξάρτητη για την υπηρεσία παρουσίας του πρωτοκόλλου και αντιμετωπίζεται σαν μοναδική. Το όνομα της πηγής δεν έχει περιορισμούς ως προς τους χαρακτήρες που θα χρησιμοποιηθούν, αντίθετα όμως με το όνομα και το domain, επηρεάζεται από πεζούς και κεφαλαίους χαρακτήρες.

Κάθε στοιχείο ενός έγκυρου JID δεν πρέπει να ξεπερνά τα 1023 bytes σε μήκος, όπου συνολικά μαζί με τα σύμβολα «@» και «/» το μήκος του JID δεν πρέπει να ξεπερνά τα 3071 bytes.

Οι «μηχανισμοί» του εξυπηρετητή που εξασφαλίζουν τους παραπάνω περιορισμούς πρέπει να αποτελέσουν σημείο ελέγχου για τον fuzzer καθώς η εύρεση αδυναμιών σε αυτά τα σημεία αποτελεί συχνό φαινόμενο. Οι εξυπηρετητές θα ελεγχθούν ως προς το μήκος κάθε στοιχείου αλλά και συνολικά ως προς το μήκος του JID, ως προς την χρήση μη-επιτρεπτών χαρακτήρων στα στοιχεία του JID και ως προς την μορφοποίησή του.

3.2. Ο πυρήνας του XMPP

Ο πυρήνας του XMPP αναλύεται στο Core RFC [4] όπου περιγράφονται όλα τα βασικά κομμάτια του πρωτοκόλλου, όπως: το XML stream, η ασφάλεια της συνεδρίας με την χρήση του SASL, η χρήση TLS και τα stanzas. Ορισμένοι εξυπηρετητές μπορεί να βασίζονται στο μη προτεινόμενο Core RFC [5].

3.3. XML Ροή (Stream)

Κάθε σύνδεση XMPP είναι μια σύνδεση TCP μακράς διάρκειας, η οποία ορίζεται ως ροή και εγκαθιδρύεται με την χρήση του στοιχείου <stream> της XML. Η ροή μπορεί να παρομοιαστεί με ένα πακέτο το οποίο περιέχει τα δεδομένα (στροφές/stanzas) που ανταλλάσσονται μεταξύ δύο κόμβων κατά την διάρκεια μιας συνεδρίας. Κατά την σύνδεση δημιουργούνται δύο ροές, μία με κατεύθυνση από τον χρήστη προς τον εξυπηρετητή και μία από τον εξυπηρετητή προς τον χρήστη. Η διακοπή μιας ροής γίνετε με την χρήση του στοιχείου </stream>, χωρίς απαραίτητα να διακόπτεται και η σύνδεση ανάμεσα στους κόμβους. Κάθε επικοινωνία του πρωτοκόλλου XMPP γίνεται με την χρήση ροών.

3.4. Στροφή (Stanza)

Μέσα στις ροές χρησιμοποιούνται οι στροφές οι οποίες περιέχουν τα δεδομένα για τις δυνατότητες που προσφέρει το πρωτόκολλο. Οι στροφές που ορίζονται για τους χρήστες και τους εξυπηρετητές είναι οι εξής:

- <message>: Ανταλλαγή μηνυμάτων. Προώθηση μηνυμάτων από ένα XMPP κόμβο σε ένα άλλο.

- <presence>: Ανανέωση κατάστασης παρουσίας. Ανακοίνωση της διαθεσιμότητας των χρηστών.

- <iq>: Ανταλλαγή info/query. Μέσω των στροφών αυτών είναι δυνατή η ανταλλαγή πληροφοριών, με την μορφή αίτηση/απάντηση, μεταξύ του χρήστη και του εξυπηρετητή. Η αποστολή απάντησης σε μία αίτηση iq είναι υποχρεωτική.

```
C: <stream:stream>

C:   <presence/>

C:   <iq type="get">
       <query xmlns="jabber:iq:roster"/>
     </iq>

S:   <iq type="result">
       <query xmlns="jabber:iq:roster">
         <item jid="alice@wonderland.lit"/>
         <item jid="madhatter@wonderland.lit"/>
         <item jid="whiterabbit@wonderland.lit"/>
       </query>
     </iq>

C:   <message from="queen@wonderland.lit"
              to="madhatter@wonderland.lit">
       <body>Off with his head!</body>
     </message>

S:   <message from="king@wonderland.lit"
              to="party@conference.wonderland.lit">
       <body>You are all pardoned.</body>
     </message>

C:   <presence type="unavailable"/>

C: </stream:stream>
```

Εικόνα 2. Παράδειγμα μίας ροής XML[2]

Η στροφή αποτελεί την ελάχιστη μονάδα πληροφορίας που μπορεί να διαδοθεί μέσω ενός δικτύου XMPP. Κατά την διάρκεια μιας σύνδεσης TCP, μέσω μιας ροής ανταλλάσσεται απεριόριστος αριθμός στροφών.

3.5. Απλή Αυθεντικοποίηση και Επίπεδο Ασφάλειας (SASL)

Για την έναρξη της επικοινωνίας με τον εξυπηρετητή ο κάθε χρήστης πρέπει να αυθεντικοποιήσει την ροή του μέσω παραδοσιακών μεθόδων αυθεντικοποίησης ή μέσω του SASL [6]. Το SASL αποτελεί ένα πλαίσιο (framework) το οποίο δίνει την δυνατότητα στην υλοποίηση (εξυπηρετητή ή εφαρμογή) να υποστηρίζει διαφορετικές μεθόδους αυθεντικοποίησης ανάλογα με τις απαιτήσεις της. Σύμφωνα με το πρωτόκολλο η προτεινόμενη διαδικασία είναι το SASL. Οι μηχανισμοί αυθεντικοποίησης που προσφέρει το πλαίσιο είναι οι παρακάτω:

- PLAIN. Η πιο απλή μέθοδος αυθεντικοποίησης η οποία βασίζεται σε συνθηματικό. Το συνθηματικό δεν στέλνεται κρυπτογραφημένο οπότε η χρήση του απαιτεί την ύπαρξη κρυπτογράφησης του καναλιού επικοινωνίας.

- DIGEST-MD5. Η μέθοδος αυτή είναι πιο ασφαλής από την plain καθώς το συνθηματικό κρυπτογραφείται πριν μεταδοθεί στο κανάλι. Παρόλα αυτά η μέθοδος αυτή παρουσιάζει πολλά προβλήματα συμβατότητας.

- SCRAM. Η πρόταση για να αντικαταστήσει την DIGEST-MD5 είναι η SCRAM (Salted Challenge Response Authentication Mechanism). Η μέθοδος προσφέρει υψηλό επίπεδο ασφάλειας ειδικά αν συνδυαστεί με την κρυπτογράφηση καναλιού TLS.

- EXTERNAL. Η μέθοδος αυτή δίνει την δυνατότητα στην οντότητα να χρησιμοποιήσει το ψηφιακό της πιστοποιητικό για την εγκαθίδρυση του TLS και με την χρήση του ίδιου πιστοποιητικού να αυθεντικοποιηθεί στην υπηρεσία. Η μέθοδος αυτή χρησιμοποιείται κυρίως για επικοινωνίες μεταξύ εξυπηρετητών καθώς μικρός αριθμός χρηστών διαθέτουν ψηφιακό πιστοποιητικό.

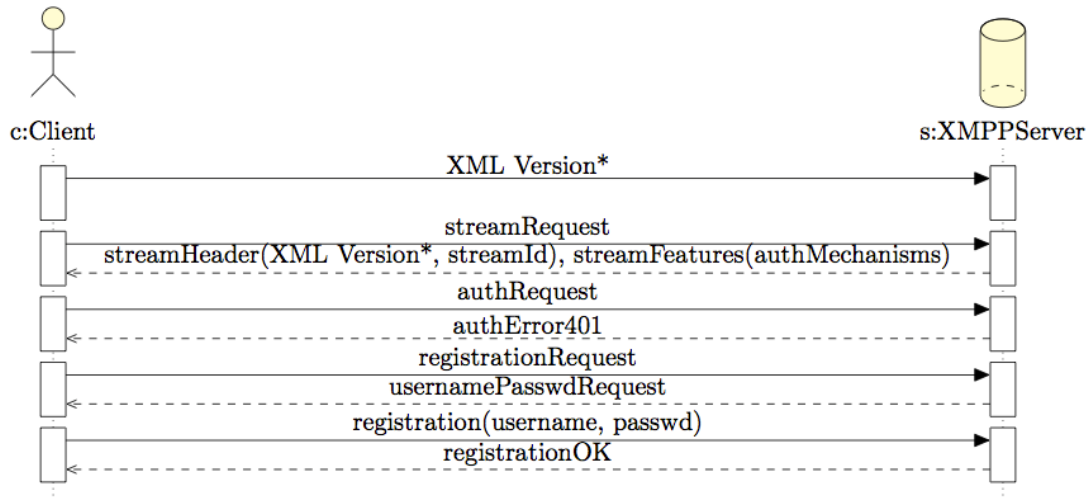
- ANONYMOUS. Η μέθοδος αυτή επιτρέπει στους χρήστες να αυθεντικοποιηθούν χωρίς να είναι απαραίτητη η ύπαρξη προηγούμενης εγγραφής στην υπηρεσία. Η χρήση της γίνεται συνήθως σε online εφαρμογές που η διαδικασία εγγραφής των χρηστών δεν χρειάζεται (πχ online βοήθεια).

Κατά την αυθεντικοποίηση του χρήστη μέσω SASL ο εξυπηρετητής στέλνει στο χρήστη μία λίστα με τους διαθέσιμους μηχανισμούς αυθεντικοποίησης κατά σειρά προτίμησης. Ο χρήστης επιλέγει τον μηχανισμό αυθεντικοποίησης που επιθυμεί και στέλνει τα διαπιστευτήρια του. Ανάλογα με τον μηχανισμό αυθεντικοποίησης που επιλέχθηκε ακολουθεί μία σειρά αιτήσεων/απαντήσεων μέχρι να αυθεντικοποιηθεί ο χρήστης. Οι λεπτομέρειες σχετικά με την διαδικασία αυθεντικοποίησης περιγράφονται στο Core RFC στην παράγραφο 6. Ύστερα από την αυθεντικοποίηση του χρήστη γίνετε επανεκκίνηση της ροής.

3.6. Εγγραφή χρηστών

Για την αυθεντικοποίηση ενός χρήστη μέσω του SASL, αυτός αρχικά θα πρέπει να είναι εγγεγραμμένος στον εξυπηρετητή. Η εγγραφή ενός χρήστη μπορεί να γίνει από τον διαχειριστή του εξυπηρετητή ή μέσω της δυνατότητας εγγραφής νέων χρηστών κατ ευθείαν στον εξυπηρετητή. Για την απ ευθείας εγγραφή του χρήστη η διαδικασία που ακολουθείται είναι η εξής (*Εικόνα 3*): Αρχικά γίνεται η διαδικασία αυθεντικοποίησης που περιγράφηκε προηγουμένως. Όταν η αυθεντικοποίηση αποτύχει ο χρήστης στέλνει μία αίτηση εγγραφής.

Ο εξυπηρετητής στέλνει στον χρήστη μία αίτηση για όνομα χρήστη και κωδικό πρόσβασης. Ο χρήστης στέλνει τα επιθυμητά διαπιστευτήρια και ο εξυπηρετητής απαντάει με την επιβεβαίωση της εγγραφής.



Εικόνα 3. Εγγραφή χρήστη στον εξυπηρετητή

Τα διαπιστευτήρια του χρήστη αποθηκεύονται στον εξυπηρετητή σε μία βάση δεδομένων. Αφού γίνει η εγγραφή, ο χρήστης αποκτά πρόσβαση στις υπηρεσίες XMPP και έχει την δυνατότητα να αλληλεπιδράσει με άλλους εγγεγραμμένους χρήστες.

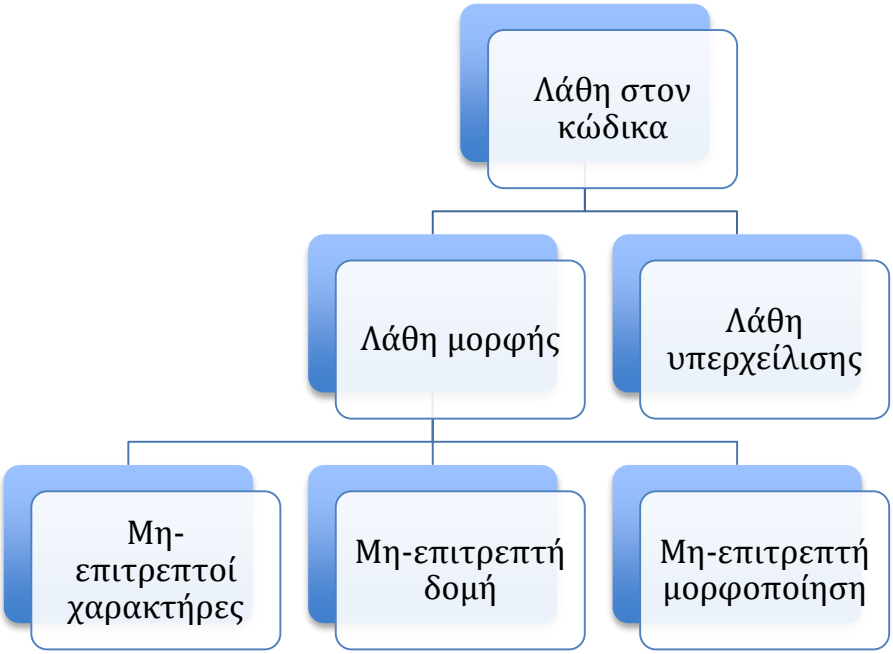
4. XMPP Fuzzer

Ένα από τα σημαντικότερα πράγματα στο fuzzing είναι η δυνατότητα αναπαραγωγής των ίδιων αποτελεσμάτων με τις ίδιες προϋποθέσεις. Κάθε έλεγχος που διεξάγεται με τις ίδιες αρχικές συνθήκες πρέπει να έχει τα ίδια αποτελέσματα. Αν δεν συμβαίνει αυτό ο fuzzer χάνει την αξιοπιστία του και την αποτελεσματικότητά του. Για την εξασφάλιση της αξιοπιστίας των αποτελεσμάτων οι βάσεις δεδομένων των εξυπηρετητών που ελέγχονται πρέπει να επανέρχονται στην αρχική τους κατάσταση, χωρίς εγγραφές από προηγούμενους ελέγχους.

Σε παλιότερες εκδόσεις εξυπηρετητών είχαν βρεθεί αδυναμίες στον τρόπο με τον οποίο διαχειριζόταν η XML παραμορφωμένα δεδομένα, επομένως ο έλεγχος της διαδικασίας αυτής μπορεί να αποκαλύψει αδυναμίες. Ωστόσο, το πρωτόκολλο XMPP ορίζει ότι οι στροφές πρέπει να ακολουθούν συγκεκριμένη σύνταξη, και όσες δεν τηρούν τον κανόνα αυτό πρέπει να απορρίπτονται από τον εξυπηρετητή. Για την αποφυγή της απόρριψης των στροφών ο fuzzer στέλνει στον εξυπηρετητή συντακτικά «ορθές» στροφές και τα παραμορφωμένα δεδομένα θα περιέχονται στα πεδία ονομάτων.

Ο XMPP fuzzer πραγματοποιεί ελέγχους σε δύο διαφορετικές περιπτώσεις, πριν και μετά την αυθεντικοποίηση του χρήστη. Ο έλεγχος πραγματοποιείται με την χρήση παραμορφωμένων *ακολουθιών χαρακτήρων*(strings) σε πεδία ονομάτων. Οι παραμορφωμένες ακολουθίες χαρακτήρων είναι συγκεντρωμένες σε ένα αρχείο. Ο fuzzer διαβάζει κάθε φορά μία ακολουθία και την χρησιμοποιεί σε ένα πεδίο εισαγωγής. Η διαδικασία αυτή επαναλαμβάνεται μέχρι να εξαντληθούν οι ακολουθίες.

Η επιλογή των ακολουθιών έγινε με βάση τα είδη των λαθών(bugs) που μπορεί να υπάρχουν στον κώδικα του εξυπηρετητή (*Εικόνα 4*). Για την κάθε περίπτωση έχουν συμπεριληφθεί ακολουθίες χαρακτήρων που μπορεί να προκαλέσουν σφάλματα. Για την περίπτωση των λαθών υπερχείλισης υπάρχουν ακολουθίες χαρακτήρων που ξεπερνούν τα όρια που ορίζονται από το πρωτόκολλο. Για τα λάθη μορφής αντίστοιχα, υπάρχουν ακολουθίες χαρακτήρων που περιέχουν μη-επιτρεπτούς χαρακτήρες. Για παράδειγμα: εναλλαγή πεζών-κεφαλαίων (‘asdncsjv’, ‘DJVNJE’, ‘achjMSDI’), ακολουθίες που χρησιμοποιούνται για να δηλώσουν λογικές τιμές(’1’, ’0’, ’true’, ’false’, ’True’, ’False’, ’TRUE’, ’FALSE’), αριθμητικές ακολουθίες με θετικούς, αρνητικούς, δεκαδικούς, ακεραίους(‘3456’, ‘-3456’, ‘3.456’, ‘-3.456’, ‘1E-345’, ‘-1e345’) και ειδικούς χαρακτήρες και σύμβολα(’m\nm’, ’m\rm’, u’bl\x00ub’, u’bl\x0Eub’, u’m\U0001F4A9’, u’\u1337’). Στόχος του fuzzer είναι να οδηγήσει τον εξυπηρετητή σε σφάλματα κατά την επεξεργασία των παραμορφωμένων ακολουθιών χαρακτήρων που του στέλνει. Για παράδειγμα, η εμφάνιση μη-επιτρεπτών χαρακτήρων στο πεδίο με το όνομα του εξυπηρετητή ή η αποστολή μεγαλύτερου μήκους ακολουθίας από αυτή που αναμένει ο εξυπηρετητής είναι πιθανό να οδηγήσουν σε σφάλματα.



Εικόνα 4 .Είδη λαθών στον κώδικα

Η επικοινωνία μεταξύ του fuzzer και του εξυπηρετητή περιορίζεται στην χρήση της SASL και των μεθόδων PLAIN και DIGEST-MD5. Η επιλογή αυτή έγινε σκόπιμα για να είναι δυνατή η παρακολούθηση των μηνυμάτων που ανταλλάσσονται με την χρήση του Wireshark. Αυτό μας δίνει την δυνατότητα να ελέγχουμε την εγκυρότητα των μηνυμάτων από και προς τον εξυπηρετητή και την διόρθωση τυχόν προβλημάτων επικοινωνίας μεταξύ του fuzzer και του εξυπηρετητή(debugging).

4.1. Βιβλιοθήκη SleekXMPP

Ο xmpp fuzzer βασίζεται στην βιβλιοθήκη SleekXMPP του Nathan Fritz. Η επιλογή της βιβλιοθήκης έγινε ύστερα από αξιολόγηση των διαθέσιμων βιβλιοθηκών για το πρωτόκολλο XMPP και κρίθηκε ως η καταλληλότερη. Η χρήση της βιβλιοθήκης διευκολύνει την υλοποίηση του fuzzer διότι αναλαμβάνει την ορθή χρήση του πρωτοκόλλου με την σωστή σύνταξη της XML, την τήρηση χρονικής σήμανσης των μηνυμάτων και των ροών. Χωρίς την χρήση της βιβλιοθήκης όλα τα παραπάνω θα έπρεπε να γίνονται από τον fuzzer. Αυτό θα είχε σαν αποτέλεσμα την αύξηση της πολυπλοκότητας του κώδικα η οποία πιθανόν να δημιουργούσε προβλήματα στην λειτουργία του ή κενά ασφάλειας. Ο fuzzer σαν λογισμικό πρέπει να είναι όσο το δυνατόν πιο απλός, εύχρηστος και επεκτάσιμος έτσι ώστε να είναι αποτελεσματικός έναντι εναλλακτικών μεθόδων εύρεσης σφαλμάτων.

Επίσης η βιβλιοθήκη ελέγχει τις ακολουθίες χαρακτήρων που εισάγονται από τον χρήστη για τυχών μη επιτρεπτούς χαρακτήρες, καθώς και το μήκος των ακολουθιών να μην ξεπερνάει τα όρια που ορίζονται από το πρωτόκολλο. Στην περίπτωση του fuzzer αυτοί οι έλεγχοι πρέπει να παρακαμφθούν για να είναι δυνατές οι δοκιμές για σφάλματα στον εξυπηρετητή.

Ύστερα από την μελέτη της βιβλιοθήκης βρέθηκαν τα κομμάτια κώδικα τα οποία θα πρέπει να τροποποιηθούν κατάλληλα έτσι ώστε να είναι δυνατή η αποστολή παραμορφωμένων ακολουθιών χαρακτήρων προς τον εξυπηρετητή, χωρίς αυτά να απορρίπτονται από την βιβλιοθήκη. Τα κομμάτια κώδικα που τροποποιήθηκαν είναι τα εξής:

- jid.py Το module jid.py είναι υπεύθυνο για την διαχείριση των JID. Αρχικά υπάρχει μια λίστα με τους μη επιτρεπτούς χαρακτήρες σε ένα JID και μία regular expression

που ορίζει την μορφή που πρέπει να έχει το JID. Οι γραμμές αυτές αφαιρέθηκαν και η regular expression αντικαταστάθηκε με μία άλλη που επιτρέπει το μεγαλύτερο μήκος ακολουθιών. Επίσης έχουν αντικατασταθεί όλα τα σημεία όπου το όριο ορίζεται στα 1023 bytes με νέο όριο τα 10250 bytes (*Εικόνα 3*).

```
#: These characters are not allowed to appear in a JID.
ILLEGAL_CHARS = '\x00\x01\x02\x03\x04\x05\x06\x07\x08\t\n\x0b\x0c\r' + \
                '\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19' + \
                '\x1a\x1b\x1c\x1d\x1e\x1f' + \
                ' !"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~\x7f'

#: The basic regex pattern that a JID must match in order to determine
#: the local, domain, and resource parts. This regex does NOT do any
#: validation, which requires application of nodeprep, resourceprep, etc.
JID_PATTERN = re.compile(
    "^(?:([^\"&'/:<>@]{1,1023})@)?([^/@]{1,1023})(?:/(.{1,1023}))?$"
```

```
#: These characters are not allowed to appear in a JID.
## ILLEGAL_CHARS = '\x00\x01\x02\x03\x04\x05\x06\x07\x08\t\n\x0b\x0c\r' + \
##                 '\x0e\x0f\x10\x11\x12\x13\x14\x15\x16\x17\x18\x19' + \
##                 '\x1a\x1b\x1c\x1d\x1e\x1f' + \
##                 ' !"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~\x7f'
ILLEGAL_CHARS = ''
#: The basic regex pattern that a JID must match in order to determine
#: the local, domain, and resource parts. This regex does NOT do any
#: validation, which requires application of nodeprep, resourceprep, etc.
# #JID_PATTERN = re.compile(
# #    "^(?:([^\"&'/:<>@]{1,10250})@)?([^/@]{1,10250})(?:/(.{1,10250}))?$"
# #)##default = 1023
JID_PATTERN = re.compile("^(?:(.{1,10250})@)?([^/@]{1,10250})(?:/(.{1,10250}))?$")
```

Εικόνα 5. Αφαίρεση μη-επιτρεπτών χαρακτήρων και έλεγχος μορφής JID

Τέλος έγιναν αλλαγές στις συναρτήσεις validate_node και validate_domain οι οποίες ελέγχουν το μήκος του node και του domain ώστε να μην είναι κενά ή να μην ξεπερνούν το όριο των 1023 bytes (*Εικόνα 4*).

```
def _validate_node(node):
    """Validate the local, or username, portion of a JID.

    :raises InvalidJID:

    :returns: The local portion of a JID, as validated by nodeprep.
    """
    try:
        if node is not None:
            node = nodeprep(node)

            if not node:
                raise InvalidJID('Localpart must not be 0 bytes')
            if len(node) > 1023:
                raise InvalidJID('Localpart must be less than 1024 bytes')
            return node
    except stringprep_profiles.StringPrepError:
        raise InvalidJID('Invalid local part')
```

```
def _validate_node(node):  #### !!!!!!!! UNCOMMENT try:!!!!
    """Validate the local, or username, portion of a JID.

    :raises InvalidJID:

    :returns: The local portion of a JID, as validated by nodeprep.
    """
    # try:
    if node is not None:
        node = nodeprep(node)

        if not node:
            raise InvalidJID('Localpart must not be 0 bytes')
        if len(node) > 10250:#default = 1023
            raise InvalidJID('Localpart must be less than 1024 bytes')
        return node
    # except stringprep_profiles.StringPrepError:
    #     raise InvalidJID('Invalid local part')
```

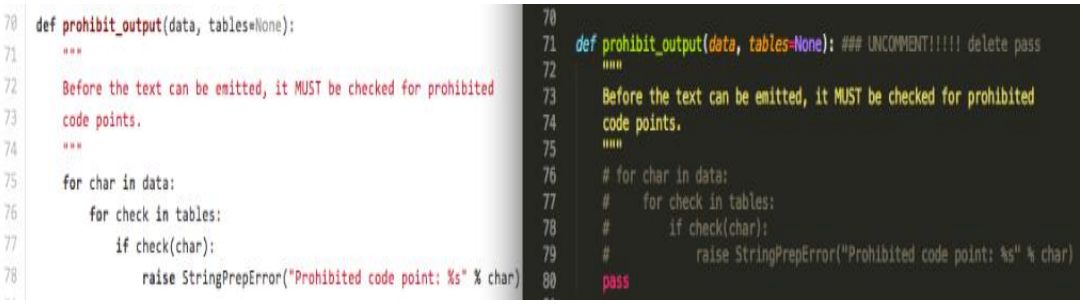
Εικόνα 6. Η συνάρτηση validate_node

- /xmlstream/tostring.py Το module tostring.py μετατρέπει τα αντικείμενα XML σε ακολουθίες χαρακτήρων Unicode. Η συνάρτηση escape μετατρέπει τους ειδικούς χαρακτήρες της XML έτσι ώστε να μην δημιουργούν σφάλματα (escape sequences). Η συνάρτηση αυτή καταργήθηκε έτσι ώστε να μην επηρεάζονται οι παραμορφωμένες ακολουθίες χαρακτήρων που στέλνει ο fuzzer (*Εικόνα 5*).

```
def escape(text, use_cdata=False):
    """Convert special characters in XML to escape sequences.

    :param string text: The XML text to convert.
    :rtype: Unicode string
    """
    # if sys.version_info < (3, 0):
    #     if type(text) != types.UnicodeType:
    #         text = unicode(text, 'utf-8', 'ignore')

    # # escapes = {'&': '&amp;',  UNCOMMENT HERE!!!
    # #            '<': '&lt;',
    # #            '>': '&gt;',
    # #            "'": '&apos;',
    # #            '"': '&quot;'}
    # escapes = {'&': '&',
    #            '<': '<',
    #            '>': '>',
    #            "'": "'",
    #            '"': '"'}

    # if not use_cdata:
    #     text = list(text)
    #     for i, c in enumerate(text):
    #         text[i] = escapes.get(c, c)
    #     return ''.join(text)
    # else:
    #     escape_needed = False
    #     for c in text:
    #         if c in escapes:
    #             escape_needed = True
    #             break
    #     if escape_needed:
    #         escaped = map(lambda x : "<![CDATA[%s]]>" % x, text.split("]]>"))
    #         return "<![CDATA[]]><![CDATA[]>]]>".join(escaped)
    #     return text
    return text
```

Εικόνα 7. Η συνάρτηση escape

- /util/stringprep_profiles.py Το module stringprep_profiles διευκολύνει την δημιουργία προφίλ για την stringprep (http://docs.python.org/2/library/stringprep.html) , όπως το nodeprep για το όνομα του κόμβου και το resourceprep για το όνομα της πηγής, έτσι ώστε να είναι έγκυρο το JID. Για την σωστή λειτουργία του fuzzer πρέπει να παρακαμφθεί η συνάρτηση prohibit_output η οποία ελέγχει τον κάθε χαρακτήρα του ονόματος εάν ανήκει στους απαγορευμένους χαρακτήρες και επιστρέφει σφάλμα εάν αυτό ισχύει (*Εικόνα 8*).



Εικόνα 8. Η συνάρτηση prohibit_output

Με τις παραπάνω αλλαγές η βιβλιοθήκη είναι έτοιμη να χρησιμοποιηθεί από τον fuzzer. Χωρίς τις παραπάνω αλλαγές δεν είναι δυνατή η λειτουργία του fuzzer διότι η βιβλιοθήκη επιστρέφει σφάλματα κατά τον έλεγχο των ακολουθιών και διακόπτει την λειτουργία της. Επίσης πολλές από τις παραμορφωμένες ακολουθίες χαρακτήρων δημιουργούν σφάλματα στην ίδια την βιβλιοθήκη αφού δεν μπορεί να τα διαχειριστεί.

4.2. Αυθεντικοποίηση

Ο XMPP Fuzzer έχει την δυνατότητα να πραγματοποιήσει ελέγχους σε δύο διαφορετικά σημεία που γίνεται η εισαγωγή δεδομένων. Το πρώτο σημείο είναι στην διαδικασία της αυθεντικοποίησης του χρήστη όπου εισάγονται το JID και το συνθηματικό του. Το δεύτερο σημείο είναι μετά την επιτυχημένη αυθεντικοποίηση του χρήστη κατά την ανταλλαγή μηνυμάτων.

Για την αυθεντικοποίηση του χρήστη από τον εξυπηρετητή γίνεται μία ανταλλαγή μηνυμάτων μεταξύ τους[2]. Αρχικά ο εξυπηρετητής ενημερώνει τον χρήστη για τις διαθέσιμες υπηρεσίες:

```
<stream:features>
  <starttls xmlns="urn:ietf:params:xml:ns:xmpp-tls">
    <optional/>
  </starttls>
  <mechanisms xmlns="urn:ietf:params:xml:ns:xmpp-sasl">
    <mechanism>PLAIN</mechanism>
    <mechanism>DIGEST-MD5</mechanism>
    <required/>
  </mechanisms>
  <compression xmlns="http://jabber.org/features/compress">
     <method>zlib</method>
  </compression>
</stream:features>
```

Για τις δοκιμές που πραγματοποιήθηκαν έγινε χρήση της μεθόδου PLAIN για να είναι δυνατή η παρακολούθηση των μηνυμάτων μέσω του Wireshark. Για τον λόγο αυτό και ο χρήστης θα επιλέξει την μέθοδο αυτή για να αυθεντικοποιηθεί. Επίσης αφού επιλέξει τον μηχανισμό

αυθεντικοποίησης στέλνει το όνομα χρήστη και το συνθηματικό κωδικοποιημένα σε Base64. Η απάντηση του χρήστη στον εξυπηρετητή θα είναι:

```
<auth xmlns="urn:ietf:params:xml:ns:xmpp-sasl" mechanism="PLAIN">
AGFsaWNlAHBhc3N3b3JkCg==
</auth>
```

Ο εξυπηρετητής επιβεβαιώνει την επιτυχημένη αυθεντικοποίηση του χρήστη:

```
<success xmlns="urn:ietf:params:xml:ns:xmpp-sasl"/>
```

Κατά τον έλεγχο της διαδικασίας αυθεντικοποίησης ο fuzzer πραγματοποιεί τέσσερις διαφορετικούς συνδυασμούς για κάθε παραμορφωμένη ακολουθία χαρακτήρων στο JID. Δεδομένης της μορφοποίησης του JID (node@domain/resource), κάθε παραμορφωμένη ακολουθία χαρακτήρων αντικαθιστά τα στοιχεία του JID. Αρχικά το node (*fuzz*@domain/resource), στην συνέχεια το domain (node@*fuzz*/resource) και τέλος το resource (node@domain/*fuzz*). Σε μία τελευταία δοκιμή γίνεται η ταυτόχρονη αντικατάσταση όλων των στοιχείων (*fuzz@fuzz/fuzz*), για τον έλεγχο διαχείρισης μεγάλου μήκους ακολουθιών από τον εξυπηρετητή αλλά και για λόγους πληρότητας.

Ο έλεγχος της μεθόδου αυθεντικοποίησης του χρήστη μπορεί να πραγματοποιηθεί σε κάθε εξυπηρετητή XMPP. Ο χρήστης αρκεί να εισάγει την IP του εξυπηρετητή και το domain και ο fuzzer θα αρχίσει να πραγματοποιεί δοκιμές. Κάθε ακολουθία χαρακτήρων που δοκιμάζεται εκτυπώνεται στην οθόνη έτσι ώστε να είναι δυνατός ο εντοπισμός της ακολουθίας που προκάλεσε το σφάλμα.

4.3. Ανταλλαγή μηνυμάτων

Το δεύτερο σημείο ελέγχου είναι κατά την ανταλλαγή μηνυμάτων. Ο XMPP fuzzer αυθεντικοποιείται στον εξυπηρετητή με τα διαπιστευτήρια που του έχει δώσει ο χρήστης και ξεκινάει να στέλνει μηνύματα προς κάποιον παραλήπτη. Για την ανταλλαγή μηνυμάτων το πρωτόκολλο χρησιμοποιεί τη στροφή <message/> η οποία προωθεί τα μηνύματα ανάμεσα στους κόμβους. Κάθε μήνυμα περιέχει τα στοιχεία του αποστολέα, του παραλήπτη, τον τύπο του μηνύματος και το id. Τα στοιχεία του αποστολέα συμπληρώνονται συνήθως από τον εξυπηρετητή για την αποφυγή επιθέσεων μεταμφίεσης (masquerade). Το πεδίο id εξυπηρετεί στον εντοπισμό των μηνυμάτων που έχουν ανταλλαγεί.

Τα μηνύματα, πέρα από τα πεδία που αναφέρθηκαν, περιέχουν κάποια επιπλέον στοιχεία τα οποία βρίσκονται εντός της στροφής <message/>. Το πρωτόκολλο XMPP ορίζει κάποια βασικά στοιχεία του μηνύματος όπως το <body/> και το <subject/>, τα οποία χρησιμοποιούνται κυρίως στην συνομιλία μεταξύ των κόμβων-χρηστών[2]. Η στροφή ενός απλού μηνύματος μεταξύ δύο χρηστών έχει την παρακάτω μορφή:

```
<message from="sender@domain/resource"
         to="receiver@domain/resource"
         type="chat">
    <body>Hi there!</body>
    <subject>Hello</subject>
</message>
```

Το πεδίο ονόματος του παραλήπτη είναι το “message to”. Ο fuzzer δημιουργεί το JID του παραλήπτη από τις παραμορφωμένες ακολουθίες χαρακτήρων και είναι της μορφής *fuzz@domain*. Με τον τρόπο αυτό ελέγχεται η ικανότητα του εξυπηρετητή να επεξεργαστεί

παραμορφωμένα δεδομένα στο JID σε διαφορετική περίπτωση από αυτή της αυθεντικοποίησης.

Εκτός από το πεδίο ονόματος του παραλήπτη ελέγχεται και το πεδίο «type». Το πεδίο αυτό μπορεί να έχει πέντε διαφορετικές τιμές: normal, chat, groupchat, headline και error. Κάθε τιμή καθορίζει τον τύπο του μηνύματος, δηλαδή αν είναι απλό μήνυμα (normal) , ανταλλαγή μηνυμάτων σε πραγματικό χρόνο (chat), ομαδική συζήτηση (groupchat), μηνύματα ειδοποιήσεων (headline) και τέλος μηνύματα εντοπισμού σφάλματος (error). Ο fuzzer θέτει κάθε φορά ως type του μηνύματος μια παραμορφωμένη ακολουθία χαρακτήρων.

Τέλος, ο fuzzer ελέγχει το στοιχείο subject. Ο έλεγχος γίνετε με την εισαγωγή παραμορφωμένων ακολουθιών χαρακτήρων στο αντίστοιχο πεδίο <subject> μέσα στην στροφή του μηνύματος(<message/>).

Για κάθε παραμορφωμένη ακολουθία χαρακτήρων ο fuzzer στέλνει την παρακάτω στροφή στον εξυπηρετητή:

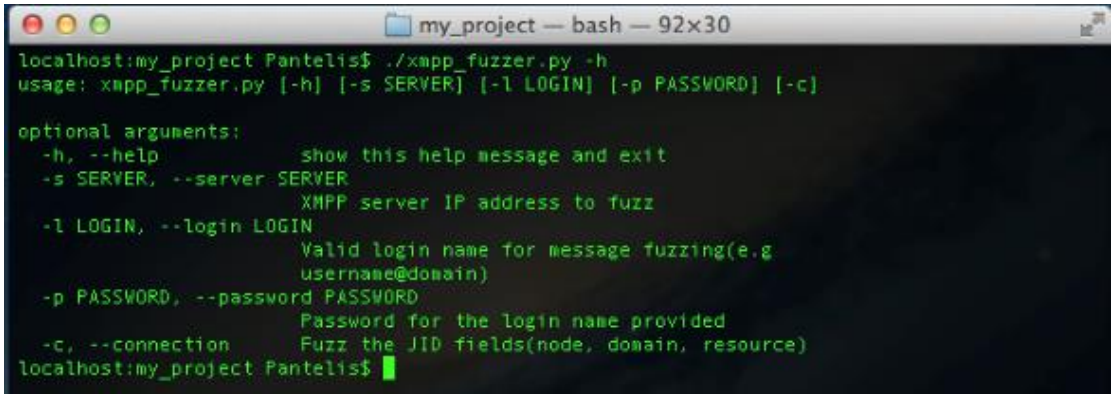
```
<message from="sender@domain"
         to="fuzz@domain/resource"
         type="fuzz">
    <body>Hi there!</body>
    <subject>fuzz</subject>
</message>
```

4.4. Χρήση του XMPP Fuzzer

Για την εκτέλεση του XMPP fuzzer απαιτείται η ύπαρξη της Python 2.7.5 στο σύστημα. Σε συστήματα με λειτουργικό Linux/Unix η Python είναι εγκατεστημένη εξ αρχής. Στην περίπτωση των Windows είναι απαραίτητη η εγκατάσταση της έκδοσης 2.7.5 (http://python.org/download/). Για την λειτουργία του προγράμματος απαιτείται και η τροποποιημένη έκδοση της βιβλιοθήκης sleekXMPP η οποία παρέχεται μαζί με το εργαλείο. Για την εγκατάσταση της βιβλιοθήκης πρέπει να γίνει αντιγραφή του φακέλου *sleekxmpp* στον φάκελο *C:\Python27\Lib* σε περιβάλλον Windows. Αντίστοιχα, στην περίπτωση του λειτουργικού Linux ο φάκελος sleekxmpp πρέπει να αντιγραφεί στην τοποθεσία */usr/local/lib/python2.7/dist-packages*.

Μετά την εγκατάσταση των προαπαιτούμενων το εργαλείο μπορεί να εκτελεστεί από την γραμμή εντολών. Κατά την εκτέλεση του εργαλείου δεν υπάρχουν διαφορές ανάμεσα στις πλατφόρμες Linux και Windows.

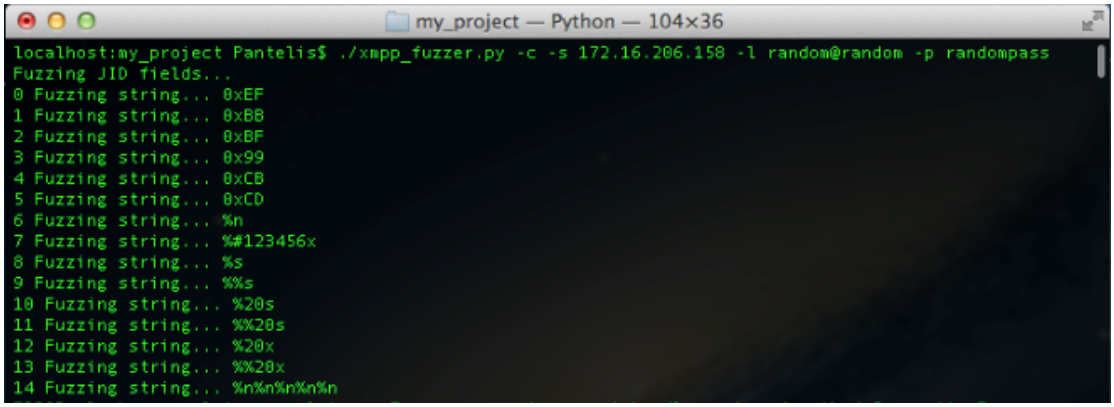
Σε ένα νέο παράθυρο κονσόλας (Linux) ή γραμμής εντολών (Windows) μεταβαίνουμε στο φάκελο που βρίσκονται τα αρχεία *xmpp_fuzzer.py* και *fuzz_strings.txt*. Στην συνέχεια εκτελούμε την εντολή *./xmpp_fuzzer.py -h* (σε περιβάλλον Windows χωρίς ./) και εμφανίζεται η βοήθεια του εργαλείου που εξηγεί τις διαθέσιμες επιλογές κατά την εκτέλεση (*Εικόνα 9*).



Εικόνα 9. Οι επιλογές εκτέλεσης του εργαλείου

Για την εκτέλεση του εργαλείου απαιτούνται τα ορίσματα *-s, -l, -p* . Στο όρισμα *-s* ορίζεται η διεύθυνση IP του εξυπηρετητή που θα γίνει ο έλεγχος. Στο όρισμα *-l* ορίζεται μια έγκυρη ταυτότητα Jabber (JID) για τον εξυπηρετητή και στο όρισμα *-p* το συνθηματικό για την ταυτότητα που δηλώθηκε. Το όρισμα *-c* δεν είναι υποχρεωτικό για την εκτέλεση του εργαλείου και χρησιμοποιείται για τον έλεγχο της διαδικασίας αυθεντικοποίησης του χρήστη στον εξυπηρετητή. Για την χρήση του ορίσματος *-c* δεν απαιτείται τα στοιχεία αυθεντικοποίησης (*-l, -p*) στον εξυπηρετητή να είναι έγκυρα, πρέπει όμως να είναι συντακτικώς ορθά (username@server).

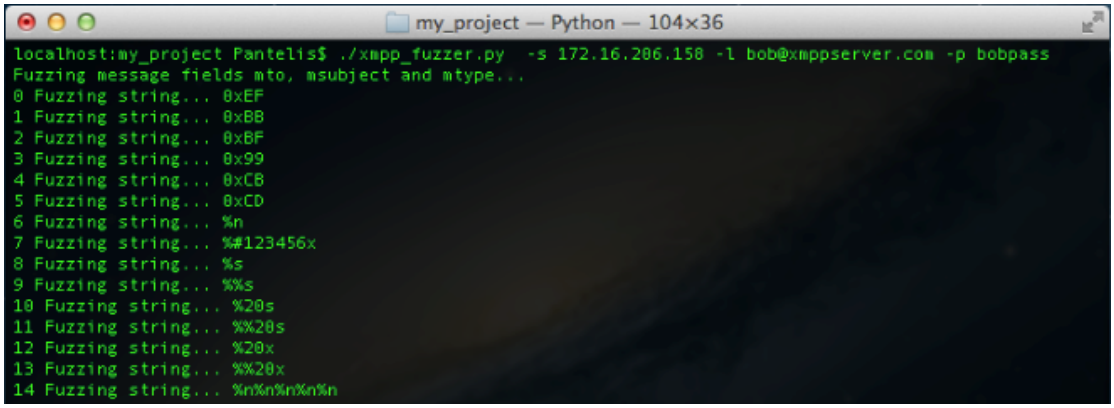
Για παράδειγμα, για τον έλεγχο της διαδικασίας αυθεντικοποίησης σε ένα εξυπηρετητή με IP 172.16.206.158 η εντολή που πρέπει να εκτελεστεί είναι : *./xmpp_fuzzer.py -c -s* 172.16.206.158 *-l random@random -p randompass* (*Εικόνα 10*).



Εικόνα 10. Εκτέλεση ελέγχου αυθεντικοποίησης

Όταν εκτελεστεί το εργαλείο εμφανίζει στην οθόνη τον τύπο ελέγχου που πραγματοποιεί (Fuzzing JID fields…). Στην συνέχεια εμφανίζει τις παραμορφωμένες ακολουθίες χαρακτήρων που δοκιμάζονται στον εξυπηρετητή.

Για τον έλεγχο της διαδικασίας ανταλλαγής μηνυμάτων πρέπει τα διαπιστευτήρια που θα δηλωθούν να είναι έγκυρα έτσι ώστε το εργαλείο να μπορεί να αυθεντικοποιηθεί στον εξυπηρετητή. Έστω ότι στον εξυπηρετητή με διεύθυνση IP 172.16.206.158 και όνομα *xmppserver.com* έχουμε τον εγγεγραμμένο λογαριασμό χρήστη *bob* με συνθηματικό *bobpass.* Η εντολή σε αυτή την περίπτωση θα είναι *./xmpp_fuzzer.py -s* 172.16.206.158 *-l bob@xmppserver.com -p bobpass* (*Εικόνα 11*).



Εικόνα 11. Έλεγχος ανταλλαγής μηνυμάτων

Στην οθόνη εμφανίζονται τα πεδία τα οποία ελέγχει το εργαλείο και είναι: ο αποδέκτης του μηνύματος (mto), το θέμα του μηνύματος (msubject) και ο τύπος του μηνύματος (mtype).

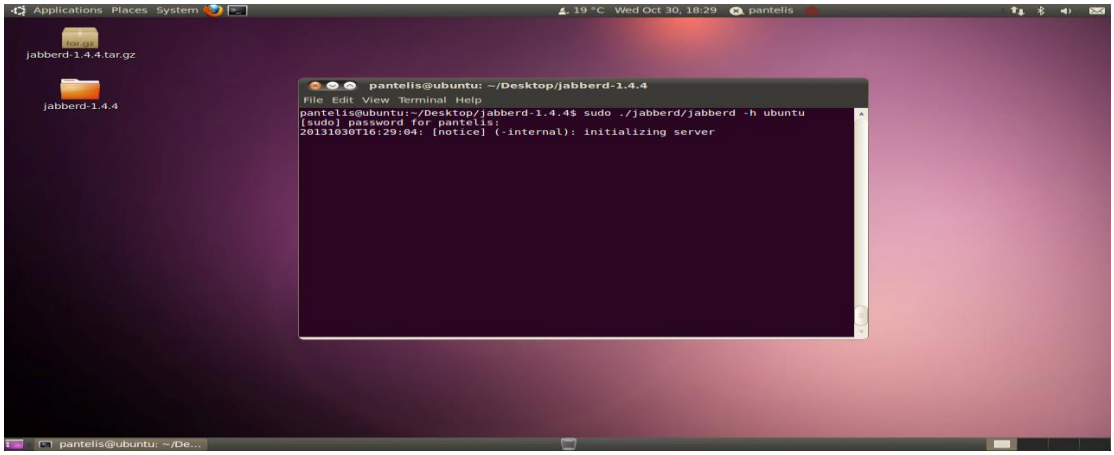
Η χρήση του εργαλείου είναι πολύ απλή. Το εργαλείο τερματίζει την λειτουργία του όταν ελεγχθούν οι περίπου τριακόσιες παραμορφωμένες ακολουθίες χαρακτήρων που βρίσκονται στο αρχείο *fuzz_strings.txt*. Ο χρήστης έχει την δυνατότητα να εισάγει στο αρχείο *fuzz_strings.txt* παραμορφωμένες ακολουθίες χαρακτήρων για να καλύψει περιπτώσεις λαθών που επιθυμεί. Η διακοπή της λειτουργίας μπορεί να γίνει πριν την ολοκλήρωση του ελέγχου με τον συνδυασμό των πλήκτρων ctr + C.

4.5. Περιβάλλον δοκιμών

Κατά την διάρκεια, αλλά και μετά την ολοκλήρωση της ανάπτυξης του εργαλείου έγιναν δοκιμές για την ορθή λειτουργία του. Οι δοκιμές έλαβαν χώρα σε ένα ελεγχόμενο περιβάλλον με την χρήση εικονικών μηχανών (virtual machines). Το περιβάλλον δοκιμών αποτελούνταν από το κυρίως σύστημα με λειτουργικό Mac OS X 10.8 και το λογισμικό VMware Fusion 6.0. Το εργαλείο εκτελείται στο κυρίως λειτουργικό σύστημα και διενεργεί ελέγχους σε εξυπηρετητές οι οποίοι έχουν εγκατασταθεί σε εικονικά μηχανήματα. Για την παρακολούθηση της επικοινωνίας μεταξύ των δύο άκρων χρησιμοποιήθηκε το εργαλείο Wireshark στην έκδοση 1.8.10.

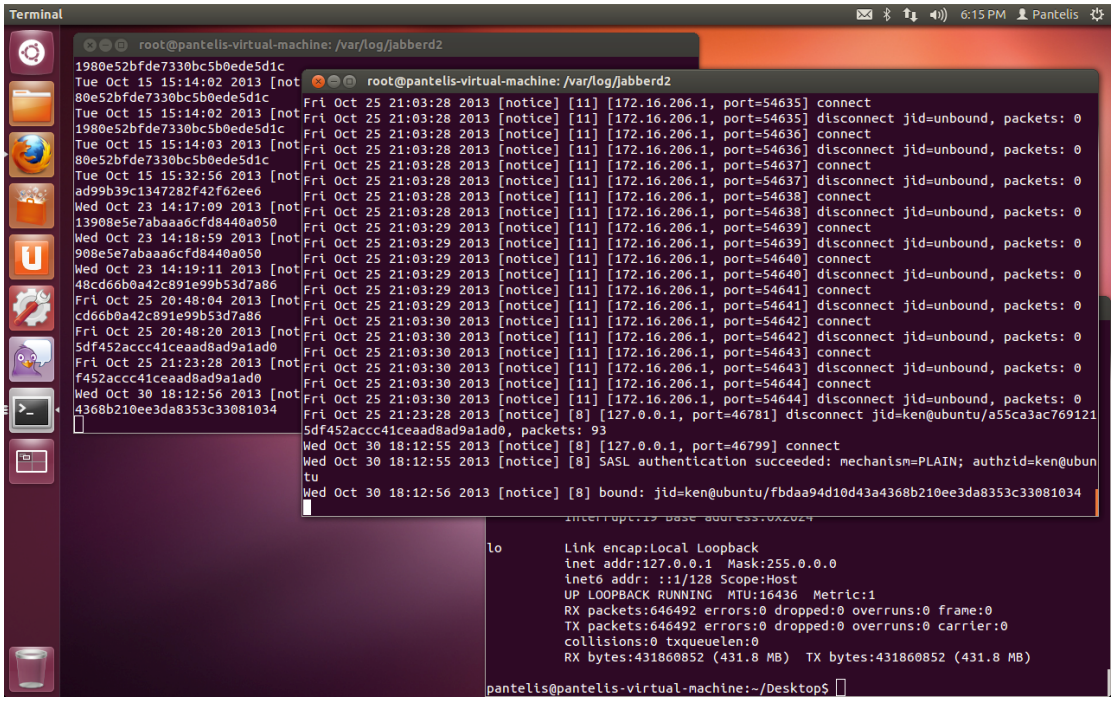
Για τις ανάγκες των δοκιμών εγκαταστάθηκαν τρεις εικονικές μηχανές με τα λειτουργικά συστήματα Ubuntu 10.04, Ubuntu 12.04 και Windows XP Professional SP 2. Σε κάθε σύστημα εγκαταστάθηκε και ένας εξυπηρετητής που προσέφερε υπηρεσίες XMPP.

Στο σύστημα με λειτουργικό Ubuntu 10.04 εγκαταστάθηκε ο εξυπηρετητής jabberd 1.4 (*Εικόνα 12*). Η εγκατάσταση του εξυπηρετητή έγινε με βάση τον οδηγό που βρίσκεται στην διεύθυνση[7].



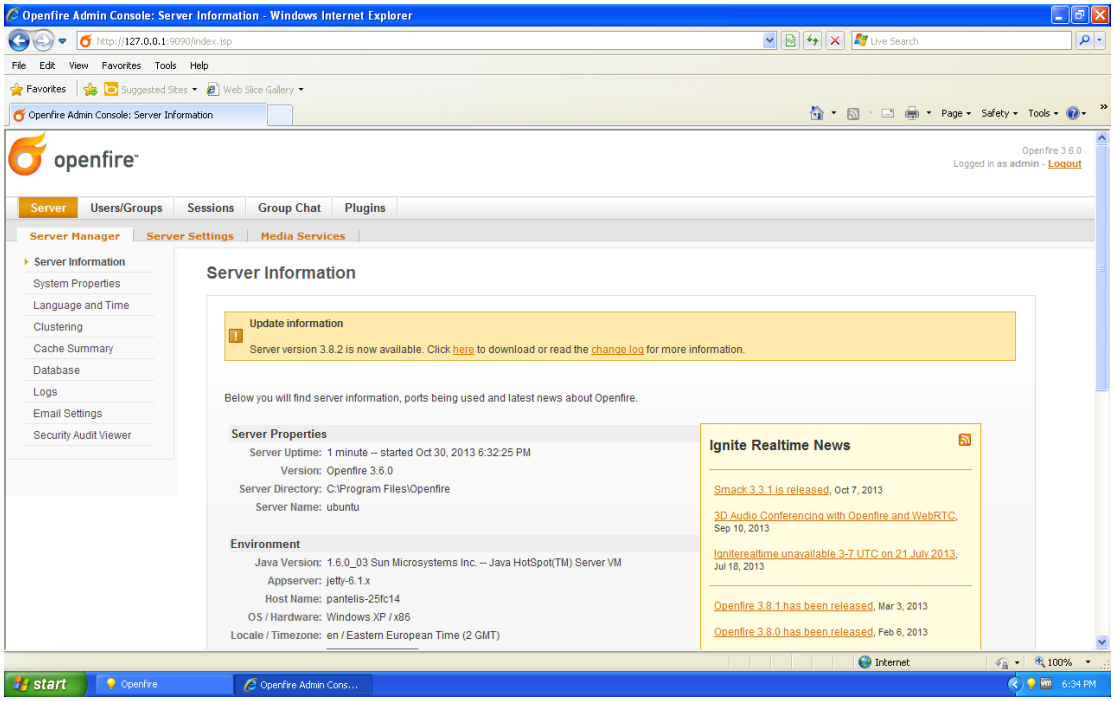
Εικόνα 12. Ο εξυπηρετητής jabberd14 σε λειτουργικό Ubuntu 10.04

Στο σύστημα με λειτουργικό Ubuntu 12.04 εγκαταστάθηκε ο εξυπηρετητής jabberd2 (*Εικόνα 13*). Η εγκατάσταση έγινε με βάση τις οδηγίες που υπάρχουν στην σελίδα του project στο Github[8].



Εικόνα 13. Ο εξυπηρετητής jabberd2 σε λειτουργικό Ubuntu 12.04

Στο σύστημα με λειτουργικό Windows XP Professional SP 2 εγκαταστάθηκε ο εξυπηρετητής Openfire (*Εικόνα 14*). Η εγκατάσταση έγινε με βάση τις οδηγίες που βρίσκονται στον ιστοσελίδα του εξυπηρετητή[9].



Εικόνα 14. Ο εξυπηρετητής Openfire σε λειτουργικό Windows XP Professional SP2

Για την αξιολόγηση του εργαλείου η διαδικασία που ακολουθήθηκε είναι η εξής. Αρχικά εκτελείται το Wireshark για να καταγράψει τα δεδομένα που ανταλλάσσονται μεταξύ του εργαλείου και του εξυπηρετητή. Στην συνέχεια εκτελείται το εργαλείο και ξεκινάει ο έλεγχος του εξυπηρετητή. Αφού ολοκληρωθεί ο έλεγχος συγκρίνονται τα δεδομένα που έχουν καταγραφεί από το Wireshark και από τα αρχεία καταγραφής (log files) του εξυπηρετητή. Σκοπός του ελέγχου αυτού είναι να διασταυρώσει ότι η επικοινωνία μεταξύ του εργαλείου και του εξυπηρετητή πραγματοποιείται χωρίς προβλήματα και ότι οι παραμορφωμένες ακολουθίες χαρακτήρων στέλνονται αναλλοίωτες στον εξυπηρετητή. Επίσης ο έλεγχος των αρχείων καταγραφής μπορεί να αποκαλύψει και σφάλματα τα οποία προκλήθηκαν από τις δοκιμές του εργαλείου.

Για την δοκιμή της διαδικασίας ανταλλαγής μηνυμάτων δημιουργήθηκε ένας δεύτερος χρήστης στον εξυπηρετητή που ορίστηκε ως παραλήπτης των μηνυμάτων. Κατά την διάρκεια των δοκιμών δεν παρατηρήθηκαν προβλήματα στην παράδοση των μηνυμάτων σχετικά με τις παραμορφωμένες ακολουθίες χαρακτήρων παρά μόνο με την όγκο των μηνυμάτων που έκανε το λογισμικό του χρήστη να σταματήσει να αποκρίνεται. Αν και αυτό μπορεί να αποτελεί ένα κενό ασφάλειας είναι πέρα από τους στόχους αυτής της εργασίας και μπορεί να εξεταστεί μελλοντικά.

5. Τεχνική ανάλυση

Στο κεφάλαιο αυτό θα γίνει η τεχνική ανάλυση του εργαλείου η οποία περιλαμβάνει την ανάλυση του πηγαίου κώδικα. Ο πηγαίος κώδικας θα χωριστεί σε κομμάτια τα οποία θα αναλυθούν έτσι ώστε να γίνει κατανοητή η λειτουργία του xmpp fuzzer.

Η λογική λειτουργίας του εργαλείου είναι απλή. Το εργαλείο διαβάζει από το αρχείο fuzz_strings.txt τις παραμορφωμένες ακολουθίες χαρακτήρων και με την χρήση της βιβλιοθήκης sleekXMPP τις εισάγει στα κατάλληλα πεδία ανάλογα με την διαδικασία που θέλει να ελέγξει ο χρήστης (αυθεντικοποίηση ή ανταλλαγή μηνυμάτων).

5.1. Εισαγωγή βιβλιοθηκών

Εξετάζοντας τον πηγαίο κώδικα αρχικά γίνεται η εισαγωγή των απαραίτητων βιβλιοθηκών (*Εικόνα 15*). Η βιβλιοθήκη sleekxmpp για την επικοινωνία με το xmpp πρωτόκολλο, η βιβλιοθήκη time για πρόσβαση σε συναρτήσεις σχετικές με τον χρόνο, η βιβλιοθήκη re για πρόσβαση σε συναρτήσεις (functions) σχετικές με τις regular expressions, η βιβλιοθήκη argparse για την διαχείριση των ορισμάτων από την γραμμή εντολών, η βιβλιοθήκη sys για την αλληλεπίδραση με το σύστημα και τέλος η βιβλιοθήκη logging για την εμφάνιση πληροφοριών κατά την εκτέλεση του εργαλείου.

```
1  #!/usr/bin/python
2
3  import sleekxmpp
4  import time
5  import re
6  import argparse
7  import sys
8  import logging
9
10 def banner():
11     """
12     XMPP Fuzzer v0.1
13
14     An XMPP Fuzzer based on a tweaked SleekXMPP: The Sleek XMPP Library
15     Copyright (C) 2010  Nathanael C. Fritz
16
17     The fields that are fuzzed are JID , "message to:" field, "subject"
18     field and "message type" field.
19
20     Pantelis Nakos, University of Piraeus, 2013
21
22     pnak.dev@gmail.com
23
24     """
25     return banner.__doc__
26
27 print banner()
28
29 #Logging levels: CRITICAL, ERROR, WARNING, INFO or DEBUG
30 #logging.basicConfig(level=logging.ERROR)
31
32 # Regular expression for a valid JID format
33 JID_PATTERN = \
34 re.compile("^(?:(.{1,10250})@)?([^/@]{1,10250})(?:/(.{1,10250}))?$")
```

Εικόνα 15. Εισαγωγή βιβλιοθηκών

Μετά την εισαγωγή των βιβλιοθηκών ακολουθεί η συνάρτηση *banner* η οποία εμφανίζει ένα μήνυμα στην οθόνη κατά την εκτέλεση του xmpp fuzzer το οποίο περιέχει πληροφορίες για το εργαλείο και στοιχεία επικοινωνίας.

Στην γραμμή 30, αφαιρώντας το σύμβολο # μπορεί να ενεργοποιηθεί η επιλογή της εμφάνισης πληροφοριών κατά την εκτέλεση. Η επιλογή αυτή είναι απενεργοποιημένη αρχικά.

Στην γραμμή 33 ορίζεται μία regular expression σύμφωνα με την μορφή που πρέπει να ακολουθεί το JID.

5.2. Η κλάση MessageFuzzer

Στην συνέχεια ακολουθεί η κλάση *MessageFuzzer* (*Εικόνα 16*). Η κλάση αυτή είναι υπεύθυνη για τον έλεγχο της διαδικασίας ανταλλαγής μηνυμάτων. «Κληρονομεί» τα χαρακτηριστικά της κλάσης ClientXMPP της βιβλιοθήκης sleekxmpp. Με την συνάρτηση __init__ δημιουργείται το αντικείμενο (object) με τα χαρακτηριστικά που θέλουμε για να πραγματοποιήσουμε τον έλεγχο στον εξυπηρετητή που έχει ορίσει ο χρήστης.

```
34  class MessageFuzzer(sleekxmpp.ClientXMPP):
35      """
36      A basic sleekXMPP bot that will log in,
37      send a message and then log out.
38
39      """
40      def __init__(self, jid, password, recipient, subject, msg_typ):
41          sleekxmpp.ClientXMPP.__init__(self, jid, password)
42          self['feature_mechanisms'].unencrypted_plain = True # Force plaintext authentication
43          self.connect(address=(args.server, 5222), use_tls = False)
44          self.process(block=False)
45          self.recipient = recipient
46          self.msg_typ = msg_typ
47          self.subject = subject
48          self.add_event_handler("session_start", self.start)
49
50      def start(self, event):
51          """
52          Process the session_start event.
53
54          """
55          self.send_presence()
56          self.get_roster()
57
58          self.send_message(mto=self.recipient, msubject = self.subject, mbody="Hello World!!!", mtype=self.msg_typ)
59
60          self.disconnect(wait=True)
61
```

Εικόνα 16. Η κλάση MessageFuzzer

Η συνάρτηση *start* ενημερώνει τον εξυπηρετητή ότι ο χρήστης είναι διαθέσιμος και ανακτά την λίστα φίλων. Μετά στέλνει ένα μήνυμα που περιέχει την παραμορφωμένη ακολουθία χαρακτήρων στα πεδία *mto, msubject, mtype* και αποσυνδέεται.

5.3. Η κλάση ConnectionFuzzer

Μετά την κλάση MessageFuzzer ακολουθεί η κλάση ConnectionFuzzer (*Εικόνα 17*). Η κλάση αυτή πραγματοποιεί ελέγχους κατά την διαδικασία αυθεντικοποίησης του χρήστη στον εξυπηρετητή. Όπως η MessageFuzzer έτσι και η ConnectionFuzzer «κληρονομεί» τα χαρακτηριστικά της ClientXMPP.

```
63  class ConnectionFuzzer(sleekxmpp.ClientXMPP):
64      """
65      A basic sleekXMPP bot that will try to log in
66      and then log out.
67
68      """
69      def __init__(self, jid, password):
70          sleekxmpp.ClientXMPP.__init__(self, jid, password)
71          self['feature_mechanisms'].unencrypted_plain = True # Force plaintext authentication
72          self.connect(address=(args.server, 5222), use_tls = False)
73          self.process(block=False)
74          self.add_event_handler("session_start", self.start, threaded=True)
75
76      def start(self, event):
77          """
78          Process the session_start event.
79
80          """
81          self.send_presence()
82          self.disconnect(wait=False)
83
```

Εικόνα 17. Η κλάση ConnectionFuzzer

Η συνάρτηση __init__ δημιουργεί το αντικείμενο με τα επιθυμητά χαρακτηριστικά και προσπαθεί να συνδεθεί στον εξυπηρετητή με την παραμορφωμένη ακολουθία χαρακτήρων. Η συνάρτηση *start* αποσυνδέεται από τον εξυπηρετητή και έτσι ολοκληρώνεται η διαδικασία ελέγχου.

5.4. Διαχείριση ορισμάτων

Μέχρι τώρα έχουν οριστεί οι βασικές κλάσεις και συναρτήσεις για την λειτουργία του εργαλείου. Στην συνέχεια ακολουθεί η διαδικασία για την διαχείριση των ορισμάτων και οι κλήσεις των κλάσεων.

Αρχικά έχουμε την διαχείριση των ορισμάτων (*Εικόνα 18*). Το εργαλείο δέχεται μέχρι τέσσερα ορίσματα. Για κάθε όρισμα συμπεριλαμβάνεται και ένα σύντομο κείμενο βοήθειας για την σωστή χρήση και σύνταξη της εντολής εκτέλεσης του εργαλείου.

```
84
85  if __name__ == '__main__':
86      try:
87          # Argument parser set-up
88          parser = argparse.ArgumentParser()
89          parser.add_argument("-s", "--server", help="XMPP server IP address to fuzz")
90          parser.add_argument("-l", "--login", help="Valid login name for message fuzzing(e.g username@domain)")
91          parser.add_argument("-p", "--password", help="Password for the login name provided")
92          parser.add_argument("-c", "--connection", help="Fuzz the JID fields(node, domain, resource)", action="store_true"
93          args = parser.parse_args()
94
95          def parse_JID(data):
96              """
97              Parse the username and domain name
98              from the JID.
99
100             """
101
102             match = JID_PATTERN.match(data)
103
104             (node, domain, resource) = match.groups()
105
106             return node, domain
107
108         jid = parse_JID(args.login)
109
110         usrname = jid[0]
111
112         domain = jid[1]
113
```

Εικόνα 18. Διαχείριση ορισμάτων

Για το όρισμα *-c* χρησιμοποιείται ο δείκτης *action=”store_true”* για να δηλώσει ότι δεν είναι υποχρεωτικό όρισμα και λαμβάνεται υπ’ όψιν μόνο όταν το θέλει ο χρήστης.

Η συνάρτηση *parse_JID* καλείται για να διαχωρίσει το όνομα χρήστη και τον όνομα του εξυπηρετητή από το JID που εισάγει ο χρήστης. Ελέγχει το JID που εισάγει ο χρήστης αν συμφωνεί με την regular expression που έχει οριστεί νωρίτερα. Αν αυτό ισχύει διαχωρίζει το όνομα χρήστη από το όνομα του εξυπηρετητή και τα αποθηκεύει σε δυο μεταβλητές για να χρησιμοποιηθούν αργότερα κατά την κλήση των κλάσεων.

5.5. Κλήση των κλάσεων

Στην γραμμή 115, το εργαλείο «ανοίγει» το αρχείο *fuzz_strings.txt* προς ανάγνωση (*Εικόνα 19*). Στην συνέχεια με ένα λογικό βρόγχο «αν» ελέγχει αν υπάρχει το όρισμα *-c* για να πραγματοποιήσει τον κατάλληλο έλεγχο. Η διαφορά ανάμεσα στους δύο ελέγχους που πραγματοποιεί το εργαλείο στην φάση αυτή είναι μόνο η κλήση διαφορετικών κλάσεων.

```
114         # Open fuzzing strings file
115         fuzz = open("fuzz_strings.txt", "r")
116 
117         if args.connection:
118             print "Fuzzing JID fields..."
119             line = fuzz.readline()
120             flag = 0
121 
122             # For every line in file call ConnectionFuzzer using the line for username
123             # and /resource.
124             while line != '':
125                 print str(flag) + " Fuzzing string... " + line.strip()
126                 ConnectionFuzzer(line.strip() + "@" + line.strip() + "/" + line.strip(), "pass")
127                 ConnectionFuzzer(line.strip() + "@" + domain + "/test", args.password)
128                 ConnectionFuzzer(usrname + "@" + line.strip() + "/test", args.password)
129                 ConnectionFuzzer(args.login + "/" + line.strip(), args.password)
130                 time.sleep(0.25)
131                 line = fuzz.readline()
132                 flag += 1
133 
134             fuzz.close()
135             sys.exit()
136 
137         else:
138 
139             print "Fuzzing message fields mto, msubject and mtype..."
140             line = fuzz.readline()
141             flag1 = 0
142 
143             # For every line in file call MessageFuzzer using the line for recipients
144             # name and message type.
145             while line != '':
146                 print str(flag1) + " Fuzzing string... " + line.strip()
147                 MessageFuzzer(args.login, args.password, line.strip() + "@" + domain, line.strip(), line.strip())
148                 time.sleep(0.25)
149                 line = fuzz.readline()
150                 flag1 += 1
151 
152             fuzz.close()
153             sys.exit()
154     except TypeError:
155         print "Wrong or no arguments given. Use -h for help."
156         sys.exit()
```

Εικόνα 19. Οι κλήσεις των κλάσεων ελέγχου

Η διαδικασία που πραγματοποιείται ξεκινάει με την «ανάγνωση» μιας γραμμής από το αρχείο *fuzz_strings.txt* και τον ορισμό μίας μεταβλητής (flag) η οποία χρησιμοποιείται ως δείκτης του αριθμού των ακολουθιών που έχουν δοκιμαστεί από το εργαλείο. Στην συνέχεια ξεκινάει η διαδικασία ελέγχου με συνεχείς κλήσεις της κατάλληλης κλάσης με την χρήση ενός επαναληπτικού βρόγχου. Ανάμεσα στις κλήσεις των κλάσεων μεσολαβεί μια καθυστέρηση της τάξης των 0.25 του δευτερολέπτου. Αυτό γίνεται καθαρά για πρακτικούς λόγους έτσι ώστε ο χρήσης να μπορεί να παρακολουθεί την διαδικασία στην οθόνη του. Η διαδικασία μετά προχωράει στην επόμενη ακολουθία κ.ο.κ. Αφού ολοκληρωθεί η διαδικασία και ελεγχθούν όλες οι ακολουθίες χαρακτήρων που βρίσκονται στο αρχείο *fuzz_strings.txt*, το εργαλείο «κλείνει» το αρχείο και τερματίζει το πρόγραμμα.

Στην περίπτωση που ο χρήστης δεν έχει εισάγει ορίσματα ή είναι λανθασμένα το εργαλείο τερματίζεται και εμφανίζει ένα μήνυμα για να ειδοποιήσει τον χρήστη. Η ειδοποίηση γίνεται με την χρήση της εντολής try - except.

6. Συμπεράσματα

Έχοντας γίνει κατανοητή η έννοια του fuzzing, διακρίνει κανείς την χρησιμότητά του και τις δυνατότητες που προσφέρει για την εύρεση λαθών στον κώδικα κατά την ανάπτυξη εφαρμογών. Αρχικά το fuzzing ήταν μια τεχνική την οποία χρησιμοποιούσαν οι ερευνητές ασφάλειας για την εύρεση αδυναμιών αλλά σύντομα αναγνωρίστηκε η χρησιμότητα της και ενσωματώθηκε στον κύκλο παραγωγής λογισμικού. Οι προγραμματιστές χρησιμοποιούν το fuzzing κατά την ανάπτυξη του λογισμικού για την εύρεση λαθών και την διόρθωσή τους πολύ πριν το λογισμικό φτάσει στην τελική του μορφή. Αυτό έχει σαν αποτέλεσμα την εξοικονόμηση χρημάτων από την ανάπτυξη διορθωτικού κώδικα (patch) και την παραγωγή ποιοτικότερων εφαρμογών.

Στην σύγχρονη βιομηχανία παραγωγής λογισμικού οι εμπλεκόμενοι με την ανάπτυξη ενός προϊόντος είναι απαραίτητο να έχουν βασικές γνώσεις ασφάλειας τις οποίες και θα ακολουθούν. Κατά τον κύκλο παραγωγής λογισμικού πρέπει να υπάρχουν διαδικασίες και εργαλεία τα οποία θα δίνουν την δυνατότητα αυτή στους εμπλεκόμενους. Το fuzzing αποτελεί μία αυτοματοποιημένη διαδικασία η οποία μπορεί να αξιοποιηθεί για τον παραπάνω σκοπό. Το fuzzing μπορεί να αποτελέσει έναν εύκολο τρόπο για την ανάπτυξη εργαλείων ασφάλειας τα οποία είναι ικανά να αποκαλύψουν μεγάλο αριθμό σφαλμάτων στον κώδικα τα οποία μπορεί να αποκαλυφθούν κατά την δυναμική εκτέλεση της εφαρμογής και όχι μέσω του στατικού ελέγχου του πηγαίου κώδικα. Με τους αυτοματισμούς που προσφέρει το fuzzing, οι παραπάνω έλεγχοι πραγματοποιούνται σε μικρότερο χρονικό διάστημα σε σχέση με συμβατικούς τρόπους εύρεσης σφαλμάτων.

Για την τεκμηρίωση των παραπάνω, στα πλαίσια της εργασίας αυτής αναπτύχθηκε ένα εργαλείο που πραγματοποιεί ελέγχους σε εξυπηρετητές που προσφέρουν υπηρεσίες XMPP. Από την διαδικασία αυτή προέκυψαν χρήσιμα συμπεράσματα. Για την ανάπτυξη ενός fuzzer σημαντικό ρόλο παίζει η γνώση του στόχου. Είτε αυτό είναι μια εφαρμογή είτε ένα πρωτόκολλο ο ερευνητής πρέπει να γνωρίζει το πως λειτουργεί έτσι ώστε να εντοπίσει τα σημεία εκείνα στα οποία είναι πιθανό να υπάρχουν σφάλματα. Το κομμάτι αυτό στην ανάπτυξη ενός fuzzer είναι το πιο σημαντικό καθώς επηρεάζει άμεσα την αποτελεσματικότητα του. Η μελέτη του στόχου πριν από τον σχεδιασμό του εργαλείου μπορεί να εξοικονομήσει χρόνο και πόρους. Τα σημεία ελέγχου περιορίζονται και δίνετε η δυνατότητα για «ποιοτικότερους» και αποτελεσματικότερους ελέγχους.

Ένα άλλο στοιχείο που επηρεάζει την αποτελεσματικότητα του fuzzing είναι οι παραμορφωμένες ακολουθίες χαρακτήρων που θα χρησιμοποιηθούν για την πραγματοποίηση των ελέγχων. Η επιλογή τους θα πρέπει να είναι τέτοια ώστε να καλύπτουν όλα τα είδη λαθών που μπορεί να εμφανιστούν στον κώδικα. Η μελέτη της εφαρμογής ή του πρωτοκόλλου αντίστοιχα, μπορεί να βοηθήσει στην επιλογή κατάλληλων παραμορφωμένων ακολουθιών χαρακτήρων οι οποίες αυξάνουν τις πιθανότητες εύρεσης σφαλμάτων. Για τον λόγο αυτό δίνεται και η επιλογή στον χρήστη να εισάγει παραμορφωμένες ακολουθίες χαρακτήρων στο εργαλείο, έτσι ώστε να έχει τον πλήρη έλεγχο της διαδικασίας.

Το fuzzing μπορεί να χρησιμοποιηθεί και ως εργαλείο αξιολόγησης εφαρμογών ή πρωτοκόλλων σε περιπτώσεις όπου ο πηγαίος κώδικας δεν είναι διαθέσιμος. Την δυνατότητα αυτή εκμεταλλεύονται κυρίως κακόβουλοι χρήστες που προσπαθούν να ανακαλύψουν κενά ασφάλειας σε κλειστές εφαρμογές ή πρωτόκολλα. Η μέθοδος του fuzzing αποτελεί μία εναλλακτική πρόταση για την διαδικασία του disassembling όντας πιο γρήγορη, ευκολότερη και με λιγότερες απαιτήσεις σε τεχνογνωσία.

Το εργαλείο xmpp fuzzer που αναπτύχθηκε στα πλαίσια της εργασίας αποτελεί τον μοναδικό fuzzer για το πρωτόκολλο XMPP. Μπορεί να χρησιμοποιηθεί για τον έλεγχο οποιουδήποτε εξυπηρετητή που προσφέρει υπηρεσίες XMPP ανεξαρτήτως πλατφόρμας. Επίσης μπορεί να χρησιμοποιηθεί και για τον έλεγχο σφαλμάτων σε εφαρμογές ανταλλαγής μηνυμάτων XMPP (XMPP clients). Κατά την διάρκεια των δοκιμών της διαδικασίας ανταλλαγής μηνυμάτων σε εξυπηρετητές τα μηνύματα στέλνονταν σε εφαρμογές ανταλλαγής μηνυμάτων XMPP για λόγους επαλήθευσης. Αρκετές εφαρμογές παρουσίασαν προβλήματα κατά την λήψη των μηνυμάτων. Μεγάλος αριθμός εφαρμογών σταμάτησε να αποκρίνεται. Μέσω των δοκιμών αυτών δίνεται η αφορμή για μελλοντική επέκταση της έρευνας στις εφαρμογές ανταλλαγής μηνυμάτων XMPP. Ο τρόπος που είναι δομημένο το εργαλείο xmpp fuzzer δίνει την δυνατότητα με ελάχιστες προσθήκες και τροποποιήσεις να χρησιμοποιηθεί για τον έλεγχο των εφαρμογών αυτών.

Κλείνοντας θα ήθελα να επισημάνω ότι σκοπός μου είναι να συνεχιστεί η υποστήριξη του εργαλείου με την προσθήκη νέων δυνατοτήτων (XMPP client fuzzing) και την πραγματοποίηση ελέγχων σε περισσότερα σημεία του εξυπηρετητή. Νέες εκδόσεις του εργαλείου θα είναι διαθέσιμες στον λογαριασμό μου στο Github (https://github.com/pnakos/my_project).

7. Βιβλιογραφία

[1] Sutton, Michael. *Fuzzing - Brute Force Vulnerability Discovery*. US: Pearson Education Inc., 2007.

[2] Peter Saint-Andre, Kevin Smith, and Remko Tronçon. XMPP: The Definitive Guide. O'Reilly Media, Inc. 2009

[3] ietf.org. RFC 2234 - Augmented BNF for Syntax Specifications: ABNF. Mar. 2012. url: http://www.ietf.org/rfc/rfc2234.txt

[4] ietf.org. RFC 6120 - Extensible Messaging and Presence Protocol (XMPP): Core. June 2012. url: http://www.ietf.org/rfc/rfc6120.txt

[5] ietf.org. RFC 3920 - Extensible Messaging and Presence Protocol (XMPP): Core. Mar. 2012. Url: http://www.ietf.org/rfc/rfc3920.txt.

[6] ietf.org. RFC 4422 - Simple Authentication and Security Layer (SASL). June 2012. Url: http://www.ietf.org/rfc/rfc4422.txt.

[7] Jabberd14 Installation Guide: https://jabberd.org/1.4/doc/adminguide

[8] Jabberd2 Installation and Administration Guide:
https://github.com/jabberd2/jabberd2/wiki/InstallGuide

[9] Openfire Installation guide:
http://www.igniterealtime.org/builds/openfire/docs/latest/documentation/install-guide.html