# Ανοικτό Πανεπιστήμιο Κύπρου

## Σχολή Θετικών και Εφαρμοσμένων Επιστημών


## Μεταπτυχιακή Διατριβή
## Στην Ασφάλεια Υπολογιστών και Δικτύων

# Automating Red Team Attacks in Cyber Ranges


**Μιχάλης Τροκκούδης**

**Επιβλέπουσα Καθηγήτρια**
**Δρ. Αδαμαντίνη Περατικού**



**Μάϊος 2022**

# Ανοικτό Πανεπιστήμιο Κύπρου

## Σχολή Θετικών και Εφαρμοσμένων Επιστημών


# Automating Red Team Attacks in Cyber Ranges


**Μιχάλης Τροκκούδης**

**Επιβλέπουσα Καθηγήτρια**
**Δρ. Αδαμαντίνη Περατικού**


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 2022**

# Περίληψη

Η παρούσα μεταπτυχιακή διατριβή έχει ως σκοπό την αυτοματοποίηση των ενεργειών που απαιτούνται για την συλλογή πληροφοριών, αναγνώριση και εκμετάλλευση ευπαθειών, καθώς και την δημιουργία ενημερωτικής αναφοράς αποτελεσμάτων, πληροφοριακών συστημάτων τύπου Virtual Machine σε περιβάλλοντα Cyber Range, με τη βοήθεια υπαρχόντων open source εργαλείων και script με στόχο την δυνατότητα εκτέλεσης του αναπτυχθέν λογισμικού από ανθρώπους με λιγοστή γνώση επί του αντικειμένου, κέρδος πολύτιμου χρόνου, σκέψης, και εν τέλη της επίτευξης του επιθυμητού αποτελέσματος.

Η έρευνα στηρίζεται στη συλλογή και ανάλυση δεδομένων με έμφαση στη πειραματική δοκιμή, με επιλογή του σπειροειδούς μοντέλου κατά την ανάπτυξη του λογισμικού.

# Summary

This dissertation aims to automate the actions required to collect information, identify and exploit vulnerabilities, as well as to create reports out of the outcome results. Virtual Machine information systems will be used, in Cyber Range environments, while with the help of existing open source tools and scripts we aim to be able to execute the developed software by people with little knowledge on the subject, saving valuable time, thinking, and finally achieving the desired result.

The research is based on data collection and analysis with emphasis on experimental testing, with the selection of the spiral model during software development.

# Ευχαριστίες

Στην επιβλέπουσα καθηγήτρια μου Δρ. Αδαμαντίνη Περατικού, για τη υπομονή της και την άψογη από κάθε άποψη καθοδήγηση της, για την εκπόνηση της διατριβής.

Στην οικογένεια μου, όπου ταυτόχρονα την αφιερώνω, καθ' ότι στάθηκε βράχος δίπλα μου, παρόλες τις δυσκολίες της καθημερινότητας μας.

# Περιεχόμενα

# Κατάλογος Πινάκων



# Κατάλογος Εικόνων

-ΛΕΥΚΗ ΣΕΛΙΔΑ-

# Κεφάλαιο 1
# Εισαγωγή

“If security were all that mattered, computers would never be turned on, let alone hooked into a network with literally millions of potential intruders”[1]. Σε απλή μετάφραση, “Αν η ασφάλεια ήταν το μόνο που είχε σημασία, οι υπολογιστές δεν θα έπρεπε ποτέ να είναι ενεργοποιημένοι, πόσο μάλλον να είναι συνδεδεμένοι σε ένα δίκτυο με κυριολεκτικά εκατομμύρια δυνητικούς εισβολείς” — Dan Farmer. Σύμφωνα με την άποψη του εν λόγω, πρωτοπόρου ερευνητή και προγραμματιστή, ο οποίος ασχολήθηκε με την ανάπτυξη συστημάτων εύρεσης ευπαθειών σε λειτουργικά συστήματα και δίκτυα υπολογιστών, η «ασφάλεια υπολογιστών», είναι όρος ο οποίος δεν συμβαδίζει με την έννοια των λέξεων, καθώς ποτέ δεν μπορεί να πει κανείς με βεβαιότητα, ότι ένας υπολογιστής ή ένα δίκτυο είναι ασφαλές χωρίς να υπάρχει η παραμικρή αμφιβολία. Επιπλέον πρέπει να σημειωθεί ότι ποτέ δεν τύγχανε ιδιαίτερης σημασίας και αναγνώρισης η ασφάλεια από τα θύματα των επιθέσεων, λόγω της μειωμένης ενημέρωσης και επίγνωσης των αποτελεσμάτων που προκύπτουν από αυτές.

Η είσοδος στη ψηφιακή εποχή απέδειξε την ολοένα περισσότερη, αλλά και παράλληλα, άκρως επιθετική νοοτροπία που επικρατεί στους κύκλους του κυβερνοχώρου. Η απαιτήσεις στον τομέα αυτό, είναι συνεχώς αυξανόμενες καθώς η ασφάλεια των πληροφοριών έχει γίνει αναπόσπαστο μέρος της κοινωνίας μας. Η ραγδαία ανάπτυξη των δυνατοτήτων των υπολογιστών της τελευταίας δεκαπενταετίας, έφερε στο προσκήνιο απειλές οι οποίες εξελίσσονται συνεχώς, ενώ η ανταπόκριση και η άμβλυνση μπορούν να γίνουν προκλητικές για οργανισμούς και άτομα, καθώς δημιουργούν δυσανάλογα κόστη λόγω των ασύμμετρων και μη προβλέψιμων αποτελεσμάτων που πιθανόν να υποστούν. Καθημερινά χιλιάδες εταιρείες, μικρές και μεγάλες, ανά το παγκόσμιο πέφτουν θύματα από κακόβουλα λογισμικά κυβερνοεπιθέσεων, όπως phishing, ransomware, trojan horse ή και καταστροφή hardware υπολογιστών από κάποιο «ιό». Επίσης πολλές είναι και οι περιπτώσεις διαρροής ευαίσθητων προσωπικών δεδομένων, κωδικών πρόσβασης σε υπηρεσίες, αριθμούς πιστωτικών καρτών, ιατρικά δεδομένα κτλ.

Σύμφωνα με τα δεδομένα που έχουν συλλεχθεί, [2] η Verizon (2018) αναφέρει ότι το 58 τοις εκατό των θυμάτων παραβίασης δεδομένων είναι μικρές επιχειρήσεις. Έχοντας αυξημένη επίγνωση των απειλών για την ασφάλεια (Pritchard, 2010), οι μικρές και μεσαίες επιχειρήσεις εξακολουθούν επί το πλείστο να είναι εκτεθειμένες, από ανεπαρκείς ή και μηδενικές ενέργειες στον τομέα της ασφάλειας των πληροφοριών. Το εν λόγω φαινόμενο εξακολουθεί να υφίσταται λόγω περιορισμένων προϋπολογισμών και τεχνογνωσίας. Αν και ορισμένοι ηγέτες μικρών επιχειρήσεων μπορεί να βλέπουν τους οργανισμούς τους ως ασήμαντους και απίθανους στόχους, οι επιτιθέμενοι προσπαθούν τακτικά να αξιοποιήσουν ερείσματα σε δίκτυα προμηθευτών συνεπώς μικρομεσαίων επιχειρήσεων και δια μέσω τους την πρόσβαση σε μεγαλύτερους οργανισμούς, όπως αποδείχθηκε στην παραβίαση του Πυρηνικού προγράμματος του Ιράν από το γνωστό σκουλήκι ονομαζόμενο Stuxnet.

Επιπλέον, το Ινστιτούτο Ponemon (2018) υπολόγισε ένα κόστος 41,55 $ ανά αρχείο για μια παραβίαση δεδομένων που αφορούσε 1 εκατομμύριο παραβιασμένες εγγραφές. Ωστόσο, δεδομένου ότι το εκτιμώμενο κόστος ανά εγγραφή ισοπεδώνεται από 15,64 $ για 10 εκατομμύρια δίσκους σε 7,63 $ για 50 εκατομμύρια εγγραφές, οι μεγάλοι οργανισμοί μπορούν να απορροφήσουν καλύτερα τα έξοδα που σχετίζονται με μια μεγάλη παραβίαση από τις μικρές επιχειρήσεις από μικρές παραβιάσεις. Έχοντας κατά νου την αξία των πληροφοριών και τις επιπτώσεις που μπορεί να επιφέρουν κατά την απώλεια δεδομένων από μια επίθεση, αντικειμενικός σκοπός είναι η αποτροπή αυτών.

Το θέμα της παρούσας διατριβής ασχολείται με τους αυτοματισμούς επιθέσεων των κόκκινων ομάδων (red teamers) σε Cyber-Range περιβάλλοντα. Οι δοκιμές των red teamers σε πεδία ασκήσεων θα τους βοηθήσουν, να αυξήσουν την ετοιμότητα και την επιδεξιότητά τους και να ανελιχθούν σε έμπειρους red teamers των οποίων οι «επιθέσεις» είναι ουσιώδης για την επιβίωση οργανισμών στο ανταγωνιστικό ψηφιακό και οικονομικό τους περιβάλλον. Σε αυτές τις ασκήσεις, οι «κόκκινες ομάδες» εξωτερικής ή εσωτερικής προέλευσης, εξομοιώνουν τους επιτιθέμενους και προσπαθούν να διασπάσουν όλες τις πτυχές της ασφαλούς υπόστασης του επιχειρησιακού δικτύου ενός οργανισμού, εξαπολύοντας επανειλημμένες και πολύπλοκες επιθέσεις εναντίον του.

Ουσιαστικά οι Red Teamers ως οι ethical hackers, βρίσκονται απέναντι από την παραδοσιακή άμυνα του δικτύου υπολογιστών, οι οποίοι διαδραματίζουν ουσιαστικό ρόλο στην αξιολόγηση της ασφάλειας ενός δικτύου διερευνώντας το ενεργά για αδυναμίες και τρωτά σημεία. Σε αντίθεση με τις δοκιμές διείσδυσης, που συνήθως επικεντρώνονται στην εκμετάλλευση τρωτών σημείων, οι κόκκινες ομάδες αξιολογούν ολόκληρη την κατάσταση ενός δικτύου μιμούμενοι πραγματικούς αντιπάλους, συμπεριλαμβανομένων των τεχνικών - τακτικών, διαδικασιών και στόχων τους. Κατά τον έλεγχο της ασφάλειας των συστημάτων επιτίθενται με αποστολή να διεισδύσουν στα σύστημα και να λάβουν υπό την κατοχή τους περιουσιακά στοιχεία, όχι για να καταχραστούν, αλλά να υποβοηθήσουν στη δοκιμή της άμυνας. Οι εν λόγω επιθέσεις πραγματοποιούνται προληπτικά και προς αποφυγή πραγματικών δυσμενών καταστάσεων.

Πολύ σημαντικό και κρίσιμο σημείο είναι το επίπεδο εκπαίδευσης και γνώσης των μελών της κόκκινης ομάδας, καθώς εάν τα μέλη της δεν είναι καλά εκπαιδευμένα, τότε το όφελος από αυτούς τους ελέγχους θα είναι ανούσιο. Επιδίωξη είναι η λύση αυτού το προβλήματος δημιουργώντας ένα πλαίσιο για αυτοματοποιημένη προσομοίωση κόκκινων ομάδων, επικεντρωμένο στις ενέργειες που πραγματοποιεί η κόκκινη ομάδα χρησιμοποιώντας μια σειρά από εργαλεία ανοιχτού κώδικα, το κάθε ένα με ξεχωριστή και αναγκαία λειτουργία, για την επίτευξη του τελικού επιθυμητού σκοπού, την αναγνώριση του αμυνόμενου, την ανεύρεση και εκμετάλλευση των αδυναμιών του και εν τέλη την πλήρη ενημέρωση των πεπραγμένων, ούτως ώστε ο αμυνόμενος να επιλύσει τα προβλήματα του, σε μελλοντικό, αλλά άμεσο χρόνο.

## 1.1 Αναγκαιότητα και Σπουδαιότητα Έρευνας

Με την εξέλιξη, την αύξηση των απειλών και των επιθέσεων των τελευταίων ετών, ένεκα και της πανδημίας η οποία προκάλεσε την έκρηξη χρήσης σε ηλεκτρονικά διαδικτυακά περιβάλλοντα, χρειάζεται καταρχήν η ευαισθητοποίηση όλων σε ότι αφορά τις προκλήσεις του διαδικτύου και τι αυτό μπορεί να προκαλέσει σε φυσικά ή και νομικά πρόσωπα.

Για αυτό χρειάζονται νέα σύγχρονα περιβάλλοντα εκπαίδευσης και εκμάθησης, για την υποστήριξη των προκλήσεων των ψηφιακών οργανισμών και κοινωνιών. Ωστόσο, με την αύξηση του αυτοματισμού των συστημάτων, καθίσταται ακόμη πιο σημαντικό, οι μέθοδοι εκμάθησης και κατάρτισης για την ασφάλεια των πληροφοριών, επίσης να εξελιχθούν. Η παρούσα έρευνα θα μπορεί μέσα από εκπαιδευτικό περιβάλλον (cyber Range) να εκπαιδεύει προσωπικό ούτως ώστε μετέπειτα, ο εκπαιδευόμενος να εντοπίζει με την χρήση του αυτοματοποιημένου εργαλείου, τις ευπάθειες του οργανισμού του σε πραγματικό περιβάλλον και χρόνο.

Σε κάθε περίπτωση χρήσης του αυτοματοποιημένου λογισμικού, σκιαγραφεί την ευελιξία της γκάμας στον κυβερνοχώρο. Συνολικά, οι περιπτώσεις χρήσης που συμβάλλουν στην ανάπτυξη ικανοτήτων και δεξιοτήτων του προσωπικού αλλά και στη διερεύνηση και συμβολή σε νέες ερευνητικές κατευθύνσεις, (π.χ. προσομοίωση και ανίχνευση επιθέσεων) καθώς ο τομέας της επίθεσης και της άμυνας πληροφοριακών συστημάτων δεν θα πάψει ποτέ να εξελίσσεται και να πολλαπλασιάζεται, με νέες ευφάνταστες τακτικές και μεθοδολογίες.

Το θέμα αυτό όμως, της εκπαίδευσης δεν μπορεί να γίνει άμεσα, αλλά σε ένα βάθος χρόνου, με την εκτίμηση να υπολογίζεται περίπου σε πενταετία. Κάπου εδώ προκύπτει η ανάγκη για προσωπικό red teaming το οποίο να είναι ικανό να εκτελέσει δοκιμές επιθέσεων σε μικρομεσαίους, αλλά και μεγάλους οργανισμούς. Η έλλειψη ανθρώπινου δυναμικού που υπάρχει σε αυτό τον τομέα, αλλά και η συνεχώς αυξανόμενη ζήτηση, λόγω των πολλαπλών επιθέσεων προκαλεί την αδυναμία εξυπηρέτησης των αναγκών όλων των οργανισμών οι οποίοι προσφεύγουν σε επαγγελματίες για να ανακαλύψουν τις αδυναμίες που υφίστανται τα δίκτυά τους, ούτως ώστε να μπορέσουν να αντισταθούν και να επιβιώσουν στις όποιες επιθέσεις δυνατόν να προκύψουν.

Το πρόγραμμά επίσης μπορεί να λειτουργήσει εκτός από μια αυτοματοποιημένη και έξυπνη κόκκινη ομάδα, που κινείται ενεργά μέσω του δικτύου στόχου για να δοκιμάσει αδυναμίες, αλλά και να εκπαιδεύσει αμυντικούς, τους λεγόμενους «Blue Teamers». Όσο σοβαρή είναι η ανεύρεση ευπαθειών σε ένα δίκτυο, άλλο τόσο είναι να υπάρχει η δυνατότητα εξεύρεσης λύσης, η οποία θα αφαιρέσει την ευπάθεια από τη φαρέτρα των επιτιθέμενων, ενώ ταυτόχρονα θα διασφαλίσει την υπόλοιπη ομαλή λειτουργία των

πληροφοριακών συστημάτων του εκάστοτε οργανισμού. Οι «Blue Teamers» έχουν ως σκοπό τη διατήρηση της ασφάλειας των οργανισμών τους σε ασφαλές περιβάλλον.

Δυστυχώς, η ανάπτυξη των Red Teaming ομάδων, για κάποιους είναι απαγορευτική. Το κόστος, η επαναληψιμότητα και η εξειδίκευση καθιστούν δύσκολη την σταθερή εκτέλεση δοκιμών κόκκινων ομάδων. Για μεγάλες και επιτυχημένες εταιρείες, με μεγάλο κύκλο εργασιών, πιθανόν να μην είναι δύσκολη η πρόσληψη προσωπικού για το συγκεκριμένο τομέα. Στην περίπτωση όμως μικρομεσαίων επιχειρήσεων όπου ο τζίρος δεν είναι τόσο αυξημένος, τότε όχι μόνο είναι απαγορευτικό το κόστος για πρόσληψη εξειδικευμένου προσωπικού, αλλά παράλληλα και η μίσθωση υπηρεσιών σε εξωτερικούς συνεργάτες. Όπως έχει σημειωθεί, η αύξηση στη ζήτηση έχει επιφέρει και την αύξηση στην προσφορά. Ως αποτέλεσμα, έχει την απαίτηση μεγάλων ποσών για την αποπληρωμή υπηρεσιών, οι οποίες πρέπει να είναι σταθερά επαναλαμβανόμενες, καθώς οι εξελίξεις το προστάζουν. Καθημερινά νέα κακόβουλα λογισμικά και ευπάθειες ανακαλύπτονται, με αποτέλεσμα τη μια μέρα ένας οργανισμός να είναι προστατευμένος από τις πλείστες ευπάθειες, ενώ την επομένη να είναι εκτεθειμένος σε εξαιρετικά μεγάλο ποσοστό, λόγω των τελευταίων ανακαλύψεων.

Εκτός των πιο πάνω, αναγκαιοί η υλοποίηση του παραπάνω αυτόματου λογισμικού για εξοικονόμηση χρόνου, καθώς με την ταχύτητα που θα προσφέρει ο αυτοματισμός και η υπολογιστική ισχύς, ο εντοπισμός και εκμετάλλευση των ευπαθειών θα επιτυγχάνονται σε ελάχιστο χρόνο. Υπάρχει έλλειψη ευχρηστίας, απλότητας, και αυτοματοποίησης των attack scripts σε περιβάλλοντα Cyber Range, ούτως ώστε να καταλήγουν γρήγορα, εύκολα και με σχετική επιτυχία σε συμπεράσματα, σε σχέση με τους στόχους που επιλέγονται για επίθεση.

## 1.2 Διάταξη Μεταπτυχιακής Διατριβής

Η Παρούσα Μεταπτυχιακή διατριβή θα αποτελείται συνολικά από έξι κεφάλαια.

α. Αρχικό κεφάλαιο η Εισαγωγή, συνοδευόμενο από την αναγκαιότητα της έρευνας.

β. Δεύτερο κεφάλαιο η Βιβλιογραφική Ανασκόπηση, κατά την οποία θα γίνει μια εκτενής μελέτη σε ότι αφορά το red teaming, το ιστορικό του καθώς και τι κατάφερε μέχρι στιγμής η επιστημονική κοινότητα.

γ. Εν συνεχεία, στο τρίτο κεφαλαίο κατά την Μεθοδολογία θα αναλυθεί ο σκοπός της έρευνας, τα βασικά ερευνητικά ερωτήματα ενώ ταυτόχρονα θα επεξηγηθεί η επιλογή του είδους της έρευνας και το μοντέλο που θα ακολουθηθεί.

δ. Στο τέταρτο κεφαλαίο θα ακολουθήσει η Υλοποίηση του θέματος της διατριβής και θα εκτελεστεί στην πράξη το ζητούμενο.

ε. Στο πέμπτο κεφαλαίο θα παρουσιαστούν τα αποτελέσματα της εν λόγω Υλοποίησης.

στ. Εν κατακλείδι στο έκτο κεφάλαιο ο Επίλογος.

# Κεφάλαιο 2
# Βιβλιογραφική Ανασκόπηση

Στο κεφάλαιο που ακολουθεί θα ασχοληθούμε με την βιβλιογραφική ανασκόπηση. Ουσιαστικά αφορά τη συλλογή επιλεγμένων δημοσιευμένων πηγών, σχετικών πάντα με το θέμα της παρούσης, με το αντικείμενο της έρευνας να εστιάζει στην διεξοδική μελέτη των υπαρχόντων red team attack scripts με σκοπό την υλοποίηση και την αυτόματη εκτέλεση τους σε Cyber Range περιβάλλοντα. Θετικό ορόσημο προς τη σωστή κατεύθυνση έχει δημιουργηθεί, καθότι έχει παρατηρηθεί σημαντική πρόοδος[2] σε σχέση με τα διάφορα προγράμματα σπουδών και την παιδαγωγική για την ασφάλεια των πληροφοριών τα τελευταία χρόνια, επιτρέποντας στους εκπαιδευόμενους να επιτίθενται και να υπερασπίζονται δίκτυα, χρησιμοποιώντας απομονωμένα εργαστηριακά Cyber Range περιβάλλοντα[3][4].

Με την παραπάνω νοοτροπία σαν στόχο, αρχικά θα ανατρέξουμε σε ιστορικά δεδομένα και γενικές πληροφορίες στο κομμάτι που αφορά το ethical hacking, από που προέκυψε, πώς μετεξελίχθηκε στην πορεία μέσα από τα χρόνια και επήλθε το λεγόμενο Red Teaming και σε ότι αυτό συμπεριλαμβάνεται π.χ Penetration Testing. Θα γίνει μια μικρή αναφορά σε ότι αφορά τα είδη των hackers μιας και υπάρχουν πολλαπλά.

Ακολούθως θα ασχοληθούμε με τα κοινά γνωρίσματα αλλά και τις διαφορές που προκύπτουν μεταξύ του Red Teaming και του Penetration Testing, σε σχέση με τη φύση της κάθε ενέργειας. Επιπρόσθετα θα εμβαθύνουμε στους τύπους των υπαρχόντων Penetration Testing, αλλά και στην μεθοδολογία και τις τακτικές που ακολουθούνται για την επίτευξη αυτών. Συγκεκριμένα θα ασχοληθούμε με τη συλλογή πληροφοριών του στόχου, το ρόλο της κοινωνικής μηχανικής στο Red Teaming, καθώς και το hardware που κυκλοφορεί για την επίτευξη των στόχων του Red Teaming και ειδικότερα κατά την κοινωνική μηχανική.

Εν συνεχεία θα διενεργηθεί μια επεξήγηση πιθανών τύπων επιθέσεων που δύναται να αντιμετωπίσουν οι οργανισμοί – στόχοι, καθώς και τη διενέργεια της αλυσίδας (Kill Chain) ανίχνευσης - εύρεσης αλλά και εκμετάλλευσης των ανευρεθέντων ευπαθειών στο εκάστοτε σύστημα. Ακολούθως θα μελετηθούν οι ενέργειες κατά την εκμετάλλευση των ευπαθειών καθώς επίσης και με το πέρας αυτής, με τα συνεπακόλουθα αποτελέσματα που αυτά θα επιφέρουν. Επιπρόσθετα θα γίνει επεξήγηση πολλαπλών εργαλείων που θα χρησιμοποιηθούν κατά την υλοποίηση του θέματος.

Επιπλέον των παραπάνω, κρίνετε σκόπιμο να αναφερθούν και οι εναλλακτικές ενέργειες που αποφασίζουν να ακολουθήσουν οργανισμοί, οι οποίοι και προφανώς θεωρούν ότι δεν μπορούν να ακολουθήσουν τον συνεχόμενο έλεγχο ευπαθειών μέσω των penetration testing και επιλέγουν άλλους τρόπους υπεράσπισης των εταιρειών τους, με ότι αυτό συνεπάγεται.

Εν τέλη θα μελετηθούν υπάρχοντα αυτοματοποιημένα συστήματα, οι τρόποι λειτουργίας τους καθώς επίσης οι δυνατότητες τους σε ότι αφορά τα πλεονεκτήματα που επιφέρουν, αλλά και τα μειονεκτήματα που πιθανόν να υπάρξουν κατά την χρήση τους, σε συνδυασμό με την υλοποίηση τους σε Cyber Range περιβάλλοντα καθώς επίσης και με την δυνατότητα αυτοματοποίησης τους, για εκτέλεση τους από προσωπικό που πιθανόν να μην είναι ιδιαίτερα σχετικό με την όλη διαδικασία

## 2.1 Ιστορική Αναδρομή – Γενικές Πληροφορίες

Η ιστορία του ηθικού hacking ξεκινά ως μια ιδέα[5]. Η λέξη εμφανίστηκε στο σύγχρονο κόσμο, όπως είναι γνωστή και σήμερα, από το διάσημο Ινστιτούτο Τεχνολογίας της Μασαχουσέτης (MIT). Η φράση «Ethical Hacking/Ηθική Πειρατεία» χρησιμοποιήθηκε για πρώτη φορά το 1995 από τον Αντιπρόεδρο της IBM, Τζον Πάτρικ.

Καθ' όλη τη διάρκεια της δεκαετίας του 1960, το hacking ήταν ένας όρος που χρησιμοποιούνταν από φοιτητές μηχανικής και σήμαινε απλώς την εύρεση διαφορετικών τρόπων βελτιστοποίησης συστημάτων και μηχανών για να λειτουργούν αποτελεσματικότερα. Σύμφωνα με την ιστορία του ηθικού hacking, το Hacking ήταν μια δημιουργική δραστηριότητα που πραγματοποιήθηκε από μερικούς από τους πιο

έξυπνους ανθρώπους στον κόσμο. Κατά τη δεκαετία του 1990, όταν η χρήση του Διαδικτύου ήταν διαδεδομένη πλέον σε όλη την υφήλιο, υπήρξε απότομη αύξηση των χάκερ, με αποτέλεσμα αυτοί να πολλαπλασιαστούν.

Στη δεκαετία του 1980 και του 1990, οι προσωπικοί υπολογιστές (PCs) κέρδισαν τεράστια δημοτικότητα. Ένα μεγάλο μέρος των προσωπικών πληροφοριών και άλλων εμπιστευτικών αρχείων αποθηκεύτηκαν με τη μορφή προγραμμάτων στους υπολογιστές. Με αυτό τον τρόπο γεννήθηκε η ιδέα στο μυαλό των χάκερ. Αντικειμενικός σκοπός τους η απόκτηση πρόσβασης σε αυτά τα συστήματα και εν συνεχεία στις αποθηκευμένες πληροφορίες τους. Αυτές οι πληροφορίες στη συνέχεια πουλήθηκαν από τους χάκερ, έχοντας τελικά επωφεληθεί τεράστιου οικονομικού κέρδους.

Οι χάκερ θεωρούνταν ως άτομα που κάθονταν κλειδωμένοι σε ένα δωμάτιο όλη την ημέρα προγραμματίζοντας για ώρες ασταμάτητα .Το χακάρισμα ήταν ένα κερδοφόρο προφίλ στα μέσα ενημέρωσης και όχι με θετικό προφίλ. Ο όρος "hacker"[6] χρησιμοποιήθηκε αρχικά για έμπειρους λάτρεις των υπολογιστών που μπορούσαν να "χακάρουν" συσκευές, μέσω τεχνικών προβλημάτων και να λάβουν προνόμια. Οι χάκερ θεωρήθηκαν εγκληματίες που χρησιμοποιούσαν τις δεξιότητές τους για να αποκτήσουν πρόσβαση σε ιδιωτικούς υπολογιστές, να κλέψουν δεδομένα και ακόμη και να εκβιάσουν επιχειρήσεις, για να παραδώσουν μεγάλα χρηματικά ποσά. Αυτού του είδους οι χάκερ είναι αυτό που περιγράφουμε σήμερα ως «Black Hat Hackers».

Οι όροι φημολογείται ότι προέρχονται από τις παλιές κινηματογραφικές ταινίες γουέστερν, όπου ο κακός φοράει μαύρο καουμπόικο καπέλο και ο καλός λευκό καπέλο.
Οι «White Hat Hackers» επιλέγουν να χρησιμοποιούν τις γνώσεις και δεξιότητες τους για καλό και όχι για κακό. Γνωστοί και ως «ηθικοί χάκερ» όπως προανάφερα, συνήθως εργάζονται για εταιρείες ως ειδικοί, που προσπαθούν να βρουν τρύπες ασφαλείας, χρησιμοποιώντας τις μεθόδους εισβολής που θα χρησιμοποιούσε ένας «Black Hat Hacker», με τη διαφορά όμως ότι είναι με την γραπτή και σύμφωνη συναίνεση του ιδιοκτήτη του συστήματος, γεγονός που καθιστά τη διαδικασία απολύτως νόμιμη.

Αντίθετα οι «Black Hat Hackers», είναι οι ανήθικοι χάκερ που χρησιμοποιούν τις μεθόδους hacking για δόλιους σκοπούς. Αυτό το χάκινγκ γίνεται παράνομα και

χρησιμοποιείται κακόβουλα για προσωπικό όφελος. Σήμερα, αποτελούν μία από τις κύριες απειλές κατά της υποδομής πληροφοριών, εκμεταλλευόμενοι τις ευπάθειες στον κώδικα και παρακάμπτοντας μέτρα ασφαλείας [6].

Υπάρχουν επίσης οι «Grey Hat Hackers», οι οποίοι οροθετούνται κάπου στην μέση κατηγορία, καθώς είναι ένα μείγμα από τους δύο προηγούμενους - τους ηθικούς και τους ανήθικους. Δεν είναι νόμιμοι εξουσιοδοτημένοι χάκερ και εργάζονται τόσο με καλές όσο και με κακές προθέσεις. Μπορούν να χρησιμοποιήσουν τις δεξιότητές τους για προσωπικό όφελος. Συνήθως, το Gray Hat Hacking γίνεται και για την ασφάλεια σε εθνικό επίπεδο.

Επιπλέον όρος για τους «White Hat Hackers» είναι η κόκκινη ομάδα (Red Team) [7]. Η Κύρια διαφορά τους είναι ότι, συνήθως είναι μια ομάδα εσωτερικών υπαλλήλων πληροφορικής που χρησιμοποιείται για την προσομοίωση των κακόβουλων ενεργειών σε αντίθεση με τους «White Hat Hackers» που συνήθως ενεργούν αυτόβουλα. Από άποψη κυβερνοασφάλειας, ο στόχος μιας κόκκινης ομάδας είναι κοινός με αυτό των «White Hat Hackers» και αυτός είναι να παραβιάσει ή να θέσει σε κίνδυνο την ψηφιακή ασφάλεια ενός οργανισμού. Η μπλέ ομάδα, από την άλλη πλευρά, χρησιμοποιείται παράλληλα ούτως ώστε να αποκρούσει την επιθετικότητα της κόκκινης ομάδας και να προστατεύσει τα αγαθά του οργανισμού. Εάν η κόκκινη ομάδα παρουσιάζεται ως ομάδα κυβερνοεγκληματιών, ο στόχος της μπλε ομάδας είναι να τους εμποδίσει να διαπράξουν μια υποθετική παραβίαση δεδομένων.

Το Red teaming ήταν μια ανακάλυψη από τον στρατό η οποία χρησιμοποιήθηκε για να αξιολογεί ρεαλιστικά τη στρατηγική δύναμη και την ποιότητα των στρατευμάτων, χρησιμοποιώντας μια εξωτερική προοπτική. Από τότε, η κόκκινη ομάδα έχει γίνει μια κοινή άσκηση εκπαίδευσης στον κυβερνοχώρο που χρησιμοποιείται από οργανισμούς στον δημόσιο και τον ιδιωτικό τομέα.

Στον τομέα της ασφάλειας του κυβερνοχώρου και της ασφάλειας των πληροφοριών, ορίζεται ως  μια διαδικασία σχεδιασμένη για τον εντοπισμό τρωτών σημείων δικτύων και συστημάτων, ακολουθώντας μια προσέγγιση παρόμοια αυτή που θα εκτελούσε ένας εισβολέας στο σύστημα. Αυτή η διαδικασία ονομάζεται επίσης «ethical hacking» αφού ο

απώτερος σκοπός της είναι η ενίσχυση της ασφάλειας. Το ηθικό hacking [6] είναι μια «τέχνη» με την έννοια ότι ο «καλλιτέχνης» πρέπει να διαθέτει τις δεξιότητες και τις γνώσεις ενός πιθανού εισβολέα (για να μιμηθεί μια επίθεση) και τους πόρους με τους οποίους μπορεί να μετριάσει τα τρωτά σημεία που χρησιμοποιούνται από τους εισβολείς.

Η ασφάλεια των πληροφοριών (Infosec) είναι ο ταχύτερα αναπτυσσόμενος τομέας στον κλάδο της Πληροφορικής (IT). Η ασφάλεια θα ήταν μια σχετικά εύκολη διαδικασία εάν το μόνο που έπρεπε να γίνει ήταν να εγκατασταθεί ένα τείχος προστασίας και ένα λογισμικό προστασίας από ιούς, αλλά η πραγματικότητα είναι ότι η διασφάλιση των πληροφοριών απαιτεί μια πολυεπίπεδη προσέγγιση. Χρειάζεται γνώση και συνεχής έρευνα για την επίτευξη της ασφάλειας, εγχείρημα ιδιαίτερα δύσκολο, απαιτητικό και χρονοβόρο.

## 2.2 Red Teaming και Penetration Testing

Το Red Teaming και το Penetration Testing είναι όροι οι οποίοι χρησιμοποιούνται συχνά εναλλακτικά για να περιγράψουν τεχνικές δοκιμών ασφαλείας. Αρκετοί θεωρούν ότι οι εν λόγω όροι είναι ένα και το αυτό, το κάθε ένα όμως εκτελεί τις δοκιμές του με διαφορετικό τρόπο. Αν και έχουν πολλά κοινά χαρακτηριστικά γνωρίσματα κρίνεται σκόπιμο να γίνει ένας διαχωρισμός, καθ' ότι στην τελική, πρόκειται για δύο διαφορετικές ενέργειες με κοινό σκοπό, την ασφάλεια.

Το Penetration Testing είναι μια ευρέως χρησιμοποιούμενη μεθοδολογική προσέγγιση που αξιολογεί την παραδοσιακή ασφάλεια του Διαδικτύου ή των συστημάτων μέσω της προσομοίωσης μιας πραγματικής επίθεσης [8]. Σύμφωνα με το PTES (penetration testing execution standard) [9] η διαδικασία του penetration testing περιλαμβάνει την αλληλεπίδραση πριν από τη δέσμευση (Pre-Engagement Interaction), τη συλλογή πληροφοριών (Information Gathering), τη μοντελοποίηση απειλών (Threat Modeling), την ανάλυση τρωτότητας (Vulnerability Analysis), την εκμετάλλευση (Exploitation), τη μετά την εκμετάλλευση (Post Exploitation) και την αναφορά (reporting). Η κύρια διαφορά μεταξύ του επιτιθέμενου εισβολέα και του Penetration Tester βασίζεται στη διαφορά του νομικού υπόβαθρου.

Οι δοκιμές διείσδυσης στοχεύουν στη βελτίωση της ασφάλειας του συστήματος παρά στην καταστροφή ή στην παράνομη πρόσβαση σε πληροφορίες και δεν επηρεάζει τη διαθεσιμότητα των συστημάτων-στόχων [10]. Κατά τη δεκαετία του 1970, ο στρατός των ΗΠΑ χρησιμοποίησε δοκιμές διείσδυσης για να ανακαλύψει πιθανά άγνωστα τρωτά σημεία και προσέλαβε χάκερ για να διερευνήσουν και να επιτεθούν σε στόχους, κάτι που επέτρεψε στους μηχανικούς λογισμικού να δημιουργήσουν ένα πιο ισχυρό σύστημα δικτύου υπολογιστών. Αναμφισβήτητα, η δοκιμή διείσδυσης είναι μία από τις πιο αποτελεσματικές μεθόδους για τη βελτίωση του επιπέδου ασφάλειας ενός συστήματος στόχου. Ένας αυξανόμενος αριθμός εταιρειών και οργανισμών έχουν αρχίσει να εκμεταλλεύονται αυτή τη μέθοδο για να εντοπίσουν και να αντιμετωπίσουν τυχόν ευάλωτα σημεία στο σύστημά τους για να αποτρέψουν μελλοντικές βλάβες. Με βάση την αρχή ότι η πρόληψη είναι η καλύτερη άμυνα, η δοκιμή διείσδυσης είναι μια προληπτική δραστηριότητα που επιτρέπει τον προσδιορισμό του εάν οι πληροφορίες είναι ασφαλείς [11].

Το Red teaming επιπλέον του Penetration Testing, είναι μια μυστική διαδικασία που συχνά στοχεύει να δοκιμάσει όχι μόνο τα συστήματα και τα πρωτόκολλα που υπάρχουν, αλλά και τους ανθρώπους που τα διαχειρίζονται [7]. Το Red teaming είναι μια εστιασμένη, στοχευμένη μέθοδος δοκιμών ασφαλείας που έχει σχεδιαστεί για την επίτευξη συγκεκριμένων στόχων. Εάν ο στόχος μιας κόκκινης ομάδας είναι να αποκτήσει πρόσβαση σε έναν ευαίσθητο διακομιστή ή σε μια κρίσιμη για τις επιχειρήσεις εφαρμογή, η επιτυχία της θα μετρηθεί από το πόσο καλά μπορεί να επιτύχει αυτόν τον στόχο.

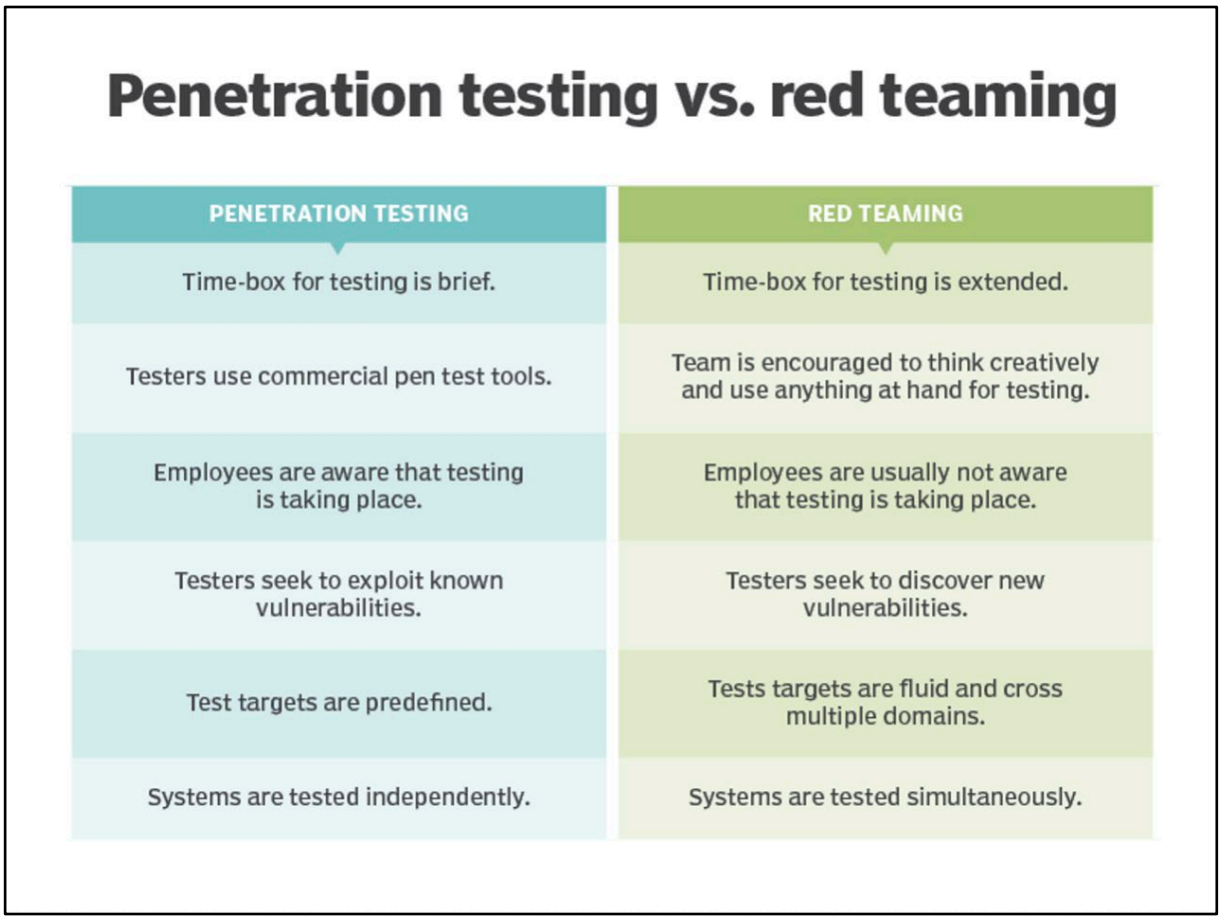
Οι Red Teamers αξιολογούν διάφορους τομείς ασφάλειας σε μια πολυεπίπεδη προσέγγιση[6]. Κάθε τομέας ασφάλειας ορίζει πώς θα αξιολογηθεί ο στόχος (σύστημα/δίκτυο). Ακολουθώντας την έννοια της Άμυνας σε βάθος, ο στόχος πρέπει να δοκιμάζεται σε κάθε επίπεδο πιθανής εισβολής/επίθεσης. Εάν η κόκκινη ομάδα πετύχει τον στόχο της, τότε η οργάνωση είναι ανεπαρκώς προετοιμασμένη για να αποτρέψει μια τέτοια επίθεση. Το Penetration Testing μπορεί να θεωρηθεί και ως αναπόσπαστο μέρος του Red Teaming, καθώς είναι μια από τις κύριες αλλά και ουσιαστικότερες ενέργειες που εκτελούνται για την επίτευξη επιτυχημένης επίθεσης.

### 2.2.1 Διαφορές μεταξύ Red Teaming και Penetration Testing

Η βασική διαφορά μεταξύ Penetration testing και Red Teaming είναι από άποψη εστίασης, λέει η Μουρ (Gemma Moore). [12] «Όταν κάνετε μια δοκιμή διείσδυσης ενός συστήματος, προσπαθείτε να καλύψετε πλήρως τα τεχνικά τρωτά σημεία εντός του καθορισμένου πεδίου εργασίας», είπε «Επομένως, αν κοιτάτε μια εφαρμογή, ψάχνετε να βρείτε όλα τα τρωτά σημεία οποιουδήποτε τύπου που υπάρχουν σε αυτήν την εφαρμογή. Αν κοιτάτε το δίκτυο, ψάχνετε να βρείτε όλα τα τρωτά σημεία που μπορείτε να εκμεταλλευτείτε».

Μια άσκηση κόκκινης ομάδας, από την άλλη πλευρά, καθοδηγείται από αντικειμενικούς σκοπούς – και η Μουρ πιστεύει ότι η «προσομοιωμένη επίθεση» είναι ένας όρος που αντικατοπτρίζει με μεγαλύτερη ακρίβεια αυτό, αν και η «κόκκινη ομάδα» υιοθετείται ευρύτερα στον κλάδο. «Αντί να πείτε, ποιες είναι όλες οι τεχνικές ευπάθειες σε αυτήν την εφαρμογή, απαντάτε σε ερωτήσεις όπως πώς θα μπορούσα να χρησιμοποιήσω αυτήν την εφαρμογή για να αποκτήσω ένα κομμάτι πολύτιμων δεδομένων – ίσως μια κρίσιμη βάση δεδομένων πελατών. Οπότε οι ερωτήσεις που κάνετε είναι πολύ διαφορετικές».

Έχοντας στο μυαλό ότι το Penetration Testing ουσιαστικά προσπαθεί να βρει όλες τις τρύπες στο σύστημά, μπορεί να θεωρηθεί γενικότερα η πιο αξιόπιστη προσέγγιση. Όμως, ενώ μπορεί να εντοπίσει και να αποκαλύψει τεχνικές ευπάθειες, σπάνια το υπερβαίνει αυτό. Όταν μια κόκκινη ομάδα ενεργεί σαν χάκερ που προσπαθούν να συμβιβάσουν έναν συγκεκριμένο στόχο, τότε ο εν λόγω στόχος αξιολογείται πολύ περισσότερο, κατά πόσο είναι αποτελεσματικές οι λύσεις ασφαλείας που έχουν εφαρμοστεί. "Είναι μια ολιστική αξιολόγηση των ανθρώπων, των διαδικασιών και της τεχνολογίας όλα μαζί." [12]




Εικόνα 1 (Διαφορές μεταξύ Penetration Testing και Red Teaming)

Όπως διαφαίνεται και στην εικόνα 1, οι κύριες διαφορές μεταξύ των Penetration Testing και Red Teaming, κυρίως εντοπίζονται στην εμβάθυνση της εκάστοτε επίθεσης και μπορούμε να πούμε εν τέλη στην ποιότητα των αποτελεσμάτων που προκύπτουν. Ενώ το Penetration Test διαρκεί αρκετές ώρες ή και ημέρες, μια Red Team ίσως χρειαστεί και μήνες για την έκδοση αποτελεσμάτων. Αυτό φυσικά οφείλεται στην πλούσια μεθοδολογία που υπάρχει, καθώς και στο γεγονός ότι σε αντίθεση με το Penetration Testing, μια Red Team δεν χρησιμοποιεί μόνο εργαλεία με στόχο την εύρεση των ευπαθειών. Μια ομάδα δύναται να εκτελέσει ενέργειες που να έχουν στο επίκεντρο την εκμετάλλευση του ανθρώπινου δυναμικού, εφαρμόζοντας κοινωνική μηχανική (Social Engineering), αλλά και καταπατώντας μέτρα της φυσικής ασφάλειας των υποστατικών ενός οργανισμού, με ότι αυτό μπορεί να επιφέρει.

Επίσης κατά τη διεξαγωγή του Penetration Testing, είναι γνωστή σε όλους ενώ αντίθετα κατά το Red Teaming, κανείς δε γνωρίζει ότι υφίσταται και ότι γίνεται ουσιαστική ενέργεια. Αυτό βοηθά στην λήψη καθαρής εικόνας, αφού διαφαίνεται ξεκάθαρα κατά πόσο οι ενέργειες των Blue Teamers είναι αρκετές ώστε να ανακόψουν τους επιτιθέμενους σε αόριστο χρόνο. Υπάρχει η δυνατότητα πάντοτε, χρήσης ευπαθειών οι οποίες δεν είναι γνωστές στην επιστημονική κοινότητα. Οι γνωστές ευπάθειες ονομαζόμενες «Zero Day» θα δώσουν το πλεονέκτημα στους Red Teamers, ενώ ταυτόχρονα θα αποκαλυφθεί, κατά πόσον θα αντιληφθεί η Blue Team την εκμετάλλευση των πληροφοριακών συστημάτων της, τις ενέργειες της για αποκατάσταση των συστημάτων, τον χρόνο που θα διαρκέσει καθώς και τη δυνατότητα να συνεχίσει ο εκάστοτε οργανισμός τις εργασίες του απρόσκοπτα.

Δεδομένου ότι στο Penetration Testing η διαδικασία εύρεσης ευπαθειών εκτελείτε σε συγκεκριμένο μηχάνημα και σε συγκεκριμένο χρόνο, κατά το Red Teaming οι ενέργειες εκτελούνται ταυτόχρονα εναντίων όλων τον στόχων, χωρίς να υπάρχει κάποια ενημέρωση για το τι θα βρεθεί υπό την επίθεση των Red Teamers. Η ρεαλιστικότητα και η αληθοφάνεια χαρακτηρίζουν το Red Teaming καθώς προσομοιώνουν πιθανά σενάρια τα οποία αν λάβουν ποτέ χώρα, θα χρειαστεί έμπειρο προσωπικό εκπαιδευμένο στο να εντοπίζει και να αντιδρά στο κάθε τι που μπορεί να αποβεί μοιραίο στον εκάστοτε οργανισμό.

## 2.3 Στρατηγικές και Τύποι Penetration Testing

Σε αυτό το τμήμα της μεταπτυχιακής διατριβής μου θα γίνει αναφορά στους τύπους αλλά και στις στρατηγικές των Penetration Testing.

### 2.3.1 Στρατηγική External Penetration Testing

Πρόκειται για επίθεση σε δίκτυο οργανισμού, όπου γίνεται χρήση διαδικασιών οι οποίες εκτελούνται εκτός των ορίων του οργανισμού, με τη χρήση διαδικτύου[13]. Αυτά τα τεστ συνήθως ξεκινούν με την συλλογή πληροφοριών οι οποίες είναι διαθέσιμες δημόσια. Στη συνέχεια ακολουθείται η συλλογή πληροφοριών όσον αφορά την τοπολογία του δικτύου, τα είδη λειτουργικών συστημάτων των οποίων γίνετε χρήση και στη συνέχεια η

στόχευση στις εξωτερικά ορατές υπηρεσίες και συσκευές δικτύου του οργανισμού, όπως π.χ το διακομιστή DNS, διακομιστή email, τείχος προστασίας ή Web διακομιστές κλπ.[14]

### 2.3.2 Στρατηγική Internal Penetration Testing

Εκτελείται σε τεχνικό περιβάλλον που προϋπάρχει μέσα σε ένα σύστημα. Αυτός ο τύπος δοκιμής προσομοιάζει επιθέσεις στη δομή του εσωτερικού δικτύου ενός συστήματος, πιθανότατα από έναν μη ικανοποιημένο υπάλληλο ενός οργανισμού ή από έναν μη εξουσιοδοτημένο χρήστη του συστήματος ο οποίος επιθυμεί να χρησιμοποιήσει το σύστημα ως μέθοδο για την προσωπική τους ψυχοθεραπεία, όπως την απομάκρυνση του θυμού και της απογοήτευσής τους, καθώς επίσης να επιτύχουν τα καταστροφικά κίνητρα τους. Το επίκεντρο των εσωτερικών δοκιμών είναι για να κατανοήσουμε ποιες θα ήταν οι δυνατότητες εάν οι παράμετροι δικτύου παραβιάζονταν επιτυχώς από τους εισβολείς μετά τη διείσδυση σε δίκτυο. Περιγράφει το «τι θα γινόταν αν;» για ένα δίκτυο, εάν ένας εξουσιοδοτημένος χρήστης αποφασίσει να κάνει κακό στον οργανισμό, ποια ζημιά μπορεί να κάνει στα συστήματα του οργανισμού, όταν έχει τη δυνατότητα να διεισδύσει στους συγκεκριμένους πόρους πληροφοριών που υπάρχουν στο δίκτυο.[13]

### 2.3.3 Στρατηγική Blind Penetration Testing

Στοχεύει στη μίμηση των ενεργειών και των διαδικασιών που ακολουθούνται από μια επίθεση σε ένα δίκτυο σε πραγματικό χρόνο. Σε αυτόν τον τύπο δοκιμής, εκτελείτε μια απόπειρα hacking όπως θα γινόταν πραγματικά. Η ομάδα δοκιμών είτε δεν λαμβάνει καμία πληροφορία, είτε λαμβάνει περιορισμένες πληροφορίες σχετικά με τον στοχοποιημένο οργανισμό, πριν από τη διεξαγωγή της δοκιμής. Σε αυτή τη διαδικασία, η ομάδα δοκιμών χρησιμοποιεί τις πληροφορίες σχετικά με τον οργανισμό που είναι διαθέσιμες δημόσια, για να συγκεντρώσει πληροφορίες σχετικά με το στόχο της και να εκτελέσει τη δοκιμή. Η δοκιμή Blind μπορεί να ανακαλύψει πολλές πληροφορίες σχετικά με τον οργανισμό που μπορεί να μην ήταν γνωστές προηγουμένως για τον οργανισμό, όπως ζητήματα που σχετίζονται με πρόσθετα σημεία πρόσβασης δικτύου, άμεσα συσχετισμένα δίκτυα και εμπιστευτικές πληροφορίες σχετικά με τον οργανισμό που μπορεί να είναι διαθέσιμες δημόσια.[13]

### 2.3.4 Στρατηγική Double Blind Penetration Testing

Είναι μια εκτεταμένη έκδοση του προηγούμενου Blind τεστ. Στη στρατηγική αυτή το προσωπικό πληροφορικής και άλλα άτομα που είναι υπεύθυνα για την ασφάλεια του οργανισμού, δεν γνωρίζουν για τις προγραμματισμένες δραστηριότητες δοκιμών διείσδυσης, ούτε ενημερώνονται σχετικά. Είναι μια πολύ σημαντική πτυχή της δοκιμής, καθώς δοκιμάζει τις παραμέτρους ασφαλείας του οργανισμού ταυτόχρονα ελέγχει τα ανταποκρινόμενα αντίμετρα που λαμβάνονται μετά τον εντοπισμό της εισβολής. Σε αυτήν τη στρατηγική δοκιμών, μόνο λίγα άτομα του οργανισμού γνωρίζουν για τις δραστηριότητες δοκιμών και τις σχετικές διαδικασίες. Σε αυτό η παρακολούθηση γίνεται από τον διαχειριστή του έργου που έχει τον έλεγχο όλων των δραστηριοτήτων δοκιμών & των παραμέτρων ανίχνευσης εισβολής δικτύου του οργανισμού. Αυτές οι διαδικασίες δοκιμών και απόκρισης μπορούν να τερματιστούν εύκολα όταν έχουν επιτευχθεί οι εκάστοτε στόχοι των δοκιμών.[13]

### 2.3.5 Στρατηγική Targeted Penetration Testing

Σε αυτήν την προσέγγιση, τόσο το τμήμα πληροφορικής του οργανισμού όσο και οι άνθρωποι μαζί με την ομάδα δοκιμών, συμμετέχουν για τη διεξαγωγή των δοκιμών. Σε αυτό, η ομάδα δοκιμών θα έχει ξεκάθαρη ιδέα σχετικά με τις δραστηριότητες δοκιμών και πληροφορίες σχετικά με τον στόχο, στον οποίο πρόκειται να πραγματοποιηθεί η παραβίαση της ασφάλειας λόγω εισβολής στο δίκτυο. Είναι μια πολύ πιο οικονομική, αλλά και αποδοτική τεχνική, καθώς εστιάζει περισσότερο στις τεχνικές πτυχές του σχεδιασμού του Δικτύου, παρά στα αντίμετρα που λαμβάνονται μετά την ανίχνευση της εισβολής. Σε αυτή τη στρατηγική δοκιμών, οι στοχευμένες δοκιμές διεξάγονται & εκτελούνται σε πολύ λιγότερο χρόνο και με ελάχιστες προσπάθειες σε σύγκριση με τις πιο πάνω Blind δοκιμές.[13]

### 2.3.6 Στρατηγική Remote Penetration Testing

Η δοκιμή απομακρυσμένης διείσδυσης[15] χρησιμοποιείται όταν ο στόχος δεν είναι φυσικά προσβάσιμος, ωστόσο η ομάδα δοκιμών μπορεί να εκτελέσει τη διαδικασία δοκιμής μέσω οποιασδήποτε επιλογής συνδεσιμότητας. Αυτή η δοκιμή γενικά εκτελείται

για να ελέγξει την προσβασιμότητα των γενικών χρηστών, ενώ η εφαρμογή ή το λογισμικό μεταδίδεται ελεύθερα παγκοσμίως. Ένα παράδειγμα για τη δοκιμή αυτή είναι, ας υποθέσουμε ότι μια ομάδα δοκιμών έλαβε πρόκληση για να δοκιμάσει το web application και έλαβε μόνο τις πληροφορίες της διεύθυνσης προορισμού. Δεδομένου ότι ο ομάδα πρέπει να ξεκινήσει τη διαδικασία δοκιμής δημιουργώντας τη σύνδεση μέσω οποιασδήποτε επιλογής συνδεσιμότητας, δεν θα έχει άλλες πληροφορίες για το σύστημα προορισμού εκτός μόνο από τη διεύθυνση.

### 2.3.7 Τύπος Black Box Penetration Testing

Σε αυτό το είδος δοκιμών, η ομάδα δοκιμών δεν έχει ιδέα για το σύστημα ή την τοπολογία του δικτύου του στόχου [16]. Αντί για τις εσωτερικές λεπτομέρειες, δίνεται κυρίως έμφαση κατά τη συλλογή πληροφοριών σχετικά με το σύστημα δικτύου στόχου. Σε αυτόν τον τύπο, οι ομάδες δεν γνωρίζουν για τα πιθανά αποτελέσματα. Στη δοκιμή Black Box, η ομάδα δεν ασχολείται με την εσωτερική δομή των πόρων δικτύου. Σε αυτόν τον τύπο

δοκιμών, το κύριο πλεονέκτημά του είναι ότι δεν απαιτούνται συγκεκριμένα τεχνικά προσόντα, καθώς επίσης ούτε συγκεκριμένες τεχνικές γνώσεις για να υλοποιηθεί.[13].

### 2.3.8 Τύπος White Box Penetration Testing

Το White-box testing γνωστό και ως clear box testing, glass box testing, transparent box testing και structural testing[15]. Σε αυτόν τον τύπο δοκιμής, η ομάδα δοκιμών έχει λάβει όλες τις πληροφορίες σχετικά με τον στόχο, οι οποίες είναι πολύ πιθανό να περιλαμβάνουν πληροφορίες σχετικά με τη δομή του δικτύου του συστήματος. Πιθανόν επίσης να περιλαμβάνει λεπτομέρειες πηγαίου κώδικα, λεπτομέρειες λειτουργικού συστήματος και διευθύνσεις IP του δικτύου. Πραγματοποιεί τη δοκιμή χρησιμοποιώντας κάλυψη κώδικα και περιλαμβάνει δοκιμή ροής διαδρομής, δοκιμή ροής δεδομένων κ.λπ. Το κύριο πλεονέκτημά του είναι ότι καλύπτει όλες τις κύριες ροές και εξασφαλίζει 100% κάλυψη κώδικα για τον εντοπισμό σφαλμάτων στο δίκτυο. Είναι σε θέση να βρίσκει τα σχεδιαστικά λάθη σε ένα δίκτυο, τα οποία μπορεί να προκύψουν λόγω των ελαττωμάτων στη λογική ροή των προγραμμάτων και μεταξύ του εκτελεστέου κώδικα.[13]

### 2.3.9 Τύπος Grey Box Penetration Testing

Κατά το Grey Box, συνήθως η ομάδα παρέχεται με περιορισμένες, ή μερικές πληροφορίες σχετικά με την εσωτερική δομή κωδικοποίησης του δικτύου που πρόκειται να στοχευτεί. Υποτίθεται ότι πρόκειται για ένα είδος επίθεσης από έναν εξωτερικό χρήστη που έχει αποκτήσει μη εξουσιοδοτημένη περιορισμένη πρόσβαση σε μια υποδομή δικτύου στον στόχο. Το κύριο πλεονέκτημά του είναι ότι δεν χρειάζεται πρόσβαση στον πηγαίο κώδικα. Επίσης, χαράσσει μια σαφή γραμμή ρόλων μεταξύ του προγραμματιστή και του υπεύθυνου δοκιμών, επιτρέποντας έτσι έναν ελάχιστο ή μικρό κίνδυνο προσωπικής σύγκρουσης μεταξύ τους. Επιπλέον, χρειάζεται να παρέχει κανείς τις λεπτομέρειες σχετικά με τις εσωτερικές δομικές πληροφορίες σχετικά με το πρόγραμμα ή το δίκτυο που πρόκειται να δοκιμαστεί.[13]

## 2.4 Τομείς Δοκιμών – Ελέγχου κατά το Penetration Testing

Η δοκιμή διείσδυσης δρα σε τρεις τομείς: δικτύου, εφαρμογών και ροής εργασιών – αντίληψης των συστημάτων. Οι τρεις τομείς είναι αλληλένδετοι και μοιράζονται έναν κοινό στόχο, τον εντοπισμό των ευπαθειών των συστημάτων και την έκθεση τους σε πιθανούς κινδύνους. Στην περίπτωση του δικτύου, γίνετε έλεγχος σε όλη τη φυσική δομή για τον εντοπισμό παραβιάσεων και απειλών που μπορεί να δημιουργήσουν ευπάθειες και κινδύνους. Κατά το Penetration Test εναντίον του τομέα εφαρμογών, περιλαμβάνει τη δοκιμή όλης της λογικής δομής του συστήματος, προσομοιώνοντας μια πραγματική επίθεση για τον έλεγχο της αποτελεσματικότητας των εφαρμογών σε θέματα ασφάλειας. Στον τελευταίο τομέα, όλες οι δοκιμές έχουν σχεδιαστεί για τη ροή εργασιών του οργανισμού και ελέγχουν την ικανότητα του οργανισμού να αποτρέπει μη εξουσιοδοτημένη πρόσβαση στα πληροφοριακά του συστήματα. [17]

### 2.4.1 Network Layer Security

Το επίπεδο δικτύου είναι υπεύθυνο για τη μετάδοση πληροφοριών μεταξύ του επιπέδου εφαρμογής και του επιπέδου αντίληψης. Το επίπεδο δικτύου είναι ένας συνδυασμός μιας ποικιλίας δικτύων συμπεριλαμβανομένου του Διαδικτύου, του δικτύου κινητής

επικοινωνίας, του δορυφορικού δικτύου, του δικτύου GSM, του GPRS, του 3G/4G/5G, του δικτύου WIFI και ούτω καθεξής. Υπάρχουν διάφορα ζητήματα ασφαλείας αυτών των δικτύων καθώς είναι ευάλωτα σε επιθέσεις DDOS, sniffing, παραβίασης δεδομένων, επανάληψης δεδομένων, παρεμβολής σήματος κ.λπ. Επιπλέον, ο συνδυασμός διαφορετικών αρχιτεκτονικών δικτύου προκαλεί επίσης νέα ζητήματα ασφάλειας. [10]

### 2.4.2 Application Layer Security

Το επίπεδο εφαρμογής παρέχει ουσιαστικά τις διαδραστικές υπηρεσίες με τους χρήστες. Είναι σε αυτό το επίπεδο που ο κύριος κίνδυνος ασφαλείας του επιπέδου εφαρμογής (παρόμοιος με άλλους) είναι η ευπάθειά του για επίθεση ανάλογα με το σενάριο π.χ. επίθεση buffer overflow, SQL injection, XSS, επίθεση κωδικού πρόσβασης και επίθεση Social Engineering. [10]

### 2.4.3 Perception Layer Security

Το επίπεδο αντίληψης, γνωστό και ως επίπεδο αναγνώρισης ή φυσικό επίπεδο, συλλέγει πληροφορίες από τον πραγματικό κόσμο και ενσωματώνει αυτές τις πληροφορίες στον ψηφιακό κόσμο μέσω RFID, αισθητήρων, δεκτών GPS και άλλων συσκευών υλικού. Από το δίκτυο αντίληψης, συγκεκριμένα ζητήματα ασφάλειας προκαλούν περισσότερες ευπάθειες και εκτελέσεις επιθέσεων, όπως το skimming, υποκλοπής (eavesdropping), πλαστογράφησης(spoofing), κλωνοποίησης, θανάτωσης (kilkling), παρεμβολής και θωράκισης κ.λπ. [10]

## 2.5 Μεθοδολογία κατά το Red Teaming

Σε αυτό το σημείο θα ασχοληθούμε με την μεθοδολογία, όσον αφορά τις ενέργειες οι οποίες υλοποιούνται αυστηρά κατά σειρά προτεραιότητας για την υλοποίηση ενός πετυχημένου έργου από τους Red teamers. Ακολουθήθηκε αρχικά το πρότυπο εκτέλεσης δοκιμών διείσδυσης (PTES) το οποίο οργανώνει τις δραστηριότητες σε επτά στάδια: Αλληλεπιδράσεις πριν από την εμπλοκή (Pre-engagement Interactions), Συγκέντρωση πληροφοριών (Intelligence Gathering), Μοντελοποίηση απειλών (Threat Modeling), Ανάλυση ευπάθειας (Vulnerability Analysis), Εκμετάλλευση(Exploitation), Μετά την

Εκμετάλλευση (Post Exploitation) και Αναφορά (Reporting). [9] Κρίνεται όμως σκόπιμο να επισημανθεί, ότι κατά τις δοκιμές σχεδίων Red Teaming, όπως έχει προαναφερθεί εξάλλου, δύναται να χρησιμοποιηθούν επιπλέον εργαλεία και τακτικές για να εκτελεστούν οι ενέργειες, όπως θα γίνονταν στην πραγματικότητα. Αυτά μπορεί να είναι οι επιθέσεις κατά της φυσικής ασφάλειας ενός οργανισμού, εύρεσης - σπάσιμο κωδικών ασύρματων δικτύων ή και εφαρμογή κοινωνικής μηχανικής στο προσωπικό του εκάστοτε οργανισμού. Όλες αυτές οι ενέργειες μπορούν να συγκαταλεχθούν στο πρότυπο του PTES και ουσιαστικά να δημιουργηθεί η μεθοδολογία η οποία θα υποστηριχθεί κατά την υλοποίηση του Red Teaming.

### 2.5.1 Pre-engagement Interactions

Ο στόχος αυτής της ενότητας του είναι να παρουσιάσει και να εξηγήσει τα διαθέσιμα εργαλεία και τεχνικές που βοηθούν σε ένα επιτυχημένο βήμα πριν από την εμπλοκή μιας δοκιμής διείσδυσης. Αφορά επί το πλείστο τις κύριες ερωτήσεις που πρέπει να απαντηθούν πριν ξεκινήσει μια δοκιμή, υπό τύπο συνάντησης, καθότι ο χαρακτήρας του τεστ διείσδυσης δεν είναι ποτέ συγκρουσιακός, αλλά βοηθητικός και αντιμετωπίζεται ως μια δραστηριότητα που αφορά τον εντοπισμό του επιχειρηματικού κινδύνου που σχετίζεται με επίθεση και όχι για το αν δύναται κάποιος να μας "χακάρει".

Είναι ουσιαστικά οι πληροφορίες οι οποίες θα υποβοηθήσουν μια ομάδα Red teaming ούτως ώστε να γνωρίσει τα όρια της στο πεδίο εφαρμογής, στην επικείμενη επίθεση αλλά και για να ξεκαθαρίσει το κόστος με το οποίο πρόκειται να επιβαρύνει τον οργανισμό. Μερικές από τις ερωτήσεις οι οποίες γίνονται, αφορούν τη διάρκεια σε χρόνο του όλου εγχειρήματος, τον τρόπο με τον οποίο πληρώνεται η ομάδα (ανά ώρα συνήθως), αν υπάρχουν συγκεκριμένες χρονικές στιγμές που επιθυμεί να εκτελεστούν οι επιθέσεις, εάν θα υλοποιηθεί web application penetration testing, wireless network penetration testing, penetration test φυσικής ασφαλείας, εφαρμογή κοινωνικής μηχανικής, προσδιορισμός του εύρους των διευθύνσεων IP και των Domains.

Αναλύονται εκτενώς οι κύριοι αλλά και οι δευτερεύοντες στόχοι για τους οποίους εκπονείται ένα Penetration Test, συνοδευόμενοι με όρους τους οποίους πρέπει ο εκάστοτε οργανισμός να γνωρίζει. Επίσης καταγράφονται επαφές σε περίπτωση

έκτακτης επικοινωνίας λόγω σοβαρών και έκτακτων περιστατικών που κάτι δεν πάει τόσο καλά ή έχει δημιουργηθεί κάποιο επικίνδυνο κενό στα συστήματα. Συνοπτικά μπορούμε να πούμε ότι πρόκειται δικλείδες ασφαλείας οι οποίες θα καθορίσουν το μέγεθος της επίθεσης, της ρεαλιστικότητας και της σχέσης μεταξύ κόστους – οφέλους. Όσα πιο λίγα από τα ερωτήματα απαντηθούν αυτομάτως θα ανεβάζουν την τιμή αλλά και την ρεαλιστικότητα της επίθεσης. Εν τέλη λαμβάνεται η ενυπόγραφη συγκατάθεση ενός οργανισμού για την δοκιμή του Red teaming, η οποία θα χρησιμοποιηθεί για την νομική κάλυψη της ομάδας ελεγκτών. [9]

### 2.5.2 Intelligence Gathering

Σε αυτή η ενότητα θα ασχοληθούμε με τις δραστηριότητες συλλογής πληροφοριών ενός τεστ διείσδυσης. Η διαδικασία σκέψης σε συνδυασμό με την αναγνώριση που εκτελείται εναντίον ενός στόχου (συνήθως εταιρικού, στρατιωτικού ή σχετικού) και η σωστή χρησιμοποίηση, βοηθά την ομάδα ελέγχου να δημιουργήσει ένα δομημένο στρατηγικά σχέδιο για την επίθεση σε έναν στόχο. Εκτελεί αναγνώριση (reconnaissance) εναντίον ενός στόχου για τη συλλογή όσο το δυνατόν περισσότερων πληροφοριών που θα χρησιμοποιηθούν κατά τη διείσδυση του στόχου κατά τη διάρκεια των φάσεων αξιολόγησης τρωτότητας και εκμετάλλευσης. Όσο περισσότερες πληροφορίες συλλεχθούν κατά τη διάρκεια αυτής της φάσης, τόσο περισσότερους φορείς επίθεσης θα δύναται να χρησιμοποιηθούν μελλοντικά.

Κάθε αξιολόγηση ασφάλειας ξεκινά με μια αυστηρή προσπάθεια συλλογής όλων των πληροφοριών πηγής για το θέμα που ενδιαφέρει. Κατά τη διαδικασία συλλογής πληροφοριών, μπορούν να αποκαλυφθούν τεχνικές λεπτομέρειες σχετικά με το εύρος IP, τις ρυθμίσεις διαμόρφωσης και τις εφαρμογές που χρησιμοποιούνται. Αυτό θα βοηθήσει την Red Team στην αλληλεπίδρασή της στα υπόψη συστήματα/δίκτυα. Οι ιστοσελίδες είναι ένας πλούτος πληροφοριών. Οι διαμορφώσεις συστημάτων, οι λεπτομέρειες του στόχου, οι πληροφορίες επικοινωνίας, τα εύρη IP, κ.λπ. μπορούν να εντοπιστούν μέσω διαδικτυακών πόρων. Μέσω της χρήσης υπηρεσιών όπως το whois, το nslookup και του Domain Name Server (DNS), μπορούν να εξαχθούν τεχνικές πληροφορίες ερωτημάτων που σχετίζονται με τον στόχο. Η αναγνώριση αποτελείται επίσης από δραστηριότητες footprinting, οι οποίες θα συζητηθούν στη συνέχεια.[18]

### 2.5.2.1 Footprinting

Το Footprinting βοηθά το προφίλ ενός οργανισμού επιτρέποντας την ανακάλυψη ονομάτων τομέα, μπλοκ δικτύου και περιοχών IP. Ενώ υπάρχουν πολλοί τύποι τεχνικών Footprinting, αυτοί στοχεύουν κυρίως στην ανακάλυψη πληροφοριών που σχετίζονται με τα ακόλουθα περιβάλλοντα: Internet, intranet, απομακρυσμένη πρόσβαση και extranet[19]. Ο κύριος στόχος του σταδίου Footprinting είναι να προσδιορίσει το εύρος στόχου.

#### 2.5.2.1.1 Αντιστοίχιση δικτύου

Η χαρτογράφηση δικτύου βοηθά στην οπτικοποίηση της διάταξης του δικτύου και αποτελεί τη βασική γραμμή του αριθμού των χρηστών και των συσκευών. Οι χάρτες δικτύου είναι μια πλούσια πηγή πληροφοριών σχετικά με τον τύπο των συσκευών που υπάρχουν και τις τεχνικές λεπτομέρειες που τις αφορούν.

#### 2.5.2.1.2 Σάρωση θυρών

Οι σαρώσεις θυρών βοηθούν στον εντοπισμό ανοιχτών θυρών και των υπηρεσιών που εκτελούνται σε αυτές τις θύρες. Αυτό είναι ιδιαίτερα σημαντικό για τον εντοπισμό ενεργών συσκευών, οι οποίες μπορούν να γίνουν πιθανοί στόχοι και τα μέσα μέσω των οποίων μπορούν να αξιοποιηθούν.

#### 2.5.2.1.3 Ασύρματη σάρωση

Κατά τους Deraison και Gula [20], όταν οι χρήστες προσθέτουν σημεία ασύρματης πρόσβασης, μπορεί να ανοίγουν το δίκτυο για μη ασφαλή πρόσβαση από απομακρυσμένους χρήστες και συνεπώς παραβιάσεις ασφαλείας. Μέσω της ασύρματης σάρωσης, μπορεί να ανιχνευτούν πληροφορίες σχετικά με ευάλωτα δίκτυα.

### 2.5.2.2 Fingerprinting

Οι δραστηριότητες δακτυλικών αποτυπωμάτων σχετίζονται με τον προσδιορισμό του τύπου λειτουργικού συστήματος που εκτελεί το (στοχευμένο) σύστημα. Συγκεκριμένες πληροφορίες λειτουργικού συστήματος (OS) θα είναι χρήσιμες κατά τη φάση της χαρτογράφησης ευπαθειών. Το Fingerprinting μπορεί να πραγματοποιηθεί μέσω σάρωσης, ανίχνευσης λειτουργικού συστήματος και αναγνώρισης υπηρεσιών. Η ανίχνευση λειτουργικού συστήματος και υπηρεσίας βοηθά στον εντοπισμό συγκεκριμένων μέσων μέσω των οποίων μπορεί να γίνει εκμετάλλευση ενός συστήματος.

Για παράδειγμα, η ανίχνευση ενός συγκεκριμένου διακομιστή web και της έκδοσης του, μπορεί να υποδεικνύει τον τύπο του λειτουργικού συστήματος που εκτελεί το μηχάνημα. Οι σαρωτές θυρών υποδεικνύουν επίσης υπηρεσίες που εκτελούνται σε συγκεκριμένες θύρες. Με αυτόν τον τρόπο, η άσκηση δακτυλικών αποτυπωμάτων χρησιμεύει για την παροχή υποκείμενων πληροφοριών για τον στόχο που μπορούν αργότερα να χρησιμοποιηθούν για την εκμετάλλευση του συστήματος.

## 2.5.2.3 Open source intelligence (OSINT)

Είναι μια μορφή διαχείρισης συλλογής πληροφοριών που περιλαμβάνει την εύρεση, την επιλογή και την απόκτηση πληροφοριών από δημοσίως διαθέσιμες πηγές και την ανάλυσή τους για την παραγωγή αξιόπιστων πληροφοριών. Χωρίζεται σε τρείς τύπους την παθητική, Ημι-παθητική και ενεργητική συλλογή πληροφοριών.

### 2.5.2.3.1 Παθητική Συλλογή Πληροφοριών

Η Παθητική Συλλογή Πληροφοριών είναι γενικά χρήσιμη μόνο εάν υπάρχει πολύ σαφής απαίτηση και βασικός παράγοντας κατά τη συλλογή των πληροφοριών είναι η αδυναμία ανίχνευσης από το στόχο. Αυτός ο τύπος συλλογής είναι τεχνικά δύσκολο να εκτελεστεί, καθώς δεν υπάρχει διακίνηση πληροφοριών από την ομάδα ελέγχου προς τον οργανισμό-στόχο, ούτε από "ανώνυμους" κεντρικούς υπολογιστές ή υπηρεσίες στο Διαδίκτυο. Αυτό σημαίνει ότι υπάρχει μόνο η δυνατότητα συλλογής πληροφοριών μόνο από αρχειοθετημένες ή αποθηκευμένες πληροφορίες. Ως εκ τούτου, ορθό είναι αυτές οι πληροφορίες να μην λαμβάνονται ως αξιόπιστες διότι μπορεί να είναι παλιές ή εσφαλμένες, καθώς περιοριζόμαστε σε αποτελέσματα που συλλέγονται από τρίτους.

#### 2.5.2.3.2 Ημι-παθητική συλλογή πληροφοριών

Κατά την ημι-παθητική, σκοπός είναι η συλλογή πληροφοριών του στόχου να διενεργούνται με μεθόδους που θα φαίνονται σαν κανονική κίνηση και συμπεριφορά στο διαδίκτυο. Γίνετε ζήτηση μόνο από δημοσιευμένους διακομιστές ονομάτων για πληροφορίες, δεν εκτελούνται σε βάθος αντίστροφες αναζητήσεις(reverse lookups) ή brute force αιτήματα DNS, καθώς επίσης δεν αναζητούνται μη δημοσιευμένοι διακομιστές. Δεν εκτελούνται portscans ή άλλα εργαλεία ανίχνευσης σε επίπεδο δικτύου αλλά εξετάζονται μόνο μεταδεδομένα (metadata) σε δημοσιευμένα έγγραφα και αρχεία, χωρίς να αναζητείται κρυφό περιεχόμενο. Επιβάλλεται η μυστικότητα σε όλες τις δραστηριότητες καθώς με το πέρας των επιθετικών ενεργειών, ο στόχος να μην μπορεί να είναι σε θέση να ανακαλύψει τις αναγνωριστικές και εν τέλη να μην μπορεί να αποδώσει τη δραστηριότητα σε κανέναν.

#### 2.5.2.3.3 Ενεργή συλλογή πληροφοριών

Η ενεργή συλλογή πληροφοριών γίνεται ενεργά έλεγχος της υποδομής του δικτύου, με τη βοήθεια portscanner εργαλείων. Γίνετε ενεργή σάρωση και απαρίθμηση όλων των ανοιχτών πορτών και ευπαθειών που πιθανόν να εντοπιστούν. Γίνετε αναζήτηση για μη δημοσιευμένους καταλόγους, αρχεία και διακομιστές.

### 2.5.3 Threat Modeling

Η μοντελοποίηση των απειλών αφορά την εφαρμογή μοντέλων για τον εντοπισμό κινδύνων ασφαλείας για σύστημα, ένα άτομο ή έναν οργανισμό. Αυτή η διαδικασία περιλαμβάνει τον εντοπισμό των πιο επιθυμητών περιουσιακών στοιχείων, τον προσδιορισμό των εσωτερικών και εξωτερικών παραγόντων απειλών, οι οποίοι έχουν τι περισσότερο ενδιαφέρον για την απόκτηση των περιουσιακών στοιχείων και την αξιολόγηση των πιθανών φορέων επίθεσης που θα μπορούσαν να επιδιώξουν. Τα μοντέλα απειλών θα πρέπει να αξιοποιούν τις πληροφορίες που συγκεντρώθηκαν από το προηγούμενο βήμα και να επικεντρωθούν στην απόσπαση των περιουσιακών στοιχείων και των ενεργειών του οργανισμού[9].

### 2.5.4 Vulnerability Analysis

Ο έλεγχος ευπαθειών είναι η διαδικασία ανακάλυψης ελαττωμάτων σε συστήματα και εφαρμογές που μπορούν να αξιοποιηθούν από έναν εισβολέα. Αυτά τα ελαττώματα μπορεί να κυμαίνονται οπουδήποτε, από εσφαλμένη διαμόρφωση κεντρικού υπολογιστή και υπηρεσίας ή ανασφαλή σχεδιασμό εφαρμογής. Αν και η διαδικασία που χρησιμοποιείται για την αναζήτηση ευπαθειών ποικίλλει, δεν παύει να εξαρτάται σε μεγάλο βαθμό από το συγκεκριμένο στοιχείο που ελέγχεται, ορισμένες βασικές αρχές ισχύουν για τη διαδικασία. Κατά τη διεξαγωγή ανάλυσης ευπάθειας οποιουδήποτε τύπου, η ομάδα ελέγχου θα πρέπει να ελέγχει κατάλληλα τη δοκιμή, ώστε να ανταποκρίνεται στους στόχους ή/και τις απαιτήσεις του επιθυμητού αποτελέσματος.[9]

Μόλις ληφθούν αρκετές πληροφορίες και διαμορφωθούν μοντέλα απειλών, οι ομάδες ελέγχου θα πρέπει να αρχίσουν να συντάξουν τις αρχικές τους προτάσεις για να αξιολογήσουν τα ύποπτα τρωτά σημεία. Αυτές οι δραστηριότητες θεωρούνται ανάλυση τρωτότητας[2]. Μια κοινή τεχνική για τον έλεγχο για τρωτά σημεία είναι μέσω της εφαρμογής αυτοματοποιημένων σαρωτών. Η Red Team μπορεί έτσι να χρησιμοποιήσει σαρωτές για να δώσει οδηγίες ως προς τον τρόπο χειρισμού ενός συστήματος/δικτύου. Οι τύποι σαρωτών περιλαμβάνουν σαρωτές Common-Gateway Interface (CGI) ή εργαλεία δέσμης ενεργειών μεταξύ τοποθεσιών. Με αυτόν τον τρόπο επιχειρείται αρχικά η ανίχνευση τρωτών σημείων. Αυτό είναι σημαντικό γιατί από την πλευρά της Κόκκινης Ομάδας, η σάρωση και η ανίχνευση μπορούν να υποδείξουν μια διαδρομή που μπορεί να χρησιμοποιηθεί για πρόσβαση σε ένα σύστημα/δίκτυο.

### 2.5.5 Expoitation

Η φάση εκμετάλλευσης μιας δοκιμής διείσδυσης εστιάζει αποκλειστικά στην καθιέρωση πρόσβασης σε ένα σύστημα, εντοπίζοντας και παρακάμπτοντας τους μηχανισμούς άμυνας και περιορισμούς ασφαλείας [21]. Εάν η προηγούμενη φάση, η ανάλυση ευπάθειας εκτελέστηκε σωστά, αυτή η φάση θα πρέπει να είναι καλά σχεδιασμένη και να είναι ακριβής. Η κύρια εστίαση είναι να εντοπιστεί το κύριο σημείο εισόδου στον οργανισμό και να εντοπιστούν στοχευμένα περιουσιακά στοιχεία υψηλής αξίας. Εάν η

φάση της ανάλυσης ευπάθειας είχε ολοκληρωθεί σωστά, θα έπρεπε να έχει συμμορφωθεί μια λίστα στόχων υψηλής αξίας[9].

Κατά τη διάρκεια της εκμετάλλευσης της ευπάθειας, διεξάγονται πιο εκλεπτυσμένες επιθέσεις κοινωνικής μηχανικής και γίνεται προσπάθεια εκμετάλλευσης συστημάτων με ύποπτες αδυναμίες που εντοπίζονται στο βήμα της ανάλυσης ευπάθειας. Παράλληλα το spear phishing και το vishing βοηθά στη στόχευση συγκεκριμένων υπαλλήλων ενός οργανισμού με σκοπό την λήψη συγκεκριμένων πληροφοριών ή και δεδομένων υψηλής αξίας. Αν και ο χρόνος που δαπανάται για διάφορες μεθόδους επίθεσης ποικίλλει σε κάθε οργανισμό, έχουν συμπεριληφθεί επιθέσεις τεχνικής και κοινωνικής μηχανικής. Πολλοί υποθέτουν ότι η εισβολή σε μια επιχείρηση θα συνίστατο κυρίως σε επιθέσεις υψηλής τεχνικής, αντιθέτως η κοινωνική μηχανική έχει αποδειχθεί πολύ πιο αποτελεσματική[2].

Μπορούν να επιχειρηθούν και πολλά διάφορα άλλα exploit τα οποία εκμεταλλευόνται εκτός όπως τα παραπάνω, εφαρμογής κοινωνικής μηχανικής, προβλήματα αρχιτεκτονικής σε λογισμικά ή και δίκτυα, διαμόρφωσης διαπιστευτηρίων αναλόγως της πολιτικής κωδικού πρόσβασης του εκάστοτε οργανισμού, διαρροής πληροφοριών (συμπερασματικά πληροφορίες από μηνύματα σφάλματος), συγκεκριμένα προσαρμοσμένα exploit ή ακόμα και zero day ή συνδυασμοί αυτών των επιθέσεων. Ότι πληροφορία κυκλοφορεί μέσω ασύρματης τεχνολογίας μπορεί να υποκλαπεί μέσω του σπάσιμου των κωδικών πρόσβασης μέσω Brute Force, η οποία θα επέτρεπε και την πρόσβαση στο σύστημα. Τα τεχνικά παραδείγματα δοκιμών διείσδυσης περιλαμβάνουν την εξαγωγή και το σπάσιμο αρχείων κωδικού πρόσβασης, τη δημιουργία σύνδεσης με σκοπό την ανίχνευση δεδομένων και την ανάκτηση ευαίσθητων πληροφοριών, τη μεταφόρτωση και εκτέλεση κακόβουλων προγραμμάτων, την εκμετάλλευση υπηρεσιών απομακρυσμένου ελέγχου μιας συσκευής και τη χρήση συλλεγόμενων πληροφοριών για την πλαστογράφηση μιας συσκευής [18].

Κατά τη διάρκεια της δοκιμής διείσδυσης, τα εντοπισμένα τρωτά σημεία μπορούν να διερευνηθούν περαιτέρω για να προσδιοριστεί ο αντίκτυπός τους. Για παράδειγμα, ένας λογαριασμός ο οποίος έχει δεν έχει κωδικό πρόσβασης επιβάλλεται όπως συνδεθεί και να καθορίσει ποια δεδομένα είναι προσβάσιμα. Άλλο παράδειγμα θα ήταν κατά την αλληλεπίδραση με μια ιστοσελίδα όταν εμφανίζεται ένα μήνυμα σφάλματος SQL, ο ομάδα

ελέγχου θα μπορούσε να εκμεταλλευτεί την ευπάθεια και να επιχειρήσει μια επίθεση SQL Injection. Ο ελεγκτής ασφαλείας πρέπει να διερευνήσει άμεσα και σε βάθος τους κινδύνους ασφαλείας, που πιθανώς οφείλονται σε περιπτώσεις κατάχρησης και λανθασμένης αρχιτεκτονικής, για να προσδιορίσει πώς συμπεριφέρεται το σύστημα υπό επίθεση. Έτσι, κατά τη φάση της δοκιμής διείσδυσης, διερευνώνται πιθανές περιοχές χειραγώγησης των συστημάτων[18].

### 2.5.6 Post Expoitation

Ο σκοπός της φάσης μετά την εκμετάλλευση είναι να προσδιοριστεί η αξία του μηχανήματος που έχει παραβιαστεί και να διατηρηθεί ο έλεγχος του μηχανήματος για μελλοντική χρήση. Η αξία καθορίζεται από την ευαισθησία των δεδομένων που είναι αποθηκευμένα και τη χρησιμότητα του μηχανήματος περαιτέρω κατά το Lateral Movement, στην προσπάθεια εύρεσης ευπαθειών του δικτύου. Οι μέθοδοι που περιγράφονται σε αυτή τη φάση προορίζονται να βοηθήσουν την ομάδα ελέγχου να εντοπίσει και να καταγράψει πιθανά ευαίσθητα δεδομένα που υπάρχουν αποθηκευμένα (Pillaging), να αναγνωρίσει ρυθμίσεις διαμόρφωσης, κανάλια επικοινωνίας και σχέσεις με άλλες συσκευές δικτύου που μπορούν να χρησιμοποιηθούν για να αποκτηθεί περαιτέρω πρόσβαση στο δίκτυο και να ρυθμιστούν μία ή περισσότερες μέθοδοι πρόσβασης στο μηχάνημα για μελλοντικές συνδέσεις.[9]

Η επιτυχής εκμετάλλευση των περιουσιακών στοιχείων των οργανισμών οδηγεί αναπόφευκτα σε πρόσθετες ευκαιρίες, καθώς οι νέες πληροφορίες που αποκτώνται ενσωματώνονται, καθώς η ομάδα ελέγχου προχωρά στις επόμενες προγραμματισμένες δραστηριότητες της. Δεδομένων των χρονικών περιορισμών στους οποίες η κόκκινη ομάδα είναι δεσμευμένη, η αξιολόγηση συνεχίζεται όσο το δυνατόν με λιγότερη καθυστέρηση. Επιπλέον πριν τη λήξη, η ομάδα ελέγχου καθαρίζει πιθανά ίχνη τα οποία έχει αφήσει κατά τη διάρκεια των ενεργειών της, τα οποία πιθανόν να μπορούν να την φωτογραφίσουν. Για την απόδειξη όμως της επιτυχούς εκμετάλλευσης ενός περιουσιακού στοιχείου από την ομάδα ελέγχου, είναι η τοποθέτηση μιας «σημαίας» η οποία επαληθεύεται από τον πελάτη μετά τη λήξη της επίθεσης. Για παράδειγμα, κατά την φυσική πρόσβαση σε μια εγκατάσταση ενός οργανισμού, θα μπορούσαν να κρύψουν ένα μικρό διακριτικό, να τραβήξουν μια φωτογραφία ή ένα βίντεο και να το αναφέρουν

λεπτομερώς στην αναφορά, ως ακράδαντο αποδεικτικό στοιχείο, το οποίο δεν επιδέχεται αμφισβήτησης.[2]

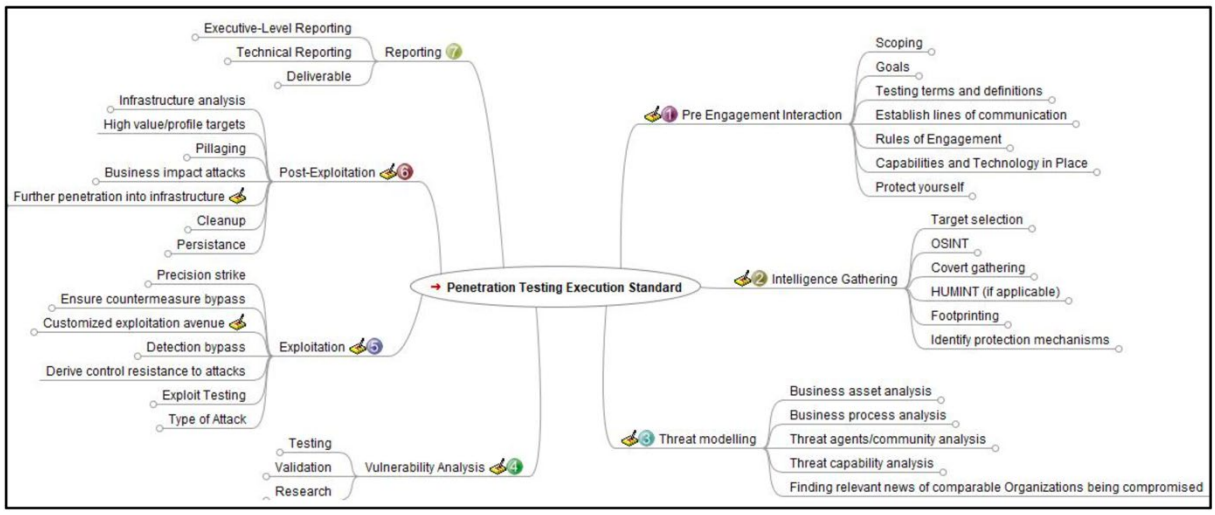
### 2.5.7 Reporting

Κατά την τελική φάση ενεργειών της Red Team, η ομάδα ελέγχου υποβάλλει στον οργανισμό όλα τα δεδομένα τα οποία έχουν συλλεχθεί. Αυτά αποτελούνται από τη συμπύκνωση των εκατοντάδων σελίδων πληροφοριών, με σκοπό να καταλήξει σε μια εύκολα κατανοητή αναφορά για τους προϊσταμένους του οργανισμού, ενέργεια ιδιαίτερα δύσκολη, καθώς είναι το έγγραφο το οποίο προορίζεται να καθορίσει τα βασικά κριτήρια για την αναφορά των δοκιμών διείσδυσης με γλώσσα η οποία τις πλείστες των περιπτώσεων είναι δυσνόητη σε ανθρώπους εκτός του κύκλου της πληροφορικής. Τα αποτελέσματα είναι οργανωμένα κατά κατηγορία και κατά την ανάπτυξη της έκθεσης, περιγράφονται επαρκώς τα ευρήματα, τεκμηριωμένα με την παροχή λεπτομερών εξηγήσεων[2]. Η έκθεση αναλύεται σε δύο μεγάλες ενότητες προκειμένου να κοινοποιηθούν οι στόχοι, στις μεθόδους οι οποίες ακολουθήθηκαν και τα αποτελέσματα των δοκιμών που λήφθηκαν.[9]

Τα γενικά ευρήματα παρέχουν μια σύνοψη των θεμάτων που εντοπίστηκαν κατά τη δοκιμή διείσδυσης σε βασική και στατιστική μορφή με γραφικές αναπαραστάσεις των στόχων που δοκιμάστηκαν, αποτελέσματα δοκιμών, διεργασίες, σενάρια επιθέσεων, ποσοστά επιτυχίας και άλλες μετρήσεις, όπως ορίστηκαν στο Pre engagement interaction. Προχωρώντας στις συστάσεις της έκθεσης παρέχονται στους υπεύθυνους του οργανισμού μια υψηλού επιπέδου κατανόηση των εργασιών που απαιτούνται να υλοποιηθούν για την εξάλειψη των ευπαθειών που εντοπίστηκαν και του γενικού επιπέδου προσπάθειας που απαιτείται για την εφαρμογή της προτεινόμενης επίλυσης.

### 2.5.8 Οπτικό διάγραμμα Μεθοδολογίας

Στον παρακάτω πίνακα φαίνεται ο χάρτης των προαναφερθέντων ενεργειών κατά τη μεθοδολογία.




*Εικόνα 2 (Ενέργειες κατά το Red Teaming) [2]*

## 2.6 Ο Ρόλος της Κοινωνικής Μηχανικής

Η κοινωνική μηχανική έχει αναδειχθεί ως σοβαρή απειλή στις εικονικές κοινότητες και είναι ένα αποτελεσματικό μέσο επίθεσης σε συστήματα πληροφοριών. Οι υπηρεσίες που χρησιμοποιούνται από τους σημερινούς εργαζόμενους, προετοιμάζουν το έδαφος για εξελιγμένες επιθέσεις κοινωνικής μηχανικής [22]. Οι δοκιμές διείσδυσης σε συστήματα πληροφορικής μερικές φορές συνδυάζονται με δοκιμές φυσικής διείσδυσης και κοινωνική μηχανική. Σε δοκιμές φυσικής διείσδυσης όπου επιτρέπεται η κοινωνική μηχανική, η ομάδα ελέγχου αλληλεπιδρά άμεσα με τους εργαζομένους. Αυτές οι αλληλεπιδράσεις βασίζονται τις πλείστες των περιπτώσεων στην εξαπάτηση και επιβάλλεται να γίνονται σωστά, ούτως ώστε να μην αναστατώσουν τους εργαζόμενους, να μην παραβιάσουν την ιδιωτικότητα ή να βλάψουν την εμπιστοσύνη τους προς τον οργανισμό οι οποίες μπορεί να οδηγήσουν σε μηνύσεις και απώλεια παραγωγικότητας.[2]

### 2.6.1 Physical Attacks

Όπως υποδηλώνει το όνομα, οι φυσικές προσεγγίσεις είναι εκείνες όπου ο επιτιθέμενος εκτελεί κάποια μορφή σωματικής ενέργειας με φυσική παρουσία, προκειμένου να συγκεντρώσει πληροφορίες για ένα μελλοντικό θύμα. Αυτό μπορεί να κυμαίνεται από προσωπικά στοιχεία (όπως αριθμός κοινωνικής ασφάλισης, ημερομηνία γέννησης) έως έγκυρα διαπιστευτήρια για ένα σύστημα υπολογιστή. Μια συχνά χρησιμοποιούμενη

μέθοδος είναι το λεγόμενο dumpster diving [23], δηλαδή η αναζήτηση στα σκουπίδια ενός οργανισμού. Ένας κάδος απορριμμάτων μπορεί να είναι μια πολύτιμη πηγή πληροφοριών για τους επιτιθέμενους, οι οποίοι μπορεί να βρουν προσωπικά δεδομένα για υπαλλήλους, εγχειρίδια, σημειώσεις και ακόμη και εκτυπώσεις ευαίσθητων πληροφοριών, όπως διαπιστευτήρια χρήστη. Εάν η ομάδα ελέγχου αποκτήσει πρόσβαση στα γραφεία του οργανισμού ο οποίος βρίσκεται υπό έλεγχο, μπορεί να βρει πληροφορίες όπως κωδικούς πρόσβασης γραμμένες σε σημειώσεις Post-it. Υπάρχουν και οι περιπτώσεις επιθέσεων κατά τις οποίες χρησιμοποιείται βία, κλοπή ή εκβιασμός για τη λήψη πληροφοριών.[22]

### 2.6.2 Technical Attacks

Οι τεχνικές επιθέσεις πραγματοποιούνται κυρίως μέσω του Διαδικτύου. Ο Granger [23] σημειώνει ότι το Διαδίκτυο είναι ιδιαίτερα ενδιαφέρον για όσους εφαρμόζουν κοινωνική μηχανική, καθ' ότι δύναται να συλλέγουν κωδικούς πρόσβασης, καθώς οι χρήστες χρησιμοποιούν συχνά τους ίδιους (απλούς) κωδικούς πρόσβασης για πολλαπλούς λογαριασμούς τους. Οι περισσότεροι άνθρωποι επίσης δεν γνωρίζουν ότι παρέχουν άφθονα προσωπικά δεδομένα στο διαδίκτυο, με αποτέλεσμα οι επιτιθέμενοι να τα χρησιμοποιούν εναντίον τους. Οι επιτιθέμενοι συχνά χρησιμοποιούν μηχανές αναζήτησης για να συγκεντρώσουν προσωπικές πληροφορίες σχετικά με μελλοντικά θύματα. Υπάρχουν επίσης εργαλεία που μπορούν να συλλέγουν και να συγκεντρώνουν πληροφορίες από διαφορετικούς πόρους Ιστού. Τα μέσα κοινωνικής δικτύωσης γίνονται επίσης πολύτιμες πηγές πληροφόρησης.[22]

### 2.6.3 Social Attacks

Η σημαντικότερη πτυχή των επιτυχημένων επιθέσεων κοινωνικής μηχανικής είναι οι κοινωνικές προσεγγίσεις. Ως εκ τούτου, οι επιτιθέμενοι βασίζονται σε κοινωνικο-ψυχολογικές τεχνικές, όπως οι αρχές της πειθούς του Cialdini [24] για να χειραγωγήσουν τα θύματά τους. Παραδείγματα μεθόδων πειθούς περιλαμβάνουν τη χρήση (υποτιθέμενης) εξουσίας. Ένας κοινός κοινωνικός φορέας που δεν αντιμετωπίζεται ρητά από τον Cialdini [24] είναι η περιέργεια, η οποία χρησιμοποιείται, για παράδειγμα, σε επιθέσεις Spear Phishing. Προκειμένου να αυξηθούν οι πιθανότητες επιτυχίας τέτοιων

επιθέσεων, οι δράστες προσπαθούν συχνά να αναπτύξουν μια σχέση με τα μελλοντικά θύματά τους. Σύμφωνα με το Granger [23] ο πιο διαδεδομένος τύπος κοινωνικών επιθέσεων εκτελείται μέσω τηλεφώνου, το λεγόμενο Vishing attack.

### 2.6.4 Social - Technical Attacks

Οι επιτυχείς επιθέσεις κοινωνικής μηχανικής συχνά συνδυάζουν πολλές ή όλες τις διαφορετικές προσεγγίσεις που συζητήθηκαν παραπάνω. Ωστόσο, οι κοινωνικοτεχνικές προσεγγίσεις έχουν δημιουργήσει τα πιο ισχυρά όπλα των κοινωνικών μηχανικών. Ένα παράδειγμα είναι το λεγόμενο baiting attack: Οι εισβολείς αφήνουν ελευθέρα μέσα αποθήκευσης, τα οποία είναι μολυσμένα με κακόβουλο λογισμικό σε τοποθεσία όπου είναι πιθανό να βρεθούν από μελλοντικά θύματα. Τα λεγόμενα «apple roads» θα μπορούσαν, για παράδειγμα, να είναι μια μονάδα USB που περιέχει έναν Trojan Horse[25]. Οι επιτιθέμενοι εκμεταλλεύονται επιπλέον την περιέργεια των ανθρώπων προσθέτοντας δελεαστικές ετικέτες σε αυτά τα «apple roads», όπως «εμπιστευτικό» ή «απόρρητο» για να αναγκάσουν την ανάγκη αυτή να επενέβη της λογικής. Η χρήση των «apple roads» γίνεται καθημερινά, ενώ έχουν σημειωθεί αμέτρητες επιτυχημένες προσπάθειες. Στόχοι βρέθηκαν μέχρι και Στρατιωτικοί σχηματισμοί οι οποίοι υπέπεσαν σε αυτού του είδους επιθέσεις.

Ένας άλλος κοινός συνδυασμός τεχνικών και κοινωνικών προσεγγίσεων είναι το phishing. Το ηλεκτρονικό ψάρεμα (phishing) γίνεται συνήθως μέσω e-mail ή άμεσων μηνυμάτων και απευθύνεται σε μια μεγάλη ομάδα χρηστών με έναν μάλλον αδιάκριτο τρόπο, παρόμοιο με το spam. Η κοινωνική μηχανική, αντίθετα, συνήθως απευθύνεται σε άτομα ή μικρές ομάδες ανθρώπων. Οι επιτιθέμενοί στοχεύουν ότι με τη μαζική αποστολή μηνυμάτων σε έναν τεράστιο αριθμό χρηστών, θα ξεγελάσουν αρκετούς ανθρώπους για να κάνουν την επίθεση phishing τους κερδοφόρα. Οι Herley και Florencio [26] υποστηρίζουν ότι το κλασικό phishing δεν είναι επικερδές, γεγονός που μπορεί να εξηγήσει γιατί οι επιθέσεις phishing πλέον κινούνται προς πιο εξελιγμένες επιθέσεις «spear-phishing». Οι επιθέσεις Spear-phishing είναι ιδιαίτερα στοχευμένα μηνύματα τα οποία κατευθύνονται σε συγκεκριμένα άτομα, συνήθως υψηλών θέσεων σε ιεραρχία ή που κατέχουν ιδιαίτερες θέσεις σε οργανισμούς στόχους , που πραγματοποιούνται μετά πολύωρη και διεξοδική συλλογή προσωπικών δεδομένων. Αυτά μπορεί να είναι

προσωπικές επιδιώξεις για ανέλιξη, πρόσφατα συμβάντα που να υποδεικνύουν προσωπικούς μελλοντικούς στόχους κλπ. Χρησιμοποιώντας τα ελεύθερα αυτά δεδομένα οι επιτιθέμενοι μπόρεσαν να αυξήσουν το ποσοστό επιτυχίας του phishing από 16 σε 72 τοις εκατό. Ως εκ τούτου, το spear-phishing θεωρείται ένας συνδυασμός τεχνολογικών προσεγγίσεων και κοινωνικής μηχανικής. [22]

## 2.7 Τύποι Επιθέσεων

Υπάρχουν πολλοί τύποι επίθεσης, όπως η επίθεση συλλογής πληροφοριών, επίθεση διαμόρφωσης, επίθεση buffer overflow, επίθεση κωδικού πρόσβασης, επίθεση ιστού, επίθεση sniffer, επίθεση κοινωνικής μηχανικής και επίθεση άρνηση υπηρεσίας (DOS ). Ο Πίνακας 2 παρέχει μια επισκόπηση των επιθέσεων στα Penetration Testing. Η πρώτη σειρά του πίνακα περιγράφει την ταξινόμηση των επιθέσεων και οι στήλες περιγράφουν αντίστοιχα τις προσεγγίσεις ή τους στόχους επίθεσης.

| Information Gathering | Configuration Error Attack | Buffer Overflow Attack | Password Attack | Web Attack | Sniffer Attack | Social Engineer Attack | Denial of Service Attack |
|---|---|---|---|---|---|---|---|
| Port | Robot.txt | FTP | HTTP/HTTPS | SQL Injection | Man In The Middle Attack | Forge Email | DDos Attack |
| Application | SSH Conf | Browser | SSH | XSS | FTP | Forge Link | SYN Flood |
| OS Type | FTP Conf | Windows | TELNET | CSRF | SSH | Forge Website | TCP Flood |
| Whois | TELNET Conf | Linux | FTP | Broken Authentication | TELNET | Forge File | ICMP Flood |
| Network Topology | Sendmail Conf | Network Device | Database | Sensitive Data Exposure | HTTP | Forge SMS | UDP Flood |
| Defense Mechanism | Web Server Conf | Web App | Mailsystem | Broken Access Control | Database | Forge Wifi | DNS Flood |
| Configuration | Database Conf | Database | VNC | Security Misconf | VNC | Spoofing | Slow POST |
| Vulnerability | VNC Conf | Mailsystem | Netbios | File Upload | Mailsystem | Physical | HTTP Flood |

*Πίνακας 1 (Επισκόπηση Επιθέσεων στο Penetration Testing)*

### 2.7.1 Denial Of Service Attack

Σε μια επίθεση DoS, οι εισβολείς προσπαθούν να αποτρέψουν την πρόσβαση των νόμιμων χρηστών σε μια υπηρεσία, αποστέλλοντας συνήθως υπερβολική ροή δεδομένων στο δίκτυο ή στο διακομιστή για να εξαντλήσει τους πόρους του. Η επίθεση DoS δεν χρησιμοποιείται συνήθως στο Penetration Testing. Μπορεί να χρησιμοποιηθεί για επανεκκίνηση του συστήματος του στόχου. Αυτό το είδος επίθεσης περιλαμβάνει επιθέσεις τύπου SYN flood, TCP/UDP, SMTP και ICMP. Εάν η επίθεση προέρχεται από διαφορετικές συσκευές ή από πολλές πηγές ταυτόχρονα τότε πρόκειται για επίθεση κατανεμημένης άρνησης υπηρεσίας (DDoS).

### 2.7.2 Social Engineering Attack

Όπως έχει προαναφερθεί και στο υποκεφάλαιο 2.6, οι επιθέσεις κοινωνικής μηχανικής στρέφονται εναντίον ανθρώπων, όπως διαχειριστών ή χρηστών, οι οποίοι έχουν γενικότερα αντίληψη σε θέματα ασφαλείας . Η κοινωνική μηχανική αναφέρεται σε μια ποικιλία κακόβουλων δραστηριοτήτων που πραγματοποιούνται μέσω ανθρώπινων αλληλεπιδράσεων. Στο απομακρυσμένο Penetration Testing, αυτές οι επιθέσεις συνήθως εκτελούνται με επιθέσεις spear-phishing μέσω email ή συνδέσμων και επιθέσεις κλωνοποιήμενων ιστοτόπων, με την εισαγωγή διαπιστευτηρίων.

### 2.7.3 Password Attack

Η επίθεση κωδικών πρόσβασης είναι ένα ουσιαστικό μέρος του Penetration Testing, καθώς ένας εισβολέας μπορεί να αποκτήσει άδεια από το στοχευμένο σύστημα, εάν μια επίθεση κωδικού πρόσβασης είναι επιτυχής. Οι περισσότερες επιθέσεις κωδικών πρόσβασης βασίζονται στη χρήση Brute Force, Dictionary Attack, Rainbow Attack (κατακερματισμένα), τα οποία βασίζονται σε κοινές λέξεις και φράσεις που χρησιμοποιούνται ως διαπιστευτήρια.

### 2.7.4 Buffer Overflow Attack

Ένα buffer overflow ουσιαστικά είναι ένα λάθος στην κωδικοποίησης του λογισμικού το οποίο εκμεταλλεύονται οι επιτιθέμενοι για να αποκτήσουν πρόσβαση στο σύστημα στόχο [56]. Κατά την εγγραφή δεδομένων σε ένα buffer, ένα πρόγραμμα υπερβαίνει τα όρια του buffer και αντικαθιστά παρακείμενες θέσεις μνήμης. Με αυτόν τον τρόπο επιτρέπει στους επιτιθέμενους να αλλάξουν τη ροή του προγράμματος και να εκτελέσουν απομακρυσμένες εντολές ή τα προγράμματά τους. Το buffer overflow είναι μια ευρέως διαδεδομένη και πολύ επικίνδυνη ευπάθεια. Εμφανίζεται σε πολλά λειτουργικά συστήματα και λογισμικά εφαρμογών. Είναι μια διάσημη επίθεση που χρησιμοποιείται στα Penetration Testing.

### 2.7.5 Configuration Error Attack

Αυτός ο τύπος επίθεσης βασίζεται επίσης στο ανθρώπινο καθώς δεν γίνετε σωστή διαμόρφωση των συστημάτων στόχων από τον διαχειριστή. Για παράδειγμα, αν ο κατάλογος επιτρέπει στους χρήστες να ανεβάζουν αρχεία, ή ακόμα δίνει δικαίωμα εκτέλεσης αρχείων, τότε οι εισβολείς μπορούν να ανεβάσουν και να εκτελέσουν ένα κακόβουλο αρχείο.

### 2.7.6 Web Application Attack

Η διαδικτυακή επίθεση είναι μια επίθεση εναντίον διαδικτυακών εφαρμογών. Σύμφωνα με το OWASP [27] το οποίο κάθε χρόνο δημοσιεύει τον κατάλογο με τα 10 πιο τρωτά σημεία για να αυξήσει την ευαισθητοποίηση μεταξύ των προγραμματιστών και των διαχειριστών. Για το έτος 2021 τα τρία πιο συνηθισμένα τρωτά σημεία που εντοπίστηκαν είναι το Broken Access Control, οι Cryptographic Failures και το Injection - Cross Site Scripting (XSS).

### 2.7.7 Sniffer Attack

Εάν ο στόχος δεν έχει γνωστά τρωτά σημεία, ένας έμπειρος επιτιθέμενός συνήθως εκτελεί μια sniffer Attack. Πρώτα εισβάλλουν σε άλλα συστήματα στο ίδιο υποδίκτυο με τον αρχικό στόχο, μετά παρακολουθούν και αναλύουν όλη τη ροή του δικτύου για να αποκτήσουν ευαίσθητες πληροφορίες, όπως έναν κωδικό πρόσβασης.

### 2.7.8 Information Gathering

Όπως έχει αναλυθεί στην υπό ενότητα 2.5.2 η συλλογή πληροφοριών είναι το πιο κρίσιμο βήμα στο Penetration Testing. Συνήθως, οι πληροφορίες στόχου που συλλέγονται περιλαμβάνουν διεύθυνση IP, ανοιχτές θύρες, εφαρμογές σε χρήση, τύπο λειτουργικού συστήματος, πληροφορίες που αφορούν το ανθρώπινο δυναμικό ή τον οργανισμό, την τοπολογία του δικτύου, τους αμυντικούς μηχανισμούς εάν είναι διαθέσιμοι, την διαμόρφωση, τις ευπάθειες και εν τέλη το φυσικό περιβάλλον. Η συλλογή των παραπάνω πληροφοριών καθορίζει εάν το Penetration Testing θα είναι επιτυχημένο ή όχι.

## 2.8 Hardware που Χρησιμοποιείται από τους Red Teamers

Μερικά από τα τυποποιημένα εργαλεία που χρησιμοποιούνται στον τομέα του Red Teaming, από την εταιρεία Hak5 [28]. Κύρια προϋπόθεση για τα πλείστα είναι η χρήση κοινωνικής μηχανικής, αφού χρειάζεται η εφαρμογή τους σε φυσικά μηχανήματα.


*Εικόνα 5 (Pineapple Express)*


*Εικόνα 4 (Usb Rubber Ducky)*


*Εικόνα 3 (Lan Turtle)*

**BASH BUNNY**

A quad-core Linux-box-on-USB-stick mimicking multiple trusted devices to deploy advanced pentest and IT automation payloads.

*Εικόνα 8 (Bash Bunny)*

**SHARK JACK**

Jack into a network and instantly run advanced recon, exfiltration, attack and automation payloads.

*Εικόνα 7 (Shark Jack)*

**KEY CROC**

A keylogger armed with pentest tools, remote access and payloads that trigger multi-vector attacks when chosen keywords are typed.

*Εικόνα 6 (Key Croc)*

## 2.9 Εργαλεία Red Teaming

### 2.9.1 Nmap (Information Gathering/Vuln Scanner)

Το Nmap [29] είναι ένα εργαλείο ανοιχτού κώδικα και είναι ο πιο γνωστός και πιο επαγγελματικός σαρωτής ασφαλείας και μπορεί να χρησιμοποιηθεί για την ανακάλυψη θυρών, κεντρικών υπολογιστών και υπηρεσιών σε ένα δίκτυο. Γράφτηκε σε C/C++ και Python από τον Gordon Lyon ξεκινώντας το 1997. Για να ανακαλύψει κεντρικούς υπολογιστές σε ένα δίκτυο. Το Nmap στέλνει ειδικά διαμορφωμένα πακέτα στον κεντρικό υπολογιστή-στόχο και στη συνέχεια αναλύει τις απαντήσεις. Χρησιμοποιώντας τα IP πακέτα δύναται να βρει υπολογιστές διαθέσιμους στο δίκτυο, ποια λειτουργικά συστήματα χρησιμοποιούν, ποιο είδος firewall, αλλά και πολλά άλλα. Έχει σχεδιαστεί για γρήγορη σάρωση μεγάλων δικτύων αλλά λειτουργεί και να μεμονωμένους υπολογιστές. Λειτουργεί σε όλα τα μεγάλα λειτουργικά συστήματα Linux, Windows και MAC και θεωρείται ως ένα από τα ουσιαστικότερα εργαλεία στο στάδιο της συλλογής πληροφοριών.

### 2.9.2 Metasploit Framework (Exploitation)

Το πιο χρησιμοποιημένο penetration testing framework στον κόσμο. Βοηθά τις Red και Blue Teams για να εκπληρώσουν τους στόχους τους. Παρέχει εργαλεία προς

εκμετάλλευση έναντι απομακρυσμένων στόχων και περιέχει εκατοντάδες επαγγελματικά εργαλεία εκμετάλλευσης για γνωστά τρωτά σημεία λογισμικού. Το Metasploit [30] όχι μόνο συλλέγει exploits αλλά επιτρέπει στους χρήστες να αναπτύξουν exploit στο περιβάλλον τους.

### 2.9.3 Nessus και OpenVAS (Vulnerability Scanners)

Ο σαρωτής ευπάθειας είναι ένα πρόγραμμα που εντοπίζει και ανακαλύπτει αυτόματα τρωτά σημεία ασφαλείας σε υπολογιστές, συστήματα πληροφοριών, δίκτυα και

εφαρμογές. Εντοπίζει τα τρωτά σημεία στέλνοντας συγκεκριμένα πακέτα στον στόχο και στη συνέχεια αναλύοντας τις απαντήσεις ώστε να ταιριάζουν με τη βάση δεδομένων ευπάθειας. Ο Nessus [31] είναι ο πιο διάσημος σαρωτής ευπάθειας στον κόσμο, που χρησιμοποιείται από περισσότερους από 75.000 οργανισμούς παγκοσμίως. Το εν λόγω εργαλείο παρέχει μια πλήρη λειτουργία σάρωσης ευπαθειών η οποία ενημερώνεται συνεχώς. Δεν είναι δωρεάν πλην όμως η έκδοση essentials διατίθεται δωρεάν για scanning μέχρι και 16 Ips. Παρόμοια με το Nessus, το OpenVAS [32] είναι ανοιχτού κώδικα και είναι ένας από τους πιο δημοφιλείς σαρωτές ευπάθειας. Στο στάδιο της συλλογής πληροφοριών, ένας σαρωτής ευπάθειας είναι ο καλύτερος τρόπος για να ανακάλυψη γνωστών τρωτών σημείων στο σύστημα στόχος.

### 2.9.4 Hydra και John the Ripper (Password Attack Tools)

Το Hydra είναι ένα ισχυρό διαδικτυακό εργαλείο επίθεσης κωδικού πρόσβασης που μπορεί να υποστηρίξει τα περισσότερα πρωτόκολλα ή εφαρμογές, όπως FTP, HTTP, HTTPS, MySQL, MSSQL, Oracle, Cisco, IMAP και VNC. Το John the Ripper είναι ένα διάσημο εργαλείο επίθεσης κωδικού πρόσβασης στο σύστημα Linux. Το ποσοστό επιτυχίας της διάρρηξης κωδικού πρόσβασης σχετίζεται με το λεξικό το οποίο χρησιμοποιείται (συνήθως γίνεται χρήση της rockyou.txt λίστας κωδικών).

### 2.9.5 Sqlmap (Injection)

Το Sqlmap [33] είναι ένα άλλο εργαλείο διαδικτυακής επίθεσης που αυτοματοποιεί τη διαδικασία ανίχνευσης και εκμετάλλευσης SQL Injection. Είναι ισχυρό και αυτοματοποιεί τις ακόλουθες λειτουργίες: (I) Αναγνώριση βάσης δεδομένων, (II) λήψη δεδομένων από τη βάση δεδομένων, (III) πρόσβαση στο υποκείμενο σύστημα αρχείων και (IV) εκτέλεση εντολών στη λειτουργία συστήματος.

### 2.9.6 Ettercap (Man In the Middle)

Μια επίθεση MITM εκτελείται μέσω επιθέσεων παραποίησης δεδομένων και sniffing μέσω υποκλοπής δεδομένων επικοινωνίας σε ένα δίκτυο-στόχο το οποίο είναι ιδιαίτερα δύσκολο να εντοπιστούν. Το Ettercap [34] είναι μια ολοκληρωμένη σουίτα για επιθέσεις MITM, η οποία μπορεί να χρησιμοποιηθεί για ανάλυση πρωτοκόλλου δικτύου υπολογιστών και έλεγχο ασφαλείας. Διαθέτει, μεταξύ άλλων στοιχείων, sniffing ζωντανών συνδέσεων και φιλτράρισμα.

### 2.9.7 Social Engineering Tool (SET)

Η επίθεση κοινωνικής μηχανικής [22] είναι ένας φορέας επίθεσης που βασίζεται σε μεγάλο βαθμό στην ανθρώπινη αλληλεπίδραση και συχνά περιλαμβάνει χειραγώγηση ανθρώπων ώστε να παραβιάζουν τις τυπικές διαδικασίες ασφάλειας και τις βέλτιστες πρακτικές για να αποκτήσουν πρόσβαση σε συστήματα, δίκτυα ή φυσικές τοποθεσίες ή για οικονομικό όφελος. Όταν οι στόχοι προστατεύονται καλά, οι επιθέσεις κοινωνικής μηχανικής είναι συχνά το κλειδί της επιτυχίας για τους επιτιθέμενους. Το SET [35] είναι το πιο γνωστό εργαλείο κοινωνικής μηχανικής και μπορεί να εκτελέσει 11 είδη επιθέσεων κοινωνικής μηχανικής.

### 2.9.8 Recon - NG (Information Gathering)

Το Recon-ng είναι ένα δωρεάν και ανοιχτού κώδικα εργαλείο διαθέσιμο στο GitHub. Το Recon-ng [36] βασίζεται στο Open Source Intelligence (OSINT) και θεωρείται το πιο εύκολο και χρήσιμο εργαλείο για αναγνώριση. Αυτό το εργαλείο μπορεί να χρησιμοποιηθεί για τη λήψη πληροφοριών σχετικά με τον στόχο. Η διαδραστική κονσόλα παρέχει μια σειρά από χρήσιμες λειτουργίες. Το Recon-ng είναι ένα εργαλείο Web Reconnaissance γραμμένο σε Python. Έχει πολλές ενότητες, αλληλεπίδρασης με βάση

δεδομένων, ενσωματωμένες λειτουργίες για ευκολία, διαδραστική βοήθεια και ολοκλήρωση εντολών.

## 2.10 Αυτοματισμοί

Ο εντοπισμός μιας λειτουργικής διαδρομής επίθεσης (Cyber Kill Chain) μπορεί να είναι χρονοβόρα για τους Red Teamers και έτσι οι τεχνικές αυτοματοποιημένου σχεδιασμού θεωρούνται ως μια εφικτή μέθοδος ανακάλυψης πιθανών μονοπατιών επίθεσης για την αυτοματοποίηση των παραπάνω ενεργειών.

### 2.10.1 Επίπεδα Αυτοματισμών

Τα αυτοματοποιημένα συστήματα διακρίνονται σε τέσσερα διαφορετικά επίπεδα, τα οποία αναλόγως της γνωσιακής βάσης του συστήματος ως προς τη συλλεγόμενη και γενικευμένη τεχνογνωσία διαχωρίζονται όπως παρακάτω [37]:

#### 2.10.1.1 Πλήρως Αυτόνομο

Το σύστημα έχει εξ ολοκλήρου τον έλεγχο των δοκιμών, ώστε να μπορεί να εκτελεί εργασίες με τον ίδιο τρόπο που θα κάνει ο ανθρώπινος ειδικός, χωρίς να χρειάζεται οποιαδήποτε καθοδήγηση ή κάποιον για να επιλέγει τις αποφάσεις οι οποίες θα εκτελεστούν.

#### 2.10.1.2 Μερικώς Αυτόνομο

Το σύστημα είναι ημιαυτόνομο στην εκτέλεση εργασιών. Σε αυτή την περίπτωση, το σύστημα βρίσκεται υπό συνεχή επίβλεψη από ειδικούς, οι οποίοι αποφασίζουν για τις επιμέρους ενέργειες. Το σύστημα από μόνο του μπορεί να προχωρήσει μερικώς σε ενέργειες αφού πρώτα υπάρχει η καθοδήγηση για το τι επιθυμεί ο ειδικός να εκτελέσει.

#### 2.10.1.3 Βοηθός Λήψης Αποφάσεων

Το σύστημα ενεργεί μαζί με τον ανθρώπινο ειδικό και τον βοηθά στη λήψη αποφάσεων, προτείνοντας του τις ενέργειες, αναλόγως των υποθέσεων για τα οποία υπάρχει κοινό

ιστορικό, καθώς επίσης αποτρέπει τις άσκοπες επαναλήψεις σε περίπτωση όπου υποπίπτει ο εκάστοτε ειδικός.

#### 2.10.1.4 Λειτουργία Εκμάθησης

Το σύστημα λειτουργεί στο παρασκήνιο και μαθαίνει κατά τη διάρκεια του Penetration Testing, αναλόγως των ενεργειών που λαμβάνονται από τους εκτελών τη διείσδυση.

### 2.10.2 Αυτοματισμοί Βασισμένοι στον Κανόνα Rule Tree

Ο σκοπός του σχεδιασμού ενός δέντρου κανόνων είναι να ακολουθήσει προκαθορισμένη διαδρομή επίθεσης κατά το Penetration Testing. Το δέντρο κανόνων είναι η βάση για την επίτευξη αυτοματοποιημένης δοκιμής διείσδυσης. Σύμφωνα με τους θεματοθέτες ο παραπάνω κανόνας είναι ουσιαστικά μια μαθηματικά δομημένη έννοια η οποία μπορεί να προσδιοριστεί με όρους και να ενεργοποιηθεί με τη χρήση ψευδοκώδικα. Κατασκεύασαν λοιπόν μια αυτοματοποιημένη πρόοδο δοκιμών διείσδυσης χρησιμοποιώντας τη μέθοδο κανόνων δέντρου, η οποία διαδικασία εξελίσσεται και γίνετε αποδοτικότερη με την συνεχή προσθήκη καινούργιων δέντρων κανόνων [38].

Τα διαγράμματα των δέντρων κανόνων επιτρέπουν στους χρήστες να οπτικοποιήσουν τη λογική ενός μεγάλου συστήματος κανόνων. Οι κανόνες ορίζουν τις προϋποθέσεις (διαδικαστικές και ουσιαστικές) υπό τις οποίες δικαιολογούνται συγκεκριμένη δράσης. Κάθε κόμβος ενός δέντρου κανόνων είναι μια πρόταση, ικανή να έχει οποιαδήποτε από τις τρεις τιμές αλήθειας ("αληθής" / "αναποφάσιστος" / "ψευδής") [39]. Ο κορυφαίος κόμβος ενός δέντρου αντιπροσωπεύει το απόλυτο ζήτημα που πρέπει να επιτευχθεί, ο «ριζικός κόμβος» που αντιπροσωπεύει τον στόχο της επίθεσης. Οι κόμβοι φύλλων (Leaf Nodes) αντιπροσωπεύουν διαφορετικούς τρόπους για την επίτευξη του στόχου και οι θυγατρικοί κόμβοι αντιπροσωπεύουν βήματα επίθεσης.

Εν συνεχεία γίνεται συσχετισμός λογικού AND ή λογικού OR με κάθε κόμβο. Ένας κόμβος OR μπορεί να προκύψει εάν συμβεί κάποιο από τα θυγατρικά συμβάντα του. για να εμφανιστεί ένας κόμβος AND, απαιτούνται όλα τα θυγατρικά συμβάντα του. Οι κόμβοι

μπορούν να αυξηθούν με πολλές διαφορετικές τιμές, όπως πιθανότητες ή κόστος, έτσι ώστε να μπορούμε να υπολογιστεί η επικρατέστερη διαδρομή επίθεσης [40].

### 2.10.3 Αυτοματισμοί Βασισμένοι στον Κανόνα BDI (Belief, Desire, Intention)

Το μοντέλο BDI περιγράφει τη διαδικασία του τρόπου με τον οποίο η ομάδα ελέγχου είναι σε θέση να επιλέξει ενέργειες σε σχέση με τις πληροφορίες στόχου κατά τη διάρκεια της δοκιμής του Penetration Testing. Η αρχιτεκτονική του πράκτορα BDI βασίζεται στη φιλοσοφική θεωρία του Michael Bratman (Bratman 1987) που εξηγεί τη λογική μέσω των ακόλουθων στάσεων: πεποιθήσεις, επιθυμίες και προθέσεις. Κατά συνέπεια, ένας πράκτορας BDI είναι ένας αυτόνομος πράκτορας που χρησιμοποιεί αυτές τις τρεις έννοιες για τη λειτουργία του, που χρησιμοποιείται κυρίως σε εφαρμογές όπου εφαρμόζεται Artificial Intelligence (AI) ή και Machine Learning.

Οι πεποιθήσεις είναι το μοντέλο που ακολουθεί ο πράκτορας σε σχέση με το περιβάλλον του, βασικά αυτό που πιστεύει ότι είναι αληθινό, το οποίο όμως δεν μπορεί να θεωρηθεί ως δεδομένο, καθώς ορισμένες από τις πεποιθήσεις του πιθανόν να είναι και ψευδείς.

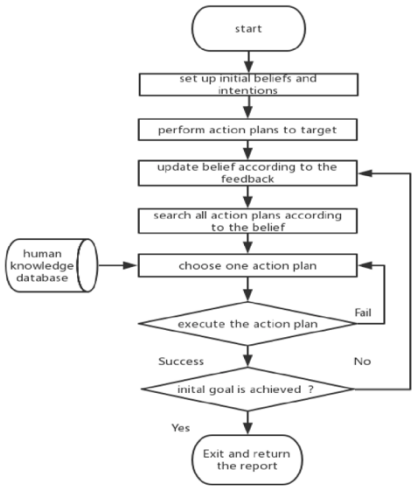
Οι επιθυμίες αντιπροσωπεύουν την ιδανική κατάσταση του περιβάλλοντος για τον πράκτορα. Όπως και στο ανθρώπινο μυαλό, αυτά αντιπροσωπεύουν πράγματα που θα θέλαμε να δούμε να πραγματοποιούνται μελλοντικά. Μια επιθυμία μπορεί να είναι ρεαλιστική ή όχι, όπως συμβαίνει με την ανθρώπινη σκέψη, και μπορεί να είναι ή να μην είναι εφικτή. Οι επιθυμίες μπορεί να είναι αμοιβαία περιεκτικές ή αποκλειστικές.

Οι προθέσεις αντιπροσωπεύουν ένα υποσύνολο επιθυμιών που ο πράκτορας έχει λάβει ως στόχους που πρέπει να επιτευχθούν σύντομα. Αυτές οι προθέσεις δεν μπορούν να είναι αντίθετες με τις πεποιθήσεις.

Το μοντέλο ακολουθεί τις συμβάσεις που υιοθετήθηκαν στο Jason Interpreter, ένα μοντέλο που βασίζεται στο Procedural Reasoning System (PRS) [41]. Ο παράγοντας BDI ορίζεται ως μια πλειάδα <Ag, B, D, I, P, A, S> [10], όπου το Ag είναι το όνομα πράκτορα., το B είναι ένα σύνολο πεποιθήσεων το οποίο αντιπροσωπεύει τις πληροφορίες σχετικά με τον στόχο, το οποίο ενημερώνεται αμέσως μετά την εκτέλεση ενεργειών, το D είναι

ένα σύνολο επιθυμιών, το οποίο αντιπροσωπεύει όλες τις επιλογές ή πιθανά υποψήφια σχέδια δοκιμής διείσδυσης για το μέσο, το I είναι ένα σύνολο προθέσεων, το οποίο αντιπροσωπεύει τους στόχους, ή ποιο σχέδιο έχει αποφασίζει να πραγματοποιήσει ο πράκτορας, το P είναι ένα σύνολο σχεδίων, το οποίο αποτελείται από διαθέσιμα σχέδια, καθένα από τα οποία παρέχει πληροφορίες σχετικά με τον τρόπο επίτευξης των στόχων, το A είναι ένα σύνολο ενεργειών, το οποίο περιλαμβάνει τις ενέργειες του πράκτορα μπορεί να εκτελέσει, το S είναι ένα σύνολο αντίληψης, το οποίο αποθηκεύει τις πληροφορίες από το περιβάλλον.

Ο συλλογιστικός κύκλος του παράγοντα BDI φαίνεται στην παρακάτω Εικόνα 9.




*Εικόνα 9 (Συλλογιστικός κύκλος BDI)* [10]

## 2.10.4 Αυτοματισμοί Βασισμένοι σε Attack Graphs

Τα γραφήματα επίθεσης προτάθηκαν για πρώτη φορά από τον Swiler το 1998. Είναι ένας τύπος κατευθυνόμενου γραφήματος που μπορεί να περιγράψει όλα τα μονοπάτια που μπορεί να ακολουθήσει ένας επιτιθέμενος για να φτάσει στον στόχο από το αρχικό σημείο της επίθεσης. Οι κόμβοι σε ένα γράφημα επίθεσης αντιπροσωπεύουν την κατάσταση της επίθεσης, συγκεκριμένα μηχανές στόχους στις οποίες έχει αποκτήσει πρόσβαση ο επιτιθέμενος και τα δικαιώματα χρήστη που έχει παραβιάσει η επίθεση. Τα τόξα αντιπροσωπεύουν μια αλλαγή κατάστασης που προκαλείται από μια μεμονωμένη ενέργεια του εισβολέα. Με την ανάθεση πιθανοτήτων επιτυχίας της επίθεσης στα τόξα, διάφοροι αλγόριθμοι γραφημάτων, όπως αλγόριθμοι συντομότερης διαδρομής, μπορούν να προσδιορίσουν μονοπάτια επίθεσης με την υψηλότερη πιθανότητα επιτυχίας.

Το γράφημα επίθεσης μπορεί να δημιουργηθεί από τρία στοιχεία πρότυπα επίθεσης [42], ένα αρχείο διαμόρφωσης και ένα προφίλ εισβολέα. Τα πρότυπα επίθεσης αποτελούνται από τις πληροφορίες ή τις συνθήκες που πρέπει να υπάρχουν για την επίθεση, όπως η έκδοση του λειτουργικού συστήματος ή η ανοιχτή θύρα. Το αρχείο διαμόρφωσης παρέχει πληροφορίες σχετικά με τα συστήματα προορισμού, συμπεριλαμβανομένης της τοπολογίας του δικτύου, των πληροφοριών διαμόρφωσης των σταθμών εργασίας, των εκτυπωτών ή των δρομολογητών. Το προφίλ του εισβολέα παρέχει πληροφορίες σχετικά με τις δυνατότητες του εισβολέα, όπως ενέργειες επίθεσης. Το γράφημα επίθεσης όχι μόνο περιγράφει πολλούς επιτιθέμενους και πολλούς στόχους, αλλά υποστηρίζει επίσης τη λογική.

Υπάρχουν δύο τύποι γραφημάτων επίθεσης [43]. Αυτά είναι τα state-based και τα attribute based. Στα γραφήματα επίθεσης που βασίζονται σε state-based, κάθε κόμβος αντιπροσωπεύει καταστάσεις δικτύου ή καταστάσεις επίθεσης, όπως έκδοση λειτουργικού συστήματος, ανοιχτές θύρες, υπηρεσίες, ευπάθειες και δικαιώματα χρήστη. Τα τόξα αντιπροσωπεύουν τη διαδρομή μετάβασης από τη μια κατάσταση στην άλλη. Το γράφημα επίθεσης κατάστασης μπορεί να εμφανίσει όλες τις πιθανές διαδρομές επίθεσης από την αρχική κατάσταση σε μια κατάσταση στόχου. Ωστόσο, ο αριθμός των μονοπατιών επίθεσης αυξάνεται εκθετικά ανάλογα με την κλίμακα και τον αριθμό των τρωτών σημείων στόχου, με αποτέλεσμα ο συγκεκριμένος τύπος γραφήματος επίθεσης να μην είναι κατάλληλος για δίκτυα μεγάλης κλίμακας. Αντίθετα, τα γραφήματα επίθεσης που βασίζονται σε attribute based, έχουν καλύτερη επεκτασιμότητα σε δίκτυα μεγάλης κλίμακας.

Σύμφωνα με την επιστημονική κοινότητα [43] [44], έχουν γίνει άλματα στον τομέα με τη δημιουργία αλγορίθμων για την αυτοματοποίηση των ενεργειών, καθώς όμως υπάρχουν είδη και τα αυτοματοποιημένα εργαλεία όπως το NetSPA, το οποίο δημιουργεί γραφήματα επίθεσης με βάση τις πληροφορίες εισόδου σε συνδυασμό με χρήση αλγορίθμων. Το εν λόγω εργαλείο σε ένα ρεαλιστικό δίκτυο με 17 υπολογιστές, χρειάστηκε λιγότερο από 90 δευτερόλεπτα για να δημιουργήσει τρία γραφήματα επίθεσης. Ο χρόνος της αυτόματης υλοποίησης σε σχέση με μια χειροκίνητη προσπάθεια είναι πρακτικά άπιαστος.

### 2.10.5 Αυτοματισμοί Βασισμένοι σε Planning Domain Definition Language (PDDL)

Η PDDL είναι μια επίσημη γλώσσα αναπαράστασης γνώσης που έχει σχεδιαστεί για να εκφράζει και να αναπαριστά διάφορα μοντέλα σχεδιασμού. Χρησιμοποιείται συνήθως για την κωδικοποίηση της γνώσης τομέα καθώς και για τη διαχείριση πολλαπλών προβλημάτων σε πολλούς τομείς χρησιμοποιώντας διαφορετικά υποσύνολα γλωσσικών χαρακτηριστικών για την υποστήριξη της κοινής χρήσης τομέων σε διαφορετικούς σχεδιαστές που χειρίζονται διαφορετικά επίπεδα εκφραστικότητας [45]. σε ένα αρχείο τομέα PDDL και σε ένα αρχείο προβλήματος. Κάθε περιγραφή προβλήματος PDDL περιλαμβάνει ένα σύνολο από αντικείμενα, μια αρχική συνθήκη και μια περιγραφή στόχου – αντικειμενικού σκοπού.

Με την υποβοήθηση από κλασικούς και ευρετικούς σχεδιαστές (heuristic planners), όπως το Metric-FF [46], που χρησιμοποιούν τεχνικές επίλυσης προβλημάτων, δημιουργούν σχέδια επίθεσης. Ένας σχεδιαστής (planner) ξεκινά την εκτέλεσή του από την αρχική κατάσταση με μια αναπαράσταση βασισμένη σε γράφημα που ονομάζεται plangraph. Το plangraph δημιουργείται ξεκινώντας από την αρχική κατάσταση. Η διαδοχική εφαρμογή των τελεστών μετάβασης κατάστασης σε όλες τις περιπτώσεις χρησιμοποιείται στη συνέχεια για να χαρτογραφήσει καταστάσεις και στόχους και να τους μετατρέψει σε μελλοντικές ενέργειες [47].

Πολλές μελέτες [48] [49] χρησιμοποιούν το PDDL για να εκφράσουν τη δράση που απαιτείται για τον σχεδιασμό επίθεσης και να μοντελοποιήσουν το πρόβλημα του PT, όπως χρησιμοποιώντας την εν λόγω γλώσσα, για να περιγράψει δίκτυα υπολογιστών, των τρωτών σημείων και exploit τους, για τη δημιουργία μονοπατιών επίθεσης που ενσωματώθηκαν σε ένα εργαλείο δοκιμής διείσδυσης. Αυτή η προσέγγιση σχεδιασμού χρησιμοποιήθηκε επίσης και για την πρόβλεψη των ενεργειών των επιτιθέμενων.

### 2.10.6 Αυτοματισμοί Βασισμένοι σε Partially Observable Markov Decision Processes (POMDP)

Το POMDP [50] είναι ένα μοντέλο λήψης αποφάσεων σε συνθήκες αβεβαιότητας, το οποίο έχει στόχο να ανακαλύψει τη βέλτιστη πολιτική και να αντιστοιχίσει καταστάσεις

με ενέργειες. Η βέλτιστη πολιτική δίνει τις καλύτερες ενέργειες σε κάθε κατάσταση με βάση τις παρατηρήσεις της και μεγιστοποιεί το μελλοντικό της κέρδος (συνολική ανταμοιβή). Ο εκάστοτε agent αλληλεπιδρά με ένα σύστημα με ένα αβέβαιο δυναμικό περιβάλλον και του οποίου η τρέχουσα κατάσταση είναι άγνωστη. Η επιλογή και το αποτέλεσμα των ενεργειών είναι επίσης αβέβαιες. Δεδομένου ότι τα Penetration Testing ενεργούν υπό αβέβαια σενάρια, το POMDP είναι ένα φυσικά επακόλουθο υποψήφιο μοντέλο για υλοποίηση.

Παρά όλα τα πλεονεκτήματά του, το POMDP έχει δύο βασικούς περιορισμούς. Πρώτον, η επεκτασιμότητα του είναι ένα σημαντικό ζήτημα. Δεύτερον, είναι δύσκολο να σχεδιαστεί η αρχική πεποίθηση για κάθε πραγματικό πρόβλημα καθώς επίσης και η ακριβής κατανομή πιθανοτήτων. Στο σενάριο του Penetration Testing, δεν είναι σαφές πώς οι agents μπορούν να αποκτήσουν αυτές τις κατανομές. Επιπλέον, τα μοντέλα POMDP είναι πολύπλοκα και απαιτούν ακριβούς υπολογιστικούς πόρους.

## 2.11 Cyber Ranges

Τα Cyber Ranges είναι ένας σχετικά νέος τρόπος για την ανάλυση των επιθέσεων στον κυβερνοχώρο και ειδικότερα, για τη διδασκαλία της ασφάλειας στον κυβερνοχώρο. Το Εθνικό Ινστιτούτο Προτύπων και Τεχνολογίας (NIST) ορίζει τα Cyber Ranges ως [51]:
«Τα Cyber Ranges είναι διαδραστικές, προσομοιωμένες αναπαραστάσεις του τοπικού δικτύου, του συστήματος, των εργαλείων και των εφαρμογών ενός οργανισμού που είναι συνδεδεμένα σε ένα προσομοιωμένο περιβάλλον σε επίπεδο Διαδικτύου. Παρέχει ένα ασφαλές, νομικό περιβάλλον για να αποκτηθούν πρακτικές δεξιότητες στην ασφάλεια του κυβερνοχώρου και ένα ασφαλές περιβάλλον για την ανάπτυξη προϊόντων και δοκιμών ασφαλείας.»

Τα Cyber Ranges μπορεί να περιλαμβάνουν πραγματικό υλικό και λογισμικό ή μπορεί να είναι ένας συνδυασμός πραγματικών και εικονικών στοιχείων και μπορεί να είναι διαλειτουργικά σε συνδυασμό με άλλα παρόμοια περιβάλλοντα. Τα Cyber Range σε επίπεδο Διαδικτύου προσομοιώνουν όχι μόνο τα πακέτα κίνησης, (traffic) αλλά επίσης μπορεί να αναπαράγει υπηρεσίες δικτύου όπως ιστοσελίδες, προγράμματα περιήγησης και email.Ως εκ τούτου, τα Cyber Ranges είναι οποιοδήποτε λογισμικό ή υλικό που διαθέτει αυτές τις δυνατότητες. Για αυτόν τον λόγο, υπάρχει διαθέσιμος ένας μεγάλος

αριθμός διαφορετικών σειρών κυβερνοχώρου, από scripts, frameworks έως ξεχωριστά λειτουργικά συστήματα. Γενικά, όσο πιο ξεχωριστό είναι το περιβάλλον του Cyber Range από το λειτουργικό σύστημα, τόσο περισσότερες επιλογές προσφέρει [52].

Με την χρήση των Cyber Ranges επιτυγχάνεται η εκπαίδευση και η αξιολόγηση σύμφωνα με την απόδοση καθώς παρέχετε ένα περιβάλλον προσομοίωσης όπου ομάδες μπορούν να συνεργαστούν για να βελτιώσουν την ομαδική εργασία και τις δυνατότητες της ομάδας, με την παροχή σχολίων σε πραγματικό χρόνο, καθώς μπορούν να δοκιμαστούν νέες ιδέες ή να εκτελεστεί εργασία για την επίλυση περίπλοκων προβλημάτων, η οποία θα αποδώσει πολύτιμη εμπειρία στην κοινότητα.[51]

Η δημιουργία ενός Cyber Range δεν είναι εύκολη υπόθεση. Απαιτεί πολύ χρόνο και γι' αυτό πολλά από αυτά τα τρέχοντα projects δεν έχουν αναπτυχθεί εδώ και πολύ καιρό. Αυτό στη συνέχεια οδηγεί σε ξεπερασμένο και επακόλουθη ασυμβατότητα με το τρέχον υλικό ή λογισμικό. Στην περίπτωση της πλατφόρμας ανοιχτού κώδικα, η κοινότητα είναι επίσης σημαντική, καθώς βοηθά στην υποστήριξη. Η ανάπτυξη σεναρίων είναι επίσης χρονοβόρα, επομένως ακόμη και εδώ είναι απαραίτητο να υπάρχει κοινοτική παρουσία καθώς μοιράζονται μεμονωμένα σενάρια μεταξύ αυτών. Ως εκ τούτου, κατά την επιλογή ενός Cyber Range ανοιχτού κώδικα, πρέπει να δίνεται έμφαση: στην επικαιρότητα, την υποστήριξη και τα διαθέσιμα σενάρια.[52]

Μια ποικιλία ασκήσεων έχει εμφανιστεί τα τελευταία 15 χρόνια. Οι ασκήσεις για την ασφάλεια στον κυβερνοχώρο στοχεύουν σε διαφορετικούς στόχους (π.χ. να δημιουργήσουν ικανότητες, να αξιολογήσουν ικανότητες ή απλώς να διασκεδάσουν τους εκπαιδευομένους). Οι ασκήσεις ασφάλειας στον κυβερνοχώρο μπορούν να δομηθούν και να σχεδιαστούν με διάφορους τρόπους. Για παράδειγμα, οι ασκήσεις Capture-The-Flag (π.χ. iCTF, DEFCON CTF, NYU-CSAW) έχουν σχεδιαστεί έτσι ώστε οι συμμετέχοντες (ομάδες ή άτομα) να βρούνε και να «πιάσουν» μια συγκεκριμένη σημαία (π.χ. ένα αρχείο ή κείμενο). Τα CTF συχνά χρησιμοποιούν ειδικά σχεδιασμένες πλατφόρμες. Άλλα παραδείγματα είναι οι ασκήσεις ασφάλειας στον κυβερνοχώρο ή οι ασκήσεις άμυνας στον κυβερνοχώρο (CDX) που συχνά φιλοξενούνται σε Cyber Range περιβάλλοντα.[4]

## 2.12 Υπάρχοντα Αυτοματοποιημένα Συστήματα

### 2.12.1 CALDERA

Το CALDERA™ [53] είναι ένα πλαίσιο κυβερνοασφάλειας που έχει σχεδιαστεί για να εκτελεί εύκολα αυτόνομες ασκήσεις παραβίασης και προσομοίωσης. Μπορεί επίσης να χρησιμοποιηθεί για την εκτέλεση ενεργειών Red Teaming ή αυτοματοποιημένης απόκρισης περιστατικών. Το CALDERA βασίζεται στο πλαίσιο MITER ATT&CK™ [54] και είναι ένα ενεργό ερευνητικό έργο στη MITRE και αποτελείται από δύο βασικά χαρακτηριστικά:

1. Το βασικό σύστημα, όπου αφορά τον κώδικα του πλαισίου, που περιλαμβάνει έναν ασύγχρονο διακομιστή εντολών και ελέγχου (C2) με REST API και διεπαφή ιστού.

2. Πρόσθετα (Plugins). Αυτά είναι ξεχωριστά αποθετήρια που εξαρτώνται από το βασικό σύστημα, παρέχοντας πρόσθετη λειτουργικότητα. Παραδείγματα περιλαμβάνουν οι Agents, διεπαφές GUI, συλλογές TTP και άλλα.

Πρόκειται για ένα λογισμικό το οποίο χρησιμοποιεί ένα σύστημα «Client – Server», όπου ο διακομιστής χρησιμοποιείται για τη ρύθμιση Agents (πελατών) και την εκκίνηση των επιθυμητών λειτουργιών ελέγχου. Οι Agents ουσιαστικά είναι προγράμματα λογισμικού που συνδέονται με το CALDERA σε συγκεκριμένα χρονικά διαστήματα και λαμβάνουν οδηγίες. Οι Agents επικοινωνούν με τον διακομιστή CALDERA μέσω μιας μεθόδου επαφής η οποία ορίζεται κατά την αρχική εγκατάσταση.

Αρχικά τοποθετούνται σε ομάδες (groups) για τον προσδιορισμό, στις οποίες πρόκειται να εκτελεστούν οι παραπάνω λειτουργίες. Επίσης κατά τον καταμερισμό σε groups γίνεται και ο αρχικός διαχωρισμός των agents σε Red Teams και Blue Teams και των ανάλογων λειτουργιών που θα υπάρχουν διαθέσιμα στη φαρέτρα της κάθε ομάδας.

Abilities (Ικανότητες) είναι μια συγκεκριμένες εφαρμογές τακτικής/τεχνικής ATT&CK που μπορεί να εκτελεστεί στους Agents. Οι ικανότητες περιλαμβάνουν τις εντολές προς

εκτέλεση, τις πλατφόρμες / εκτελεστές στις οποίες μπορούν να εκτελεστούν οι εντολές (π.χ. Windows / PowerShell), Payloads που θα συμπεριληφθούν και στην αναφορά και ανάλυση της εξόδου στον διακομιστή CALDERA.

Τα Adversary profiles (προφίλ αντιπάλου) είναι ομάδες ικανοτήτων, που αντιπροσωπεύουν τις τακτικές, τις τεχνικές και τις διαδικασίες (T-Tactics/T-Techniques/P-Procedures) που είναι διαθέσιμες για μια απειλή. Τα προφίλ αντιπάλου χρησιμοποιούνται κατά την εκτέλεση μιας λειτουργίας για να καθοριστεί ποιεά από τα παραπάνω Abilities θα εκτελεστούν.

Τα Operations αφορούν στην εκτέλεση των λειτουργιών σε group των agents. Τα adversary profiles χρησιμοποιούνται για να καθοριστούν ποια abilities θα εκτελεστούν και τα group agents χρησιμοποιούνται για να καθοριστεί σε ποιους agents θα εκτελεστούν οι τα abilities. Η σειρά με την οποία εκτελούνται τα abilities καθορίζεται αναλόγως του Plan (σχεδίου) που θα ακολουθηθεί.

Κρίνεται σκόπιμο να αναφερθούν περεταίρω προσπάθειες της επιστημονικής κοινότητας, που αφορούν εμβάθυνση στη χρήση προαναφερόμενου λογισμικού CALDERA.

#### 2.12.1.1 LAVA

Ο στόχος του υποσυστήματος LAVA [55] είναι να παρέχει έναν έξυπνο τρόπο για το CALDERA, ούτως ώστε να επιλέγει ενέργειες που θα εκτελεστούν προκειμένου να προσομοιώσει μια κόκκινη ομάδα. Οι αρχικές επαναλήψεις του CALDERA χρησιμοποιούσαν μια απλή μηχανή πεπερασμένης κατάστασης, καθώς οι ενέργειες υλοποιούνταν με σταθερή σειρά, συμπεριλαμβανομένων ορισμένων δευτερευόντων συνθηκών. Συγκεκριμένα αν και αυτό το σύστημα ήταν αποτελεσματικό, παρουσίαζε αδυναμίες στον τομέα ενσωμάτωσης νέων χαρακτηριστικών. Κατά την εφαρμογή νέων ενεργειών, το υπάρχον σύστημα έπρεπε να διαμορφωθεί ξανά χειροκίνητα. Επίσης παρουσίαζε αδυναμίες σε θέματα προβλεψιμότητας λόγω της άκαμπτης εσωτερικής λογικής που χρησιμοποιεί σε αντίθεση με ότι παρατηρείται σε κόκκινες ομάδες.

Εν τέλη παρατηρήθηκαν δυσκολίες κατά την προσαρμογή, αφού προσέγγιση του συστήματος σε θέματα κωδικοποίησης προφίλ ή προτιμήσεων που συνήθως συναντώνται σε κόκκινες ομάδες δεν ήταν ικανοποιητική. Έτσι σχεδιάστηκε το LAVA για να αντιμετωπίσει αυτά τα προβλήματα αντικαθιστώντας εξ ολοκλήρου την προσέγγιση πεπερασμένης κατάστασης-μηχανής, αντ' αυτού μεταβαίνοντας σε μια αρθρωτή προσέγγιση βασισμένη στον κλασικό σχεδιασμό. Αντί να εκτελούνται οι ενέργειες με μια προκαθορισμένη σειρά, οι ενέργειες στο LAVA ορίζονται ατομικά με τις λογικές απαιτήσεις και τα αποτελέσματά τους κατά την εκτέλεση.

Για τη λήψη αποφάσεων, το LAVA συνδυάζει ενέργειες για τη δημιουργία δυναμικών σχεδίων με βάση τη διαθέσιμη γνώση και ένα δεδομένο προφίλ αντιπάλου. Το CALDERA συμβουλεύεται το LAVA για το καλύτερο σχέδιο – αφού επιλεγεί, η πρώτη ενέργεια σε αυτό το σχέδιο εκτελείται, με τα επόμενα σχέδια να ενημερώνονται καθώς το σύστημα εξερευνά το δίκτυο και ενημερώνει τη βάση γνώσεων.

### 2.12.2 TROGDOR

Πρόκειται για ένα αυτοματοποιημένο σύστημα επικεντρωμένο στην αποστολή των Red Teamers. Το Trogdor [56] αναλαμβάνει την ανάλυση βάσει μοντέλου και κρίσιμων κόμβων για να παρουσιάσει οπτικά τον αντίκτυπο των ευάλωτων πόρων σε αποστολές στον κυβερνοχώρο. Συγκεκριμένα, δύναται να παρέχει κατανόηση των πιθανών επιπτώσεων που προκύπτουν σε μια αποστολή, από ευπάθειες στον κυβερνοχώρο και την υποστήριξη αποφάσεων κατά την επιλογή πιθανών στρατηγικών για τον μετριασμό αυτών των επιπτώσεων, καθώς επίσης είναι σε θέση να προχωρήσει περαιτέρω και να διερευνήσει αποτελέσματα πιθανών σεναρίων.

Κατά την κρίσιμη πτυχή της διαχείρισης των ευπαθειών όπου εμπίπτει η ιεράρχηση της προτεραιότητας των παραπάνω, που ουσιαστικά αφορά τον σαφή προσδιορισμό των περιουσιακών στοιχείων που είναι ευάλωτα, την πιθανότητα πώς και πότε μπορούν να εκμεταλλευτούν και, το πιο σημαντικό, τον αντίκτυπο που θα έχουν στην αποστολή που οδηγεί στο επίπεδο ανθεκτικότητας ή πλεονασμού που έχει ο οργανισμός όταν αυτά τα περιουσιακά στοιχεία υποβαθμίζονται ή καταστρέφονται, το Trogdor αναλαμβάνει με τη χρηση της βιβλιοθήκης των Τακτικών, Τεχνικών και Διαδικασιών (TTP) που προαναφέρθηκαν, για να μοντελοποιήσει την επίθεση κάθε αποστολής δημιουργώντας

αυτόματα γραφήματα επίθεσης, που μπορούν ενδεχομένως να προσεγγίσουν κρίσιμους πόρους για την αποστολή. Στη συνέχεια, υποστηρίζει τον χρήστη στην ανάλυση επιπτώσεων και στην ιεράρχηση πιθανών τρόπων δράσης για τη βελτίωση της ανθεκτικότητας της αποστολής.

Το Trogdor χρησιμοποιεί πολλαπλούς σχεδιαστές Τεχνητής Νοημοσύνης (AI) για την εκτέλεση αυτοματοποιημένης αξιολόγησης ευπάθειας δημιουργώντας γραφήματα επίθεσης που στοχεύουν κρίσιμους πόρους της αποστολής. Οι τεχνικές Visual Analytic (VA) εφαρμόζονται για την υποστήριξη της αξιολόγησης, της ανάλυσης κρίσιμου κόμβου και της ιεράρχησης προτεραιοτήτων, επισημαίνοντας τους βασικούς πόρους στα γραφήματα επίθεσης. Επίσης χρησιμοποιεί πρόσθετες τεχνολογίες, συμπεριλαμβανομένου ενός οντολογικού παγκόσμιου μοντέλου και μιας μηχανής αξιολόγησης ποιότητας μετά τον σχεδιασμό.

### 2.12.3 W3AF

Το w3af [57] το οποίο προήλθε από τη συντόμευση του Web Application Attack and Audit Framework, είναι ένα πλήρες περιβάλλον για έλεγχο και επίθεση σε εφαρμογές web. Αυτό το περιβάλλον παρέχει μια σταθερή πλατφόρμα για έλεγχο και Pentration Testing. Είναι εύκολο στη χρήση και την επεκτασιμότητά του, αφού το w3af έχει περισσότερες από 130 προσθήκες, για δοκιμές όπως SQL injection και Cross-Site Scripting (XSS). Το w3af είναι ένα δωρεάν εργαλείο, επομένως είναι αρκετά διαδεδομένο και χρησιμοποιείται για την ανίχνευση τρωτών σημεία του ιστού. Με τον πυρήνα του αλλά και των πρόσθετων (plugins) του, να είναι γραμμένα σε Python, το w3af έχει τη δυνατότητα να λειτουργήσει σε όλες τις πλατφόρμες λειτουργικών συστημάτων, αφού έχει εγκατασταθεί η Python φυσικά.

Συγκεκριμένα υπάρχουν τρεις τύποι πρόσθετων (plugins) που αφορούν την ανακάλυψη, νέων διευθύνσεων URL, φορμών και άλλων injection points, του ελέγχου, που λαμβάνουν τα injection points που ανευρέθηκαν κατά την ανακάλυψη και αποστέλλει ειδικά διαμορφωμένα δεδομένα σε όλα αυτά για να εντοπιστούν πιθανά τρωτά σημεία και εν τέλη της επίθεσης που σκοπό έχουν την εκμετάλλευση των τρωτών σημείων που εντοπίζονται από τους παραπάνω. ελέγχους. Συνήθως επιστρέφουν shell στον

απομακρυσμένο διακομιστή ή μια ένδειξη απομακρυσμένων πινάκων κατά την περίπτωση εκμετάλλευσης, SQL injection. Τέλος, μετά τη σάρωση του w3af, με την έκδοση των αποτελεσμάτων υπό τύπον αναφοράς, οι Red Teamers ενημερώνονται για τις λεπτομέρειες των τρωτών σημείων.

### 2.12.4 CARTT

Το CARTT [58] είναι γραμμένο σε γλώσσα Python και είναι χτισμένο πάνω από το MSF Console . Δεδομένου ότι το MSF είναι ένα εργαλείο εντατικής γραμμής εντολών και συνήθως είναι δύσκολο στη χρήση για τους άπειρους, το CARTT για να μετριάσει αυτό το μειονέκτημα, αντικαθιστά τη διεπαφή γραμμής εντολών (CLI), με μια γραφική διεπαφή χρήστη (GUI) για να μειώσει τη γνώση που χρειάζεται ο χρήστης για την εκτέλεση των διαδικαστικών βημάτων σε μια κυβερνοεπίθεση με χρήση του MSF. Για να γίνει αυτό, το CARTT χρησιμοποιεί scripts για να αυτοματοποιήσει πολύπλοκες εντολές που κανονικά θα έπρεπε να εισαχθούν σε κονσόλα. Επίσης έχει τη δυνατότητα να εμφανίζει τα αποτελέσματα του MSF στο GUI για προβολή τους άμεσα στον χρήστη.

Για να ξεκινήσει η διαδικασία, το CARTT πρέπει αρχικά να λάβει μια αναφορά ευπάθειας που δημιουργείται από το OpenVAS Greenbone Security Assistant (GSA) κατά τη σάρωση ευπάθειας. Η λήψη της αναφοράς γίνεται χειροκίνητα από το GSA GUI στον τοπικό κατάλογο αρχείων υπό τύπο XML, η οποία είναι και η μόνη χειροκίνητη λειτουργία που εκτελείται κατά τη διάρκεια της επίθεσης. Στη συνέχεια, το CARTT εισάγει την αναφορά ευπάθειας από τον τοπικό κατάλογο στη βάση δεδομένων του MSF. Μετά την εισαγωγή, κάθε ευπάθεια εμφανίζεται σε παράθυρο περιγραφής ευπαθειών στο CARTT GUI.

Στη συνέχεια, το CARTT GUI επιτρέπει στο χρήστη να πραγματοποιήσει αναζήτηση στη βάση δεδομένων του MFS για αντίστοιχες λειτουργικές μονάδες εκμετάλλευσης για καθεμία από τις ευπάθειες του κεντρικού υπολογιστή που έχουν εντοπιστεί. Ο χρήστης επιτρέπεται να επιλέξει έναν συγκεκριμένο κεντρικό υπολογιστή και να οπτικοποιήσει τα τρωτά σημεία που εντοπίζονται από τη σάρωση ευπάθειας GSA. Ο χρήστης μπορεί στη συνέχεια να επιλέξει μια μεμονωμένη ευπάθεια προς εκμετάλλευση. Μόλις επιλεγεί η ευπάθεια, το CARTT εμφανίζει μια λίστα από πιθανά. Exploits που μπορεί να χρησιμοποιηθούν για την πραγματοποίηση της επίθεσης εναντίον του επιδιωκόμενου κεντρικού υπολογιστή. Αυτή η λίστα παρουσιάζεται στον χρήστη για επιλογή.

Εν τέλη, το CARTT παρουσιάζει στον χρήστη μια περιγραφή του επιλεγμένου exploit για χρήση. Αυτό βοηθά τον χρήστη να προσδιορίσει εάν το επιλεγμένο exploit είναι το κατάλληλο exploit για τον προσδιορισμένο υπολογιστή-στόχο. Αφού ο χρήστης επιλέξει το exploit,μ τότε γίνεται αυτόματα αποστολή payload, με τη διεπαφή GUI του CARTT να παρέχει ανατροφοδότηση στον χρήστη σχετικά με την κατάσταση της εκτέλεσης της εκμετάλλευσης.

### 2.12.5 HARMer

Σύμφωνα με τους Hong και Kim [59][60] Το HARM ή αλλιώς Hierarchical Attack Representation Model, αφορά ένα πλαίσιο όπου ο σχεδιασμός και η εφαρμογή του βασίζεται σε ένα επεκτάσιμο γραφικό μοντέλο ασφαλείας. Αποτελείται κυρίως από δύο επίπεδα, το ανώτερο επίπεδο που εεντοπίζει τις πληροφορίες προσβασιμότητας δικτύου (χρησιμοποιώντας Attack Graphs όπου μοντελοποιεί μόνο τις πληροφορίες προσβασιμότητας) και το κατώτερο επίπεδο που εντοπίζει τις πληροφορίες ευπάθειας κάθε κόμβου στο δίκτυο, χρησιμοποιώντας Attack trees. Εν συνεχεία το HARMer [61] είναι ένα νέο πλαίσιο για την αυτοματοποίηση, τη μοντελοποίηση, την εκτέλεση επιθέσεων στον κυβερνοχώρο και της ανίχνευσης απειλών, καθώς αναπτύσσει μια ντετερμινιστική στρατηγική σχεδιασμού (που ονομάζεται metric based planning) με βάση του το HARM, για τη σχεδίαση συστηματικών επιθέσεων για αυτοματοποιημένες ενέργειες.

Το αυτοματοποιημένο επιθετικό πλαίσιο επιτυγχάνεται με την υλοποίηση τεσσάρων κυρίων φάσεων, την συλλογή των δεδομένων, την κατασκευή και ανάλυση του μοντέλου ασφαλείας, τον επιθετικό σχεδιασμό και εν τέλη την υλοποίηση και εκτίμηση της επίθεσης. Κατά την πρώτη φάση, της συλλογής των δεδομένων – πληροφοριών, το εν λόγω πλαίσιο με ενσωματωμένα εργαλεία σάρωσης ευπαθειών, εντοπισμού δικτύων και ανοιχτών θυρών, αυτόματα συλλέγει όλες τις απαιτούμενες πληροφορίες. Μερικά από τα εργαλεία που χρησιμοποιούνται για την εκπλήρωση των παραπάνω είναι το NMAP [29], το Nessus [31] και το OpenVAS [32].

Στη δεύτερη φάση δημιουργείται ένα HARM δύο επιπέδων του δικτύου, χρησιμοποιώντας τις πληροφορίες που συλλέχθηκαν από τη φάση 1. Όλες οι πιθανές

διαδρομές επίθεσης καταγράφονται και απαριθμούνται στο HARM, οπότε καταγράφονται και τα πιθανά σενάρια – σχέδια επίθεσης. Για την ανάλυση ασφάλειας, ο υπεύθυνος λήψης αποφάσεων ασφαλείας θα επιλέξει τις μετρήσεις ασφαλείας που θα χρησιμοποιηθούν με το μοντέλο ασφαλείας. Σε αυτό το σημείο, οι υπολογισμένες μετρήσεις ασφαλείας μέσω του μοντέλου θα χρησιμοποιηθούν για τη λήψη αποφάσεων σχετικά με το σχέδιο επίθεσης. Επομένως, αυτή η φάση αξιολογεί κάθε μονοπάτι επίθεσης με βάση τον κίνδυνο.

Προχωρώντας, η τρίτη φάση είναι υπεύθυνη για το σχεδιασμό και τη δημιουργία των ενεργειών υπό τύπο σχεδίων για τον επιτιθέμενο. Το σχέδιο μπορεί να περιλαμβάνει μια ανταπόκριση από τον επόμενο κεντρικό υπολογιστή/στόχο, για σάρωση θυρών του εύρους των διατιθέμενων IP ή στοχευμένες ενέργειες, όπως η εκμετάλλευση μιας ευπάθειας λογισμικού ενός κεντρικού υπολογιστή, η αποστολή κακόβουλου λογισμικού phishing/spear phishing με ηλεκτρονικό ταχυδρομείο κ.λπ. Μπορεί να χρησιμοποιηθούν διάφορες προσεγγίσεις, με το HARM να δημιουργεί στρατηγικά πιθανά σχέδια επίθεσης.

Με την χρήση αλγόριθμων ως τα βασικά στοιχεία στη φάση σχεδιασμού επίθεσης του αυτοματοποιημένου πλαισίου, χρησιμοποιούνται τρείς διαφορετικές προσεγγίσεις που βασίζονται σε μετρήσιμα δεδομένα για την επιλογή του attack plan. Συγκεκριμένα χρησιμοποιούνται προσεγγίσεις βάσει διαδρομής, όπως για παράδειγμα είναι η συντομότερη διαδρομή επίθεσης (shortest attack path), ακολουθούν οι σύνθετες μετρήσεις, όπως για παράδειγμα ο συνδυασμός πιθανότητας επιτυχίας της επίθεσης αναλόγως του attack plan) και τελευταίο η ατομική μέτρηση όπου λαμβάνει υπόψη μόνο τον παράγοντα του τελικού κόστους της επίθεσης.

Εν τέλη, τα δεδομένα που παράγονται από τη φάση σχεδιασμού επίθεσης χρησιμοποιούνται στην τελευταία φάση, της εκτέλεσης και αξιολόγησης της επίθεσης, για τις περαιτέρω ενέργειες του λογισμικού. Οι επιθέσεις εκτελούνται στα τρωτά σημεία που έχουν ανακαλυφθεί από την αρχική φάση, της συλλογής δεδομένων - πληροφοριών. Είναι σημαντικό να τονιστεί ότι η φάση εκτέλεσης επίθεσης λειτουργεί αναδρομικά με τη φάση σχεδιασμού επίθεσης. Κατά την εκμετάλλευση εξάγονται οι πληροφορίες του κάθε κόμβου, παράλληλα με την εύρεση της εκμετάλλευση που εξάγεται και στη συνέχεια ξεκινά η διαδικασία της εκμετάλλευσης.

Με το πέρας της όλης διαδικασίας παράγονται σχόλια ανατροφοδότησης της ακολουθίας εκμετάλλευσης. Συγκεκριμένα, ανατροφοδότηση παράγεται για τους κόμβους όπου κατά τη διαδρομή επίθεσης έχουν εκμεταλλευτεί με επιτυχία καθώς επίσης και ποιοι από τους κόμβους στη διαδρομή επίθεσης δεν μπόρεσαν να αξιοποιηθούν κατάλληλα. Η αποτυχία ευθύνεται για διάφορους λόγους. Πιθανόν ένας κόμβος στη διαδρομή επίθεσης να μην είναι εκμεταλλεύσιμος, διότι οι πληροφορίες ευπάθειας που παρέχονται από τον σχεδιαστή (φάση 3) δεν ταιριάζουν με τις πραγματικές ευπάθειες του κεντρικού υπολογιστή ή ο κεντρικός υπολογιστής είναι εξ' αρχής εκτός λειτουργίας ή μη διαθέσιμος.

# Κεφάλαιο 3
# Μεθοδολογία

## 3.1 Σκοπός και Είδος Έρευνας

Η παρούσα μεταπτυχιακή διατριβή έχει ως σκοπό την αυτοματοποίηση των ενεργειών που απαιτούνται για την συλλογή πληροφοριών, αναγνώριση και εκμετάλλευση ευπαθειών, καθώς και την δημιουργία ενημερωτικής αναφοράς αποτελεσμάτων, διαφόρων πληροφοριακών συστημάτων τύπου Virtual Machine σε περιβάλλοντα Cyber Range, με τη βοήθεια υπαρχόντων open source εργαλείων και script με στόχο την δυνατότητα εκτέλεσης του αναπτυχθέν λογισμικού από ανθρώπους με λιγοστή γνώση επί του αντικειμένου, κέρδος πολύτιμου χρόνου, σκέψης, και εν τέλη της επίτευξης του επιθυμητού αποτελέσματος. Η εν λόγω ποσοτική έρευνα όπου η ερευνητική στρατηγική εστιάζει στην ποσοτικοποίηση, στηρίζεται στη συλλογή και ανάλυση δεδομένων με έμφαση στη πειραματική δοκιμή.

## 3.2 Βασικά Ερευνητικά Ερωτήματα

Για την υλοποίηση της παρούσης προέκυψαν τα παρακάτω ερευνητικά ερωτήματα:

1. Υπάρχουν αρκετά script στη βιβλιογραφία τα οποία δύναται να χρησιμοποιηθούν σε επιθέσεις σε Cyber Range περιβάλλοντα;

2. Πώς μπορούν να αυτοματοποιηθούν τα διάφορα scripts ή και tools με επιτυχία;

3. Μπορούν να αποδώσουν τα επιλεχθέντα tools – scripts σε αυτοματοποιημένες επιθέσεις;

4. Μπορεί να εξοικονομηθεί πολύτιμος χρόνος με χρήση μιας αλυσίδας επίθεσης;

5. Υπάρχουν είδη αυτοματοποιημένα εργαλεία τα οποία χρησιμοποιούνται σε cyber range περιβάλλοντα ή και σε πραγματικά συστήματα ;

## 3.3 Μοντέλο Spiral

Το σπειροειδές μοντέλο [62] συνδυάζει την ιδέα της επαναληπτικής ανάπτυξης (Iterative Model) με τις συστηματικές, ελεγχόμενες πτυχές του μοντέλου καταρράκτη (Waterfall Model). Το εν λόγω μοντέλο είναι ένας συνδυασμός του μοντέλου επαναληπτικής διαδικασίας ανάπτυξης και του μοντέλου διαδοχικής γραμμικής ανάπτυξης, δηλαδή του μοντέλου καταρράκτη με πολύ μεγάλη έμφαση στην ανάλυση κινδύνου. Επιτρέπει σταδιακές απελευθερώσεις του παραχθέντος λογισμικού και η σταδιακή βελτίωση του σε κάθε επανάληψη γύρω από τη σπείρα.

### 3.3.1 Επαναληπτικές Φάσεις Μοντέλου

Το σπειροειδές μοντέλο έχει τέσσερις φάσεις. Ένα λογισμικό περνά επανειλημμένα από αυτές τις φάσεις σε επαναλήψεις που ονομάζονται Spirals.

#### 3.3.1.1 Ταυτοποίηση

Αυτή η φάση ξεκινά με τη συγκέντρωση των απαιτήσεων στο βασικό σπιράλ. Στις επόμενες σπείρες καθώς το λογισμικό ωριμάζει, ο προσδιορισμός των απαιτήσεων συστήματος, των απαιτήσεων υποσυστήματος και των απαιτήσεων μονάδας γίνονται όλα σε αυτή τη φάση. Αυτή η φάση περιλαμβάνει επίσης την κατανόηση των απαιτήσεων του συστήματος από τον αναλυτή του συστήματος. Στο τέλος της σπείρας, το λογισμικό αναπτύσσεται και γίνετε η αποδέσμευση του στο κοινό για χρήση.

### 3.3.1.2 Σχεδίαση

Η φάση Σχεδίασης ξεκινά με τον εννοιολογικό σχεδιασμό στη βασική σπείρα και περιλαμβάνει αρχιτεκτονικό σχεδιασμό, λογικό σχεδιασμό ενοτήτων, φυσικό σχεδιασμό προϊόντος και τον τελικό σχεδιασμό στις επόμενες σπείρες.

### 3.3.1.3 Κατασκευή

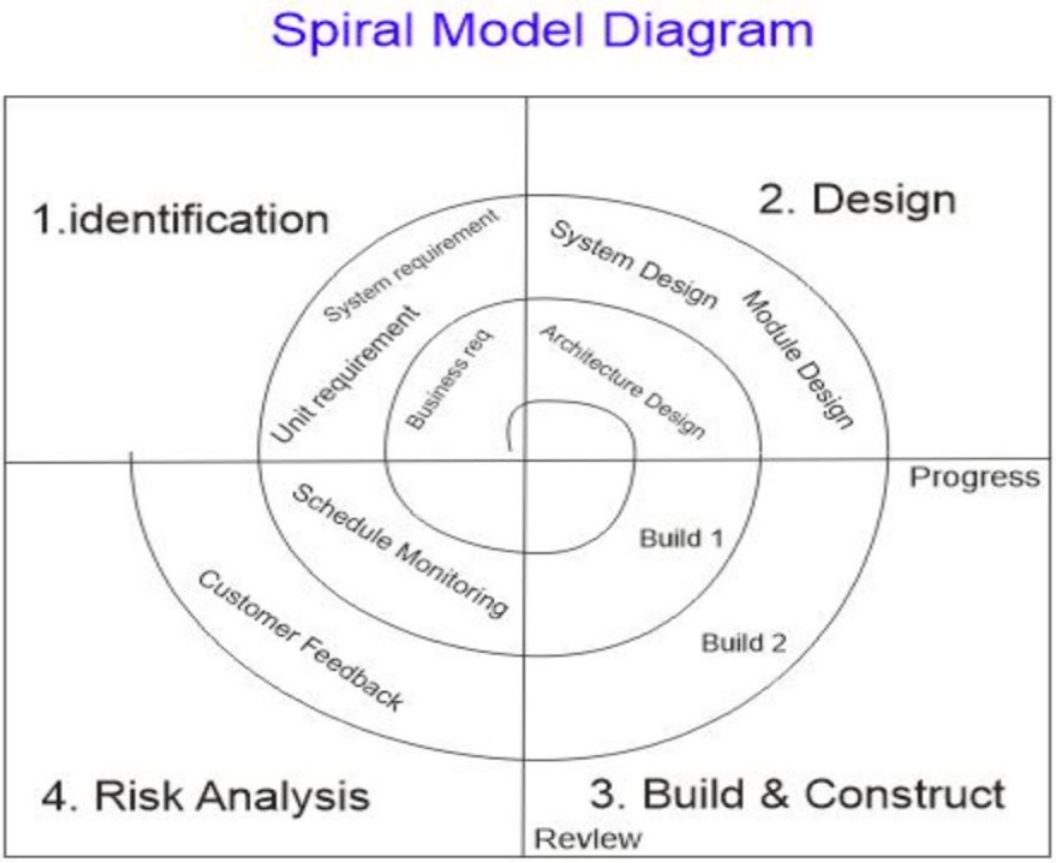
Η φάση κατασκευής αναφέρεται στην παραγωγή του πραγματικού λογισμικού σε κάθε σπείρα. Στην αρχική σπείρα, όταν ο σχεδιασμός του λογισμικού υλοποιείται, σε αυτή τη φάση αναπτύσσεται ένα POC (Proof of Concept) για να λάβει τα σχόλια των πελατών. Στη συνέχεια, στις επόμενες σπείρες, με μεγαλύτερη σαφήνεια στις απαιτήσεις και τις λεπτομέρειες σχεδιασμού, παράγεται ένα λειτουργικό μοντέλο του λογισμικού που ονομάζεται build με έναν αριθμό έκδοσης.

### 3.3.1.4 Αξιολόγηση και Ανάλυση Κινδύνων

Η Ανάλυση Κινδύνου περιλαμβάνει τον εντοπισμό, την εκτίμηση και την παρακολούθηση των κινδύνων τεχνικής σκοπιμότητας και διαχείρισης, όπως η απόκλιση χρονοδιαγράμματος και η υπέρβαση του κόστους. Μετά τη δοκιμή της κατασκευής, στο τέλος της πρώτης επανάληψης, κατά την αξιολόγηση του λογισμικό παρέχεται ανατροφοδότηση.

## 3.3.2 Οπτικό Διάγραμμα Μοντέλου

Η παρακάτω Εικόνα 10 είναι μια αναπαράσταση του σπειροειδούς μοντέλου, παραθέτοντας τις δραστηριότητες σε κάθε φάση.




*Εικόνα 10 (Διάγραμμα Σπειροειδούς Μοντέλλου)* [63]

### 3.3.3 Παράγοντες Επιλογής Μοντέλου

Αν και όλη η διαδικασία, αλλά και διαχείριση του Spiral Model είναι ιδιαίτερα πολύπλοκη το εν λόγω μοντέλο πλεονεκτεί στο ότι οι μεταβαλλόμενες απαιτήσεις μπορούν να ικανοποιηθούν σχετικά εύκολα λόγω της φύσης του, ενώ παράλληλα αυτές μπορούν να αποτυπωθούν με μεγαλύτερη ακρίβεια. Η περαιτέρω ανάπτυξη μπορεί να διαχωριστεί σε μικρότερα μέρη, όπου τα επικίνδυνα μπορούν να αναπτυχθούν νωρίτερα, κάτι που βοηθά ιδιαίτερα στην καλύτερη διαχείριση κινδύνου [64].

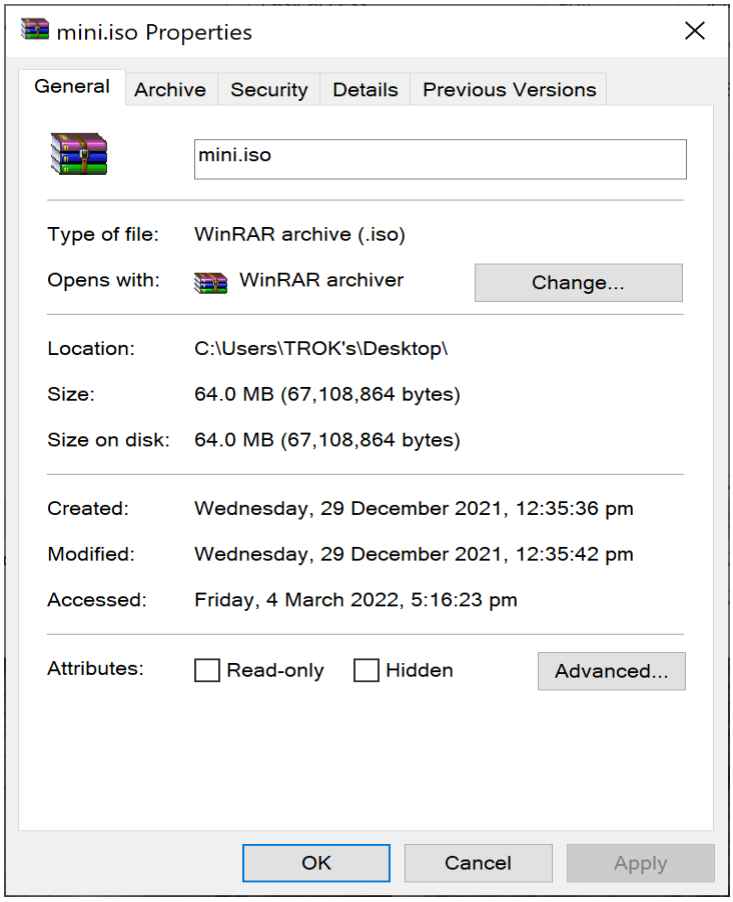
# Κεφάλαιο 4
# Διαδικαστικά Υλοποίησης

## 4.1 Αρχική Ιδέα

Ο αυτοματισμός για να επιτύχει, πρέπει να είναι βασισμένος σε κάποια ιδιαίτερα χαρακτηριστικά. Αυτά δεν είναι άλλα, από την απλότητα στη χρήση και συγκεκριμένα στις ελάχιστες απαραίτητες ενέργειες του χρήστη με τη μέγιστη αποδοτικότητα στην επίτευξη του ζητούμενου, στην εργονομική και εύχρηστη διεπαφή χρήστη – μηχανής, καθώς το λογισμικό πρέπει να είναι εγκατεστημένο σε ένα απλό και φιλόξενο περιβάλλον ούτως ώστε να μπορεί να χρησιμοποιηθεί και από μη έμπειρο προσωπικό. Εν τέλη το όλο εγχείρημα πρέπει να είναι ελαφρύ και να μην καταναλώνει ιδιαίτερους λειτουργικούς πόρους με απώτερο σκοπό να είναι διαθέσιμο και για παλαιότερα συστήματα που πιθανότερα να είναι και πιο ευάλωτα, με τα σημερινά δεδομένα.

Έχοντας τα παραπάνω ως κατευθυντήριες γραμμές, ξεκίνησαν οι ενέργειες για ανεύρεσης μιας βάσης, ενός απλού λειτουργικού συστήματος ελαφριού σχεδιασμού, αλλά παράλληλα με όλες τις δυνατότητες για να μπορεί να αποδώσει σε ότι πιθανή ανάγκη προκύψει. Η κατάληξη ήταν στην λήψη και εγκατάσταση του λογισμικού Ubuntu 18.04 το οποίο βρισκόταν διαθέσιμο με minimum requirements. Το εν λόγω λειτουργικό λαμβάνει πακέτα από ηλεκτρονικά αρχεία κατά την εγκατάσταση αντί να τα παρέχει στο ίδιο το μέσο εγκατάστασης. Η λήψη πακέτων κατά τον χρόνο εγκατάστασης μειώνει το μέγεθος της εικόνας iso, καθώς και παρέχει μόνο τα πακέτα που απαιτούνται για την εγκατάσταση. Η εξοικονόμηση χρόνου λήψης που επιτυγχάνεται με τη χρήση ενός mini.iso μπορεί να είναι σημαντική, καθώς γίνεται λήψη μόνο των τρεχόντων πακέτων, επομένως δεν χρειάζεται να αναβαθμιστούν τα πακέτα αμέσως μετά την εγκατάσταση καθώς επίσης κατά την εγκατάσταση ο χρήστης μπορεί να προσθέσει τα πακέτα τα οποία του είναι χρήσιμα και παράλληλα τα εξαρτώμενα τους για να μπορούν να λειτουργήσουν. Γενικότερα μπορούμε να την χαρακτηρίσουμε ως την πλέον προσαρμοσμένη εγκατάσταση. Επίσης κρίνετε σκόπιμο να αναφερθεί ότι είναι και μια τακτική που χρησιμοποιείται από επαγγελματίες Red Teamers στο χώρο, αφού

εγκαθιστούν ότι τους είναι χρήσιμο και αναγκαίο για την επίτευξη του τελικού τους στόχου, χωρίς να χρειάζεται να επιλέγουν από την πληθώρα των εργαλείων και scripts που κυκλοφορούν. Στην παρακάτω εικόνα φαίνονται τα αρχικά χαρακτηριστικά του εν λόγω mini.iso[65] του οποίου χρησιμοποίησα για την εγκατάσταση του λειτουργικού.



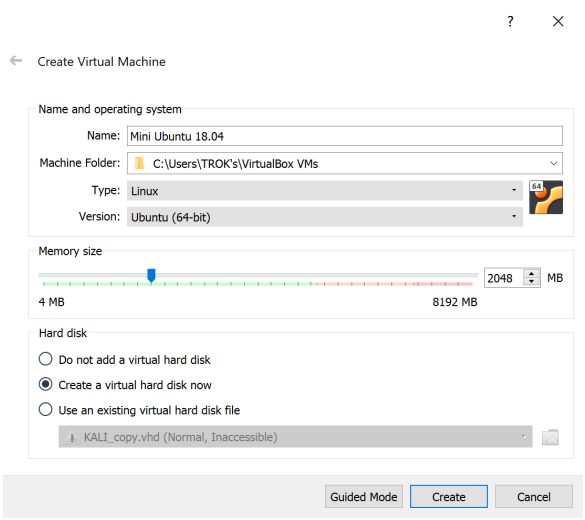

*Εικόνα 11(Ιδιότητες mini.iso)*

## 4.2 Εγκατάσταση Λειτουργικού Mini Ubuntu 18.04

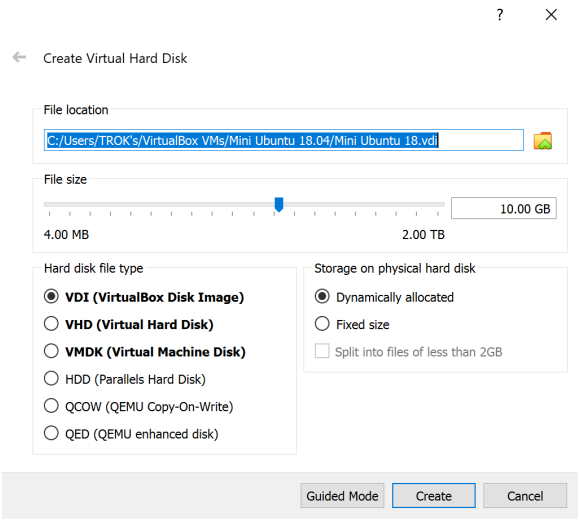
Η Εγκατάσταση έγινε σε περιβάλλον Virtualization, με την χρήση του λογισμικού Virtual Box [66], το οποίο επιτρέπει την εγκατάσταση λειτουργικών συστημάτων "Guests" πάνω σε υφιστάμενα χρησιμοποιούμενα λειτουργικά συστήματα χαρακτηριζόμενα ως "Hosts". Αυτή η πρακτική έχει πολλά πολλαπλά οφέλη καθώς δίνεται η δυνατότητα χρήσης πολλαπλών λειτουργικών συστημάτων παράλληλα, ενώ η εγκατάσταση τους είναι ιδιαίτερα απλή και φιλική προς όλους τους χρήστες που Θα το χρησιμοποιήσουν. Τα εγκατεστημένα λειτουργικά στο Virtual Box δύναται να αποθηκευτούν για σκοπούς ασφαλείας σε εξωτερικούς δίσκους, σε cloud ή και ακόμα να αποσταλούν σε τρίτα

πρόσωπα για να χρησιμοποιηθούν εκ νέου. Ακόμα ένα βασικό συστατικό τους είναι η δικλείδα ασφάλειας κατά την χρήση τους καθώς είναι απομονωμένα από τον “host” με αποτέλεσμα να μην υπάρχει κίνδυνος λήψης κακόβουλου λογισμικού στο κεντρικό λειτουργικό. [67]

Η Εγκατάσταση έγινε με σχετικά μικρές απαιτήσεις, αφού είναι μια βασική παράμετρος για την χρήση του συστήματος. Χρησιμοποιήθηκαν 2GB για κύρια μνήμη (RAM), 10 GB για βοηθητική μνήμη (HDD), ενώ για την επεξεργασία 1 μόνο επεξεργαστή (CPU), όπως φαίνεται και στις παρακάτω εικόνες 12 και 13.

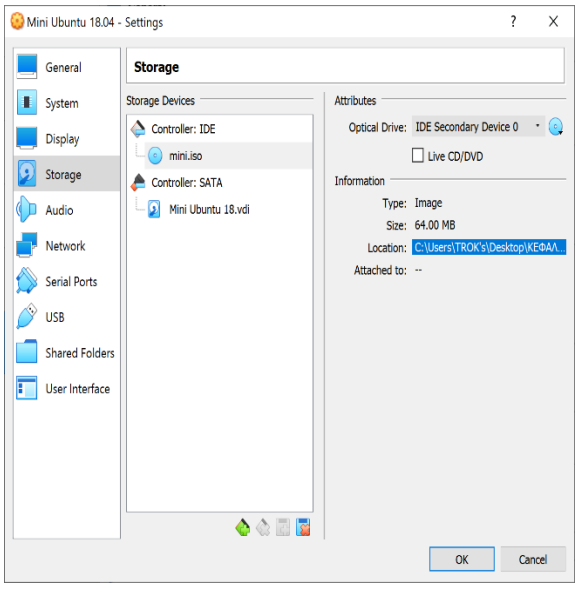



*Εικόνα 12 (Αρχική Παραμετροποίηση Κύριας Μνήμης)*

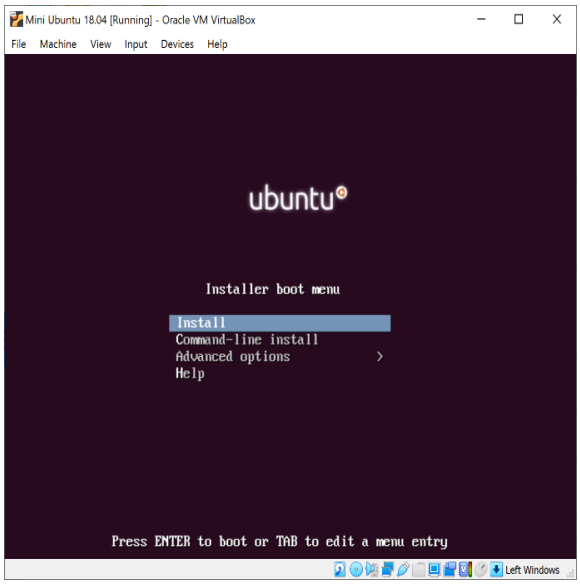



*Εικόνα 13(Αρχική Παραμετροποίηση Βοηθητικής Μνήμης)*

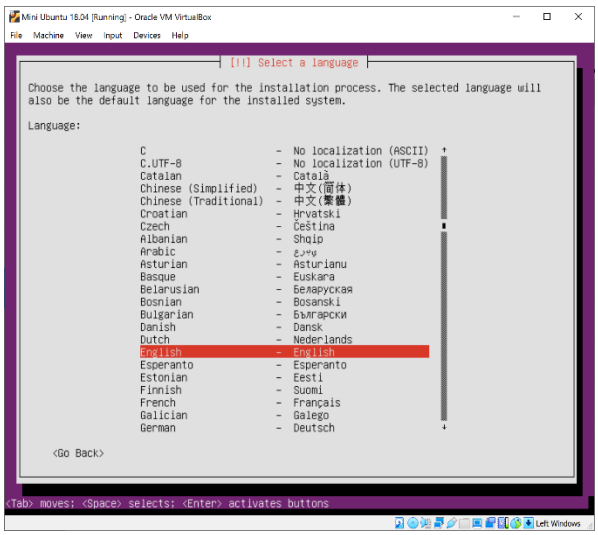
Η Εγκατάσταση διάρκεσε περίπου 60 λεπτών και έγινε με την παρακάτω σειρά όπως τις εικόνες 14 έως 22.



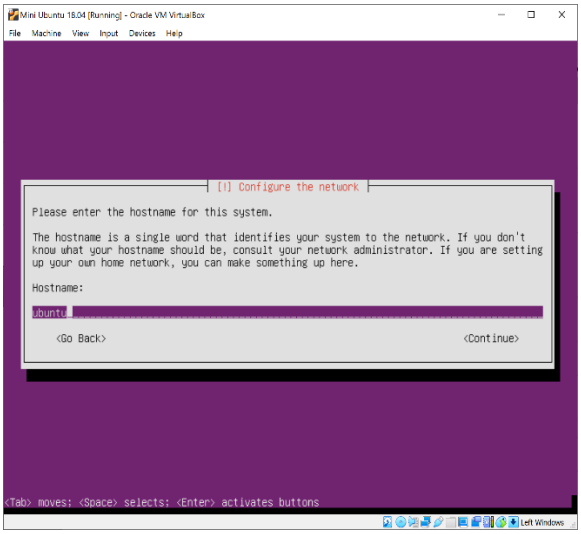

*Εικόνα 14 (Εισαγωγή του mini.iso)*



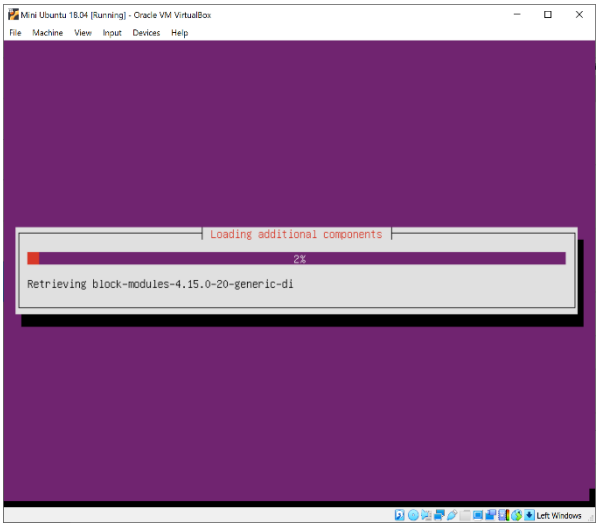

*Εικόνα 15 (Αρχική Εικόνα Εγκατάστασης)*



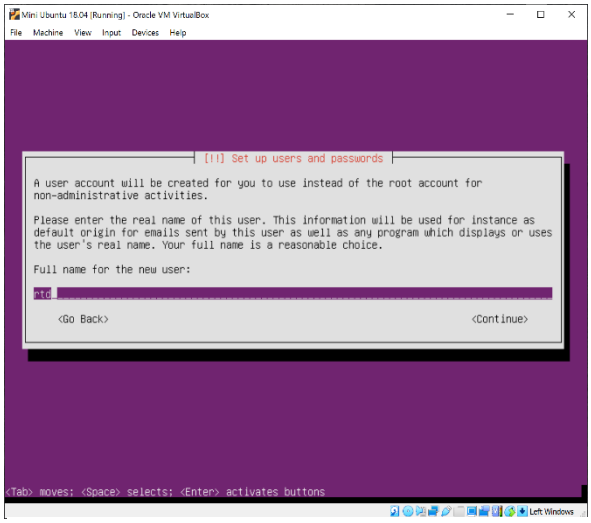

*Εικόνα 16 (Επιλογή Γλώσσας)*



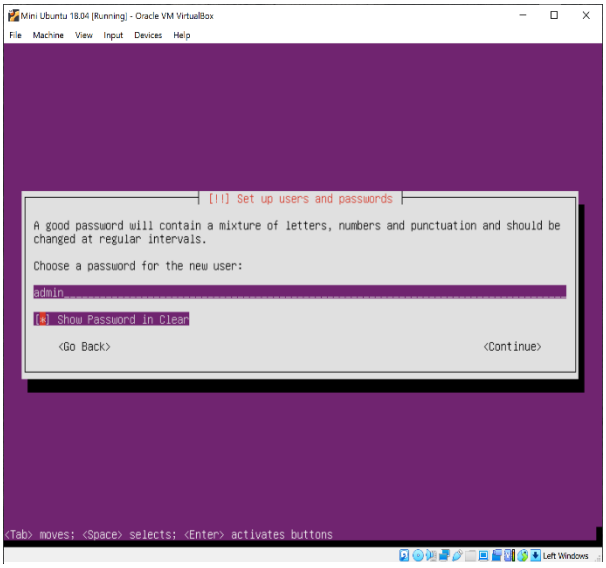

*Εικόνα 17 (Ρυθμίσεις Δικτύου)*



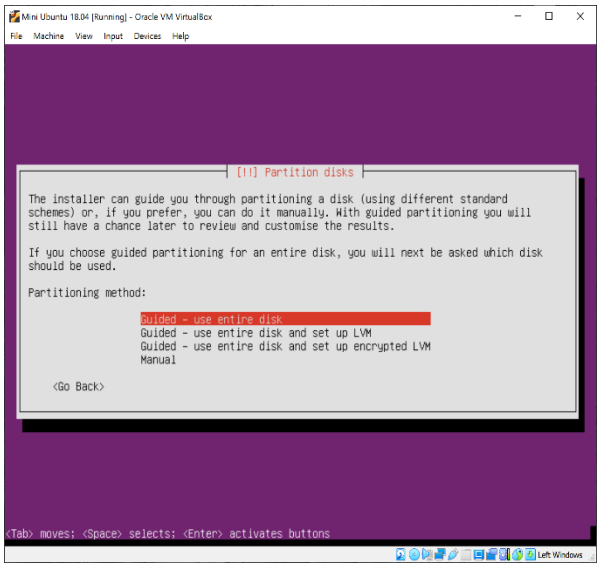

*Εικόνα 18 (Λήψη Επιπρόσθετων Πακέτων)*



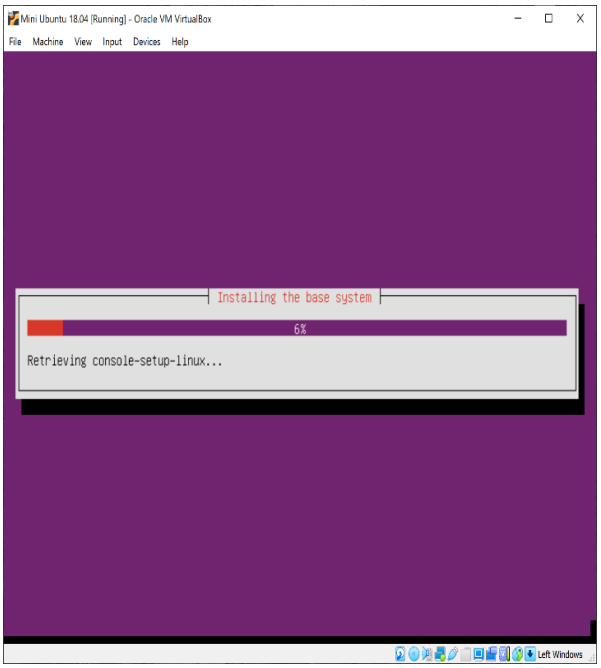

*Εικόνα 19 (Ρύθμιση Username Χρήστη)*



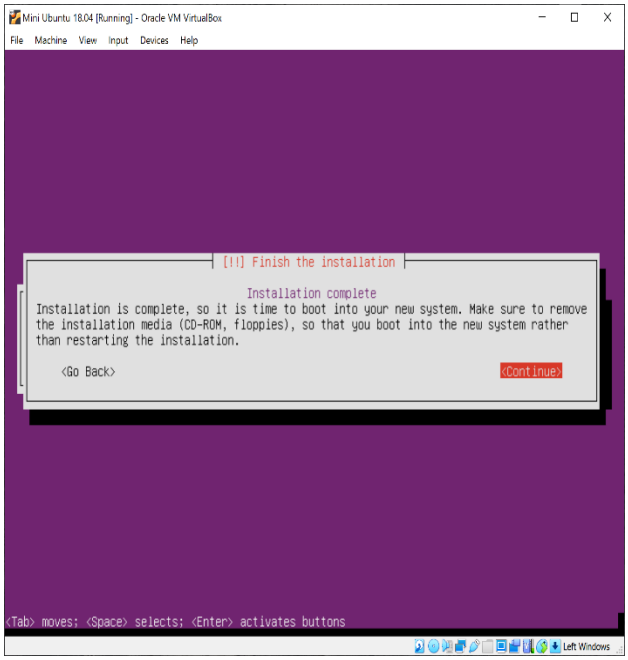

*Εικόνα 20 (Επιλογή συνθηματικού)*



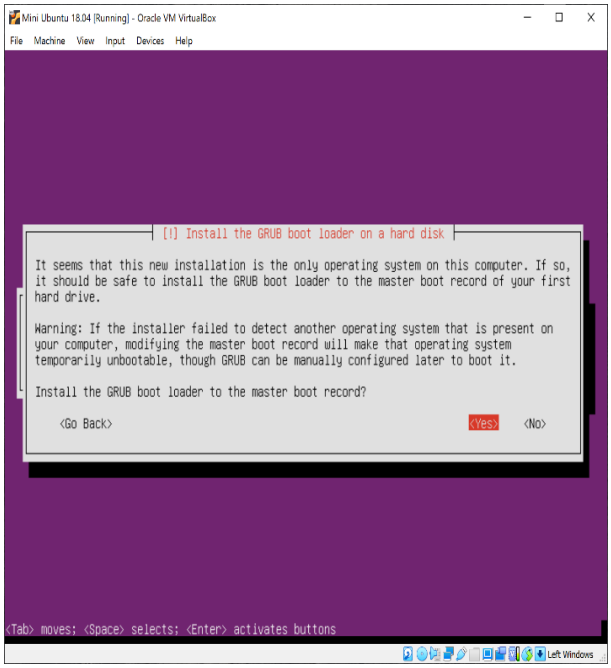

*Εικόνα 21 (Επιλογή εγκατάστασης σε όλο το Partition)*



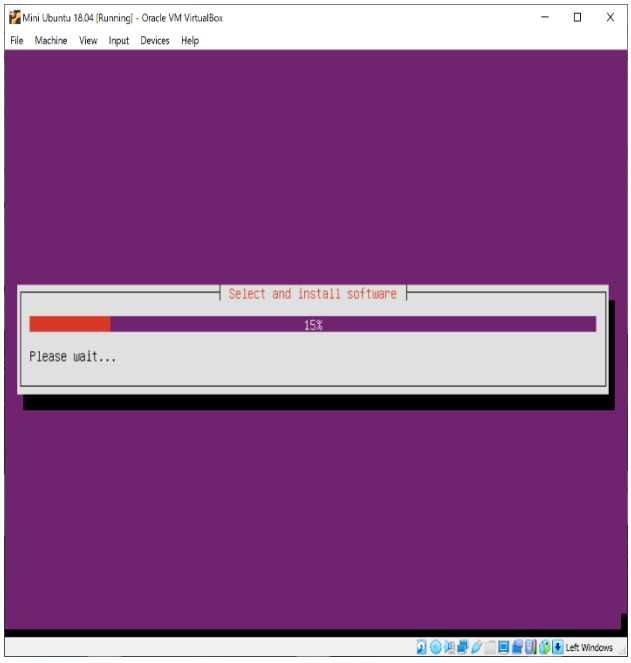

*Εικόνα 22 (Εγκατάσταση βασικών λειτουργειών)*




*Εικόνα 23 (Εγκατάσταση του συστήματος)*




*Εικόνα 24 (Εγκατάσταση Bootloader)*

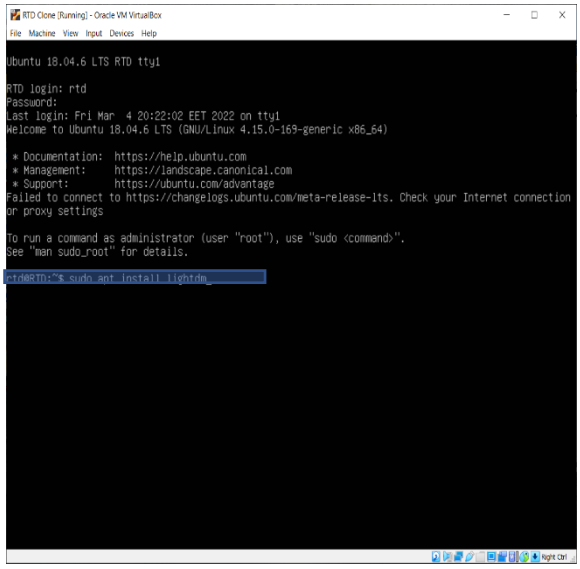



*Εικόνα 25 (Επιτυχής εγκατάσταση)*

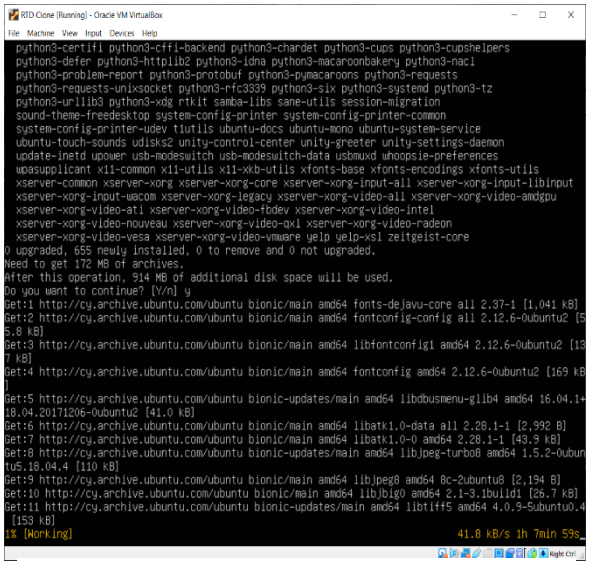
Αφού έγινε η επιτυχής εγκατάσταση του λειτουργικού, έφτασα ουσιαστικά σε μια μαύρη εικόνα η οποία ουσιαστικά ξεκίνησε η εγκατάσταση όλων των πακέτων και των εξαρτημένων τους από κονσόλα, για να μπορεί το Distribution εκτός από το να είναι ελαφρύ, να είναι και φιλικό προς τους χρήστες.

## 4.3 Εγκατάσταση GUI και Αναγκαίων Πακέτων

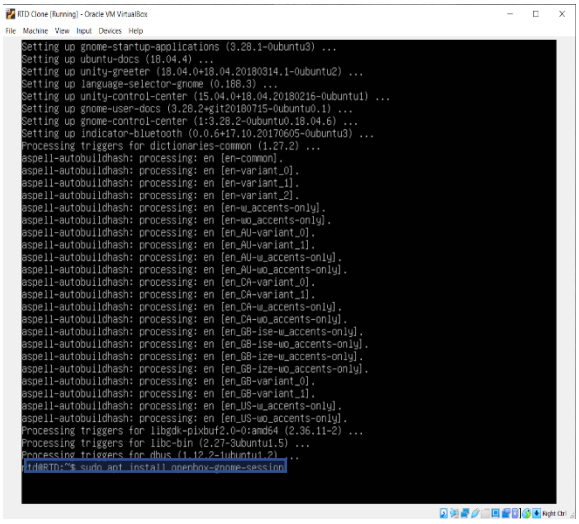
Η εγκατάσταση ξεκίνησε με την λήψη του lightdm, το οποίο και έδωσε το graphical interface στο Distribution, εν συνεχεία του openbox-gnome-session ως session manager, του openbox ως window manager και του gnome-terminal για χρήση της κονσόλας, όπως φαίνονται στις εικόνες 26 εώς 30.



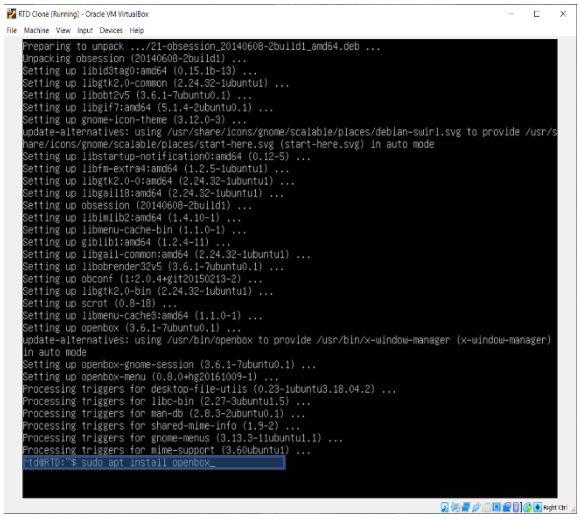

*Εικόνα 26 (Εγκατάσταση lightdm)*



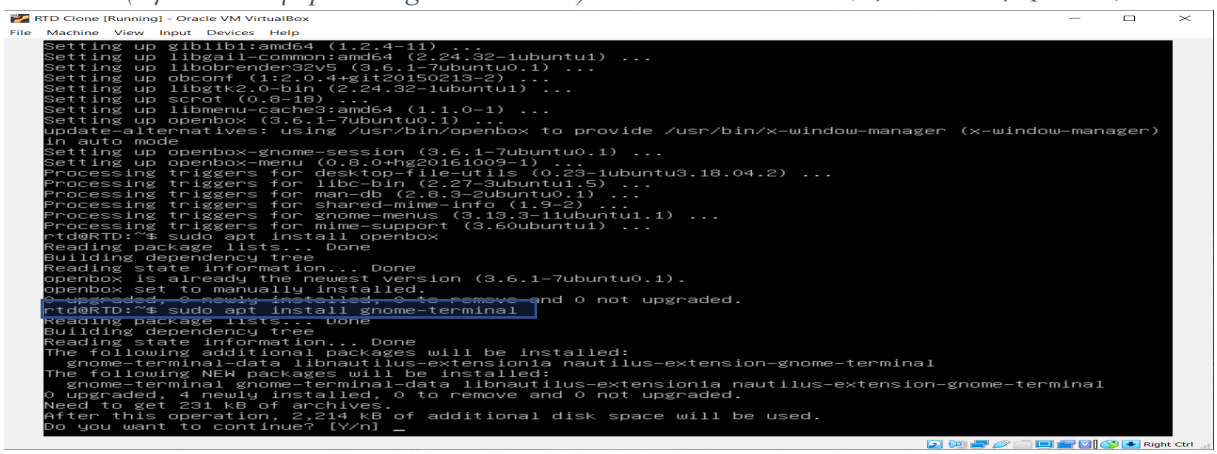

*Εικόνα 27(Εγκατάσταση lightdm)*




*Εικόνα 28 (Εγκατάσταση openbox-gnome-session)*




*Εικόνα 29 (Εγκατάσταση openbox)*



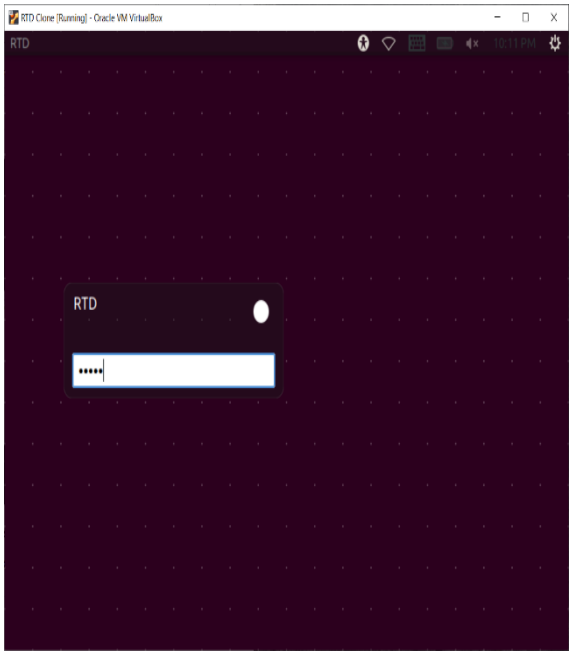

*Εικόνα 30 (Εγκατάσταση gnome-terminal)*

που θα βοηθήσουν στο κτίσιμο της εμφάνισης αλλά και λειτουργίας ούτως ώστε να ολοκληρωθεί το Distribution. Συγκεκριμέν έγινε η εγκατάσταση των πακέτων σύμφωνα με τον παρακάτω Πίνακα 2 χρησιμοπιώντας την κονσόλα και sudo apt install <ΟΝΟΜΑΣΙΑ ΠΑΚΕΤΟΥ>, ένα κάθε φορά με την παρακάτω σειρά σύμφωνα με τον αύξων αριθμό.

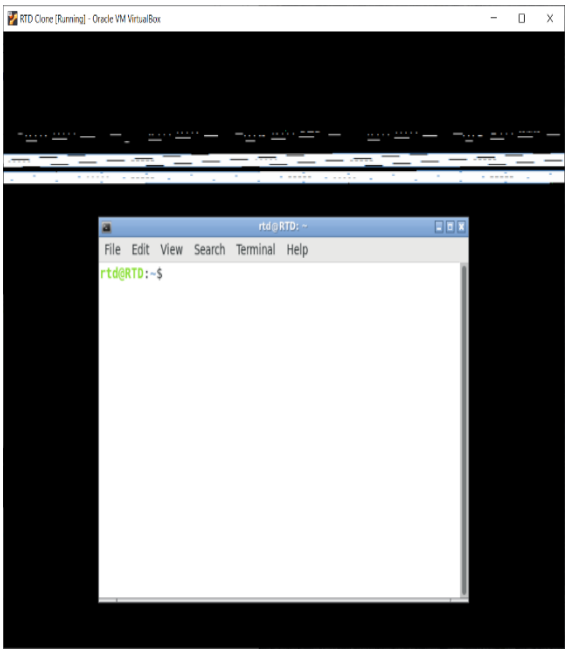
| Α/Α | ΟΝΟΜΑΣΙΑ ΠΑΚΕΤΟΥ | ΧΡΗΣΙΜΟΤΗΤΑ ΠΑΚΕΤΟΥ |
|---|---|---|
| 1 | obmenu | Openbox Editor |
| 2 | Gedit | Text Editor Installation |
| 3 | Tint2 | Taskbar Installation |
| 4 | docky | Dock Installation |
| 5 | xcompmgr | Dock to Hide Dependency |
| 6 | nitrogen | Background Installation |
| 7 | Ubuntu-wallpapers | Wallpapers Download |
| 8 | pcmanfm | File Manager Installation |
| 9 | lxappearance | Theme Switcher Installation |
| 10 | firefox | Browser Installation |
| 11 | Pulseaudio | Sound Server Installation |
| 12 | Pavucontrol | Volume Control Installation |
| 13 | volti | Tray Audio System Control |
| 14 | Sox | Sound Player in Terminal Installation |
| 15 | Libsox-fmt-all | Audio Codecs Installation |
| 16 | Lxtask | Task Manager Installation |
| 17 | xrandr | Screen Resolution Installation |
| 18 | arandr | Screen Resolution Installation (GUI) |
| 19 | Net-tools | Net tools Installation |
| 20 | Software-properties-common | Adding Repositories |
| 21 | Gparted | Disk Space Management Installation |
| 22 | zip | Zip archiver Installation |
| 23 | Unzipp | Zip unarchiver Installation |
| 24 | Git | Git Cloner Installation |
| 25 | Python-pip | Python Pip Dependencies/Repository Installer |
| 26 | Python3-pip | |
| 27 | Pip3 install -upgrade pip | Upgrade of PIP |
| 28 | Libssl-dev | SSL Libraries Installation |
| 29 | Curl | Terminal Downloads Installation |

*Πίνακας 2 (Ονομασία Πακέτων που Εγκαταστάθηκαν)*

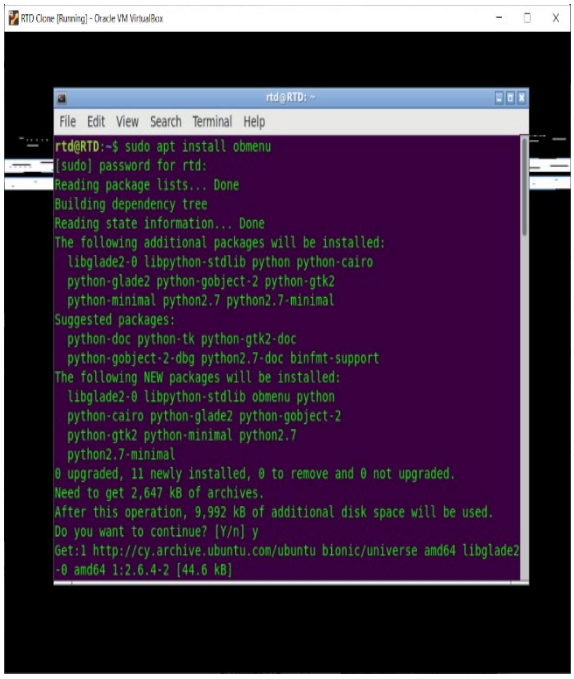
Με την εγκατάσταση του openbox-gnome-session έλαβα την πρώτη οθόνη και ακολούθως με την εισαγωγή των συνθηματικών εισήλθα στο Desktop όπου και συνέχισα με κονσόλα τις εντολές που παρατέθηκαν στον Πίνακα 2.



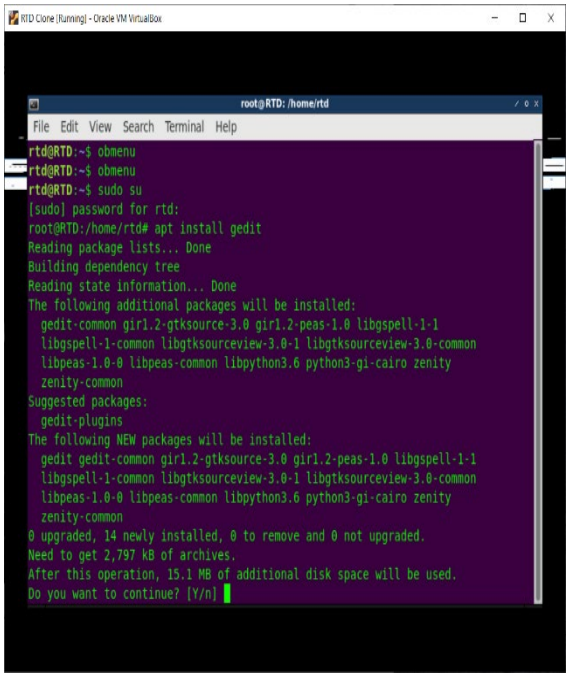

*Εικόνα 31 (Αρχικό Session – Είσοδος)*




*Εικόνα 32 (Terminal)*



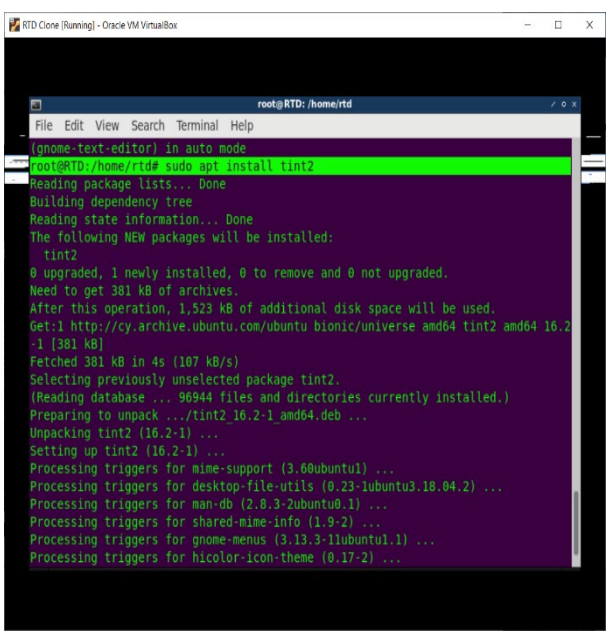

*Εικόνα 33( Εγκατάσταση obmenu)*

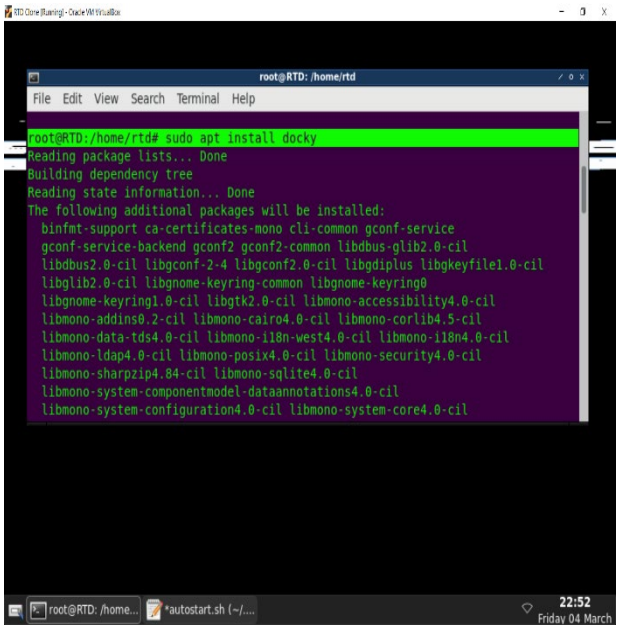


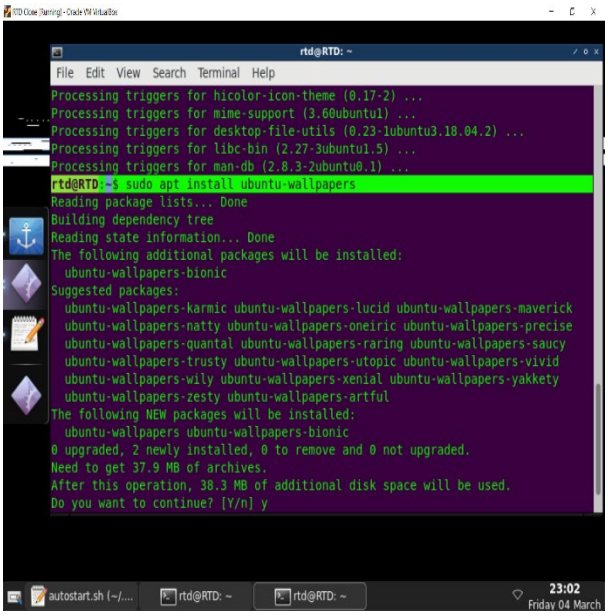

*Εικόνα 34 (Εγκατάσταση gedit)*

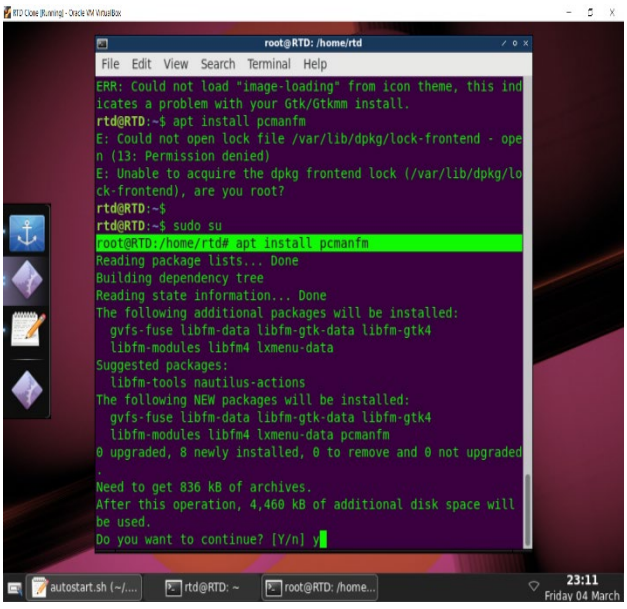
Στο Autostart.sh προστέθηκαν όλα τα πακέτα τα οποία χρειάζεται να τρέξουν αυτόματα με την εκκίνηση του Distribution. Αυτά φένονται στην εικόνα 35 όπως παρακάτω και ουσιαστικά αφορούν την εμφάνιση.


*Εικόνα 35 (Autostart.sh – Τρέξιμο των Παραπάνω με την Εκκίνηση του RTD)*

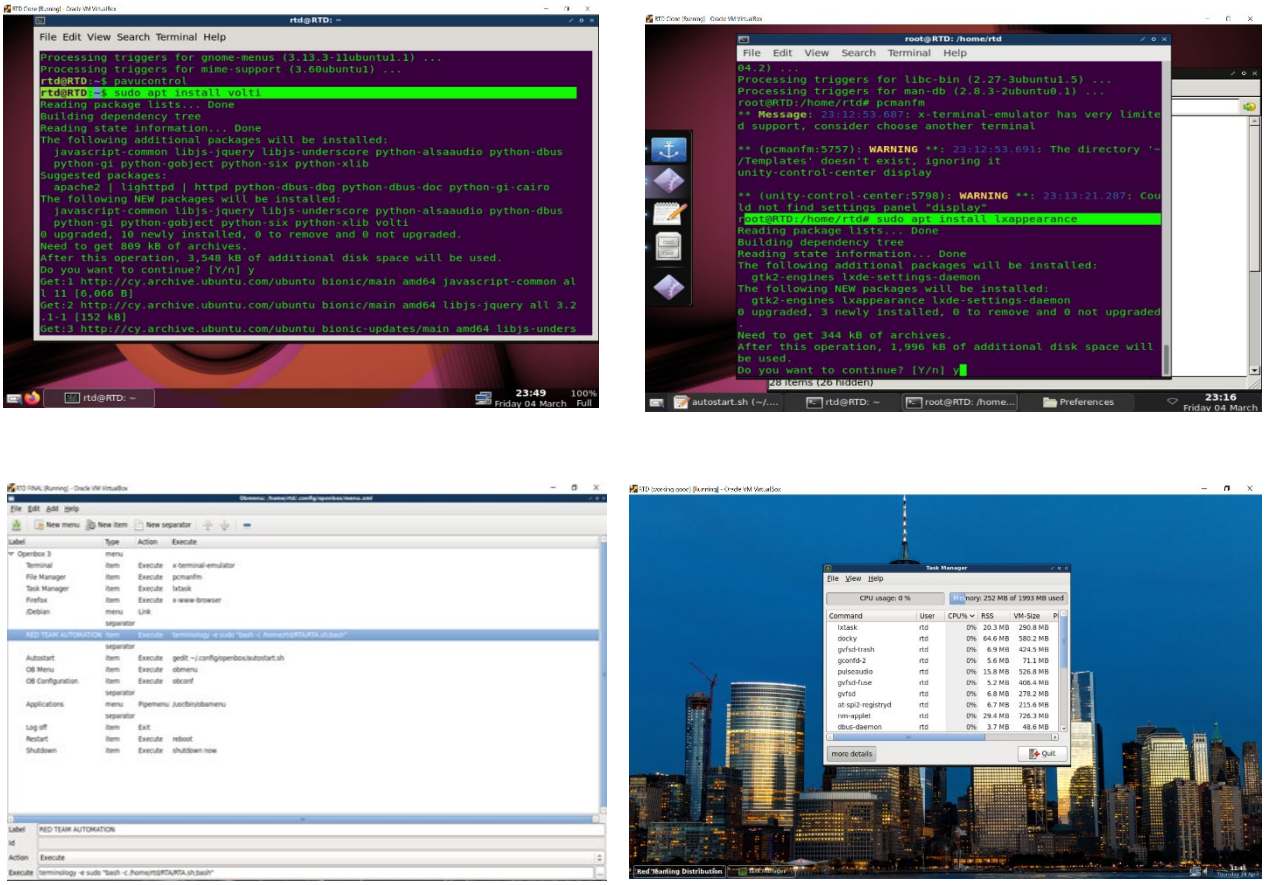



*Εικόνα 36 (Εγκατάσταση tint2, docky, ubuntu-wallpapers, pcmanfm)*

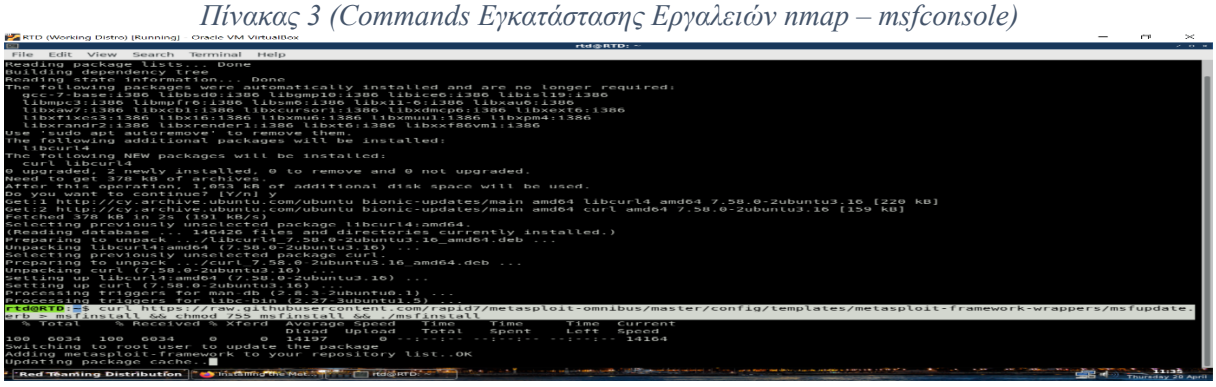


*Εικόνα 37 (Εγκατάσταση volti,lxappearance – Openbox Menu – Task Manager/Resources)*

Με την πραγματοοποίηση του Distribution έγινε η εγκατάσταση των εργαλείων nmap και msfconsole, μέσω κονσόλας, σύμφωνα με τον Πίνακα 3 όπως και τις εικόνες παρακάτω:

| Α/Α | ΟΝΟΜΑΣΙΑ ΕΡΓΑΛΕΙΟΥ | COMMANDS ΕΓΚΑΤΑΣΤΑΣΗΣ |
|---|---|---|
| 1 | Nmap | Sudo apt install nmap<br>Sudo apt install xlstproc (*.xml to *.html) |
| 2 | Msfconsole | Curl https://raw.githubusercontent.com/rapid7/metasploit-omnibus/master/config/templates/metasploit-framework-wrappers/msfupdate.erb>msfinstall && chmod 755 msfinstall && ./msfinstall |

*Πίνακας 3 (Commands Εγκατάστασης Εργαλειών nmap – msfconsole)*



*Εικόνα 38 (Εγκατάσταση msfconsole)*

Πλέον τα εργαλεία και πακέτα είναι εγκατεστημένα. Το RTD (Red Teaming Distribution) έχει υλοποιηθεί και είναι έτοιμο για χρήση. Έχει απομείνει η αυτοματοποίηση τους, η οποία πραγματοποιήθηκε με bash scripts και Python scripts. Η Bash [68] είναι μια γλώσσα προγραμματισμού Scripting και θεωρείτε απλή καθώς επιτρέπει την κλήση του συστήματος εντολών Unix, βοηθά στην αυτοματοποίηση εργασιών και εκτελεί νέες εντολές χρησιμοποιώντας κάποια εντολή ως στοιχείο της. Η Bash είναι μια Unix shell γλώσσα προγραμματισμού που γράφτηκε από τον Brian Fox ως αντικατάσταση του Bourne shell. Κυκλοφόρησε το 1998 και μέχρι στιγμής χρησιμοποιείται ως η προεπιλεγμένη για τα πλείστα shell.

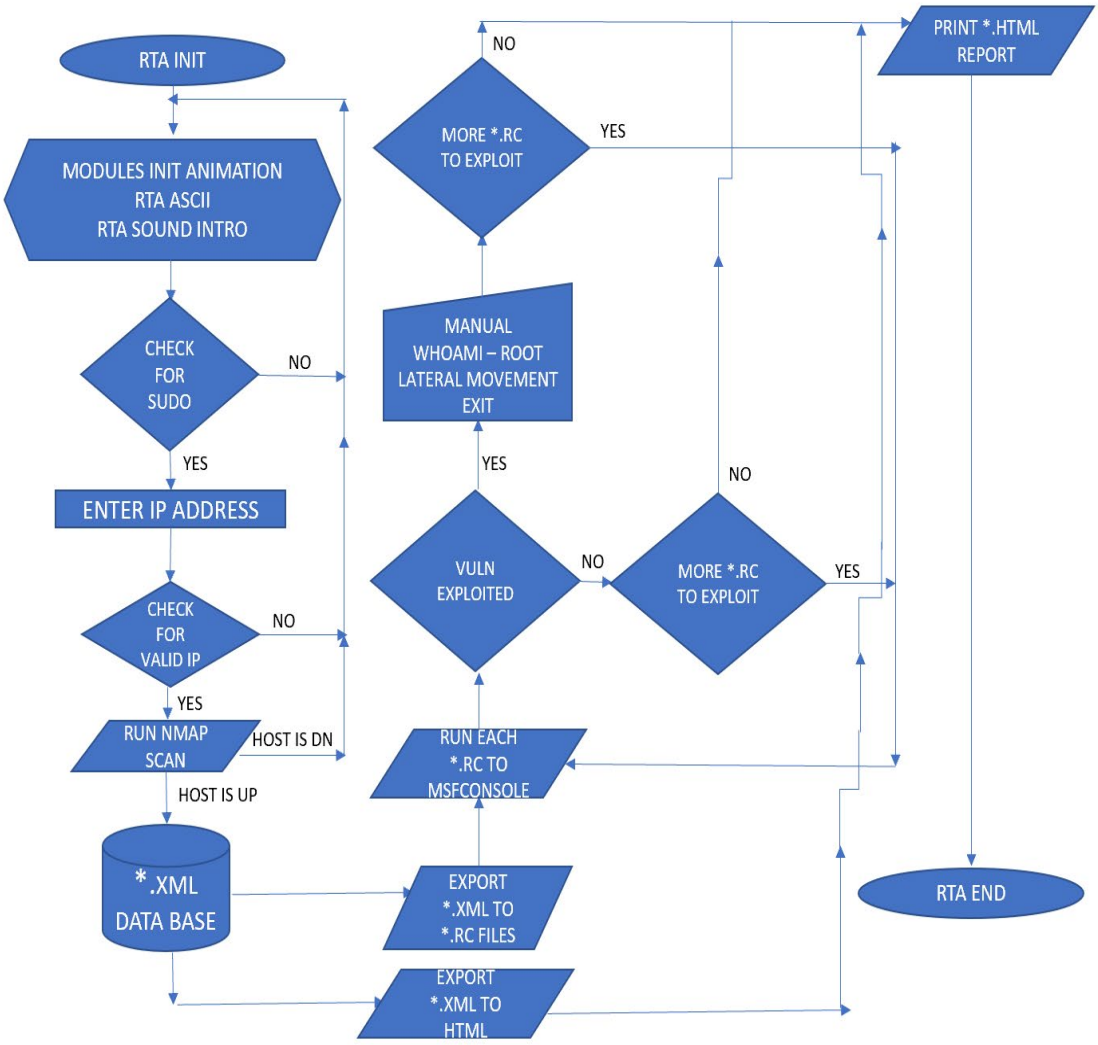
Παράλληλα η Python [69] είναι μια ερμηνευτική, αντικειμενοστραφής, υψηλού επιπέδου γλώσσα προγραμματισμού με δυναμική σημασιολογία. Οι ενσωματωμένες δομές δεδομένων υψηλού επιπέδου, σε συνδυασμό με τη δυναμική πληκτρολόγηση και τη δυναμική δέσμευση, την καθιστούν πολύ ελκυστική για ταχεία ανάπτυξη εφαρμογών, καθώς και για χρήση της ως γλώσσα Scripting ή διασύνδεσης υπαρχόντων στοιχείων μεταξύ τους. Είναι απλή, εύκολη στην εκμάθηση σύνταξη και δίνει έμφαση στην αναγνωσιμότητα. Η Python υποστηρίζει modules και πακέτα, τα οποία ενθαρρύνουν τη σπονδυλωτότητα του προγράμματος και την επαναχρησιμοποίηση κώδικα. Ο διερμηνέας Python και η εκτεταμένη βιβλιοθήκη της είναι ευρέως διαθέσιμα, χωρίς χρέωση για όλες τις μεγάλες πλατφόρμες και μπορούν να διανεμηθούν ελεύθερα.

Και οι δύο αυτές γλώσσες προγραμματισμού είναι οι κατεξοχήν χρησιμοποιούμενες και είναι κυρίως γνωστές ως οι καλύτερες γλώσσες προγραμματισμού των μηχανισμών αυτοματισμού. Όλες αυτές οι γλώσσες έχουν πλεονεκτήματα και μειονεκτήματα, όμως η συνδυασμένη χρήση τους μπορεί να λύσει πολλά προβλήματα, όπως έγινε και στη συγκεκριμένη διατριβή. Ο κύριος κορμός είναι γραμμένος σε Bash script, ο οποίος ανάλογα με την ανάγκη καλεί τα προ εγκατεστημένα εργαλεία – python scripts με συγκεκριμένα -flags, τα οποία εξάγουν μετά τους ελέγχους τους, τα δικά τους αποτελέσματα υπό τύπο report, όπου εκτελεί η bash για να τα υποδείξει και ουσιαστικά για να ολοκληρώσει τη διαδικασία. **Όλος ο κώδικας που χρησιμοποιήθηκε για την εκπλήρωση της Μεταπτυχιακής Διατριβής βρίσκεται στο Παράρτημα Α.**

## 4.4 Flowchart

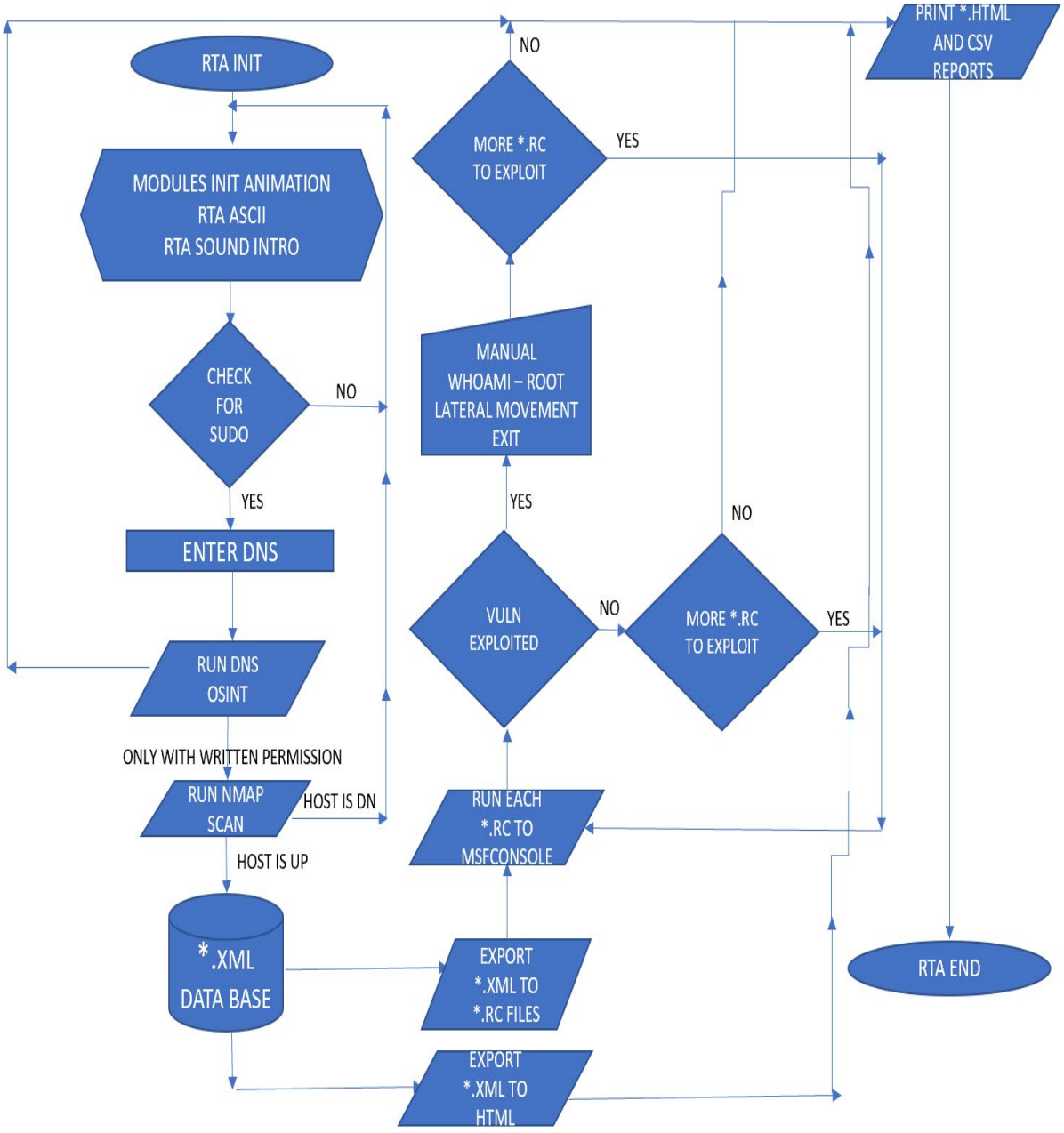
Ένα διάγραμμα ροής [70] είναι μια εικόνα, στην οποία με διάφορα σχήματα αναπαρίστανται ξεχωριστά βήματα μιας διαδικασίας με διαδοχική σειρά. Είναι ένα γενικό εργαλείο που μπορεί να προσαρμοστεί σε μια μεγάλη ποικιλία σκοπών και μπορεί να χρησιμοποιηθεί για να περιγράψει διάφορες διαδικασίες, όπως μια διαδικασία παραγωγής ή σχέδιο έργου. Στον προγραμματισμό μπορεί να αναπαραστήσει τη λογική σειρά των ενεργειών/ υπολογισμών ως διαγραμματική αναπαράσταση ενός αλγορίθμου. Ένα διάγραμμα ροής μπορεί να είναι χρήσιμο τόσο για τη σύνταξη προγραμμάτων όσο και για την επεξήγηση του προγράμματος σε άλλους.

Για την υλοποίηση του RTA (Red Teaming Automation) ακολουθήθηκαν τα παρακάτω λογικά διαγράμματα ροής. Για την επίθεση με IP η εικόνα 39 ενώ για το DNS η εικόνα 40.




*Εικόνα 39 (IP Mode Flowchart)*

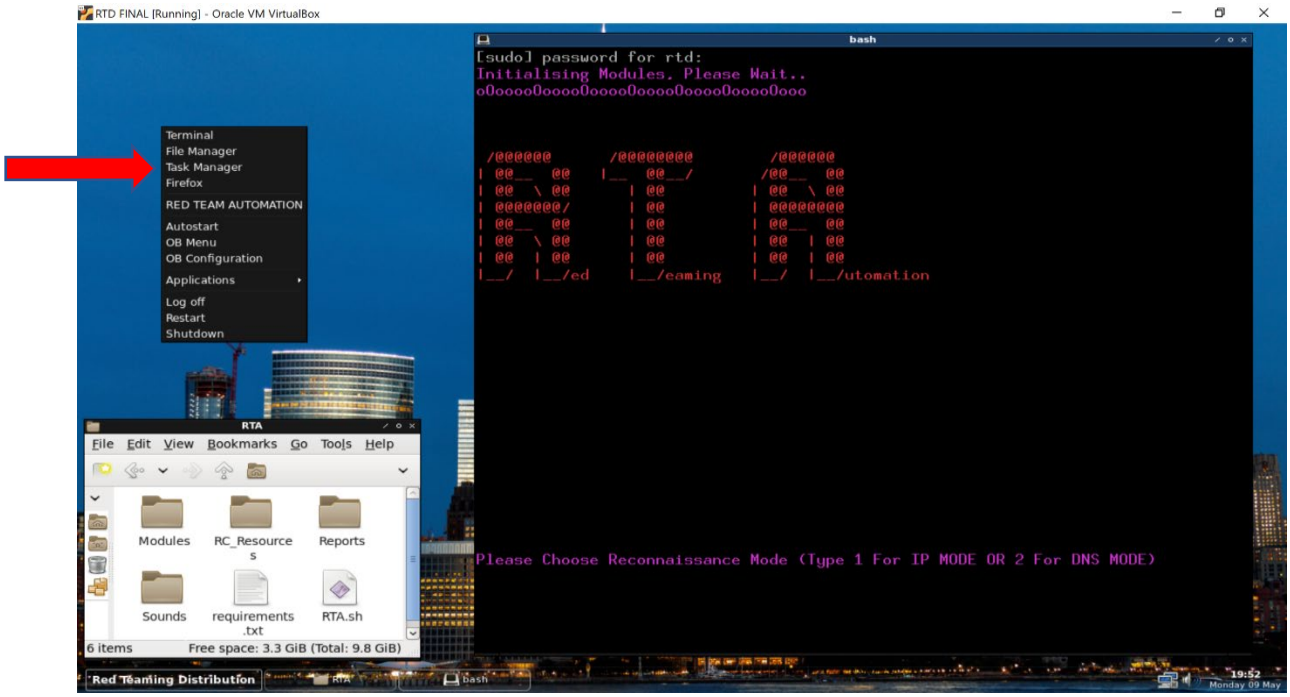
Κρίνεται σκόπιμο να επισημανθεί ότι για νομικούς λόγους η έκδοση με την DNS περιορίστηκε μόνο σε OSINT (Open Source Intelligence), καθώς χωρίς την γραπτή έγκριση – συγκατάθεση κάποιου οργανισμού δεν επιτρέπεται η επίθεση. Έστω όμως και με την παραπάνω παραδοχή, τα αποτελέσματα που λήφθηκαν κρίνονται ως πολύ ικανοποιητικά.




*Εικόνα 40 (DNS Mode Flowchart)*

## 4.5 RTA (Red Teaming Automation)

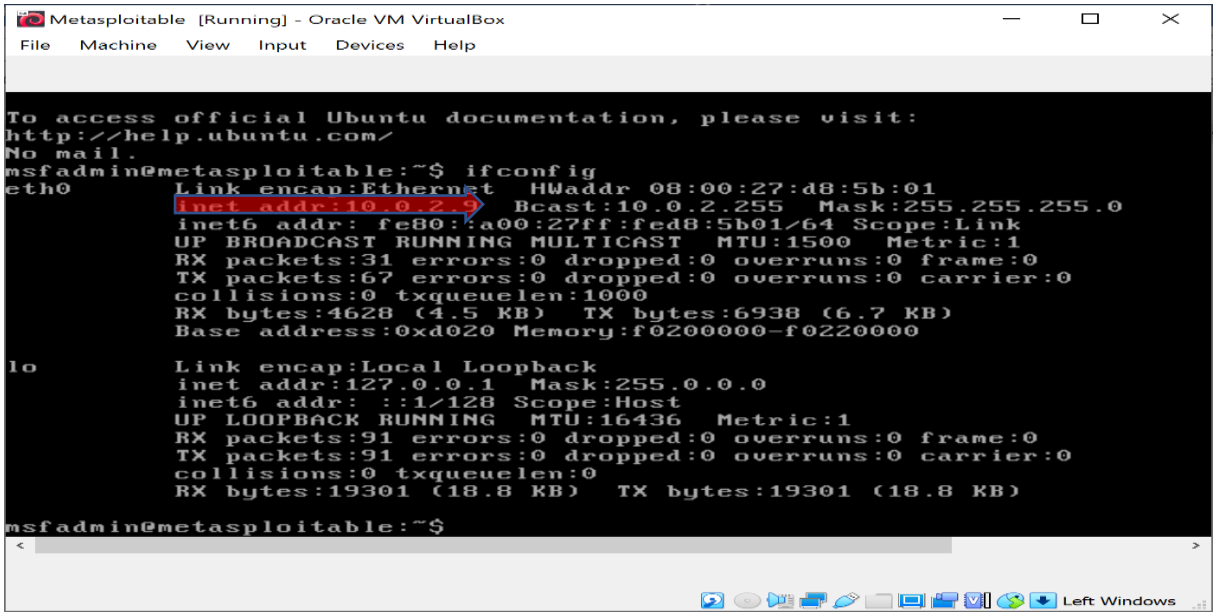
Η εικόνα του RTA διαφαίνεται στην επόμενη Εικόνα 41, καθώς επίσης ο τρόπο που επιλέγεται. Ένα απλό δεξί κλίκ είναι αρκετό για να εμφανιστεί το RED TEAM AUTOMATION για να μπορεί να τρέξει, αφού επιλεχθεί αφού περιηγηθεί ο κέρσορας και πατηθεί το αριστερό κλίκ.




*Εικόνα 41 (Αρχική Εικόνα RTA)*

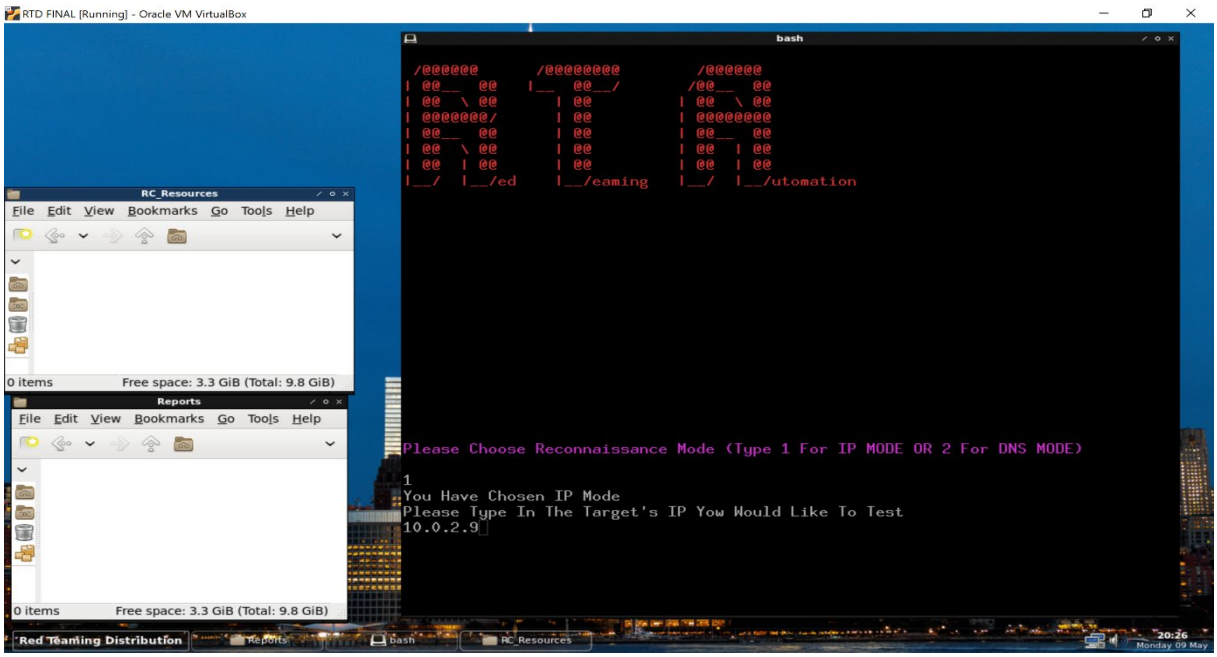
Ότι χρειάζεται το RTA για να λειτουργήσει απρόσκοπτα, βρίσκεται εγκατεστημένο σαν εργαλείο στο RTD (nmap, msfconsole) ενώ τα python scripts βρίσκονται στο directory ~/home/rtd/RTA/Modules. Εντός του directory ~/home/rtd/RTA/RC_Resources βρίσκονται τα αρχεία τα οποία δημιουργούνται μέσω ενός python script (export_vuln_as_rc) εξάγοντας την απαραίτητη πληροφορία από το *.xml αρχείο που δημιουργείται με την ολοκλήρωση του nmap. Τρέχουν όμως μέσα από το bash και ουσιαστικά εισάγουν την απαραίτητη πληροφορία για το msfconsole.

Παράλληλα με το RTD τρέχει και μια vulnerable μηχανή metasploitable, η οποία βρίσκεται στο ίδιο δίκτυο. Η εν λόγω μηχανή διαθέτει αρκετές ευπάθειες που θα προσπαθήσουμε να εκμεταλλευτούμε επιτυγχάνοντας τα exploits μας. Ουσιαστικά δημιουργήθηκε ένα μικρό cyber range με σκοπό την υλοποίηση. Στην επόμενη Εικόνα 42 μπορούμε να δούμε την IP της εν λόγω μηχανής.



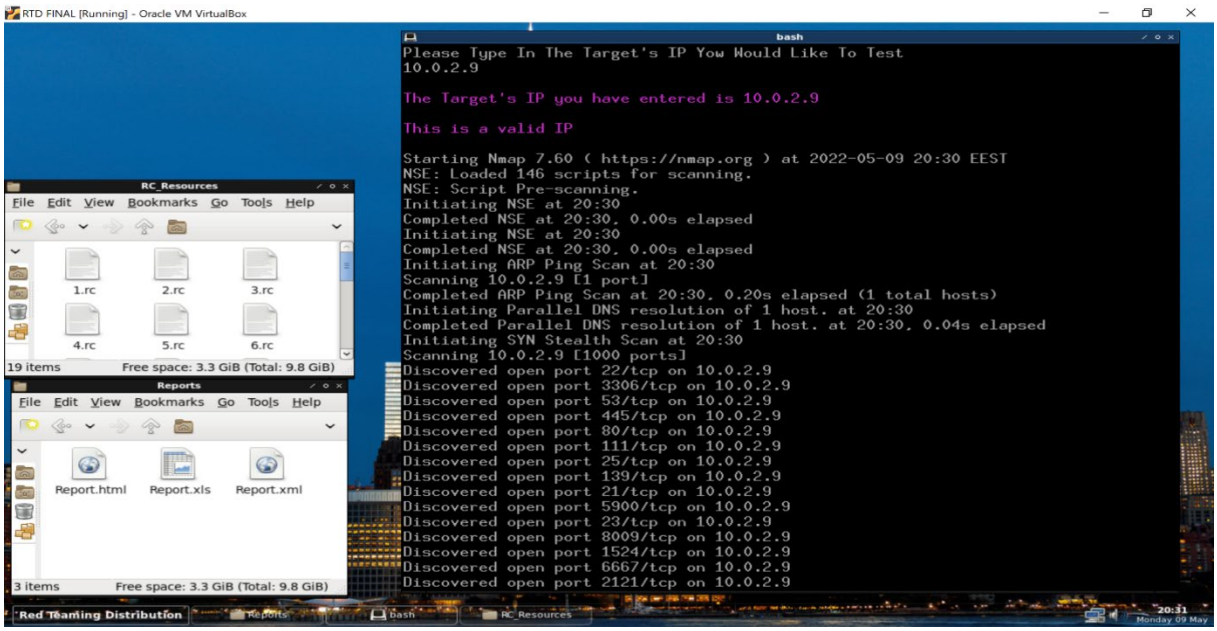

*Εικόνα 42 (IP Address Ευπαθούς Μηχανής Metasploitable)*

Με την εισαγωγή του Mode 1, το RTA ζητά την εισαγωγή της IP της metasploitable μηχανής. Μπορούμε να δούμε στην εικόνα 43 τη διαδικασία.



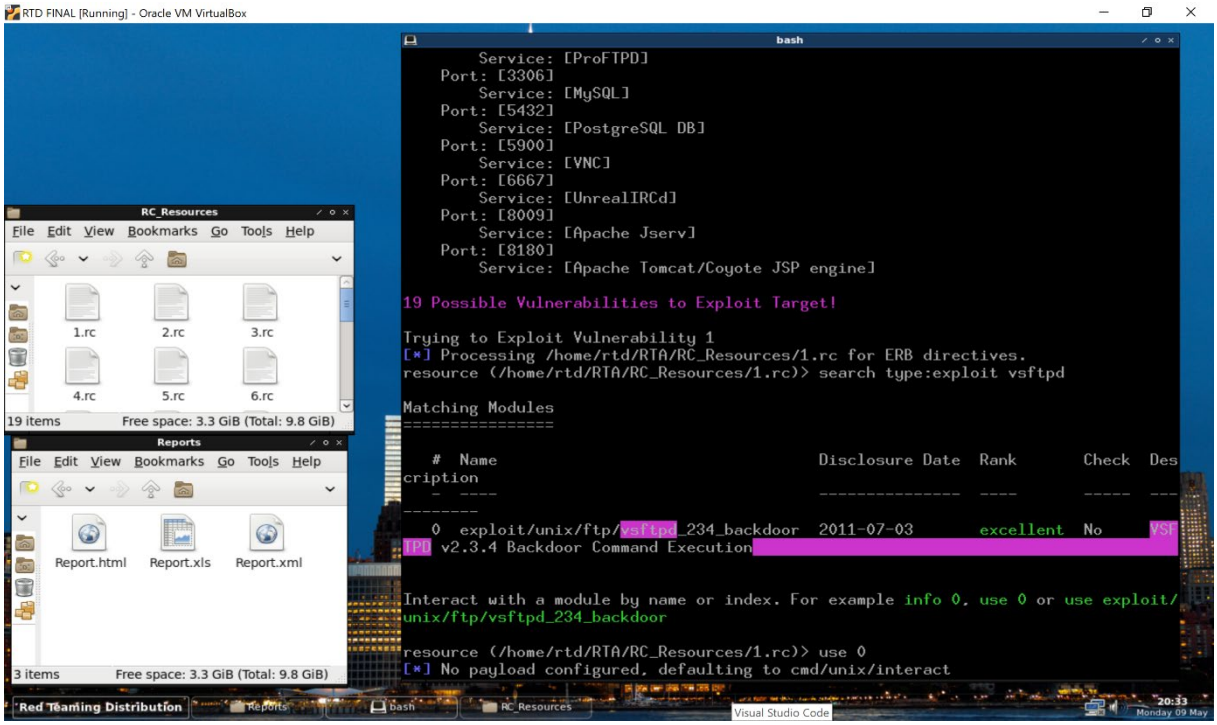

*Εικόνα 43 (Επιλογή IP Mode και Είσοδος IP Address)*

Αφού τρέξει το λογισμικό λαμβάνουμε τις παρακάτω πληροφορίες οι οποίες φαίνονται στις εικόνες 44 εώς 46. Στην εικόνα 44 μπορούμε να δούμε ότι έγινε ο έλεγχος και ότι η IP είναι έγκυρη, ενώ αμέσως αναλαμβάνει το nmap. Η ταχύτητα που γίνονται όλα είναι τόσο μεγάλη που μέχρι να αποκτήσω το screenshot το RTA είχε είδη τελειώσει με το scanner του nmap, έλαβα όλα τα report, ενώ είχε τρέξει και το script export_vuln_as_rc.py, αφού βλέπουμε τα αποτελέσματα αριθμημένα και με κατάληξη *.rc.



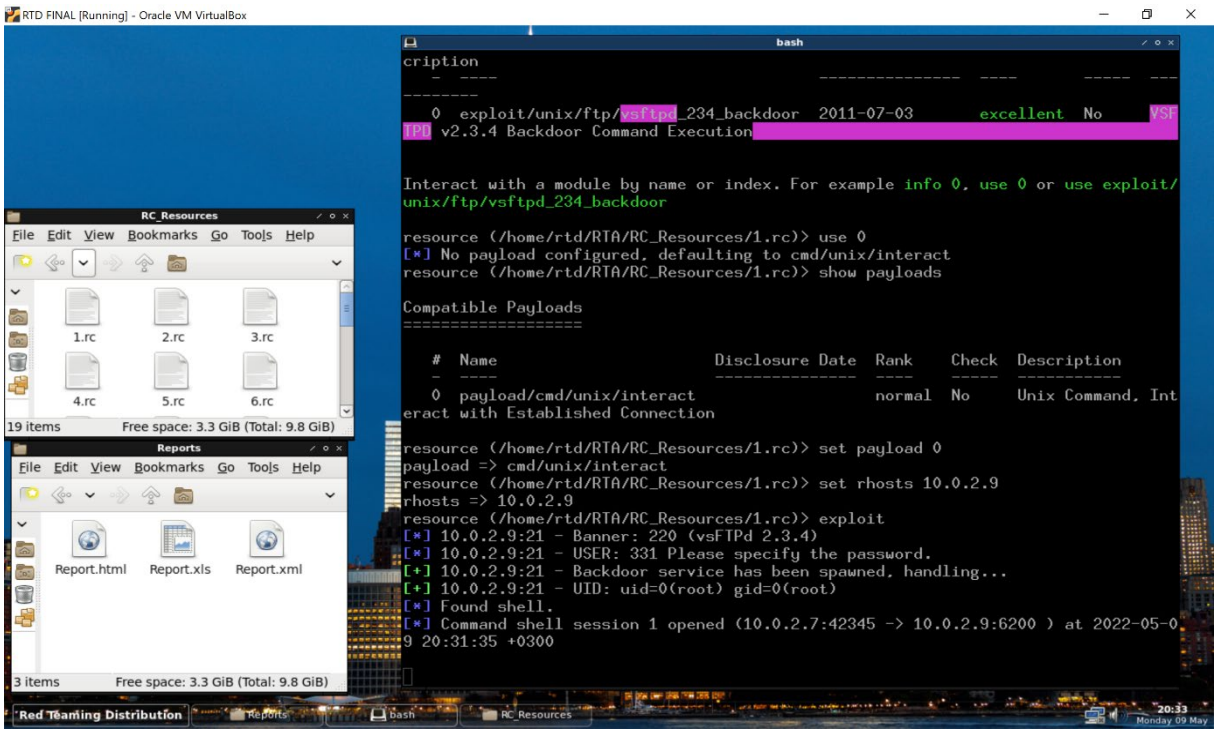

*Εικόνα 44 (Ελεγχος Εγκυρότητας IP, Εκκίνηση nmap, nmap Reports, Δημιουργία Αρχείων*.rc )*

Σε αυτή τη φάση έχουν ανευρεθεί και αριθμηθεί όλες οι ευπάθειες και ξεκινά η διαδικασία των αρχείων *.rc. Μπορούμε να διακρίνουμε όλα τα προαναφερθέντα δεδομένα, καθώς επίσης και την εύρεση της ευπάθειας στην Εικόνα 45.



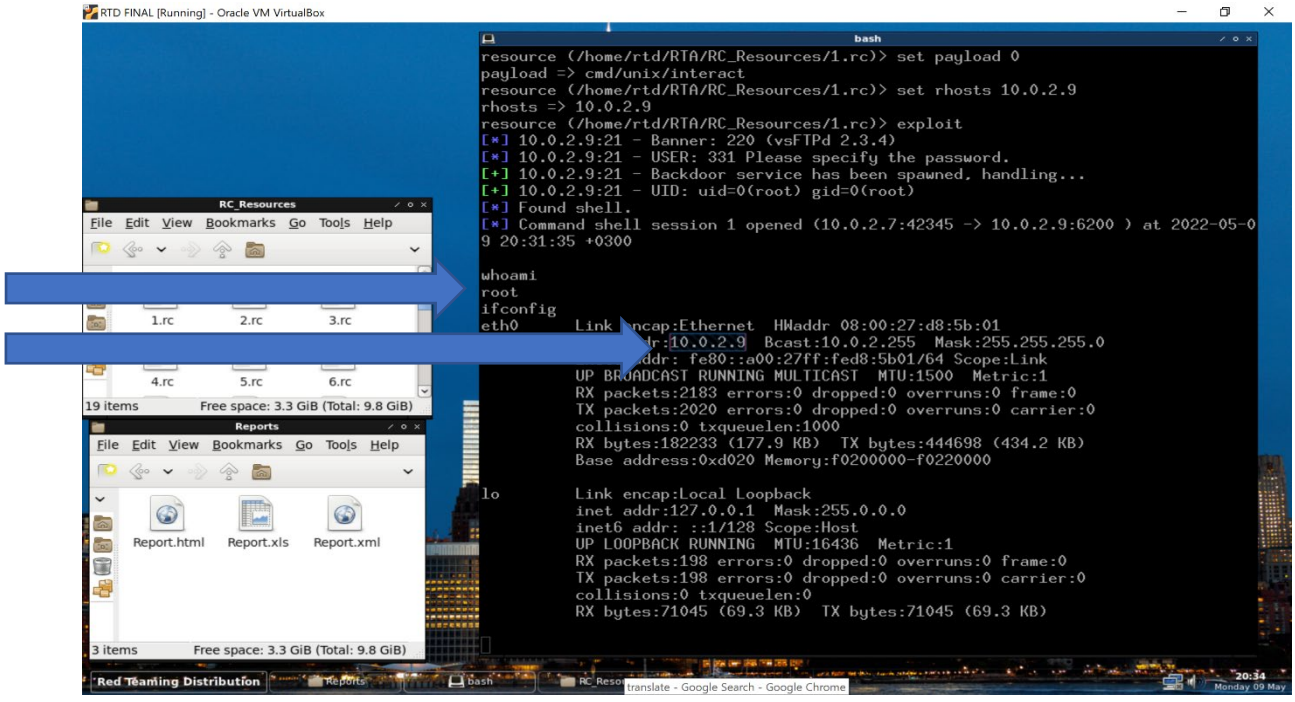

*Εικόνα 45 (Αριθμός Ευπαθειών που Ανευρέθηκαν – Εκκίνηση του msfconsole)*

Εν συνεχεία στην Εικόνα 46, μπορούμε να διακρίνουμε ότι το exploit πέτυχε, ενώ έχουμε ανοιχτό session. Από εδώ και εμπρός ο επιτιθέμενος Red Teamer, αφού έλαβε root privileges πλέον μπορεί να ενεργήσει όπως επιθυμεί.



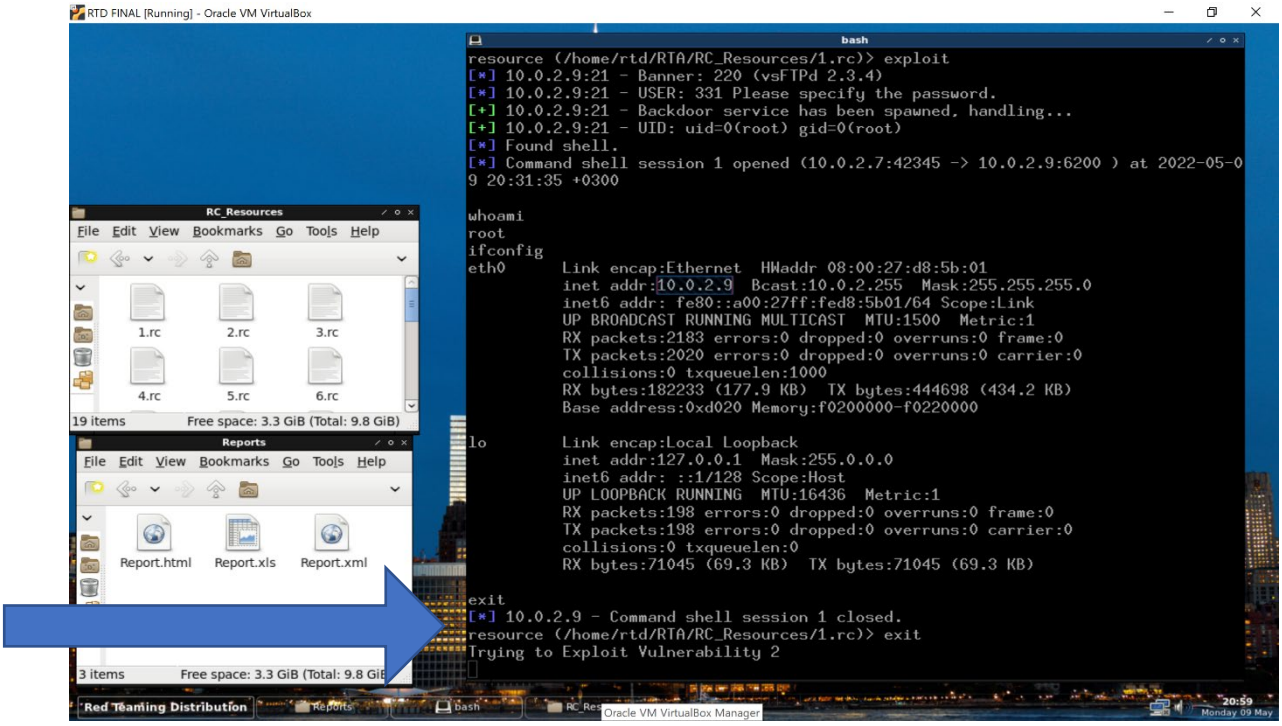

*Εικόνα 46 (Πετυχημένο Exploit)*

Η επόμενη Εικόνα 47 επιβεβαιώνει του λόγου το αληθές. Είναι πλέον ξεκάθαρο ότι έχουμε αποκτήσει απομακρυσμένη και μη εξουσιοδοτημένη πρόσβαση στη μηχανή.



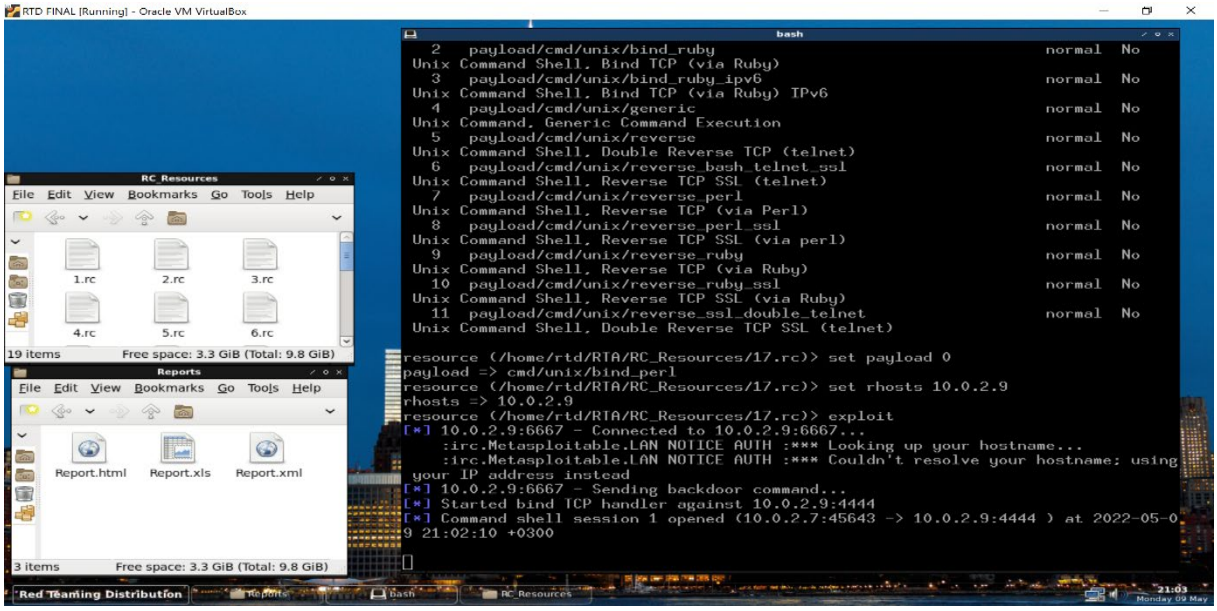

*Εικόνα 47 (Επιβεβαίωση του Exploit)*

Το RTA για να προχωρήσει στη διερεύνηση χρειάζεται ο χρήστης να λάβει γνώση ότι η ευπάθεια έχει εκμεταλλευτεί επιτυχώς και για να προχωρήσει, πρέπει ο ίδιος να πληκτρολογήσει exit ή ctrl + c για να διακοπεί το session, όπως μπορούμε να διακρίνουμε στην Εικόνα 48.



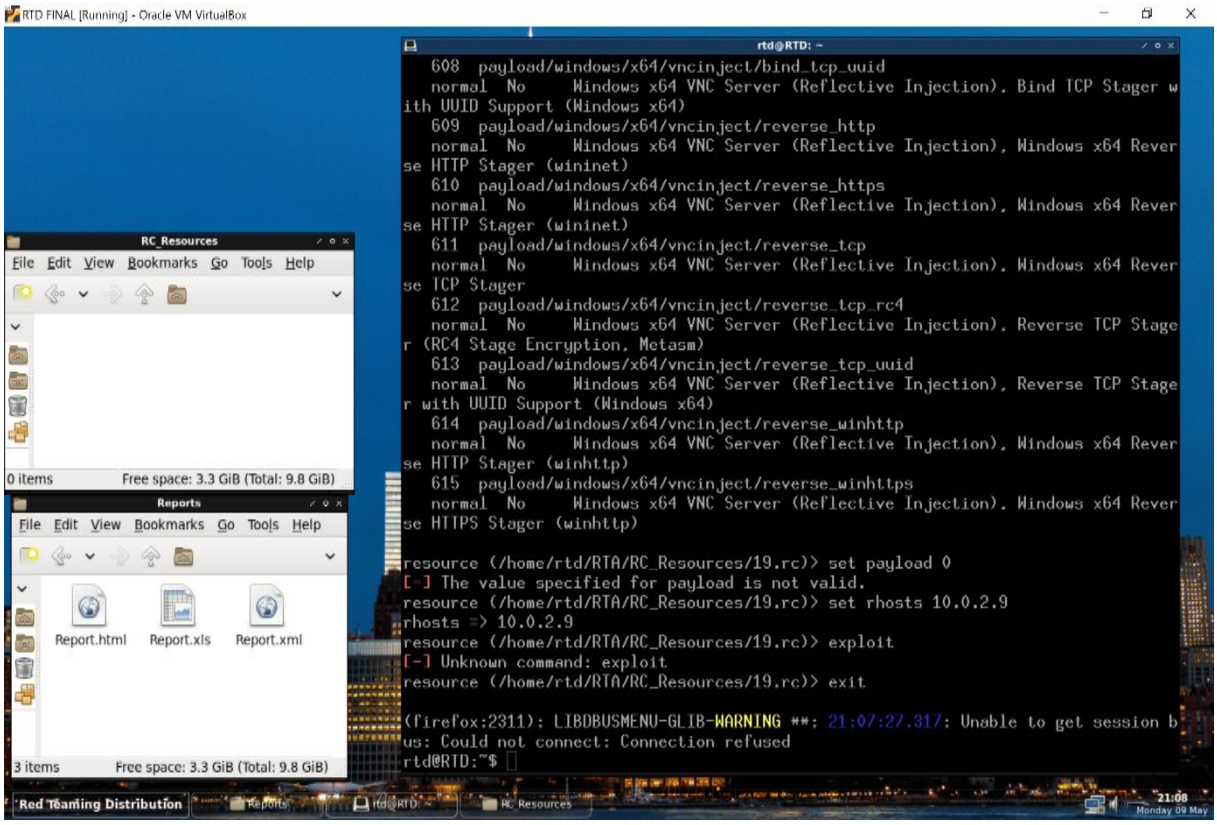

*Εικόνα 48 (Έξοδος από το Session και Συνέχεια Προσπάθειας Εκμετάλλευσης Ανευρεθέντων Ευπαθειών )*

Προχωρώντας λαμβάνουμε ακόμα μια επιτυχημένη επίθεση. Αυτό φαίνεται στην Εικόνα 49. Το όλο θέμα είναι να επιτύχουμε έστω σε μια επίθεση ούτως ώστε να μπορούμε να ενεργήσουμε ως επιτιθέμενοι.

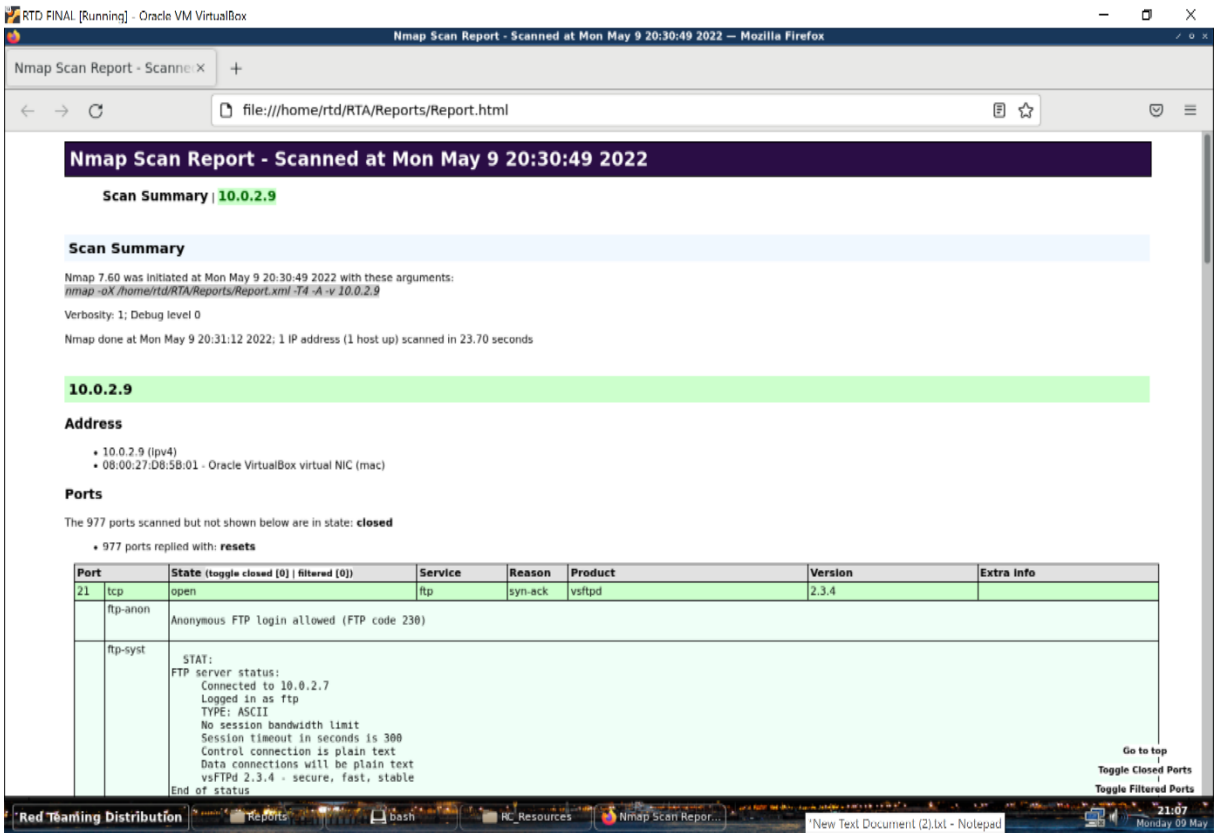



*Εικόνα 49 (2η Πετυχημένη Επίθεση)*

Ενώ προχωρά το RTA φτάνει στο σημείο που δεν υπάρχουν άλλες ευπάθειες προς εκμετάλλευση. Έτσι προχωρά στη διαγραφή των αρχείων *.rc, ενώ ταυτόχρονα τρέχει το report για έλεγχο. Αυτά φαίνονται στις εικόνες 50 και 51.




*Εικόνα 50 (Τέλος των Vulnerabilities, διαγραφή των αρχείων *.rc)*




*Εικόνα 51 (HTML Report)*

# Κεφάλαιο 5
# Αποτελέσματα

## 5.1 RTD και RTA

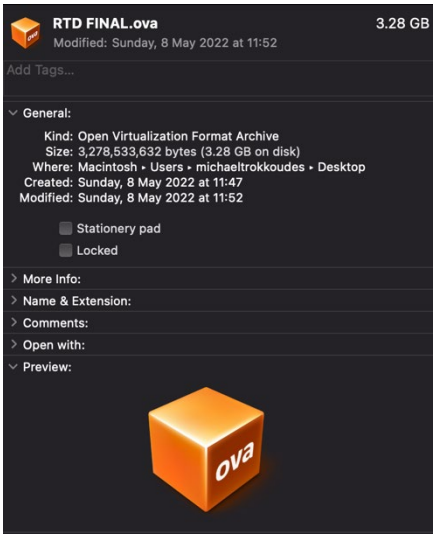
Το RTD και το RTA έχουν αποδείξει ότι σε περιβάλλοντα Cyber Range, ο λόγος για τον οποίο δημιουργήθηκαν και εφαρμόστηκαν έχει εκπληρωθεί. Ο πρωταρχικός στόχος του RTA, η αναγνώριση, η εύρεση, η εκμετάλλευση και η εξαγωγή αναφοράς αυτοματοποιημένα έχει επιτευχθεί με επιτυχία. Δύο στις δεκαεννέα ευπάθειες έτυχαν αυτοματοποιημένης εκμετάλλευσης σε χρόνο που δεν δύναται να ακολουθήσει το ανθρώπινο μάτι, συνεπώς και μπορεί να θεωρηθεί ακραίο το σενάριο, κάποιος να μπορεί να ενεργήσει σε αυτές τις ταχύτητες.

## 5.2 Τεχνικά Χαρακτηριστικά και Δυνατότητες

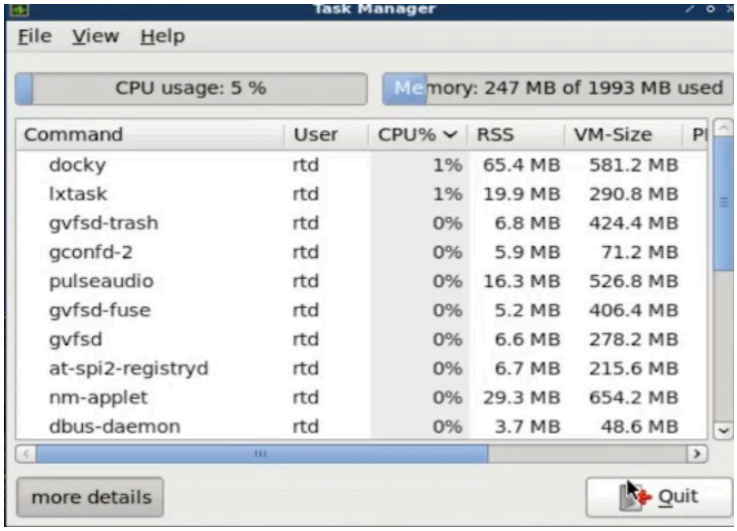
Το ελαφρύ από όλες τις απόψεις RTD, απέδειξε ότι μπορεί να λειτουργήσει με επιτυχία, με ταχύτητα και με σταθερότητα σε περιπτώσεις όπου υπάρχουν ελάχιστες απαιτήσεις συστημάτων, χωρίς δυσκολίες, bugs ή glitches. Στην Εικόνα 52 μπορούμε να παρατηρήσουμε τα χαρακτηριστικά του αφού εξήχθη από το λογισμικό Virtual Box, ενώ στην Εικόνα 53 διακρίνονται οι λιγοστοί πόροι που χρειάζεται το λειτουργικό αυτό σύστημα για να τρέξει. Το RTA παράλληλα με τον ελαφρύ του κώδικα κατάφερε χωρίς ιδιαίτερη δυσκολία να εξάγει τα επιθυμητά αποτελέσματα, αφού μέσα σε χρόνο 5 λεπτών και 09 δευτερολέπτων κατάφερε να εκτελέσει όλες τις ενέργειες που προβλέπονται κατά το Penetration Testing, ενώ αντίστοιχα χωρίς τους αυτοματισμούς ο χρόνος που χρειάστηκα για να πράξω το ίδιο ήταν λίγο περισσότερο από διπλάσιος. Στο RTA χρησιμοποιήθηκαν εν τέλη τα παρακάτω εργαλεία – script:

| Α/Α | ΕΡΓΑΛΕΙΟ - SCRIPT | ΧΡΗΣΙΜΟΠΟΙΗΣΗ |
|---|---|---|
| 1 | RTA.sh (Bash Script) | Βασικός Κορμός |
| 2 | Nmap [29] | Port - Vulnerability Scanner |
| 3 | Nmap-converter Script.py [71] | Μετατροπή *.xml σε *.xls |
| 4 | Export_Vuln_as_rc.py | Μετατροπή *.xml σε αριθμό *.rc |
| 5 | Msfconsole [30] | Exploitation με χρήση των *.rc αρχείων |
| 6 | DNS (Attack Surface Mapper) [72] | Osint (Συμπεριλαμβανομένου όλων των Modules Buckethunter.py, Censys.py, Dnsdumpster.py, Hosthunter.py, Hunterio.py, Shodan.py, Subbrute.py, Subhunter.py, Urlscanio.py, Webscraper.py, Weleakinfo.py, Whois_collector.py |

*Πίνακας 4 (Εργαλεία και Script όπου Χρησιμοποιήθηκαν)*




*Εικόνα 52 (Χαρακτηριστικά RTD.ova)*




*Εικόνα 53 (Μειωμένοι Χρησιμοποιούμενοι Πόροι)*

Επίσης κρίνεται σκόπιμο να αναφερθεί ότι λόγω της απλότητας του συστήματος, εκτιμάται ότι πιθανό η χρήση από προσωπικό που να μην είναι εξοικειωμένο δεν είναι καθόλου απαγορευτική καθ' ότι δεν χρειάζεται να υπάρχει κάποια ιδιαίτερη γνώση επί του θέματος, πλην των βασικών, ενώ με λιγοστή καθοδήγηση, υπάρχει η δυνατότητα να γίνει χρήση του πολύ εύκολα. Στον παρακάτω Πίνακα 4 φαίνονται τα αποτελέσματα των πυλώνων που στοιχειοθέτησαν την ανάπτυξη του εν λόγω Λειτουργικού και Λογισμικού.

| Α/Α | ΠΕΡΙΓΡΑΦΗ ΧΑΡΑΚΤΗΡΙΣΤΙΚΟΥ | ΑΥΤΟΜΑΤΟΠΟΙΗΜΕΝΟ ΑΠΟΤΕΛΕΣΜΑ | ΧΕΙΡΟΚΙΝΗΤΟ ΑΠΟΤΕΛΕΣΜΑ |
|---|---|---|---|
| 1 | Ταχύτητα (Χρόνος) | Ταχύτατο (05:09) | Γρήγορο (11:02) |
| 2 | Ευκολία Χρήσης | Πάρα πολύ εύκολο (Χρειάστηκε μόνο την IP) | Εύκολο ( Χρειάστηκε να επαναλάβω διαδικασίες, ενώ ουσιαστικά καθυστερούσα λόγω πληκτρολόγησης) |
| 3 | Αποτελεσματικότητα | 2/19 (Μπόρεσε να εκμεταλλευτεί ότι υπήρχε στο msfconsole) | 3/19 (Μπόρεσα να εκμεταλλευτώ επιπλέον πηγές μέσω του διαδικτύου και scripts, πέραν του msfconsole) |
| 4 | Θα μπορεί να χρησιμοποιηθεί και από άλλους με λίγη πείρα στον τομέα; | Ναι με ελάχιστη καθοδήγηση | Όχι, χρειάζεται εκπαίδευση |

*Πίνακας 5 (Αποτελέσματα Κύριων Πυλώνων)*

# Κεφάλαιο 6
# Επίλογος

## 6.1 Γενικός Επίλογος

Σκοπός της παρούσης Μεταπτυχιακής Διατριβής ήταν η διεξοδική μελέτη, υπαρχόντων Ped Team attack scripts και tools με σκοπό την υλοποίηση τους με την αυτοματοποίηση των ενεργειών που απαιτούνται, για την αναγνώριση και εκμετάλλευση ευπαθειών, πληροφοριακών συστημάτων του τύπου VM σε περιβάλλοντα Cyber Range, για κέρδος πολύτιμου χρόνου, σκέψης και εν τέλη της επίτευξης του τελικού στόχου.

Έγινε μια εκτενής βιβλιογραφική επισκόπηση με αναφορές στο Red Teaming, στρατηγικές και τύπους Penetration Testing, τη γενική μεθοδολογία που ακολουθείται κατά τα προαναφερθέντα, τους τύπους των επιθέσεων, τα υπάρχοντα εργαλεία που χρησιμοποιούνται από την πλειοψηφία στο χώρο του Red Teaming, τα είδη των αυτοματισμών και τα υπάρχοντα αυτοματοποιημένα συστήματα που υπάρχουν.

Επιλέχθηκε το σπειροειδές μοντέλο Spiral, καθώς συνδυάζει την ιδέα της επαναληπτικής ανάπτυξης (Iterative Model) καθώς επιτρέπει τις απελευθερώσεις του παραχθέντος λογισμικού και τη σταδιακή βελτίωση του σε κάθε επανάληψη γύρω από τη σπείρα, Έχοντας ως βάση τη πειραματική δοκιμή η εν λόγω ποσοτική έρευνα στηρίχθηκε στη συλλογή και ανάλυση των εξαχθέντων δεδομένων.

Με την ολοκλήρωση της παρούσας διατριβής, μπορώ να εξάγω ως γενικό συμπέρασμα την επιτυχία της αρχικής ιδέας, καθώς ολοκληρώθηκε και λειτούργησε με τον τρόπο που θα έπρεπε.

## 6.2 Ερευνητικά Ερωτήματα

Τα ερευνητικά ερωτήματα που είχαν τεθεί στην αρχή της παρούσας διατριβής έχουν απαντηθεί πλήρως. Υπάρχουν αφθονία επιλογών σε εργαλεία και Script, που δύναται να χρησιμοποιηθούν σε περιβάλλοντα Cyber Range, καθώς επίσης σε ρεαλιστικά και

πραγματικά περιβάλλοντα. Υπάρχει εκτενής κοινότητα η οποία ενδιαφέρεται να ενημερώσει, να εκπαιδεύσει και να ανταλλάξει λογισμικό, ούτως ώστε να πετύχουν οι Red Teamers τον τελικό στόχο τους. Αυτή όμως η θετική πλευρά, δυστυχώς δεν περιορίζεται μόνο στους Ethical Hackers αλλά σε όλο το φάσμα των hackers και ιδίως αυτών που επιθυμούν να ζημιώσουν άλλους για ίδιον όφελος.

Τα εργαλεία και Script τα οποία χρησιμοποιήθηκαν, αυτοματοποιήθηκαν με τη χρήση Bash και Python. Με την μεθοδολογία που ακολουθήθηκε, του μοντέλου Spiral, δύναται να αναβαθμιστούν, να αυξηθούν και να προστεθούν ακόμα περισσότερα , ούτως ώστε να δημιουργηθεί ένα ακόμα καλύτερο τελικό αποτέλεσμα. Όπως διαφάνηκε τα επιλεχθέντα εργαλεία και Script απέδωσαν με επιτυχία, για το σκοπό που χρησιμοποιήθηκαν, να λειτουργήσουν αυθαίρετα και αυτοματοποιημένα, ακολουθώντας τη λογική σειρά ενεργειών. Με την αυτοματοποίηση εξοικονομήθηκε χρόνος, πράγμα που σημαίνει ότι υπάρχει προοπτική, καθώς όποιος διαθέτει χρόνο στη σημερινή κοινωνία θεωρείται ιδιαίτερα τυχερός και ευνοημένος, αν δεν λάβουμε υπόψη μόνο τη γενική ρήση ότι ο χρόνος ισοδυναμεί με χρήμα.

Με την μελέτη της υπάρχουσας βιβλιογραφίας της επιστημονικής κοινότητας, ενημερώθηκα για τα υφιστάμενα αυτοματοποιημένα εργαλεία. Υπάρχουν, όπως παρουσίασα στο Κεφάλαιο 2, πλην όμως, θεωρώ ότι εξίσου αυτά όπως και το παρόν, δύναται να μετεξελιχθούν και να αποδώσουν ακόμα καλύτερα, με κύριο γνώμονα την απλότητα, κάτι το οποίο δεν παρατήρησα στα άλλα εργαλεία, τα οποία ήταν πιο επικεντρωμένα στην εκτέλεση του σκοπού, αλλά όχι τόσο στην ευχρηστία του χρήστη.

## 6.3 Προσωπικός Αντίκτυπος

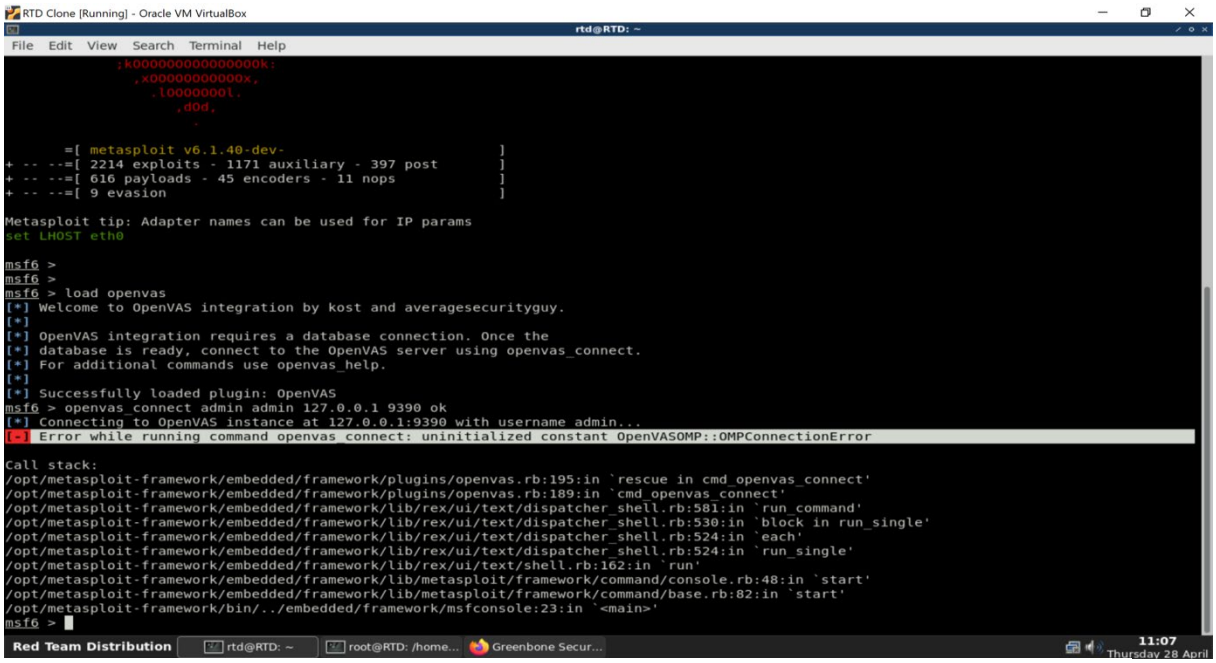
Προσωπική εκτίμηση μου είναι ότι, μετά από την εκπλήρωση της Μεταπτυχιακής μου Διατριβής έχω λάβει ιδιαίτερες γνώσεις και δεξιότητες. Ήρθα σε επαφή με δύο γλώσσες προγραμματισμού οι οποίες χρησιμοποιούνται κατά κόρον (bash, python) στον τομέα της ασφάλειας των υπολογιστών και δικτύων καθώς έχω επίσης εντρυφήσει σε ότι αφορά λειτουργικά συστήματα Linux και πληθώρας εργαλείων.

## 6.4 Περιορισμοί

Κατά την εκπόνηση της διατριβής, είχα βρεθεί αντιμέτωπος με πολλαπλά εμπόδια, είτε λόγω δικής μου απειρίας, είτε αδυναμίας του λογισμικού – εργαλείων να εκτελέσουν τα ζητούμενα. Μια περίπτωση στην οποία σπαταλήθηκε αρκετός από τον συνολικό χρόνο, ήταν η εγκατάσταση του Openvas (2.9.3) , γνωστού λογισμικού ανοιχτού κώδικα για εύρεση ευπαθειών. Προφανώς λόγω κάποιων προβλημάτων στη σύνδεση του με το msfconsole όπως αυτό είχε αρχικά επινοηθεί, δεν κατέστη δυνατή η χρησιμοποίηση του. Συγκεκριμένα έγινε εγκατάσταση με δύο τρόπους, απευθείας στη μηχανή RTD, αλλά και ως container σε docker. Δυστυχώς το αποτέλεσμα κατά τη διασύνδεση του με το msfconsole έδινε το ίδιο σφάλμα. Στην Εικόνα 54 μπορούμε να δούμε το εν λόγω σφάλμα που δεν επέτρεψε τη χρησιμοποίηση του.

Επιπλέον αδυναμία υπήρξε στην εξαγωγή των αποτελεσμάτων σε ότι αφορούσε τις επιτυχημένες εκμεταλλεύσεις ευπαθειών, λόγω αδυναμίας του msfconsole, να το πράξει, καθώς το εν λόγω λογισμικό τρέχει αυτόνομα και αναγνωρίζει συγκεκριμένες εντολές.

Εν τέλη υπήρχε μια σκέψη διανομής του RTD και RTA σε πιθανούς χρήστες οι οποίοι θα το δοκίμαζαν. Αυτή η πτυχή δεν ολοκληρώθηκε διότι για κάποια από τα Scripts για να λειτουργήσουν χρειάζονταν λογαριασμούς και εισαγωγή προσωπικών συνθηματικών και «APIs».




*Εικόνα 54 (Αδυναμία σύνδεσης Openvas με msfconsole)*

## 6.5 Μελλοντικά Σχέδια

Το RTA έχει προοπτικές εξέλιξης καθώς έχει αποδείξει στην πράξη τις δυνατότητες του. Καταρχήν για την ολοκλήρωση των ενεργειών των Red Teamers, δύναται να ενταχθούν επιπλέον εργαλεία και Scripts στην αλυσίδα αυτοματοποιημένων ενεργειών. Συγκεκριμένα αφού εντοπιστούν οι διευθύνσεις ηλεκτρονικού ταχυδρομείου ενός οργανισμού, να αποστέλλονται αυτόματα phishing emails, με συνημμένα αρχεία τύπου *.pdf, *.xls/xlsx, *.doc/docx, με κακόβουλο λογισμικό, το οποίο με την επιτυχή του εκτέλεση να δίνει απομακρυσμένη εξουσιοδότηση (Reverse Shell) μέσω αντίστροφης σύνδεσης από το θύμα προς το RTA, με ότι αυτό συνεπάγεται.

Επιπλέον στην αλυσίδα αυτοματοποιημένων ενεργειών δύναται να εισαχθεί εργαλείο το οποίο μπορεί να σπάει συνθηματικά ασύρματων δικτύων WiFi. Με αυτό τον τρόπο ο εκάστοτε Red Teamer θα μπορεί να εισέλθει στα εσωτερικά του δικτύου του οργανισμού, με τη δυνατότητα να λάβει περαιτέρω πληροφορίες για τη δομή του δικτύου, την τοπολογία αλλά πάνω από όλα την εύρεση των ευπαθειών που πιθανόν να οδηγήσουν σε εκμετάλλευση.

Με την εκμετάλλευση των ευπαθειών στον στόχο επιβάλλεται η πλευρική κίνηση (Lateral Movement), η οποία είναι αυτή που θα λάβει αρχεία, password hashes/dumps, θα εγκαταστήσει «κοιμισμένο» κακόβουλο λογισμικό για μελλοντική χρήση, το οποίο φυσικά θα είναι δύσκολο να ανιχνευτεί, θα εκμεταλλευτεί όσο το δυνατό περισσότερους πόρους και αγαθά, ενώ τελικά θα καλύψει τα ίχνη του, ούτως ώστε να μην γίνει γνωστή ποτέ η δράση του.

Επιβάλλεται η εύρεση τρόπου για να χρησιμοποιηθεί το Openvas, καθώς θεωρείται κορυφαίο εργαλείο στο χώρο της εύρεσης των ευπαθειών. Σε αντίθετη περίπτωση ίσως να χρειαστεί η εύρεση κάποιου άλλου αντίστοιχου Vulnerability Scanner ανοιχτού κώδικα, για μια πιο ολοκληρωμένη αξιολόγηση και σφαιρικό έλεγχο των στόχων.

Με την ολοκλήρωση των παραπάνω, η απόδοση το RTA θα πρέπει να δοκιμαστεί και σε άλλα λειτουργικά συστήματα όπως Windows, Mac OSX, Κινητά τηλέφωνα, συσκευές Internet of Things, ιστοσελίδες κτλ. Για να μπορεί να είναι ολοκληρωμένο σαν λογισμικό τα παραπάνω πρέπει να ενταχθούν στο δυναμικό της αλυσίδας αυτοματοποιημένων ενεργειών.

Ίσως με την ένταξη ενός όχι τόσο κλειστού κυκλώματος αλυσίδας αυτοματοποιημένων ενεργειών, το λογισμικό να ήταν πιο εύχρηστο για κάποιους οι οποίοι ζητούν την ταχύτητα αλλά θέλουν να έχουν τον έλεγχο των κινήσεων. Για αυτό καλό θα ήταν κατά την αρχική εκτέλεση του RTA να υπάρχει η επιλογή για πλήρως αυτοματοποιημένες ενέργειες ή μερικώς αυτοματοποιημένες ενέργειες, ανάλογα με την τελική επιθυμία του εκάστοτε χειριστή.

Καλή ιδέα θα ήταν η ένταξη και αυτόματη ενημέρωση των Database που διαθέτουν τα γνωστά αλλά και καινούργια Vulnerabilities που εμφανίζονται καθημερινά καθώς επίσης και την εγκατάσταση AI (Artificial Intelligence) και Machine Learning ούτως ώστε να μπορεί από μόνο του το λογισμικό να μαθαίνει και να γίνεται ακόμα πιο αποδοτικό, ανάλογα με τα δεδομένα που θα υπάρχουν διαθέσιμα. Η γλώσσα προγραμματισμού Python έχει πολλές βιβλιοθήκες οι οποίες υποστηρίζουν τη παραπάνω πρόταση προς την πραγματοποίηση της.

Επιπλέον των παραπάνω δεν πρέπει να ξεχνούμε ότι το Red Teaming δεν τελειώνει ποτέ και γι' αυτό κάποιες φορές δεν γίνονται όλα αυτόματα. Είναι μερικές φορές που χρειάζεται η ανθρώπινη παράμετρος για να επιτευχθεί μια εκμετάλλευση. Η μετατροπή της ανθρώπινης εμπειρίας στη μορφή δομημένων δεδομένων, με την άπειρη πληροφορία που υπάρχει σήμερα, αυτή τη στιγμή μπορεί να θεωρηθεί πολύ δύσκολο έργο. Μια δυσκολότερη περίπτωση από αυτή που πειραματικά ασχοληθήκαμε, ίσως να μην είχε τα ίδια αποτελέσματα με αυτά που έχουμε σήμερα, πράγμα που σημαίνει ότι υπάρχει πρόσφορος χώρος για βελτίωση και εξέλιξη.

# Βιβλιογραφία

[1] 'Dan Farmer Quote https://thehabitstacker.com/cyber-security-quotes-that-will-blow-your-mind Last Accessed 08/11/2021'.

[2] J. A. Young, 'The Development of a Red Teaming Service-Learning Course', vol. 31, p. 23, 2020.

[3] D. A. Harris, 'Journal of Information Systems Education', vol. 14, p. 7.

[4] M. Leitner *et al.*, 'AIT Cyber Range: Flexible Cyber Security Environment for Exercises, Training and Research', in *Proceedings of the European Interdisciplinary Cybersecurity Conference*, Rennes France, Nov. 2020, pp. 1–6. doi: 10.1145/3424954.3424959.

[5] 'The history of hacking leads us to the history of ethical hacking which makes it quite interesting to note that the hist'.

[6] C. Peake, 'Red Teaming: The Art of Ethical Hacking', p. 17, 2003.

[7] 'Penetration testing vs. red teamingWhat is red teaming?'

[8] M. Denis, C. Zena, and T. Hayajneh, 'Penetration testing: Concepts, attack methods, and defense strategies', in *2016 IEEE Long Island Systems, Applications and Technology Conference (LISAT)*, Farmingdale, NY, USA, Apr. 2016, pp. 1–6. doi: 10.1109/LISAT.2016.7494156.

[9] 'The Penetration Testing Execution Standard Documentation', p. 233.

[10] G. Chu and A. Lisitsa, 'Penetration Testing for Internet of Things and Its Automation', in *2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, Exeter, United Kingdom, Jun. 2018, pp. 1479–1484. doi: 10.1109/HPCC/SmartCity/DSS.2018.00244.

[11] J. Yeo, 'Using penetration testing to enhance your company's security', *Computer Fraud & Security*, vol. 2013, no. 4, pp. 17–20, Apr. 2013, doi: 10.1016/S1361-3723(13)70039-3.

[12] S. Mansfield-Devine, 'The best form of defence – the benefits of red teaming', *Computer Fraud & Security*, vol. 2018, no. 10, pp. 8–12, Jan. 2018, doi: 10.1016/S1361-3723(18)30097-6.

[13] P. Vats, M. Mandot, and A. Gosain, 'A Comprehensive Literature Review of Penetration Testing & Its Applications', p. 7.

[14] T. Brown, ‘A Description of External Penetration Testing in Networks’, in *Information Technology - New Generations*, vol. 558, S. Latifi, Ed. Cham: Springer International Publishing, 2018, pp. 165–168. doi: 10.1007/978-3-319-54978-1_23.

[15] P. Ami and A. Hasan, ‘Seven Phrase Penetration Testing Model’, *IJCA*, vol. 59, no. 5, pp. 16–20, Dec. 2012, doi: 10.5120/9543-3991.

[16] D. Satria, A. Alanda, A. Erianda, and D. Prayama, ‘Network Security Assessment Using Internal Network Penetration Testing Methodology’, *JOIV*, vol. 2, no. 4–2, p. 360, Oct. 2018, doi: 10.30630/joiv.2.4-2.190.

[17] T. Guarda, W. Orozco, M. F. Augusto, G. Morillo, S. A. Navarrete, and F. M. Pinto, ‘Penetration Testing on Virtual Environments’, in *Proceedings of the 4th International Conference on Information and Network Security - ICINS ’16*, Kuala Lumpur, Malaysia, 2016, pp. 9–12. doi: 10.1145/3026724.3026728.

[18] N. Veerasamy, ‘High-Level Methodology for Carrying out Combined Red and Blue Teams’, in *2009 Second International Conference on Computer and Electrical Engineering*, Dubai, UAE, 2009, pp. 416–420. doi: 10.1109/ICCEE.2009.177.

[19] ‘S. Mclure, J. Scambray and G. Kurtz, Hacking exposed, Network Security Secrets and Solutions, USA: Osborne/Mcgraw Hill, ’.

[20] ‘R. Deraison and R. Gula, “Using Nessus to detect wireless access points”, Whitepaper from Tenable Security, Available on’.

[21] J. Rajendran, V. Jyothi, and R. Karri, ‘Blue team red team approach to hardware trust assessment’, in *2011 IEEE 29th International Conference on Computer Design (ICCD)*, Amherst, MA, USA, Oct. 2011, pp. 285–288. doi: 10.1109/ICCD.2011.6081410.

[22] K. Krombholz, H. Hobel, M. Huber, and E. Weippl, ‘Advanced social engineering attacks’, *Journal of Information Security and Applications*, vol. 22, pp. 113–122, Jun. 2015, doi: 10.1016/j.jisa.2014.09.005.

[23] ‘S. Granger. Social Engineering Fundamentals, Part I: Hacker ’.

[24] ‘R. Cialdini. Influence: science and practice. Allyn and Bacon, 2001.’

[25] ‘S. Stasiukonis. Social Engineering, the USB Way. 2006. avail- able at http://www.darkreading.com/security/perimeter/show’.

[26] ‘C. Herley and D. Florencio. Phishing as a Tragedy of the Commons. NSPW 2008, Lake Tahoe, CA, 2008.’

[27] ‘https://owasp.org/www-project-top-ten/ Last Accessed 22/01/2022’.

[28] ‘https://hak5.org Last Accessed 22/01/2022’.

[29] ‘Gordon Fyodor Lyon. Nmap network scanning: The official Nmap project guide to network discovery and security scanning. I’.

[30] ‘David Kennedy, Jim O’gorman, Devon Kearns, and Mati Aharoni. Metasploit: the penetration tester’s guide. No Starch Press’.

[31] ‘Jay Beale, Haroon Meer, Charl van der Walt, and Renaud Deraison. Nessus Network Auditing: Jay Beale Open Source Security’.

[32] ‘M Ugur Aksu, Kemal Bıc¸ak¸cı, and Enes Altuncu. A first look at the usability of openvas vulnerability scanner. In Wor’.

[33] ‘AG Bernardo Damele and M Stampar. Sqlmap: automatic sql injection and database takeover tool, 2012.’

[34] ‘Duane Norton. An ettercap primer. SANS Institute InfoSec Reading Room, 5, 2004.’

[35] ‘Sharon Conheady. Social engineering in IT security: Tools, tactics, and techniques. McGraw-Hill Education Group, 2014.’

[36] ‘https://www.geeksforgeeks.org/recon-ng-installation-on-kali-linux/ Last Accessed on 22/01/2021’.

[37] M. C. Ghanem and T. M. Chen, ‘Reinforcement Learning for Efficient Network Penetration Testing’, *Information*, vol. 11, no. 1, p. 6, Dec. 2019, doi: 10.3390/info11010006.

[38] J. Zhao, W. Shang, M. Wan, and P. Zeng, ‘Penetration testing automation assessment method based on rule tree’, in *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, Shenyang, China, Jun. 2015, pp. 1829–1833. doi: 10.1109/CYBER.2015.7288225.

[39] ‘Rule Tree https://www.lltlab.org/rule-tree-diagrams/ Last Accessed 01/02/2022’.

[40] H. T. Ray, R. Vemuri, and H. R. Kantubhukta, ‘Toward an Automated Attack Model for Red Teams’, *IEEE Secur. Privacy Mag.*, vol. 3, no. 4, pp. 18–25, Jul. 2005, doi: 10.1109/MSP.2005.111.

[41] ‘R. H. Bordini, J. F. Hu¨bner, and M. Wooldridge, Program- ming multi-agent systems in AgentSpeak using Jason. John Wile’.

[42] C. Phillips and L. P. Swiler, ‘A graph-based system for network-vulnerability analysis’, in *Proceedings of the 1998 workshop on New security paradigms - NSPW ’98*, Charlottesville, Virginia, United States, 1998, pp. 71–79. doi: 10.1145/310889.310919.

[43] ‘Jha et al. - 2002 - Two formal analyses of attack graphs.pdf’.

[44] M. L. Artz, ‘NetSPA: A Network Security Planning Architecture’, p. 96.

[45] ‘Drew McDermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins, P’.

[46] ‘Jo¨rg Hoffmann. The metric-ff planning system: Translating“ignoring delete lists”to numeric state variables, Journal of’.

[47] M. Martín and H. Geffner, ‘Learning Generalized Policies from Planning Examples Using Concept Languages’, *Applied Intelligence*, vol. 20, no. 1, pp. 9–19, Jan. 2004, doi: 10.1023/B:APIN.0000011138.20292.dd.

[48] ‘J. L. Obes, C. Sarraute, and G. Richarte, “‘Attack planning in the real world,’” in Proc. AAAI Workshop Intell. Secur. (’.

[49] ‘D. Elsbroek, D. Kohlsdorf, D. Menke, and L. Meyer, “‘Fidius: Intelligent support for vulnerability testing,’” in Proc IJ’.

[50] ‘POMD http://www.pomdp.org Last Accessed 10/02/2022’.

[51] ‘Cyber Ranges https://www.nist.gov/system/files/documents/2018/02/13/cyber_ranges.pdf Last Accessed 01/02/2022’.

[52] T. Lieskovan and J. Hajný, ‘Building Open Source Cyber Range To Teach Cyber Security’, in *The 16th International Conference on Availability, Reliability and Security*, Vienna Austria, Aug. 2021, pp. 1–11. doi: 10.1145/3465481.3469188.

[53] ‘CALDERA https://caldera.readthedocs.io/en/latest/ Last Accessed 02/02/2022’.

[54] A. Georgiadou, S. Mouzakitis, and D. Askounis, ‘Assessing MITRE ATT&CK Risk Using a Cyber-Security Culture Framework’, *Sensors*, vol. 21, no. 9, p. 3267, May 2021, doi: 10.3390/s21093267.

[55] A. Applebaum, D. Miller, B. Strom, C. Korban, and R. Wolf, ‘Intelligent, automated red team emulation’, in *Proceedings of the 32nd Annual Conference on Computer Security Applications*, Los Angeles California USA, Dec. 2016, pp. 363–373. doi: 10.1145/2991079.2991111.

[56] S. Randhawa, B. Turnbull, J. Yuen, and J. Dean, ‘Mission-Centric Automated Cyber Red Teaming’, in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, Hamburg Germany, Aug. 2018, pp. 1–11. doi: 10.1145/3230833.3234688.

[57] J.-K. Ke, C.-H. Yang, and T.-N. Ahn, ‘Using w3af to achieve automated penetration testing by live DVD/live USB’, in *Proceedings of the 2009 International Conference on*

*Hybrid Information Technology - ICHIT '09*, Daejeon, Korea, 2009, pp. 460–464. doi: 10.1145/1644993.1645078.

[58] E. Preston L, 'Cyber Automated Red Team Tool', Naval Postgraduate School, Monterey, California, 2019.

[59] 'J. Hong and D. S. Kim, "'Harms: Hierarchical attack representation models for network security analysis,'" in Proc. 10th'.

[60] 'J. B. Hong and D. S. Kim, "'Towards scalable security analysis using multi- layered security models,'" J. Netw. Comput. '.

[61] S. Y. Enoch, Z. Huang, C. Y. Moon, D. Lee, M. K. Ahn, and D. S. Kim, 'HARMer: Cyber-Attacks Automation and Evaluation', *IEEE Access*, vol. 8, pp. 129397–129414, 2020, doi: 10.1109/ACCESS.2020.3009748.

[62] 'https://www.tutorialspoint.com/sdlc/sdlc_spiral_model.htm Last Accessed on 21 Feb 2022'.

[63] 'https://zitoc.com/spiral-model/ Last Accessed On 21 Feb 2022'.

[64] 'https://sceweb.sce.uhcl.edu/helm/SWEN5432-SDLC/myfiles/TableContents/Module-4/module4page.html Last Accessed On 21 Feb '.

[65] 'https://help.ubuntu.com/community/Installation/MinimalCD#A64-bit_PC_.28amd64.2C_x86_64.29_.28Recommended.29 Last Accesse'.

[66] 'https://www.virtualbox.org/ Last Accessed on 09 May 2022'.

[67] 'https://www.virtualbox.org/manual/ch01.html#virt-why-useful Last Accessed on 09 May 2022'.

[68] 'Https://www.compsuccess.com/is-bash-a-coding-language/ Lasrt Accessed 09 May 2022'.

[69] 'https://www.python.org/doc/essays/blurb/ Last Accessed 09 May 2022'.

[70] 'https://asq.org/quality-resources/flowchart Last Accessed on 09 May 2022'.

[71] 'https://github.com/mrschyte/nmap-converter Last Accessed on 10 May 2022'.

[72] 'https://github.com/superhedgy/AttackSurfaceMapper Last Accessed on 10 May 2022'.

# ΠΑΡΑΡΤΗΜΑ Α

# Κώδικας

## A.1 Bash Script

```
#!/bin/bash

# 1.------ Checks for sudo privillege ======---
if [ "$(whoami)" != 'root' ]; then
  echo -e " You run as a non-root user.\n"
  echo -e " Please run again with sudo privilleges.\n"
  exit;
fi


# 2.------ User enters the ip address ======---
function enter_ip
{
echo "Please Type In The Target's IP Yow Would Like To Test"
  read target_ip
        echo ""
tput setaf 5; echo "The Target's IP you have entered is $target_ip"
              echo ""
}


# 3.------ check_ips Function - Validates, Expands CIDR ranges and Sorts IPs addresses ======---
function ip_validation
{
  if [[ "$target_ip" =~ ^(([1-9]?[0-9]|1[0-9][0-9]|2([0-4][0-9]|5[0-5]))\.){3}([1-9]?[0-9]|1[0-9][0-9]|2([0-4][0-9]|5[0-5]))$ ]];
then
    echo "This is a valid IP"
  else
    echo ""
    echo "This is not a valid IP, Please Try Again !!!"
    tput sgr0; echo ""
    exit;
  fi
}


# 4.------ Nmap Scanner commmands for IP ======---
function nmap_scan_ip
{
  tput sgr0 echo ""
  nmap -oX /home/rtd/RTA/Reports/Report.xml -T4 -A -v $target_ip
  xsltproc /home/rtd/RTA/Reports/Report.xml -o /home/rtd/RTA/Reports/Report.html
  python3 /home/rtd/RTA/Modules/nmap-converter.py /home/rtd/RTA/Reports/Report.xml -o
/home/rtd/RTA/Reports/Report.xls
```

```
}


# ---=== Nmap Scanner commmands for DNS ===---
function nmap_scan_DNS
{
  tput sgr0 echo ""
  nmap -oX /home/rtd/RTA/Reports/Report.xml -T4 -A -v $DNS
  xsltproc /home/rtd/RTA/Reports/Report.xml -o /home/rtd/RTA/Reports/Report.html
  python3 /home/rtd/RTA/Modules/nmap-converter.py /home/rtd/RTA/Reports/Report.xml -o
/home/rtd/RTA/Reports/Report.xls
}


# 5.---=== Creation of .rc files ===---
function rc
{
 python3 /home/rtd/RTA/Modules/Export_Vuln_as_rc.py
}


# 6. ---=== MSFcosnole commmands ===---
function msf
{
 numfiles=$(ls /home/rtd/RTA/RC_Resources | wc -l)
 echo ""
 tput setaf 5; echo $numfiles Possible Vulnerabilities to Exploit Target!
 tput sgr0; echo ""
  for ((vuln = 1; vuln <= $numfiles; vuln++))
   do
    echo "Trying to Exploit Vulnerability $vuln"
    msfconsole -q -r "/home/rtd/RTA/RC_Resources/$vuln.rc"
   done
}

# 7.---=== Deletion of .rc files ===---
function rc_del
{
 rm -rf /home/rtd/RTA/RC_Resources/*
}

#--------------------------------------------------------------------------------

# ---=== Animation ===---

function startup_animation
{
 spinner=('OoooooOoooooOoooooOoooooOoooooOoooooOoooo' 'oOoooooOoooooOoooooOoooooOoooooOoooooOooo'
'ooOoooooOoooooOoooooOoooooOoooooOoooooOoo' 'oooOoooooOoooooOoooooOoooooOoooooOoooooOo'
'ooooOoooooOoooooOoooooOoooooOoooooOoooooO' 'OoooooOoooooOoooooOoooooOoooooOoooooOoooo')
 intmod(){
   tput setaf 5; echo "Initialising Modules, Please Wait.."
   spin &
   pid=$!
```

```
  for i in `seq 1 2`
  do
    sleep 2;
  done
  kill $pid
 }

 spin(){
  while [ 1 ]
  do
    for i in ${spinner[@]};
    do
      echo -ne "\r$i";
     sleep 0.2;
    done;
  done
 }
 intmod
}


# ---=== Intro ASCII Art ===---

function ASCII_ART
{
tput setaf 0;   echo ""
tput setaf 1; echo ""
tput setaf 1; echo  " /@@@@@@    /@@@@@@@@     /@@@@@@"
tput setaf 1; echo  "| @@__  @@  |__  @@__/    /@@__  @@"
tput setaf 1; echo  "| @@ \ @@     | @@       | @@ \ @@"
tput setaf 1; echo  "| @@@@@@@/    | @@       | @@@@@@@@"
tput setaf 1; echo  "| @@__  @@    | @@       | @@__  @@"
tput setaf 1; echo  "| @@ \ @@     | @@       | @@ | @@"
tput setaf 1; echo  "| @@ | @@     | @@       | @@ | @@"
tput setaf 1; echo  "|__/ |__/ed   |__/eaming  |__/ |__/utomation"
   echo ""
   echo ""
}


# ---=== Play Intro Sound ===---

function intro_sound
{
 echo ""
 echo ""
 tput setaf 0; play /home/rtd/RTA/Sounds/intro.wav
}

#-------------------------------------------------------------------------

# ---====  RTA STARTS HERE  ====---
```

```
startup_animation
ASCII_ART
intro_sound

# ---=== Define Use of IP or DNS Reconnaissance Mode ===---
tput setaf 5; echo "Please Choose Reconnaissance Mode (Type 1 For IP MODE OR 2 For DNS MODE)"
tput sgr0; echo ""
read mode
 if [ $mode -eq 1 ]
 then
   echo -e "You Have Chosen IP Mode"
     enter_ip
       ip_validation
   nmap_scan_ip
 rc
 msf
 rc_del
 su rtd firefox /home/rtd/RTA/Reports/Report.html
 elif [ $mode -eq 2 ]
 then
   echo ""
   echo -e " You Have Chosen DNS Mode"
   echo -e " Enter a DNS For Recconaissanse (as google.com)"
   echo ""
   read DNS
   #nmap_scan_DNS (Need Written Permission to Run This)
   python3 /home/rtd/RTA/Modules/DNS/asm.py -t $DNS -w
/home/rtd/RTA/Modules/DNS/resources/top100_sublist.txt -o /home/rtd/RTA/Reports/$DNS -f csv
 else
   tput sgr0; echo ""
   echo -e "You Haven't Chosen Right, Please Try Again !!!"
   exit;
 fi
```

## A.2 Export_Vuln_as_rc.py Script

```
from xml.dom.minidom import parse
# .rc file structure
file_content = 'search type:exploit <PRODUCT>\n' \
            'use 0\n' \
            'show payloads\n' \
            'set payload 0\n' \
            'set rhosts <HOST>\n' \
            'exploit\n' \
          'exit'
def create_file(filename, content):

# open text file
    text_file = open(filename, "w")
    # write string to fill
    text_file.write(content)
    # close file
    text_file.close()

def parse_xml(filename):
    document = parse(filename)
    hosts = document.getElementsByTagName("host")
    for host in hosts:
        print(f'Host: [{ str(host.getElementsByTagNameNS("*", "address")[0].getAttribute("addr"))}]')
        rhost = str(host.getElementsByTagNameNS("*", "address")[0].getAttribute("addr"))
        ports = host.getElementsByTagName("port")
        idx = 0
        for port in ports:
            service = port.getElementsByTagNameNS('*','service')[0]
            if service.getAttribute("product") != '':
                idx += 1
                print(f'    Port: [{port.getAttribute("portid")}]')
                print(f'        Service: [{service.getAttribute("product")}]')
                new_file = file_content
                new_file = new_file.replace("<PRODUCT>", str(service.getAttribute("product")))
                new_file = new_file.replace("<HOST>", rhost)
                create_file( '/home/rtd/RTA/RC_Resources/'f'{str(idx)}.rc',new_file)

if __name__ == '__main__':
    parse_xml('/home/rtd/RTA/Reports/Report.xml') #path to XML File
```

## A.3 Nmap-converter.py Script [71]

```
#!/usr/bin/env python

from libnmap.parser import NmapParser, NmapParserException
from xlsxwriter import Workbook
from datetime import datetime

import os.path


class HostModule():
    def __init__(self, host):
        self.host = next(iter(host.hostnames), "")
        self.ip = host.address
        self.port = ""
        self.protocol = ""
        self.status = ""
        self.service = ""
        self.tunnel = ""
        self.method = ""
        self.source = ""
        self.confidence = ""
        self.reason = ""
        self.reason = ""
        self.product = ""
        self.version = ""
        self.extra = ""
        self.flagged = "N/A"
        self.notes = ""


class ServiceModule(HostModule):
    def __init__(self, host, service):
        super(ServiceModule, self).__init__(host)
        self.host = next(iter(host.hostnames), "")
        self.ip = host.address
        self.port = service.port
        self.protocol = service.protocol
        self.status = service.state
        self.service = service.service
        self.tunnel = service.tunnel
        self.method = service.service_dict.get("method", "")
        self.source = "scanner"
        self.confidence = float(service.service_dict.get("conf", "0")) / 10
        self.reason = service.reason
        self.product = service.service_dict.get("product", "")
        self.version = service.service_dict.get("version", "")
        self.extra = service.service_dict.get("extrainfo", "")
        self.flagged = "N/A"
        self.notes = ""
```

```
class HostScriptModule(HostModule):
    def __init__(self, host, script):
        super(HostScriptModule, self).__init__(host)
        self.method = script["id"]
        self.source = "script"
        self.extra = script["output"].strip()


class ServiceScriptModule(ServiceModule):
    def __init__(self, host, service, script):
        super(ServiceScriptModule, self).__init__(host, service)
        self.source = "script"
        self.method = script["id"]
        self.extra = script["output"].strip()


def _tgetattr(object, name, default=None):
    try:
        return getattr(object, name, default)
    except Exception:
        return default


def generate_summary(workbook, sheet, report):
    summary_header = ["Scan", "Command", "Version", "Scan Type", "Started", "Completed", "Hosts Total", "Hosts
Up", "Hosts Down"]
    summary_body = {"Scan": lambda report: _tgetattr(report, 'basename', 'N/A'),
                    "Command": lambda report: _tgetattr(report, 'commandline', 'N/A'),
                    "Version": lambda report: _tgetattr(report, 'version', 'N/A'),
                    "Scan Type": lambda report: _tgetattr(report, 'scan_type', 'N/A'),
                    "Started": lambda report: datetime.utcfromtimestamp(_tgetattr(report, 'started', 0)).strftime("%Y-%m-
%d %H:%M:%S (UTC)"),
                    "Completed": lambda report: datetime.utcfromtimestamp(_tgetattr(report, 'endtime', 0)).strftime("%Y-
%m-%d %H:%M:%S (UTC)"),
                    "Hosts Total": lambda report: _tgetattr(report, 'hosts_total', 'N/A'),
                    "Hosts Up": lambda report: _tgetattr(report, 'hosts_up', 'N/A'),
                    "Hosts Down": lambda report: _tgetattr(report, 'hosts_down', 'N/A')}

    for idx, item in enumerate(summary_header):
        sheet.write(0, idx, item, workbook.myformats["fmt_bold"])
        for idx, item in enumerate(summary_header):
            sheet.write(sheet.lastrow + 1, idx, summary_body[item](report))

    sheet.lastrow = sheet.lastrow + 1


def generate_hosts(workbook, sheet, report):
    sheet.autofilter("A1:E1")
    sheet.freeze_panes(1, 0)

    hosts_header = ["Host", "IP", "Status", "Services", "OS"]
    hosts_body = {"Host": lambda host: next(iter(host.hostnames), ""),
                  "IP": lambda host: host.address,
                  "Status": lambda host: host.status,
```

```
                "Services": lambda host: len(host.services),
                "OS": lambda host: os_class_string(host.os_class_probabilities())}

    for idx, item in enumerate(hosts_header):
        sheet.write(0, idx, item, workbook.myformats["fmt_bold"])

    row = sheet.lastrow
    for host in report.hosts:
        for idx, item in enumerate(hosts_header):
            sheet.write(row + 1, idx, hosts_body[item](host))
        row += 1

    sheet.lastrow = row


def generate_results(workbook, sheet, report):
    sheet.autofilter("A1:N1")
    sheet.freeze_panes(1, 0)

    results_header = ["Host", "IP", "Port", "Protocol", "Status", "Service", "Tunnel", "Source", "Method",
"Confidence", "Reason", "Product", "Version", "Extra", "Flagged", "Notes"]
    results_body = {"Host": lambda module: module.host,
                    "IP": lambda module: module.ip,
                    "Port": lambda module: module.port,
                    "Protocol": lambda module: module.protocol,
                    "Status": lambda module: module.status,
                    "Service": lambda module: module.service,
                    "Tunnel": lambda module: module.tunnel,
                    "Source": lambda module: module.source,
                    "Method": lambda module: module.method,
                    "Confidence": lambda module: module.confidence,
                    "Reason": lambda module: module.reason,
                    "Product": lambda module: module.product,
                    "Version": lambda module: module.version,
                    "Extra": lambda module: module.extra,
                    "Flagged": lambda module: module.flagged,
                    "Notes": lambda module: module.notes}

    results_format = {"Confidence": workbook.myformats["fmt_conf"]}

    print("[+] Processing {}".format(report.summary))
    for idx, item in enumerate(results_header):
        sheet.write(0, idx, item, workbook.myformats["fmt_bold"])

    row = sheet.lastrow
    for host in report.hosts:
        print("[+] Processing {}".format(host))

        for script in host.scripts_results:
            module = HostScriptModule(host, script)
            for idx, item in enumerate(results_header):
                sheet.write(row + 1, idx, results_body[item](module), results_format.get(item, None))
            row += 1
```

```
        for service in host.services:
            module = ServiceModule(host, service)
            for idx, item in enumerate(results_header):
                sheet.write(row + 1, idx, results_body[item](module), results_format.get(item, None))
            row += 1

            for script in service.scripts_results:
                module = ServiceScriptModule(host, service, script)
                for idx, item in enumerate(results_header):
                    sheet.write(row + 1, idx, results_body[item](module), results_format.get(item, None))
                row += 1

    sheet.data_validation("O2:O${}".format(row + 1), {"validate": "list",
                                      "source": ["Y", "N", "N/A"]})
    sheet.lastrow = row


def setup_workbook_formats(workbook):
    formats = {"fmt_bold": workbook.add_format({"bold": True}),
               "fmt_conf": workbook.add_format()}

    formats["fmt_conf"].set_num_format("0%")
    return formats


def os_class_string(os_class_array):
    return " | ".join(["{0} ({1}%)".format(os_string(osc), osc.accuracy) for osc in os_class_array])


def os_string(os_class):
    rval = "{0}, {1}".format(os_class.vendor, os_class.osfamily)
    if len(os_class.osgen):
        rval += "({0})".format(os_class.osgen)
    return rval


def main(reports, workbook):
    sheets = {"Summary": generate_summary,
              "Hosts": generate_hosts,
              "Results": generate_results}

    workbook.myformats = setup_workbook_formats(workbook)

    for sheet_name, sheet_func in sheets.items():
        sheet = workbook.add_worksheet(sheet_name)
        sheet.lastrow = 0
        for report in reports:
            sheet_func(workbook, sheet, report)
    workbook.close()


if __name__ == "__main__":
    import argparse
    parser = argparse.ArgumentParser()
```

```
parser.add_argument("-o", "--output", metavar="XLS", help="path to xlsx output")
parser.add_argument("reports", metavar="XML", nargs="+", help="path to nmap xml report")
args = parser.parse_args()

if args.output == None:
    parser.error("Output must be specified")

reports = []
for report in args.reports:
    try:
        parsed = NmapParser.parse_fromfile(report)
    except NmapParserException as ex:
        parsed = NmapParser.parse_fromfile(report, incomplete=True)

    parsed.basename = os.path.basename(report)
    reports.append(parsed)

workbook = Workbook(args.output)
main(reports, workbook)
```

## A.4 DNS (ATTACK SURFACE MAPPER)[72]

```
#!/usr/bin/python3
# Example:
# $ python3 asm.py -t example.com
#

# Standard Libraries
import argparse
import ipaddress
import json
import os
import signal
import socket
import sys
from datetime import datetime
from time import time, sleep
from urllib import parse

# External Libraries
import colorama
import pymongo
import requests
from colorama import Fore, Style
from netaddr import IPNetwork
from validator_collection import checkers

# ASM Modules
from modules import buckethunter
from modules import dnsdumpster
from modules import hosthunter
from modules import hunterio
from modules import screencapture
from modules import shodan
from modules import subhunter
from modules import urlscanio
from modules import whois_collector
from modules import censys

# Constants
__author__ = ""
__version__ = "v1.2"


# Classes
class TargetIP:
    def __init__(self, addr):
        self.address = addr
        self.hostname = []
        self.ports = []
        self.asn = ""
        self.asn_name = ""
```

```
        self.whois = []
        self.server = ""
        self.vulns = []
        self.cidr = ""
        self.location = ""
        self.country = ""


class Target:
    def __init__(self):
        self.primary_domain = ""
        self.subdomains = []
        self.orgName = ""
        self.dnsrecords = []
        self.buckets = []
        self.mx = None
        self.spf = None
        self.dmarc = []
        self.dmarc_status = ""
        self.emails = []
        self.guessed_emails = []
        self.creds = []
        self.hashes = []
        self.breaches = {}
        self.employees = []
        self.pattern = "{f}{last}"  # Default Email Pattern
        self.urls = []
        self.ipv4 = False
        self.resolved_ips = []


class Counter:
    def __init__(self):
        Counter.targets = 0
        Counter.ports = 0
        Counter.hostnames = 0
        Counter.subdomains = 0
        Counter.urls = 0
        Counter.buckets = 0
        Counter.sc = 0
        Counter.employees = 0
        Counter.intel = 0
        Counter.ips = 0
        Counter.vulns = 0
        Counter.emails = 0
        Counter.guessed_emails = 0
        Counter.creds = 0
        Counter.hashes = 0


class MasterSwitch:
    def __init__(self):
        self.shodan = True
        self.hunterio = True
```

```
        self.whois_collector = True
        self.subhunter = True
        self.dnsdumpster = True
        self.urlscanio = True
        self.weleakinfo = True
        self.weleakinfo_private = True
        self.screencapture = True
        self.webscraper = True
        self.censys = False
        self.expand = False
        self.stealth = False
        self.verbose = False
        self.debug = False


# Initial Checks & Argument Parsing
def init_checks(master_switch, outpath):
    global args
    # Argument Parser
    parser = argparse.ArgumentParser(description='|<---------->|',
                                     epilog="Authors:" + __author__ + "\n\n ",
                                     formatter_class=argparse.RawTextHelpFormatter)
    parser.add_argument("-f", "--format", help="Choose between CSV and TXT output file formats.", default="csv")
    parser.add_argument("-o", "--output", help="Sets the path of the output file.", type=str, default=outpath)
    parser.add_argument("-sc", "--screen-capture", help="Capture a screen shot of any associated Web
Applications.",
                        action="store_true", default=False)
    parser.add_argument("-sth", "--stealth", help="Passive mode allows reconnaissance using OSINT techniques
only.",
                        action="store_true", default=False)
    parser.add_argument("-t", "--target", help="Set a single target IP.")
    parser.add_argument("targets", nargs='?', help="Sets the path of the target IPs file.", type=str, default="")
    parser.add_argument("-V", "--version", help="Displays the current version.", action="store_true", default=False)
    parser.add_argument("-w", "--wordlist", help="Specify a list of subdomains.", type=str,
                        default="/home/rtd/RTA/Modules/DNS/resources/bitquark_top100k_sublist.txt")
    parser.add_argument("-sw", "--subwordlist", help="Specify a list of child subdomains.", type=str,
                        default="/home/rtd/RTA/Modules/DNS/resources/top1000_sublist.txt")
    parser.add_argument("-e", "--expand", help="Expand the target list recursively.", action="store_true",
                        default=False)
    parser.add_argument("-d", "--debug", help="Enables debugging
information.",action="store_true",default=False)
    parser.add_argument("-v", "--verbose", help="Verbose output in the terminal window.", action="store_true",
                        default=False)
    args = parser.parse_args()


    if not (args.target or args.targets):
        cprint("error", "Please specify a single target or a list of targets.", 1)
        cprint("info", "Example Usage: python3 asm.py -t superhedgy.com", 1)
        exit()

    if args.version:
        print("HostHunter version", __version__)
        exit()
```

```
    if args.target and args.targets:
        cprint("error", "Too many arguments! Either select single target or specify a list of targets.", 1)
        cprint("info", "Example Usage: python3 asm.py -t superhedgy.com", 1)
        exit()
    # Targets Input File
    if args.targets and not os.path.exists(args.targets):
        cprint("error", "Targets file \"" + args.targets + "\" does not exist", 1)
        exit()

    if args.wordlist is None or args.wordlist == "":
        cprint("error", "Wordlist file argument is empty", 1)
        exit()

    if args.subwordlist is None or args.subwordlist == "":
        cprint("error", "Wordlist file argument is empty", 1)
        exit()

    if args.wordlist and not os.path.exists(args.wordlist):
        cprint("error", "Wordlist file \"" + str(args.wordlist) + "\" does not exist", 1)
        exit()

    if args.subwordlist and not os.path.exists(args.subwordlist):
        cprint("error", "SubWordlist file \"" + str(args.subwordlist) + "\" does not exist", 1)
        exit()

    if args.output and os.path.exists(args.output):
        cprint("info", "\n[?] {0} file already exists, would you like to overwrite it?".format(args.output), 1)
        while True:
            answer = input(
                Fore.WHITE + "[" + Fore.RED + Style.BRIGHT + ">" + Style.RESET_ALL + Fore.WHITE + "]" +
Fore.WHITE + " Answer with [Y]es or [N]o : ").lower()
            if answer.startswith("n"):
                exit()
            elif answer.startswith("y"):
                break

    print(Style.RESET_ALL)

    amionline()

    if args.expand:
        cprint("info", "\n[!] Expand Mode Enabled, out-of-scope targets might be included.\n", 1)  # Success Msg
        master_switch.expand = True
        sleep(0.5)

    if args.verbose:
        master_switch.verbose = True


    if args.debug:
        master_switch.debug = True

    return args.output
```

```
# MongoDB
mongo_client = pymongo.MongoClient('localhost', 27017)
db = mongo_client.asm
store_targets = db.targets

# Checks for an Internet Connection
def amionline():
    try:
        socket.setdefaulttimeout(5)
        socket.socket(socket.AF_INET, socket.SOCK_STREAM).connect(("8.8.8.8", 53))  # Google DNS IPv4
    except:
        cprint("white", "\n" + 82 * "*", 0)
        cprint("error", "No Internet Connection! Ensure that you are online and run ASM again.", 0)
        print(82 * "*" + "\n")
        while 1:
            try:
                socket.setdefaulttimeout(8)
                socket.socket(socket.AF_INET, socket.SOCK_STREAM).connect(("8.8.4.4", 53))  # OpenDNS IPv4
                return True
            except:
                input("[>] Press any key to resume . . .")
    return True

# Resolve Domain Function - Returns a list
def asn_expansion(mswitch, hostx):
    cprint("info", "[i] Searching for ASNs based on: " + hostx.orgName, 1)
    answer3 = input(
        Fore.WHITE + "[" + Fore.RED + Style.BRIGHT + ">" + Style.RESET_ALL + Fore.WHITE + "]" + "[EXPAND-
MODE]" + Fore.WHITE + " Enter Company Name: ")
    print(Style.RESET_ALL)
    if answer3 == "":
        asn_query = parse.quote(hostx.orgName)
    else:
        asn_query = parse.quote(answer3)

    user_agent = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/74.0.3729.169 Safari/537.36'}
    api_endpoint = "https://api.bgpview.io/search?query_term=" + asn_query
    try:
        response = requests.get(api_endpoint, headers=user_agent)
        api = json.loads(response.text)
    except:
        cprint("error", "Module expansion failed.", 1)
        return -1
    asns = []
    prefixes = []

    if response.status_code == 200:
        try:
            for item in api['data']['asns']:
                # print (item)
```

```
        if mswitch.verbose is True:
            print(60 * "*")
            print("ASN: " + str(item["asn"]))
            print("ASN Name: " + item.get("name"))
            print("Description: " + item["description"])
            print("Location: " + item["country_code"])
            print(60 * "*")
        asns.append(item["asn"])
    except:
        cprint("error", "Module expansion failed.", 1)
        return -1

    try:
        for item in api['data']['ipv4_prefixes']:

            if mswitch.verbose is True:
                print(60 * "*")
                print("CIDR: " + item["ip"] + "/" + str(item["cidr"]))
                print("Prefix Name: " + item.get("name"))
                print("Prefix Description: " + item["description"])
                print("Location: " + item["country_code"])
                print(60 * "*")

            prefixes.append(item["prefix"])
    except:
        cprint("error", "Module expansion failed while searching for IPv4 prefixes.", 1)
        return -1

    if len(asns) > 0:
        for asn in asns:
            try:
                response2 = requests.get("https://api.bgpview.io/asn/{0}/prefixes".format(str(asn)),
                                         headers=user_agent)
                api2 = json.loads(response2.text)
                # print(response2.text)
                print(api2["data"]["ipv4_prefixes"])
                if response2.status_code == 200:
                    try:
                        for item in api2["data"]["ipv4_prefixes"]:
                            # print (item)
                            if mswitch.verbose is True:
                                cprint("info", "Expanding prefix:" + str(item["prefix"]), 1)
                            prefixes.append(item["prefix"])
                    except:
                        cprint("error", "Module expansion failed.", 1)
                        return -1
            except:
                return -1

    if len(prefixes) > 0:
        for prefix in prefixes:
            if mswitch.verbose is True:
                cprint("info", "[i] Adding IPv4 Prefix to the targets list:" + prefix, 1)
```

```
            for ip in IPNetwork(prefix):
                tmp = TargetIP(ip)
                if ip not in hostx.resolved_ips:
                    hostx.resolved_ips.append(tmp)


# Resolve Domain Function - Returns a list
def resolve_domain(domain):
    try:
        resolve = socket.gethostbyname_ex(domain)
        IP = resolve[2]
        return IP
    except Exception:
        return ""


# validate Funciton - Validates IP
def add_target_domain(list_domain, input_domain, validated_input):
    input_domain = input_domain.replace("\n", "")
    if not input_domain:
        return 0

    t = Target()
    # Valid Input IPv4
    if checkers.is_ipv4(input_domain):
        if ipaddress.ip_address(input_domain).is_private:
            cprint("error", "" + input_domain + "" + " is a Private IPv4 address", 1)
            return 0
        else:
            validated_input.append(input_domain)
            return 0
    # Valid Input Domain
    elif checkers.is_domain(input_domain):
        validated_input.append(input_domain)
        t.ipv4 = False
        t.primary_domain = input_domain
        associated_ips = resolve_domain(t.primary_domain)
        # if not associated_ips:
        #     return 0
        if len(associated_ips) > 0:
            for x in associated_ips:
                tmp = TargetIP(x)
                if not ipaddress.ip_address(x).is_private:
                    t.resolved_ips.append(tmp)

        if t.primary_domain in list_domain.keys():
            cprint("info", "[i] Target Not Added - Domain {0} Exists".format(t.primary_domain), 1)
            pass
        else:
            list_domain[t.primary_domain] = t
            cprint("info", "[i] Target [{0}] Added.".format(t.primary_domain), 1)
    else:
        cprint("error", input_domain + " is not a valid IPv4 address or Domain Name", 1)
        return 0
```

```
def add_target_ip(target_list, IP):
    validated_input = []  # Dummy list
    tmp = TargetIP(IP)

    # Validate IPv4 Input Address
    if checkers.is_ipv4(IP):
        if ipaddress.ip_address(IP).is_private:
            return 0

        for key in target_list.keys():
            # print("Key: "+key)
            for x in target_list[key].resolved_ips:
                if (IP == x.address):
                    cprint("info", "Target Not Added - Address {0} already exists.".format(IP), 1)
                    return 0

        domain = urlscanio.get_domain(IP)
        if domain:
            add_target_domain(list, domain, validated_input)
            try:
                target_list[domain].resolved_ips.append(tmp)
            except:
                pass
        # Create a List with Unsorted IPs
        # add_target_domain(list,"",validated_input)


# keyloader Function - Load API Keys
def keyloader(keychain, master_switch):
    keyfile = open("/home/rtd/RTA/Modules/DNS/keylist.asm", "rt")  # Read keylist.asm File

    for line in keyfile:
        words = line.split()
        try:
            keychain[words[0]] = words[2].replace("\"", "")
        except:
            pass

    print("{0}   HostHunter Module  : [{1}Enabled{2}]{3}".format(Fore.WHITE + Style.BRIGHT, Fore.GREEN,
Fore.WHITE, Style.RESET_ALL))
    master_switch.hosthunter = True

    if args.screen_capture:
        print("{0}   ScreenCapture Module   : [{1}Enabled{2}]{3}".format(Fore.WHITE + Style.BRIGHT, Fore.GREEN,
Fore.WHITE, Style.RESET_ALL))
        master_switch.screencapture = True
    else:
        print("{0}   ScreenCapture Module   : [{1}Disabled{2}]{3}".format(Fore.WHITE + Style.BRIGHT, Fore.RED,
Fore.WHITE, Style.RESET_ALL))
        master_switch.screencapture = False
    print("{0}   DNSdumpster Module : [{1}Enabled{2}]{3}".format(Fore.WHITE + Style.BRIGHT, Fore.GREEN +
Style.BRIGHT, Fore.WHITE, Style.RESET_ALL))
```

```
    print("{0}   URLScanIO Module   : [{1}Enabled{2}]{3}".format(Fore.WHITE + Style.BRIGHT, Fore.GREEN,
Fore.WHITE, Style.RESET_ALL))

    if len(keychain["hunterio"]) == 40:
        print("{0}   HunterIO Module    : [{1}Enabled{2}]{3}".format(Fore.WHITE + Style.BRIGHT, Fore.GREEN,
Fore.WHITE, Style.RESET_ALL))
        master_switch.hunterio = True
    else:
        print("{0}   HunterIO Module    : [{1}Disabled{2}]{3}".format(Fore.WHITE + Style.BRIGHT, Fore.RED,
Fore.WHITE, Style.RESET_ALL))
        master_switch.hunterio = False

    if len(keychain["shodan"]) == 32:
        print("{0}   Shodan Module      : [{1}Enabled{2}]{3}".format(Fore.WHITE + Style.BRIGHT, Fore.GREEN,
Fore.WHITE, Style.RESET_ALL))
        master_switch.shodan = True
    else:
        print("{0}   Shodan Module      : [{1}Disabled{2}]{3}".format(Fore.WHITE + Style.BRIGHT, Fore.RED,
Fore.WHITE, Style.RESET_ALL))
        master_switch.shodan = False

    if len(keychain["virustotal"]) == 64:
        print(
            "{0}   VirusTotal Module    : [{1}Enabled{2}]{3}".format(Fore.WHITE + Style.BRIGHT, Fore.GREEN,
Fore.WHITE, Style.RESET_ALL))
        master_switch.virustotal = True
    else:
        print(
            "{0}   VirusTotal Module    : [{1}Disabled{2}]{3}".format(Fore.WHITE + Style.BRIGHT, Fore.RED,
Fore.WHITE, Style.RESET_ALL))
        master_switch.virustotal = False

    if len(keychain["censys_id"]) > 0 and len(keychain["censys_secret"]) > 0:
        print("{0}   Censys Module  : [{1}Enabled{2}]{3}".format(Fore.WHITE + Style.BRIGHT, Fore.GREEN,
Fore.WHITE, Style.RESET_ALL))
        master_switch.censys = True
    else:
        print("{0}   Censys Module  : [{1}Disabled{2}]{3}".format(Fore.WHITE + Style.BRIGHT, Fore.RED,
Fore.WHITE, Style.RESET_ALL))
        master_switch.censys = False

    if args.expand:
        print("{0}   SubHunter Module   : [{1}Recursive{2}]{3}".format(Fore.WHITE + Style.BRIGHT, Fore.YELLOW,
Fore.WHITE, Style.RESET_ALL))
        master_switch.subhunter = True
    else:
        print("{0}   SubHunter Module   : [{1}Active{2}]{3}".format(Fore.WHITE + Style.BRIGHT, Fore.GREEN,
Fore.WHITE, Style.RESET_ALL))


# msave Function - Store output in MongoDB [UAT]
def msave(t):
    mtarget = {
        "Address": t.address,
```

```
        "ASN": t.asn,
        "Hostnames": t.hname,
        "Apps": t.urls,
        "IPv6": t.ipv6,
        "CIDR": t.cidr
    }
    store_targets.insert_one(mtarget)
    print("Stored")


# cprint Function - Prints Coloured Messages
def cprint(type, msg, reset):
    colorama.init()
    message = {
        "action": Fore.YELLOW,
        "positive": Fore.GREEN + Style.BRIGHT,
        "info": Fore.YELLOW,
        "reset": Style.RESET_ALL,
        "red": Fore.RED,
        "white": Fore.WHITE,
        "green": Fore.GREEN,
        "yellow": Fore.YELLOW
    }
    style = message.get(type.lower())

    if type == "error":
        print("{0}\n[*] Error: {1}".format(Fore.RED + Style.BRIGHT, Style.RESET_ALL + Fore.WHITE + msg))
    else:
        print(style + msg, end="")
    if reset == 1:
        print(Style.RESET_ALL)


# Print Results Function -
def store_results(hostx, output_path):
    results_path = output_path + "/" + hostx.primary_domain

    try:
        os.mkdir(results_path)
    except:
        pass

    emails_filepath = results_path + "/" + 'emails.csv'
    emails_file = open(emails_filepath, 'w')

    g_emails_filepath = results_path + "/" + 'guessed_emails.csv'
    g_emails_file = open(g_emails_filepath, 'w')

    usernames_filepath = results_path + "/" + 'usernames.csv'
    usernames_file = open(usernames_filepath, 'w')

    creds_filepath = results_path + "/" + 'creds.csv'
    creds_file = open(creds_filepath, 'w')
```

```
hashes_filepath = results_path + "/" + 'hashes.csv'
hashes_file = open(hashes_filepath, 'w')

subs_filepath = results_path + "/" + 'subdomains.csv'
subs_file = open(subs_filepath, 'w')

dns_filepath = results_path + "/" + 'dns_records.csv'
dns_file = open(dns_filepath, 'w')

employees_filepath = results_path + "/" + 'employees.csv'
employees_file = open(employees_filepath, 'w')

targetips_filepath = results_path + "/" + 'target_ips.csv'
targetips_file = open(targetips_filepath, 'w')

buckets_filepath = results_path + "/" + 's3buckets.csv'
buckets_file = open(buckets_filepath, 'w')

spoofchecks_filepath = results_path + "/" + 'email_spoof_checks.csv'
spoofchecks_file = open(spoofchecks_filepath, 'w')

for email in hostx.emails:
    emails_file.write(email + '\n')

for email in hostx.emails:
    usernames_file.write(email.split('@', 1)[0] + '\n')

for cred in hostx.creds:
    creds_file.write(cred + '\n')

for hash in hostx.hashes:
    hashes_file.write(hash + '\n')

for sub in hostx.subdomains:
    subs_file.write(sub + '\n')

for record in hostx.dnsrecords:
    dns_file.write(record + '\n')

for bucket in hostx.buckets:
    buckets_file.write(bucket + '\n')

for gemail in hostx.guessed_emails:
    g_emails_file.write(gemail + '\n')

spoofchecks_file.write("Target : " + hostx.primary_domain + '\n')
spoofchecks_file.write("SPF : " + str(hostx.spf) + '\n')
spoofchecks_file.write("DMARC Status : " + hostx.dmarc_status + '\n')
spoofchecks_file.write("DMARC Record : " + '\n')

for record in hostx.dmarc:
    spoofchecks_file.write(record + '\n')
```

```
    if len(hostx.employees) > 0:
        employees_file.write("\"" + "Email Address" + "\",\"" + "Full Name" + "\",\"" + "First Name" + "\",\"" + "Last
Name" + "\",\"" + "LinkedIn Profile URL" + "\"\n")
        for employee in hostx.employees:
            employees_file.write("\"" + employee[0] + "\",\"" + employee[1] + "\",\"" + employee[2] + "\",\"" +
employee[3] + "\",\"" + employee[4] + "\"\n")

    # Write Header
    targetips_file.write(
        "\"" + "IP Address" + "\",\"" + "Port/Protocol" + "\",\"" + "ASN" + "\",\"" + "ASN Description" + "\",\"" +
"Vulnerabities" + "\",\"" + "Location" + "\"\n")
    for ip in hostx.resolved_ips:
        targetips_file.write("\"" + ip.address + "\",\"" + ", ".join(
            map(str, ip.ports)) + "\",\"" + ip.asn + "\",\"" + ip.asn_name + "\",\"" + ", ".join(
            map(str, ip.vulns)) + "\",\"" + ip.location + "\"\n")


# Print Results Function - Terminal Output
def print_results(count1, stime):
    ttime = round(time() - stime, 2)
    cprint("green", "\n" + "-" * 80, 1)
    print(Fore.YELLOW + "\n[%] Analysed {0} Targets in {1} sec".format(
        Fore.RED + Style.BRIGHT + str(count1.targets) + Style.RESET_ALL + Fore.YELLOW,
        Fore.WHITE + Style.BRIGHT + str(ttime) + Fore.YELLOW))
    print(Fore.WHITE + Style.BRIGHT + "\n[%] Discovered:" + Style.RESET_ALL)
    print(" {0} IPs".format(Fore.RED + Style.BRIGHT + str(count1.ips) + Style.RESET_ALL + Fore.WHITE))
    print(" {0} Open Ports".format(Fore.RED + Style.BRIGHT + str(count1.ports) + Style.RESET_ALL +
Fore.WHITE))
    print(" {0} Hostnames".format(Fore.RED + Style.BRIGHT + str(count1.hostnames) + Style.RESET_ALL +
Fore.WHITE))
    print(" {0} Subdomains".format(Fore.RED + Style.BRIGHT + str(count1.subdomains) + Style.RESET_ALL +
Fore.WHITE))
    print(" {0} Vulnerabities".format(Fore.RED + Style.BRIGHT + str(count1.vulns) + Style.RESET_ALL +
Fore.WHITE))
    print(Fore.WHITE + Style.BRIGHT + "\n[%] Intelligence Extracted:" + Style.RESET_ALL)
    print(" {0} Employees' Details".format(
        Fore.RED + Style.BRIGHT + str(count1.employees) + Style.RESET_ALL + Fore.WHITE))
    print(" {0} AWS Buckets Discovered".format(
        Fore.RED + Style.BRIGHT + str(count1.buckets) + Style.RESET_ALL + Fore.WHITE))
    print(" {0} Email Addresses".format(Fore.RED + Style.BRIGHT + str(count1.emails) + Style.RESET_ALL +
Fore.WHITE))
    print(" {0} Guessed Emails".format(
        Fore.RED + Style.BRIGHT + str(count1.guessed_emails) + Style.RESET_ALL + Fore.WHITE))
    print(" {0} Credentials".format(Fore.RED + Style.BRIGHT + str(count1.creds) + Style.RESET_ALL +
Fore.WHITE))
    print(" {0} Password Hashes".format(Fore.RED + Style.BRIGHT + str(count1.hashes) + Style.RESET_ALL +
Fore.WHITE))
    print(" {0} Screenshots {1}".format(Fore.RED + Style.BRIGHT + str(count1.sc) + Style.RESET_ALL +
Fore.WHITE,
                                        Style.RESET_ALL))
    cprint("green", "\n" + "-" * 80, 1)


# Main Function
```

```
def main(keychain, switch, output_path, count):
    validated_input = []
    targets = []
    target_list = dict()  # Creates a Dict of targets
    if args.target:
        targets.append(args.target)
    else:
        targets = open(args.targets, "rt")  # Read File

    # Add Domain Name Targets
    for line in targets:
        if line == "" or line == "\n":
            continue
        if not add_target_domain(target_list, line, validated_input):
            continue
    # End of For Loop

    # Add IPv4 Targets
    for line in validated_input:
        if not add_target_ip(target_list, line):
            continue
    # End of For Loop

    # Debug Functionality
    if switch.debug is True:
        for k in target_list.keys():
            print("*************************")
            print("[DEBUG] IP Target Added to Dictionary:", k)
            for x in target_list[k].resolved_ips:
                print(">> " + x.address)
            print("*************************")

    # Iterates Through the Target List
    for key in target_list.keys():
        # [B] Target - Domain Name - Execution Flow
        cprint("white", "\n[+] Target Domain: ", 0)
        cprint("red", target_list[key].primary_domain, 1)

        for ip in target_list[key].resolved_ips:
            cprint("white", "    ", 1)
            cprint("white", "  [{0}]".format(ip.address), 1)

        if switch.expand is True:
            subhunter.active(switch, target_list[key], args.wordlist, args.subwordlist, recursive=True)  # Passive
        else:
            subhunter.active(switch, target_list[key], args.wordlist, args.subwordlist,
                             recursive=False)  # Passive Recursive Depth=2

        # HTTP Based
        if switch.stealth is False:
            hosthunter.active(target_list[key], count)  # Active

        # IP Based
```

```
    if switch.shodan is True:
        shodan.port_scan(target_list[key], keychain["shodan"], count)  # Passive

    if switch.whois_collector is True and switch.stealth is False:
        whois_collector.wlookup(target_list[key])  # Active

    if switch.censys is True:
        censys.port_scan(target_list[key], keychain["censys_id"], keychain["censys_secret"], count)  # Passive

    # hosthunter.query_api(target_list[key]) # Passive
    hosthunter.org_finder(target_list[key])  # Passive

    buckethunter.passive_query(switch,target_list[key], keychain["grayhatwarfare"])  # Passive

    if switch.expand is True:
        asn_expansion(target_list[key])  # Passive

    # Domain Based
    hosthunter.dnslookup(target_list[key])  # Passive
    urlscanio.query(target_list[key])  # Passive

    if switch.virustotal is True:
        subhunter.passive_query(target_list[key], keychain["virustotal"])  # Passive

    if switch.stealth is not True:
        hosthunter.dnsquery(target_list[key])  # Active
        pass

    if switch.hunterio is True:
        hunterio.query(target_list[key], keychain["hunterio"])

    map_path = dnsdumpster.get_map(target_list[key], output_path)  # Passive

    if (args.screen_capture and not args.stealth):
        screencapture.main(target_list[key], output_path)

    if len(target_list[key].orgName) > 0:
        print("\n {0}|| Organisation Name : {1}".format(Fore.WHITE,
                                            Fore.YELLOW + target_list[key].orgName + Style.RESET_ALL))

    if len(target_list[key].subdomains) > 0:
        print(Fore.WHITE + " || Subdomains: " + Fore.YELLOW + str(
            len(target_list[key].subdomains)) + Style.RESET_ALL)
        for i in range(len(target_list[key].subdomains)):
            cprint("info", target_list[key].subdomains[i], 0)
            if i > 50:
                if len(target_list[key].subdomains) > 50:
                    cprint("info", "\n...", 1)
                else:
                    cprint("info", "", 1)
                break
            if i == (len(target_list[key].subdomains)) - 1:
                cprint("info", "", 1)
```

```
            else:
                cprint("info", ",", 0)

        if len(target_list[key].emails) > 0:
            print(Fore.WHITE + " || Emails: " + Fore.YELLOW + str(len(target_list[key].emails)) + Style.RESET_ALL)
            for i in range(len(target_list[key].emails)):
                cprint("yellow", target_list[key].emails[i], 0)
                if i > 50:
                    if len(target_list[key].emails) > 50:
                        cprint("yellow", "\n...", 1)
                    else:
                        cprint("info", "", 1)
                    break
                if i == (len(target_list[key].emails)) - 1:
                    cprint("info", "", 1)
                else:
                    cprint("info", ",", 0)

        if len(target_list[key].guessed_emails) > 0:
            print(Fore.WHITE + " || Guessed Emails: " + Fore.YELLOW + str(
                len(target_list[key].guessed_emails)) + Style.RESET_ALL)
            for i in range(len(target_list[key].guessed_emails)):
                cprint("yellow", target_list[key].guessed_emails[i], 0)
                if i > 50:
                    if len(target_list[key].guessed_emails) > 50:
                        cprint("yellow", "\n...", 1)
                    else:
                        cprint("info", "", 1)
                    break
                if i == (len(target_list[key].guessed_emails)) - 1:
                    cprint("info", "", 1)
                else:
                    cprint("info", ",", 0)

# WeLeakInfo Code - Left here for future use
#     if len(target_list[key].breaches) > 0:
#         cprint("white", " || WeLeakInfo Data Breaches: ", 1)
#             for email, breach in target_list[key].breaches.items():
#                 cprint("yellow", "{0} : {1}".format(email, breach), 1)
#         if len(target_list[key].creds) > 0:
#             cprint("white", " || WeLeakInfo Credentials Discovered: ", 0)
#             cprint("info", "" + str(len(target_list[key].creds)), 1)
#             for i in range(len(target_list[key].creds)):
#                 cprint("yellow", target_list[key].creds[i], 1)
#                 if i > 8:
#                     if len(target_list[key].creds) > 8:
#                         cprint("yellow", "...", 1)
#                     break
#         if len(target_list[key].hashes) > 0:
#             cprint("white", " || WeLeakInfo Hashes Discovered: ", 0)
#             cprint("info", "" + str(len(target_list[key].hashes)), 1)
#             for i in range(len(target_list[key].hashes)):
#                 cprint("yellow", target_list[key].hashes[i], 1)
```

```
#             if i > 8:
#                 if len(target_list[key].hashes) > 5:
#                     cprint("yellow", "...", 1)
#                 break
#

        if len(target_list[key].buckets) > 0:
            cprint("white", " || AWS Buckets Discovered: ", 0)
            cprint("info", "" + str(len(target_list[key].buckets)), 1)
            for i in range(len(target_list[key].buckets)):
                cprint("yellow", target_list[key].buckets[i], 1)
                if i > 5:
                    if len(target_list[key].buckets) > 5:
                        cprint("yellow", "\n...", 1)
                    break

        print(" {0}|| DNS Records : {1}".format(Fore.WHITE, Fore.YELLOW + str(len(target_list[key].dnsrecords)) +
Style.RESET_ALL))
        for i in range(len(target_list[key].dnsrecords)):
            cprint("yellow", target_list[key].dnsrecords[i], 1)
            if i > 2:
                if len(target_list[key].dnsrecords) > 2:
                    cprint("info", "...", 1)
                break

        if target_list[key].mx is not None:
            cprint("white", " || MX Records :", 1)
            for dt in target_list[key].mx:
                cprint("info", str(dt.exchange), 1)

        if target_list[key].spf is not None:
            if target_list[key].spf:
                print(
                    " {0}|| SPF : {1}".format(Fore.WHITE, Fore.GREEN + str(target_list[key].spf) + Style.RESET_ALL))
            else:
                print(" {0}|| SPF   : {1}".format(Fore.WHITE, Fore.RED + Style.BRIGHT + str(
                    target_list[key].spf) + Style.RESET_ALL))

        if len(target_list[key].dmarc) > 0:
            print(
                " {0}|| DMARC : {1}".format(Fore.WHITE, Fore.GREEN + target_list[key].dmarc_status +
Style.RESET_ALL))
        else:
            print(" {0}|| DMARC : {1}".format(Fore.WHITE, Fore.RED + Style.BRIGHT + "False" + Style.RESET_ALL))

        print(" {0}|| dnsDumpster Map: {1}".format(Fore.WHITE, Fore.YELLOW + str(map_path) +
Style.RESET_ALL))

        if args.screen_capture:
            print(
                Fore.WHITE + " || Screenshots: " + Fore.YELLOW + os.getcwd() + output_path + "/screenshots" +
Style.RESET_ALL)
```

```
        # Scan each IP pointing to the same domain
        for ip in target_list[key].resolved_ips:
            print("")
            print(Fore.WHITE + " [-] IP Address: " + Style.BRIGHT + Fore.YELLOW + str(ip.address) +
Style.RESET_ALL)
            if ip.hostname:
                print(Fore.WHITE + "    || Hostname: " + Fore.YELLOW + ','.join(map(str, ip.hostname)) +
Style.RESET_ALL)
            if ip.server:
                print(Fore.WHITE + "    || Server: " + Fore.YELLOW + ip.server)
            if ip.ports:
                print(Fore.WHITE + "    || Ports: " + Fore.YELLOW + '/tcp, '.join(map(str, ip.ports)) + "/tcp" +
Style.RESET_ALL)
            if ip.vulns:
                print(Fore.WHITE + "    || Possible Vulnerabities: " + Fore.YELLOW + ','.join(map(str, ip.vulns)) +
Style.RESET_ALL)
            if ip.location:
                print(Fore.WHITE + "    || Location: " + Fore.YELLOW + ip.location + Style.RESET_ALL)
            print(Fore.WHITE + "    || ASN: " + Fore.YELLOW + ip.asn + Style.RESET_ALL)
            if ip.asn_name:
                print(Fore.WHITE + "    || ASN Name: " + Fore.YELLOW + str(ip.asn_name) + Style.RESET_ALL)
            print(Fore.WHITE + "    || CIDR: " + Fore.YELLOW + ip.cidr + Style.RESET_ALL)
            print("")
            if ip.ports:
                count.ports += len(ip.ports)
            if ip.vulns:
                count.vulns += len(ip.vulns)

        # Update Counters
        count.ips += len(target_list[key].resolved_ips)
        count.targets += 1
        count.subdomains += len(target_list[key].subdomains)
        count.employees += len(target_list[key].employees)
        count.emails += len(target_list[key].emails)
        count.guessed_emails += len(target_list[key].guessed_emails)
        count.creds += len(target_list[key].creds)
        count.hashes += len(target_list[key].hashes)
        count.buckets += len(target_list[key].buckets)

        store_results(target_list[key], output_path)


# Capture SIGINT
def sig_handler(signal, frame):
    cprint("info", "\n\n[i] Shutting down AttackSurfaceMapper. . .\n", 1)  # Success Msg
    try:
        signal.pause()
    except:
        pass
    cprint("info", "\n[i] Bye, bye!\n", 1)  # Success Msg
    sys.exit(0)

if __name__ == "__main__":
    signal.signal(signal.SIGINT, sig_handler)  # Signal Listener
```

```
    now = datetime.now()
    output_path = 'DNS_REC'
    sw1 = MasterSwitch()
    keychain = dict()
    c1 = Counter()
    start_time = time()  # Start Counter

    output_path = init_checks(sw1, output_path)  # Initialisation Checks
    keyloader(keychain, sw1)  # Key Loader

    cprint("info", "\n\n[i] Recconaissance is running. . .\n", 1)  # Success Msg

    if not os.path.exists(output_path):
        os.mkdir(output_path)

    main(keychain, sw1, output_path, c1)

    print_results(c1, start_time)  # Print terminal output

    exit()
```

## A.4.1 Buckethunter.py

```
#!/usr/bin/python3
#   Filename: buckethunter.py
#   Module: BucketHunter

# External Libraries
import colorama
import requests
from colorama import Fore, Style
from tld import get_tld


def cprint(type, msg, reset):
    colorama.init()
    message = {
        "action": Fore.YELLOW,
        "positive": Fore.GREEN + Style.BRIGHT,
        "info": Fore.YELLOW,
        "reset": Style.RESET_ALL,
        "red": Fore.RED,
        "white": Fore.WHITE,
        "green": Fore.GREEN,
        "yellow": Fore.YELLOW
    }
    style = message.get(type.lower())

    if type == "error":
```

```
        print("{0}\n[*] Error: {1}".format(Fore.RED + Style.BRIGHT, Style.RESET_ALL + Fore.WHITE + msg))
    else:
        print(style + msg, end="")
    if (reset == 1):
        print(Style.RESET_ALL)


def passive_query(mswitch,hostx, key):
    keywords = get_tld(hostx.primary_domain, as_object=True, fail_silently=True, fix_protocol=True).domain

    if keywords is None:
        return

    if mswitch is True:
        print("[DEBUG] Keywords : ",keywords)

    par = {'access_token': key, 'keywords': keywords}
    try:
        response = requests.get("https://buckets.grayhatwarfare.com/api/v1/buckets", params=par, timeout=4)
        gwf_api = response.json()
        if gwf_api["buckets_count"] > 0:
            try:
                for bucket in gwf_api["buckets"]:
                    hostx.buckets.append(bucket["bucket"])
            except:
                pass

    except:
        cprint("error", "[*] Error: connecting with GrayHatWarfare API", 1)


def active(mswitch, hostx, wordlist, recursive=False):
    pass
```

## A.4.2 Censys.py

```
#!/usr/bin/python3
#   Filename: censys.py
#   Module: Censys


# Standard Libraries
import time

# External Libraries
from censys.search import CensysHosts

__version__ = "v2.0"

def port_scan(hostx, censys_id, censys_secret, counter):
    c = CensysHosts(censys_id, censys_secret)
```

```
    for ip in hostx.resolved_ips:
        try:
            response = c.view(ip.address)
            ports = response.get("ports", [])
            if ip.ports:
                ip.ports.update(ports)
            else:
                ip.ports = ports
            counter.ports = counter.ports + len(hostx.ports)
        except:
            time.sleep(1)
            continue
```

### A.4.3 Dnsdumpster.py

```
#!/usr/bin/python3
#   Filename: dnsdumpster.py
#   Module: DNSdumpster

# Standard Libraries
import os

# External Libraries
import requests

DNS_DUMPSTER = 'https://dnsdumpster.com/'


# get_map Function - Downloads DNS Map from dnsdumpster.com
def get_map(hostx, out_path):
    headers = {'user-agent': "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0) Gecko/20100101 Firefox/47.0",
               'Referer': DNS_DUMPSTER}
    map_url = "{0}static/map/{1}.png".format(DNS_DUMPSTER, str(hostx.primary_domain))
    target_directory = out_path + "/" + hostx.primary_domain
    map_path = out_path + "/" + hostx.primary_domain + "/" + hostx.primary_domain + "_map" + ".png"

    if hostx.primary_domain == "":
        return None
    try:

        res1 = requests.get(DNS_DUMPSTER, headers=headers)
        csrftoken = res1.cookies.get('csrftoken')
        data = {
            'csrfmiddlewaretoken': csrftoken,
            'targetip': hostx.primary_domain
        }

        headers = {'user-agent': "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0) Gecko/20100101 Firefox/47.0",
                   'Referer': DNS_DUMPSTER, 'Cookie': 'csrftoken=' + csrftoken}
        res2 = requests.post(DNS_DUMPSTER, data=data, headers=headers)

        response = requests.get(map_url, headers=headers)
```

```
        # print("Response Status: " + response.status_code) # Debug Statement
        if response.status_code == 200:

            try:
                os.mkdir(target_directory)
            except:
                pass
            with open(map_path, 'wb') as new_file:
                new_file.write(response.content)
        return map_path
    except:
        return ""
```

## A.4.5 Hosthunter.py

```
#!/usr/bin/python3
#   Filename: hosthunter.py
#   Module: HostHunter

import socket
# Standard Libraries
import ssl

# External Libraries
import OpenSSL
import dns.resolver
import requests
import urllib3
from requests.packages.urllib3.exceptions import InsecureRequestWarning

requests.packages.urllib3.disable_warnings(InsecureRequestWarning)
context = ssl.create_default_context()
context.check_hostname = False
context.verify_mode = ssl.CERT_OPTIONAL
context.load_default_certs()


# ssl_grabber Function
def ssl_grabber(resolved_ip, port):
    try:
        cert = ssl.get_server_certificate((resolved_ip.address, port))
        x509 = OpenSSL.crypto.load_certificate(OpenSSL.crypto.FILETYPE_PEM, cert)
        cert_hostname = x509.get_subject().CN

        # Add New HostNames to List
        for host in cert_hostname.split('\n'):
            # print(host)
            if (host == "") or (host in resolved_ip.hostname):
                pass
            else:
                try:
                    resolved_ip.hostname.append(host)
```

```
            except:
                pass
    except (urllib3.exceptions.ReadTimeoutError, requests.ConnectionError, urllib3.connection.ConnectionError,
            urllib3.exceptions.MaxRetryError, urllib3.exceptions.ConnectTimeoutError,
urllib3.exceptions.TimeoutError,
            socket.error, socket.timeout) as e:
        pass


# r_dns Function
def r_dns(targetIP):
    try:
        hostname = socket.gethostbyaddr(targetIP.address)
        if hostname[0] != "":
            targetIP.hostname.append(hostname[0])
        # print("[Debug] r_dns result " + hostname[0]) ## Debug Statement
    except:
        pass


# query_api Function
def query_api(hostx):
    try:
        url = "https://api.hackertarget.com/reverseiplookup/?q="
        r2 = requests.get(url + hostx.resolved_ips[0].address, verify=False).text
        if (r2.find("No DNS A records found") == -1) and (
                r2.find("API count exceeded") == -1 and r2.find("error") == -1):
            for host in r2.split('\n'):
                if (host == "") or (host in hostx.hname):
                    pass
                else:
                    hostx.hname.append(host)
        # Add API count exceed detection
        else:
            pass
    except (
    requests.exceptions.ConnectionError, urllib3.connection.ConnectionError,
urllib3.exceptions.ConnectTimeoutError,
    urllib3.exceptions.MaxRetryError, urllib3.exceptions.TimeoutError, socket.error, socket.timeout):
        print("error", "query_api failed, connecting with HackerTarget.com API", 1)


def org_finder(hostx):
    target = hostx.primary_domain

    try:
        cert = ssl.get_server_certificate((target, 443))
        x509 = OpenSSL.crypto.load_certificate(OpenSSL.crypto.FILETYPE_PEM, cert)
        orgName = x509.get_subject().organizationName
        unit = x509.get_subject().organizationalUnitName
        hostx.orgName = str(orgName)
    except:
        pass
```

```
#  dnsloopkup Function
def dnslookup(hostx):
    try:
        url = "https://api.hackertarget.com/dnslookup/?q="
        r2 = requests.get(url + hostx.primary_domain, verify=False).text
        if (r2.find("No DNS A records found") == -1) and (
                r2.find("API count exceeded") == -1 and r2.find("error") == -1):
            hostx.dnsrecords = r2.splitlines()
            # Add API count exceed detection
            for record in hostx.dnsrecords:
                word = record.rsplit(':')
                try:
                    if ("TXT" in word[0]) and ("v=spf1" in word[1]):
                        hostx.spf = True
                except:
                    pass
            if hostx.spf is None:
                hostx.spf = False
        else:
            pass
    except (
    requests.exceptions.ConnectionError, urllib3.connection.ConnectionError,
urllib3.exceptions.ConnectTimeoutError,
    urllib3.exceptions.MaxRetryError, urllib3.exceptions.TimeoutError, socket.error, socket.timeout):
        print("error", " dnslookup failed, connecting with HackerTarget.com API", 1)


def dnsquery(hostx):
    try:
        response = dns.resolver.query('_dmarc' + '.' + hostx.primary_domain, 'TXT')

        for rdata in response:
            dmarc_record = str(rdata).replace(' ', '')
            if "v=DMARC" in dmarc_record:
                if ";p=reject;" in dmarc_record:
                    hostx.dmarc_status = "Primary Domain is not Spoofable. "
                elif ";p=quarantine;" in dmarc_record:
                    hostx.dmarc_status = "Primary Domain accepts spoofed emails but they will be marked as
suspicious. "
                elif ";p=none;" in dmarc_record:
                    hostx.dmarc_status = "Primary Domain allows Email Spoofing. "

                if ";sp=none;" in dmarc_record:
                    hostx.dmarc_status += "Subdomains allow Email Spoofing."
                elif ";sp=reject;" in dmarc_record:
                    hostx.dmarc_status += "SubDomains are not Spoofable."
                elif ";sp=quarantine;" in dmarc_record:
                    hostx.dmarc_status += "SubDomains will accept spoofed emails but they will be marked as
suspicious."
                else:
                    hostx.dmarc_status = "Spoofable Email Domain."

            hostx.dmarc.append(str(rdata))
    except:
```

```
        hostx.dmarc = []

    try:
        hostx.mx = dns.resolver.query(hostx.primary_domain, 'MX')
    except:
        pass


def active(hostx, count):
    for ip in hostx.resolved_ips:
        ssl_grabber(ip, "443")  # SSL
        ssl_grabber(ip, "993")  # IMAP - SSL
        ssl_grabber(ip, "22")  # FTPs - SSL

        count.hostnames += len(ip.hostname)  # Updates Counter
```

## A.4.6 Hunterio.py

```
#!/usr/bin/python3
#   Filename: hunterio.py
#   Module: HunterIO

# Standard Libraries
import json

# External Libraries
import requests


def query(hostx, key):
    try:
        domain = hostx.primary_domain
        par = {'domain': domain, 'api_key': key}
        user_agent = {'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0) Gecko/20100101
Firefox/47.0'}
        req = requests.get("https://api.hunter.io/v2/domain-search", params=par, headers=user_agent)
        hunterio_api = json.loads(req.text)

        try:
            for item in hunterio_api["data"]["emails"]:
                # print(item) # [Debug]: Prints Email Address
                hostx.emails.append(item["value"])
        except:
            pass

        try:
            hostx.pattern = hunterio_api["data"]["pattern"]
        except:
            pass

    except:
```

```
        pass
```

## A.4.7 Shodan.py

```
#!/usr/bin/python3
#   Filename: shodan.py
#   Module: Shodan

# Standard Libraries
import time

# External Libraries
import shodan


def port_scan(hostx, key, counter):
    api = shodan.Shodan(key)
    numports = 0

    for IP in hostx.resolved_ips:
        try:
            query = api.host(IP.address)
            IP.ports = query['ports']
            IP.vulns = query['vulns']
            IP.server = query['server']
            # print (query['vulnerabilities'])
            counter.ports = counter.ports + len(hostx.ports)
            counter.vulns = counter.vulns + len(hostx.vulns)
        except:
            time.sleep(1)
            continue
```

## A.4.8 Subbrute.py

```
#!/usr/bin/env python
#
# SubBrute v2.0
# A (very) fast subdomain spider.
#
import ctypes
import datetime
import itertools
import optparse
import os
import random
```

```
import re
import signal
import string
import sys
import time
import uuid

import dnslib

# Python 2.x and 3.x compatiablity
# We need the Queue library for exception handling
try:
    import queue as Queue
except:
    import Queue

# The 'multiprocessing' library does not rely upon a Global Interpreter Lock (GIL)
import multiprocessing

# Microsoft compatiablity
if sys.platform.startswith('win'):
    # Drop-in replacement,  subbrute + multiprocessing throws exceptions on windows.
    import threading

    multiprocessing.Process = threading.Thread

import json, colorama
from colorama import Fore, Style


def cprint(type, msg, reset):
    colorama.init()
    message = {
        "action": Fore.YELLOW,
        "positive": Fore.GREEN + Style.BRIGHT,
        "info": Fore.YELLOW,
        "reset": Style.RESET_ALL,
        "red": Fore.RED,
        "white": Fore.WHITE,
        "green": Fore.GREEN,
        "yellow": Fore.YELLOW
    }
    style = message.get(type.lower())


# A resolver wrapper around dnslib.py
class Resolver:
    # Google's DNS servers are only used if zero resolvers are specified by the user.
    pos = 0
    rcode = ""
    wildcards = {}
    failed_code = False
    last_resolver = ""
```

```
def __init__(self, nameservers=['8.8.8.8', '8.8.4.4']):
    self.nameservers = nameservers

def query(self, hostname, query_type='ANY', name_server=False, use_tcp=False):
    ret = []
    response = None
    if name_server == False:
        name_server = self.get_ns()
    else:
        self.wildcards = {}
        self.failed_code = None
    self.last_resolver = name_server
    query = dnslib.DNSRecord.question(hostname, query_type.upper().strip())
    try:
        response_q = query.send(name_server, 53, use_tcp, timeout=30)
        if response_q:
            response = dnslib.DNSRecord.parse(response_q)
        else:
            raise IOError("Empty Response")
    except Exception as e:
        # IOErrors are all conditions that require a retry.
        raise IOError(str(e))
    if response:
        self.rcode = dnslib.RCODE[response.header.rcode]
        for r in response.rr:
            try:
                rtype = str(dnslib.QTYPE[r.rtype])
            except:  # Server sent an unknown type:
                rtype = str(r.rtype)
            # Fully qualified domains may cause problems for other tools that use subbrute's output.
            rhost = str(r.rname).rstrip(".")
            ret.append((rhost, rtype, str(r.rdata)))
        # What kind of response did we get?
        if self.rcode not in ['NOERROR', 'NXDOMAIN', 'SERVFAIL', 'REFUSED']:
            trace('!Odd error code:', self.rcode, hostname, query_type)
        # Is this a perm error?  We will have to retry to find out.
        if self.rcode in ['SERVFAIL', 'REFUSED', 'FORMERR', 'NOTIMP', 'NOTAUTH']:
            raise IOError('DNS Failure: ' + hostname + " - " + self.rcode)
        # Did we get an empty body and a non-error code?
        elif not len(ret) and self.rcode != "NXDOMAIN":
            raise IOError("DNS Error - " + self.rcode + " - for:" + hostname)
    return ret

def was_successful(self):
    ret = False
    if self.failed_code and self.rcode != self.failed_code:
        ret = True
    elif self.rcode == 'NOERROR':
        ret = True
    return ret

def get_returncode(self):
    return self.rcode
```

```
  def get_ns(self):
    if self.pos >= len(self.nameservers):
      self.pos = 0
    ret = self.nameservers[self.pos]
    # we may have metadata on how this resolver fails
    try:
      ret, self.wildcards, self.failed_code = ret
    except:
      self.wildcards = {}
      self.failed_code = None
    self.pos += 1
    return ret

  def add_ns(self, resolver):
    if resolver:
      self.nameservers.append(resolver)

  def get_authoritative(self, hostname):
    ret = []
    while not ret and hostname.count(".") >= 1:
      try:
        trace("Looking for nameservers:", hostname)
        nameservers = self.query(hostname, 'NS', use_tcp=False)
      except IOError:  # lookup failed.
        nameservers = []
      for n in nameservers:
        # A DNS server could return anything.
        rhost, record_type, record = n
        if record_type == "NS":
          # Return all A records for this NS lookup.
          a_lookup = self.query(record.rstrip("."), 'A', use_tcp=False)
          for a_host, a_type, a_record in a_lookup:
            ret.append(a_record)
        # If a nameserver wasn't found try the parent of this sub.
      hostname = hostname[hostname.find(".") + 1:]
    return ret

  def get_last_resolver(self):
    return self.last_resolver


class VerifyNameservers(multiprocessing.Process):
  def __init__(self, target, query_type, resolver_q, resolver_list, authoritative=False):
    multiprocessing.Process.__init__(self, target=self.run)
    self.daemon = True
    signal_init()
    self.authoritative = authoritative

    self.start_time = 0
    self.resolver_q = resolver_q
    self.query_type = query_type
    self.resolver_list = resolver_list
    self.resolver = Resolver()
```

```
    # The domain provided by the user.
    self.target = target
    # Resolvers that will work in a pinch:
    self.backup_resolver = ['8.8.8.8', '8.8.4.4', '127.0.0.1']
    self.prev_wildcards = {}

# This process cannot block forever,  it  needs to check if its time to die.
def add_nameserver(self, nameserver):
    keep_trying = True
    while keep_trying:
        try:
            self.resolver_q.put(nameserver, timeout=1)
            trace("Added nameserver:", nameserver)
            keep_trying = False
        except Exception as e:
            if type(e) == Queue.Full or str(type(e)) == "<class 'queue.Full'>":
                keep_trying = True

def verify(self, nameserver_list):
    added_resolver = False
    for server in nameserver_list:
        server = server.strip()
        if server:
            try:
                # Only add the nameserver to the queue if we can detect wildcards.
                verified_server = self.find_wildcards(self.target, server)
                if verified_server:
                    # wildcards have been added to the set, it is now safe to be added to the queue.
                    # blocking queue,  this process will halt on put() when the queue is full:
                    self.add_nameserver(verified_server)
                    added_resolver = True
                else:
                    trace("Rejected nameserver - wildcard:", server)
            except Exception as e:
                # Rejected server :(
                trace("Rejected nameserver - unreliable:", server, type(e))
    return added_resolver

def run(self):
    # Every user will get a different set of resovlers, this helps redistribute traffic.
    random.shuffle(self.resolver_list)
    if not self.verify(self.resolver_list):
        # This should never happen,  inform the user.
        sys.stderr.write('Warning: No nameservers found, trying fallback list.\n')
        # Try and fix it for the user:
        self.verify(self.backup_resolver)
    # End of the resolvers list.
    try:
        self.resolver_q.put(False, timeout=1)
    except:
        pass

# Only add the nameserver to the queue if we can detect wildcards.
```

```
    # Returns False on error.
    def find_wildcards(self, host, server):
        wildcards = {}
        resolver_fail_code = False
        # We want sovle the following three problems:
        # 1)The target might have a wildcard DNS record.
        # 2)The target maybe using geolocaiton-aware DNS.
        # I have seen a CloudFlare Enterprise customer with these two conditions.
        try:
            # start_time means this thread isn't dead
            self.start_time = datetime.datetime.now()
            # make sure we can query the host
            blanktest = self.resolver.query(self.target, self.query_type, server)
            if self.query_type == "ANY":
                # If the type was ANY we should have gotten some records
                if not len(blanktest) and not self.authoritative:
                    return False
            elif not self.resolver.was_successful():
                trace("Cannot perform ", self.query_type, " request:", host)
                return False
        except Exception:
            if not self.authoritative:
                trace("Cannot perform ", self.query_type, " request:", host)
                return False
        start_counter = 128
        test_counter = start_counter
        looking_for_wildcards = True
        while looking_for_wildcards and test_counter >= 0:
            looking_for_wildcards = False
            # Don't get lost, this nameserver could be playing tricks.
            test_counter -= 1
            try:
                # Using a 32 char string every time may be too predictable.
                x = uuid.uuid4().hex[0:random.randint(6, 32)]
                testdomain = "%s.%s" % (x, host)
                self.start_time = datetime.datetime.now()  # I'm not dead yet!
                wildtest = self.resolver.query(testdomain, self.query_type, server)
                # This record may contain a list of wildcards.
                if len(wildtest):
                    for w in wildtest:
                        return_name, record_type, data = w
                        if record_type in ["CNAME", "A", "AAAA", "MX"]:
                            data = str(data)
                            # All authoritative NS for the same hsot *should* have the same wildcards
                            if self.prev_wildcards:
                                # Have we need this wildcard before?
                                if data in self.prev_wildcards:
                                    # We have seen this wildcards before.
                                    # We do an update, because we may have found a new wildcard
                                    # specific to the NS server we are testing.
                                    wildcards.update(self.prev_wildcards)
                                    # Look for afew more wildcards, and then return.
                                    if test_counter > 2:
```

```
                                test_counter = 2
                        if data not in wildcards:
                            # wildcards were detected.
                            wildcards[data] = None
                            # found atleast one wildcard, look for more.
                            looking_for_wildcards = True
                    # If we keep getting wildcards, keep looking (N * 8) + 8 times,
                    # where N is the total number of wildcards found.
                    # Test case: airbnb.com
                    if test_counter >= start_counter - len(wildcards) * 8 - 8:
                        looking_for_wildcards = True
                except Exception as e:
                    # This resolver maybe flakey, we don't want it for our tests.
                    if not self.authoritative:
                        trace("wildcard exception:", server, type(e))
                        return False
                    else:
                        # The authoritative server isn't going to give us wildcards
                        looking_for_wildcards = False
                finally:
                    # We always need the return code, it can be None
                    resolver_fail_code = self.resolver.get_returncode()
            # If we hit the end of our depth counter and,
            # there are still wildcards, then reject this nameserver because it smells bad.
            if test_counter >= 0 or self.authoritative:
                self.prev_wildcards = wildcards
                return (server, wildcards, resolver_fail_code)
            else:
                return False


class Lookup(multiprocessing.Process):
    def __init__(self, in_q, in_q_priority, out_q, resolver_q, domain):
        multiprocessing.Process.__init__(self, target=self.run)
        signal_init()
        self.required_nameservers = 16
        self.in_q = in_q
        self.in_q_priority = in_q_priority
        self.out_q = out_q
        self.resolver_q = resolver_q
        self.domain = domain
        # Passing an empty array forces the resolver object to use our nameservers
        self.resolver = Resolver([])
        self.start_time = 0
        self.current_work = None

    def get_ns(self):
        ret = False
        try:
            ret = self.resolver_q.get_nowait()
            if ret == False:
                # Queue is empty,  inform the rest.
                self.resolver_q.put(False)
        except:
```

```
            pass
        return ret

    def get_ns_blocking(self):
        ret = False
        ret = self.resolver_q.get()
        if ret == False:
            trace("get_ns_blocking - Resolver list is empty.")
            # Queue is empty,  inform the rest.
            self.resolver_q.put(False)
            ret = []
        return ret

    def check(self, host, record_type="ANY", total_rechecks=0):
        trace("Checking:", host)
        cname_record = []
        retries = 0
        if len(self.resolver.nameservers) <= self.required_nameservers:
            # This process needs more nameservers,  lets see if we have one available
            self.resolver.add_ns(self.get_ns())
        # Ok we should be good to go.
        while True:
            try:
                # Query the nameserver, this is not simple...
                if not record_type or record_type == "ANY":
                    resp = self.resolver.query(host)
                    # A DNS record may exist without data. Usually this is a parent domain.
                    if self.resolver.was_successful() and not resp:
                        resp = [(host, self.resolver.get_returncode(), "")]
                    return resp
                if record_type == "CNAME":
                    added_cname = False
                    # A max 20 lookups
                    cname_host = host
                    resp = self.resolver.query(cname_host, "A", total_rechecks)
                    if not resp:
                        resp = self.resolver.query(cname_host, "AAAA", total_rechecks)
                    if not resp:
                        resp = self.resolver.query(cname_host, "CNAME", total_rechecks)
                    for r in resp:
                        return_name, record_type, record_data = r
                        # if record_type in ["CNAME", "A", "AAAA"]:
                        cname_host = str(record_data).rstrip(".")
                        cname_record.append(cname_host)
                    if not added_cname:
                        break
                    if cname_record:
                        ret = [(host, record_type, cname_record)]
                    else:
                        ret = False
                        # No response?  then return what we have.
                    return ret
                else:
```

```
                # All other records:
                return self.resolver.query(host, record_type)
            except (IOError, TypeError) as e:
                if total_rechecks >= 2 or \
                        (retries >= 1 and self.resolver.get_returncode() == "NOERROR"):
                    # Multiple threads have tried and given up
                    trace('Giving up:', host, self.resolver.get_returncode())
                    return [(host, self.resolver.get_returncode(), "")]
                elif retries >= 2:
                    # This thread has tried and given    up
                    trace('Exception:', type(e), " - ", e)
                    self.in_q_priority.put((host, record_type, total_rechecks + 1))
                    return False
                else:
                    # Retry the same request on the same thread.
                    time.sleep(retries)
                    # Give the DNS server a chance to cool off,  there maybe a rate-limit.
                    retries += 1

    def get_work(self):
        work = False
        # Check the priority queue first,  these results are more likely to have data.
        try:
            work = self.in_q_priority.get_nowait()
        except:
            work = False
        # the priority queue is empty, check the normal queue
        if not work:
            work = self.in_q.get()
        # Is this the end all work that needs to be done?
        if not work:
            trace('End of work queue')
            # Perpetuate the end marker for all threads to see
            self.in_q.put(False)
            # Notify the parent that we have died of natural causes
            self.out_q.put(False)
        return work

    def run(self):
        # This process needs one resolver before it can start looking.
        self.resolver.add_ns(self.get_ns_blocking())
        work = True
        while work:
            response = None
            work = self.get_work()
            # if the code above found work
            if work:
                # Keep track of what we are working on
                self.current_work = work
                self.start_time = datetime.datetime.now()
                # keep track of how many times this lookup has timedout.
                (hostname, query_type, timeout_retries) = work
                response = self.check(hostname, query_type, timeout_retries)
```

```
            sys.stdout.flush()
            # This variable doesn't need a muetex, because it has a queue.
            # A queue ensure nameserver cannot be used before it's wildcard entries are found.
            reject = False
            found = []
            if response:
                trace(response)
                for record in response:
                    return_name, record_type, data = record
                    data = str(data)
                    if len(data) and len(record_type) and not len(return_name):
                        # The server we are dealing with is a monster.
                        return_name = hostname
                    if data in self.resolver.wildcards:
                        trace("resovled wildcard:", hostname)
                        reject = True
                        # reject this domain.
                        break
                    else:
                        found.append(record)
                if not reject:
                    for f in found:
                        # This request is filled, send the results back
                        self.out_q.put(f)


# The multiprocessing queue will fill up, so a new process is required.
class Loader(multiprocessing.Process):
    def __init__(self, in_q, subdomains, query_type, permute_len=0):
        multiprocessing.Process.__init__(self, target=self.run)
        signal_init()
        self.in_q = in_q
        self.subdomains = subdomains
        self.query_type = query_type
        self.permute_len = permute_len

    # Python blocks on in_q for large datasets, even though the queue size is 'unlimited' :(
    def run(self):
        self.permute()
        # Remove items from the list that will be in the permutation set.
        permute_filter = re.compile("^[a-zA-Z0-9]{" + str(self.permute_len) + "}\.")
        # A list of subdomains is the input
        for s in self.subdomains:
            if not permute_filter.match(s):
                # Domains cannot contain whitespace,  and are case-insensitive.
                self.in_q.put((s, self.query_type, 0))
        # Terminate the queue
        self.in_q.put(False)

    # bruteforce a range.
    def permute(self):
        full_range = string.ascii_lowercase + string.digits + "_-"
        for l in range(1, self.permute_len + 1):
            for i in itertools.permutations(full_range, l):
```

```
            if i:
                self.in_q.put((i, self.query_type, 0))


# Extract relevant hosts
# The dot at the end of a domain signifies the root,
# and all TLDs are subs of the root.
host_match = re.compile(r"((?<=[^a-zA-Z0-9_-])[a-zA-Z0-9_-]+\.(?:[a-zA-Z0-9_-]+\.?)+(?=[^a-zA-Z0-9_-]))")


def extract_hosts(data, hostname=""):
    # made a global to avoid re-compilation
    global host_match
    ret = []
    hosts = re.findall(host_match, " " + data)
    for fh in hosts:
        host = fh.rstrip(".")
        # Is this host in scope?
        if host.endswith(hostname) and host != hostname:
            ret.append(host)
    return ret


# Return a unique list of subdomains to a given host
def extract_directory(dir_name, hostname=""):
    ret = []
    dupe = {}
    for root, subdirs, files in os.walk(dir_name):
        for filename in files:
            full_path = os.path.join(root, filename)
            raw = open(full_path).read()
            for h in extract_hosts(raw, hostname):
                if h not in dupe:
                    dupe[h] = None
                    ret.append(h)
    return ret


def print_target(target, query_type="ANY", subdomains="names.txt", resolve_list="resolvers.txt",
process_count=16,
                 print_data=False, output=False, json_output=False):
    json_struct = {}
    if not print_data:
        dupe_filter = {}
    for result in run(target, query_type, subdomains, resolve_list, process_count):
        (hostname, record_type, record) = result
        if not print_data:
            # We just care about new names, filter multiple records for the same name.
            if hostname not in dupe_filter:
                dupe_filter[hostname] = None
                result = hostname
            else:
                result = False
        else:
```

```
            if type(record) is type([]):
                record = ",".join(record)
            result = "%s,%s,%s" % (hostname, record_type, record)
        if result:
            print(result)
            sys.stdout.flush()
            if hostname in json_struct:
                if record_type in json_struct:
                    json_struct[hostname][record_type].append(record)
                else:
                    json_struct[hostname][record_type] = []
                    json_struct[hostname][record_type].append(record)
            else:
                json_struct[hostname] = {}
                json_struct[hostname][record_type] = []
                json_struct[hostname][record_type].append(record)
            if output:
                output.write(result + "\n")
                output.flush()
    # The below formats the JSON to be semantically correct, after the scan has been completed
    if json_output:
        json_output = open(options.json, "w")
        json_output.write(json.dumps(json_struct))


def run(target, query_type="ANY", subdomains="names.txt", resolve_list=False, process_count=15):
    spider_blacklist = {}
    result_blacklist = {}
    found_domains = {}
    # A thread fills the in_q, reduce memory usage, wait until we have space
    in_q = multiprocessing.Queue()
    in_q_priority = multiprocessing.Queue()
    out_q = multiprocessing.Queue()
    # Have a buffer of at most two new nameservers that lookup processes can draw from.
    resolve_q = multiprocessing.Queue(maxsize=2)

    if os.path.isdir(subdomains):
        subdomains = extract_directory(subdomains, target)
    else:
        subdomains = check_open(subdomains)

    is_authoritative = False
    if resolve_list:
        resolve_list = check_open(resolve_list)
        if (len(resolve_list) / 16) < process_count:
            sys.stderr.write(
                'Warning: Fewer than 16 resolvers per process, consider adding more nameservers to resolvers.txt.\n')
    else:
        # By default, use the authoritative nameservers for the target
        resolve = Resolver()
        resolve_list = resolve.get_authoritative(target)
        is_authoritative = True
        if not resolve_list:
            sys.stderr.write("Unable to find authoritative resolvers for:" + target)
```

```
        return

# If we are resolving against the authoratative NS, check AXFR,  we might get lucky :)
if is_authoritative:
    ar = Resolver(resolve_list)
    # Check every authoritative NS for AXFR support
    # These are distinct servers, one could be misconfigured
    for i in range(len(resolve_list)):
        res = []
        try:
            res = ar.query(target, 'AXFR')
        except:
            pass
        if res:
            trace("AXFR Successful for:", ar.get_last_resolver())
            for r in res:
                result_blacklist[str(r)] = None
                yield r
        # Even if the AXFR was a success, keep looking. Don't trust anyone.
# Make a source of fast nameservers available for other processes.
verify_nameservers_proc = VerifyNameservers(target, query_type, resolve_q, resolve_list, is_authoritative)
verify_nameservers_proc.start()
# test the empty string
in_q.put((target, query_type, 0))
spider_blacklist[target + query_type] = None
clean_subs = []
for s in subdomains:
    s = str(s).strip().lower()
    find_csv = s.find(",")
    if find_csv > 1:
        # SubBrute should be forgiving, a comma will never be in a hostname
        # but the user might try an use a CSV file as input.
        s = s[0:find_csv]
    s = s.rstrip(".")
    if s:
        # A subbrute.py -o output.csv maybe our input.
        if not s.endswith(target):
            hostname = "%s.%s" % (s, target)
        else:
            # A user might feed an output list as a subdomain list.
            hostname = s
        spider_lookup = hostname + query_type
        if spider_lookup not in spider_blacklist:
            spider_blacklist[spider_lookup] = None
            clean_subs.append(hostname)

# Free up some memory before the big show.
del subdomains

# load in the subdomains, can be quite large
load = Loader(in_q, clean_subs, query_type)
load.start()
```

```
# We may not have the resolvers needed to backup our thread count.
list_len = len(resolve_list)
if list_len < process_count:
    # // is floor division.  always return a full number.
    # We need a minimum of 2 resolvers per thread to hold by the 1 query per 5 sec limit.
    process_count = list_len // 2
    if process_count <= 0:
        process_count = 1
    trace("Too few resolvers:", list_len, " process_count reduced to:", process_count)
worker_list = []
for i in range(process_count):
    worker = Lookup(in_q, in_q_priority, out_q, resolve_q, target)
    worker.start()
    worker_list.append(worker)
threads_remaining = process_count
while True:
    try:
        # The output is valid hostnames
        result = out_q.get(True, 10)
        # we will get an empty exception before this runs.
        if not result:
            threads_remaining -= 1
        else:
            s_result = str(result)
            if s_result not in result_blacklist:
                result_blacklist[s_result] = None
                record_name, record_type, record_data = result
                # Does this DNS record contain a useful subdomain?
                # If the query_type is CNAME, then lookup() takes care of the CNAME record chain.
                if query_type != "CNAME" and record_type not in ["AAAA", "A"]:
                    # did a record contain a new host?
                    hosts = extract_hosts(str(record_data), target)
                    for h in hosts:
                        spider_lookup = h + query_type
                        if spider_lookup not in spider_blacklist:
                            spider_blacklist[spider_lookup] = None
                            # spider newly found hostname
                            in_q_priority.put((h, query_type, 0))
                if type(record_name) is tuple:
                    pass
                # If we are using open resolvers we need to attempt every record type.
                if query_type == "ANY" and record_name.endswith(target):
                    # Simulate an ANY query by requesting ALL types
                    for qt in dnslib.QTYPE.reverse:
                        # These query types are usually disabled and are not typically enabled on a per-sub basis.
                        if qt not in ["AXFR", "IXFR", "OPT", "TSIG", "TKEY"]:
                            spider_lookup = record_name + qt
                            if spider_lookup not in spider_blacklist:
                                spider_blacklist[spider_lookup] = None
                                # This will produce many NOERROR retries, reduce the retries.
                                in_q_priority.put((record_name, qt, 2))
                # if this is an error response, check if we have already found data for this domain.
                if not record_data:
```

```
                    if not record_name in found_domains:
                        found_domains[record_name] = None
                        yield result
                else:
                    found_domains[record_name] = None
                    yield result
                    # run() is a generator, and yields results from the work queue
        except Exception as e:
            # The cx_freeze version uses queue.Empty instead of Queue.Empty :(
            if type(e) == Queue.Empty or str(type(e)) == "<class 'queue.Empty'>":
                pass
            else:
                raise (e)
        # make sure everyone is complete
        if threads_remaining <= 0:
            break
    trace("About to kill nameserver process...")
    # We no longer require name servers.
    try:
        killproc(pid=verify_nameservers_proc.pid)
    except:
        # Windows threading.tread
        verify_nameservers_proc.end()
    trace("End")


# exit handler for signals.  So ctrl+c will work.
# The 'multiprocessing' library each process is it's own process which side-steps the GIL
# If the user wants to exit prematurely,  each process must be killed.
def killproc(signum=0, frame=0, pid=False):
    if not pid:
        pid = os.getpid()
    if sys.platform.startswith('win'):
        try:
            kernel32 = ctypes.windll.kernel32
            handle = kernel32.OpenProcess(1, 0, pid)
            kernel32.TerminateProcess(handle, 0)
        except:
            # Oah windows, the above code *may* throw an exception and still succeed :/
            pass
    else:
        os.kill(pid, 9)


# Toggle debug output
verbose = False


def trace(*args, **kwargs):
    if verbose:
        for a in args:
            sys.stderr.write(str(a))
            sys.stderr.write(" ")
        sys.stderr.write("\n")
```

```
# display error message, and then quit
def error(*args, **kwargs):
    for a in args:
        sys.stderr.write(str(a))
        sys.stderr.write(" ")
    sys.stderr.write("\n")
    sys.exit(1)


def check_open(input_file):
    ret = []
    # If we can't find a resolver from an input file, then we need to improvise.
    try:
        lines = open(input_file).readlines()
        # Check if this is CSV,  if it is, then use the first column.
        for l in lines:
            find_csv = l.find(",")
            if find_csv:
                ret.append(l[0:find_csv])
            else:
                ret.append(l)
    except:
        cprint("error", "File not found: " + str(input_file), 1)
        return ret

    if not len(ret):
        cprint("error", "File is empty:" + str(input_file), 1)
    return ret


# Every 'multiprocessing' process needs a signal handler.
# All processes need to die, we don't want to leave zombies.
def signal_init():
    # killproc() escalates the signal to prevent zombies.
    signal.signal(signal.SIGINT, killproc)
    try:
        # These hanlders don't exist on every platform.
        signal.signal(signal.SIGTSTP, killproc)
        signal.signal(signal.SIGQUIT, killproc)
    except:
        # Windows
        pass


if __name__ == "__main__":
    if getattr(sys, 'frozen', False):
        # cx_freeze windows:
        base_path = os.path.dirname(sys.executable)
        multiprocessing.freeze_support()
    else:
        # everything else:
        base_path = os.path.dirname(os.path.realpath(__file__))
```

```
    parser = optparse.OptionParser("\n%prog [options] target_domain\n%prog -p target_domain")
    parser.add_option("-s", "--subs", dest="subs", default=os.path.join(base_path, "names.txt"),
                      type="string",
                      help="(optional) A list of subdomains, accepts a single file, or a directory of files. default =
'names.txt'")
    parser.add_option("-r", "--resolvers", dest="resolvers", default="./resources/resolvers.txt",
                      type="string",
                      help="(optional) A list of DNS resolvers, if this list is empty it will OS's internal resolver default =
'resolvers.txt'")
    parser.add_option("-t", "--targets_file", dest="targets", default="",
                      type="string",
                      help="(optional) A file containing a newline delimited list of domains to brute force.")
    parser.add_option("-p", "-P", action='store_true', dest="print_data", default=False,
                      help="(optional) Print data from found DNS records (default = off).")
    parser.add_option("-o", "--output", dest="output", default=False,
                      help="(optional) Output to file (Greppable Format)")
    parser.add_option("-j", "--json", dest="json", default=False, help="(optional) Output to file (JSON Format)")
    parser.add_option("--type", dest="type", default=False,
                      type="string",
                      help="(optional) Print all reponses for an arbitrary DNS record type (CNAME, AAAA, TXT, SOA,
MX...)")
    parser.add_option("-c", "--process_count", dest="process_count",
                      default=8, type="int",
                      help="(optional) Number of lookup theads to run. default = 8")
    parser.add_option("-v", "--verbose", action='store_true', dest="verbose", default=False,
                      help="(optional) Print debug information.")
    (options, args) = parser.parse_args()

    verbose = options.verbose

    if len(args) < 1 and options.targets == "":
        parser.error("You must provide a target. Use -h for help.")

    if options.targets != "":
        targets = check_open(options.targets)
    else:
        targets = args  # multiple arguments on the cli:  ./subbrute.py google.com gmail.com yahoo.com

    output = False
    if options.output:
        try:
            output = open(options.output, "w")
        except:
            error("Failed writing to file:", options.output)

    json_output = False
    if options.json:
        try:
            json_output = open(options.json, "w")
        except:
            error("Failed writing to file:", options.json)

    # subbrute with find the best record to use if the type is None.
```

```
    record_type = "ANY"
    if options.type:
        record_type = str(options.type).upper()

    threads = []
    for target in targets:
        target = target.strip()
        if target:
            trace("dnslib:", dnslib.version)
            trace(target, record_type, options.subs, options.resolvers, options.process_count, options.print_data,
                  output, json_output)
            print_target(target, record_type, options.subs, options.resolvers, options.process_count,
                         options.print_data, output, json_output)
```

## A.4.9 Subhunter.py

```
#!/usr/bin/python3
#   Filename: subhunter.py
#   Module: SubHunter

# Standard Libraries
import time
import colorama
import ipaddress
# External Libraries
import requests
from colorama import Fore, Style
from validator_collection import checkers

from modules import subbrute
from subprocess import STDOUT, check_output

class TargetIP:
    def __init__(self, addr):
        self.address = addr
        self.hostname = []
        self.ports = []
        self.asn = ""
        self.asn_name = ""
        self.server = ""
        self.vulns = []
        self.cidr = ""
        self.location = ""
        self.country = ""

def cprint(type, msg, reset):
    colorama.init()
    message = {
        "action": Fore.YELLOW,
        "positive": Fore.GREEN + Style.BRIGHT,
        "info": Fore.YELLOW,
```

```
        "reset": Style.RESET_ALL,
        "red": Fore.RED,
        "white": Fore.WHITE,
        "green": Fore.GREEN,
        "yellow": Fore.YELLOW
    }
    style = message.get(type.lower())

    if type == "error":
        print("{0}\n[*] Error: {1}".format(Fore.RED + Style.BRIGHT, Style.RESET_ALL + Fore.WHITE + msg))
    else:
        print(style + msg, end="")
    if (reset == 1):
        print(Style.RESET_ALL)


def passive_query(hostx, key):
    par = {'apikey': key, 'domain': hostx.primary_domain}
    try:
        response = requests.get("https://www.virustotal.com/vtapi/v2/domain/report", params=par, timeout=4)
        tv_api = response.json()

        try:
            for sibling in tv_api["domain_siblings"]:
                if (sibling in hostx.subdomains) or (sibling == ""):
                    pass
                else:
                    hostx.subdomains.append(sibling)
        except:
            pass

        try:
            for subdomain in tv_api["subdomains"]:
                if (subdomain in hostx.subdomains) or (subdomain == ""):
                    pass
                else:
                    hostx.subdomains.append(subdomain)
        except:
            pass
    except:
        cprint("error", "[*] Error: connecting with VirusTotal API", 1)


def active(mswitch, hostx, wordlist, subwordlist, recursive=False):
    results=subbrute.run(hostx.primary_domain, subdomains=wordlist)
    for d in results:
        if mswitch.debug is True:
            current_time = time.strftime("%H:%M:%S", time.localtime())
            print("[DEBUG]["+str(current_time)+"][Item Discovered]: ",d)
        added_ips = []
        if (d[0] in hostx.subdomains) or (d[0] is hostx.primary_domain) or ("REFUSED" in d[1]) or ("NOERROR" in
d[1]) or ("NXDOMAIN" in d[1]) or ("HINFO" in d[1]):
            pass
        else:
```

```
            # Verbose Mode
            if mswitch.verbose is True:
                print(d[0] + "," + d[1] + "," + d[2])
            hostx.subdomains.append(d[0])
            if d[1] == "A" or d[1] == "MX":
                if checkers.is_ipv4(d[2]) and (ipaddress.ip_address(d[2]) is False):
                    tmp = TargetIP(d[2])
                    tmp.hostname.append(d[0])
                    if not d[2] in added_ips:
                        hostx.resolved_ips.append(tmp)
                        cprint("white", "    |", 1)
                        cprint("white", "  [{0}]".format(d[2]), 1)
                        if mswitch.verbose is True:
                            cprint("info", "[i] Adding target IPv4:" + d[2], 1)
    return True

    if recursive is True:
        for sub in hostx.subdomains:
            cprint("info", "[i] Enumerating: xxx." + sub, 1)
            for item in subbrute.run(sub, query_type="A", subdomains=subwordlist, process_count=60):
                if item[0] in hostx.subdomains or ("REFUSED" in item[1]) or ("NOERROR" in item[1]) or ("NXDOMAIN"
in d[1]) or ("HINFO" in item[1]):
                    pass
                else:
                    # Verbose Mode
                    if mswitch.verbose is True:
                        print(item[0] + "," + item[1] + "," + item[2])
                    hostx.subdomains.append(item[0])
                    # print (d[0]) # [Debug] Prints succesfully resolved domain
```

## A.4.10 Urlscanio.py

```
#!/usr/bin/python3
#   Filename: urlscanio.py
#   Module: URLScanIO Module

# Standard Libraries
import json

# External Libraries
import requests

import asm


def get_domain(IP):
    # print (ip.address)
    # $ curl -v --url "https://urlscan.io/api/v1/search?ip=<IP>"
    user_agent = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/74.0.3729.169 Safari/537.36'}
```

```
    try:
        response = requests.get('https://urlscan.io/api/v1/search?q=ip:' + IP, headers=user_agent)
        api = json.loads(response.text)
        if api['total'] == 0:
            return
        try:
            return api['results'][0]['page']['domain']
        except:
            pass
    except:
        pass


def query(hostx):
    # print(hostx.address)
    # curl -v --url "https://urlscan.io/api/v1/search?ip=148.251.165.186"
    for ip in hostx.resolved_ips:
        par = {'q': ip.address}
        # print (ip.address)
        user_agent = {
            'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/74.0.3729.169 Safari/537.36'}
        try:
            response = requests.get("https://urlscan.io/api/v1/search", params=par, headers=user_agent)
            # print(response.status_code)
            # print(response.text)
            api = json.loads(response.text)
            if api['total'] == 0:
                return
            try:
                ip.location = api['results'][0]['page']['city'] + ", " + api['results'][0]['page']['country']
            except:
                pass

            try:
                ip.asn = api['results'][0]['page']['asn']
            except:
                pass

            try:
                ip.asn_name = api['results'][0]['page']['asnname']
            except:
                pass

            if ip.server == "":
                try:
                    ip.server = api['results'][0]['page']['server']
                except:
                    pass

            try:
                for item in api['results']:
                    if (item['page']['domain'] in hostx.hname) or (item['page']['domain'] == ""):
                        pass
```

```
                else:
                    hostx.hname.append(item['page']['domain'])
            except:
                pass

        except:
            asm.cprint("error", "[*] Error: connecting with URLScanIO.com API", 1)
    return
```

## A.4.11 Webscraper.py

```
#!/usr/bin/python3
#   Filename: webscraper.py
#   Module: webscraper

# Standard Libraries
import re

# External Libraries
import requests
from bs4 import BeautifulSoup


def extract_info(hostx):
    print("webscraper Enabled")
    url = "https://" + hostx.primary_domain
    user_agent = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/74.0.3729.169 Safari/537.36'}
    try:
        response = requests.get(url, headers=user_agent)

        html_response = BeautifulSoup(response.text, 'html.parser')

        r = session.get(url)
        links = r.html.absolute_links
        print(links)
        # print (html_response)
        # print(html_response)

        phone = re.findall("((?:\d{3}|\(\d{3}\))?(?:\s|-|\.)?\d{3}(?:\s|-|\.)\d{4})", html_response.text)
        emails = re.findall(">[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,3}<", html_response.text)

        print("")
        # Emails
        print("")
        print("Emails Found:" + emails)

    except:
        pass
```

## A.4.12 Weleakinfo.py

```
#!/usr/bin/python3
#   Filename: weleakinfo.py
#   Module: WeLeakInfo Module

# Standard Libraries
import json
import time
import traceback

# External Libraries
import requests

import asm


def query(hostx, api_key, priv_key):
    head = {
        'user-agent': "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0) Gecko/20100101 Firefox/47.0",
        'authorization': "Bearer" + " " + api_key
    }

    if not hostx.emails:
        return 0

    ## Free API
    for email in hostx.emails:
        try:
            url = "https://api.weleakinfo.com/v3/public/email/" + email
            response = requests.get(url, headers=head)

            if not (response.status_code == 200):
                return -1

            api = json.loads(response.text)

            if api['Total'] == 0:
                return 0

            try:
                for item in api['Data']:
                    result = result + "," + item

                hostx.breaches[email] = result.replace(',', '', 1)
            except:
                pass

            # print (breaches.items())
```

```
        except:
            asm.cprint("error", "[*] Error: connecting with WeLeakInfo API", 1)

        time.sleep(2)

    return


def priv_api(hostx, api_key, priv_key):
    headers = {
        'User-agent': "Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:47.0) Gecko/20100101 Firefox/47.0",
        "Content-Type": "application/x-www-form-urlencoded",
        'authorization': "Bearer " + priv_key
    }

    try:
        api_endpoint = "https://api.weleakinfo.com/v3/search"
        query = {
            "limit": "1000",
            "offset": "",
            "query": "*" + "@" + hostx.primary_domain,
            "regex": "",
            "type": "email",
            "wildcard": "true"
        }

        response = requests.request("POST", api_endpoint, data=query, headers=headers)

        if response.status_code is not 200:
            return -1

        api = json.loads(response.text)

        if api['Total'] == 0:
            return 0

        try:
            for result in api['Data']:
                try:
                    # print(result['Password'])
                    # print(result['Email'])
                    hostx.creds.append(result['Email'] + "::" + result['Password'])

                    if result['Email'] not in hostx.emails:
                        hostx.emails.append(result['Email'])
                except:
                    pass

                try:
                    # print(result['Hash'])
                    # print(result['Email'])
                    hostx.hashes.append(result['Email'] + "::" + result['Hash'])

                    if result['Email'] not in hostx.emails:
```

```
                hostx.emails.append(result['Email'])
            except:
                pass
            # hostx.breaches[email] = result
        except:
            traceback.print_exc()
            pass

    except:
        return
```

## A.4.13 Whois_collector.py

```
#!/usr/bin/python3
#   Filename: whois_collector.py
#   Module: Whois Collector

# External Libraries
from ipwhois import IPWhois


def wlookup(hostx):
    for ip in hostx.resolved_ips:
        try:
            ip_object = IPWhois(ip.address)
            query = ip_object.lookup_rdap(depth=1)
            #           hostx.whois.append(query)
            #           net_sec = query.get('network', {})
            ip.location = query.get('asn_country_code')
            ip.asn = query.get('asn')
            ip.cidr = query.get('asn_cidr')
        #           print("Executed")
        #           print(query['network']['name'])
        except:
            pass
```